## Title

Machine learning methods for the detection of complex patterns in biological data



## Author

Xiong, Jingwei

## Publication Date

2023

Peer reviewed|Thesis/dissertation

# Machine Learning Methods For The Detection Of Complex Patterns In Biological Data

By

Jingwei Xiong

Dissertation

Submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Biostatistics

in the

Office of Graduate Studies

of the

University of California

Davis

Approved:

---

Javier Arsuaga, Chair

---

Wenqing Luo

---

Mariel Vazquez

Committee in Charge

2023

*This work is wholeheartedly dedicated to my family, the bedrock of my life and the keepers of my heart. Special homage is paid to my grandparents, Kangsheng Xiong and Cuilan Gao, whose care and guidance have been the nurturing forces of my upbringing. To my mother, Yan He, and my father, Yan Xiong, your enduring love and support have been my pillars of strength. Each of you has instilled in me the values and resilience that have been instrumental in my pursuit of knowledge and the completion of this work. Your love and sacrifices have not only fostered my growth but have also been the guiding light that led me to this very moment.*

# Contents

List of Figures . . . . . vii
List of Tables . . . . . xviii
Abstract . . . . . xxi
Acknowledgments . . . . . xxiii

**1 Introduction** **1**

1.1 Thesis overview . . . . . 1
1.2 Deep learning background . . . . . 2
1.2.1 Historical Background of Deep Learning . . . . . 2
1.2.2 The concept of Multilayer Perceptron (MLP) and Fully Connected (FC) layers . . . . . 4
1.2.3 Deep Learning Model Training . . . . . 7
1.2.4 Convolution Neural Networks . . . . . 12
1.3 Modern Convolution neural network designs . . . . . 20
1.3.1 LeNet . . . . . 21
1.3.2 AlexNet . . . . . 22
1.3.3 Block design in VGG . . . . . 23
1.3.4 Network in Network: 1 by 1 convolution and Global average pooling layer . . . . . 25
1.3.5 Batch normalization . . . . . 26
1.3.6 ResNet and ResNext . . . . . 28
1.4 Recurrent Neural Networks . . . . . 32
1.4.1 RNN . . . . . 33
1.4.2 LSTM . . . . . 34
1.4.3 GRU . . . . . 37
1.5 Attention Mechanisms and Transformers . . . . . 39
1.5.1 Attention . . . . . 39
1.5.2 Transformer . . . . . 41
1.6 Interpretability of Convolution models in transcription factor analysis . . . . 44

1.7 Opportunities and obstacles for deep learning in biology and medicine . . . . 47
1.7.1 Disease and patient categorization . . . . . 48
1.7.2 Transfer learning . . . . . 49
1.7.3 Multimodal learning . . . . . 50
1.7.4 Multi-task learning . . . . . 51
1.7.5 Model interpretability . . . . . 52
1.8 Multimodal machine learning in precision health . . . . . 53

**2 Scratch-AID, a deep learning-based system for automatic detection of mouse scratching behavior with high accuracy 55**

2.1 Abstract . . . . . 55
2.2 Introduction . . . . . 56
2.3 Data and methods . . . . . 58
2.3.1 Videotaping apparatus setting . . . . . 60
2.3.2 Video annotation . . . . . 62
2.3.3 Deep learning neural network methodology . . . . . 63
2.4 Results . . . . . 67
2.4.1 Model training procedure . . . . . 67
2.4.2 The assessment of model performance on test datasets . . . . . 68
2.4.3 Neural network recognized mouse scratching by focusing on the hind paw . . . . . 71
2.4.4 Analysis of Prediction Errors for the CRNN model . . . . . 71
2.4.5 Performance of the Scratch-AID system on other major acute itch models . . . . . 76
2.4.6 Performance of the Scratch-AID system on a chronic itch model . . . 78
2.4.7 Application of the Scratch-AID system in anti-itch drug screening . . 80
2.5 Discussion . . . . . 81
2.6 Future work . . . . . 83
2.7 Data Availability . . . . . 84
2.8 acknowledgments . . . . . 84

**3 Cancer survival prediction using multilevel multimodal co-attention deep learning model** **95**

3.1 Introduction . . . 95

3.1.1 Traditional cancer survival analysis: Cox proportional hazards (CPH) model . . . 96

3.1.2 Deep learning survival models . . . 97

3.1.3 Multi-instance learning for whole slide images . . . 99

3.1.4 Our proposed methodology . . . 101

3.2 Method . . . 103

3.2.1 Problem Formulation . . . 103

3.2.2 Feature extraction . . . 105

3.2.3 Hierarchical Genomic-WSI Fusion based on Multi-level Co-Attention Module . . . 107

3.2.4 Set-based Transformer encoder . . . 115

3.3 Experiment & Results . . . 116

3.3.1 Data and evaluation metric . . . 116

3.3.2 Model training . . . 117

3.3.3 Comparison with previously published models . . . 118

3.3.4 Ablation Study: Dissecting the Role of Genomic Features . . . 121

3.4 Interpretation of the full model . . . 123

3.4.1 interpretation methods . . . 123

3.4.2 Attention Visualization in BRCA . . . 124

3.4.3 Integrated gradient (IG) score indicating influential genes . . . 124

3.4.4 Influential gene analysis for BRCA subtypes . . . 125

3.4.5 Discussion . . . 131

3.5 Conclusion . . . 132

**4 Domain adaptation of foundation models for image classification in digital pathology** **138**

4.1 Abstract . . . 138

4.2 Introduction . . . . . 139
4.2.1 Digital pathology . . . . . 139
4.2.2 Deep learning in digital pathology . . . . . 139
4.2.3 Challenges: generalizability . . . . . 140
4.2.4 Foundation models in computer vision . . . . . 141
4.2.5 Challenges in foundation model domain adaptation . . . . . 143
4.3 Method . . . . . 144
4.3.1 Vision Transformer . . . . . 144
4.3.2 LoRA . . . . . 145
4.3.3 Data and model training . . . . . 149
4.4 Results . . . . . 151
4.5 Conclusion . . . . . 152

**5 Conclusion and Discussion** **154**

# List of Figures


1.1 An MLP with a hidden layer of five hidden units . . . . . 5

1.2 Two-dimensional convolution operation. The areas that are shaded represent the initial output element, as well as the input and kernel tensors that are used to calculate the output.: $0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$ . . . . . 15

1.3 Two-dimensional convolution layer with padding . . . . . 16

1.4 Convolution layer with strides of 3 and 2 for height and width, respectively. . 18

1.5 Convolution layer with two input channels . . . . . 19

1.6 Max-pooling with a pooling window shape of 2 by 2. The shaded portions are the first output element as well as the input tensor elements used for the output computation. . . . . 20

1.7 Model design of LeNet. The input is a handwritten digit, and the output is a probability of the digit belonging to one of the ten possible outcomes. . . . 21

1.8 From LeNet (left) to AlexNet (right). Dark blue represent max pooling layer, light blue represent convolution layer. Pad is padding. Parentheses indicate the channel number. . . . . 22

1.9 From AlexNet to VGG. VGG consists of blocks of layers, whereas AlexNet's layers are all designed individually. . . . . 24

1.10 For non-nested function classes, a larger area does not guarantee that we will be nearer to the "truth" function ($f^*$). This is not the case with nested function classes. . . . . 28

1.11 In a regular block (left panel), the network within the dotted-line box must learn the mapping $f(\mathbf{x})$ directly. On the other hand, in a residual block (right panel), the network within the dotted-line box has to learn the residual mapping $g(\mathbf{x}) = f(\mathbf{x}) - \mathbf{x}$, which makes it simpler to learn the identity mapping $f(\mathbf{x}) = \mathbf{x}$. . . . . 29

1.12 ResNet block with and without $1 \times 1$ convolution, which transforms the input into the desired shape for the addition operation. . . . . 30

1.13 Architecture of ResNet-18. . . . . 31

1.14 Architecture of ResNeXt block with $g$ groups. . . . . 32
1.15 The illustration on the left depicts recurrent connections through cyclic edges. The diagram on the right unfolds the RNN across a sequence of time steps, where recurrent edges connect adjacent time steps. . . . . 33
1.16 RNN with a hidden state architecture . . . . . 34
1.17 Computing the input gate, input node, the forget gate, and the output gate in an LSTM model. . . . . 35
1.18 Computing the memory cell internal state and the hidden state in an LSTM model. . . . . 37
1.19 Architecture of a GRU model. . . . . 38
1.20 The attention mechanism calculates a linear combination of the values $\mathbf{v}_i$ through attention pooling, with the weights determined by the similarity between the query $\mathbf{q}$ and the keys $\mathbf{k}_i$. It combines query and keys to achieve a **attention weight** towards values . . . . . 40
1.21 The Transformer architecture. FFN means feed forward network, which is fully connected layer. . . . . 43
1.22 (Left) Self-attention process. (Right) Multi-head attention. . . . . 44

2.1 **The overview of construction of a customized videotaping box for the recording of mouse scratching behavior.**. (A) A flow chart illustrating the process of developing a system based on deep learning for the automatic detection and quantification of rodent clawing behavior. (B) An image of the videotaping enclosure designed to capture high-quality video of the clawing behavior of the mouse. The scale bar measures 5 cm. (C) Caricature depicting the acute itching model induced by injection of chloroquine (CQ) into the nape, followed by video recording in the custom video recording box. (D) Images illustrating the various phases (P1–P4) of a striking train (upper). The red arcs represent the clawing rear paw. A cartoon depicting the dynamic movement of a scratching train's rear hand (bottom). In a scratching train, the cycle of clawing bout (P2) and lapping (P3) can occur multiple times. Scale bar, 1 cm. Each video contains a total number of scraping trains. (F) The distribution of the duration of the scratching trains (n = 1135 scratching trains). The inset is a magnified version of the red rectangle. 59

2.2 **Design and training of neural networks using deep learning.** (A) cartoon depicting the architecture of a deep learning neural network comprised of convolution neural networks (CNN), recurrent neural networks (RNN), and a classifier. (B) A cartoon depicting the compilation of training dataset inputs. N consecutive frames were chosen as one training input. Four to ten frames separated two adjacent inputs in a video. (C) The sample training and test datasets' information. During the training procedure with various input lengths (N = 3, 5, 7, 13, 23, 45 frames), the training loss decreased (D) while accuracy increased (E). The inset is a magnified portion of the figure. . 65

2.3 **The architecture of deep learning neural network.**(A) Image illustrating the structure of a convolution recurrent neural network (CRNN). First, convolution neural networks (CNN, ResNet-18) were provided the input, followed by a two-layer bidirectional gated recurrent unit (GRU). The results of the binary prediction were derived from the final full connection layer (FC). (B) Information about the modified ResNet-18 network. The final FC layer was altered to output frame-wise embedding as opposed to classificationmangel. convolution layer, or Conv; Batch norm, stratum of batch normalization; ReLU, rectified linear unit; +, vector element-wise addition. (C) Specifics of one GRU of the bidirectional, two-layer GRU. The unit connected to the embedding of the frame $t(x_t)$ used the output of the previous unit as input and produced a new output ($h_t$). Inside the unit there were three yellow squares representing the reset gate, the update gate, and the candidate activation vector. x represents the Hadamard product for vectors; + represents the vector plus. . . . . 66

2.4 **The top model's performance on test videos.** The top model's average recall, precision, and F1 score (A) or in individual videos (B). The average recall, precision, and F1 score of manual annotation (C) or for specific videos (D). Comparison between model prediction, manual quantification, and reference annotation. Red line indicates that the reference annotation has been normalized to 100 percent. Correlations between model prediction or human quantification and reference annotation. Pearson's correlation coefficient, $R^2$. (G) An illustration of a scratching probability trace predicted by the model and aligned with the reference annotation (green bar). (H) The two zoomed-in portions of figure (G) demonstrating the excellent alignment between the model prediction and the reference annotation. . . . . 70

2.5 **For scratching behavior identification, the prediction model focused on the scratching hind paw.** (A, B) Saliency map displaying the gradient value of each pixel of clawing frames during the best model's prediction of mouse scratching behavior. The model highlighted the scratching rear paw (A, B) as well as other body regions, such as the front claws (B).Scale bar measures 2 cm. Saliency map displaying the gradient value of each pixel during mouse clawing behavior prediction during cleaning (C), grooming (D), rearing (E), and locomotion (F) frames. . . . . 72

2.6 **Analysis of the top prediction model's error rates.** (A) Cartoon depicting five prediction mistakes. Red curves showed the predicted scratching probability from the model, while green bars showed the reference scratching trains. (B) The incidence rate of each type of mistake, obtained by dividing the total frames in each error by the total scratching frames of 8 test videos. SEM error bar. A real example of false positive prediction (C1), the duration of all false positive scratching trains (C2), and the frequency distribution of distances between them and the next real scratching train (C3). A actual false negative prediction (D1), the duration of all false negative scratching trains (D2), and the Type 2 error rate for scratching trains of different lengths (D3). Real example of blurred boundary prediction (E1), distribution (E2), and start/end shift average lengths (E3). SEM error bar. Actual missed interval (F1), duration of all missed intervals (F2), and Type 4 error rate for intervals of varying durations (F3). A actual split scratching train (G1), split frame length (G2), and Type 5 error distribution from paw licking (G3). . . 75

2.7 **The Scratch-AID (Automatic Itch Detection) performance on other acute itch models.** (A) A cartoon showing an acute itch model induced by chloroquine (CQ) injection in the mouse cheek. Average recall, precision, and F1 score of Scratch-AID (B) or manual annotation (C). Error bar, standard error of the mean (SEM). The correlation between model prediction (D) or manual quantification (E) and reference annotation. $R^2$, Pearson correlation coefficient. (F) The comparison among model prediction, manual quantification, and reference annotation. The reference annotation is normalized to 100% shown as the red line. (G) Cartoon showing an acute itch model induced by histamine injection in the mouse nape. Average recall, precision, and F1 score of Scratch-AID (H) or manual annotation (I). Error bar, SEM. The correlation between model prediction (J) or manual quantification (K) and reference annotation. $R^2$, Pearson correlation coefficient. (L) The comparison among model prediction, manual quantification, and reference annotation. The ref between annotation is normalized to 100% shown as the red line. . . . . 77

2.8 **The Scratch-AID performance on a chronic itch model.** The cartoon in (A) illustrates a squaric acid dibutylester (SADBE) induced chronic itch model. The performance of Scratch-AID (B) and manual annotation (C) is evaluated by the average recall, precision, and F1 score, with the standard error of the mean (SEM) shown as error bars. The correlation between model prediction (D) and manual quantification (E) with the reference annotation is measured by the Pearson correlation coefficient ($R^2$). (F) compares the model prediction, manual quantification, and reference annotation, with the reference annotation normalized to 100%. (G) shows an example of the scratching probability trace (red curve) predicted by the model and aligned with the reference annotation (green bar). A zoom-in (right panel) of the blue square part reveals a good alignment of the model prediction with the reference annotation. . . . . 78

2.9 **Different dynamic features of chronic and acute itch models.** (A) The percentage of scratching and non-scratching frames in the squaric acid dibutylester (SADBE) chronic itch model (n = 9 videos) and chloroquine (CQ) nape acute itch model (n = 40 videos). (B) Frequency distribution of scratching train duration of SADBE chronic itch model and CQ nape acute itch model. . . . . 79

2.10 **Application of the Scratch-AID (Automatic Itch Detection) system in a drug screening paradigm.** (A) A diagram showing the experimental design of an anti-itch drug test. Quantification of scratching behavior in anti-itch cream treated group or control group by Scratch-AID (B) or manual annotation (C). Error bar, standard error of the mean (SEM). Differences between the two groups were analyzed using unpaired two-tailed Student's t-test, ** $p < 0.01$. . . . . 80

2.11 **Dynamic and static features of scratching behavior in the chloroquine (CQ) nape acute itch model.**, B) The scratching hind paw (red arrow in A) vibrated rhythmically during scratching behavior, but the contralateral hind paw did not (B). Distance between scratching hind paw and nose (C) or mouth (D), angle formed by nose, scratching hind paw, and ipsilateral front paw (E), and angle formed by scratching hind paw, mouth, and tail (F) in scratching or non-scratching frames (N = 1756 for non-scratching, 2214 for scratching). Standard error of the mean (SEM) error gauge. The differences between the two groups were analyzed using an unpaired two-tailed Student's t-test with a significance level of 0.001. . . . . 86

2.12 **Distribution of scratching episode duration and paw licking in the nape acute itch model induced by chloroquine (CQ).** The frequency distribution of scratching bout (A) or paw lapping (B) duration. The figure inset is a magnification of the red square portion. . . . . 87

2.13 **Cross-validation and parameter tuning for the prediction models.** (A) Cross-validation of the trained prediction models through the rotation of the training and test datasets across 40 videos. The F1 rating was determined using the eight evaluation videos. (B) The optimal performance of models trained with N=45 inputs and various combinations of training and test datasets. (C) Simulate outputs with varying input lengths. . . . . 88

2.14 **Error analysis for the manual annotation.** (A) The incidence rate of each category of manual annotation error, calculated as the ratio of the total number of frames containing each type of error to the total number of scratching frames in eight test videos. Standard error of the mean (SEM) error gauge. (B) The length of time for all false-positive scratching trains. The duration of all false negative trains (C1) and the Type 2 error rate for scratching trains with varying durations (C2). The distribution (D1) and average length (D2) of shift start and end durations. The duration of all missed intervals (E1) and the Type 4 error rate for intervals with different duration (E2). The duration of divided frames. . . . . 89

2.15 **Other mouse behaviors that were not identified as scratching.** The predicted probabilities of scratching behavior during wiping (A), grooming (B), rearing (C), locomotion (D), and resting (E). . . . . 90

2.16 **Relationship between prediction errors and the length of the input or the extent of the scratching train.** (A–C) The five type error rates of models trained with varying lengths of input data. Standard error of the mean (SEM) error gauge. The relationship between the duration of the scratching train and the length of the start shift, end shift, or start shift plus end shift. The correlation between the average scratching train duration in a video and the accuracy of the prediction (F1 score). (F) Frequency distribution of all durations of paw lapping and those associated with Type 5 error. . . . . 91

2.17 **Prediction saliency map of mouse scratching and other activities.** Additional saliency maps displaying the gradient value of every pixel for frames of scratching (A), wiping (B), grooming (C), rearing (D), and locomotion (E) during the best model's scratching behavior prediction. . . . . 92

3.1 Our Genomic-Guided Hierarchical Attention Multi-Scale Pathology Transformer (GG-HAMPT) model architecture has two inputs: a gigabyte whole slide image and genomics features (including gene expression RNA seq and copy number matrix). The model is composed of three parts: feature extraction (as described in Section 3.2.2), partial attention and multi-scale co-attention (as described in Section 3.2.3), and Transformer encoder and pooling (as described in Section 3.2.4). Finally, Section 3.2.1 discusses how the features extracted from the network are used to predict survival. . . . . 103

3.2 Schematic representation of the hierarchical segmentation and feature extraction process for whole slide images (WSIs). Starting with the complete slide set, an initial segmentation segregates tissue-containing regions, forming patches at the organ level with dimensions $4096 \times 4096$. These patches are further segmented to produce finer-grained patches at the tissue level ($1024 \times 1024$) and subsequently at the cellular level ($256 \times 256$). Each granularity level undergoes feature embedding through a ResNet-50 CNN encoder. The flowchart illustrates the transformation of a single WSI into a myriad of patches, culminating in feature sets $H_{\text{organ}}$, $H_{\text{tissue}}$, and $H_{\text{cell}}$. The varying patch sizes denote the progression from macroscopic organ-level details to intricate cellular-level insights. The methodology aims to capitalize on the rich hierarchical information embedded within WSIs, surpassing conventional techniques that might overlook the intrinsic relationships between these diverse patch levels. . . . . 106

3.3 Illustration showcasing the process of genomic data classification and feature extraction. Starting with a diverse array of genomic features from the TCGA portal, genes are categorized into six distinct functional groups based on their biological roles. Each group is then processed through a dedicated Fully Connected (FC) layer, capturing the nuanced genomic characteristics specific to that category. The final product is an aggregate genetic bag-of-features that integrates information across all functional groups, providing a complete representation of the genomic landscape. . . . . 107
3.4 Schematic representation of the multi-tier attention layer in action. Beginning with the extracted patch features from whole slide images, the figure details the process of hierarchical attention across organ, tissue, and cell levels. The attention mechanism is finely-tuned to emphasize key features within each hierarchy, facilitating a nuanced understanding of the intricate relationships embedded within the whole slide images. . . . . 110
3.5 Schematic visualization of the Multi Layer Genomic-Guided Co-Attention mechanism. This multi-layered representation showcases the interaction between genomic features and pathology features across the organ, tissue, and cell levels. Emphasizing the relationships between the different levels, GCA integrates genomic-guided attention to yield refined pathology feature embeddings. . . . . 114
3.6 Kaplan-Meier Analysis was used to compare the survival time of low-risk (blue) and high-risk (red) patients based on their predicted risk scores from our model. The Logrank test (weight set as wilcoxon) was used to measure the statistical significance of the difference between the two survival distributions. 121
3.7 Co-attention visualization for low risk cases in BRCA patient TCGA-B6-A01A, with corresponding high attention patches and high-attributed genes in each heatmap. . . . . 133
3.8 Co-attention visualization for high risk cases in BRCA, with corresponding high attention patches and high-attributed genes in each heatmap. . . . . 134

4.1 Computer Vision Benchmark on ImageNet (Top 1 accuracy / 1000 classes) . 141
4.2 Segment Anything Model demo with Brooklyn Bridge (Kirillov *et al.,* 2023). 143
4.3 The framework of ViT (Dosovitskiy *et al.,* 2020). The image was divided into patches of a fixed size, ordered, then embedded linearly, and position embeddings were added. The sequence of vectors that resulted was then fed to a regular Transformer encoder. To classify, the usual approach of adding a trainable "classification token" to the sequence was used. . . . . 145
4.4 The LoRA layer. Only A and B will be trained. . . . . 146
4.5 Comparison between transformer models tuned with and without LoRA and ResNet . . . . . 147
4.6 The LoRA layer in ViT transformer layer. The LoRA layer was applied to the $q$ (query) and $v$ (value) projection layer of each of the transformer block in the ViT (excluding $k$ (key) layer). "Proj.q", "Proj.k", "Proj.v", "Proj.o" represent the projection layer of $q, k, v$ and $o$, respectively. Plus means element wise add, and multiply means matrix production. . . . . 148
4.7 Sample and numbers of pictures for cell type classes: Epithelial, Lymphocyte, Neutrophil, Plasma, Stroma, Tumor. . . . . 150

# List of Tables

1.1 Evolution of Dataset Size, Memory, and Computational Power over Decades (adapted from A. Zhang *et al.,* 2023) . . . . . 4

2.1 **Residual model parameters, total parameters 11M** . . . . . 85
2.2 **Model parameters, total parameters 526 M** . . . . . 85
2.3 Mouse information used in the recording of the training and test videos . . . 93
2.4 Scratching behavior summary in the 40 training and test videos (reference annotation) . . . . . 94

3.1 Performance of different models on various cancer types. Values represent the C-index with the standard deviation. BLCA: bladder urothelial carcinoma, BRCA: breast invasive carcinoma, GBM / LGG: glioblastoma and lower grade glioma, LUAD: lung adenocarcinoma, UCEC: uterine corpus endometrial carcinoma. . . . . 120
3.2 Performance comparison of different feature types and modalities on different cancer types. All means with all 3 genomics modality: RNASEQ, CNV and Mutation. Mut stands for Mutation, and Image means using no genomic modality data. . . . . 123
3.3 Average normalized absolute IG score of all patients from five different cancer cohorts. The highest average normalized absolute IG score across five cross validation models is referred to as the "Absmax". These IG scores are used to demonstrate the significance of genetics of different cancer cohorts. Many cancer oncogenes are on the high-impact list. . . . . 135

3.4 Average absolute IG score of all patients from 4 different BRCA cancer subtypes: Her2, LumA, LumB, Basal. The highest average normalized absolute IG score across five cross validation models among all BRCA patients is referred to as the "Absmax", the highest average normalized absolute IG score across five cross validation models among all BRCA patients from a specific subtype is referred to as the "SubtypeMax". Maxsubtype indicating the maximum of 4 "SubtypeMax". These IG scores are used to demonstrate the significance of BRCA subtype specific genetics. This table was sorted by the Maxsubtype. . . . . 136

3.5 Comparison of Average Absolute IG Scores Across BRCA Cancer Subtypes (Positive difference only). The table lists genes with their corresponding "Absmax" scores, which represent the highest average normalized absolute IG score across all BRCA patients, and the "Maxsubtype" scores, indicating the maximum of the four subtype-specific scores. The "Difference" column illustrates the difference between the "Absmax" and "Maxsubtype" scores. Genes with a larger difference may have a more pronounced role in specific breast cancer subtypes. This table is sorted by the "Difference" in descending order to highlight genes of potential subtype-specific significance. . . . . 137

4.1 The performance of Vision Transformers (ViT) with Low-Rank Adaptation (LoRA) and a ResNet50 model across various configurations in experiment 1. Test Acc means accuracy on the test set of the best model. Infer Time (s) means for one batch of data, how many seconds the model will spend on inference stage. . . . . 152

4.2 The performance of Vision Transformers (ViT) with Low-Rank Adaptation (LoRA) and a ResNet50 model across various configurations in experiment 2. Test Acc means accuracy on the test set of the best model. Infer Time (s) means for one batch of data, how many seconds the model will spend on inference stage. . . . . 153

4.3 The performance of Vision Transformers (ViT) with Low-Rank Adaptation (LoRA) and a ResNet50 model across various configurations in experiment 3. 153

# Abstract

**Machine Learning Methods For The Detection Of Complex Patterns In Biological Data**

The biomedical field is being revolutionized by deep learning. The accumulation of more, and more complex data opens the opportunity for the development of new advanced deep learning models that can address fundamentally difficult question in biology. Nevertheless, the integration of deep learning into the life sciences is complicated and necessitates a careful strategy in order to tackle specific domain challenges while optimizing the vast quantities of very heterogeneous data. This thesis addresses very different biological problems using deep learning methods. A common theme to these problems is the complexity and richness of the large data sets collected. In chapter 2, I introduce *Scratch-AID*, an exceptionally precise deep learning-based system designed to automate the identification of mouse scratching in a controlled environment. By employing a convolution recurrent neural network trained on video data, this system attains a recall percentage of 97.6% and a precision percentage of 96.9%. Therefore, *Scratch-AID* can be considered a feasible substitute for manual quantification techniques in the context of pharmacological screenings and behavioral studies.

Following this, in Chapter 3, I address the problem of survival in cancer patients. I present a deep learning framework called *Genomic-Guided Hierarchical Attention Multi-Scale Pathology Transformer* (GG-HAMPT). I introduce an multi-level early data fusion strategy that uses the co-attention paradigm to combine genomic data and 3 level representation of whole slide Images (WSI) of cancer tissues. Employing the co-attention module, our model adeptly discerns the significance of pathological patches in correlation with grouped genomic features. In the context of predicting survival outcomes using multimodal data, including gigapixel WSIs, our GG-HAMPT model outperformed existing other weakly supervised approaches.

Last, we examine the domain adaptation of pre-trained models for the purpose of classifying digital pathology images through the implementation of the LoRA Vision Transformer (LoRA-ViT). The model exhibits remarkable efficiency and accuracy. Furthermore, in a 6-class cell classification task, LoRA-ViT outperformed the native ResNet 50 model. In

experiments conducted with constrained datasets, the LoRA-ViT model significantly outperformed the ResNet 50, showing its superior generalizability compared to the baseline model.

# Acknowledgments

First and foremost, I extend my deepest gratitude to Professor Javier Arsuaga for his exceptional guidance and mentorship. His profound research in breast cancer not only sparked my interest in this critical field but also provided continual inspiration throughout my journey. I am particularly thankful for Professor Arsuaga's invaluable assistance and insightful inputs in the writing process of my thesis. His dedication to his students is truly remarkable, and working under his tutelage has been both a privilege and a pleasure.

I would also like to express my sincere appreciation to Dr. Huasheng Yu and Professor Wenqin Luo for the opportunity to collaborate on the significant dataset of mouse itch detection. Their trust and the autonomy they granted me in this work have been deeply motivating.

Special thanks are due to Professor Radmila Sazdanovic for her continuous involvement in the cancer group meetings and her insightful discussions on cancer research. Collaborating with her, Dr. Jai Aslam, and Professor Sergio Ardanza-Trevijano on the TAaCGH Suite for Detecting Cancer has been a remarkable experience that introduced me to cancer research.

I would like to extend my deepest gratitude to Professor Jeffrey L. Thorne for guiding me into the realm of scientific research during my undergraduate studies. His exceptional mentorship and inspiring approach to teaching have been instrumental in my decision to pursue a Ph.D. His unwavering support and valuable insights have significantly shaped my academic journey, and for that, I am truly thankful.

I am also grateful to Professor Mariel Vazquez and Georgina Gonzalez for their efforts in organizing the journal club alongside Javier. Their contribution to fostering a stimulating academic environment has been significant.

This research was partially supported by the National Science Foundation (NSF) grant DMS-185477, for which I am profoundly thankful. The support and resources provided by this grant were instrumental in the completion of my research.

I must also extend my heartfelt thanks to my family. Their love, understanding, and sacrifices have been the bedrock of my journey. Their unwavering belief in my abilities and constant encouragement have been my greatest strength. I am indebted to them for their

immeasurable support and patience throughout my academic pursuits.

Lastly, I extend my gratitude to the Department of Statistics at UC Davis. The department's exceptional teaching and support have been foundational in my academic development.

# Chapter 1

# Introduction

This thesis explores the use of deep learning models in the biomedical field, taking advantage of the abundance of data available to tackle complex datasets that is difficult to analyze using conventional methods. It will focus on how to design cutting-edge deep learning methods for biomedical applications.

## 1.1 Thesis overview

This thesis is structured as follows. Chapter 1 provides general motivation, background, and a comprehensive overview of deep learning in the biomedical field. Additionally, it introduces some definitions and notation of deep learning model structures that are necessary for the subsequent chapters. The structure of the deep learning methodology introduction follows the *Dive into Deep Learning* book (A. Zhang *et al.,* 2023).

Chapter 2 presents a new approach to automating the recognition of mouse scratching behavior with remarkable accuracy. This system, which uses a convolution recurrent neural network trained on high-quality video data, provides a viable replacement for manual quantification techniques in both behavioral studies and pharmacological screenings.

Chapter 3 introduces the Genomic-Guided Hierarchical Attention Multi-Scale Pathology Transformer (GG-HAMPT), a model that integrates genomic data with detailed pathology information using a co-attention mechanism for enhanced cancer survival prediction, outperforming other multimodal data-based weakly supervised methods.

Chapter 4 presents the LoRA Vision Transformer (LoRA-ViT), a model adept at adapting

pre-trained models for digital pathology image classification, surpassing the baseline ResNet 50 in accuracy, especially when trained with limited datasets.

In the last chapter, a general summary is presented, along with some pertinent information related to the thesis and potential paths for further exploration.

## 1.2 Deep learning background

Research in biology and medicine is rapidly transitioning and becoming a data-based sciences. It is anticipated that in the next ten years, genomics will be on par with, or even exceed, other data-driven fields such as social media, online multimedia, and related areas in terms of data production and analysis needs (Ching *et al.,* 2018). This surge of data brings with it a plethora of possibilities, but also presents many new difficulties related to big data. The solution to the challenge of big data in biology and medicine is to develop deep learning algorithms that can successfully handle the large scale data that traditional methods cannot evaluate. This could revolutionize the way we approach treatment, categorize patients, and predict diseases, while still protecting privacy.

In recent years, deep learning has become a major approach to solving scientific problems in data science. These methods have made considerable progress compared to traditional machine learning algorithms in many areas. For example, in the last five years, these techniques have revolutionized image recognition and voice processing. Furthermore, applications of deep learning are not limited to these areas. Generic versions of these deep learning algorithms, without much customization, have achieved accuracy values that are similar to or even better than the best methods in the genomics area (Novakovsky *et al.,* 2023, Zou *et al.,* 2019). Moreover, specialized versions are now being widely used in industrial settings (Jumper *et al.,* 2021).

### 1.2.1 Historical Background of Deep Learning

Deep learning dates back to the concept of artificial neural networks (ANNs), which has its roots deep embedded in the mid-20th century, drawing inspiration from biological neural networks found within the human brain. It was in 1943 that the pioneering work of Warren McCulloch and Walter Pitts introduced the notion of an artificial neuron (McCulloch & Pitts, 1943). Their proposal was not just revolutionary; it provided the rudimentary foun-

dation for the subsequent development of neural networks. The model they postulated was a computational representation that captured the essence of how biological neurons might process information. After the seminal work of McCulloch and Pitts, the field experienced several peaks and troughs of interest and development. One of the first notable milestones in this journey was the perceptron, introduced by Frank Rosenblatt in the late 1950s (Rosenblatt, 1958). The perceptron was an algorithmic approach to pattern recognition based on a two-layer network, and it generated significant interest in the potential of these networks. However, when Marvin Minsky and Seymour Papert mathematically demonstrated the many restrictions of two-layer feedforward neural networks, such as their inability to learn non-linear functions or execute the Boolean exclusive OR (XOR) operation (Marvin & Seymour, 1969), the enthusiasm for the perceptron temporarily diminished. Interest in neural networks was rekindled in the 1980s, with the invention of the backpropagation algorithm by David Rumelhart, Geoffrey Hinton, and Ronald Williams (Rumelhart *et al.,* 1986). This algorithm enabled the training of multilayer neural networks, overcoming many of the earlier obstacles and leading to the current era of deep learning.

Deep learning is a subfield of machine learning characterized by the use of layered architectures that allow for complex model structures. Unlike traditional machine learning methods, such as Support Vector Machines and Decision Trees, which require human expert for feature engineering, deep learning models are capable of automatically extracting features from raw data through multiple layered architectures. This is why deep learning models are referred to as "deep" - they encompass numerous layers. These multi-layered architectures enable deep learning models to identify intricate, nonlinear relationships in data, something that was difficult for the perceptron or traditional algorithms. With the increased power of computers and newly collected large datasets, deep learning models, particularly Convolution Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have become more successful than conventional models in areas such as image recognition and natural language processing (LeCun, Bengio, *et al.,* 2015b). The World Wide Web, the emergence of companies providing services to hundreds of millions of users online, the proliferation of low-cost, high-quality sensors, the availability of inexpensive data storage (Kryder's law), and the affordability of computing (Moore's law) have drastically altered the

landscape of data (Walter, 2005, Schaller, 1997). In particular, the development of graphic processing units (GPU), initially designed for computer gaming, has revolutionized the field of deep learning, making algorithms and models, previously thought to be computationally impossible, achievable. This is best demonstrated in Table 1.1.

| **Decade** | **Dataset** | **Memory** | **Floating point calculations per second** |
|---|---|---|---|
| 1970 | 100 (Iris) | 1 KB | 100 KF (Intel 8080) |
| 1980 | 1 K (house prices in Boston) | 100 KB | 1 MFLOPS (Intel 80186) |
| 1990 | 10 K (optical character recognition) | 10 MB | 10 MFLOPS (Intel 80486) |
| 2000 | 10 M (web pages) | 100 MB | 1 GFLOPS (Intel Core) |
| 2010 | 10 G (advertising) | 1 GB | 1 TFLOPS (NVIDIA C2050) |
| 2023 | 1 T (social network) | 100 GB | 204.9 TFLOPS (NVIDIA H100) |

Table 1.1: Evolution of Dataset Size, Memory, and Computational Power over Decades (adapted from A. Zhang *et al.,* 2023)

### 1.2.2 The concept of Multilayer Perceptron (MLP) and Fully Connected (FC) layers

Statistical models often rely on affine transformations, which are based on linear mappings from one vector space to another. These transformations typically presume a linear relationship between input variables and the predicted output, implies monotonicity where any increase (or decrease) in an input feature leads to a proportional increase (or decrease) in the output, assuming the associated weight is positive (or negative).

However, this presumption of linear behavior can be overly simplistic, especially in complex tasks such as image classification. For instance, consider the problem of distinguishing between images of cats and dogs. A linear model might suggest that increasing the intensity of a particular pixel would consistently affect the likelihood of the image being recognized as a dog in a predictable manner. Yet, image classification is inherently non-linear and context-dependent—a change in pixel intensity could mean something entirely different depending on the overall pattern of pixels. Furthermore, factors such as texture, shape, and spatial relationships between pixels are important. Simply put, inverting an image's colors does not alter the subject of the image, but a linear model might fail to recognize this due to its reliance on individual pixel values rather than the broader context they create together. Thus, while linear models are powerful tools, their application is limited in scenarios that require

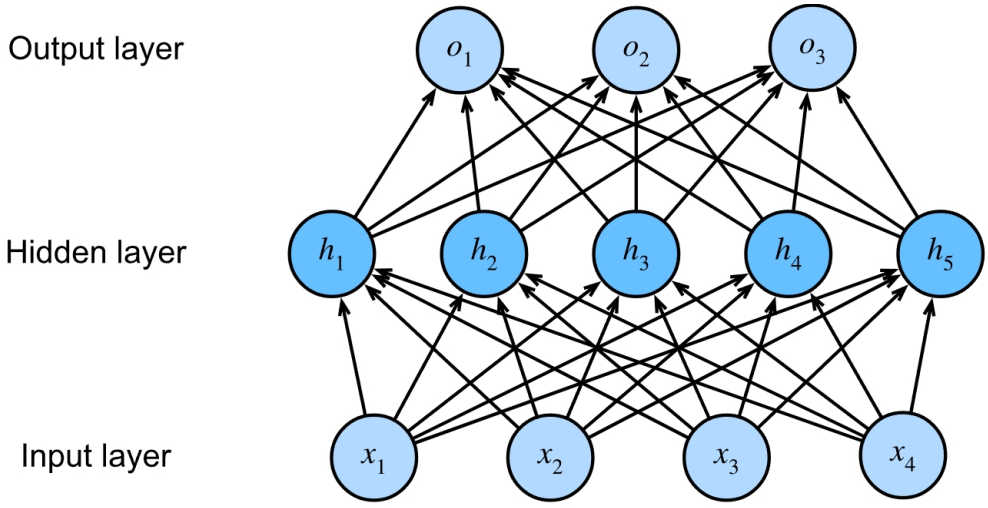



Figure 1.1: An MLP with a hidden layer of five hidden units

the modeling of complex, non-linear relationships as seen in image recognition tasks."

For more than a century, scientists have been exploring the challenge of dealing with nonlinearity, such as the example mentioned above. Decision trees (Salzberg, 1994), Kernel methods (Aronszajn, 1950), Nonparametric spline models (Wahba, 1990) and kernel methods combined with support vector machines (Schölkopf & Smola, 2002) are long-standing techniques to identify nonlinear relationships. It should be noted that this kind of nonlinear processing is also observed in the brain, where neurons transmit signals to other neurons, forming a sequence of basic transformations.

In deep learning, we can overcome the limitations of linear models by incorporating one or more hidden layers. The easiest way to do this is to stack many fully connected layers. Each layer feeds into the next layer. Suppose we have $L$ layers, we can think of the first $L-1$ layers as our representation and the final layer as our linear predictor. This architecture is commonly called a *multilayer perceptron*, often abbreviated as MLP.

The structure of MLP was illustrated in Figure 1.1. MLP comprises *input nodes*, *output nodes*, and a *hidden layer* consisting of *hidden units*. While the input layer serves as a passive conduit for the raw data, computations are actively performed in the hidden and output layers, making the total number of layers in this MLP two. Each layer is fully connected, leading to the designation of MLPs as *fully connected layers* (*FC layers*). In this architecture, every input node directly affects each neuron in the hidden layer, which

subsequently influences each neuron in the output layer.

Now let's mathematically define the MLP and fully connected (FC) layers. Consider a data representation as $\mathbf{X} \in \mathbb{R}^{n \times d}$, where we have a minibatch of $n$ data points and each data point has a $d$ dimension vector (or feature). For a MLP with one hidden layer consisting of $h$ hidden units, the hidden and output layers are both fully connected, meaning that the hidden-layer weights $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times h}$ and biases $\mathbf{b}^{(1)} \in \mathbb{R}^{1 \times h}$ and output-layer weights $\mathbf{W}^{(2)} \in \mathbb{R}^{h \times q}$ and biases $\mathbf{b}^{(2)} \in \mathbb{R}^{1 \times q}$ are present. This allows us to compute the outputs $\mathbf{O} \in \mathbb{R}^{n \times q}$ of the one-hidden-layer MLP as follows:

$$\mathbf{H} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)},$$
$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}.$$

At first glance, it might appear that simply stacking more of these layers would allow the model to learn more complex representations. However, the reality is slightly counter-intuitive. Without introducing some non-linearity between these layers, multiple affine transformations can always be reduced to a single affine transformation, rendering the stacking redundant. When we define $\mathbf{W} = \mathbf{W}^{(1)}\mathbf{W}^{(2)}$ and $\mathbf{b} = \mathbf{b}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$, we have:

$$\mathbf{O} = (\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)} = \mathbf{X}\mathbf{W}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)} = \mathbf{X}\mathbf{W} + \mathbf{b}.$$

This brings us to a pivotal component in MLPs: **non-linear** activation functions. Possible activation functions include:

- ReLU (rectified linear unit):
$$\sigma(x) = \max(0, x)$$

- Sigmoid function:
$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$

- Hyperbolic tangent function function:
$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}$$

- Parametrized ReLU (pReLU) function (K. He *et al.*, 2015):

$$\text{pReLU}(x) = \max(0, x) + \alpha \min(0, x)$$

The ReLU (rectified linear unit) activation function has become increasingly popular due to its simplicity and effectiveness (Nair & Hinton, 2010). Additionally, it is much easier to optimize than the sigmoid or tanh functions. The use of ReLU was a major factor in the resurgence of deep learning in the last decade. With the activation function $\sigma$, the fully connected layer becomes:

$$\mathbf{H} = \sigma(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})$$

Subsequent stacking of fully connected layer interspersed with non-linear activations results in the traditional MLP architecture, e.g., $\mathbf{H}^{(1)} = \sigma_1(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})$ and $\mathbf{H}^{(2)} = \sigma_2(\mathbf{H}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)})$. Empirically, the model can approximate **any** highly complex functions, thereby capturing intricate patterns and relationships inherent in the data (Simonyan & Zisserman, 2014).

### 1.2.3 Deep Learning Model Training

The process underlying training a deep learning neural network is similar to that of training classical statistic models. Yet, an untrained model, regardless of its complexity, can be visualized as a blank slate, devoid of any "intelligence." Transforming this untrained model into an adept predictor involves several steps:

1. Start with a model with randomly initialized parameters.
2. Acquire data samples, such as virus sequences, coupled with their corresponding labels.
3. Update the model's parameters to enhance its performance on these samples.
4. Iteratively repeat steps (2) and (3) until the model's performance reaches a satisfactory threshold.

An example of designing a "cat-dog detector" could be based on giving a deep learning system many pictures of cats and dogs. Over time, the model will be able to recognize the

unique characteristics of each. If we set the cat as positive labels, eventually, it would give a large positive number for cat images and a pronounced negative number for dog images. For images that are hard to classify, the model would likely output a value close to zero.

**Supervised Learning** Supervised learning is a type of machine learning where the goal is to construct a predictive model based on a set of labeled training data. Each instance in the training dataset consists of one *feature vector*, which is a collection of measurements or observable characteristics, paired with a corresponding 'label', which is the outcome or category we aim to predict. For example, in a dataset for email classification, the features could include the words in the email, while the label would indicate whether the email is 'spam' or 'not spam'.

The process begins with the model learning patterns from the provided examples. This learning phase involves adjusting the model's parameters to minimize the difference between its predictions and the actual labels—a process known as 'training'. Once the model is trained, it can generalize from the patterns it has learned to make predictions on new, unseen data.

Supervised learning models range from basic linear regression used for forecasting continuous results to complex deep neural networks that can manage intricate tasks such as recognizing objects in images or translating text between languages. The success of supervised learning is not only dependent on the quality and amount of the training data, but also on the model's capacity to capture the underlying connections between features and labels without overfitting to noise or anomalies in the training set.

**Objective Function and Model Evaluation** Deep learning is often described as "learning from experience". This means that the model is able to improve itself autonomously in order to complete certain tasks. To measure the success of this improvement, a metric is needed. This metric, which is usually defined by the user, is known as the objective function. As its main purpose is to minimize the cost of the model in order to achieve the best performance, it is often referred to as the loss or cost function.

The value of the loss function depends on the model's parameters, and the data it's being trained on. The goal during training is to adjust the model parameters to minimize this loss, which indicates better performance and more accurate predictions. To reduce total loss,

the model parameters are modified for a given dataset. This data set, which is made up of samples used for training, is called the *training dataset*. However, a model that performs well on the training data may not be as successful on a new dataset, known as the *test dataset*. In the field of deep learning, this problem is called *overfitting*. Overfitting is reflected in a model's statistical metrics: it often has a low loss function because it fits the training data well, but high loss function when predictions are made on new data.

**Optimization and Gradient Descent** In order to find the optimal parameters to minimize the loss function, an algorithm is needed. In deep learning, *gradient descent* is the most commonly used optimization algorithm. This algorithm evaluates each parameter at each step and determines the direction of the least amount of loss when making small changes. It then adjusts the parameters in that direction to reduce the loss.

**Forward propagation** Forward propagation is the process of calculating and storing intermediate variables (including outputs) for a neural network, beginning from the input layer and ending at the output layer. This approach is similar to the one used in maximum likelihood estimation (MLE), but while MLE computes the likelihood of an entire dataset, forward propagation only computes the loss function for the current batch of data.

**Back propagation** Backpropagation is a technique to calculate the gradient of the parameters of the neural network. The algorithm stores any intermediate variables (partial derivatives) necessary to calculate the gradient with respect to the parameters. This is done by applying the chain rule and computing the gradient of each intermediate variable and parameter in reverse order, starting with the final loss function and back towards all intermediate parameters.

**Network training** When training neural networks, forward and backward propagation are interdependent. After initializing the model parameters, we alternate between forward and backpropagation, using the backpropagation gradients to update the model parameters. Backpropagation reuses the intermediate results from forward propagation for all of the layers to avoid repeating calculations. This means that these intermediate results must be stored until backpropagation is complete. This is why training requires more memory than prediction. Additionally, the memory size of the intermediate results is proportional to the

number of layers in the network and the batch size (Mo *et al.,* 2017). Therefore, training deeper networks with larger batch sizes can easily lead to out-of-memory errors.

**Mathematical formulation** We can express the training procedure for a MLP model using mathematical formulas. In the training of a MLP with one hidden layer, the forward pass computes outputs using the following sequence of operations for input $\mathbf{x} \in \mathbb{R}^d$:

1. The hidden layer transformation without bias: $\mathbf{z} = \mathbf{W}^{(1)}\mathbf{x} \in \mathbb{R}^h$, where $\mathbf{W}^{(1)} \in \mathbb{R}^{h \times d}$.
2. Applying the activation function $\phi$: $\mathbf{h} = \phi(\mathbf{z})\mathbb{R}^h$.
3. The output layer transformation: $\mathbf{o} = \mathbf{W}^{(2)}\mathbf{h}\mathbb{R}^h$, with weights $\mathbf{W}^{(2)} \in \mathbb{R}^{q \times h}$.

The regularization term $s$ in the context of training a Multi-Layer Perceptron (MLP) with $\ell_2$ regularization is defined as the sum of the squared Frobenius norms of the weight matrices of the network. Specifically, for a MLP with one hidden layer, the term $s$ is given by:

$$s = \frac{\lambda}{2}\left(\|\mathbf{W}^{(1)}\|_{\mathrm{F}}^2 + \|\mathbf{W}^{(2)}\|_{\mathrm{F}}^2\right),$$

where $\lambda$ is the regularization parameter, and $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm of a matrix.

The loss for a single example with label $y$ is computed as $L = l(\mathbf{o}, y)$. Incorporating $\ell_2$ regularization parameterized by $\lambda$, the total regularized loss becomes $J = L + \frac{\lambda}{2}\left(\|\mathbf{W}^{(1)}\|_{\mathrm{F}}^2 + \|\mathbf{W}^{(2)}\|_{\mathrm{F}}^2\right)$.

For backpropagation, the partial derivative of the total loss function $J$ with respect to the weight matrices can be derived by taking into account both the loss $L$ and the regularization term $s$. Gradients of $J$ with respect to the output layer's activation $\mathbf{o}$ and the regularization term $s$ are calculated as follows:

$$\frac{\partial J}{\partial \mathbf{o}} = \frac{\partial L}{\partial \mathbf{o}},$$

For the partial derivatives of the regularization term $s$ with respect to the weight matrices:

The Frobenius norm of a matrix $\mathbf{W}$ is defined as the square root of the sum of the squares of its elements. Mathematically, $\|\mathbf{W}\|_{\mathrm{F}}^2 = \sum_{i,j} \mathbf{W}_{ij}^2$. When we differentiate this with respect to $\mathbf{W}$, we treat each element $\mathbf{W}_{ij}$ as an independent variable. Therefore, the derivative of $\|\mathbf{W}\|_{\mathrm{F}}^2$ with respect to $\mathbf{W}$ is simply $2\mathbf{W}$ (since the derivative of $x^2$ with respect to $x$ is $2x$).

So, the derivative of $s$ with respect to $\mathbf{W}^{(1)}$ is:

$$\frac{\partial s}{\partial \mathbf{W}^{(1)}} = \frac{\lambda}{2} \cdot 2\mathbf{W}^{(1)} = \lambda \mathbf{W}^{(1)}.$$

Similarily, the derivative of the Frobenius norm squared of $\mathbf{W}^{(2)}$ with respect to $\mathbf{W}^{(2)}$ is $2\mathbf{W}^{(2)}$. Hence, the derivative of $s$ with respect to $\mathbf{W}^{(2)}$ is:

$$\frac{\partial s}{\partial \mathbf{W}^{(2)}} = \frac{\lambda}{2} \cdot 2\mathbf{W}^{(2)} = \lambda \mathbf{W}^{(2)}.$$

Therefore, the gradient of the objective function with respect to the weights of the output layer $\mathbf{W}^{(2)}$ is given by:

$$\frac{\partial J}{\partial \mathbf{W}^{(2)}} = \frac{\partial J}{\partial \mathbf{o}} \mathbf{h}^\top + \lambda \mathbf{W}^{(2)}.$$

To propagate the gradient back to the weights of the hidden layer $\mathbf{W}^{(1)}$, we first compute the gradient with respect to the hidden layer outputs $\mathbf{h}$:

$$\frac{\partial J}{\partial \mathbf{h}} = {\mathbf{W}^{(2)}}^\top \frac{\partial J}{\partial \mathbf{o}}.$$

Then we compute the gradient with respect to the pre-activation $\mathbf{z}$:

$$\frac{\partial J}{\partial \mathbf{z}} = \frac{\partial J}{\partial \mathbf{h}} \odot \phi'(\mathbf{z}),$$

where $\odot$ denotes elementwise multiplication and $\phi'(\mathbf{z})$ is the derivative of the activation function.

Finally, we obtain the gradient with respect to the weights of the hidden layer $\mathbf{W}^{(1)}$:

$$\frac{\partial J}{\partial \mathbf{W}^{(1)}} = \frac{\partial J}{\partial \mathbf{z}} \mathbf{x}^\top + \lambda \mathbf{W}^{(1)}.$$

These gradients are used to update the weights of the network in the direction that minimizes the loss function, similar to the Newton Raphson method.

Generally, to train the deep learning model, we use *Stochastic Gradient Descent* (SGD) to minimize the loss function by iteratively update the model's parameters in the opposite

direction of the gradient of the objective function with respect to the parameters. The general update rule for a parameter $\theta$ (refers to all parameters of the model as a whole) at iteration $t$ is given by:

$$\theta_{t+1} = \theta_t - \eta_t \cdot \nabla_\theta J(\theta_t; x^{(i)}, y^{(i)}),$$

where: - $\theta_t$ is the parameter vector at iteration $t$, - $\eta_t$ is the learning rate at iteration $t$, - $\nabla_\theta J(\theta_t; x^{(i)}, y^{(i)})$ is the gradient of the objective function $J$ with respect to $\theta$ at iteration $t$, computed using a single data point (or a mini-batch) $(x^{(i)}, y^{(i)})$.

In our MLP model, SGD would update each weight matrix using the gradients computed during backpropagation:

For the weights of the output layer:

$$\mathbf{W}^{(2)}_{t+1} = \mathbf{W}^{(2)}_t - \eta_t \cdot \left( \frac{\partial J}{\partial \mathbf{W}^{(2)}} \right)_t,$$

For the weights of the hidden layer:

$$\mathbf{W}^{(1)}_{t+1} = \mathbf{W}^{(1)}_t - \eta_t \cdot \left( \frac{\partial J}{\partial \mathbf{W}^{(1)}} \right)_t,$$

These updates are applied until convergence, that is, until the changes in the loss function or the weights themselves are smaller than some pre-defined threshold, or a maximum number of iterations is reached. The learning rate $\eta_t$ can be constant or adaptively changed during training using various strategies like learning rate decay, momentum, or more sophisticated methods such as Adam (Kingma & Ba, 2014).

### 1.2.4 Convolution Neural Networks

The use of images necessitates the formation of spatial hierarchies, which are essential for creating patterns and motifs that can be identified by both the human eye and machine learning algorithms. This has led to the development of *Convolution Neural Networks* (CNNs), a type of neural network specifically designed for image data. CNNs were first introduced in (LeCun, Jackel, *et al.*, 1995) and have since revolutionized the field of computer vision. By encapsulating local spatial relationship and expanding it to wider regions, CNNs have

been able to accurately capture visual motifs. The computational efficiency, parallelization support on modern GPUs, has been a major factor in the widespread use of CNNs (Chetlur *et al.*, 2014).

#### 1.2.4.1 Convolution layer theory

Now let's construct the convolution layer of a CNN. We can imagine that when we are trying to detect an object in an image, it is reasonable to focus on the local relation between the object and its neighboring pixels rather than the precise location of the object. This principle suggests three operational guidelines:

1. In the earliest layers, the network should respond similarly to the same patch, regardless of where it appears in the image. This is known as **translation invariance**.

2. The earliest layers should focus on local regions, without considering the contents of the image in distant regions. This is the **locality principle**.

3. As we move deeper, the layers should be able to capture long-range relations of the image, similar to higher-level vision in nature.

We can demonstrate these three operational guidelines by using MLPs with two-dimensional images $\mathbf{X}$ as input. Hidden representations $\mathbf{H}$ are also two-dimensional tensors, which maintain the same dimension as $\mathbf{X}$. We define $[\mathbf{X}]_{i,j}$ as the pixel at location $(i, j)$ in the input image and $[\mathbf{H}]_{i,j}$ as the hidden representation of that pixel, respectively.

To ensure every hidden representation $[\mathbf{H}]_{i,j}$ is influenced by each input pixel, we transition from the usual weight matrices to fourth-order weight tensors $\mathsf{W}$. If $\mathbf{U}$ is the bias, the fully connected layer becomes: ($k, l$ and $a, b$ are reparameterization)

$$\begin{aligned}[\mathbf{H}]_{i,j} &= [\mathbf{U}]_{i,j} + \sum_k \sum_l [\mathsf{W}]_{i,j,k,l} [\mathbf{X}]_{k,l} \\ &= [\mathbf{U}]_{i,j} + \sum_a \sum_b [\mathsf{V}]_{i,j,a,b} [\mathbf{X}]_{i+a,j+b}\end{aligned}$$

Here we set $[\mathsf{V}]_{i,j,a,b} = [\mathsf{W}]_{i,j,i+a,j+b}$. The values of $a$ and $b$ range from both positive and negative numbers, covering the whole image. For any location $(i,\ j)$ in $[\mathbf{H}]_{i,j}$, its value is a summation over input pixels centered at $(i, j)$, weighted by $[\mathsf{V}]_{i,j,a,b}$. It is important to take

into account the computational complexity; A single layer mapping a $1000 \times 1000$ image to an identical hidden representation requires incredible $10^{12}$ parameters.

Building on the principle of translation invariance, a shift in the input $\mathbf{X}$ should equate to a corresponding shift in the hidden representation $\mathbf{H}$. This invariance implies that $\mathsf{V}$ and $\mathbf{U}$ are not dependent on the specific location $(i, j)$. Therefore, we can represent them as $[\mathsf{V}]_{i,j,a,b} = [\mathbf{V}]_{a,b}$ and $\mathbf{U}$ becomes a constant, which we'll denote as $u$. This simplifies our equation for $\mathbf{H}$ to:

$$[\mathbf{H}]_{i,j} = u + \sum_a \sum_b [\mathbf{V}]_{a,b} [\mathbf{X}]_{i+a,j+b}.$$

$[\mathbf{V}]_{a,b}$ requires significantly fewer parameters than $[\mathsf{V}]_{i,j,a,b}$ because it's not tied to a specific image location. Thus, the number of parameters drastically drops from the previously mentioned $10^{12}$ to a more manageable $4 \times 10^6$, given the range of $a, b$ is (-1000, 1000).

By invoking the principle of locality, we can limit the complexity of the previous computation, setting all $[\mathbf{V}]_{a,b} = 0$ for $|a| > \Delta$ or $|b| > \Delta$.

$$[\mathbf{H}]_{i,j} = u + \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} [\mathbf{V}]_{a,b} [\mathbf{X}]_{i+a,j+b}.$$

This constraint reduces our parameters from $4 \times 10^6$ to $4\Delta^2$, with $\Delta$ usually less than 10. Hence we have cut down parameters by four orders of magnitude, leading to the *convolution layer* $[\mathbf{H}]$. Networks containing such layers are termed Convolution Neural Networks (CNNs), with $\mathbf{V}$ named as *convolution kernel*, *convolution filter*, or simply the weights. By significantly reducing the number of parameters, the fixed convolution kernels in a sliding window manner throughout all pixels of an image imparts CNNs with the property of translation invariance. This is because each kernel is limited to learning local patterns and is not affected by the specific location of these patterns within the image space. The reduction of parameters also facilitates the creation of more complex network architectures.

#### 1.2.4.2 Construction of the convolution layer

We will now build convolution neural network using an example of image data. Image data are typically represented in a three-dimensional format, comprising height, width, and color channels. The first two dimensions correspond to the spatial resolution of the image

(size of the image in pixels), while the depth is a 3 dimensional vector that represents color channels such as red, green, and blue in a standard RGB image. For grayscale images, the depth is just one number, since it only captures varying shades of gray.

**2-D convolution layer** Ignoring color channels for now and just considering 2-D image data with 1 channel we can observe how the convolution layer works. In Figure 1.2, the input is a two-dimensional $3 \times 3$ (or (3,3)) tensor with a height equal to 3 pixels and a width equal to 3 pixels. Since the height and width of the kernel are 2 pixels, the shape of the *kernel window* (or *convolution window*) is $2 \times 2$. In actual settings, the values will be real numbers instead of integers, and the kernel tensors will be parameter of the model which will be updated while training the model.

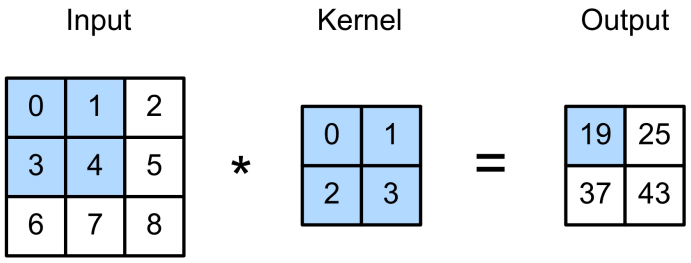



Figure 1.2: Two-dimensional convolution operation. The areas that are shaded represent the initial output element, as well as the input and kernel tensors that are used to calculate the output.: $0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$
. Kernel will be the parameter to be updated while training.

We start by placing the convolution window in the upper-left corner of the input tensor and then move it across the input tensor, both horizontally and vertically. At each position, the input tensor within the window and the kernel tensor are multiplied element by element and the resulting tensor is summed up to give a single scalar value. This result is the value of the output tensor at the corresponding location. The output tensor has a height of 2 and width of 2 and the four elements are obtained from the two-dimensional *cross-correlation* operation.

*Remarks: Actually the operation of the kernel in convolution neural network is cross-correlation instead of convolution as defined in mathematics.*

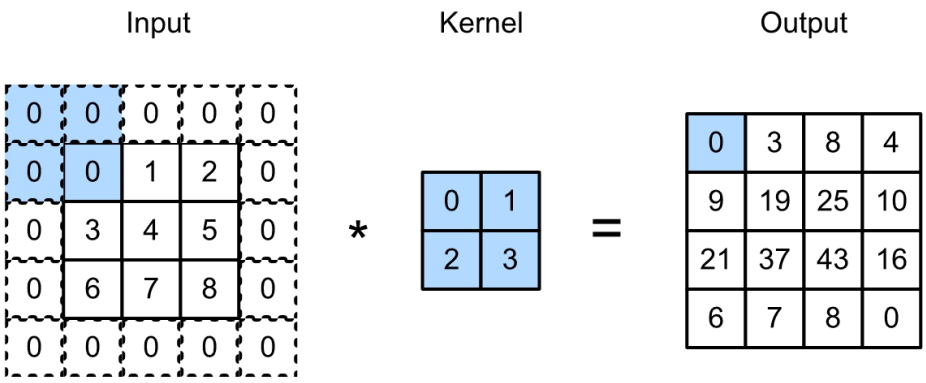



Figure 1.3: Two-dimensional convolution layer with padding

$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19,$$
$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25,$$
$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37,$$
$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43$$

The size of the output tensor is slightly smaller than the input size along each axis because we can only accurately compute the cross-correlation when the kernel fits completely within the image. In general, the size of the output tensor is given by the input size $n_\text{h} \times n_\text{w}$ minus the size of the convolution kernel $k_\text{h} \times k_\text{w}$: $(n_\text{h} - k_\text{h} + 1) \times (n_\text{w} - k_\text{w} + 1)$. When $k_\text{h}$ and $k_\text{w}$ are bigger than 1, the output layer size will be smaller than the input layer.

**Padding** Padding can be used to address the problem that the output size is smaller than the input size. Without padding, each convolution layer is smaller than the previous layer and bigger than the next layer. This is undesirable, as we wish to maintain the same dimension in each convolution layer. Padding will add extra pixels around the boundary of our input image, thus increasing the effective size of the image. Typically, one assigns a value of zero to the extra pixels in the boundary of the input layer.

In Figure 1.3, we add padding to a $3 \times 3$ input (adding rows and columns of zeros), increasing its size to $5 \times 5$. The output then grows to a $4 \times 4$ tensor. The shaded parts are the first output element and the input and kernel tensor elements used for the output

calculation: $0 \times 0 + 0 \times 1 + 0 \times 2 + 0 \times 3 = 0$. Generally, if we add a combined total of $p_{\mathrm{h}}$ rows of padding, with approximately half of them at the top and the other half at the bottom, and a combined total of $p_{\mathrm{w}}$ columns of padding, with approximately half of them on the left and the other half on the right, the output shape will be:

$$(n_{\mathrm{h}} - k_{\mathrm{h}} + p_{\mathrm{h}} + 1) \times (n_{\mathrm{w}} - k_{\mathrm{w}} + p_{\mathrm{w}} + 1)$$

In many cases, we want to set $p_{\mathrm{h}}$ to $k_{\mathrm{h}} - 1$ and $p_{\mathrm{w}}$ to $k_{\mathrm{w}} - 1$ in order to make the input and output have the same height and width, especially when constructing a deep neural network.

**Convolution kernel size** CNNs often employ kernels with odd width and height values, such as 1, 3, 5, or 7. This practice has the advantage of maintaining the same dimensions when padding the same number of rows on the top and bottom, and the same number of columns on the left and right. An odd-sized kernel has a central pixel, making it symmetric around this point. This symmetry allows for a natural anchor point for the convolution operation, ensuring that each convolution operation has a well-defined spatial relationship to the input. In other words, each output pixel is a transformed representation of a neighborhood centered on a corresponding input pixel. With an even-sized kernel, there's no central pixel, complicating this spatial relationship.

Additionally, this approach of using odd kernels and padding to precisely maintain the same dimensions offers a practical benefit. For any two-dimensional tensor $X$, when the kernel size is odd and the number of columns and rows padding on all sides is equal, the output $Y[i, j]$ is calculated by cross-correlating the input and convolution kernel with the window centered on $X[i, j]$, resulting in an output with the same height and width as the input. If the kernel size is an even number, applying padding with an equal number of rows on the top and bottom, and an equal number of columns on the left and right, will not result in the output layer having the same dimensions as the input layer.

**Stride** We refer to the number of rows and columns traversed per slide as stride. So far, we have used strides of 1, both for height and width. Sometimes, we may want to use a larger stride. Convolutions with a larger stride are a popular technique that can help reducing the

dimensionality of the input layer drastically, either for computational efficiency or because we wish to downsample.

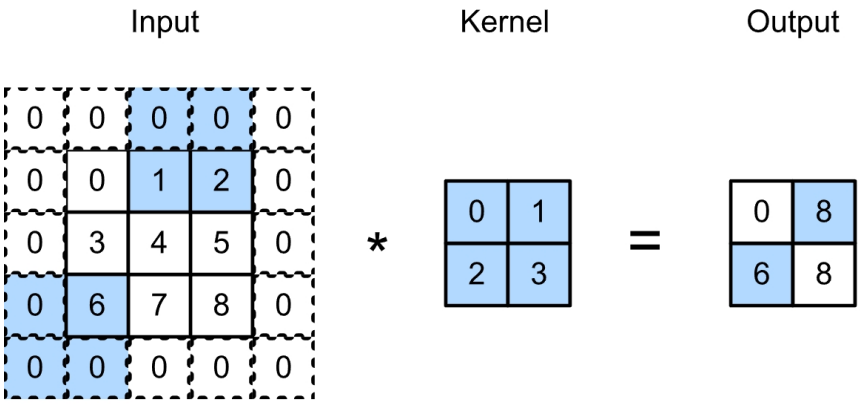



Figure 1.4: Convolution layer with strides of 3 and 2 for height and width, respectively.

Figure 1.4 illustrates a two-dimensional convolution layer with a stride of 3 vertically and 2 horizontally. The shaded areas represent the output elements as well as the input and kernel tensor elements used for the output computation: $0 \times 0 + 0 \times 1 + 1 \times 2 + 2 \times 3 = 8$, $0 \times 0 + 6 \times 1 + 0 \times 2 + 0 \times 3 = 6$.

The example shows that when the convolution produces the second element of the first column, the convolution window shifts down three rows. When the second element of the first row is created, the convolution window moves two columns to the right. Notice that in the absence of an additional column of padding, the convolution window is unable to calculate the third column due to insufficient column space.

In general, when the stride for the height is $s_\mathrm{h}$ and the stride for the width is $s_\mathrm{w}$, the output shape is

$$\lfloor (n_\mathrm{h} - k_\mathrm{h} + p_\mathrm{h} + s_\mathrm{h})/s_\mathrm{h} \rfloor \times \lfloor (n_\mathrm{w} - k_\mathrm{w} + p_\mathrm{w} + s_\mathrm{w})/s_\mathrm{w} \rfloor.$$

If the height and width of the input tensor are divisible by the strides on the height and width, then the output shape will be $(n_\mathrm{h}/s_\mathrm{h}) \times (n_\mathrm{w}/s_\mathrm{w})$.

**Multiple channels** Channels can be generalized not only for colors, but as a hyperparameter for the CNN models. When the input data has multiple channels, we must create a convolution kernel with the same number of input channels. This allows us to perform the

convolution operation with the input data. For each channel, we can carry out a convolution operation on the two-dimensional tensor of the input and the two-dimensional tensor of the convolution kernel, and then add all the results together (summing over the channels) to produce a two-dimensional tensor. Figure 1.5 provides an example of a two-dimensional cross-correlation with two input channels. The shaded portions are the first output element as well as the input and kernel tensor elements used for the output computation:

$$(1 \times 1 + 2 \times 2 + 4 \times 3 + 5 \times 4) + (0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3) = 56$$

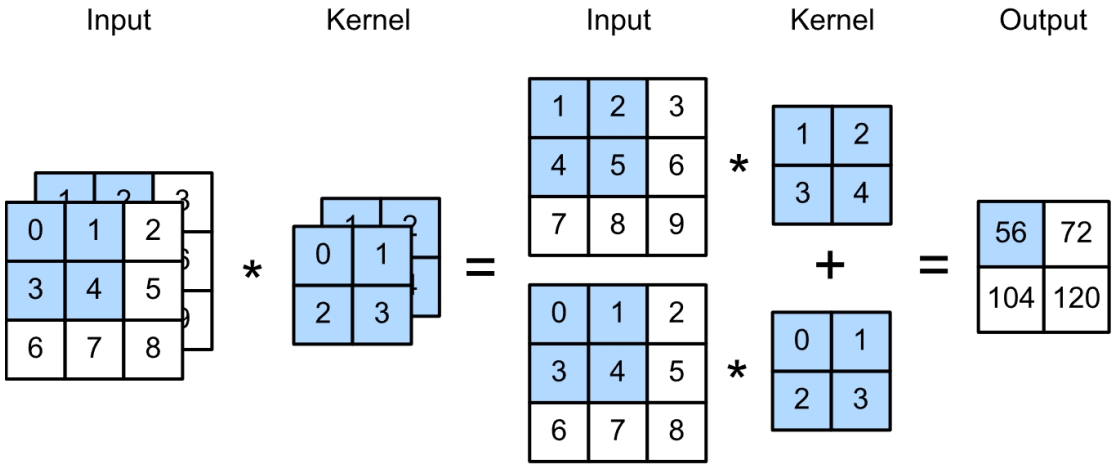



Figure 1.5: Convolution layer with two input channels

Let $c_\mathrm{i}$ and $c_\mathrm{o}$ denote the number of input and output channels, respectively, and $k_\mathrm{h}$ and $k_\mathrm{w}$ the height and width of the kernel. To generate an output with multiple channels, we can create a kernel tensor of shape $c_\mathrm{i} \times k_\mathrm{h} \times k_\mathrm{w}$ for each output channel, resulting in a $c_o$ number of $k_\mathrm{h} \times k_\mathrm{w}$-shaped outputs. We then stack those outputs in the output channel dimension, resulting in a $c_o \times k_\mathrm{h} \times k_\mathrm{w}$-shaped output. Hence, those $c_o$ independent kernels of shape $c_\mathrm{i} \times k_\mathrm{h} \times k_\mathrm{w}$ can be treated as a convolution kernel of shape $c_\mathrm{o} \times c_\mathrm{i} \times k_\mathrm{h} \times k_\mathrm{w}$. During convolution operations, the result for each output channel $i \in \{1, 2, ..., c_o\}$ is calculated from the $i$-th convolution kernel associated with $i$-th output channel.

Channels enable us to take advantage of both MLPs (significant nonlinearities) and convolution (localized analysis of features). Channels allow processing multiple aspects of input data, like edges and shapes, in parallel. Furthermore, channels offer a strategic compromise:

they preserve the model's complexity while still capitalizing on the efficiency gains from the translation invariance and locality principles of convolutions. This balance ensures that CNNs remain both computationally efficient and functionally powerful.

**Maximum Pooling and Average Pooling** *Pooling layers* are another important part of CNN, which are composed of a fixed-sized window that is moved across all regions of the input, with a stride size, to produce a single output for each location it passes through (also known as the pooling window). Unlike convolution layers, pooling layers have no parameters (no kernel). Instead, they are deterministic, usually calculating either the maximum or the average value of the elements in the pooling window. These operations are called maximum pooling (max-pooling) and average pooling, respectively. Average pooling is a technique that has been used since the inception of CNN. It is similar to downsampling an image, where instead of taking the value of every second (or third) pixel for the lower resolution image, the adjacent pixels are averaged to obtain an image with better signal-to-noise ratio. Max-pooling (Figure 1.6) was introduced in (Riesenhuber & Poggio, 1999) in the context of cognitive neuroscience to explain how information can be aggregated hierarchically for object recognition. In most cases, max-pooling is preferred over average pooling. Pooling layers, similar to convolution layers, alter the output dimension. We can modify the operation to get the desired output shape by padding the input and adjusting the stride.

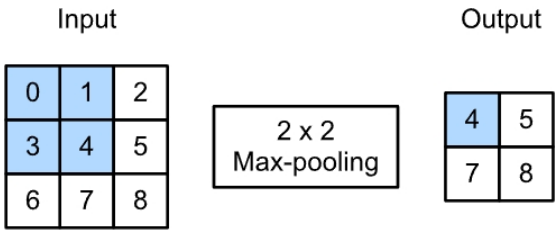



Figure 1.6: Max-pooling with a pooling window shape of 2 by 2. The shaded portions are the first output element as well as the input tensor elements used for the output computation.

## 1.3 Modern Convolution neural network designs

We're now equipped with the essential components to construct a comprehensive CNN. In this section, we'll start with the foundational LeNet, progressing through the groundbreaking AlexNet and ResNet, and culminating in the cutting-edge designs of modern convolution

neural networks.

### 1.3.1 LeNet

Yann LeCun's LeNet was the first convolution Neural Network (CNN) to gain widespread attention for its performance on computer vision tasks. It was designed to recognize handwritten digits in images (LeCun, Bottou, *et al.,* 1998). At the time, LeNet outperformed the then-dominant approach of Support Vector Machines (SVMs), achieving an error rate of less than 1% per digit. Figure 1.7 illustrate the design of LeNet.

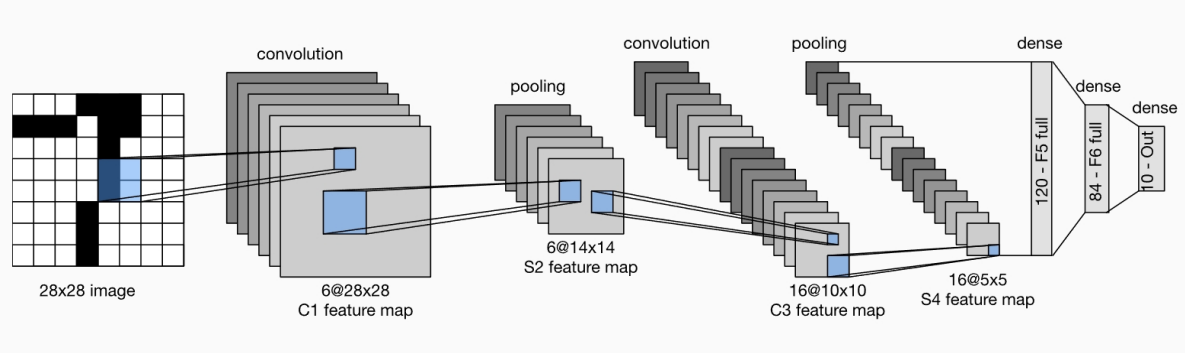



Figure 1.7: Model design of LeNet. The input is a handwritten digit, and the output is a probability of the digit belonging to one of the ten possible outcomes.

The convolution block consists of convolution layers, sigmoid activation functions, and average pooling operations. Each convolution layer uses a $5 \times 5$ kernel and a sigmoid activation function after the convolution layer to map spatially arranged inputs to a number of two-dimensional *feature maps* (output of a previous layer in the network), usually increasing the number of channels. The first convolution layer has 6 output channels, while the second has 16. Each $2 \times 2$ pooling operation (stride 2) reduces the dimensionality by a factor of 4 through spatial downsampling. The output of the convolution block has a shape of (batch size, number of channel, height, width). Note that ReLUs and max-pooling, which are more effective, had not yet been discovered at the time.

In order to pass the output from the convolution block to the dense block, we must flatten each example in the minibatch. This means transforming the four-dimensional input into a two-dimensional input that is expected by fully connected layers. The two-dimensional representation has the first dimension to index examples in the minibatch and the second to give the flat vector representation of each example. LeNet's dense block has three fully

connected layers, with 120, 84, and 10 outputs, respectively. The 10-dimensional output layer is the number of possible output classes for the classification task.

### 1.3.2 AlexNet

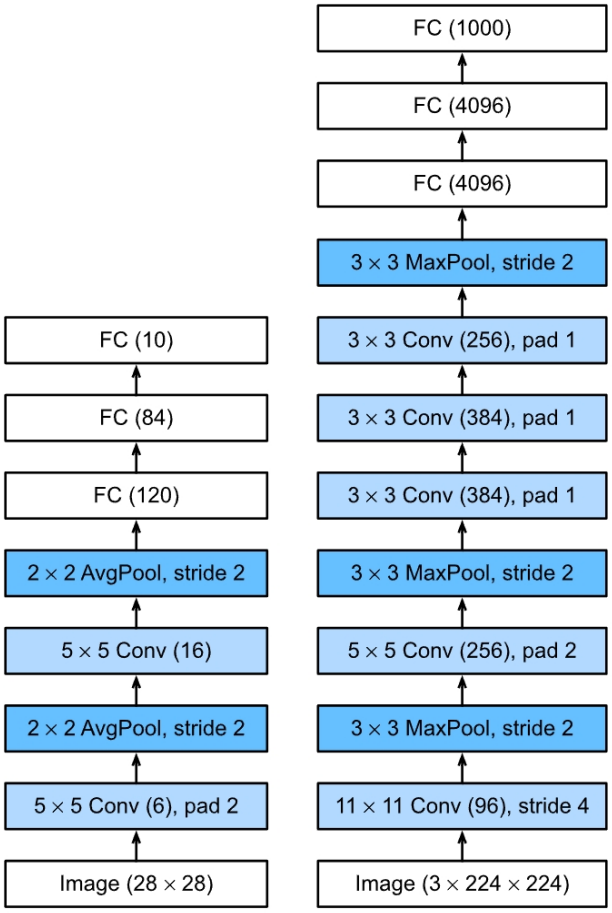



Figure 1.8: From LeNet (left) to AlexNet (right). Dark blue represent max pooling layer, light blue represent convolution layer. Pad is padding. Parentheses indicate the channel number.

AlexNet (Krizhevsky *et al.*, 2012), an 8-layer CNN, was a game-changer in the ImageNet Large Scale Visual Recognition Challenge 2012, outperforming all other competitors by a wide margin. This network demonstrated, for the first time, that features obtained through learning can surpass those designed manually, thus overturning the existing paradigm in computer vision. The architectures of AlexNet and LeNet are quite alike, as demonstrated in Figure 1.8, but there are also significant differences between AlexNet and LeNet:

1. AlexNet is a much deeper network than LeNet-5, with eight layers: five convolution layers, two fully connected hidden layers, and one fully connected output layer. The

convolution kernel shape of the first layer is $11 \times 11$, as the images in ImageNet are eight times larger in length and width. Second layer shape is reduced to $5 \times 5$, then followed by $3 \times 3$. After the first, second, and fifth convolution layers, the network adds max-pooling layers with a window shape of $3 \times 3$ and a stride of 2. AlexNet also has ten times more convolution channels than LeNet (as indicated in parentheses). The final fully connected layers of AlexNet also contains much more parameters than the counterpart of LeNet.

2. AlexNet utilized the ReLU activation function in place of the sigmoid due to its simpler computation, which does not require the exponentiation operation found in the sigmoid activation function. Furthermore, the ReLU activation function makes model training simpler when using different parameter initialization techniques. If the model parameters are not initialized correctly, the sigmoid function may have a gradient close to 0, making it difficult to train the model.

3. AlexNet controls the complexity of the fully connected layer by using dropout (Bishop, 1995), in contrast to LeNet which only applies weight decay (Krogh & Hertz, 1991). Furthermore, the AlexNet training loop incorporates image augmentation techniques, such as flipping, clipping, and color changes, to further increase the data set. This makes the model more reliable and the larger sample size helps to prevent overfitting.

### 1.3.3 Block design in VGG

AlexNet provided evidence that deep convolution neural networks (CNNs) can be successful, yet it did not provide a general template for future researchers to use when designing new networks. The evolution of neural network architectures has, in many ways, mirrored advancements in chip design. Just as chip engineers transitioned from manually arranging transistors to employing logical blocks, neural network design too has seen a transition from a focus on individual neurons to layers, and now, to "blocks" — repeating sequences of layers with predefined patterns. This modular approach was popularized by the Visual Geometry Group (VGG) at Oxford University in their namesake VGG network (Simonyan & Zisserman, 2014).

Before VGG, the basic unit encompassed a sequence of a convolution layer (often with padding to preserve resolution), a nonlinear activation like ReLU, followed by a pooling layer for resolution reduction. VGG's groundbreaking insight was to introduce blocks, where multiple convolution layers were stacked before a downsampling step. This approach allowed for deeper and narrower networks. For example, two successive $3 \times 3$ convolutions would cover the same area as one $5 \times 5$ convolution, but with fewer parameters. Research conducted by VGG showed that deep and narrow networks are more effective than shallower architectures. This led to the development of deep networks with more than 100 layers, and the use of $3 \times 3$ convolutions became a standard design in subsequent deep learning models. The diagram in Figure 1.9 shows how VGG used a block design to create a much deeper network than AlexNet.

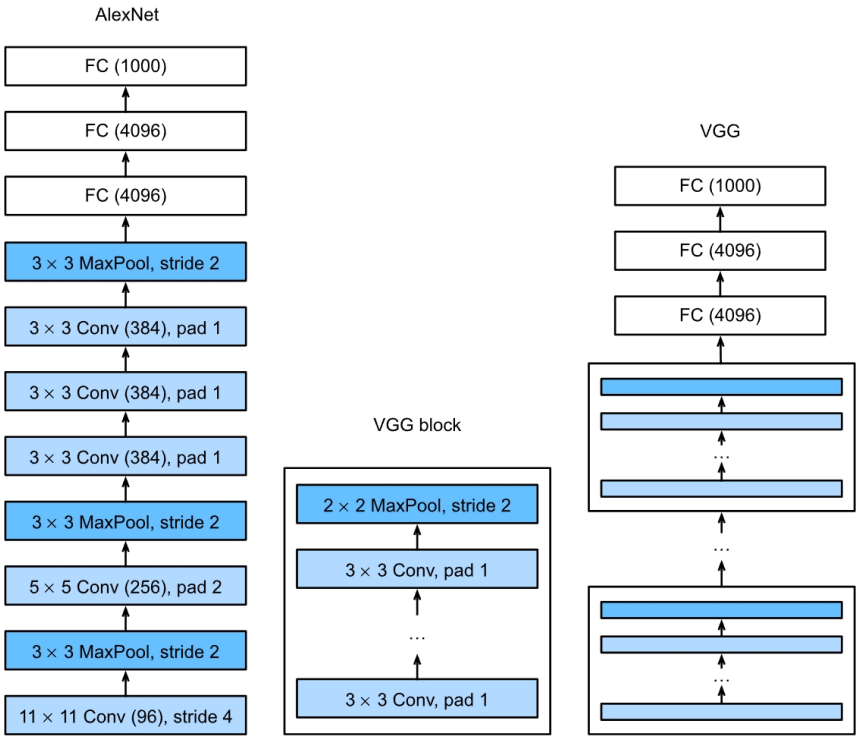



Figure 1.9: From AlexNet to VGG. VGG consists of blocks of layers, whereas AlexNet's layers are all designed individually.

### 1.3.4 Network in Network: 1 by 1 convolution and Global average pooling layer

LeNet, AlexNet, and VGG have a shared design pattern: they extract features through a series of convolutions and pooling layers, followed by post-processing via fully connected layers. Two major challenges are presented by this design. First, the fully connected layers at the end of the architecture require a large number of parameters. Second, it is not possible to add fully connected layers earlier in the network to increase the degree of nonlinearity, which will require even more memory. The network in network (NiN) blocks (M. Lin *et al.*, 2013) offer a solution to both of these problems. This strategy is based on two simple ideas:

1. **Use $1 \times 1$ convolutions to add local non-linearities across channel activations.** In fact, the convolution in CNN essentially involves operations between multi-channel feature maps (the number of channels is the same as the number of filters) and multi-channel convolution kernels. If a 1x1 convolution kernel is used, the resulting value will be independent of the surrounding pixels. Therefore, this operation realizes a linear combination of multiple feature maps, achieving a change in the number of channels in the feature map. If not use the $1 \times 1$ convolution kernel, the only way is fully connected layer, and that will add much more parameters. Cascading multiple 1x1 convolution kernels can implement nonlinear combinations of multi-channel feature maps. Moreover, a 1x1 convolution kernel can be used to **diminish the number of channels and enlarge** the kernel size, with fewer parameters than using fully connected layers.

2. **Leverage Global Average Pooling (GAP) layer to emphasize spatial hierarchies and category relationships with much fewer parameters.** In the context of CNNs, GAP serves as a strategic substitute for traditional fully connected layers. The technique involves generating a feature map for each specific category within the last convolution layer and then computing the average of every feature map, forming a vector in the same shape. This vector, encompassing the averages, is directly integrated into the softmax layer. The primary advantages of employing GAP are manifold. Firstly, GAP is more native to the convolution structure to FC layer by enforcing

correspondences between feature maps and categories. Thus the feature maps can be easily interpreted as categories confidence maps. Secondly, GAP's design eliminates the need for optimizable parameters, inherently **reducing overfitting risks**. Finally, by aggregating spatial information, GAP enhances the model's **resilience to spatial variations**, making the neural network more adaptable and robust.

The implementation of both $1\times1$ convolutions and global average pooling had a significant influence on the advancement of subsequent convolution neural networks.

### 1.3.5 Batch normalization

Batch Normalization (BatchNorm) is a groundbreaking approach in deep learning, designed to make the training of deep networks simpler and faster. Proposed by Ioffe and Szegedy (Ioffe & Szegedy, 2015), its primary objective is to mitigate the challenges associated with the *internal covariate shift*. Internal Covariate Shift refers to the phenomenon in deep learning where the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. The inception of BatchNorm has revolutionized the realm of deep learning, rendering the training of networks with more than 100 layers significantly more tractable. At its core, this normalization acts synergistically with optimization algorithms, inherently aligning the scale of parameters.

The fundamental operation behind batch normalization is the normalization of layer activations. Specifically, for every mini-batch, it standardizes the activations of the previous layer, meaning it makes the activations have zero mean and unit variance. The technique then introduces two learnable parameters, scale and shift (usually represented by $\gamma$ and $\beta$ respectively), for each activation. The purpose of these parameters is to allow the model to adjust the normalized activations as needed, preserving the network's ability to represent a wide range of functions and effectively learn from the data.

Mathematically, given a mini-batch of activations $X$, the batch normalization operation can be denoted as:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i \quad \text{(Compute the mean of the mini-batch)}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)^2 \quad \text{(Compute the variance of the mini-batch)}$$

$$\hat{x_i} = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \quad \text{(Normalize the activations)}$$

$$y_i = \gamma \hat{x_i} + \beta \quad \text{(Scale and shift)}$$

Here, $\epsilon$ is a small constant added for numerical stability.

By consistently normalizing activations throughout the training process, batch normalization offers several key benefits:

- **Accelerated Training:** It allows for the use of higher learning rates, as the risk of divergence is reduced.
- **Regularization Effect:** The noise introduced by normalizing activations acts as a form of regularization, reducing the need for other regularization techniques such as dropout.
- **Reduced Sensitivity to Initialization:** With BatchNorm, initial weights become less critical, mitigating the challenges of initializing very deep networks.

During the prediction phase, it is not possible to use the mean and variance of the current batch for normalization because we predict for a single data point. Instead, we use the moving averages of the mean and variance accumulated during training. These stored statistics, treated as constants, provide a consistent basis for normalizing the activations. Afterward, the learned scaling (gamma) and shifting (beta) parameters from the training phase are applied to adjust the normalized activations, speeding up the convergence of the model training. When we want to apply trained model for inference, these accumulated statistics and learned parameters will be applied to BN layer, hence guarantee stable and consistent predictions for inference data.

### 1.3.6 ResNet and ResNext

ResNets (K. He *et al.,* 2016a), also known as Residual Networks, have revolutionized the design of deep neural architectures, providing solutions to the difficulties encountered when training very deep networks. The core motivation behind ResNets is the observation that as networks grow deeper, their ability to learn can be hampered, often leading to a degradation problem where adding more layers results in a higher training error rate. ResNets introduce "skip" or "shortcut" connections that bypass one or more layers. The key insight is to allow the network to learn *residual mapping*s, which are the differences between the input and the desired output, rather than attempting to learn the desired mapping directly.

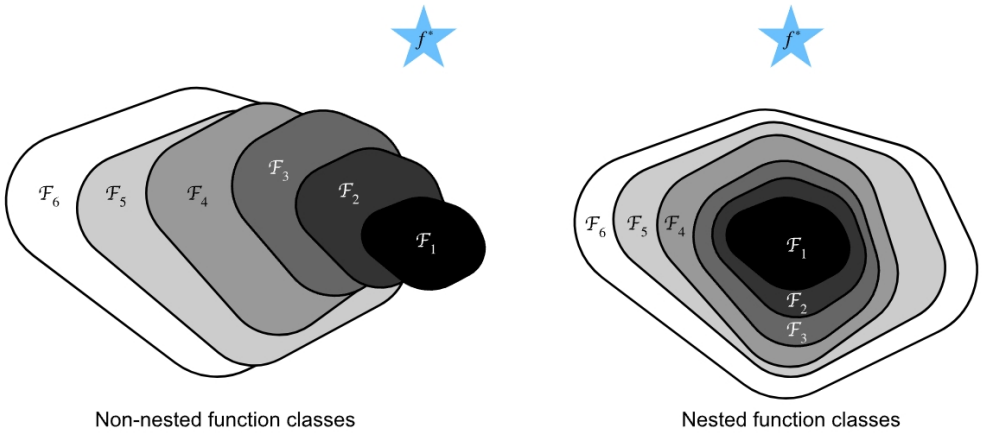



Figure 1.10: For non-nested function classes, a larger area does not guarantee that we will be nearer to the "truth" function ($f^*$). This is not the case with nested function classes.

**Nested function** To understand the power and efficacy of residual bypass, we first consider the class of functions, $\mathcal{F}$, that a specific network architecture, with its hyperparameters, can attain. For every $f \in \mathcal{F}$, there's a corresponding set of parameters (e.g., weights and biases) acquirable via training. The ultimate aspiration is the true function $f^*$. If $f^*$ is part of $\mathcal{F}$, we're optimally positioned, but such scenarios are rare. More commonly, the goal is to approximate $f^*$ with $f^*_{\mathcal{F}}$, the best within $\mathcal{F}$. This can be mathematically represented as: ($L$ is the loss function)

$$f^*_{\mathcal{F}} \stackrel{\text{def}}{=} \underset{f}{\operatorname{argmin}}\, L(\mathbf{X}, \mathbf{y}, f) \text{ subject to } f \in \mathcal{F}.$$

Ideally, a superior architecture, $\mathcal{F}'$, should yield better results. However, if $\mathcal{F} \not\subseteq \mathcal{F}'$, $f^*_{\mathcal{F}'}$

might not be better than $f^*_{\mathcal{F}}$. As delineated in Figure 1.10, unless larger function classes encapsulate smaller ones, there is no assurance of increased network performance. Here $F_i$ with larger $i$ represent the function space of the $i$-th layer.

**Residual block** Deep neural networks benefit when newly added layers can easily mimic an identity function $f(\mathbf{x}) = \mathbf{x}$, preserving the efficacy of the original model while potentially reducing training errors. The Figure 1.11 the right pannel shows the residual block of ResNet, with a solid line carrying the layer input to the addition operator referred to as a residual connection (or shortcut connection).

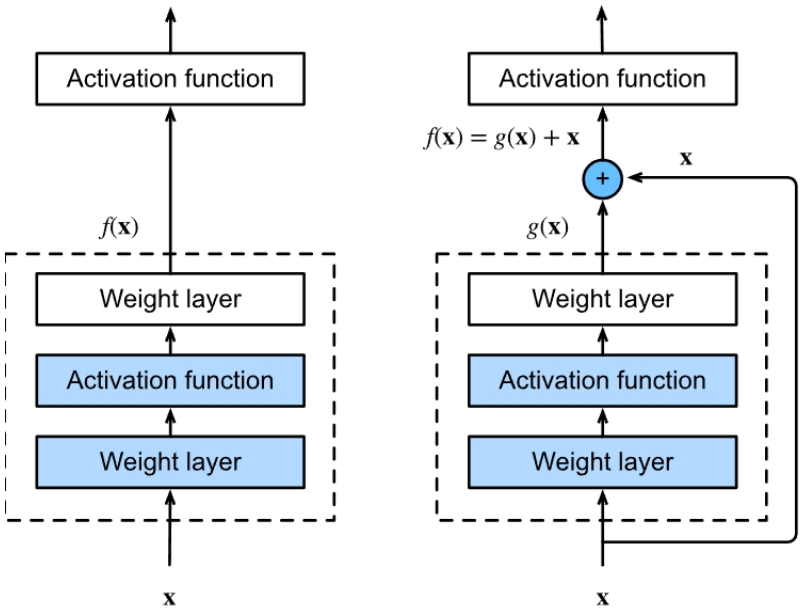



Figure 1.11: In a regular block (left panel), the network within the dotted-line box must learn the mapping $f(\mathbf{x})$ directly. On the other hand, in a residual block (right panel), the network within the dotted-line box has to learn the residual mapping $g(\mathbf{x}) = f(\mathbf{x}) - \mathbf{x}$, which makes it simpler to learn the identity mapping $f(\mathbf{x}) = \mathbf{x}$.

ResNet has a full $3 \times 3$ convolution layer design, similar to VGG. The residual block consists of two $3 \times 3$ convolution layers with the same number of output channels. Each convolution layer is followed by a batch normalization layer and a ReLU activation function. The two convolution operations are then bypassed and the input is added directly before the final ReLU activation function. This design necessitates that the output of the two convolution layers must be the same size as the input, so that they can be combined. To alter the number of channels, an extra $1 \times 1$ convolution layer must be used to transform

the input into the desired shape for the combination, as shown in Figure 1.12.

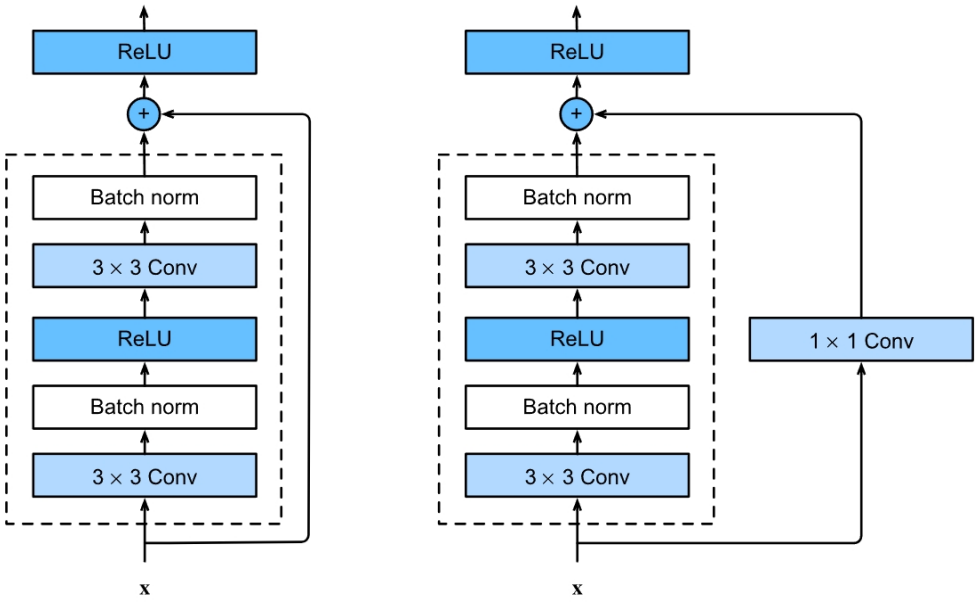



Figure 1.12: ResNet block with and without $1 \times 1$ convolution, which transforms the input into the desired shape for the addition operation.

**ResNet Model** ResNet stands out from other architectures because the design of its initial layers. While the inception model includes a $7 \times 7$ convolution layer with 64 output channels (stride of 2) followed by a $3 \times 3$ max-pooling layer (stride of 2) like other networks, ResNet adds batch normalization after each convolution. In particular, the core of ResNet is made up of four structured modules, each composed of identical residual blocks with the same output channel numbers. The first module keeps the channel number from the input, since the prior max-pooling layer has already reduced the dimensionality. The following modules have a pattern: the first residual block in each module doubles the channels from the previous one, while reducing the height and width by half. The network ends with a global average pooling layer and a final fully connected layer. Excluding the convolution layers $1 \times 1$, the model, with its initial convolution $7 \times 7$ and last fully connected layer, has 18 layers, thus being called ResNet-18.

By adjusting the number of channels and residual blocks in the module, we can construct various ResNet models, such as the 152-layer ResNet-152. Compared to VGG, ResNet has a simpler and more flexible structure. This has led to its rapid and widespread adoption. Figure 1.13 shows the full ResNet-18.

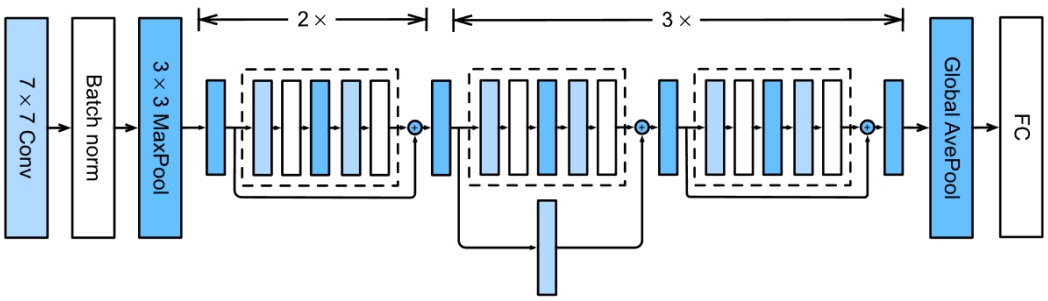



Figure 1.13: Architecture of ResNet-18.

**ResNeXt Model** The design of ResNet presents a challenge in terms of the trade-off between non-linearity and dimensionality within a given block. ResNeXt (S. Xie *et al.*, 2017) is an improved version of ResNet that seeks to optimize deep learning performance. The name "ResNeXt" stands for "ResNet with Next-Level aggregations," and it is distinguished from its predecessor by its *cardinality*, which is the number of parallel paths or transformations within a block. Rather than simply increasing the depth or width of the networks, ResNeXt emphasizes the use of multiple pathways to increase the representational power. This approach offers competitive or superior performance to other architectures with relatively efficient computational resources. Additionally, the modular design of ResNeXt makes it easier to scale and adapt, making it a key step forward in the development of deep learning architectures.

Figure 1.14 shows the ResNeXt block with $g$ groups, which are called grouped convolution. The basic idea is to fragment a convolution from the $c_\mathrm{i}$ to $c_\mathrm{o}$ channels into $g$ groups. Each group is of size $c_\mathrm{i}/g$ and produces $g$ outputs of size $c_\mathrm{o}/g$. This adjustment leads to a reduction in computational costs by a factor of $g$. More specifically, the computational burden drops from $\mathcal{O}(c_\mathrm{i} \cdot c_\mathrm{o})$ to $\mathcal{O}(c_\mathrm{i} \cdot c_\mathrm{o}/g)$. Furthermore, this methodology results in a reduction in parameters for convolution, effectively decreasing the required parameters by a factor of $g$.

However, a limitation arises from this partitioning: the groups operate in isolation, with no intergroup information sharing. The design of ResNeXt addresses this limitation by integrating the grouped convolution with $3 \times 3$ kernels between two $1 \times 1$ convolutions. The subsequent $1 \times 1$ convolution both corrects the number of channels and ensures cross-group communication. This strategy strikes a balance between computational efficiency and representational power, where the major computational load is shouldered by the $1 \times 1$

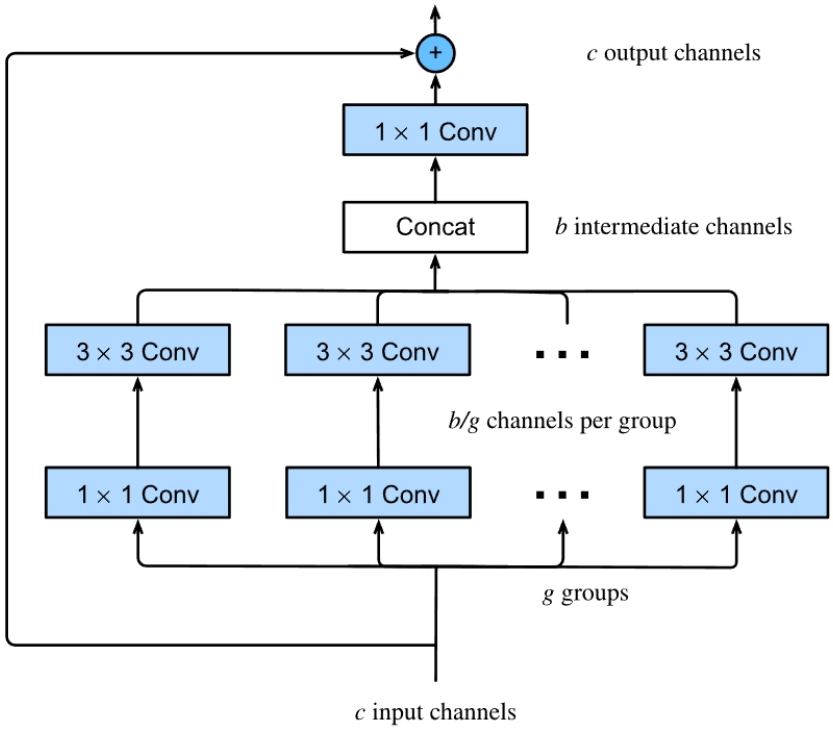



Figure 1.14: Architecture of ResNeXt block with $g$ groups.

convolutions, leaving the grouped $3 \times 3$ kernels to enhance the network capacity without an excessive computational penalty. As depicted in Figure 1.14, the conventional residual connection in ResNet is generalized in ResNeXt with a $1 \times 1$ convolution, further enriching the model's expressiveness.

## 1.4 Recurrent Neural Networks

Many tasks need to process sequential data. This is especially evident in tasks such as time series forecasting, video content analysis, and the extraction of musical patterns. Sequential data is not limited to these areas, in biology ordered structure of genomic sequences, the intricate sequence-to-structure relationships in protein folding, and the time-dependent patterns in ecological series all emphasize the importance of sequences. It is essential for a model to be able to handle such sequential inputs effectively, recognizing the inherent order and patterns.

Recurrent Neural Networks (RNNs) are deep learning models that use recurrent connections to capture the dynamics of sequences. Unlike most feed-forward neural networks,

which process data in a unidirectional manner, RNNs maintain a loop mechanism that allows persistence of information. This cyclical structure endows RNNs with the capability of remembering prior information and utilizing it in current computations, making them particularly useful for time series forecasting, natural language processing, and other tasks where temporal dynamics and sequence order are pivotal. As Figure 1.15 shows, RNNs can be thought of as feedforward neural networks where the parameters of each layer (both conventional and recurrent) are shared across time steps.

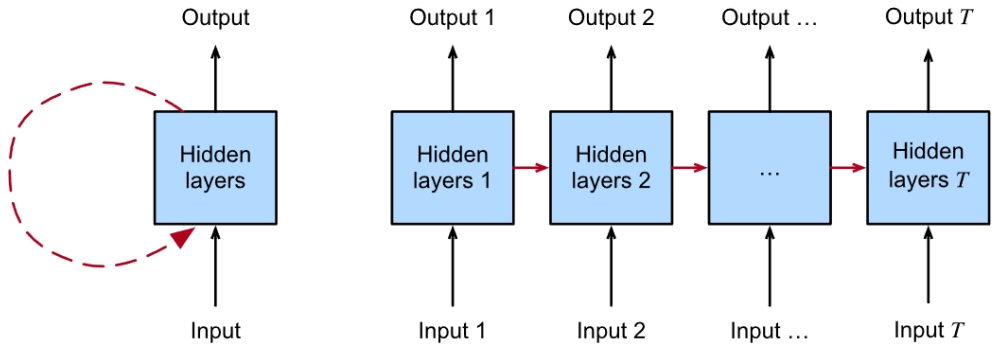



Figure 1.15: The illustration on the left depicts recurrent connections through cyclic edges. The diagram on the right unfolds the RNN across a sequence of time steps, where recurrent edges connect adjacent time steps.

### 1.4.1 RNN

Recurrent neural networks (RNNs) (Elman, 1990) are neural networks with hidden states. Consider a given time step, $t$, with a minibatch of inputs denoted as $\mathbf{X}_t \in \mathbb{R}^{n \times d}$. For a minibatch comprising $n$ sequential samples, each row in $\mathbf{X}_t$ represents a sample at time step $t$. The hidden layer output at this time step is symbolized by $\mathbf{H}_t \in \mathbb{R}^{n \times h}$.

In contrast to traditional MLP, in RNNs, we retain the hidden layer output, $\mathbf{H}_{t-1}$, from the preceding time step. This approach requires the introduction of an additional weight parameter, $\mathbf{W}_{\text{hh}} \in \mathbb{R}^{h \times h}$, which illustrates how the output from the previous hidden layer is used in the current time step. Formally, the hidden layer's output at the current step is computed as:

$$\mathbf{H}_t = \phi(\mathbf{X}_t \mathbf{W}_{\text{xh}} + \mathbf{H}_{t-1} \mathbf{W}_{\text{hh}} + \mathbf{b}_{\text{h}}).$$

This equation highlights an important part of the RNN: the current hidden state, $\mathbf{H}_t$, is

a function of both the current input and the hidden state from the previous time step, $\mathbf{H}_{t-1}$. This dependency ensures that $\mathbf{H}_t$ encapsulates historical information from the sequential data up to the current step, effectively acting as the network's memory or state. Such an output, due to its dependency on prior states, is aptly termed a **hidden state**. Layers within RNNs that execute this recurrent computation are labeled **recurrent layers**.

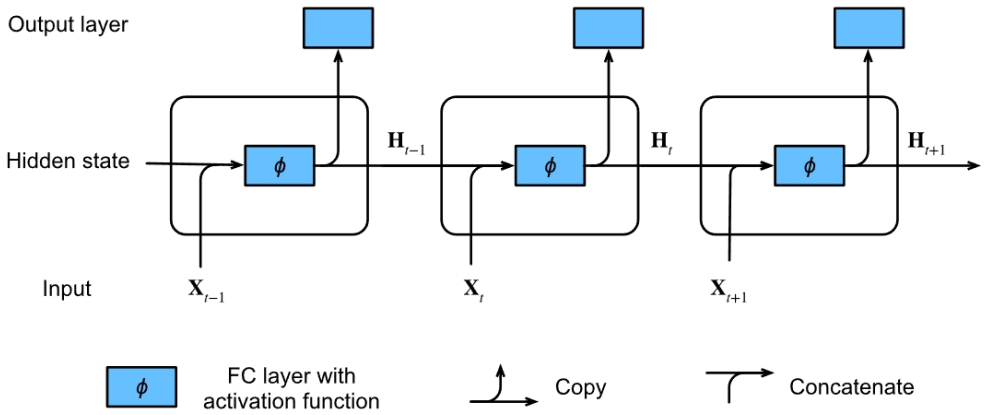



Figure 1.16: RNN with a hidden state architecture

Figure 1.16 shows the computational logic of an RNN at three consecutive time steps. At any time step $t$, the hidden state is calculated by combining the input $\mathbf{X}_t$ of the current time step and the hidden state $\mathbf{H}_{t-1}$ of the previous time step, and then feeding the result into a fully connected layer with the activation function $\phi$. The output of this layer is the hidden state $\mathbf{H}_t$ of the current time step. The model parameters are the combination of $\mathbf{W}_{\text{xh}}$ and $\mathbf{W}_{\text{hh}}$, and a bias of $\mathbf{b}_{\text{h}}$. The hidden state $\mathbf{H}_t$ of the current time step is used to calculate the hidden state $\mathbf{H}_{t+1}$ of the next time step and is also fed into the fully connected output layer to calculate the output $\mathbf{O}_t$ of the current time step.

### 1.4.2 LSTM

RNNs have difficulty dealing with long-term dependencies due to vanishing and exploding gradient problems (Bengio *et al.,* 1994), making it difficult for them to remember information from distant past steps. To address this, long short-term memory (LSTM) networks (Schmidhuber, Hochreiter, *et al.,* 1997) were developed, which incorporate memory cells in the hidden layer of the network. The name of "long short-term memory" is derived from the idea that the network possess both long-term and short-term memory. The long-term

memory is represented by the weights of the memory cell, which store general information about the data. The short-term memory is represented by the transient activations that are passed from one node to the next.

**Gates and input node** The data fed into the Long Short-Term Memory (LSTM) gates is the input at the current time step and the hidden state of the previous time step, as shown in Figure 1.17. Three fully connected layers with sigmoid activation functions are used to calculate the values of the *input*, *forget*, and *output* gates. The sigmoid activation ensures that all values of the three gates are in the range of $(0, 1)$. Input node was used to transmit memory, and the tanh activation function was typically employed to calculate it. The input gate decides how much of the input node's value should be added to the current memory cell internal state. The forget gate decides whether to keep the current value of the memory or discard it. Then the output gate determines whether the memory cell should affect the output at the current time step.

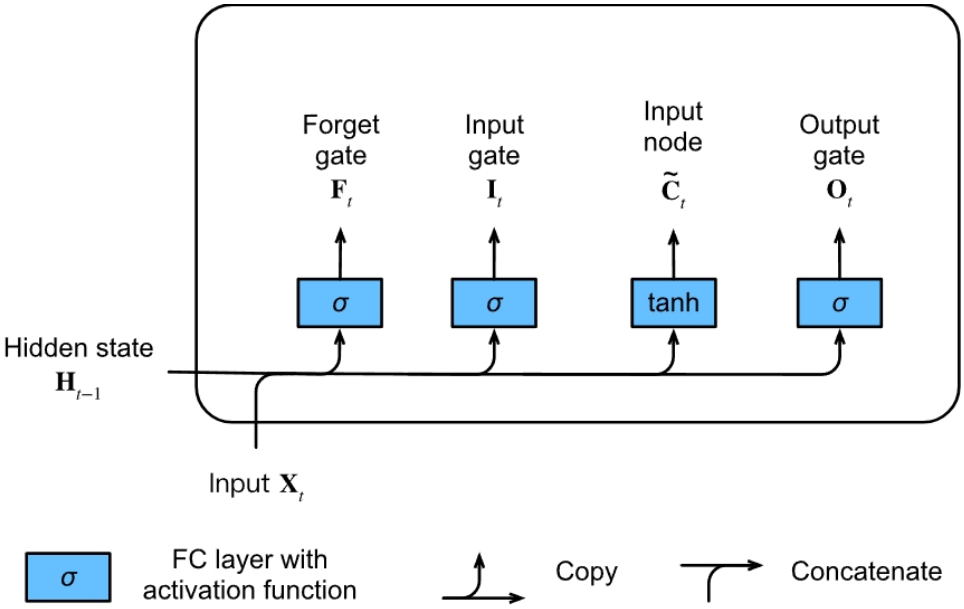



Figure 1.17: Computing the input gate, input node, the forget gate, and the output gate in an LSTM model.

Mathematically, given a batch size $n$, a number of hidden units $h$, and the input dimensionality $d$, the input at time $t$ can be represented as $\mathbf{X}_t \in \mathbb{R}^{n \times d}$. Similarly, the prior hidden state can be denoted as $\mathbf{H}_{t-1} \in \mathbb{R}^{n \times h}$. At any given time step $t$, the network incorporates three essential gates: the input gate, $\mathbf{I}_t \in \mathbb{R}^{n \times h}$; the forget gate, $\mathbf{F}_t \in \mathbb{R}^{n \times h}$; and the output

gate, $\mathbf{O}_t \in \mathbb{R}^{n \times h}$. The mathematical formulations for these gates, given the input and the prior hidden state, are:

$$\mathbf{I}_t = \sigma(\mathbf{X}_t \mathbf{W}_{\text{xi}} + \mathbf{H}_{t-1} \mathbf{W}_{\text{hi}} + \mathbf{b}_{\text{i}}),$$
$$\mathbf{F}_t = \sigma(\mathbf{X}_t \mathbf{W}_{\text{xf}} + \mathbf{H}_{t-1} \mathbf{W}_{\text{hf}} + \mathbf{b}_{\text{f}}),$$
$$\mathbf{O}_t = \sigma(\mathbf{X}_t \mathbf{W}_{\text{xo}} + \mathbf{H}_{t-1} \mathbf{W}_{\text{ho}} + \mathbf{b}_{\text{o}}),$$

where the weight parameters, $\mathbf{W}_{\text{xi}}, \mathbf{W}_{\text{xf}}, \mathbf{W}_{\text{xo}} \in \mathbb{R}^{d \times h}$ and $\mathbf{W}_{\text{hi}}, \mathbf{W}_{\text{hf}}, \mathbf{W}_{\text{ho}} \in \mathbb{R}^{h \times h}$, and the bias parameters, $\mathbf{b}_{\text{i}}, \mathbf{b}_{\text{f}}, \mathbf{b}_{\text{o}} \in \mathbb{R}^{1 \times h}$. Then the input node (used to calculate the next internal state), $\tilde{\mathbf{C}}_t$, can be defined as:

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{\text{xc}} + \mathbf{H}_{t-1} \mathbf{W}_{\text{hc}} + \mathbf{b}_{\text{c}}),$$

with the weight parameters $\mathbf{W}_{\text{xc}} \in \mathbb{R}^{d \times h}$ and $\mathbf{W}_{\text{hc}} \in \mathbb{R}^{h \times h}$, and the bias parameter $\mathbf{b}_{\text{c}} \in \mathbb{R}^{1 \times h}$.

**Internal State and Hidden State** The internal state $\mathbf{C}_t \in \mathbb{R}^{n \times h}$ is updated by a combination of input gate, $\mathbf{I}_t$, and the forget gate, $\mathbf{F}_t$: ($\odot$ stands for element wise multiplication)

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \tilde{\mathbf{C}}_t.$$

If the forget gate is always set to 1 and the input gate is always 0, the internal state of the memory cell $\mathbf{C}_{t-1}$ will remain unchanged over time. With this design, the model can update the memory by the current state information with the control of forget gate.

The hidden state, $\mathbf{H}_t \in \mathbb{R}^{n \times h}$, is the output of the memory cell, calculated by the output gate, $\mathbf{O}_t$:

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t).$$

This mechanism ensures that $\mathbf{H}_t$ values are confined to the range (-1, 1). The modulating action of the output gate allows LSTM to regulate when the internal memory impacts the subsequent network layers. Consequently, LSTMs are equipped with the ability to selectively propagate memory over extended temporal intervals, thus mitigating challenges such as the

vanishing gradient problem prevalent in traditional RNNs. In RNNs, especially with long sequences, the gradients can become exceedingly small as they are propagated back through each timestep. This occurs due to the repeated multiplication of gradients through the layers, which can lead to extremely small values if the gradients are less than 1. When gradients vanish, the weights in the early layers of the network barely change, meaning that the network doesn't learn the long-range dependencies in the data. This severely limits the model's ability to learn from data where the context is far back in time. LSTM as shown in Figure 1.18, provide additional memory state that can keep the long range dependencies, therefore fix the vanishing gradient problem.

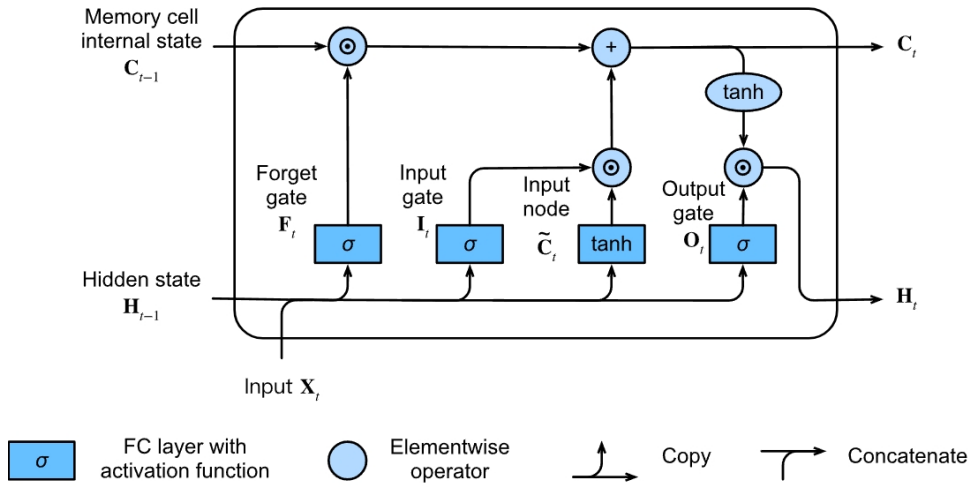



Figure 1.18: Computing the memory cell internal state and the hidden state in an LSTM model.

### 1.4.3 GRU

The Gated Recurrent Unit (GRU) (Cho *et al.*, 2014) was created as a simplified version of the Long Short-Term Memory (LSTM) model, with the intention of preserving the essential concept of incorporating an internal state and multiplicative gating mechanisms while accelerating the computation process.

Figure 1.19 shows the architecture of the GRU, consisting the **reset gate** ($\mathbf{R}_t$) and the **update gate** ($\mathbf{Z}_t$). The mathematical formulations of these gates are:

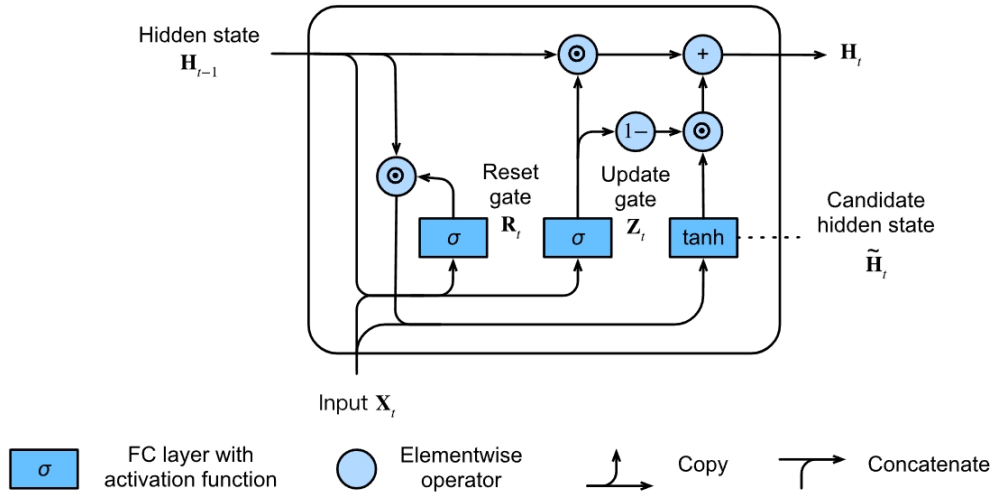



Figure 1.19: Architecture of a GRU model.

$$\mathbf{R}_t = \sigma(\mathbf{X}_t \mathbf{W}_{\text{xr}} + \mathbf{H}_{t-1} \mathbf{W}_{\text{hr}} + \mathbf{b}_{\text{r}}), \tag{1.1}$$

$$\mathbf{Z}_t = \sigma(\mathbf{X}_t \mathbf{W}_{\text{xz}} + \mathbf{H}_{t-1} \mathbf{W}_{\text{hz}} + \mathbf{b}_{\text{z}}). \tag{1.2}$$

The **candidate hidden state**($\tilde{\mathbf{H}}_t$) emerges as a fusion of the input in the current time step and the previous hidden state modulated by the reset gate, represented as:

$$\tilde{\mathbf{H}}_t = \tanh(\mathbf{X}_t \mathbf{W}_{\text{xh}} + (\mathbf{R}_t \odot \mathbf{H}_{t-1}) \mathbf{W}_{\text{hh}} + \mathbf{b}_{\text{h}}). \tag{1.3}$$

The final hidden state is a linear combination of the previous state and the candidate state, guided by the update gate:

$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t. \tag{1.4}$$

This design allows GRUs to adeptly manage short-term sequences dependencies by reset gates, and long-term sequence dependencies by update gates. In comparison to LSTM networks, GRUs can achieve similar performance but require less computational power.

## 1.5 Attention Mechanisms and Transformers

The visual system of primates receives a vast amount of sensory input, far exceeding what the brain can fully process (Thorpe *et al.,* 1996). However, not all stimuli are treated equally. The convergence and focus of consciousness allow primates to direct their attention towards objects of interest in complex visual environments, such as prey and predators. The ability to access only a small fraction of information is evolutionarily significant, enabling humans to survive and thrive (Campbell & Kulikowski, 1966). It is this notion of selective attention that influenced the creation of Attention Mechanisms, leading to the development of the Transformer by (Vaswani *et al.,* 2017). This architecture demonstrated exceptional performance, and by 2018, the Transformer had emerged in most state-of-the-art natural language processing models.

At present, the majority of natural language processing projects are based on the Transformer architecture. The prototypical methodology for any new task is to use a pre-trained Transformer model, such as BERT (Devlin *et al.,* 2018). The output layers are then adapted to the specific task and the model is fine-tuned with the relevant data. In recent years, OpenAI's large language models, such as GPT-3 (Brown *et al.,* 2020) and GPT-4 (OpenAI, 2023), have been the focus of much attention. These large language models have demonstrated remarkable success in a variety of NLP tasks such as text generation, document classification, and automatic translation, highlighting their versatility and superiority.

Simultaneously, in the computer vision domain, the Vision Transformer has emerged as the archetypal model for a multitude of tasks, ranging from image recognition to super-resolution, as highlighted in (Dosovitskiy *et al.,* 2020) and (Z. Liu *et al.,* 2021). Furthermore, the applicability of Transformers transcends beyond NLP and computer vision. Their effectiveness has been demonstrated in speech recognition (Gulati *et al.,* 2020), reinforcement learning (L. Chen *et al.,* 2021) and predicting complex protein structures (Jumper *et al.,* 2021) among others.

### 1.5.1 Attention

In the attention mechanism, we have three important parts: query, key and value. Imagine you're a researcher walking into a library with a specific question in mind. This question

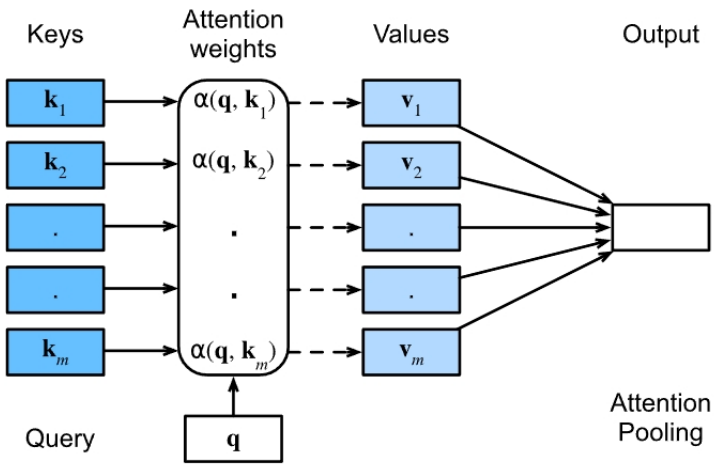



Figure 1.20: The attention mechanism calculates a linear combination of the values $\mathbf{v}_i$ through attention pooling, with the weights determined by the similarity between the query $\mathbf{q}$ and the keys $\mathbf{k}_i$. It combines query and keys to achieve a **attention weight** towards values

is your "*query*." It's what you're actively looking to find more information about. In the library, each book has a unique title and subject matter. These are like the "*keys*" in the attention mechanism. Just as a book's title and summary give you an idea of what information it contains, the keys in a model provide a way to match with relevant queries. Finally, once you find books with titles and summaries (keys) that seem to match your query, you'll actually delve into these books to extract information. In the attention mechanism, this refers to the "*value*." It's the actual content or the detailed information you extract from the data that is relevant to your query.

Here we present the general math formulation. Given a database represented as $\mathcal{D} \stackrel{\text{def}}{=} (\mathbf{k}_1, \mathbf{v}_1), \ldots (\mathbf{k}_m, \mathbf{v}_m)$ consisting of $m$ tuples of keys and values. Let $\mathbf{q}$ be a designated query. The attention over $\mathcal{D}$ is defined as:

$$\text{Attention}(\mathbf{q}, \mathcal{D}) \stackrel{\text{def}}{=} \sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i,$$

Here, the scalar attention weights, $\alpha(\mathbf{q}, \mathbf{k}_i) \in \mathbb{R}$ for $i = 1, \ldots, m$, play a pivotal role. This operation is generally termed as *attention pooling*. The significance of the term "attention" is derived from the fact that higher weight value $\alpha$ corresponds to higher attention. As a result, the attention over $\mathcal{D}$ essentially computes a linear combination of the database's values.

For ensuring a summation of weights to unity, normalization is typically applied:

$$\alpha(\mathbf{q}, \mathbf{k}_i) = \frac{\alpha(\mathbf{q}, \mathbf{k}_i)}{\sum_j \alpha(\mathbf{q}, \mathbf{k}_j)}.$$

In particular, to ensure that the weights are also nonnegative, we can use exponentiation.

$$\alpha(\mathbf{q}, \mathbf{k}_i) = \frac{\exp(a(\mathbf{q}, \mathbf{k}_i))}{\sum_j \exp(a(\mathbf{q}, \mathbf{k}_j))},$$

To guarantee that the variance of the attention function stays at 1 regardless of vector length, we employ the *scaled dot product attention* scoring function. This rescales the dot product by $1/\sqrt{d}$, leads to the commonly used attention function that is used in Transformers (Vaswani *et al.,* 2017).

$$a(\mathbf{q}, \mathbf{k}_i) = \mathbf{q}^\top \mathbf{k}_i / \sqrt{d}$$

Integrating the above dot product attention scoring function into the softmax operation:

$$\alpha(\mathbf{q}, \mathbf{k}_i) = \text{softmax}\left(\frac{\mathbf{q}^\top \mathbf{k}_i}{\sqrt{d}}\right) = \frac{\exp(\mathbf{q}^\top \mathbf{k}_i / \sqrt{d})}{\sum_{j=1} \exp(\mathbf{q}^\top \mathbf{k}_j / \sqrt{d})}.$$

In practice, we usually consider minibatches for efficiency, hence the *scaled dot product attention* of query $\mathbf{Q} \in \mathbb{R}^{n \times d}$, key $\mathbf{K} \in \mathbb{R}^{m \times d}$, and value $\mathbf{V} \in \mathbb{R}^{m \times v}$ can be expressed as follows:

$$\text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V} \in \mathbb{R}^{n \times v}.$$

‘

### 1.5.2 Transformer

The Transformer model represents a paradigm shift in model architecture for language based tasks, abandoning recurrent neural network architecture in favor of a fully attention-based approach. This method focuses on leveraging attention mechanisms to establish global relationships between input and output elements.

**Multihead Attention** Given query $\mathbf{q}$ in $\mathbb{R}^{d_q}$, key $\mathbf{k}$ in $\mathbb{R}^{d_k}$, and value $\mathbf{v}$ in $\mathbb{R}^{d_v}$, each attention head $\mathbf{h}_i$ $(i = 1, \ldots, h)$ is computed as

$$\mathbf{h}_i = f(\mathbf{W}_i^{(q)}\mathbf{q}, \mathbf{W}_i^{(k)}\mathbf{k}, \mathbf{W}_i^{(v)}\mathbf{v}) \in \mathbb{R}^{p_v},$$

where $\mathbf{W}_i^{(q)} \in \mathbb{R}^{p_q \times d_q}$, $\mathbf{W}_i^{(k)} \in \mathbb{R}^{p_k \times d_k}$, and $\mathbf{W}_i^{(v)} \in \mathbb{R}^{p_v \times d_v}$ are learnable parameters and $f$ is a *scaled dot product attention*. The output of the multi-head attention is a linear transformation via learnable parameters $\mathbf{W}_o \in \mathbb{R}^{p_o \times hp_v}$ of the combination of $h$ heads:

$$\mathbf{W}_o \begin{bmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_h \end{bmatrix} \in \mathbb{R}^{p_o}.$$

This design allows each head to access different parts of the input, enabling more complex functions than a simple weighted average. When the query $\mathbf{q}$ in $\mathbb{R}^{d_q}$, key $\mathbf{k}$ in $\mathbb{R}^{d_k}$, and value $\mathbf{v}$ in $\mathbb{R}^{d_v}$ are the same, it is called multihead self-attention. The multihead self-attention design allows the model to model more complex patterns.

**Positional Encoding** Given an input representation $\mathbf{X} \in \mathbb{R}^{n \times d}$ which contains $d$-dimensional embeddings for $n$ tokens of a sequence, the positional encoding outputs $\mathbf{X} + \mathbf{P}$ with a positional embedding matrix $\mathbf{P} \in \mathbb{R}^{n \times d}$ of the same dimension. The elements of the matrix are determined by the following equations: $p_{i,2j} = \sin\left(\frac{i}{10000^{2j/d}}\right)$ and $p_{i,2j+1} = \cos\left(\frac{i}{10000^{2j/d}}\right)$ for the row $i^{\text{th}}$ and the column $(2j)^{\text{th}}$ or $(2j+1)^{\text{th}}$.

**Transformer Architecture** The Transformer model can process the input and output sequences in parallel rather than sequentially. As illustrated in Figure 1.21, The model consists of an encoder and a decoder. The source and target sequence embeddings are augmented with positional encoding before being fed into the encoder and decoder, which are composed of modules based on self-attention. The "add norm" component in Figure 1.21 is a residual connection immediately followed by layer normalization. Layer normalization is the same as batch normalization except that the former normalizes across the feature dimension, thus enjoying benefits of scale independence and batch size independence.

The Transformer encoder is composed of multiple identical layers, each of which has two

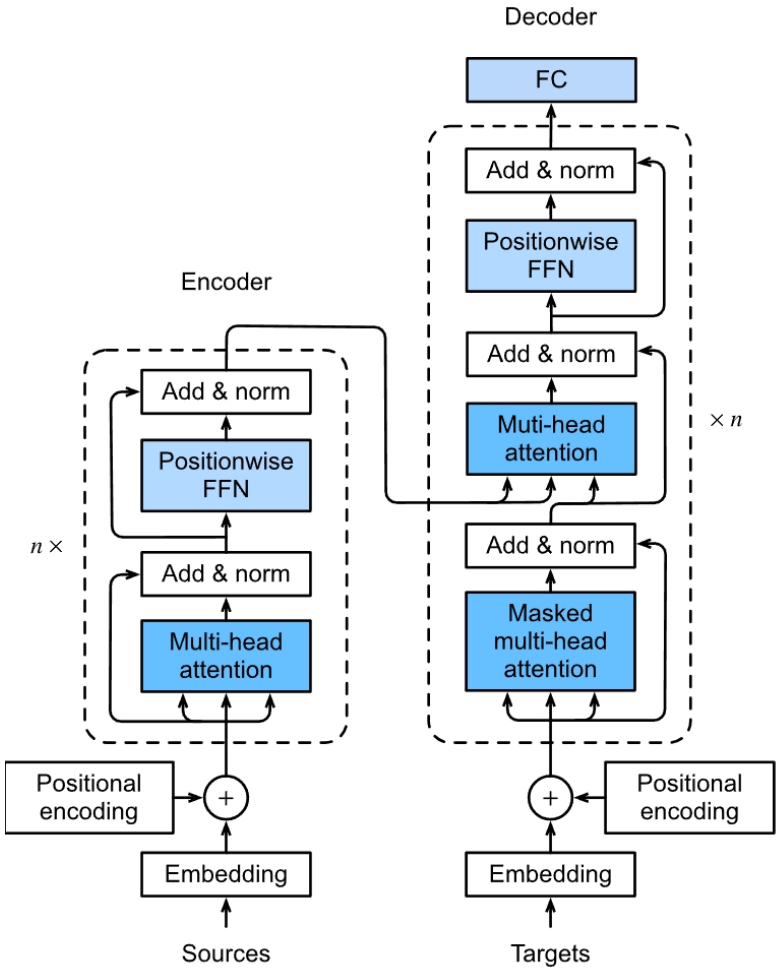



Figure 1.21: The Transformer architecture. FFN means feed forward network, which is fully connected layer.

sublayers. The first is a multi-head self-attention pooling and the second is a positionwise feed-forward network. The queries, keys, and values for the self-attention are all taken from the outputs of the previous encoder layer. The ResNet (K. He *et al.,* 2016a) is the inspiration for the residual connection that is used around both sublayers. For any input $\mathbf{x} \in \mathbb{R}^d$ at any position of the sequence, the output of the sublayer must be $\text{sublayer}(\mathbf{x}) \in \mathbb{R}^d$ so that the residual connection $\mathbf{x} + \text{sublayer}(\mathbf{x}) \in \mathbb{R}^d$ is possible. This is followed by layer normalization.

The Transformer decoder is composed of multiple identical layers with residual connections and layer normalizations. Additionally, it has a third sublayer, known as the encoder–decoder attention, between the two sublayers described in the encoder. In the encoder–decoder attention, queries come from the outputs of the decoder's self-attention sublayer,

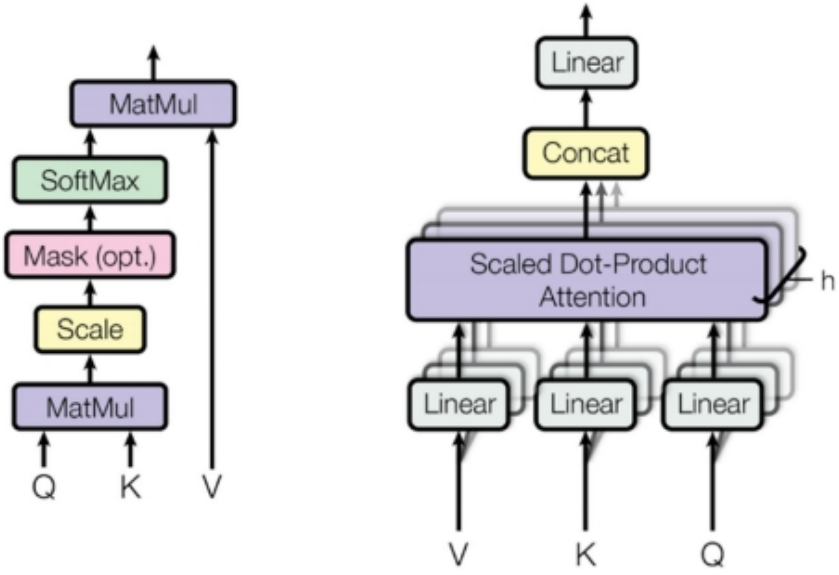



Figure 1.22: (Left) Self-attention process. (Right) Multi-head attention.

while the keys and values are from the Transformer encoder outputs. In the decoder self-attention, queries, keys, and values are all from the outputs of the previous decoder layer. However, the **masked attention** feature ensures that each position in the decoder can only access all positions in the decoder up to that position. This preserves the autoregressive property, meaning that the prediction only depends on the output tokens that have been generated.

## 1.6 Interpretability of Convolution models in transcription factor analysis

Predicting transcription factor binding sites constitutes a central thrust in cancer genomics research (Alipanahi *et al.,* 2015). Among the computational tools employed for this prediction task, Convolution Neural Networks (CNNs) have consistently demonstrated superior efficacy. In contrast to traditional, non-deep learning computational techniques that typically rely on k-mers and position weight matrices, CNNs have emerged as more accurate predictive tools. As in other fields, the precise rationale underlying the augmented performance of deep learning models over their traditional counterparts remains an area of active investigation.

Koo and his colleagues conducted a thorough investigation into how deep convolution

neural networks (CNNs) extract information from sequence motifs and compared the results across different CNN architectures (Koo & Eddy, 2019). It has been speculated that the first convolution layer is mainly responsible for motif representations and that subsequent deeper layers integrate and process these initial representations, leading to the identification of more complex patterns, known as regulatory grammars (Quang & X. Xie, 2016, Kelley *et al.*, 2016). To explore this hypothesis, Koo and his team examined the relationship between the internal convolution kernel representation and the overall network architecture. They proposed that the filters in the convolution layer(s) may be functionally similar to position-weight matrices, suggesting that these filters can interpret sequences in a manner similar to how position-weight matrices work. Based on their findings, which are discussed in more detail in the following sections, Koo and his colleagues concluded that the motif representations identified by CNN filters are significantly affected by certain aspects of the CNN model's design and hyperparameters, such as convolution filter size, filter stride, max-pooling size, and max-pooling stride. This conclusion emphasizes the importance of understanding how model architectures and their hyperparameters influence learning and representation.

To test their proposed hypothesis, Koo and colleagues conducted experiments using synthetic datasets. These datasets were derived from 25 position weight matrices, each representing binding motifs of 12 distinct transcription factors. For clarity, a position weight matrix quantitatively illustrates the binding motifs of transcription factors, providing a probabilistic measure of the nucleotides present at each position in a binding site. In addition to synthetic data, they also utilized real ChiP-seq data from the DeepSEA dataset (J. Zhou & Troyanskaya, 2015). One salient finding from their experiments was the pronounced influence of max-pooling size on the motif representations acquired by the initial convolution layer's kernel. In neural networks, max-pooling refers to a downsampling strategy, and its size dictates the dimensions of the region considered when selecting the maximum value. In their experiments, they discovered that keeping other hyperparameters constant, using a larger max-pooling size (10 or more) enabled the convolution kernel in the first layer to identify more than 90% of the ground truth motifs. Conversely, when the max-pooling size was reduced to 4 or less, this identification rate drastically fell to below 25%. This implies that the kernel in the initial layer can, to some extent, distinguish a motif based on the size

of the max-pooling.

In their study, the impact of the stride parameter in the max-pooling layer on the ability of the first convolution layer's kernel to represent motifs was examined. They compared a stride of 2, which allows the max-pooling kernel to overlap, with a stride of 50, which does not. In both scenarios, more than 90% of the ground truth motifs were accurately identified. However, the model with the smaller stride produced a more cohesive representation of the entire motif. These findings suggest that the overlapping of the max-pooling filter, along with the subsequent convolution layer's filter, improves the model's capability to create comprehensive kernel representations from the initial layer, thereby enhancing the overall quality of motif representation. Additionally, they observed that even when the initial layer's kernel representation did not match the ground truth, the second layer could still detect motif patterns. It is important to note that while the models are informative, they are still relatively basic, so it may be too ambitious to apply the same insights to more complex models.

Koo and his research team investigated the trade-offs between the complexity of deep learning models and their interpretability Koo, Qian, *et al.,* 2019. They used two synthetic datasets with either 30,000 or 10,000 sequences, which either contained known sequence motifs (Mathelier *et al.,* 2016) or did not, to test the potential of robust training methods such as regularization, data augmentation with random noise, and adversarial training. To measure the interpretability of the models, they used the area under the receiver-operator characteristic curve (AU-ROC) and the area under the precision-recall curve (AU-PR). They found that while deeper neural networks had better prediction accuracy, this added depth decreased the clarity of motif representations, which is essential for understanding the biological implications of model outputs. They also discovered that a model's prediction accuracy is not intrinsically tied to its interpretability. Notably, while both single-layer and multi-layer CNN models achieved comparable accuracies (0.96), the former boasted significantly higher AU-ROC and AU-PR metrics. This suggests that CNN models with simpler architecture are more interpretable. The team hypothesized that this discrepancy in interpretability might be due to the nature of the neural network's complexity, as deeper neural networks might adapt to more intricate and potentially "noisier" functions, making them less amenable to

interpretations. They also observed the pronounced effect of regularization techniques on model interpretability, as the application of such techniques improved the transparency of single-layer neural networks, even though there was a small loss in the accuracy. Furthermore, they noted the utility of input perturbation methods, such as Gaussian noise injection, in enhancing interpretability. These perturbation techniques effectively highlight the robust features the model relies on, thereby aiding in model interpretation. Lastly, the authors discussed the potential of adversarial training (W. Zhu & X. Xie, 2016) as a method to increase the transparency of multi-layer CNN models. The conclusion here was that adversarial training can expose and subsequently strengthen the model against subtle features that could lead to incorrect interpretations, thus making the model's internal workings more transparent and reliable.

Koo's work mostly focus on interpretation model in transcription factor analysis, and the next section will introduce more about deep learning in biology and medicine.

## 1.7 Opportunities and obstacles for deep learning in biology and medicine

The paper by Ching *et al.,* 2018 delves into the potential of deep learning in the realms of biology and medicine, scrutinizing both the challenges and opportunities inherent in applying these models in such complex fields. The authors investigate whether deep learning can bring about a transformative impact on the study of human diseases. Their analysis includes examples where deep learning has effectively tackled problems previously deemed challenging or labor-intensive. They concluded that deep learning holds significant promise, having already made notable strides in these areas. However, they emphasize that interpretability remains a crucial hurdle. The ability to understand the patterns that these models learn is as vital as their performance in fitting the data. This point underscores the importance of incorporating domain-specific knowledge to ensure accurate data representation. Reflecting on my study of cancer, where we didn't collaborate with a pathologist but still managed to enhance prediction accuracy, it becomes evident that integrating expert knowledge can be a valuable asset in refining these models, even in instances where direct collaboration with domain experts is not feasible.

### 1.7.1 Disease and patient categorization

In Chapter 3, we focus on the challenge of cancer survival and patient categorization. The critical issue here is the precise classification of diseases and their subtypes. Deep learning models provide a data-driven approach that could potentially reveal new disease classes or shared mechanisms. However, it's important to be cautious about the capability of these models for diagnosis. The models often rely on post-diagnosis data, which is essential for training but can introduce biases if not handled correctly. The current literature shows that deep learning is already contributing to the discovery of novel categories, but there's a need to address challenges like data availability and proper labeling (Ching *et al.,* 2018).

Shifting to the contributions of this thesis in medical imaging, deep learning has significantly advanced imaging applications in healthcare. These methods have improved the classification and detection of lesions and nodules in medical images. A major challenge, however, is the scarcity of consistent and well-labeled datasets, a stark contrast to the extensive collections like ImageNet (Russakovsky *et al.,* 2015). Medical images often vary in resolution and pixel sizes, necessitating the development of task-specific models. Researchers have employed data augmentation (Dhungel *et al.,* 2017) and adversarial training (W. Zhu & X. Xie, 2016) to enrich the diversity of medical images. For instance, Roth *et al.,* 2015 demonstrated the superiority of 2.5D CNN models (2D CNN model with a branch for spatial prior) over 3D models in abnormal detection from CT scans due to shorter training times and the availability of pretrained models. In contrast, Nie *et al.,* 2016 highlighted the effectiveness of multimodal and multichannel 3D models in learning features of brain tumors from MRI datasets. Furthermore, D. Wang *et al.,* 2016 developed a model for detecting metastatic breast cancer in WSIs of sentinel lymph node biopsies, and Rakhlin *et al.,* 2018 successfully applied deep neural networks with data augmentation techniques for breast cancer detection from small datasets. These studies are particularly relevant to my work in Chapter 3 as they provide insights into effective methodologies and highlight the importance of deep learning in cancer diagnosis.

The most difficult aspect of patient categorization is the lack of ground-truth labels and the limited size of datasets. Deep learning classification models require ground-truth labels for training, but in the biomedical field, assigning such labels can be costly and time-

consuming (Zheng *et al.,* 2017). Even for well-resourced national consortia with expert-validated labelled data, such as the eMERGE consortia and PheKB database (Kirby *et al.,* 2016), they may only contain a few hundred patients, which is too small to train deep learning models. To address these challenges, (X. Wang *et al.,* 2017) developed a deep neural network model based on automatically generated weak labels, even though these labels are not verified by human experts and are of limited accuracy. They first used NLP models to extract all diseases mentioned in the associated chest X-ray radiological reports, and then applied NegBio (Peng *et al.,* 2018) to filter out negative and equivocal findings in those reports. With these unified weakly supervised labels and the associated X-ray image dataset, the authors were able to successfully train a model to classify and detect common thoracic diseases **accuracy?**.

Other authors have combined automated label generation with human oversight. In Iglovikov *et al.,* 2018, the author employed an iterative process to obtain high-quality semi-automatic labels. Initially, they manually labeled 100 of the 12,600 unlabeled radiographs, and then used the labeled ones to predict the labels for the remaining images. Subsequently, experts inspected the results and discarded the poor-quality predictions, while the correct predictions were added back to the training set. After six iterations, they achieved good-quality segmentation labeling for all radiographs except for a few cases. This semi-automatic annotation procedure significantly reduced human labor and improved the performance of deep learning models. **what type of images/data?**

### 1.7.2 Transfer learning

Due to the limited number of samples or labels in biomedical datasets, deep learning models are prone to overfitting when trained on these small datasets. To address this issue, transfer learning, also known as domain adaptation, can be used to extract features between different datasets. This approach involves training a model on a baseline task with a large dataset, and then fine-tuning the trained model for the target problem or relatively small datasets. As the model was trained with a large number of observations, it can extract better feature representations and reduce overfitting, thus improving performance compared to a model trained on the smaller dataset directly using random initialization. In image analysis, W. Zhang *et al.,* 2016 and Zeng *et al.,* 2015 have shown that transfer learning using

pre-trained models can be a good generic feature extractor for biological images, however the transferability of these features decreases as the distance between the base task and the target task increases (Yosinski *et al.,* 2014). Additionally, there is no theory to guarantee the feature transferability. Therefore, fine-tuning on the final small dataset is essential to improve the prediction accuracy. The Basset package (Kelley *et al.,* 2016) can accurately predict on new datasets by leveraging a model pre-trained on previous datasets, and another model, DeepEnhancer (Min *et al.,* 2016), successfully trained based on the experimentally validated FANTOM5 permissive enhancer datasets, and then fine-tuned on ENCODE enhancer datasets, outperforming the state-of-the-art results.

### 1.7.3 Multimodal learning

Multimodal learning is a popular approach that combines information from various types of inputs, such as histology images, text, and genomic datasets. Ngiam *et al.,* 2011 showed that generative graphical models, such as restricted Boltzmann machines, deep Boltzmann machines, and deep belief networks, can successfully learn a joint probability distribution from inputs of images or videos when jointly learned with other deep learning models focusing on audio or text. Furthermore, when there is not enough labeled data, these models can be pre-trained unsupervised and then fine-tuned on a smaller dataset with labels. Jha *et al.,* 2017 designed a feed-forward neural network trained on RNA-seq data with additional sets of CLIP-seq, knockdown, and over-expression based input features, and showed that the integrated model achieved a large improvement for alternative splicing event estimation compared to non-integrated models. Chaudhary *et al.,* 2018 trained a deep autoencoder model integrating the RNA-seq, miRNA-seq, and methylation data from the TCGA dataset to predict survival subgroups of hepatocellular carcinoma (HCC) patients, and their multimodal approach outperformed both traditional principal component analysis and deep learning models using single-omic data. They also found that the integration of clinical information in the multimodal did not improve performance, possibly because the correlated genomic features learned by the model included the clinical information. M. Liang *et al.,* 2014 used deep belief networks (DBNs) to jointly extract unsupervised representation features from cancer datasets, including gene expression, DNA methylation, and miRNA expression data. Their approach allowed for the integration of correlations in different modalities and thus

performed better than traditional k-means clustering methods. FIDDLE Eser & Churchman, 2016 applied multimodal learning with convolution neural networks (CNNs) to predict transcription start sites, where a collection of individual networks took NET-seq, MNase-seq, CHIP-seq, RNA-seq, and raw DNA sequences as input. Each network learned representations from different datasets and then concatenated them in the fully connected layers. They demonstrated that the combined model greatly improved performance over separately trained single dataset networks, suggesting that multimodal learning can gain knowledge of relationships between datasets.

### 1.7.4 Multi-task learning

An alternate deep learning approach to transfer learning is multi-task learning. This framework involves one model that is responsible for multiple tasks, with shared features. In J. Zhou & Troyanskaya, 2015, the authors proposed a multi-task deep learning model, DeepSEA, which can learn diverse chromatin factors from raw DNA sequence. This model shared the DNA sequence feature in recognizing the binding of a specific transcription factor and another physically interacting transcription factor. H.-J. Yoon *et al.,* 2016 showed that a multi-task model predicting the primary cancer site along with the cancer laterality significantly improved the performance of the latter task. This result demonstrated that multi-task learning can improve the model performance by leveraging the commonality between different tasks with a shared representation extraction. W. Zhang *et al.,* 2016 combined transfer learning and multitask learning for the annotation of biological images in different domains, which showed that multitask learning, multimodal learning and transfer learning can work together, leveraging relationships between various inputs and task objectives. In the biomedical field, transfer learning, multimodal learning, and multitask learning have demonstrated their potential with the adoption of methodology from image, language and audio analysis, but medical domain challenges still exist. There are no theoretically sound guidelines for pre-training and fine-tuning for transfer learning, and the performance of such models depends greatly on the experience of researchers to choose hyper parameters wisely. Researchers can only access the relation between different medical datasets and different tasks by trial and empirical evaluation, and negative results may not be reported. Further investigation of theoretical guarantees of transferability of features and different medical learning tasks is

still needed.

### 1.7.5 Model interpretability

While deep learning models often exhibit superior performance compared to traditional methods in biology and medicine, their interpretability remains a significant concern. Biologists and doctors, who rely on these models, seek a clearer understanding of the underlying data patterns — a task that is challenging when dealing with opaque, black-box models. The imperative for interpretability is not just intellectual curiosity; it's also about ensuring that deep learning models are reflecting genuine biological or clinical patterns rather than identifying spurious relationships in the dataset. To tackle the issues of interpretability, two approaches have been suggested:

1. **Example-specific Importance Scores Method:** This involves calculating an importance score for every observation within each input. Two prevalent approaches underpin this method:

    - *Perturbation-based approaches:* These approaches modify segments of the input to gauge the resulting impact on the model's output. For instance, LIME, as introduced in Ribeiro *et al.,* 2016, generates a local linear model to emulate the neural network results based on altered versions of the input, thereby determining the significance scores.

    - *Backpropagation-based approaches:* These methods determine importance scores by backpropagating from the output neuron to the input layer or by computing the partial derivative of the output with respect to the input. A prime example is the saliency map presented by Simonyan, Vedaldi, *et al.,* 2013. Numerous adaptations have been developed to address inherent challenges like non-linearity in deep neural networks. Some notable examples include Grad-CAM (Selvaraju *et al.,* 2017a), deconvolution networks (Zeiler *et al.,* 2010), and integrated gradients (Sundararajan *et al.,* 2017). A distinct advantage of backpropagation-based methods is that they do not necessitate a separate forward propagation for collecting output importance scores; they can be performed concomitantly with model

training. Intriguingly, Lundberg & S.-I. Lee, 2016 observed that many of these importance score techniques can be viewed as approximations to the Shapley values (Shapley, 1953) from game theory, which allocate contributions to players in cooperative games.

2. **Activation Maximization Method** (Erhan *et al.,* 2009): This strategy aims to optimize the activation of specific neurons. Post-training, for a designated class or neuron, the method iteratively seeks an input that maximizes the score for that class. However, a limitation is its potential lack of information. Given that neural networks typically learn extensive distributed representations, results from activation maximization might be insufficiently descriptive, particularly for image-centric models.

The scientific community is striving to close the gap between the impressive abilities of deep learning models and the urgent requirement for them to be interpretable so that they can be used responsibly and knowledgeably in essential areas such as biology and medicine.

## 1.8 Multimodal machine learning in precision health

The review article: *Multimodal machine learning in precision health: A scoping review* (Kline *et al.,* 2022) explores the application of machine learning in health, particularly focusing on its use in clinical decision-support through the fusion of multimodal data. Historically, machine learning in health has primarily relied on single-modal data. This review highlights the shift towards integrating diverse data types, such as imaging and electronic health records (EHR), to enhance prediction accuracy and emulate clinical decision-making processes. The article notes an average 6.4% increase (mean improvement in AUC) in predictive when employing data fusion compared to unimodal approaches, underscoring the robustness of multimodal machine learning (Kline *et al.,* 2022). However, it also acknowledges challenges, including scalability issues and the complexity of information concatenation (L. V. Rasmussen, Brandt, *et al.,* 2019, Zhong *et al.,* 2018, L. V. Rasmussen, Hoell, *et al.,* 2020).

The study emphasizes the importance of data fusion, a process underpinned by information theory that merges different data sources to create a comprehensive information state.

Data fusion is expected to improve predictive power in machine learning, making results more robust by incorporating a variety of information factors (Castanedo *et al.,* 2013). However, it also introduces complexities in model specification and reduces result interpretability (Blasch *et al.,* 2021). Addressing data heterogeneity is crucial, as clinical data from various sources often differ in naming conventions, units of measure, and local population biases (Kohane *et al.,* 2021). Effective data fusion requires harmonization techniques to ensure quality control and interoperability. The paper notes innovative applications of data fusion in areas like heart failure (Ahmad *et al.,* 2022), where multidisciplinary research is essential for integrating deep phenotypic and trans-omic information (Luo *et al.,* 2017).

The paper categorizes data fusion into three types: early, intermediate, and late fusion. Early fusion converts multiple data sources into a single information space, often through vectorization or numerical conversion. Intermediate fusion allows for more flexible model architecture, combining features from different data types to create a new, more expressive representation. Late fusion typically involves training multiple models for each data source, akin to ensemble learning, to obtain a more accurate decision. The paper's definition of early fusion is not the same with our early fusion of co-attention model in Chapter 3. Our co-attention module should be classified as intermediate fusion based on the defninition of this review paper.

The paper also highlights that neurology and oncology are the most common health areas using multimodal methods (Koutsouleris *et al.,* 2021, S.-C. Huang *et al.,* 2020, Xiong *et al.,* 2022), with a particular emphasis on Alzheimer's disease research (El-Sappagh *et al.,* 2021, L. Yang *et al.,* 2021). The review further identifies the challenges in model validation, the need for diverse data representation (Colubri *et al.,* 2019), and the common limitations in the current studies, such as small sample sizes and biased models (P. Lin *et al.,* 2020). It also outlines future research gaps, emphasizing the need for more studies in underrepresented areas like medication/drug topics and advocating for the use of machine learning to improve clinical decision-making and outcomes (Kinreich *et al.,* 2021). It also stresses the importance of addressing data management tenets like Findability, Accessibility, Interoperability, and Reuse (FAIR) of datasets (Wilkinson *et al.,* 2016).

# Chapter 2

# Scratch-AID, a deep learning-based system for automatic detection of mouse scratching behavior with high accuracy

## 2.1 Abstract

Mice are frequently used as the primary animal model for investigating itch-related phenomena and for the advancement of pharmacological interventions that target itch. To evaluate the level of itch, several laboratories employ manual quantification of the mouse scratching behavior. The aforementioned procedure is characterized by a high demand for manual work and imposes constraints on conducting extensive genetic or drug testing on a large scale. Here, we present the development of a novel system, Scratch-AID (Automatic Itch Detection), designed to accurately detect and quantify mouse scratching behavior in an automated manner. The system utilized a specially built enclosure for video recording, aiming to maintain consistent and superior quality in the capture of mouse behavior. Additionally, a deep learning network (convolution recurrent neural network) was employed, which had been trained using videos of mice engaged in scratching behavior. This behavior was produced by administering chloroquine through a nape injection. The highest-performing network attained a recall rate of 97.6% and a precision rate of 96. 9% when evaluated in test movies that had not been encountered during training. Significantly, Scratch-AID demon-

strated consistent efficacy in detecting scratching activity in many prominent mice's itch models, such as the acute cheek model, the histaminergic model and a chronic itch model. Furthermore, our method has identified notable disparities in the act of scratching between the control group and the group of mice who received treatment with an anti-itch medication. In conclusion, we have developed an innovative deep learning system that has the potential to supplant manual quantification methods in the assessment of mouse scratching behavior in various scratching models and in the context of drug screening.

## 2.2 Introduction

Itching is a vexing symptom that is commonly observed in the context of dermatological conditions, immunological dysregulation, systemic diseases, and psychiatric disorders (Stander *et al.,* 2007; Hong *et al.,* 2011; Cevikbas & Lerner, 2020; Kremer *et al.,* 2020). The prevalence of chronic itch in the community is estimated to be around 13%-17% (Matterne *et al.,* 2009; Weisshaar & Dalgard, 2009). This condition significantly deteriorates the overall quality of life experienced by individuals affected by it. Regrettably, the available therapeutic modalities for various chronic itch disorders remain restricted (H. Yu *et al.,* 2021; Yosipovitch, Rosen, *et al.,* 2018).

Mice are widely used as model organisms in the field of itch research, serving as valuable tools for investigating itch processes and facilitating the development of novel preclinical antiitch medications (Q. Liu, Z. Tang, *et al.,* 2009; Solinski *et al.,* 2019; Sun & Z. F. Chen, 2007; L. Han *et al.,* 2013). The feeling of irritation is characterized by an unpleasant feeling that elicits the need to scratch (Ikoma *et al.,* 2006). To measure the degree of itch, researchers have used scratching activity as an indicator in mice (Q. Liu, Z. Tang, *et al.,* 2009; Morita *et al.,* 2015). Up until this point, the quantification procedure has mostly included observation of movies and manual tallying of scratching episodes or the overall duration of scratching. This approach is laborious and time-consuming, leading to inevitable human mistakes and biases, and imposing constraints on the feasibility of conducting extensive genetic or pharmacological testing.

Various research groups have made attempts to automate this process, recognizing its biological significance and evident necessity. These efforts have involved the exploration of

different strategies, such as the utilization of an acoustic recording method (Elliott *et al.,* 2017), the implementation of a method employing magnetic fields and metal rings to detect paw movement (Mu *et al.,* 2017), as well as the adoption of several approaches based on video analysis (Bohnslav *et al.,* 2021; Kobayashi *et al.,* 2021; Sakamoto *et al.,* 2022; Park *et al.,* 2019). However, the utilization of these techniques has not been widely accepted by other scientific laboratories. This could be attributed to the unknown efficacy of trained models in varying laboratory settings, the need for specific equipment, and/or insufficient validation of these approaches in other mice itch models.

In recent years, the science of artificial intelligence has witnessed significant advancements, leading to the widespread use of deep learning techniques in numerous domains of scientific inquiry. Convolution neural networks (CNN) are extensively used in the field of computer vision for tasks related to visual recognition (Gu *et al.,* 2018). On the other hand, recurrent neural networks (RNNs) have been specifically designed for the analysis of temporal dynamic data (Graves, 2013). Furthermore, there has been a significant advancement in computing power, particularly in the capacity of graphics processing units (GPUs). This, coupled with the emergence of new open-source deep learning libraries such as PyTorch (Paszke *et al.,* 2019), Keras (Gulli & Pal, 2017) and Tensorflow (Abadi *et al.,* n.d.), has greatly expedited the widespread adoption and utilization of deep learning techniques.

The use of deep learning has proven to be advantageous in the field of animal behavior analysis. An illustration of this may be seen in the work of DeepLabCut, which is capable of accurately monitoring and identifying various anatomical landmarks on the bodies of animals in unrestricted motion. This technology is particularly useful in the field of behavior analysis (Mathis *et al.,* 2018). The DeepEthogram has been developed to effectively identify and classify various behavioral patterns shown by mice and flies, as documented in the study by the authors (Bohnslav *et al.,* 2021). These examples provide evidence that deep learning is an effective approach for automating the study of animal behavior, thereby demonstrating its proof-of-principle. However, to substitute human observers in studying animal behavior, such as mouse scratching, a unique approach is necessary that can reach high levels of sensitivity, specificity, and generalization.

In order to address this particular difficulty, a novel system called Scratch-AID (Auto-

matic Itch Detection) was created. This system utilizes deep learning techniques and has demonstrated a remarkable ability to accurately quantify mouse scratching activity in an automated manner. Initially, a videography apparatus was developed with the purpose of capturing mouse activity in a consistent setting, ensuring the acquisition of superior-quality recordings. A total of 40 movies were captured, featuring a cohort of 10 wild-type adult mice (comprising 5 males and 5 females). Mice were subjected to nape injection of a nonhistamine pruritogen known as chloroquine (CQ). Subsequently, all frames within the recorded videos were meticulously annotated by hand, serving as a reference for further analysis. Subsequently, a convolution recurrent neural network (CRNN) was developed by integrating convolution neural network (CNN) and recurrent neural network (RNN) architectures. The CRNN was trained using a data set consisting of 32 movies that capture scratching behavior, collected from a cohort of 8 mice selected at random. A set of prediction models was generated employing various training parameters and, afterwards, these models were evaluated using test videos. Test vi consisted of eight previously unknown recordings taken from the remaining two mice. The highest performing model achieved a recall rate of 97.6% and an accuracy rate of 96.9% when evaluated on test videos, which aligns closely with the quantification findings obtained from manual analysis. Interestingly, Scratch-AID demonstrated the ability to accurately measure scratching behavior in several models of acute and chronic itch, in addition to its impressive performance. Finally, the Scratch-AID technique was implemented in a screening paradigm for anti-itch drugs, and it was observed that the technique consistently identified the presence of drug-induced effects. In conclusion, a novel approach has been developed for the precise and automated measurement of mouse scratching activity. On the basis of the observed results, it can be concluded that Scratch-AID has the potential to serve as a viable alternative to manual quantification in mice itch models, as well as in pharmacological or genetic tests.

## 2.3 Data and methods

The method used in the development of a novel system for the detection and measurement of mouse scratching behavior encompasses four primary phases, as illustrated in figure 2.1 part A. 1) The behavior of the mice scratching in response to an acute nape itch model

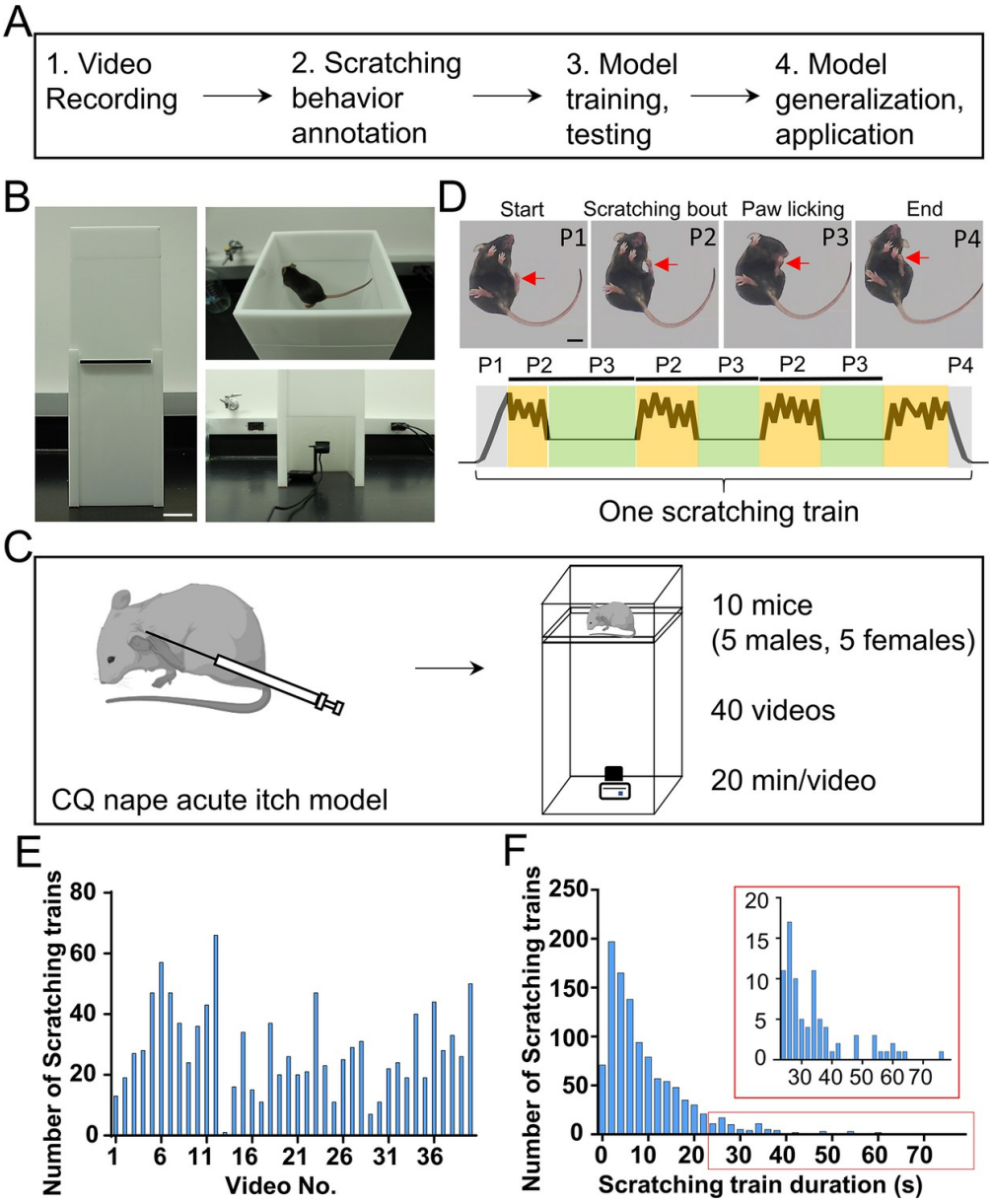



Figure 2.1: **The overview of construction of a customized videotaping box for the recording of mouse scratching behavior.**. (A) A flow chart illustrating the process of developing a system based on deep learning for the automatic detection and quantification of rodent clawing behavior. (B) An image of the videotaping enclosure designed to capture high-quality video of the clawing behavior of the mouse. The scale bar measures 5 cm. (C) Caricature depicting the acute itching model induced by injection of chloroquine (CQ) into the nape, followed by video recording in the custom video recording box. (D) Images illustrating the various phases (P1–P4) of a striking train (upper). The red arcs represent the clawing rear paw. A cartoon depicting the dynamic movement of a scratching train's rear hand (bottom). In a scratching train, the cycle of clawing bout (P2) and lapping (P3) can occur multiple times. Scale bar, 1 cm. Each video contains a total number of scraping trains. (F) The distribution of the duration of the scratching trains (n = 1135 scratching trains). The inset is a magnified version of the red rectangle.

was recorded on a videotape. 2) The process of manually annotating scratching frames in all recorded videos was conducted to create training and test data sets. The objective is to develop a deep learning neural network architecture. This network will be trained using a set of randomly selected training videos while adjusting various training parameters. Subsequently, the performance of the built neural networks will be assessed using test videos. 4) Assess the applicability of the trained neural network in several itch models and a drug screening paradigm.

### 2.3.1 Videotaping apparatus setting

The utilization of high-quality videos captured within a replicable setting is of utmost importance in ensuring the consistent performance of trained prediction models and facilitating their adoption by other research laboratories. Therefore, a mouse videotaping box was built for the aforementioned objective. The structure consisted of two boxes with white acrylic walls that were connected by a clear acrylic floor (Figure 2.1 part B). The upper compartment, known as the "mouse" box, has dimensions of 14.68 cm in length, 14.68 cm in width and 5 cm in height. This box was equipped with a cover that facilitated access for the mice. The lower compartment, with dimensions of 14.68 cm in length, 14.68 cm in width, and 23.6 cm in height, was equipped with a door to facilitate access to the Logitech C920e Business Webcam.

In order to reduce the impact of external visual cues, the walls and lid of the box were designed to be non-transparent. The walls were permeated with ambient light, which provided adequate lighting for the purpose of recording behavior. The top box allowed unrestricted movement for a mouse, while a camera situated at the bottom captured the mouse's activities at a rate of 30 frames per second. Unlike the top or side perspectives, the bottom view offers a distinct advantage in capturing the essential anatomical components involved in scratching activity, including the hind paw used for scratching, the mouth, and other relevant body parts. Furthermore, this vantage point allows for a more comprehensive examination of the intricate motions associated with scratching behavior. The magnification, quality, and brightness of the video may be modified by the camera recording software (Logitech C920e Business Webcam driver and software) to create consistent video recording. The recording parameters were adjusted using Logitech Capture 2.06.12 software. The brightness was set

to 170, the contrast to 0, the resolution to 720X720, and the frame rate to 30 fps. Consistent lighting may be achieved by adjusting the brightness and contrast in accordance with ambient light conditions. In summary, this specialized video recording apparatus enables the capture of high-fidelity videos depicting mouse scratching and other behavioral patterns inside a controlled and replicable setting.

The occurrence of spontaneous scratching in mice is infrequent in typical circumstances. The perception of itch and the act of scratching are often elicited by several itch models employed in the study. The classification of common mouse itch models encompasses many categories, including cheek or nape, histaminergic or non-histaminergic, and acute or chronic. These classifications are based on the specific body site where the sensation of itch is elicited, the kind of pruritogen used, and the length of the sensation of itch that results (Ikoma *et al.,* 2006; Q. Liu & Dong, 2015; Shimada & LaMotte, 2008; Thurmond *et al.,* 2008). The initial approach was the use of a nape itch model characterized by acute symptoms, generated by the administration of a non-histaminergic pruritogen known as chloroquine (CQ). This particular pruritogen was chosen due to its ability to cause prompt and vigorous scratching activity in mice (Q. Liu, Z. Tang, *et al.,* 2009). After administration of chloroquine (CQ) by intradermal injection at a dosage of 200 $\mu$g in 15 $\mu$l of saline solution, a 20-minute video recording was performed utilizing a specifically designed video recording apparatus (Figure 2.1 part C).

Mice scratched the affected skin area with their ipsilateral hind paw after receiving the CQ injection. Multiple episodes of scratching were observed in mice that were provoked, but these were interspersed with periods of silence. Each scratching instance, referred to as a scratching bout, typically consisted of four distinct phases: initiation (elevating the hind paw involved in scratching toward the affected area), scratching bout (repetitive motion of the hind paw against the affected area), paw licking (placing the scratching hind paw into the mouth and licking it) and termination (returning the scratching hind paw to the ground) (Figure 2.1 part D). The occurrence of scratching bouts and paw licking in mice can vary, with the cycle potentially happening once or being repeated numerous times. This is dependent on factors such as the strength of the itch and the internal state of the mouse. The duration of a specific scratching train is defined as the period from its initiation until its

completion. The cumulative duration of the scratching trains, known as the total scratching time, serves as a valuable metric for quantifying the scratching behavior and evaluating the level of scratching.

### 2.3.2 Video annotation

A total of 40 movies capturing scratching behavior were collected from a sample of 10 wild-type adult C57 mice, consisting of 5 males and 5 females. Mice were between 2-3 months old. The details of this data can be seen in (Table 2.3). Two approaches were employed to annotate mouse scratching activity in movies for the purposes of neural network training, testing, and performance comparison with manual quantification. The initial approach employed for annotation was to see the movies at their regular speed of 1X (30 frames per second) and to identify the specific time periods, converted into frame numbers, that marked the beginning and end of each instance of scratching behavior in the trains. This approach aligns with the established methodology in the field to manually assess mouse scratching activity. The findings of manual annotation were obtained by participating in ten human observers, therefore providing an average measure of precision for the manual quantification method. The second approach, known as reference annotation or ground-truth annotation, involved a meticulous analysis of each video frame to precisely identify the beginning and end of each scratching episode. The training and test data sets consisted of 40 videos, with reference annotations utilized for this purpose. Quantification of the total number of scratching train occurrences in each film and analysis of the distribution of the lengths of the scratching train were performed (Figure 2.1 part E and part F, Table 2.4).

The initiation of a scratching bout was characterized by the mouse initiation of lifting its hind paw and assuming a preparatory stance at the onset of a scratching bout. There were two scenarios seen for the conclusion of a train scratching. If the mouse did not lick its hindpaw after its most recent scratching episode, the conclusion of the sequence would occur when the mouse reestablished contact between its hindpaw and the ground. Conversely, if the mouse engaged in licking its hind paw after the aforementioned scratching session, the conclusion of the sequence would transpire when the mouse proceeded to place its hind paw into its mouth. During the manual annotation process, all the videos were seen by human annotators at the standard playback speed of 1X, which corresponds to 30 frames per second.

The beginning and final time points of each scratching train were documented manually and then transformed into frame numbers (30 frames per second) for further analysis. To generate the reference annotation, the videos were initially transformed into separate frames utilizing the OpenCV Python program (Open Source Computer Vision Library) (Bradski, 2000). The initiation and termination of each scratching train was set frame-by-frame. The frames within a train that were characterized by scratching actions were referred to as "scratching" frames, whereas frames that did not include scratching were referred to as "non-scratching" frames. In the model assessment, extended lickings (>60 frames) seen during scratching trains were marked as non-scratching frames for CQ cheek acute itch video 2 and SADBE chronic itch video 5. The annotation process for the group comparison was conducted in a double-blind fashion.

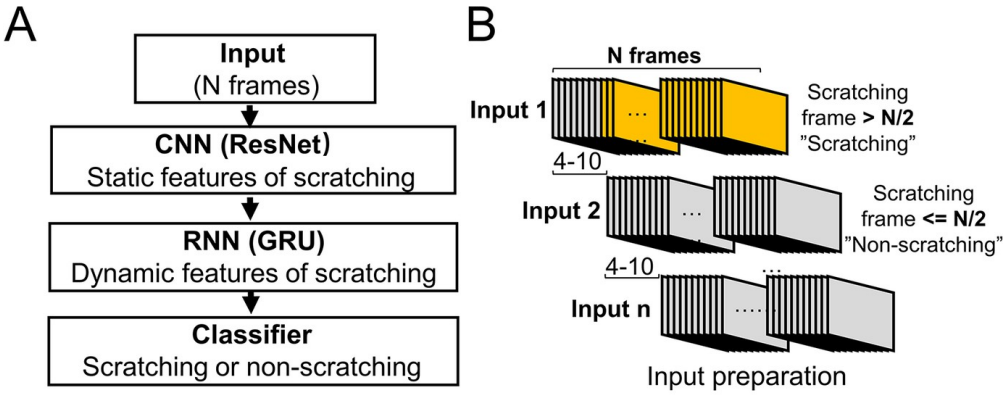
### 2.3.3 Deep learning neural network methodology

The scratching behavior of the mouse exhibited distinctive dynamic (temporal) and static (spatial) characteristics, which were emphasized by tracking the main body parts with DeepLabCut (Mathis *et al.,* 2018) (Figure 2.11).The rhythmic movement of the scratching hindlimb was one of the most obvious dynamic characteristics (Figure 2.11 A, B). Some unique static characteristics included relative positional relationships between the scratching rear limb and other body parts (Figure 2.11 C, D, E, F). To thoroughly capture these dynamic and static features, we designed a convolution recurrent neural network (CRNN) to take advantage of the various strengths of CNN and RNN (Figure 2.2 A and Figure 2.3 A). The CRNN contained a CNN (ResNet-18 (K. He *et al.,* 2016b)) (Figure 2.3 B) that extracts static features, such as the relative position of different body parts, an RNN (two-layer bidirectional gated recurrent unit (GRU) (K. He *et al.,* 2016b)) (Figure 2.3 C) that extracts dynamic features, such as the rhythmic movement of the scratching ResNet-18's CNN, was modified by replacing the final FC layer with a 256 embedding FC layer, as we only use CNN as a feature extractor. The RNN was composed of two bidirectional GRU layers with a hidden vector size of 512. The FC component was composed of two FC layers with embedding size 256 and ReLu activation, and embedding size 2 for the final prediction. In the final output vector, the prediction results were transformed into the utmost possible value.

The PyTorch framework was utilized to train the model (Paszke *et al.,* 2019). Hyperparameters for model training included setting the batch size as 16 or 32, depending on the size of the input. The maximum number of epochs was defined as 20. The ADAM optimizer was used in this study, with an initial learning rate of 10-4. The learning rate was then reduced by a factor of 0.3 every 5 epochs. The loss function employed was binary cross entropy. The dropout rate for the fully connected (FC) layer was configured to be 0.2. The model training was carried out on a bespoke desktop computer equipped with an Intel i9-10900k central processing unit (CPU) purchased from the online retailer Newegg. The system was further enhanced with 64 gigabytes (GB) of random access memory (RAM), specifically the CORSAIR Vengeance LPX 64GB variant also obtained from Newegg. Additionally, the desktop was equipped with an NVIDIA GeForce RTX 3090 graphics processing unit (GPU) boasting a 24 GB memory capacity, which was acquired from the popular online marketplace Amazon.

In the context of predicting models for new videos, the process of preparing the input was akin to that of the training dataset, with the exception that the neighboring inputs were limited to a separation of just 1 frame. The classification of each individual frame was determined as either "scratching" or "non-scratching" using the following rule: the prediction for the middle frame of each input was assigned the same label as the input prediction. The frames located at the initial or final positions of each video, which are unable to serve as the central frame of an input due to the length of the input N, were classified as either "scratching" or "nonscratching" by utilizing the prediction from the first or last input.

In order to generate the saliency map, the gradient value of each pixel was computed, and only the positive gradients were retained. Subsequently, the values were rescaled within the range of 0 to 1, using the methodology described in a previously published article (Selvaraju *et al.,* 2017b). Next, we proceeded to build the heat map by utilizing the gradient values. Specifically, any value below 0.1 was assigned transparency, values ranging from 0.1 to 0.6 were represented by a gradient from light blue to dark blue, and values over 0.6 were depicted as dark blue. Finally, we superimposed the heatmap on the original frame.




C

| Video No. | Dataset | Total video time (min) | Total scratching trains | Total scratching frames | Scratching time (%) |
|---|---|---|---|---|---|
| V1-V32 | Training | 640 | 875 | 262586 | 22.79 |
| V33-V40 | Test | 160 | 259 | 54630 | 18.97 |

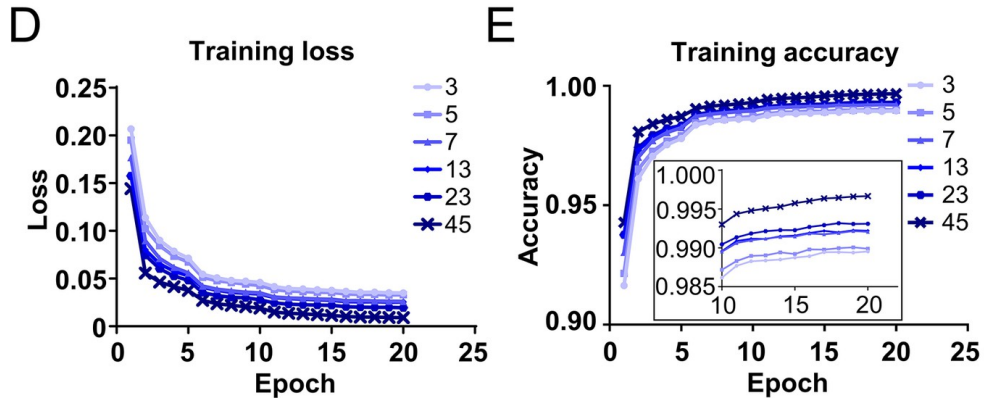



Figure 2.2: **Design and training of neural networks using deep learning.** (A) cartoon depicting the architecture of a deep learning neural network comprised of convolution neural networks (CNN), recurrent neural networks (RNN), and a classifier. (B) A cartoon depicting the compilation of training dataset inputs. N consecutive frames were chosen as one training input. Four to ten frames separated two adjacent inputs in a video. (C) The sample training and test datasets' information. During the training procedure with various input lengths (N = 3, 5, 7, 13, 23, 45 frames), the training loss decreased (D) while accuracy increased (E). The inset is a magnified portion of the figure.

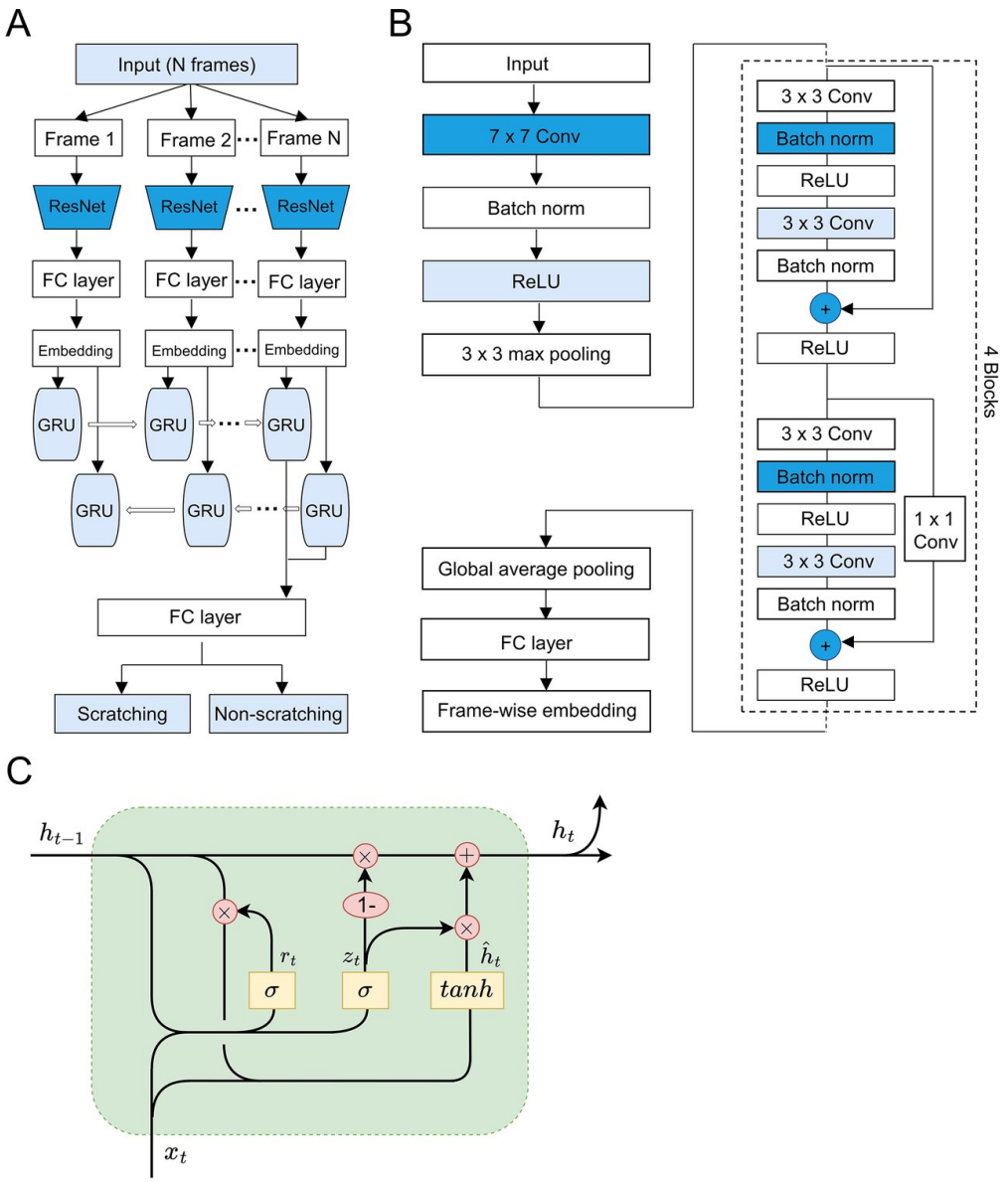



Figure 2.3: **The architecture of deep learning neural network.**(A) Image illustrating the structure of a convolution recurrent neural network (CRNN). First, convolution neural networks (CNN, ResNet-18) were provided the input, followed by a two-layer bidirectional gated recurrent unit (GRU). The results of the binary prediction were derived from the final full connection layer (FC). (B) Information about the modified ResNet-18 network. The final FC layer was altered to output frame-wise embedding as opposed to classificationmangel. convolution layer, or Conv; Batch norm, stratum of batch normalization; ReLU, rectified linear unit; +, vector element-wise addition. (C) Specifics of one GRU of the bidirectional, two-layer GRU. The unit connected to the embedding of the frame $t(x_t)$ used the output of the previous unit as input and produced a new output $(h_t)$. Inside the unit there were three yellow squares representing the reset gate, the update gate, and the candidate activation vector. x represents the Hadamard product for vectors; + represents the vector plus.

## 2.4 Results

### 2.4.1 Model training procedure

We divided the 40 videos into two groups. Specifically, we assigned 80% of the recordings (32 videos from 8 mice) to the training dataset, while we assigned the remaining 20% (8 videos from 2 mice) to the test data set (Figure 2.2 C). We converted each movie into separate frames and then classified each frame as either "scratching" (occurring inside a scratching train) or "non-scratching" (occurring outside of a scratching train) based on our reference annotation. To prepare the training dataset, we altered a parameter to choose N consecutive frames. We did this to optimize the performance of the model and capture the dynamic aspects of scratching, as shown in Figure 2.2 B. To avoid extensive duplication within the training dataset, we deliberately separated 4 to 10 frames between two consecutive inputs (Figure 2.2 B). We classified the input as "scratching" (class 1) if more than 50% frames (N/2) in the input showed scratching behavior. Otherwise, we classified it as "non-scratching" (class 0) (Figure 2.2 B).

For cross-validation, we subjected the training and test videos to rotation. We categorized the test videos into five groups, namely V1-V8, V9-V16, V17-V24, V25-V32, and V33-V40. We assign the remaining films for training. Our approach to generating input for model training was as follows: We first convert the videos into discrete frames. Then we selected a set of N consecutive frames (where N could be 3, 5, 7, 13, 23, or 45) as a single input. Our selection method ensured a consistent spacing of 4-10 frames between each adjacent input. We then classified each input as "scratching" (class 1) if the number of frames showing scratching behavior exceeded half of the total frames. On the other hand, we classified inputs as "non-scratching" (class 0) if the number of scratching frames did not exceed half of the total frames.

We converted the frames inside each input into grayscale images and resized them to dimensions of 300x300. After that, we applied a square-shaped crop ranging from 288X288 to 300X300. We then executed a stochastic operation that involved horizontal and vertical flipping, setting the probability of occurrence at 0.5. After this, we modified the frames to have a resolution of 256X256 before integrating them into the CRNN network.

The input length training parameter (N frames) is vital because the dynamic properties

of the scratching behavior span multiple frames. In the CQ-induced acute nape itch model, we observed that the average duration of a single cycle of scratching and paw licking was around 30 frames, as seen in Figure 2.12 A, B. We conducted experiments to assess the efficacy of training the model using input sequences of various lengths, ranging from 3 to 45 frames. Throughout our training process, we noticed a significant decrease in the loss. Furthermore, the prediction accuracy, measuring the correct identification of both scratching and nonscratching frames out of all frames, increased (Figure 2.2 D, E). After completing 10 epochs, which means that the training process covered the entire training data set once, the accuracy of the model reached a plateau, as seen in Figure 2.2 E. The prediction precisions for all input lengths were above 0.98, with a slight improvement observed as the input length increased (Figure 2.2 E). The results of our study show that the CRNN network, as we constructed it, effectively captures the unique characteristics of scratching and accurately identifies scratching activity within the training data set.

### 2.4.2 The assessment of model performance on test datasets

The performance of the prediction models that were trained was evaluated using a set of 8 test videos that had not been previously seen. Similarly to the aforementioned description, each test video was transformed into inputs including a certain number of frames denoted "N". It is important to note that the same value of "N" was utilized for both the training and test phases. However, it should be highlighted that the two consecutive inputs were separated by a mere single frame. Furthermore, the neural network that underwent training successfully made predictions for each input, classifying them as either "scratching" or "nonscratching". To transform the prediction from a composite input consisting of N frames into individual frame predictions, we used the following approach: the forecast for the middle frame within each input was assigned identical values as the prediction for the entire input. For instance, in the event that an input is classified as "scratching," the frame in the middle of this input would correspond to a "scratching" frame. The aforementioned analysis successfully classified each frame of the tested videos as either "scratching" or "nonscratching," with the exception of a small number of frames located at the start or conclusion of a movie. Please refer to the methodology section for further details on how missing data was handled in this context. Furthermore, the recall, precision and F1 score evaluation metrics were calculated.

Recall is defined as the ratio of properly predicted scratching frames to the number of reference scratching frames. Precision is the ratio of correctly predicted scratching frames to the total number of predicted scratching frames. F1 score is a measure that combines recall and precision, calculated as 2 times the product of recall and precision divided by their sum. Compared to the general accuracy metric, the recall, precision, and F1 score provide a more comprehensive and detailed assessment of the performance of a model, particularly in cases where scratching is infrequent within a video (Powers, 2020).

In order to exclude the potential influence of a specific pairing of training and test data sets on the favorable performance of our models, we implemented a cross-validation approach by rotating the training and test movies. F1 scores for all various combinations of training and test videos consistently exceeded 0.9, as seen in Figure 2.13 A, B. This observation provides evidence for the consistent and superior performance of our prediction models. Furthermore, it can be observed that the prediction model exhibited improved performance when the input length was extended, as depicted in Figure 2.13 C. The model that exhibited the highest performance, namely the one trained using videos 1-32 with an input length of 45 (N=45), was chosen for further analysis and experimentation.

The top model exhibited an average recall of 97.6% and precision of 96. 9% in the eight test movies (Figure 2.4 A). Furthermore, the recall and precision for individual videos above 95% in the majority of cases (Figure 2.4 B). The performance of the automated annotation system was comparable to, or maybe even superior to, that of manual annotation. Manual annotation achieved an average recall of 95.1% and an average precision of 94. 2% (Figure 2.4 C, D). When evaluating the scratch duration with respect to the reference annotation, the prediction of the model exhibited an average disparity of 1. 9%, while the manual annotation showed a discrepancy of 2.1% (Figure 2.4 E). The correlation coefficient between the model's prediction and the reference annotation was found to be 0.98, which is consistent with the findings obtained from the manual annotation (Figure 2.4 F). Upon analyzing the probability traces of the model's predictions and the reference annotation, it was observed that the model exhibited a high level of accuracy in identifying the majority of scratching trains in the test videos. Furthermore, the model predictions regarding the initiation and termination of each scratching train were found to be in good agreement with the reference

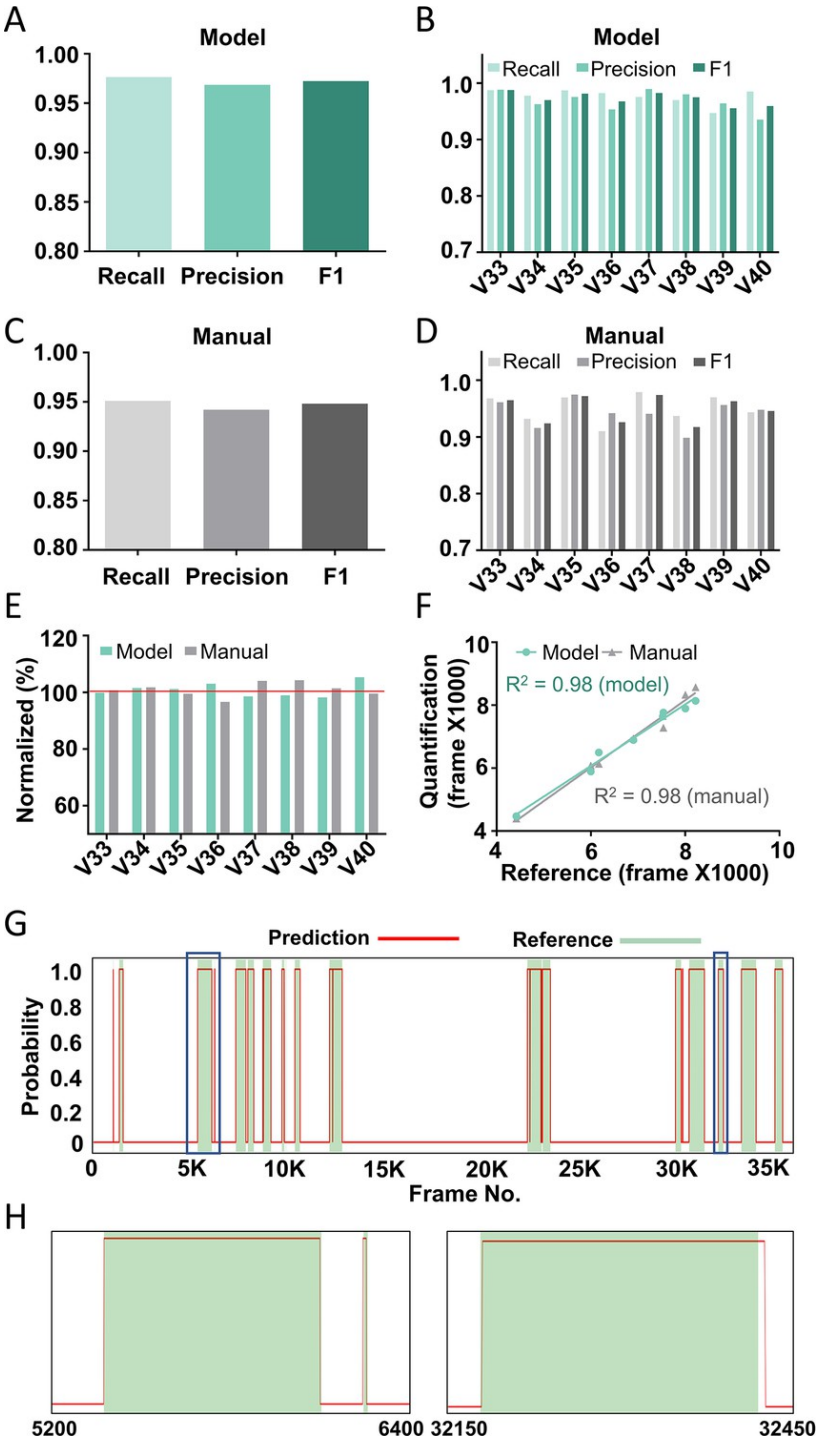



Figure 2.4: **The top model's performance on test videos.** The top model's average recall, precision, and F1 score (A) or in individual videos (B). The average recall, precision, and F1 score of manual annotation (C) or for specific videos (D). Comparison between model prediction, manual quantification, and reference annotation. Red line indicates that the reference annotation has been normalized to 100 percent. Correlations between model prediction or human quantification and reference annotation. Pearson's correlation coefficient, $R^2$. (G) An illustration of a scratching probability trace predicted by the model and aligned with the reference annotation (green bar). (H) The two zoomed-in portions of figure (G) demonstrating the excellent alignment between the model prediction and the reference annotation.

annotation. This is illustrated in Figure 2.4 G and H. Collectively, these findings provide evidence of the robustness and precision of our methodology to identify and measure mouse scratching behavior in recently recorded videos.

### 2.4.3 Neural network recognized mouse scratching by focusing on the hind paw

In what manner did the trained neural network model discern mouse scratching behavior and differentiate it from other behaviors? Deep learning neural networks are often perceived as black boxes due to their complex internal workings. However, saliency maps have been shown to provide valuable information (Selvaraju *et al.,* 2017b). These maps visually represent the areas of each frame (pixels) that were mostly utilized throughout the model prediction process. The primary focal points of interest were observed in relation to the scabby hindpaw within the scratching frames (Figure 2.5 , indicating that the prediction model prioritized the characteristics associated with the scabby hindpaw. In certain scratching frames, additional body components were also emphasized, specifically the two front paws (Figure 2.5 B). This observation suggests that the model also incorporated the spatial connection of these body parts to identify scratching activity. In contrast, with regard to other mouse activities that do not involve scratching, such as wiping, grooming, rearing and locomotion, the significant aspects did not show a distinct connection with certain regions of the body of the mouse (Figure 2.5 C-F and Figure 2.17 B-E). As a whole, these saliency maps demonstrate that the neural network, after being trained, has the ability to prioritize both the dynamic and static characteristics of scratching in order to accurately predict mouse scratching activity.

### 2.4.4 Analysis of Prediction Errors for the CRNN model

To gain a deeper understanding of the performance of the best trained neural network model, we comprehensively examined its prediction errors in 8 test videos (Figure 2.6) and contrasted them with those from manual quantification (Figure 2.14) and other trained models (Figure 2.15). Type 1, false positive (nonscratching region was predicted as a scratching train); Type 2, false negative (a real scratching train was not recognized); Type 3, blurred boundary (the prediction of the start or end of a scratching train was shifted); Type 4, missed

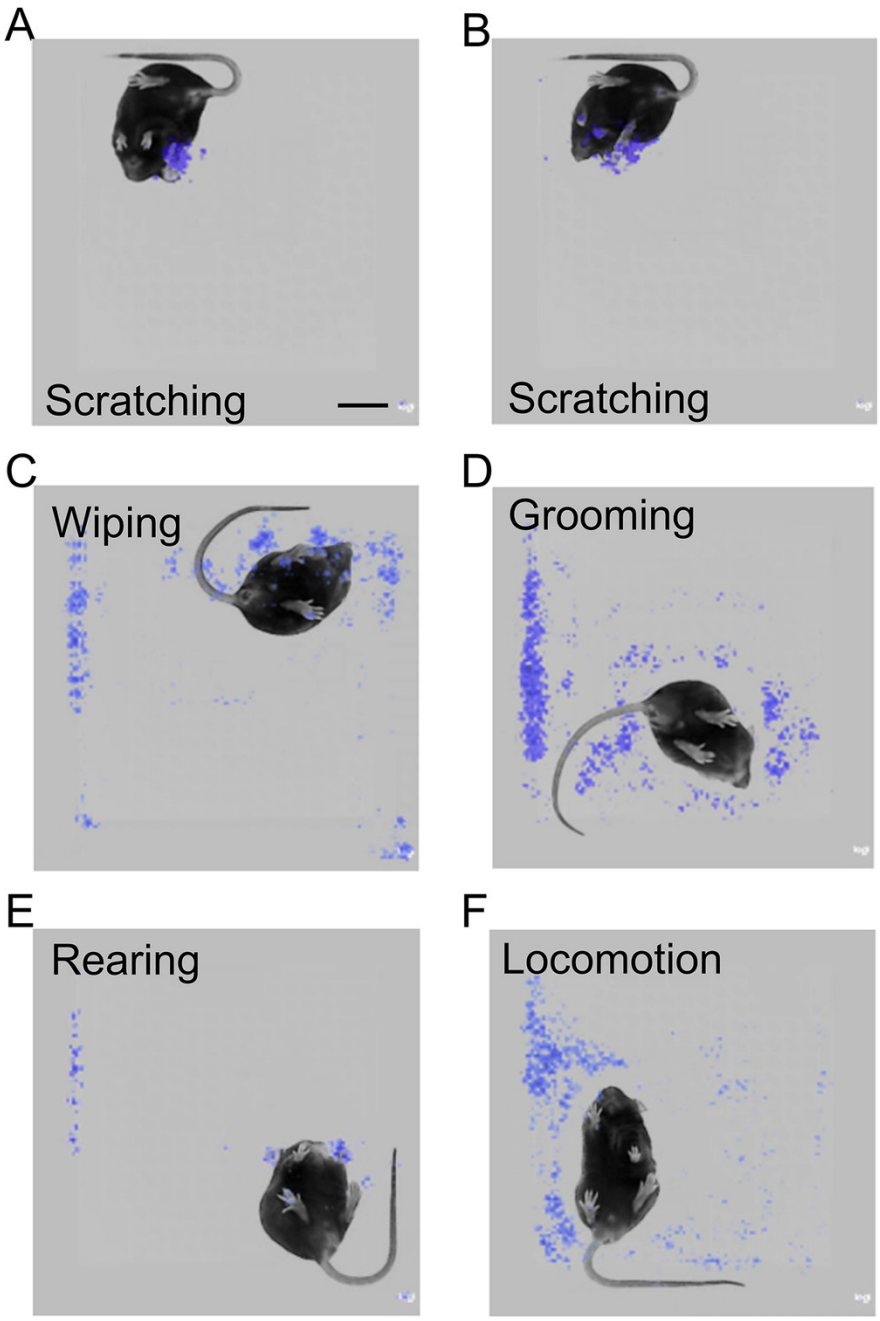



Figure 2.5: **For scratching behavior identification, the prediction model focused on the scratching hind paw.** (A, B) Saliency map displaying the gradient value of each pixel of clawing frames during the best model's prediction of mouse scratching behavior. The model highlighted the scratching rear paw (A, B) as well as other body regions, such as the front claws (B).Scale bar measures 2 cm. Saliency map displaying the gradient value of each pixel during mouse clawing behavior prediction during cleaning (C), grooming (D), rearing (E), and locomotion (F) frames.

interval (two or more adjacent scratching trains were predicted as one scratching train); and Type 5, split scratching train (one scratching train was predicted as two scratching trains). We found that the predominant prediction error of the trained neural network model was a type 3 error, accounting for approximately 3% of the total scraping frames (Figure 2.6 B), followed by Type 2 and Type 5 errors, each accounting for approximately 1%. For manual quantification, Type 3 and Type 4 represented 10% and 8%, respectively, of the total number of scraping frames (Figure 2.14 A).

For the Type 1 error of the prediction of the model (Figure 2.6 C1-C3), the durations of all false positive scratching trains were less than 10 frames (0.3 s) (Figure 2.6 C2) and momentarily close to a real scratching train (within 30 frames, 1 s) (Figure 2.6 C3). They were not confused with other behaviors such as wiping, grooming, rearing, locomotion, or resting (Figure 2.15). Type 1 errors were also rare during manual annotation (Figure 2.14 B).

The models might overlook brief scratching trains, hence causing the Type 2 error. In fact, every delayed scraping train was less than 40 frames (1.3 s) (Figure 2.6 D1, D2). 18.5% of all scraping trains lasting less than 30 frames (1 second) were not predicted by the model. This percentage dropped to 2.7% for scraping trains lasting between 30 and 60 frames (1-2 seconds). Scratching trains that were longer than 60 frames ($>$ 2 s) were not missed (Figure 2.6 D3). The correlation between the Type 2 error and the input length (N) of the prediction models was positive. It became zero or very close to zero when models were trained with input lengths of 3, 13, and 23 frames (Figure 2.16 A-C). For manual annotation, the Type 2 error was not common (Figure 2.14 C1, C2).

The type 3 error (Figure 2.6 E1-E3) predominated the other four types of errors (Figure 2.6 E1-E3). The average start and end frame shift for the model prediction was 2.2 and 7.0 frames (Figure 2.6 E3), while they were 11.5 and 12.8 frames for the manual annotation (Figure 2.14 D1, D2). The start and end frame shifts of manual quantification were comparable (350 ms), likely reflecting the processing latency of the human visual system in real time. Temporal shifts were smaller for model prediction, which was a post-event, frame-by-frame process, than for human visual processing. In addition, the model recognized the beginning of a striking train more accurately than its conclusion (Figure 2.6 E3). This may repre-

sent the characteristic of striking trains. It was relatively easy to determine when a mouse started a scratching train by lifting its hind paw, but it was more difficult to determine when a mouse completed a scratching train by placing its hind paw back on the floor. There was no correlation between the start and end shift and the duration of a scraping train (Figure 2.16 D). Consequently, the relative error (percentage of error frames) would decrease as the duration of a scratching train grew. In fact, the precision of the prediction (as indicated by the F1 score) was positively correlated ($R^2 = 0.5723$) with the average duration of the scraping train in a video (Figure 2.16 E).

The type 4 error occurred when two adjacent scratching trains were too close together and were incorrectly predicted as a single scratching train (Figure 2.6 F1-F3). All missed intervals were less than 30 frames (1 s) (Figure 2.6 F2). On the contrary, 51. 4% of the intervals between adjacent scraping trains shorter than 30 frames were not recognized. All intervals that exceeded thirty frames were identified (Figure 2.6 F3). Type 4 errors were more prevalent in manual annotations than in model predictions (Figure 2.13 E1, E2).

An error of type 5 occurred when one scratching train was predicted as two or more scratching trains with incorrectly predicted intervals separating them. The average length of these incorrectly predicted intervals was approximately 10 frames based on model prediction and approximately 40 frames based on manual annotation (Figure 2.6 G1, G2, and Figure 2.14 F). More than 80% of these intervals were within or partially overlapped with a paw lick phase (Figure 2.6 G3), especially when the paw lick phase lasted more than 30 frames (Figure 2.16 F). Consequently, it appears probable that the model predicted some extended lapping frames within a scratching train as "nonscratching." The errors of types 4 and 5 reflect the inherent complexity of the clawing behavior. Human definitions and consensus in the field, such as using 2 seconds as the threshold between two adjacent scratching trains (Darmani & Pandya, 2000), would aid in the reduction of these types of errors.

Therefore, we have developed a novel system that combines a customized video recording box and a well-trained CRNN neural network (N45) to accurately and autonomously identify and quantify mouse clawing behavior. We call it Scratch-AID system.

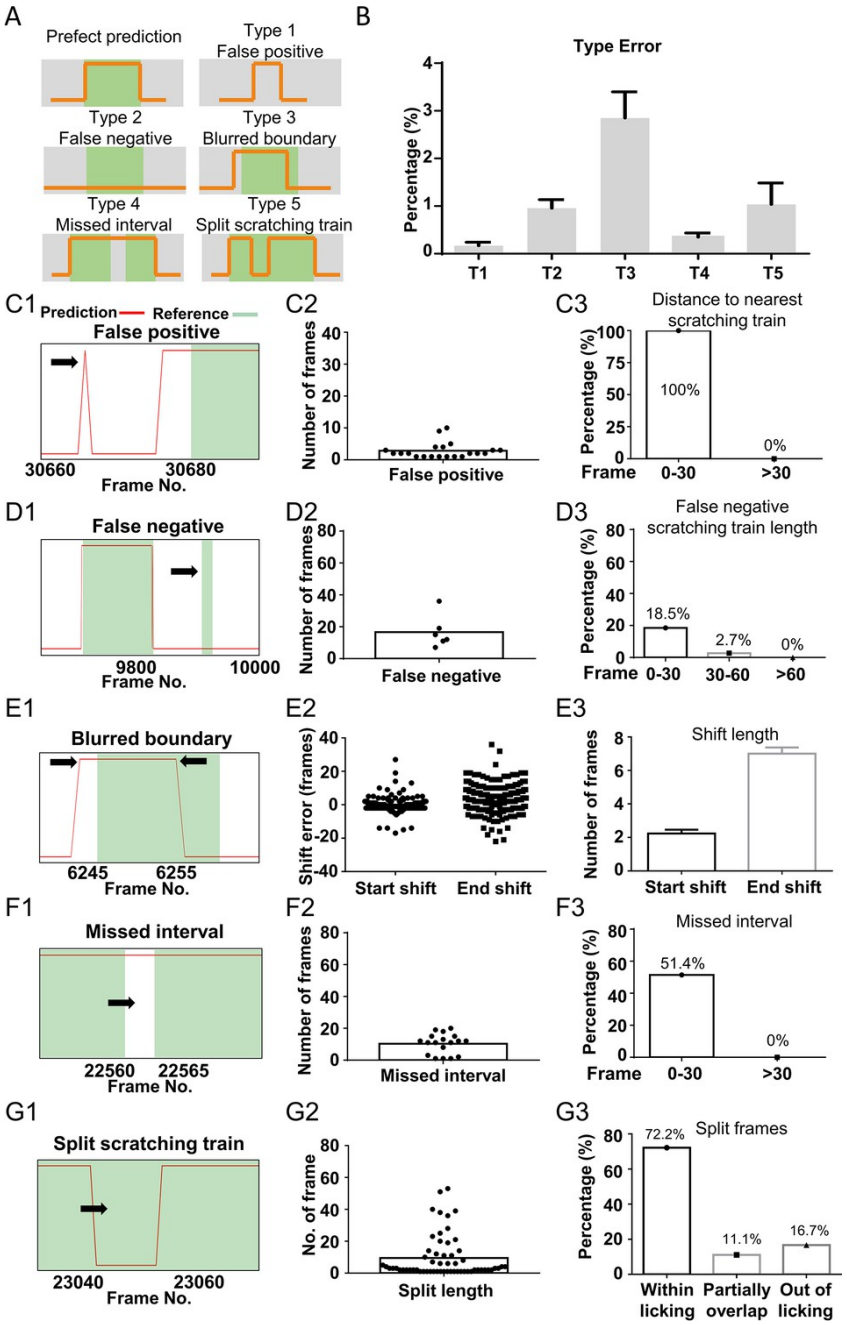



Figure 2.6: **Analysis of the top prediction model's error rates.** (A) Cartoon depicting five prediction mistakes. Red curves showed the predicted scratching probability from the model, while green bars showed the reference scratching trains. (B) The incidence rate of each type of mistake, obtained by dividing the total frames in each error by the total scratching frames of 8 test videos. SEM error bar. A real example of false positive prediction (C1), the duration of all false positive scratching trains (C2), and the frequency distribution of distances between them and the next real scratching train (C3). A actual false negative prediction (D1), the duration of all false negative scratching trains (D2), and the Type 2 error rate for scratching trains of different lengths (D3). Real example of blurred boundary prediction (E1), distribution (E2), and start/end shift average lengths (E3). SEM error bar. Actual missed interval (F1), duration of all missed intervals (F2), and Type 4 error rate for intervals of varying durations (F3). A actual split scratching train (G1), split frame length (G2), and Type 5 error distribution from paw licking (G3).

### 2.4.5 Performance of the Scratch-AID system on other major acute itch models

In addition to the nape, the cheek is another prevalent site for inducing itching in mice (Shimada & LaMotte, 2008). To test whether the Scratch-AID system trained by the nape CQ model could also recognize and quantify the scratching behavior of the cheek model, the collaborator injected 200 g of CQ in 15 l of saline into the cheeks of 5 wild-type mice (3 males and 2 females) and recorded 7 videos (Figure 2.7 A). Then we compared the Scratch-AIDuantification of scratching behavior with manual annotation. ScratchAID's prediction had 93.4% recall, 94.8% precision, and an F1 score of 0.941 (Figure 2.7 B), while manual quantification recall, precision, and F1 score were 96.0%, 88.6%, and 0.919 (Figure 2.7 C). Figures 2.7 D, E illustrate the correlation between the Scratch-AID prediction and the reference annotation, which was 0.9926, and that of the manual quantification, which was 0.9876. The total scratching time in individual recordings predicted by a model and manually annotated was comparable to the reference annotation (Figure 2.7 F). These findings indicate that the Scratch-AID system can reliably identify and quantify clawing behavior induced by an acute irritation sensation in the cheek.

Different pruritogens administered to the same body site elicit clawing behaviors with different dynamic characteristics (Wimalasena *et al.,* 2021). Therefore, we examined whether the Scratch-AID system could recognize clawing behavior induced by a distinct pruritogen, histamine, when trained with CQ injections. 100 g of histamine (in 15 $\mu$l of saline) was injected intradermally into the nape and four videos were captured (Figure 2.7 G). The recall, precision, and F1 score of the Scratch-AID prediction were 96.6%, 90.91%, and 0.936 (Figure 2.7 H), while they were 96.3%, 80.5% and 0.877 (Figure 2.7 I) for manual annotation. Figure 2.7 J, K illustrates the correlation between the Scratch-AID prediction and the reference annotation, which is 0.9707, and that of the manual quantification, which is 0.9895. The total scratching time in individual recordings of both model prediction and manual annotation was similar to that of reference annotation (Figure 2.7 L). Although trained only with the CQ nape acute itch model, our Scratch-AID system can recognize and quantify the scratching behavior of acute itch models induced at different skin locations or triggered by different pruritogens, and its prediction accuracy is comparable to that of manual annotation.

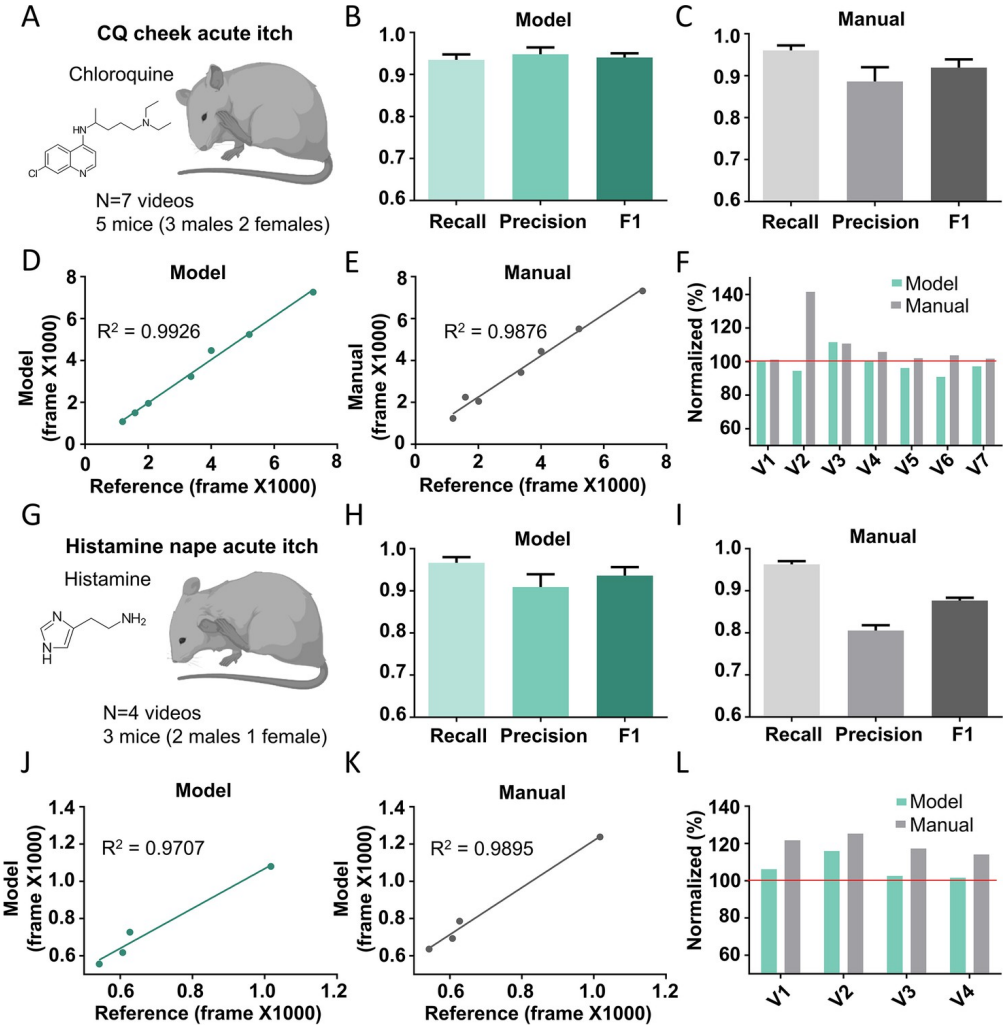



Figure 2.7: **The Scratch-AID (Automatic Itch Detection) performance on other acute itch models.** (A) A cartoon showing an acute itch model induced by chloroquine (CQ) injection in the mouse cheek. Average recall, precision, and F1 score of Scratch-AID (B) or manual annotation (C). Error bar, standard error of the mean (SEM). The correlation between model prediction (D) or manual quantification (E) and reference annotation. $R^2$, Pearson correlation coefficient. (F) The comparison among model prediction, manual quantification, and reference annotation. The reference annotation is normalized to 100% shown as the red line. (G) Cartoon showing an acute itch model induced by histamine injection in the mouse nape. Average recall, precision, and F1 score of Scratch-AID (H) or manual annotation (I). Error bar, SEM. The correlation between model prediction (J) or manual quantification (K) and reference annotation. $R^2$, Pearson correlation coefficient. (L) The comparison among model prediction, manual quantification, and reference annotation. The ref between annotation is normalized to 100% shown as the red line.

### 2.4.6 Performance of the Scratch-AID system on a chronic itch model

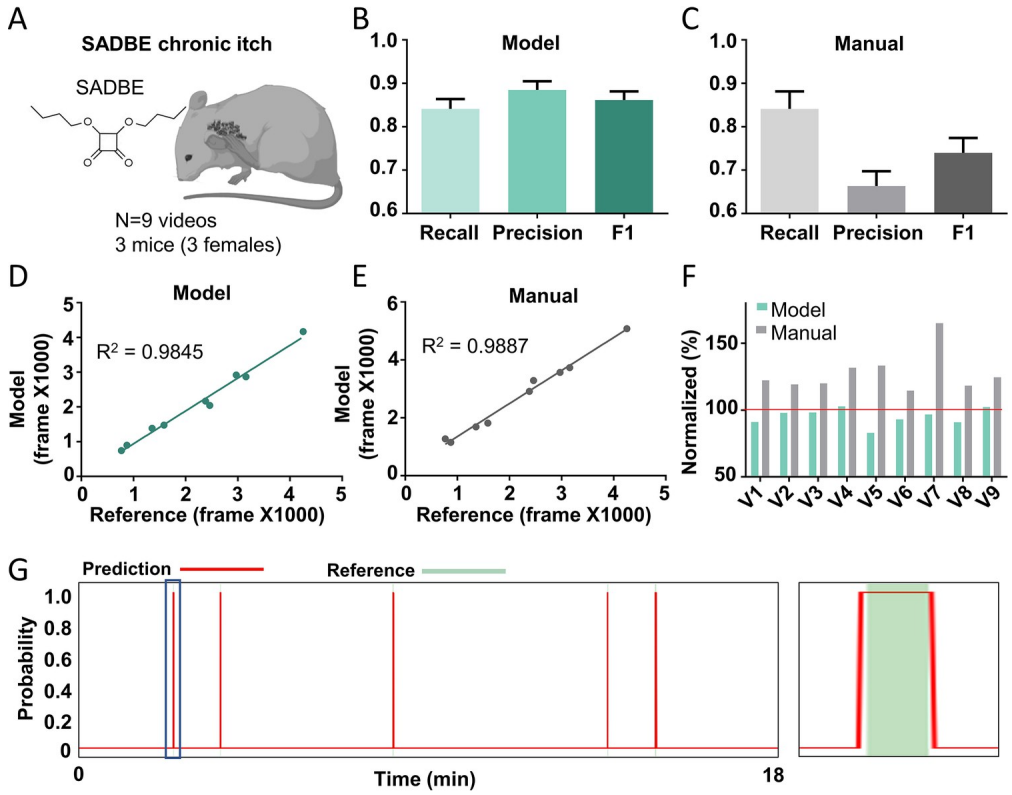



Figure 2.8: **The Scratch-AID performance on a chronic itch model.** The cartoon in (A) illustrates a squaric acid dibutylester (SADBE) induced chronic itch model. The performance of Scratch-AID (B) and manual annotation (C) is evaluated by the average recall, precision, and F1 score, with the standard error of the mean (SEM) shown as error bars. The correlation between model prediction (D) and manual quantification (E) with the reference annotation is measured by the Pearson correlation coefficient ($R^2$). (F) compares the model prediction, manual quantification, and reference annotation, with the reference annotation normalized to 100%. (G) shows an example of the scratching probability trace (red curve) predicted by the model and aligned with the reference annotation (green bar). A zoom-in (right panel) of the blue square part reveals a good alignment of the model prediction with the reference annotation.

Chronic itching is a debilitating symptom that has a detrimental impact on patient quality of life (Yosipovitch & Fleischer, 2003; H. Yu *et al.*, 2021). It is essential to investigate the underlying mechanisms using the mouse model in order to develop novel treatments for chronic itching caused by a variety of conditions. To determine whether the Scratch-AID system could be used to study mouse chronic itch models, the collaborator created a model of chronic contact dermatitis itch induced by squaric acid dibutylester (SADBE) (Beattie *et al.*, 2022; Qu *et al.*, 2015) and recorded 9 videos from 3 wild-type mice (Figure 2.8 A). Affected

mice demonstrated spontaneous clawing of the cranium and / or nape. In particular, the dynamic features of spontaneous scratching behavior under this chronic itch condition were different from those exhibited by the CQ acute itch model: the total scratching time was shorter for the same time period (20 min) (Figure 2.9 A), and the average duration of the scratching trains was shorter (53 frames on average) than the CQ-induced acute scratching behavior (280 frames on average) (Figure 2.9 B and Figure 2.1 F). Despite these substantial differences, the recall, precision, and F1 score of Scratch-AID prediction were 84.1%, 88.5%, and 0.862, respectively (Figure 2.8 B, C), compared to 84.1%, 66.3% and 0.740 for manual annotation. The dominant brief clawing trains in this chronic itch model (Figure 2.9 B) likely contributed to the decreased recall and accuracy of both model prediction and manual annotation. The correlation between the prediction of the model and the reference annotation was 0.9845 (Figure 2.8 D), which was comparable to the correlation between the manual annotation and the reference annotation (0.9887; Figure 2.8 E). The model's estimate of the total etching time was marginally more precise than the manual annotation (Figure 2.8 F). The Scratch-AID system was able to capture brief low-frequency scratch trains based on prediction traces (illustrated in Figure 2.8 G).

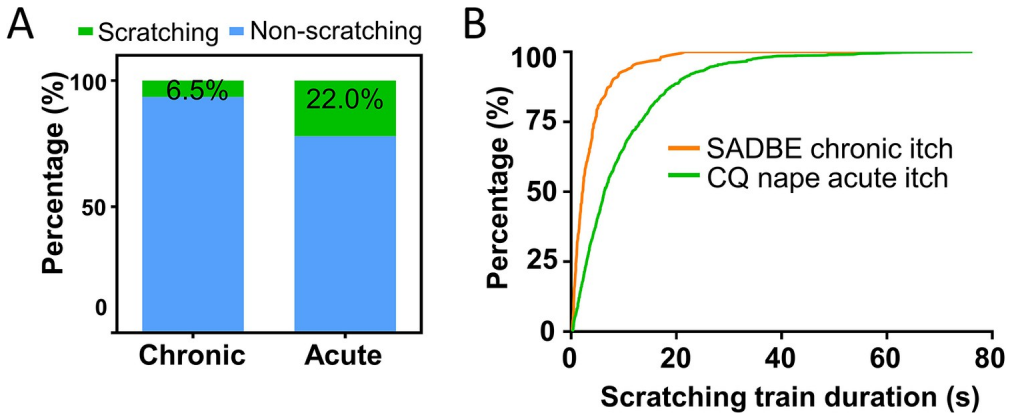



Figure 2.9: **Different dynamic features of chronic and acute itch models.** (A) The percentage of scratching and non-scratching frames in the squaric acid dibutylester (SADBE) chronic itch model (n = 9 videos) and chloroquine (CQ) nape acute itch model (n = 40 videos). (B) Frequency distribution of scratching train duration of SADBE chronic itch model and CQ nape acute itch model.

### 2.4.7 Application of the Scratch-AID system in anti-itch drug screening

Finally, we examined whether the Scratch-AID system could be used to evaluate antiitch medications. The collaborator used the histaminergic acute itch nape model and pretreatment with Benadryl (Loew *et al.,* 1946), an FDA-approved antihistaminergic itch medication. 1 hour before intradermal injection of histamine (200 g in 15 l of saline), Benadryl or a placebo lotion was administered topically to the nape skin of mice. Six wild-type C57 mice (2 male, 4 female) in the Benardryl-treated group and seven wild-type C57 mice (2 male, 5 female) in the control group scratched for 20 minutes (Figure 2.10 A). Quantification of the total clawing time (frames) using Scratch-AID demonstrated a significant reduction with Benardryl treatment (Figure 2.10 B). Figure 2.10 C shows that manual annotation yielded comparable results. These results suggest that the Scratch-AID system is sensitive enough to detect the change in clawing behavior after an antiitch drug treatment, highlighting the potential application of the Scratch-AID system in large-scale, high-throughput antiitch drug assays.

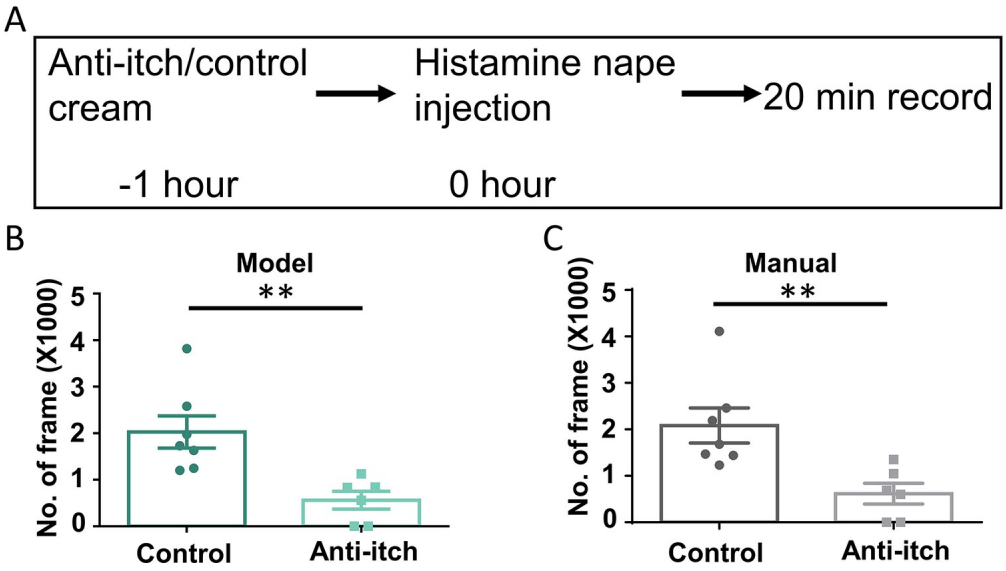



Figure 2.10: **Application of the Scratch-AID (Automatic Itch Detection) system in a drug screening paradigm.** (A) A diagram showing the experimental design of an anti-itch drug test. Quantification of scratching behavior in anti-itch cream treated group or control group by Scratch-AID (B) or manual annotation (C). Error bar, standard error of the mean (SEM). Differences between the two groups were analyzed using unpaired two-tailed Student's t-test, ** $p < 0.01$.

## 2.5 Discussion

Scratching is a behavior unique to itch, and the mouse is the primary model animal used to investigate itch mechanisms and develop new anti-itch medications. In this study, we created a new system, Scratch-AID, consisting of a customized video recording box and a well-trained neural network, for highly accurate automatic quantification of mouse claw behavior. Its efficacy is comparable to the manual annotation on main itch models and an anti-itch drug screening paradigm. To the best of our knowledge, this is the first system based on deep learning that can achieve such high accuracy and generalization.

It is remarkable that a model trained with videos of CQ-induced acute nape itch could reliably recognize scratching behavior in other itch models, regardless of body sites (nape vs. cheek) or the dynamics of scratching behavior (e.g., acute vs. chronic itch, distinct pruritogens). The remarkable performance and generalization of the Scratch-AID system is likely attributable to the high quality and reproducible video recording, the large number of high quality training datasets with frame-by-frame annotation, the efficient design of the CRNN deep learning neural network, and the optimizations of training parameters. Variable video recording conditions are a significant barrier that has prevented various laboratories from adopting a trained neural network. Videos captured under different conditions (illumination, field size, image resolution, magnification of rodents, image angle, backgrounds, etc.) typically do not perform well with trained models. To provide a reproducible and high-quality recording environment, we built a custom video recording enclosure (Figure 2.1 B). This helps to standardize the video recording procedure, reduce noise, and facilitate the stable performance of trained deep learning models. Meanwhile, this video recording device is simple to install, scale, and use. Using a well-trained prediction model with performance comparable to manual quantification, Scratch-AID is prepared to supplant manual quantification. However, it must be pointed out that when the camera used for shooting, the shooting environment, or the lighting changes, the accuracy of the model will decrease. This emphasizes the importance of maintaining consistent conditions during video recordings to ensure the utmost accuracy of the model predictions. Any significant deviation from the camera setup, environment, or lighting can potentially lead to misinterpretations or false predictions. Therefore, users of Scratch-AID need to be cautious and aware of these limita-

tions. We will come back to discuss how to solve this problem in a future work part.

The quantity and quality of training datasets are crucial for building a highly accurate and generalized model. The efficacy of a prediction model is typically proportional to the extent of the training data set. The CQ-induced acute nape itch model, which elicited robust clawing behavior in rodents, was selected for model training and evaluation in this study. The large number of scratching videos and the high frequency of desired behavior in each video provided a high-quality training set. In addition, the precise definition of scraping train and frame-by-frame annotation of videos were essential for training, testing, and error analysis.

Our research demonstrates that ResNet and GRU are an effective combination of deep learning architectures to analyze animal behavior. In addition, our network architecture is efficient so that the accuracy plateau was reached after only 10 training epochs. There is still potential for improvement in the efficacy of our forecasting models. To improve Scratch-AID's capacity to capture short scratching trains, we could train the CRNN neural network with recordings of chronic itch models scratching. In addition, optimizing training parameters could help improve prediction precision. Changing the input length resulted in various types of error (Figure 2.6 B and Figure 2.16 A-C). The number of Type 2 and Type 4 errors increased as the input length increased, while the number of Type 1, Type 3, and Type 5 errors decreased. Consequently, optimizing and selecting the optimal input length for various irritation models is a trade-off. Increasing the extent of the training dataset may also help in the development of more accurate prediction models. By checking the videos with relatively low prediction accuracy, such as video number 5 (V5) in the chronic itch model (Figure 2.8 F), we discovered that the missed scratch frames could be due to a rare posture during scratching behavior in which the hind paw scratching was partially obscured by the tail. Therefore, with more training videos comprising uncommon scratching postures, trained neural network models could be more "knowledgeable" about the diversity of scratching behavior.

Understanding the underlying molecular, cellular, and circuit mechanisms requires the quantification of animal behavior. Automatic deep learning analysis will not only increase efficiency and accuracy compared to manual analysis, but will also reduce human bias and

errors. Along these lines, we created the Scratch-AID system to automatically quantify the scratching behavior of rodents. Our research also contributes to the development of new deep learning neural network models to automate the analysis of other animal behaviors.

## 2.6 Future work

1. Exploration with 3D CNN: While our current model leverages a convolution recurrent neural network (CRNN) for video-based behavior quantification, a promising avenue of exploration is the use of 3D convolution neural networks (3D CNNs). These networks have the potential to effectively capture spatiotemporal features within video data, providing a rich understanding of the motion dynamics of the scratching behavior of mice. We aim to train a 3D CNN from scratch and subsequently compare its performance with our existing CRNN model. Such an analysis will offer insights into the relative strengths and weaknesses of both architectures in capturing and quantifying mouse behavior.

2. Harnessing Foundation Models in Visual Tasks: The rise of foundation models, particularly those built on the transformer architecture, has reshaped the landscape of computational tasks. The Vision Transformer (ViT) and its subsequent iterations, such as ViT-G and models like CoCa, which integrate convolution and transformer layers, have shown exemplary performance in various visual tasks. However, applying these models to different domains such as our mouse itch detection poses significant challenges. Their massive size demands considerable resources, and their generic training often makes them unsuitable for specialized tasks without extensive tuning. However, we see potential in harnessing their capabilities.

One of the promising methodologies to address the aforementioned challenges is the low-rank adaptation (LoRA) technique. Developed by Microsoft Research, LoRA presents an innovative approach to model fine-tuning. Instead of updating the entire model, LoRA focuses on updating a specialized bypass layer, drastically reducing the number of parameters to be modified. This method could potentially enable us to capitalize on the foundational knowledge of these transformer-based models while ensuring that they are specialized enough for our specific tasks. We believe LoRA's adaptation strategy can be a game-changer, offering an efficient way to bridge the gap between large-scale foundation models and the specific

requirements of our tasks.

## 2.7 Data Availability

The training and test videos generated during the current study can be downloaded from DRYAD:

(https://datadryad.org/stash/share/IszMIeDQoqXYne6DA01lcyPO0IZwdRsJRCdKQEvmjow).

The codes for model training and testing can be downloaded from GitHub:

(https://github.com/taimeimiaole/Scratch-AID)

## 2.8 acknowledgments

I deeply appreciate every member of the Luo and Ma labs for being our pillars of support throughout this journey. A special mention goes to Dr. Ji Zhu, Mr. Simin Liu, Dr. You Lv, Ms. Yakun Wang, and Dr. Wei Yang, whose invaluable insights and suggestions significantly shaped our project. The heartfelt gratitude to Dr. Arsuaga for supporting us with the NSF grant (DMS-1854770), to Dr. Luo for the NIH R01 (NS083702), and to both Drs. Ding and Luo for the R34 (NS118411). Your faith in our work has been a driving force behind its success.

## Supplementary Material

| Block name | Layer name | input size | output size | channels | parameters | stride |
|---|---|---|---|---|---|---|
| Input | conv1 7x7 | 256, 256 | 128,128 | 64 | 3,136 | 2 |
| | max pool 3x3 | 128, 128 | 64, 64 | 64 | 0 | 2 |
| Residual 1 | conv1.1 3x3 | 64, 64 | 64, 64 | 64 | 36,864 | 1 |
| | conv1.2 3x3 | 64, 64 | 64, 64 | 64 | 36,864 | 1 |
| Residual 2 | conv2.1 3x3 | 64, 64 | 64, 64 | 64 | 36,864 | 1 |
| | conv2.2 3x3 | 64, 64 | 64, 64 | 64 | 36,864 | 1 |
| Residual 3 | conv3.1 3x3 | 64, 64 | 32, 32 | 128 | 73,728 | 2 |
| | shortcut 1x1 | 64, 64 | 32, 32 | 128 | 8,192 | 2 |
| | conv3.2 3x3 | 32, 32 | 32, 32 | 128 | 147,456 | 1 |
| Residual 4 | conv4.1 3x3 | 32, 32 | 32, 32 | 128 | 147,456 | 1 |
| | conv4.2 3x3 | 32, 32 | 32, 32 | 128 | 147,456 | 1 |
| Residual 5 | conv5.1 3x3 | 32, 32 | 16, 16 | 256 | 294,912 | 2 |
| | shortcut 1x1 | 32, 32 | 16, 16 | 256 | 32,768 | 2 |
| | conv5.2 3x3 | 16, 16 | 16, 16 | 256 | 589,824 | 1 |
| Residual 6 | conv6.1 3x3 | 16, 16 | 16, 16 | 256 | 589,824 | 1 |
| | conv6.2 3x3 | 16, 16 | 16, 16 | 256 | 589,824 | 1 |
| Residual 7 | conv7.1 3x3 | 16, 16 | 8, 8 | 256 | 589,824 | 2 |
| | shortcut 1x1 | 16, 16 | 8, 8 | 256 | 65,536 | 1 |
| | conv7.2 3x3 | 8, 8 | 8, 8 | 256 | 589,824 | 1 |
| Residual 8 | conv8.1 3x3 | 8, 8 | 8, 8 | 256 | 589,824 | 1 |
| | conv8.2 3x3 | 8, 8 | 8, 8 | 256 | 589,824 | 1 |
| Output Block | Avg pooling | 8, 8 | 1 | 256 | 0 | 1 |
| | FC layer | 256 | 256 | | 65,792 | |

Table 2.1: **Residual model parameters, total parameters 11M**

| Layer | Input size | Output size | Parameters |
|---|---|---|---|
| ResNet | (23,256,256) | (23,256) | 526 M |
| GRU Layer 1 | (23,256) | (23,512*2) | 118 k |
| GRU Layer 2 | (23,512*2) | (23,512*2) | 944 k |
| FC layer 1 | 1024 | 256 | 262 k |
| FC layer 2 | 256 | 2 | 512 |

Table 2.2: **Model parameters, total parameters 526 M**

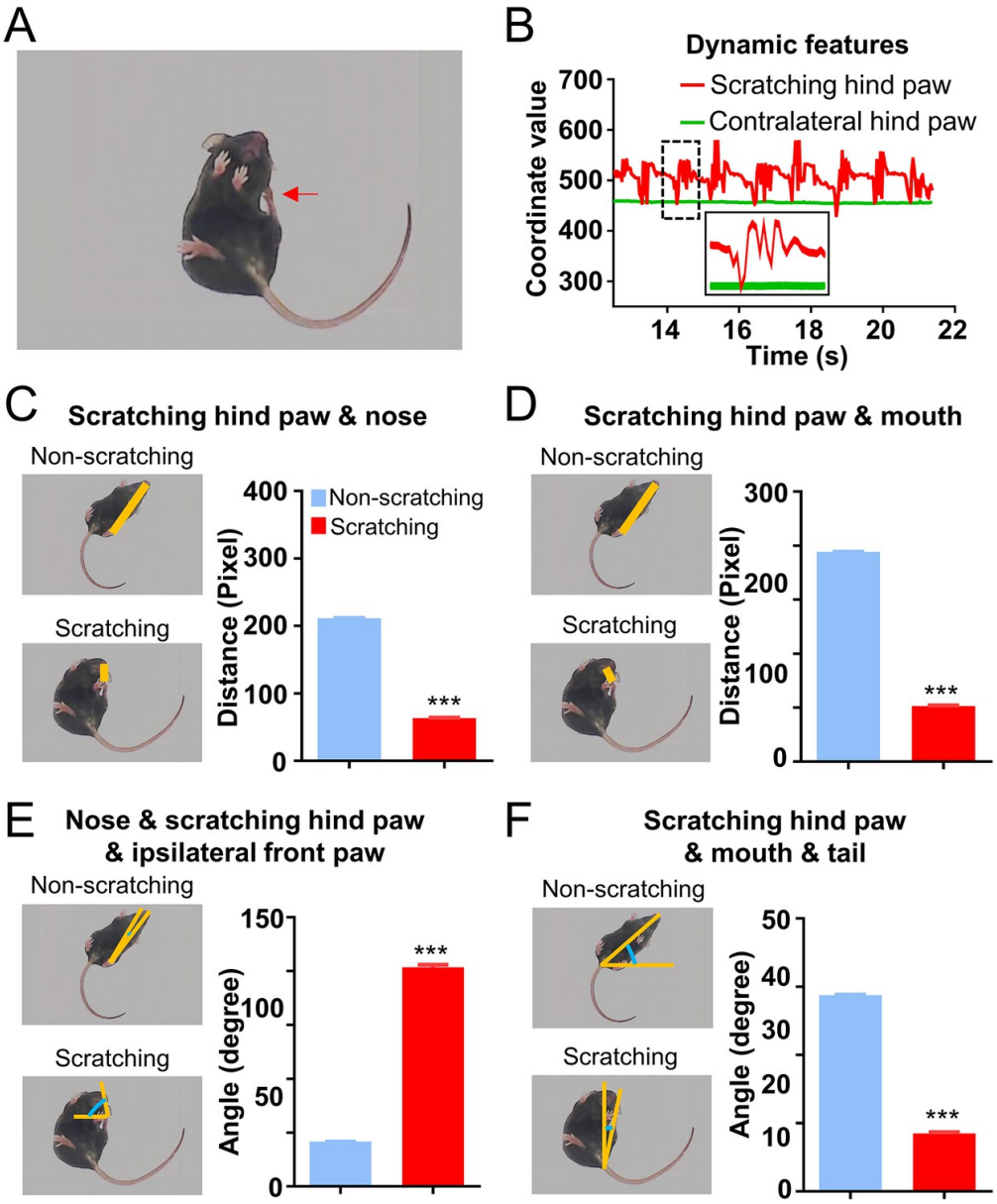



Figure 2.11: **Dynamic and static features of scratching behavior in the chloroquine (CQ) nape acute itch model.**, B) The scratching hind paw (red arrow in A) vibrated rhythmically during scratching behavior, but the contralateral hind paw did not (B). Distance between scratching hind paw and nose (C) or mouth (D), angle formed by nose, scratching hind paw, and ipsilateral front paw (E), and angle formed by scratching hind paw, mouth, and tail (F) in scratching or non-scratching frames (N = 1756 for non-scratching, 2214 for scratching). Standard error of the mean (SEM) error gauge. The differences between the two groups were analyzed using an unpaired two-tailed Student's t-test with a significance level of 0.001.

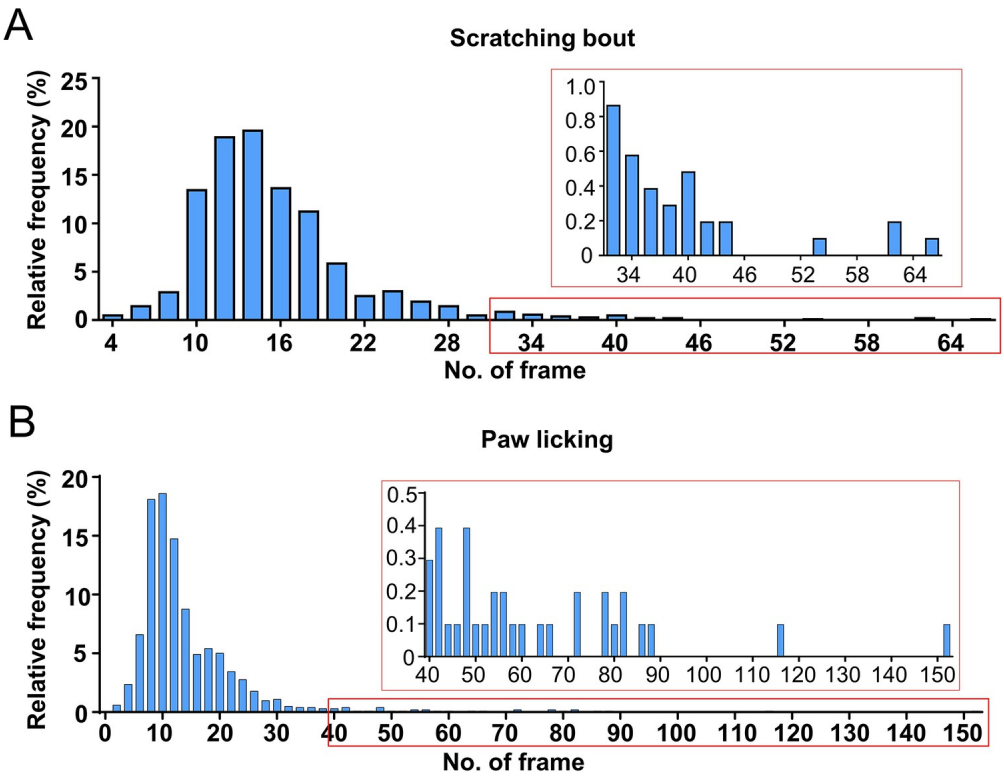



Figure 2.12: **Distribution of scratching episode duration and paw licking in the nape acute itch model induced by chloroquine (CQ).** The frequency distribution of scratching bout (A) or paw lapping (B) duration. The figure inset is a magnification of the red square portion.

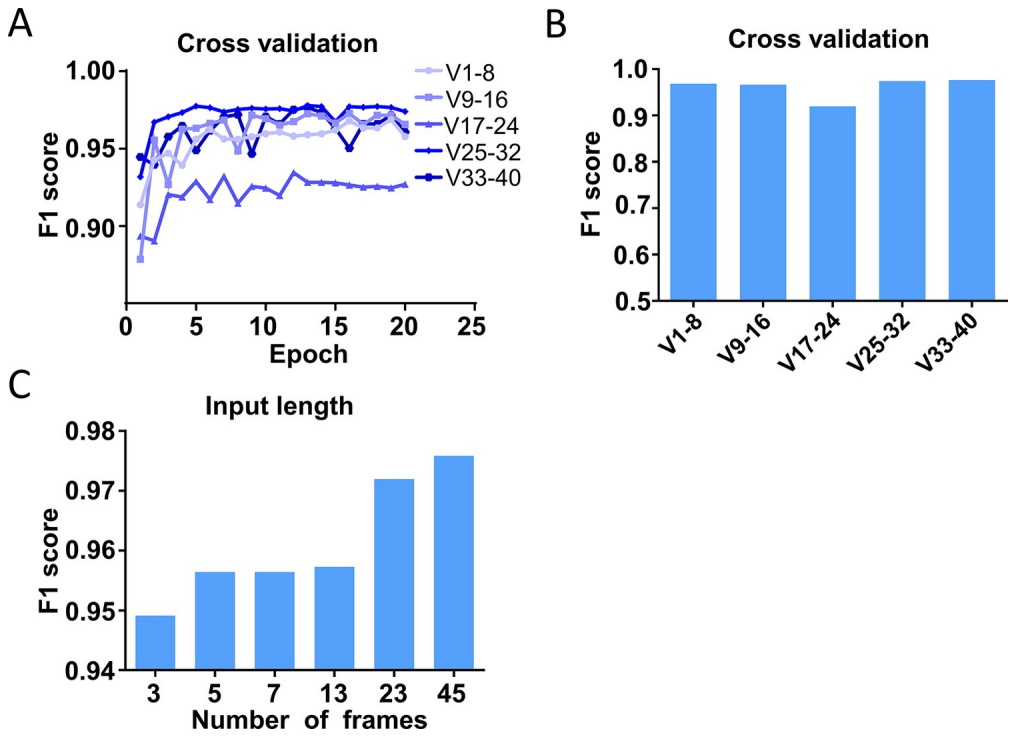



Figure 2.13: **Cross-validation and parameter tuning for the prediction models.** (A) Cross-validation of the trained prediction models through the rotation of the training and test datasets across 40 videos. The F1 rating was determined using the eight evaluation videos. (B) The optimal performance of models trained with N=45 inputs and various combinations of training and test datasets. (C) Simulate outputs with varying input lengths.

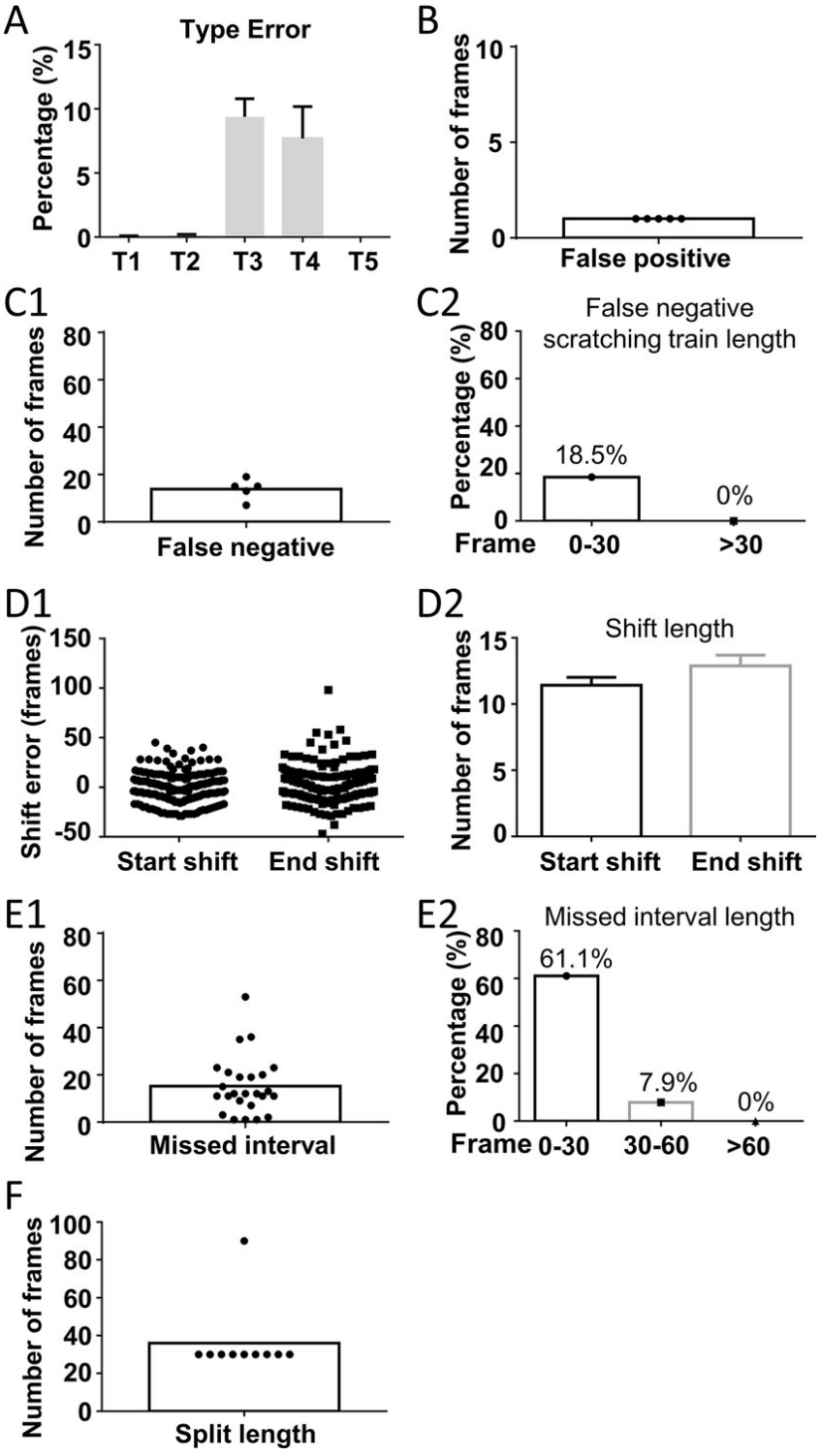



Figure 2.14: **Error analysis for the manual annotation.** (A) The incidence rate of each category of manual annotation error, calculated as the ratio of the total number of frames containing each type of error to the total number of scratching frames in eight test videos. Standard error of the mean (SEM) error gauge. (B) The length of time for all false-positive scratching trains. The duration of all false negative trains (C1) and the Type 2 error rate for scratching trains with varying durations (C2). The distribution (D1) and average length (D2) of shift start and end durations. The duration of all missed intervals (E1) and the Type 4 error rate for intervals with different duration (E2). The duration of divided frames.

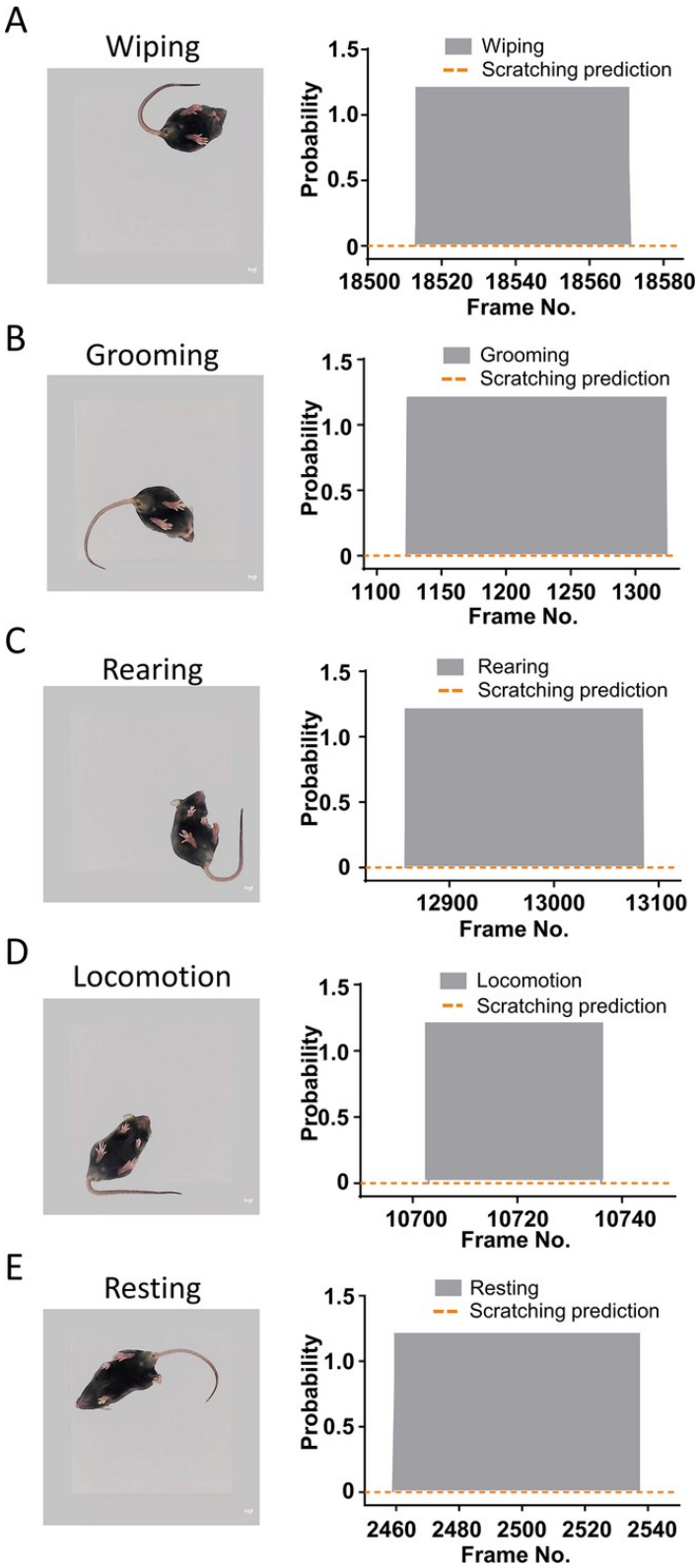



Figure 2.15: **Other mouse behaviors that were not identified as scratching.** The predicted probabilities of scratching behavior during wiping (A), grooming (B), rearing (C), locomotion (D), and resting (E).

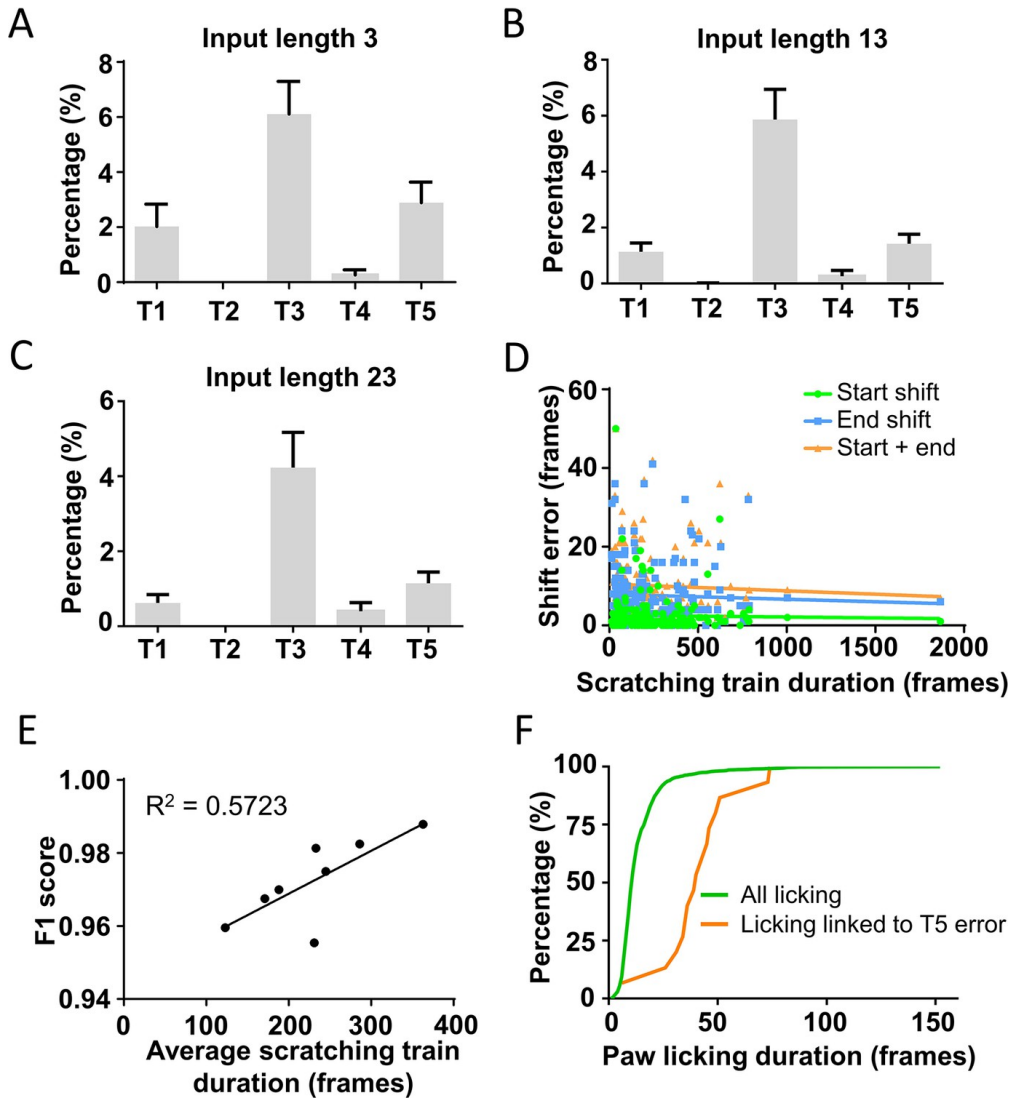



Figure 2.16: **Relationship between prediction errors and the length of the input or the extent of the scratching train.** (A–C) The five type error rates of models trained with varying lengths of input data. Standard error of the mean (SEM) error gauge. The relationship between the duration of the scratching train and the length of the start shift, end shift, or start shift plus end shift. The correlation between the average scratching train duration in a video and the accuracy of the prediction (F1 score). (F) Frequency distribution of all durations of paw lapping and those associated with Type 5 error.

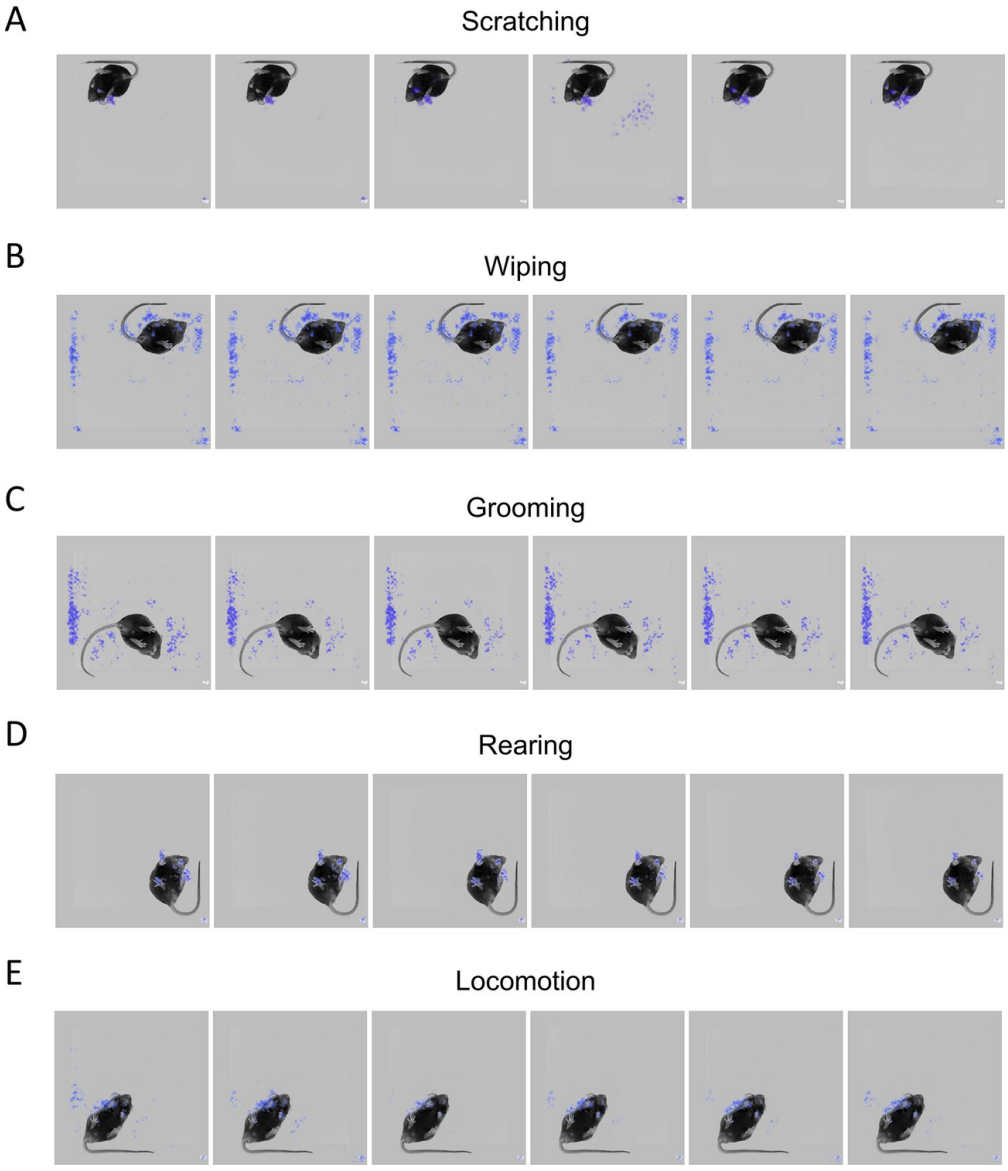



Figure 2.17: **Prediction saliency map of mouse scratching and other activities.** Additional saliency maps displaying the gradient value of every pixel for frames of scratching (A), wiping (B), grooming (C), rearing (D), and locomotion (E) during the best model's scratching behavior prediction.

| Video No. | Mouse No. | Injection site (nape) | Sex | Weight | Age |
|---|---|---|---|---|---|
| V1 | M1 | Left 1 | F | 20.6g | 8-12 week |
| V2 | | Left 2 | | | |
| V3 | | Right 1 | | | |
| V4 | | Right 2 | | | |
| V5 | M2 | Left 1 | F | 21.1g | |
| V6 | | Left 2 | | | |
| V7 | | Right 1 | | | |
| V8 | | Right 2 | | | |
| V9 | M3 | Left 1 | F | 23.3g | |
| V10 | | Left 2 | | | |
| V11 | | Right 1 | | | |
| V12 | | Right 2 | | | |
| V13 | M4 | Left 1 | F | 21.2g | |
| V14 | | Left 2 | | | |
| V15 | | Right 1 | | | |
| V16 | | Right 2 | | | |
| V17 | M5 | Left 1 | F | 21.4g | |
| V18 | | Left 2 | | | |
| V19 | | Right 1 | | | |
| V20 | | Right 2 | | | |
| V21 | M6 | Left 1 | M | 25.2g | |
| V22 | | Left 2 | | | |
| V23 | | Right 1 | | | |
| V24 | | Right 2 | | | |
| V25 | M7 | Left 1 | M | 25.5g | |
| V26 | | Left 2 | | | |
| V27 | | Right 1 | | | |
| V28 | | Right 2 | | | |
| V29 | M8 | Left 1 | M | 21.5g | |
| V30 | | Left 2 | | | |
| V31 | | Right 1 | | | |
| V32 | | Right 2 | | | |
| V33 | M9 | Left 1 | M | 27.8g | |
| V34 | | Left 2 | | | |
| V35 | | Right 1 | | | |
| V36 | | Right 2 | | | |
| V37 | M10 | Left 1 | M | 25.9g | |
| V38 | | Left 2 | | | |
| V39 | | Right 1 | | | |
| V40 | | Right 2 | | | |

Table 2.3: Mouse information used in the recording of the training and test videos

Table 2.4: Scratching behavior summary in the 40 training and test videos (reference annotation)

| **Video No.** | **Scratching trains** | **Scratching frames** | **Scratching time (s)** | **Scratching (%)** |
|---|---|---|---|---|
| V1 | 13 | 9785 | 326 | 27.2 |
| V2 | 19 | 7511 | 250 | 20.9 |
| V3 | 27 | 10359 | 345 | 28.8 |
| V4 | 28 | 11022 | 367 | 30.6 |
| V5 | 47 | 8932 | 298 | 24.8 |
| V6 | 57 | 12519 | 417 | 34.8 |
| V7 | 47 | 8532 | 284 | 23.7 |
| V8 | 37 | 9790 | 326 | 27.2 |
| V9 | 24 | 6349 | 212 | 17.6 |
| V10 | 36 | 7964 | 265 | 22.1 |
| V11 | 43 | 10615 | 354 | 29.5 |
| V12 | 66 | 7877 | 263 | 21.9 |
| V13 | 1 | 371 | 12 | 1.0 |
| V14 | 16 | 8381 | 279 | 23.3 |
| V15 | 34 | 8935 | 298 | 24.8 |
| V16 | 15 | 7356 | 245 | 20.4 |
| V17 | 11 | 4123 | 137 | 11.5 |
| V18 | 37 | 11965 | 399 | 33.2 |
| V19 | 20 | 6000 | 200 | 16.7 |
| V20 | 26 | 10794 | 360 | 30.0 |
| V21 | 20 | 8226 | 274 | 22.8 |
| V22 | 21 | 10071 | 336 | 28.0 |
| V23 | 47 | 12872 | 429 | 35.8 |
| V24 | 23 | 7306 | 244 | 20.3 |
| V25 | 11 | 4137 | 138 | 11.5 |
| V26 | 25 | 7152 | 238 | 19.9 |
| V27 | 29 | 10103 | 337 | 28.1 |
| V28 | 31 | 10424 | 347 | 29.0 |
| V29 | 7 | 2402 | 80 | 6.7 |
| V30 | 11 | 7559 | 252 | 21.0 |
| V31 | 22 | 5559 | 185 | 15.4 |
| V32 | 24 | 7595 | 253 | 21.1 |
| V33 | 19 | 6900 | 230 | 19.2 |
| V34 | 40 | 7525 | 251 | 20.9 |
| V35 | 19 | 4418 | 147 | 12.3 |
| V36 | 44 | 7538 | 251 | 20.9 |
| V37 | 28 | 8005 | 267 | 22.2 |
| V38 | 33 | 8078 | 269 | 22.4 |
| V39 | 26 | 5998 | 200 | 16.7 |
| V40 | 50 | 6168 | 206 | 17.1 |

# Chapter 3

# Cancer survival prediction using multilevel multimodal co-attention deep learning model

## 3.1 Introduction

Cancer is one of the most important challenges in modern healthcare. With global death estimates reaching approximately 10 million annually, it is poised to become the leading cause of mortality in the 21st century (Bray, 2018). The medical field is facing a growing challenge with the increasing number of cancer cases and deaths, making accurate prognosis predictions essential for making the right clinical choices and treatments (Kourou *et al.,* 2015). Survival analysis – the prediction of time-to-event outcomes such as cancer recurrence or death - is fundamental in oncology. It provides a compass for patients navigating the uncertainty of their diagnosis and offers critical insights to clinicians, researchers, and policy makers and can inform treatment choices and policy directions (Mariotto, 2014).

Cui *et al.,* 2023 introduced how recent works demonstrated significant success in employing deep learning methods for diagnosis and prognosis multimodal data (both image and non-image data). For one patient, image data is classified into radiology, pathology, and camera images. These images are further divided into two types: pixel-aligned data, which can be spatially registered and overlaid, and pixel-not-aligned data, where the pixels in different images do not correspond spatially and may even vary in dimensionality (such as 2D, 3D, and 4D). Non-image data includes structured data like genomic sequences and blood test

results, as well as clinical data comprising demographic tabular data or free text in lab reports. The diverse nature of these image and non-image data presents significant challenges in multimodal learning, which involves a set of machine learning algorithms. Furthermore, image data are generally larger and denser (containing millions of pixels), in contrast to the more sparse, lower-dimensional non-image data. This heterogeneity in formats (including various dimensions, images, free text, and tabular data) necessitates distinct preprocessing and feature extraction methods. Different types of information also require fusion techniques that effectively capture both shared and complementary information for improved diagnosis and prognosis.

### 3.1.1 Traditional cancer survival analysis: Cox proportional hazards (CPH) model

In the context of cancer, survival analysis focuses on the time between diagnosis and death from the disease. The goal is to estimate the time until this event occurs. However, in practical scenarios, some patients might be lost to follow-up before the event is observed, or they may pass away from causes unrelated to the cancer, thus the event times for these cases are not fully known; we only know that they have survived for at least a certain period. Such instances, where the event of interest has not been observed at the last known follow-up, are termed censored. Predictions for data that are right-censored are based on a collection of characteristics $x = (x_1, ..., x_p)^T \in \mathbf{R}^p$, which may vary over time. Even though censored observations do not reveal the complete event time, they are still informative for statistical models since the censoring time provides a minimum estimate of the patient's survival duration.

The semi-parametric Cox Proportional Hazards (CPH) model is a widely used statistical technique for analyzing survival data with censored observations (Cox, 1972). This model focuses on the hazards instead of the survival function, which is expressed as $h(t|x) = h_0(t)exp(\beta^T x)$. Here, $t$ is time, $x = (x_1, ..., x_p)^T$ is the covariates of dimension $p$, $\beta = (\beta_1, ..., \beta_p)^T$ is a vector of regression parameters, and $h_0(t)$ is the baseline hazard. The function $f(x) = \beta^T x$ is also referred to as the *risk function*. The regression parameters can be estimated by minimizing the negative log partial likelihood, where $n$ is the number of patients, $t_i$ is the censored or observed survival time for patient $i$, and $\delta_i$ is an indicator for

whether the survival time is censored or observed:

$$l(\beta) = -\sum_{i=1}^{n} \delta_i \left( \beta^T x_i - log \sum_{j \in R(t_i)} exp(\beta^T x_j) \right)$$

This method is widely used (S. W. Grant *et al.,* 2019), but has two main limitations. First, the CPH model is based on a linear model, meaning it is unable to capture non-linear relationships between the input data and the risk of death. If the covariates are nonlinear with the risk function, this model would not be effective. Second, it assumes that the effect of the patient's features is constant over time, meaning the predictions for different patients are proportional and only differ by scaling the baseline hazard with a factor that is constant over time. This implies that the predicted survival curves for different patients do not cross, which is not always realistic. Furthermore, the selection of covariates for survival prediction is largely based on the subjective interpretation and intuition of a clinician (Hui, 2019). This reliance on subjective judgments has raised questions about the accuracy and reproducibility of survival predictions, raising doubts about their true efficacy and reliability (Cheon, 2016).

### 3.1.2 Deep learning survival models

The healthcare system is undergoing a major transformation, moving from a population-based approach—where treatments are designed for the average patient—to a personalized model, where decisions are based on the individual's characteristics and peculiarities (Giunchiglia *et al.,* 2018). This shift has been greatly accelerated by the remarkable progress in the field of Deep Learning (Esteva, 2019, Norgeot *et al.,* 2019). Recently, there has been a surge of enthusiasm among scholars to use deep learning techniques to carefully examine survival data. Deep Learning, especially through the use of artificial neural networks, offers a major benefit: it can make predictions from raw input data, eliminating the need for manual feature engineering (LeCun, Bengio, *et al.,* 2015a). This not only increases the accuracy of the analysis, but also reduces the bias that comes with subjective assessments.

The initial integration of the Cox Proportional Hazards (CPH) model with deep learning (DL) was achieved through a basic feed-forward neural network for single-mode data inputs, as described in the study Faraggi & Simon, 1995. With the advent of big data in medicine, particularly in precision medicine, there's been a surge in various types of clinical data.

This diversity of complex data has highlighted the necessity for more sophisticated modeling techniques, leading to the development of DL-based methods. Building on this, advanced DL models have been proposed for the CPH framework, such as DeepConvSurv (X. Zhu *et al.,* 2016), which introduced a deep convolutional neural network to enhance the CPH model's capacity to analyze pathology images, a step beyond the traditional use of handcrafted features. This innovation was proven to be more effective by extensive testing on datasets like the National Lung Screening Trial (NLST) for lung cancer. To address the limitations of the CPH model, such as its linearity and proportionality assumptions, new methods have been introduced. Cox-Time (Kvamme *et al.,* 2019) model, incorporates time as an additional input to examine its relationship with other variables. RNN-SURV (Giunchiglia *et al.,* 2018) employs recurrent neural networks to dynamically track the influence of covariates over time, providing interpretable risk scores through a linear combination of time-sensitive estimates. Meanwhile, DeepSurv (Katzman, 2018) automatically computes the interaction between treatments and covariates, eliminating the need to predefine these interactions, unlike in the traditional CPH model. These DL models refine and extend the structure of the CPH model to accommodate the complexities of modern medical data.

Other deep learning methods rely on discretization schemes of the measured time and output predictions for a set of predetermined time intervals. One method employs the multi-task logistic regression (MTLR) model as its foundation, integrating it with the core of the deep learning architecture (Fotso, 2018), while another method, DeepHit (C. Lee, Zame, *et al.,* 2018) leverages a deep neural network to directly learn the distribution of survival times and handle scenarios with competing risks (more than one possible event of interest). Building upon the foundation laid by DeepHit, Dynamic-DeepHit (C. Lee, J. Yoon, *et al.,* 2019) refines the approach by parameterizing the probability mass function of the survival distribution and integrating a ranking component into the loss function. DRSA (Ren *et al.,* 2019) combines a series of long short-term memory (LSTM) units, allowing for the prediction of survival probabilities without the need for traditional event probability assumptions or segmented data approaches. In another study (Zadeh & Schmid, 2020), the authors compared various loss functions applied within different discretization schemes. They observed that substituting the commonly used cross-entropy loss with a loss function derived

from the negative log-likelihood of a discrete time-to-event model resulted in significantly improved predictions.

### 3.1.3 Multi-instance learning for whole slide images

The growing prevalence of high-dimensional multimodal data, encompassing advanced clinical, imaging and molecular data sets, underscores the need to amalgamate them through the lens of deep multimodal representation learning (W. Guo *et al.,* 2019, Baltrusaitis *et al.,* 2019). The Cancer Genome Atlas (TCGA), a project financed through public endowments, leverages microarray and next-generation sequencing technologies to meticulously profile the cancer genome. Data includes gene expression, copy number variations, mutations, DNA methylation, and whole-slide imaging (Tomczak *et al.,* 2015). Deep learning paradigms have evolved to seamlessly integrate diverse data modalities, including clinical data, digital pathology imagery, and different genomic modalities such as gene and microRNA expression. These advances are designed to predict the prognosis of cancer in a spectrum of 20 distinct cancer entities (Cheerla & Gevaert, 2019). Although the profusion of big data and the requisite for assimilating emergent data modalities champion multimodal deep learning methodologies, they are not without inherent practical challenges.

High-dimensionality emerges as a primary challenge in the realm of multimodal data analysis. Notably, whole slide images (WSIs), which are digital scans of entire pathology slides, epitomize this challenge. WSIs, central to digital pathology, empowering pathologists and researchers to meticulously dissect tissue samples, paving the way for precision diagnoses and avant-garde research (Farahani *et al.,* 2015). However, juxtaposed against conventional image classifications with dimensions hovering around 256 $\times$ 256 pixels, WSIs are behemoths, spanning multi-gigabytes and boasting resolutions reaching 150,000 $\times$ 150,000 pixels. Compounded by the high morphological variability and the diverse tissue types that they encapsulate, WSIs defy the straightforward deployment of traditional deep learning paradigms (Dimitriou *et al.,* 2019). Addressing the cancer survival conundrum, the imperative is to distill patient-centric insights from an ensemble of WSIs, necessitating a granular visual recognition approach. Such an approach must grapple with intricate interplays amongst diverse visual constructs, be it the labyrinthine interweaving of stroma, tumor conglomerates, immune cells, or other nuanced visual entities (H. Chen, X. Qi, *et al.,* 2016, Saltz *et al.,*

2018, Hosseini *et al.,* 2019).

Given the large gigapixel resolutions of WSIs, traditional image-based survival methodologies have predominantly chosen to select discriminative patches from manually annotated Regions of Interests (ROIs), subsequently extracting hand-crafted features for predictions (H. Wang *et al.,* 2014, S. Wang *et al.,* 2016, Yao, S. Wang, *et al.,* 2016). This approach, while feasible, is susceptible to overlooking crucial discriminatory patterns associated with survival, especially when only a handful of tiles are selected from the highly heterogeneous pathological slides. In response to this challenge and with the intent of harnessing the full potential of WSIs, numerous strategies have embraced the multiple instance learning-based (MIL) approach. Within this paradigm, the initial phase involves gleaning instance-level feature representations from an assortment of randomly selected image patches within the WSI. Subsequently, the second phase employs global aggregation techniques to synthesize an overall representation of the slide, setting the stage for subsequent supervision.

In the study by Hou *et al.,* 2016, a novel strategy is proposed for handling the computational complexities in training Convolution Neural Networks (CNNs) on Whole Slide Images (WSIs). This approach focuses on patch-level classifiers, combining their predictions through a decision-fusion model. Additionally, an Expectation-Maximization-based method is introduced to identify and utilize discriminative patches, which is specifically effective in refining the classification of different cancer subtypes, not limited to just one subtype. In contrast, (Wulczyn *et al.,* 2020) developed a weakly supervised deep learning framework to predict disease-specific survival rates in patients with 10 different types of cancer, using histopathology images. This marks a significant advancement in the field of medical imaging and cancer prognosis. Further contributing to this area of research, Yao, X. Zhu, *et al.,* 2019 presented a multi-instance fully convolution networks model (MI-FCN). This model, grounded in deep multiple instance learning, excels at extracting intricate patterns from WSIs. It has shown promising results in enhancing prognostic accuracy, particularly for Lung and Brain cancers. However, it remains unclear if this model was tested on other cancer types or if its effectiveness was confined to just these two malignancies.

Incorporating multimodal fusion mechanisms into weakly-supervised learning to forecast survival poses an additional obstacle owing to the substantial disparity in data heterogeneity

between genomics and WSIs. Genomic features are frequently represented as a single tabular attribute, whereas WSIs are packaged as tens of thousands of image patches. Consequently, numerous methodologies rely on delayed fusion mechanisms to integrate features, thereby impeding the acquisition of crucial multimodal interactions (M. Liang *et al.,* 2014, Yao, X. Zhu, *et al.,* 2019, Vale-Silva & Rohr, 2021)

### 3.1.4 Our proposed methodology

In order to tackle the challenges of handling the vast data of Whole Slide Images (WSIs) and the correlation between different data modalities, we present the *Genomic-Guided Hierarchical Attention Multi-Scale Pathology Transformer (GG-HAMPT)*, a novel deep learning framework that is multimodal and multi-scale, specifically designed for the purpose of predicting survival. By modeling co-attention correlation between whole slide images (WSIs) and genomes, GG-HAMPT enables survival outcomes to be intuitively interpreted.

Survival prognosis is heavily reliant on clinical metrics, particularly tumor stage grade (Frank *et al.,* 2002). However, the design of our framework is intended to extract this information inherently from the tumor image and genomics. By emulating hierarchical attention mechanisms commonly found in Natural Language Processing (NLP), which possess the ability to discern document semantics across word, sentence, and document levels, we designed a partial attention network with multiple tiers. In order to surpass the challenge presented by gigapixel WSIs' voluminous data, this network is optimized to extract pathogenic insights at three distinct resolution levels (256×256, 1024×1024, 4096×4096). Our model deviates from the prevalent practice of late fusion in modern deep learning frameworks for survival analysis by utilizing the co-attention paradigm, which is well-established in the field of Visual Question Answering (VQA). This methodology enables a more cohesive early fusion approach, proficient in deciphering the complex interaction between genomic information and the pathogenic intricacies detected at one of the different three levels. Furthermore, it dynamically evaluates the significance attributed to pathogenic information in accordance with the genomic context.

Key benefits of the GG-HAMPT framework include:

1. Unlike conventional late fusion models, which typically concatenate WSI and genomic

representations and does not have much model weights for modality fusion, our co-attention-based early fusion layer contains a substantial fraction of the model's parameters. This design facilitates concurrent learning of WSI and genomic representations across the three resolution scales, enabling a comprehensive grasp of their interrelations. Leveraging the attention scores for each patch, the deciphered interactions can be pictorially represented through attention heatmaps.

2. Our multi-scale hierarchical partial attention layer is adept at navigating gigantic WSIs. As opposed to mere patch selection from regions of interest, our multi-scale architecture resonates with human heuristic methodologies employed in tissue imagery examination. This is evidenced by the adaptive co-attention weights, which emulate the physician's strategy of pinpointing pivotal regions. Moreover, recognizing that the assimilation of cell-level information inherently shapes the tissue-level and organ-level data, our model's focal point pivots from elementary resizing and encoding to a more complete amalgamation of information, culminating in global insights.

The results in Table 3.1 confirm that our GG-HAMPT model surpasses existing state-of-the-art weakly-supervised methods in predicting survival outcomes using multimodal data, including gigapixel WSIs. To further validate the effectiveness of our model, we conducted an ablation study using five large, publicly available cancer datasets. These datasets encompassed a range of genomic information, including gene expression, copy number aberration, and mutation information. The inclusion of these diverse data types significantly enhanced the model's predictive power, as detailed in Table 3.2.

In addition to these quantitative analyses, we employed heat maps to visualize the visual concepts guided by genes. These heat maps, as depicted in Figures 3.7 and 3.8, allowed us to analyze the interactions between WSIs and genomic data. Our evaluation focused on the correlation between morphological features in the WSIs and individual genes. This approach not only illuminated the relationship between these features but also highlighted a list of genes that emerged as strong predictors in our model. The identified genes demonstrate a significant impact on the model's ability to accurately predict survival outcomes, underscoring the value of integrating genomic data with WSIs in survival analysis.

## 3.2 Method

This section presents our framework in GG-HAMPT for predicting discrete-time survival outcomes and identifying disease predictors using WSIs and genomic data. Figure 3.1 provides an illustration of the method. It consists of three components: a feature extraction module with dedicated submodels for WSI and genomics (described in section 3.2.2); a multi-level partial co-attention transformer that fuses the submodel outputs into processed representations (described in section 3.2.3); and a transformer decoder that maps the incoming feature representation to a set of discrete-time conditional survival probability predictions (described in section 3.2.4). This framework builds on the multimodal and weakly-supervised deep learning model presented in R. J. Chen *et al.*, 2021.

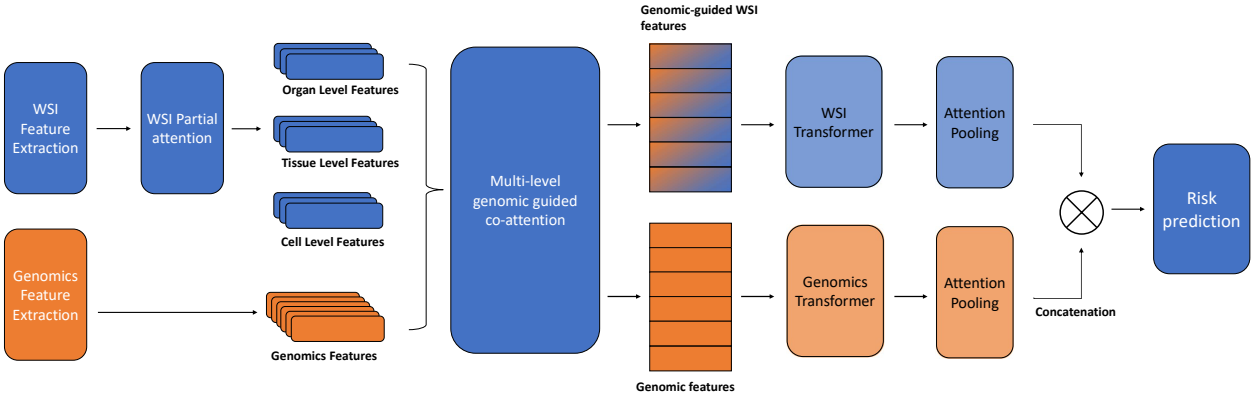



Figure 3.1: Our Genomic-Guided Hierarchical Attention Multi-Scale Pathology Transformer (GG-HAMPT) model architecture has two inputs: a gigabyte whole slide image and genomics features (including gene expression RNA seq and copy number matrix). The model is composed of three parts: feature extraction (as described in Section 3.2.2), partial attention and multi-scale co-attention (as described in Section 3.2.3), and Transformer encoder and pooling (as described in Section 3.2.4). Finally, Section 3.2.1 discusses how the features extracted from the network are used to predict survival.

### 3.2.1 Problem Formulation

Our goal is to utilize pathology WSI and genomic data to forecast the time it takes for a patient with a particular type of cancer to experience an event of interest, such as death or disease progression. The outcomes of the survival analysis can help physicians and scientists better understand the development of the disease and recognize possible treatment alternatives for individual patients.

Let $T$ denote the continuous random variable for survival time, $X$ denote the patient

information, then the survival function $f_{\text{surv}}(T \geq t|X)$ symbolizes the probability of the patient surviving longer than a certain time $t$, and the hazard function $f_{\text{hazard}}(T = t|T \geq t, X_i)$ represents the probability of the patient passing away at time $t$.

$$f_{\text{hazard}}(T = t|T \geq t) = \lim_{\partial t \to 0} \frac{P(t \leq T \leq t + \partial t|T \geq t)}{\partial t}$$

Out method don't use the assumptions of the traditional CoxPH-based model. Our method utilizes discrete time intervals and models each interval using a neuron with independent output, as described in Zadeh & Schmid, 2020. For each patient $i$, we have the set of WSIs related to them, $\{W_{ij}\}_{i=1}^{K_i}$, their gene expression vector $\boldsymbol{e}_i$, their copy number variation vector $\boldsymbol{v}_i$, their survival time (in months) $t_i \in \mathbb{R}^+$, their mutation vector $\boldsymbol{m}_i$ and their censorship status $c_i \in \{0, 1\}$. Our dataset can be represented as $\{X_i, t_i, c_i\}_{i=1}^{n} = \{\{W_{ij}\}_{j=1}^{K_i}, \boldsymbol{e}_i, \boldsymbol{v}_i, \boldsymbol{m}_i, t_i, c_i\}_{i=1}$. We then formulate our multi-modal cancer survival prediction problem as finding a neural network model $F$ that integrates the patient $i$ information $X_i = \{\{W_{ij}\}_{j=1}^{K_i}, \boldsymbol{e}_i, \boldsymbol{v}_i, \boldsymbol{m}_i\}$ to predict the hazard function $f_{\text{hazard}}(T = t|T \geq t, X_i) \in [0, 1]$.

We divide the continuous timeline into four distinct, non-overlapping intervals: $[t_0, t_1)$, $[t_1, t_2)$, $[t_2, t_3)$, $[t_3, t_4)$ with fixed boundaries $0 < t_1 < t_2 < t_3 < t_4 \in \mathbb{R}^+$ of survival time values (in months). The discrete event time of each patient, indexed by $j$, with continuous event time $T'_j$ is then determined by:

$$T_j = r \text{ if } T'_j \in [t_r, t_{r+1}), \text{ for } r \in \{0, 1, 2, 3\}$$

The probability of hazard and survival for the patient $i$ can be determined by:

$$f_{\text{hazard}}(t|X_i) = P(T = t|T \geq t, X_i)$$

$$f_{\text{surv}}(t|X_i) = P(T > t|X_i) = \prod_{s=1}^{t}(1 - f_{\text{hazard}}(s|X_i))$$

Here $T$ denotes the discrete time variable. Predictions of $f_{\text{hazard}}(t)$ are derived from the output layer of the neural network with a sigmoid activation function.

As discussed in Zadeh & Schmid, 2020, we use the negative logarithmic likelihood function

for the discrete survival model as the optimization goal.

For patient $i$, let $T_i$ be the time interval from actual recorded time$t_i$, $c_i$ be the censorship status ($c_i = 0$ if the patient passed away during the discrete time interval $T_i$), $\beta$ be the (fixed hyperparameter) weight for censored patients, then the loss function can be expressed as:

$$L = -c_i(1-\beta)\log(f_{\text{surv}}(T_i|X_i)) - (1-c_i)\{\log(f_{\text{surv}}(T_i-1|X_i)) + \log(f_{\text{hazard}}(T_i|X_i))\}$$

### 3.2.2 Feature extraction

**WSI feature extraction** To extract information from the slide set $\{W_{ij}\}_{j=1}^{K_i}$, an initial segmentation is performed on tissue sample of the WSI, producing a collection of patches with dimensions $4096 \times 4096$ at the organ level. Subsequently, each of these $4096 \times 4096$ patches undergoes further segmentation, yielding 16 patches of dimensions $1024 \times 1024$ at the tissue level. This hierarchical process culminates in the generation of 256 patches of dimensions $256 \times 256$ at the cellular level. Each set of patches, representing the three distinct granularity levels, is processed through a ResNet-50 CNN encoder (pretrained on ImageNet), producing three sets of feature embeddings, each with a dimensionality of $d_k$ (a fixed hyperparameter of embedding size). Given $M$ patches at the organ level, the embeddings extracted across the three levels are organized into specific sets: $H_{\text{organ}} \in \mathbb{R}^{M \times d_k}$, $H_{\text{tissue}} \in \mathbb{R}^{16M \times d_k}$, and $H_{\text{cell}} \in \mathbb{R}^{256M \times d_k}$. Using the complete tissue information across numerous WSIs, the typical patch size at the organ level for training is approximately $M = 60$, with some patients bag size $M$ up to $M = 890$, resulting more than 200,000 patches at the cellular level $256 \times 256$. Given the vastness of these data, conventional multi-instance learning (MIL) methods are often limited to either random sampling or straightforward concatenation of features extracted by the Residual Network, without delving into the intrinsic relationships between the patches. An example of the patching process and its handling is shown in Figure 3.2.

**Genetics feature extraction** We extracted a variety of genomic features, encompassing gene expression(RNA-seq), copy number variation, and gene mutation status, from the TCGA portal Tomczak *et al.,* 2015. Directly applying an attention mechanism to these genomic measurements with the extensive pathology information poses computational chal-

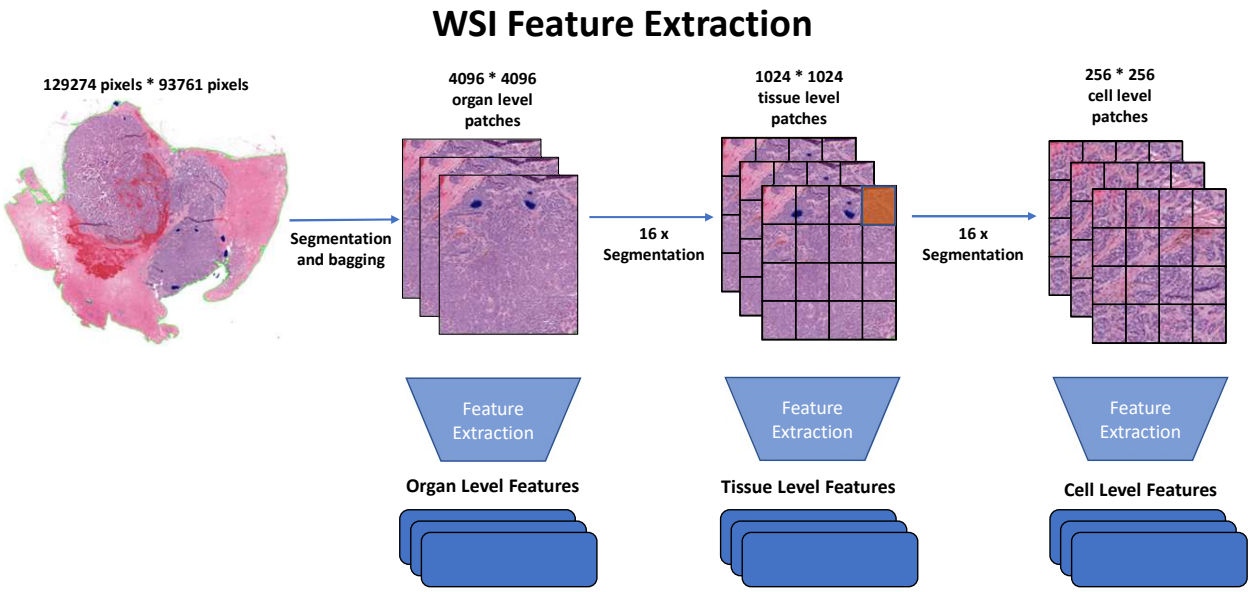



Figure 3.2: Schematic representation of the hierarchical segmentation and feature extraction process for whole slide images (WSIs). Starting with the complete slide set, an initial segmentation segregates tissue-containing regions, forming patches at the organ level with dimensions $4096 \times 4096$. These patches are further segmented to produce finer-grained patches at the tissue level ($1024 \times 1024$) and subsequently at the cellular level ($256 \times 256$). Each granularity level undergoes feature embedding through a ResNet-50 CNN encoder. The flowchart illustrates the transformation of a single WSI into a myriad of patches, culminating in feature sets $H_{\text{organ}}$, $H_{\text{tissue}}$, and $H_{\text{cell}}$. The varying patch sizes denote the progression from macroscopic organ-level details to intricate cellular-level insights. The methodology aims to capitalize on the rich hierarchical information embedded within WSIs, surpassing conventional techniques that might overlook the intrinsic relationships between these diverse patch levels.

lenges. To circumvent this, we segmented the genes into $N = 6$ distinct functional groups based on their biological roles as proposed in Liberzon *et al.*, 2015 and R. J. Chen *et al.*, 2021. These categories were formulated as: (1) tumor suppression, (2) oncogenesis, (3) protein kinases, (4) cell differentiation, (5) transcription, and (6) cytokines and growth factors. Each gene's genomic data was accordingly classified into one of these functionally relevant clusters. For every cluster, a dedicated Fully Connected (FC) layer was trained to harness the genomic nuances specific to that group, resulting in a feature vector $G_i$ of dimension $d_k$. Consolidating these outputs, we crafted the aggregate genomics bags of features, denoted as $\mathbf{G} = \{G_i\}_{i=1}^{N} \in \mathbb{R}^{N \times d_k}$. This procedure is illustrated in Figure 3.3.

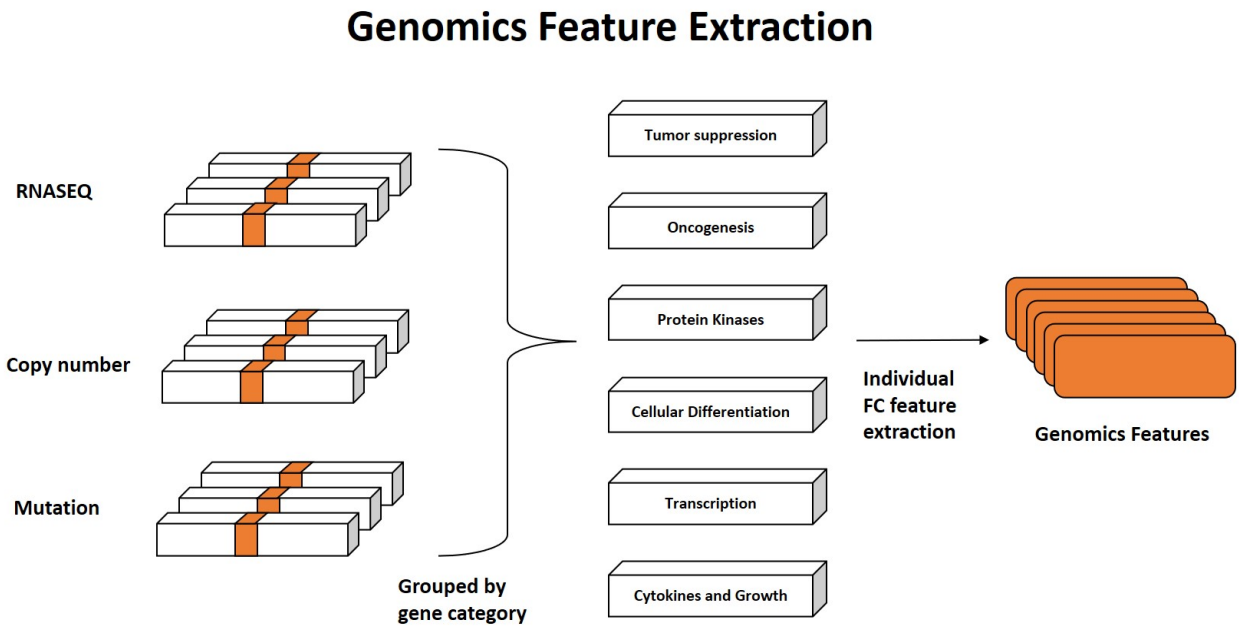



Figure 3.3: Illustration showcasing the process of genomic data classification and feature extraction. Starting with a diverse array of genomic features from the TCGA portal, genes are categorized into six distinct functional groups based on their biological roles. Each group is then processed through a dedicated Fully Connected (FC) layer, capturing the nuanced genomic characteristics specific to that category. The final product is an aggregate genetic bag-of-features that integrates information across all functional groups, providing a complete representation of the genomic landscape.

### 3.2.3 Hierarchical Genomic-WSI Fusion based on Multi-level Co-Attention Module

Modern multimodal pathology methods frequently rely on late fusion techniques, as underscored in Vale-Silva & Rohr, 2021 and Mobadersany *et al.*, 2018. This strategy fuses features mainly at the penultimate network layers, consequently sacrificing interpretability of multimodal interactions. Central to this approach is the inherent challenge of merging gigapixel Whole Slide Imaging (WSI) with genomic features, primarily due to the pronounced heterogeneity gap between them. This stark disparity serves as a formidable barrier, often obscuring the intricate genotype-phenotype interactions within the tumor microenvironment.

Building upon this groundwork, we propose the "Genomic-Guided Hierarchical Attention Multi-Scale Pathology Transformer." Our model embraces the effectiveness of hierarchical network structures seen in diverse fields like natural language processing and image recognition. It is inspired by the Hierarchical Question-Image Co-Attention mechanism in VQA. The following sections will delve into the technical intricacies of our proposed model.

While various models attempt to prioritize Whole Slide Images (WSIs) as one important modality, they often do not provide a comprehensive analytical perspective on these images. Studies like (Hou *et al.,* 2016, Wulczyn *et al.,* 2020,Chikontwe *et al.,* 2020) either analyze the doctor labeled region of interest (ROI), or downsample the whole WSI with loss of detailed information. When juxtaposed with the multimodal fusion techniques seen in Visual Question Answering (VQA) (J. Lu *et al.,* 2016), wherein image features are intertwined with word embeddings through co-attention learning, it's evident that current WSI-centric multimodal methodologies lack commensurate interpretability tools.

In search of more nuanced feature interplay, R. J. Chen *et al.,* 2021 introduced Genomic-Guided Co-Attention (GCA), which pioneers intricate feature aggregation algorithms. These algorithms are tailored to meticulously model the interactions between WSIs and genomics features. Building on these foundational insights, we propose the *Genomic-Guided Hierarchical Attention Multi-Scale Pathology Transformer*. This model leverages the proven efficacy of hierarchical network structures across domains, such as natural language processing and image recognition, and is reminiscent of the Hierarchical Question-Image Co-Attention paradigm in Visual Question Answering. Next we will present all the technical details of the proposed model.

#### 3.2.3.1 Preliminary

**Notation** Through our WSI feature extraction process, we derived WSI bag features across three hierarchical levels:

1. Organ level: $\mathbf{H}_{\text{organ}}$ defined as $\mathbf{H}_{\text{organ}} = \{H_{\text{organ},i}\}_{i=1}^{M}$ with dimensions $\in \mathbb{R}^{M \times d_k}$.
2. Tissue level: $\mathbf{H}_{\text{tissue}}$ defined as $\mathbf{H}_{\text{tissue}} = \{H_{\text{tissue},i}\}_{i=1}^{16M}$ with dimensions $\in \mathbb{R}^{16M \times d_k}$.
3. Cell level: $\mathbf{H}_{\text{cell}}$ defined as $\mathbf{H}_{\text{cell}} = \{H_{\text{cell},i}\}_{i=1}^{256M}$ with dimensions $\in \mathbb{R}^{256M \times d_k}$.

In each case, individual feature vectors are denoted by $H_{i,j}$ and have dimensions $\in \mathbb{R}^{d_k \times 1}$. These features are the outputs indicated in Figure 3.2.

Additionally, from our genomics feature extraction, we derived the genomics bag features $G$ represented as $G = \{G_i\}_{i=1}^{N}$ with dimensions $\in \mathbb{R}^{N \times d_k}$. The genomics features are shown in Figure 3.3.

**Scaled Dot-Product Attention** The attention mechanism takes as input three feature matrices, each of dimension $d_k$: the queries ($Q$), the keys ($K$), and the values ($V$). These

are represented as $Q \in \mathbb{R}^{N_q \times d_k}$, $K \in \mathbb{R}^{N_k \times d_k}$, and $V \in \mathbb{R}^{N_v \times d_k}$ respectively, as outlined in Vaswani *et al.,* 2017.

The computation of attention weights involves taking the dot product of the query matrix with all key matrices. This product is then scaled down by $\sqrt{d_k}$ before applying the softmax function, ensuring stability and manageability of the resultant weights. The final output of the attention mechanism is derived by taking a weighted sum using the value matrix.

Formally, the attention mechanism can be expressed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

#### 3.2.3.2 Hierarchical attention for WSI

Navigating through vast datasets often mandates an organized and methodical approach. Leveraging the age-old "divide and conquer" strategy, one can distill colossal challenges into more tractable segments. In the context of deep learning, particularly when dealing with sprawling gigapixel whole slide images, this principle proves invaluable. Inspired by the efficacy of the Hierarchical Attention Network—which adeptly harnesses dual attention mechanisms at both granular (word) and aggregate (sentence) levels—we put forth a refined multi-tier attention layer. This layer is designed to deftly manage three hierarchical pathologic features. Figure 3.2 offers a visual representation of our patching process and the subsequent extraction of patch features. The focal point of our hierarchical attention mechanism revolves around the features $\mathbf{H}_{\text{organ}}$, $\mathbf{H}_{\text{tissue}}$, and $\mathbf{H}_{\text{cell}}$, each corresponding to its designated patch level. In the sections that follow, we delve deep into the intricate layers of our hierarchical attention framework for WSI, using Figure 3.4 for illustrative clarity.

**Cell-Level patches:** Given an organ patch, $i$, we derive 16 tissue-level patches and a significantly larger pool of 256 cell-level patches. As a result, the organ-level feature, symbolized by $H_{\text{organ},i}$ (for $i$ ranging from 1 to $M$), correlates with an array of 256 cell-level features:

$$\mathbf{H}_{\text{organ,cell},i} = \{H_{\text{cell},256i-255}, H_{\text{cell},256i-254}, \ldots, H_{\text{cell},256i}\}$$

Additionally, every tissue-level feature, $H_{\text{tissue},j}$ (with $j$ extending from 1 to $16M$), maps to a set of 16 cell-level features, briefly represented as:

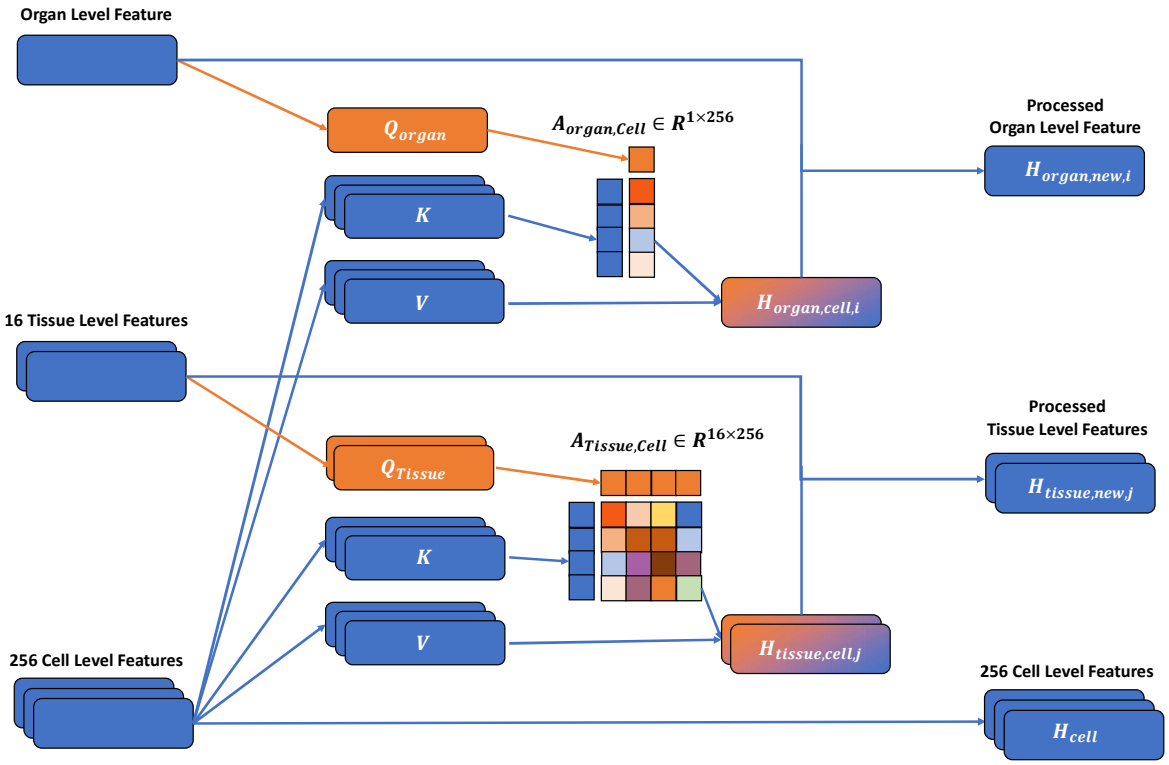



Figure 3.4: Schematic representation of the multi-tier attention layer in action. Beginning with the extracted patch features from whole slide images, the figure details the process of hierarchical attention across organ, tissue, and cell levels. The attention mechanism is finely-tuned to emphasize key features within each hierarchy, facilitating a nuanced understanding of the intricate relationships embedded within the whole slide images.

$$\mathbf{H}_{\text{tissue,cell},j} = \{H_{\text{cell},16j-15}, H_{\text{cell},16j-14}, \ldots, H_{\text{cell},16j}\}$$

By delineating these intricate relationships, our design furnishes a comprehensive representation, empowering deep learning models to unravel the details encapsulated within whole slide images.

**Tissue-Level Hierarchical Attention:** For each tissue segment, represented by the feature $H_{\text{tissue},j}$, we apply an attention mechanism to identify and highlight the most relevant cell patches within that tissue. These important cell patches are then combined to form an enhanced cell-level tissue feature, represented as $H_{\text{tissue, cell},j}$.

In this setup, the tissue feature $H_{\text{tissue},j}$ acts both as the focal point of the analysis (the query) and the base reference (the anchor). It interacts with 16 associated cell features, which function as the keys and values in the attention process. The attention mechanism

computes a weighted average of these cell features, based on their relevance to the tissue feature. This process results in a more detailed and informative tissue-level feature, denoted as $H_{\text{tissue, new},j}$. In terms of technical specifics, the matrices $W_{q,TA}$, $W_{k,TA}$, and $W_{v,TA}$ are trainable weights within the attention framework. Each of these matrices corresponds to a component of the query-key-value triplet in the attention mechanism, playing a critical role in determining how the cell features are weighted and combined.

Formally:

$$H_{\text{tissue, cell},j} = \text{Attention}\left(H_{\text{tissue},j}W_{q,TA}, \mathbf{H}_{\text{tissue,cell},j}W_{k,TA}, \mathbf{H}_{\text{tissue,cell},j}W_{v,TA}\right)$$

This design ensures adaptability:

$$H_{\text{tissue, new},j} = \alpha H_{\text{tissue, cell},j} + (1-\alpha)H_{\text{tissue},j}$$

The coefficient $\alpha$ is used to balance the influence of the attention mechanism with the original characteristics of the tissue. This balance enables a dynamic representation of the tissue features.

**Organ-Level Hierarchical Attention:** In a similar approach, for each organ segment represented by the feature $H_{\text{organ},j}$, we apply an attention mechanism to extract the most significant cell patches that contribute to the overall understanding of the organ. This process generates a cell-level organ vector. In this scenario, the organ feature $H_{\text{organ},j}$ serves as the query in the attention mechanism, and it interacts with 256 cell features which act as keys and values. This cell-level organ vector is then combined with the original organ feature to produce a more detailed and refined organ-level feature.

Defined as:

$$H_{\text{organ, cell},i} = \text{Attention}(H_{\text{organ},i}W_q, \mathbf{H}_{\text{organ,cell},i}W_k, \mathbf{H}_{\text{organ,cell},i}W_v)$$

And:

$$H_{\text{organ, new},i} = \alpha H_{\text{organ, cell},i} + (1-\alpha)H_{\text{organ},i}$$

Collating these refined features, we formulate the feature matrix:

- $\mathbf{H}_{\text{organ,new}} = \{H_{\text{organ, new},i}\}_{i=1}^{M} \in \mathbb{R}^{M \times d_k}$
- $\mathbf{H}_{\text{tissue,new}} = \{H_{\text{tissue, new},i}\}_{i=1}^{16M} \in \mathbb{R}^{16M \times d_k}$
- $\mathbf{H}_{\text{cell,new}} = \{H_{\text{cell},i}\}_{i=1}^{256M} \in \mathbb{R}^{256M \times d_k}$

These combined features now serve as the foundational input for the pathology-related modeling in the multi-level co-attention layer.

#### 3.2.3.3 Multi level co-attention

We approach the complex relationship between Whole Slide Images (WSIs) and genomic features by representing them as 'bags' of features. This approach allows us to develop sophisticated methods to combine and analyze the interactions between specific feature embeddings of instances in WSIs and their corresponding genomic embeddings. In the upcoming section, we will discuss the Genomic-Guided Co-Attention (GCA) mechanism. This multi-level structure is grounded in the principles of standard Transformer attention, which is commonly used to manage interactions between image-grid and word embeddings in Visual Question Answering (VQA). A visual illustration of the GCA mechanism can be found in Figure 3.5.

To explain further, the GCA mechanism draws inspiration from the Transformer model's ability to handle complex interactions in data. It is specifically designed to analyze relationships at different biological levels - organ, tissue, and cell. For each of these levels, we implement a co-attention layer, enabling our model to simultaneously focus on relevant features across different scales and modalities. Thus, the GCA mechanism involves layers of co-attention that correspond to the organ, tissue, and cell levels. Each level within this co-attention framework is meticulously structured to capture the nuances of the corresponding biological specificity.

$$\begin{aligned}
\text{CoAttn}_L &= \text{Attention}(\mathbf{Q}_L, \mathbf{K}_L, \mathbf{V}_L) \\
&= \text{Attention}(\mathbf{G}W_{q,L}, \mathbf{H}_{\text{L, new}}W_{k,L}, \mathbf{H}_{\text{L, new}}W_{v,L}) \\
&= A_{\text{coattn, L}}\mathbf{H}_{\text{L, new}}W_{v,L} = \hat{\mathbf{H}}_L
\end{aligned}$$

In our model, the matrices $W_{q,L}, W_{k,L}, W_{v,L} \in \mathbb{R}^{d_k \times d_k}$ act as adaptive weight coefficients. These matrices adjust the queries ($\mathbf{G}$) and the keys and values ($\mathbf{H}_{\text{L, new}}$) within the

co-attention layers. Additionally, our multi-level co-attention mechanism comprises three distinct co-attention weight matrices for different biological levels: organ, tissue, and cell, represented by $A_{\text{coattn, organ}}$, $A_{\text{coattn, tissue}}$, and $A_{\text{coattn, cell}}$, respectively.

The outputs from these co-attention layers determine the weighted average of the input features at each respective level. By utilizing the output matrices from the organ, tissue, and cell co-attention layers, we calculate a weighted summation. This process results in the final integrated pathology-related feature, denoted as $\hat{\mathbf{H}}$:

$$\hat{\mathbf{H}} = w_{\text{organ}}\mathbf{H}_{\text{organ, new}} + w_{\text{tissue}}\mathbf{H}_{\text{tissue, new}} + w_{\text{cell}}\mathbf{H}_{\text{cell, new}}$$

**Multi Layer Genomic-Guided Co-Attention interpretation:** The Genomic-Guided Co-Attention (GCA) layer can be thought of as a mechanism that evaluates the similarity between specific genomic embeddings (which include gene expression, copy number, and mutation data) and their corresponding pathology features in WSIs. For example, consider a genomic embedding $g_n$ from the set $G = \{g_1, ..., g_N\}$. At each biological level $L$, the GCA layer assesses how much a particular feature $h_m$ from the set $H_{\text{L}}$ aligns or resonates with $g_n$. This relationship is represented in a row vector, such as $[a_{n1}, a_{n2}, ..., a_{nm}]$ in the matrix $A_{\text{coattn}}$. These attention weights are then applied to $H_{\text{L}}$, resulting in a new feature embedding $h_{Ln}$ in $\mathbb{R}^{n\times 1}$, which reflects the biological characteristics of $g_n$. For example, if $g_n$ represents a genomic feature associated with tumor development, the GCA layer will highlight image patches with tumor cells, assigning high attention scores to them.

By combining insights from the three levels $L$, we obtain $\hat{\mathbf{H}}$, a consolidated feature at the WSI level that primarily emphasizes tumor cells. This process leads to what we call 'gene-guided visual concepts'. These are high-attention image patches that strongly correlate with a specific genomic embedding $g_n$, indicating similar phenotypic traits. The GCA layer can identify as many different gene-guided visual concepts as there are unique genomic embeddings in $G$. To visualize these concepts and their association with specific genomic features, one can refer to the attention heatmaps presented in Figures 3.7 and 3.8.

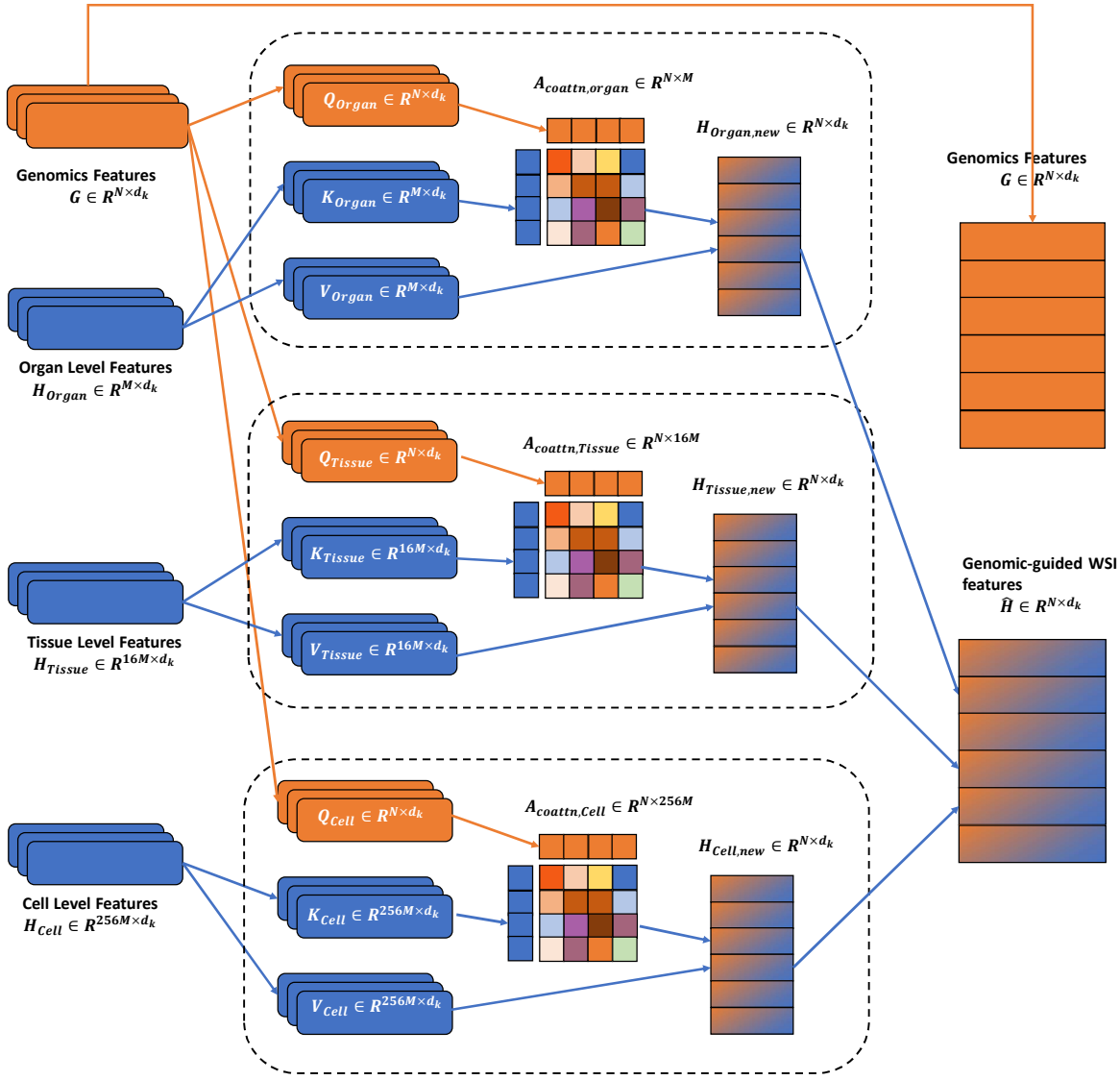



Figure 3.5: Schematic visualization of the Multi Layer Genomic-Guided Co-Attention mechanism. This multi-layered representation showcases the interaction between genomic features and pathology features across the organ, tissue, and cell levels. Emphasizing the relationships between the different levels, GCA integrates genomic-guided attention to yield refined pathology feature embeddings.

### 3.2.4 Set-based Transformer encoder

In the encoder segment of our architecture, we adopt insights from Zaheer et al. Zaheer *et al.,* 2017, emphasizing that set-based network architectures can maintain permutation invariance if constructed upon layers that are permutation-equivariant. Given this principle, our approach applies the set-based transformer to encode features post the genomic-guided co-attention phase. The overall structure of the encoder's formulation can be articulated as:

$$\begin{aligned}
\mathcal{E}^{(l)}\left(H^{(l)}\right) &= \zeta^{(l)}\left(\psi^{(l)}\left(\left\{\phi^{(l)}\left(\mathbf{x}_i\right) : \mathbf{h}_i^{(l)} \in H^{(l)}\right\}\right)\right) \\
\mathcal{F}^{(L)}\left(H^{(L)}\right) &= \zeta^{(L)}\left(\rho^{(L)}\left(\left\{\phi^{(L)}\left(\mathbf{x}_i\right) : \mathbf{h}_i^{(L)} \in H^{(L)}\right\}\right)\right) \\
\mathcal{T}(X) &= \mathcal{F}^{(L)}\left(\mathcal{E}^{(L-1)}\left(\ldots \mathcal{E}^{(1)}\left(\{(\mathbf{x}_i) : \mathbf{x}_i \in X\}\right)\right)\right)
\end{aligned}$$

Here $\mathbf{h}_i^{(l)}$ denotes an arbitrary embedding from the set input $H^{(l)}$ at the $l^{th}$ hidden layer. The encoder block, $\mathcal{E}^{(l)}$, is stackable and incorporates a permutation-equivariant set function $\psi$ indicating the multi-head self-attention layer delineated in Vaswani *et al.,* 2017. The functions $\zeta$ and $\phi$ represent the position-wise fully-connected layers (FC). These layers are inherently permutation-invariant and are adeptly applied to feature embeddings. In the last layer $L$ a global pooling function $\mathcal{F}^{(L)}$ is employed. Subsequently, the signature of our architecture, $\mathcal{T}$, represents our set-based evolved Transformer encoder, obtained from Vaswani *et al.,* 2017. This encoder ingeniously amalgamates stacked permutation-equivariant layers, subsequently segueing into a permutation-invariant pooling function, preserving the structural integrity of our desired architecture.

To illustrate our model, we start with the inputs:

- We take $\hat{\mathbf{H}} = \{\hat{H}_i\}_{i=1}^{N} \in \mathbb{R}^{N \times d_k}$, representing the WSI-level consolidated pathogenic feature set.

- Additionally, we use $\mathbf{G} = \{G_i\}_{i=1}^{N} \in \mathbb{R}^{N \times d_k}$, symbolizing the genomic features.

To encode the feature embeddings of both $\hat{\mathbf{H}}$ and $\mathbf{G}$, two distinct set-based transformers, namely $\mathcal{T}_H$ and $\mathcal{T}_G$, are instantiated. The encoder block, $\mathcal{E}^{(l)}$, employs the multi-head self-attention layer, $\psi^{(l)}$, which can be formulated as:

$$\psi^{(l)}\left(\left\{\mathbf{h}_i^{(l)}\right\}_{i=1}^{M}\right) = \left\{\sum_{i=1}^{M} \frac{\exp\left(\mathbf{h}_i^{(l)}\mathbf{h}_j^{(l)\top}\right)}{d_k \sum_j \exp\left(\mathbf{h}_i^{(l)}\mathbf{h}_j^{(l)\top}\right)} \cdot \mathbf{h}_i^{(l)} \rightarrow \mathbf{h}_i^{(l+1)}\right\}$$

For each layer, the position-wise fully-connected layers $\phi^{(l)}$ and $\zeta^{(l)}$ are defined as:

$$\phi^{(l)}\left(\mathbf{h}_i^{(l)}\right) = \mathbf{W}_\phi^{(l)}\mathbf{h}_i^{(l)}$$
$$\zeta^{(l)}\left(\mathbf{h}^{(l)}\right) = \mathbf{W}_\zeta^{(l)}\mathbf{h}^{(l)}$$

At the terminal layer, the global attention pooling function, $\rho^{(L)}$, is articulated as:

$$\rho^{(L)}\left(\left\{\mathbf{h}_i^{(L)}\right\}_{i=1}^{M}\right) = \sum_{i=1}^{M} a_i \phi^{(L)}\left(\mathbf{h}_i^{(L)}\right) \rightarrow \mathbf{h}^{(l)} \text{ where}$$
$$a_i = \frac{\exp\left\{\mathbf{W}_\rho\left(\tanh\left(\mathbf{V}_\rho \mathbf{h}_i^{(L)\top}\right) \odot \text{sigm}\left(\mathbf{U}_\rho \mathbf{h}_i^{(L)\top}\right)\right)\right\}}{\sum_{j=1}^{M}\exp\left\{\mathbf{W}_\rho\left(\tanh\left(\mathbf{V}_\rho \mathbf{h}_j^{(L)\top}\right) \odot \text{sigm}\left(\mathbf{U}_\rho \mathbf{h}_j^{(L)\top}\right)\right)\right\}}$$

Here, the matrices $\mathbf{W}_\phi^{(l)}, \mathbf{W}_\zeta^{(l)}, \mathbf{W}_\rho, \mathbf{V}_\rho$, and $\mathbf{U}_\rho$ are trainable weight matrices of dimension $\mathbb{R}^{d_k \times d_k}$. The sigmoid function, sigm$(x)$, is given by $\frac{1}{1+e^{-x}}$. The attention score, $a_i$, quantifies the importance of the $i^{th}$ embedding $\mathbf{h}_i^{(L)}$ among the entire feature set $\mathbf{h}^{(L)}$.

In conclusion, the processed features derived from the outputs of both $\mathcal{T}_H$ and $\mathcal{T}_G$ are concatenated as $\left[\zeta_h^{(L)}\left(\mathbf{h}^{(L)}\right), \zeta_g^{(L)}\left(\mathbf{g}^{(L)}\right)\right]$. This integrated feature set is then channeled through several fully connected layers to deduce the ultimate survival prediction result.

## 3.3 Experiment & Results

### 3.3.1 Data and evaluation metric

Genomic data, including whole slide images (WSI), were obtained from the GDC Data Portal (https://portal.gdc.cancer.gov/). We use the TCGAbiolinks package v2.25.3 in the R statistical computing software environment v4.1.1 to retrieve the clinical data, gene expression (RNA seq), and mutation (MAF) from the TCGA project. The copy number data (discretized) was downloaded from Firehose (https://gdac.broadinstitute.org/). In our survival prediction study, we considered the following cancer types: Breast Invasive Carcinoma (BRCA) (n = 1022), Bladder Urothelial Carcinoma (BLCA) (n = 437), Glioblastoma and Lower Grade Glioma (GBMLGG) (n = 1011), Lung Adenocarcinoma (LUAD) (n = 515) and Uterine Corpus Endometrial Carcinoma (UCEC) (n = 538). The BRCA dataset includes 549 cases of luminal A, 206 cases of luminal B, 81 cases of HER2 + and 185 cases of basal-like. Clinical data contain patient codes, clinical characteristics, and survival labels,

including "vital status" (corresponding to the censorship indicator) and follow-up duration ("days to last follow-up" and "days to death"). We applied our proposed method to each cancer dataset and conducted a 5-fold cross-validation. We then used the cross-validated concordance index (c-Index) to measure the accuracy of our method in correctly ranking the predicted patient risk scores with respect to overall survival .

### 3.3.2 Model training

Patient-derived data was systematically stratified according to the type of cancer. This stratified data was then randomized into an 80% training dataset and a 20% validation dataset. Additionally, we implemented a five-fold cross-validation to further ensure model robustness. The validation dataset served as a metric for performance evaluation throughout the iterative development of our model. The model training was executed employing the Adam stochastic gradient descent optimization technique, implemented via the PyTorch v2.0.1 framework. While most default parameters were retained, the learning rate was specifically adjusted. The determination of the initial learning rates was achieved through a pre-training run utilizing a learning rate range test. This process involved the observation of training loss across a continuous spectrum of learning rate values.

The initial learning rate values adopted for the main model training ranged from $1 \times 10^{-4}$ to $5 \times 10^{-4}$. A scheduler was integrated, tasked with modulating the learning rate in response to potential learning plateaus. Typically, when there was an absence of discernible improvement in validation performance over four successive epochs, the learning rate was reduced by a factor of three. To further optimize our model's efficiency and counteract potential overfitting, an early stopping mechanism was employed. This mechanism preserved model states upon detecting learning stagnation, specifically prior to noting an escalation in validation loss. It was observed that models, when trained ab initio, generally reached convergence within a span of fewer than 20 epochs. Owing to the inherent heterogeneity in the sizes of the sample bags, we adopted a singular batch size, complemented with 32 steps of gradient accumulation.

### 3.3.3 Comparison with previously published models

Here, we undertook a rigorous evaluation of our proposed model's predictive capabilities, focusing on the BRCA, BLCA, GBMLGG, LUAD, and UCEC datasets. The configuration of the data set and the split of cross-validation used are kept same as the experiments of R. J. Chen *et al.,* 2021. The results of this evaluation are detailed in Table 3.1. For a comprehensive understanding and comparative assessment, we also considered the performance of several state-of-the-art models as described below:

1. **Deep Sets** Zaheer *et al.,* 2017:

    - *Description*: Deep Sets presents a neural network paradigm adept at handling sets characterized by variable sizes and order. This is achieved through the incorporation of aggregation functions.
    - *Significance*: Representing one of the pioneering neural network architectures tailored for set-based deep learning, Deep Sets capitalizes on sum pooling over feature instances.

2. **Attention MIL** Ilse *et al.,* 2018:

    - *Description*: Attention MIL, or Multi-Instance Learning, harnesses attention mechanisms to seamlessly aggregate data from multiple instances, facilitating predictions at the aggregate or 'bag' level.
    - *Significance*: Demonstrating its efficacy in diverse applications such as image classification and drug discovery, Attention MIL replaces the sum pooling inherent in Deep Sets with a global attention pooling strategy.

3. **MCAT** R. J. Chen *et al.,* 2021:

    - *Description*: The Multimodal Co-Attention Transformer (MCAT) integrates a co-attention-based early-fusion layer, a departure from the traditional late-fusion approach.

- *Significance*: When juxtaposed against preceding multimodal multiple instance learning frameworks, MCAT emerges superior, boasting an average performance increment of 8%-10%.

4. **Multilevel Image**:

   - *Description*: This approach exploits the multi-tiered architecture of pathology images. By leveraging an attention-driven hierarchical structure, it extracts three distinct levels of whole-slide image features.
   - *Significance*: These tri-tiered whole-slide image features are subsequently channeled into the Attention MIL model, notably in the absence of genomic data.

5. **Genomic-Guided Hierarchical Attention Multi-Scale Pathology Transformer (Full)**:

   - *Description*: Building upon the foundational MCAT model, we introduced our bespoke Genomic-Guided Hierarchical Attention Multi-Scale Pathology Transformer. Drawing inspiration from the inherent multi-level architecture of pathology images and the hierarchical nuances of text in natural language processing (NLP), an attention-centric hierarchical structure is employed.
   - *Significance*: This structure meticulously extracts three levels of whole-slide image features, which, through co-attention mechanisms, undergo early fusion with the patient's genetic data.

6. **Partial Attention Comparison**:

   - *Description*: We embarked on a comparative analysis to gauge the performance of our model in diverse scenarios—incorporating and excluding multi-level analysis and with and without the integration of attention-driven hierarchical feature extraction from whole-slide images. The model iteration devoid of hierarchical pathology attention feature extraction is denoted as 'multilevel'.

| Model | BLCA | BRCA | GBM/LGG | LUAD | UCEC | Average |
|---|---|---|---|---|---|---|
| Deep sets | 0.604 ± 0.042 | 0.521 ± 0.079 | 0.803 ± 0.046 | 0.616 ± 0.027 | 0.598 ± 0.077 | 0.629 |
| Attention MIL | 0.605 ± 0.045 | 0.551 ± 0.077 | 0.816 ± 0.011 | 0.563 ± 0.050 | 0.614 ± 0.052 | 0.630 |
| MCAT | 0.624 ± 0.034 | 0.580 ± 0.069 | 0.817 ± 0.021 | 0.620 ± 0.032 | 0.622 ± 0.019 | 0.653 |
| Multilevel Image | 0.579 ± 0.076 | 0.604 ± 0.070 | 0.802 ± 0.026 | 0.569 ± 0.068 | 0.634 ± 0.060 | 0.638 |
| Multilevel | 0.659 ± 0.025 | 0.648 ± 0.019 | **0.859** ± 0.024 | 0.666 ± 0.026 | 0.708 ± 0.061 | 0.708 |
| Full | **0.672** ± 0.029 | **0.680** ± 0.024 | 0.852 ± 0.023 | **0.673** ± 0.026 | **0.724** ± 0.059 | **0.720** |

Table 3.1: Performance of different models on various cancer types. Values represent the C-index with the standard deviation. BLCA: bladder urothelial carcinoma, BRCA: breast invasive carcinoma, GBM / LGG: glioblastoma and lower grade glioma, LUAD: lung adenocarcinoma, UCEC: uterine corpus endometrial carcinoma.

**Supremacy of Our Proposed Model Over Existing State-of-the-Art** Our proposed model exemplifies marked performance improvements relative to the current state-of-the-art pathonic genomics fusion methodologies employed in cancer survival pathology examinations. In particular, when juxtaposed against the MCAT model, the prevailing state-of-the-art, our full-fledged model registered an increase of 10.26% in the overarching c-Index. Meanwhile, the multi-tier model reflected a gain of 8.42%. Even when benchmarked against well-established baselines such as Deep Sets and Attention MIL, our full model depicted an impressive 14.29% surge in the overall c-Index, while the multi-level counterpart exhibited a 12.38% increment. Across the five cohorts under examination, our full model consistently superseded all other state-of-the-art contenders. The graph in Figure 3.6 demonstrates that our model was able to accurately predict the survival disparity between the high and low risk groups.

**Efficacy of Multilevel Whole-Slide Image (WSI) Feature Extraction** The potency of the multi-level model was underscored by its performance metrics, particularly when contrasted against the MCAT model. With an overall performance boost of 8.42%, the multi-level model showcased variability in its prowess - from a commendable 5.14% in the GBMLGG cohort to a staggering 13.83% in the UCEC cohort. Remarkably, while the Attention MIL model harnessed both whole-slide image features and genomic metrics, the multi-level model remained exclusive to the former. Yet, their performance metrics were remarkably proximate, gauged at an average c-Index of 6.38 for the multi-level model and 6.30 for the Attention MIL. Significantly, within the datasets for BRCA and UCEC, the

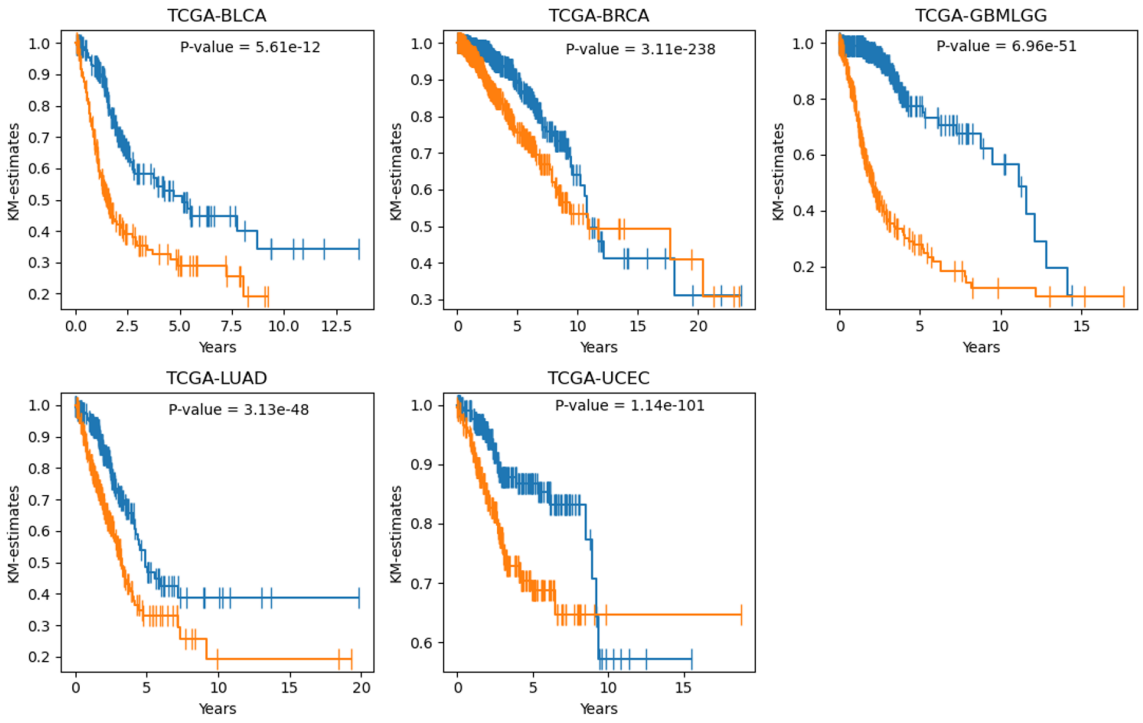



Figure 3.6: Kaplan-Meier Analysis was used to compare the survival time of low-risk (blue) and high-risk (red) patients based on their predicted risk scores from our model. The Logrank test (weight set as wilcoxon) was used to measure the statistical significance of the difference between the two survival distributions.

multi-level paradigm even eclipsed the MCAT model's performance.

**The Cutting Edge of Hierarchical Attention Feature Extraction** We compared the full model to its multi-level counterpart and found a slight but noticeable improvement in the c-Index of 0.12 on average. This improvement was seen in four out of five cohorts, with the full model outperforming the multi-level version. This suggests that an Attention-assisted hierarchical feature extraction framework can bring even small gains, which can be meaningful in terms of clinical significance. These small advances can lead to more accurate clinical decisions and patient management strategies.

### 3.3.4 Ablation Study: Dissecting the Role of Genomic Features

To gain a deeper understanding of the intricate relationship between various genomic features and their collective influence on the accuracy of cancer survival predictions, we executed an exhaustive ablation study. This analysis was built upon our comprehensive model and spanned a gamut of genomic inputs, encompassing:

- Complete Genomic Features (All)
- Mutation and Gene Expression (Mut+RNASEQ)
- Copy Number Variation and Gene Expression (CNV+RNASEQ)
- Gene Expression (RNASEQ) exclusively
- Mutation and Copy Number Variation (Mut+CNV)
- Mutation (Mut) in isolation
- Copy Number Variation (CNV) solely
- The Multilevel Image model, reliant predominantly on image data

The results are shown in Table 3.2. It is essential to emphasize that the MCAT paper's dataset mainly consists of gene expression data, with a small number of copy number variation data. To fill this data gap and allow for a thorough investigation, we created a custom dataset, including a slightly adjusted patient group. This strategic approach enabled us to gather a more comprehensive set of genomic information. As a result, the genomic dataset and the cross-validation group division differed from the specifications mentioned in the previous section.

Our research has highlighted the importance of gene expression as a major factor in the c-Index performance. The inclusion of mutation and copy number variation with gene expression had a small positive effect, but the lack of gene expression caused a significant decrease in performance. Interestingly, in some cases, the addition of either copy number or mutation data did not improve the accuracy of the predictions compared to scenarios without any genomic features. This emphasizes the value of gene expression in providing precise cancer survival predictions. Additionally, our findings demonstrate the inherent predictive power of whole-slide images in this context.

| Feature Type/Modality | BLCA | BRCA | GBM/LGG | LUAD | UCEC | Average |
|---|---|---|---|---|---|---|
| All | **0.675** ± 0.043 | **0.676** ± 0.081 | 0.843 ± 0.042 | **0.689** ± 0.043 | 0.731 ± 0.038 | 0.721 |
| Mut+RNASEQ | **0.675** ± 0.059 | 0.672 ± 0.023 | 0.841 ± 0.048 | **0.689** ± 0.046 | **0.737** ± 0.044 | **0.723** |
| CNV+RNASEQ | 0.667 ± 0.043 | 0.653 ± 0.074 | **0.850** ± 0.036 | 0.681 ± 0.032 | 0.720 ± 0.027 | 0.714 |
| RNASEQ | 0.654 ± 0.074 | 0.675 ± 0.039 | 0.838 ± 0.041 | 0.684 ± 0.042 | 0.728 ± 0.047 | 0.716 |
| Mut+CNV | 0.613 ± 0.048 | 0.596 ± 0.045 | 0.831 ± 0.047 | 0.599 ± 0.044 | 0.671 ± 0.043 | 0.662 |
| Mut | 0.565 ± 0.043 | 0.573 ± 0.041 | 0.788 ± 0.053 | 0.601 ± 0.059 | 0.696 ± 0.066 | 0.645 |
| CNV | 0.575 ± 0.056 | 0.616 ± 0.061 | 0.830 ± 0.049 | 0.595 ± 0.022 | 0.671 ± 0.045 | 0.657 |
| Image | 0.581 ± 0.087 | 0.594 ± 0.044 | 0.750 ± 0.039 | 0.569 ± 0.068 | 0.670 ± 0.053 | 0.633 |

Table 3.2: Performance comparison of different feature types and modalities on different cancer types. All means with all 3 genomics modality: RNASEQ, CNV and Mutation. Mut stands for Mutation, and Image means using no genomic modality data.

## 3.4 Interpretation of the full model

### 3.4.1 interpretation methods

**Attention weight interpretation** In the scaled dot-product attention module $\text{Attention}(Q, K, V)$, the weight of the value features, $\text{softmax}(\frac{QK^T}{\sqrt{d_k}})$, represents how the query features $Q$ attend to the key features $K$. For our tissue-level and organ-level hierarchical attention modules, the attention scores $(a_1, a_2, ..., a_{16}) \in A_{\text{tissue}}$ and $(a_1, a_2, ..., a_{256}) \in A_{\text{organ}}$ can be interpreted as how the model constructs tissue- and organ-level features from lower-level features. For the multilevel co-attention modules, which use attention-assisted hierarchical feature extraction to extract cell-, tissue-, and organ-level features from pathology images, the attention scores $(a_1, a_2, ..., a_M) \in A_{\text{coattn, organ}}$, $(a_1, a_2, ..., a_{16M}) \in A_{\text{coattn, tissue}}$, and $(a_1, a_2, ..., a_{256M}) \in A_{\text{coattn, cell}}$ represent how the model constructs new weighted whole-slide image features $\hat{\mathbf{h}}$ at the cell, tissue, and organ levels, respectively. These co-attention weights can reflect the biological relationship between genomic embeddings $g_n$ and pathonic features at the three different levels. We combine the attention weights for each level of the hierarchy by using level weights $w_{\text{organ}}, w_{\text{tissue}}, w_{\text{cell}}$ to obtain integrated attention weights. These attention weights can be represented as attention heatmaps for the $N$ genomic embeddings in $G$.

### 3.4.2 Attention Visualization in BRCA

We can gain insight into the relationship between genomic information and pathologic features by visualizing the genomic-guided WSI embeddings used as input in our full model. To do this, we compute the integrated co-attention weights from the 3-level whole slide image features and each genomic embedding $g_n$ for both low and high risk cases in the BRCA dataset. We then use these attention weights to generate hierarchical attention heatmaps, which highlight the regions of interest in the whole slide image that contribute most to the model's prediction. Additionally, we use Integrated Gradients to identify the top-10 genes in each embedding with the highest absolute attribution value. The results are illustrated in Figure 3.7 and Figure 3.8, allowing us to understand which genes are most important for the prediction of cancer risk for each patient.

### 3.4.3 Integrated gradient (IG) score indicating influential genes

Integrated Gradients is a sophisticated technique designed for attributing the importance of individual features to a model's predictive outcome. This method was first introduced by Z. Qi *et al.,* 2019. The core principle underpinning Integrated Gradients is to compute the integral of the model's output gradient concerning its input features. This calculation is performed over a path that extends from a baseline input, representative of feature absence, to the actual input in question. Such attributions facilitate a profound understanding of model predictions.

The versatility of this method is evident from its successful application across diverse domains ranging from natural language processing and computer vision to healthcare. Furthermore, empirical studies have demonstrated that Integrated Gradients is consistently more reliable than many prevailing attribution techniques.

Let's mathematically represent the concept. Given:

- $f$: A model where input $x$ produces output $y$

- $x'$: A baseline input, symbolizing the lack of features

The integrated gradient of $f$ concerning input $x$ for a particular input feature $x_i$ is denoted as:

$$\text{IntegratedGrads}_i(x) = (x_i - x_i') \times \int_{\alpha=0}^{1} \partial f(x' + \alpha \times (x - x'))/\partial x_i d\alpha$$

In this equation, $\partial f(x' + \alpha \times (x - x'))$ represents the partial derivative of the model's output with respect to the $i$-th input feature. This derivative is assessed at a specific point along the trajectory from the baseline $x'$ to the genuine input $x$. The parameter $\alpha$ aids in the path parameterization, and its integration occurs over the entire path.

In our research, the Integrated Gradients approach was essential. We used it to calculate integrated gradient scores for the genomic data of each patient. It was easy to interpret the results: a negative score indicated a gene's association with a lower risk of tumor, while a positive score indicated a higher risk of cancer.

We assessed the collective genomic impact of our deep learning model on a patient cohort by normalizing each patient's absolute integrated gradient score to a range of 0 to 1. We then took the average of these normalized absolute scores to obtain a group-level integrated gradient score. To ensure accuracy, we calculated the "absmax" integrated gradient score by taking the highest normalized absolute score from five iterations of a five-fold cross-validation. These scores act as markers of gene significance in different cancer types. Table 3.3 catalogs the genes of paramount influence, ranked by their "absmax" scores, for cancer cohorts including BLCA, GBMLGG, LUAD, and UCEC.

### 3.4.4 Influential gene analysis for BRCA subtypes

To gain a better understanding of the impact of genetics on the four distinct BRCA cancer subtypes—Her2, LumA, LumB, and Basal—we calculated the Average Absolute IG score for each subtype. We started by averaging the Integrated Gradients (IG) across all subtypes. As expected, the genes that were most prominent were those that are traditionally known to be general cancer genes, confirming the universal oncogenic pathways. However, a more subtype-specific averaging showed distinct IG signals for each subtype, demonstrating the subtle genetic differences that are unique to each subtype. Interestingly, the genes that had the most significant differences in IG scores between the subtypes have already been documented in the literature as breast cancer-associated genes. This correlation not only confirms the accuracy and biological relevance of our model but also highlights its potential

in identifying subtype-specific genetic signatures that are essential for targeted therapeutic interventions. In Table 3.4, the term "Absmax" refers to the highest Average Absolute IG score observed across five cross-validation models for all BRCA patients. On the other hand, the "SubtypeMax" denotes the peak Average Absolute IG score across these models, but only for patients of a single BRCA subtype.

Based on this, we obtain Table 3.5, which specifically spotlight genes with notable differences in their IG scores across subtypes. Within this table, the "Absmax" column captures the apex normalized absolute IG score spanning all BRCA patients. Concurrently, the "Maxsubtype" column reveals the maximum score among the four delineated BRCA subtypes. The computed "Difference" column highlights the disparity between the "Absmax" and "Maxsubtype" scores, and a larger difference may suggest that certain genes exert a more dominant influence within specific breast cancer subtypes. For ease of interpretation and to emphasize genes with heightened subtype specificity, the table entries are arranged in descending order based on the "Difference" values. Table 3.5 shows that 46 of the examined genes had higher variability when looking at the different subtypes compared to the average IG score among patients. Many of these genes have been identified as having a major impact on breast cancer, and are key factors in its various subtypes. Out of these 46 genes, only 7 (SMO, CLTC, SDHD, FNBP1, PNOC, GUCY2D, ITGA2B) have not been associated with the prognosis of breast cancer in the existing literature and therefore require further research.

- **PALB2**: Previous studies have revealed that breast cancer associated with PALB2 is particularly aggressive in terms of its clinicopathological features, particularly the triple-negative subtype, and has a higher mortality rate regardless of the tumor stage, the type of chemotherapy used, or the hormone receptor status (Antoniou *et al.,* 2014).

- **CDK12**: There is a growing body of evidence that CDK12 is associated with breast cancer. Furthermore, CDK12 appears to have different functions in the various subtypes of breast cancer, particularly in HER2-positive breast cancer and triple-negative breast cancer (TNBC) (S. Liang *et al.,* 2020).

- **CSF3**: High levels of G-CSF (CSF3) produced by cancer cells can activate neutrophils to create neutrophil extracellular traps, which can facilitate the movement of cancer

cells. This discovery provides insight into why the C3 (LumA) subtype is associated with a poor prognosis (L. Guo *et al.,* 2019).

- **BLM**: Tumors of the LumA type and the Genufu subtype (ER(+)/Her2(-)/low proliferation) were more likely to have low levels of BLM mRNA, suggesting that BLM could be a useful biomarker for breast cancer (Arora *et al.,* 2015).

- **ICAM2**: High concentrations of ICAM2 were found to increase the adhesion of the blood-cerebrospinal fluid barrier, facilitate the trans-barrier migration, and enhance the stemness abilities, thereby determining the specificity of leptomeningeal metastasis of triple-negative breast cancer. (Pan *et al.,* 2023)

- **PDGFRA**: The expression of PDGFRA and PDGF-CC were found to be highly associated with breast cancer that has an aggressive biological nature, such as the triple-negative subtype. Moreover, a high PDGF-CC expression was found to increase the risk of a distant recurrence within five years (Jansson *et al.,* 2018).

- **RIPK3**: The expression of RIPK3 was found to be strongly associated with increased proliferation and cystine-dependence in recurrent breast cancer (Stoll *et al.,* 2017). Moreover, the positive correlation between RIPK3 and lymphoid cells was limited to HER2+ and triple negative/basal-like breast cancer subtypes (Y. Xu *et al.,* 2022).

- **SDHB**: The HER-2 subtype of breast cancer was most commonly associated with high levels of SDHA expression in tumor cells, while the luminal A subtype was more likely to have low or no SDHA expression (P=0.032) (Kim *et al.,* 2013).

- **TGFB2**: The long noncoding RNA (lncRNA) TGFB2-AS1 is a significant regulator of the reversibility and plasticity of noncancer stem cell populations in TNBC, TGFB2-AS1 weakens the breast cancer stem-like cell (BCSC) characteristics of TNBC cells in the laboratory and significantly reduces tumorigenic frequency and lung metastasis in living organisms (C. Zhou *et al.,* 2022).

- **PIK3R1, AKT3**: PIK3R1 is a tumor suppressor gene related to the PI3K pathway, which is made up of oncogenes such as PIK3CA, AKT and MTOR (Pascual & Turner,

2019).

- **RARA**: Overexpression of RARA caused the breakdown of mammary luminal structures. Additionally, it was seen that the RARA-induced epithelial to mesenchymal transition (EMT) occurred in mammary epithelial cells (Doi *et al.,* 2015).

- **RET**: Research on RET has mainly been conducted in relation to diseases that are estrogen-receptor positive (ER+). Genomic changes in the RET gene were observed in all types of breast cancer, with the majority of cases being ER-negative (65%) or not amplified in ERBB2 (82%) (Paratala *et al.,* 2018).

- **THRA**: The alpha-2 variant of the thyroid hormone receptor (THRA-2) has been found to impede the effects of triiodothyronine (T3) and a low expression of this receptor has been linked to unfavorable tumor characteristics and a higher mortality rate in breast cancer patients.(Sandsveden *et al.,* 2021)

- **PTK2B**: Al-Juboori *et al.,* 2019 have confirmed that PTK2B (PYK2) plays a role in promoting the invasion of metformin-resistant HER2 breast cancer cells by examining the effect of PYK2 knockdown and metformin on cell invasion and by analyzing the related cellular pathways through proteomics. Additionally, they have found a correlation between high levels of PYK2 expression and decreased survival in pure HER2 breast cancer patients.

- **AURKA**: Amplification of the AURKA gene is a frequent genetic abnormality in breast cancer, particularly in tumors that have a basal-like appearance (Staff *et al.,* 2010).

- **ITGB4**: Previous research has demonstrated that ITGB4 is more abundant in tissues from patients with triple-negative breast cancer, as indicated in Table 3 of S. Lu *et al.,* 2008.

- **BRCA1, BRCA2**: Individuals with BRCA1 mutations are mainly prone to developing triple negative breast cancers, while those with BRCA2 mutations are more likely

to have tumors that are positive for estrogen and/or progesterone receptors (Rennert *et al.,* 2007).

- **KSR1**: Patients with breast cancer and high levels of KSR1 (a breast cancer tumor suppressor) had better disease-free and overall survival, additionally, KSR1 expression was found to be positively correlated with levels of breast cancer 1, early onset (BRCA1), BARD1, and Chk1 in breast cancer specimens (H. Zhang *et al.,* 2015).

- **NFIB**: NFIB could be a potential target for estrogen receptor-negative breast cancers. Research has revealed that NFIB protein was more abundant in ER negative breast cancer tissues, however, the expression level was comparable between HER2 subtype and triple negative subtype (Moon *et al.,* 2011).

- **MSI2**: In Triple Negative Breast Cancer (TNBC), a high level of MSI2a expression was found to be associated with a worse overall survival rate for patients. In both laboratory and real-world settings, an increase in MSI2a inhibited the invasion of TNBC cells and decreased the activity of the ERK1/2 protein (M. Li *et al.,* 2020).

- **ERBB2**: The HER2-Enriched (HER2-E) intrinsic subtype is characterized by the high expression level of ERBB2 (Schettini & Prat, 2021).

- **FGFR3**: Chew *et al.,* 2020 utilized mass spectrometry (MS) to analyze tyrosine phosphorylation in a subset of triple-negative breast cancer (TNBC) cell lines. The results revealed aberrant activation of the FGFR3 kinase, prompting further investigation into its potential as a therapeutic target.

- **SDHAF2**: Clinically, a reduction in the quantity of SDHAF2 is associated with a poorer prognosis for ER+ breast cancer patients in terms of relapse-free survival. Research has revealed that the accumulation of succinate, which is caused by a lack of SDH activity, is functionally linked to endocrine therapy resistance in ER+ breast cancer cells (Sengupta *et al.,* 2022).

- **C3**: C3, a major molecule in the complement pathway, was found to be highly expressed and its expression was inversely associated with that of CAS in breast cancer.

Lower C3 expression was linked to a worse prognosis. Additionally, C3 expression was positively correlated with the infiltration of multiple immune cells. Our results indicate that CAS may be involved in the progression of TNBC through C3-mediated immune cell suppression and could be a potential therapeutic target for TNBC (Ye *et al.,* 2022).

- **GAS7**: The wild-type form of the p53 gene increases the expression of GAS7, a gene associated with early-onset breast cancer, in order to inhibit metastasis through a signaling pathway that involves GAS7 and CYFIP1 (J.-W. Chang *et al.,* 2018).

- **JUN**: Activated c-Jun is primarily observed at the invasive forefront in breast cancer and has links with cell proliferation and blood vessel formation. Targeting c-Jun/AP-1 could offer novel approaches to inhibit tumor-related angiogenesis (Vleugel *et al.,* 2006).

- **CCL25**: CCL25, which is the only chemokine for CCR9+ cells, is not expressed in both human and mouse triple-negative breast cancers (TNBCs). This suggests that introducing CCL25 directly into the tumor might boost the effectiveness of immunotherapy in TNBCs (H. Chen, Cong, *et al.,* 2020).

- **HSPB8**: In estrogen receptor-positive breast cancer cells, the HSPB8 gene's expression is stimulated by estrogen. Additionally, this protein operates as a chaperone, working alongside Bag3, which promotes macroautophagy (Piccolella *et al.,* 2017).

- **FANCD2**: FANCD2 expression is lacking in 10–20% of both sporadic and BRCA1-related breast cancers, suggesting its somatic inactivation plays a role in both hereditary and non-hereditary breast cancer development, with FANCD2 also serving as an independent prognostic indicator in sporadic cases (Van Der Groep *et al.,* 2008).

- **CXCL1**: CXCL1/CXCR2 induces ER-negative breast cancer cell invasion and migration via the ERK1/2 pathway (C. Yang *et al.,* 2019).

- **STK11**: The tumor-suppressing LKB1/STK11 gene, linked to PJS, aids in inhibiting breast cancer, and its reduced expression in sporadic breast cancer correlates with decreased survival (Nakanishi *et al.,* 2005).

- **MAP2K6**: MAP2K6 is a prognostic indicator in breast cancer, with higher expression being associated with a more favorable outcome (X. Zhou *et al.,* 2020).

- **BUB1B**: Reducing BUB1B expression disrupts the SAC, causing chromosomal misalignment and abnormalities in breast cancer cells, leading to cell death, and its suppression in the TNBC cell line, MDA-MB-468, curtails tumor growth in mice, emphasizing SAC's pivotal role in breast cancer and its potential as a therapeutic target (Koyuncu *et al.,* 2021).

- **NBN**: Mutations in the NBN gene can lead to Nijemen breakage syndrome, which increases the likelihood of developing various illnesses, particularly a heightened risk of breast cancer. (Uzunoglu *et al.,* 2016).

- **EDN1**: Endothelin-1 genetic polymorphism as predictive marker for bevacizumab in metastatic breast cancer (Gampenrieder *et al.,* 2017).

- **EDN3**: Targeting EDN3 with epigenetic treatment and combining DNA demethylation agents with EDNR inhibitors could potentially restore ET-axis-mediated cellular signaling, which could provide therapeutic advantages for breast cancer (Wiesmann *et al.,* 2009).

- **POU2AF1**: According to Thalor *et al.,* 2022, machine learning-enhanced analysis of gene expression profiles in breast cancer has identified new potential prognostic markers for triple-negative breast cancer including POU2AF1.

### 3.4.5 Discussion

In a study by S. Li *et al.,* 2020, the framework begins with a CNN-based module for cancer detection, which assesses each patch in WSIs to predict tumor probability. Following this, they employ the Deep Conventional Gaussian Mixture Model (Zong *et al.,* 2018) to conduct unsupervised nuclear segmentation and to generate nuclear density maps. Tumor regions and Regions of Interest are identified and extracted using the generated probability heatmaps and nuclear density maps. Subsequently, a conditional autoencoder is applied to merge pathological features with genomic features. Lastly, the combined image and

gene features are processed through the DeepHit (C. Lee, Zame, *et al.,* 2018) model, which provides predictions regarding patient survival.

The biggest difference between this model and ours lies in its structure. The model in question comprises three separately trained components. It initially encodes the Whole Slide Images (WSIs) into features using ResNet, followed by the Deep Conventional Gaussian Mixture Model to obtain tumor probability. Only after this step are the regions of interest selected based on the tumor probabilities for input into the final DeepHit model. Unlike their model, which is not end-to-end, ours does not require the calculation of tumor probability for each WSI patch, making our model not only less complex in terms of training but also more robust.

It's important to note that their C-index is actually a time-dependent C-index (Antolini *et al.,* 2005). Although numerically higher than ours, it's not directly comparable as they are not the same type of C-index. In our model, the final survival classifier can be replaced with their DeepHit framework. This potential modification opens avenues for future research to experiment and see if our time-dependent C-index could perform better.

## 3.5 Conclusion

The study introduces the Genomic-Guided Hierarchical Attention Multi-Scale Pathology Transformer (GG-HAMPT), a cutting-edge multi-scale, multi-modal pan-cancer survival prediction system adept at processing inputs from gigapixel Whole Slide Images (WSIs) and varied genomic attributes, including RNA-sequenced gene expression, copy number variations, and mutation profiles. GG-HAMPT, born from an exhaustive analysis of different data modalities for each cancer cohort, employs intricate feature aggregation techniques, leveraging multi-scale hierarchical attention and co-attention mechanisms. This allows it to extract and formulate impactful internal data representations, crucial for accurate long-term survival forecasts across diverse cancer types. GG-HAMPT is strategically placed to take advantage of the ever-evolving world of precision medicine. It is designed to make the most of the immense amount of high-dimensional data available, offering not only improved prognosis but also the ability to make evidence-based clinical decisions for cancer patients. Its validity, underscored by integration gradient results aligning with prior academic findings,

cements GG-HAMPT's role as a revolutionary instrument in cancer prognosis, promising heightened precision in predictions and enriching patient management in oncology.

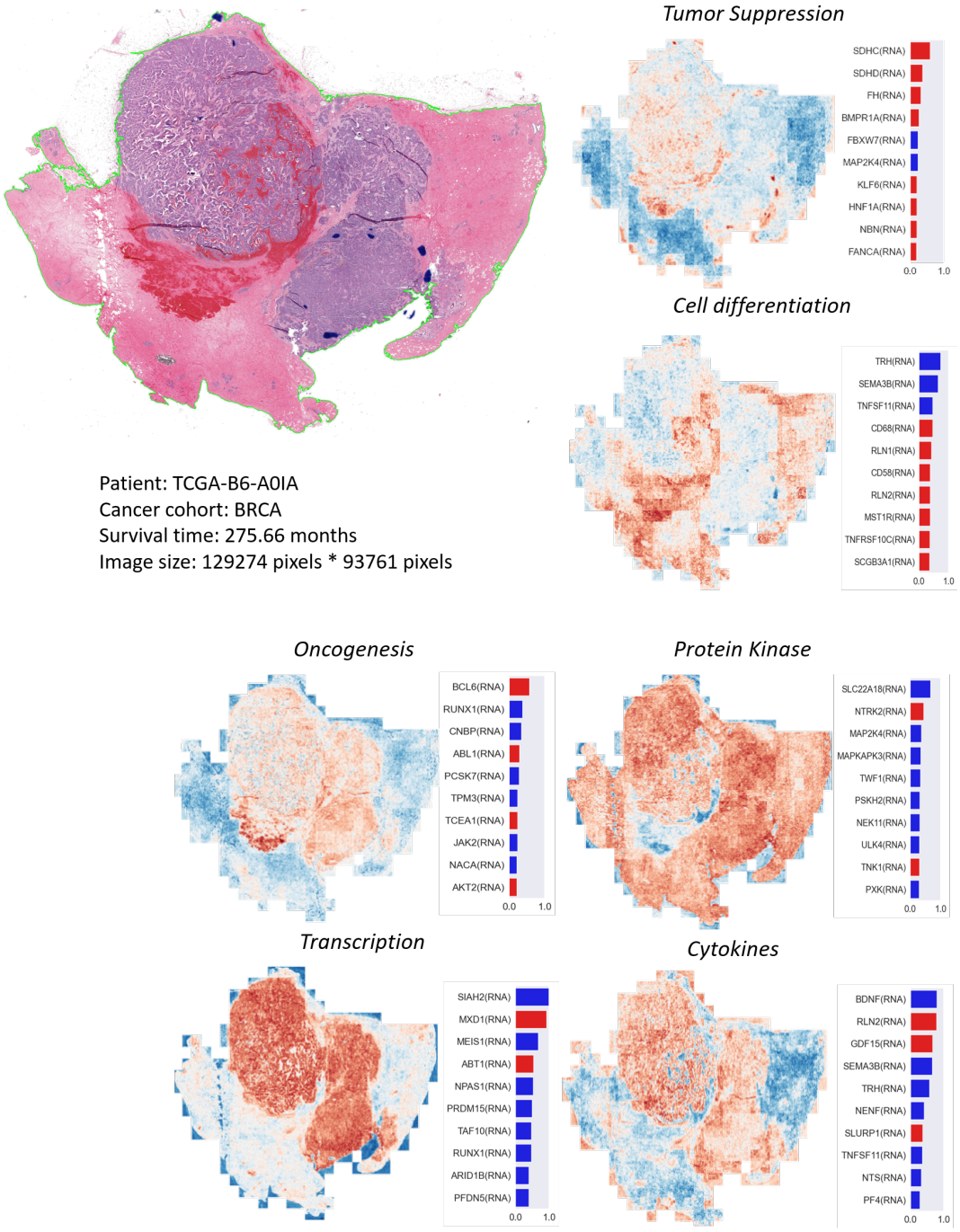



Figure 3.7: Co-attention visualization for low risk cases in BRCA patient TCGA-B6-A01A, with corresponding high attention patches and high-attributed genes in each heatmap.

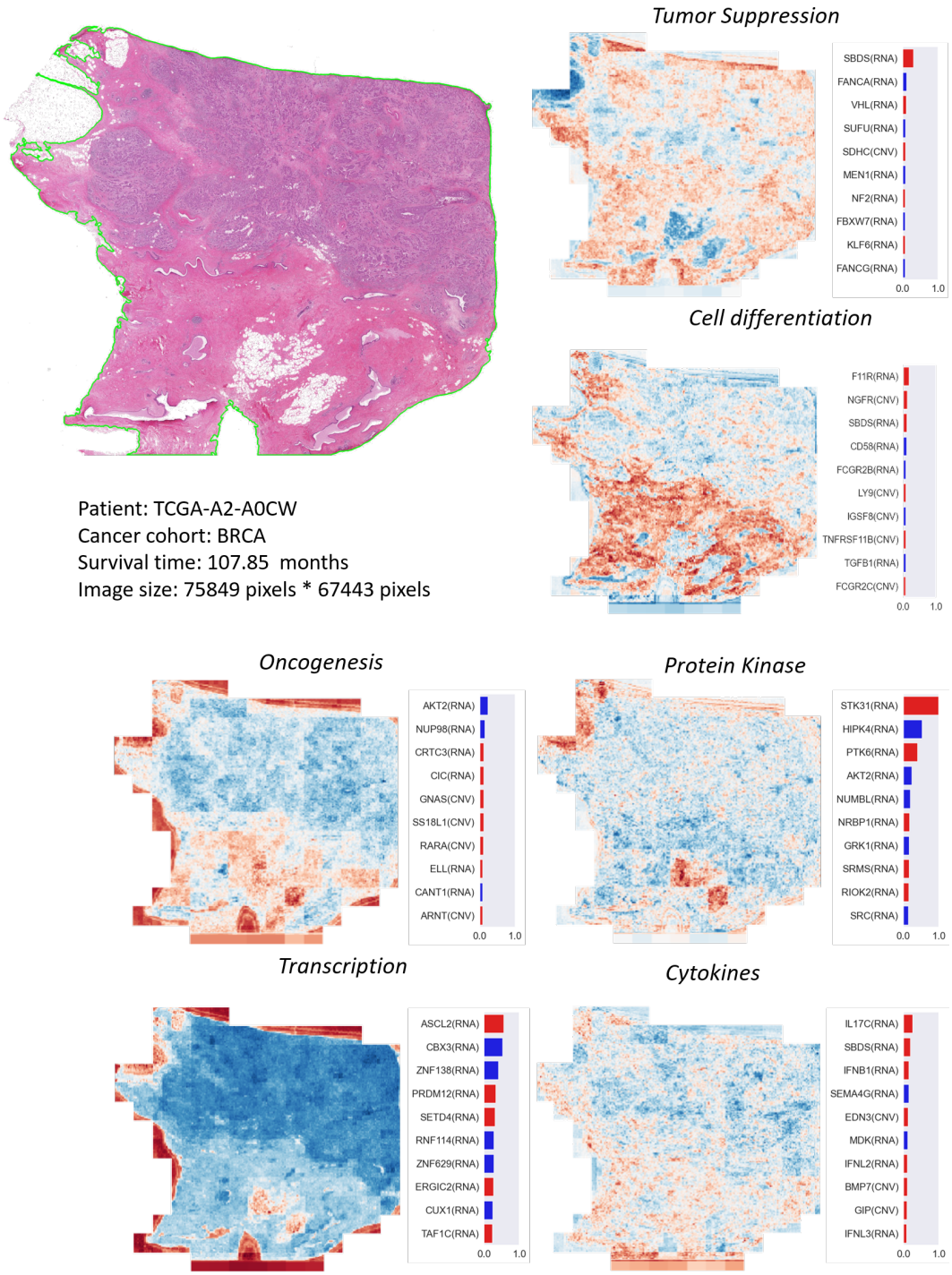



Figure 3.8: Co-attention visualization for high risk cases in BRCA, with corresponding high attention patches and high-attributed genes in each heatmap.

| BLCAgene | Absmax | GBMLGGgene | Absmax | LUADgene | Absmax | UCECgene | Absmax |
|---|---|---|---|---|---|---|---|
| ERBB2 | 1 | CD22 | 1 | PRF1 | 1 | CYLD | 1 |
| PMS1 | 1 | COL1A1 | 1 | MTCP1 | 1 | CD1C | 1 |
| CYLD | 1 | TNFRSF12A | 1 | REG1A | 1 | FANCE | 1 |
| BRCA1 | 1 | KLF6 | 1 | FAM3D | 1 | SPECC1 | 1 |
| FANCE | 1 | CGB8 | 0.955 | CBLB | 0.97 | KDM6A | 1 |
| KNL1 | 0.993 | CHEK2 | 0.875 | IL12A | 0.944 | CDC73 | 0.962 |
| CARD11 | 0.93 | PRF1 | 0.87 | EXT1 | 0.933 | PTCH1 | 0.955 |
| EP300 | 0.922 | TNFRSF11B | 0.787 | NBN | 0.906 | KDM5C | 0.92 |
| IL7 | 0.894 | DICER1 | 0.783 | ERBB2 | 0.903 | NF1 | 0.895 |
| SETD2 | 0.884 | NOTCH2 | 0.766 | KLF6 | 0.881 | FANCD2 | 0.871 |
| PRF1 | 0.845 | FCGR2B | 0.761 | BST1 | 0.835 | PHOX2B | 0.842 |
| ETV6 | 0.837 | F3 | 0.757 | CGB7 | 0.807 | CBLB | 0.836 |
| LAMP3 | 0.837 | MSN | 0.75 | NF2 | 0.791 | EP300 | 0.817 |
| CD3G | 0.821 | ADGRE2 | 0.742 | SDHC | 0.79 | TRPM7 | 0.8 |
| SOCS1 | 0.812 | VCAM1 | 0.708 | CXCL3 | 0.755 | RECQL4 | 0.796 |
| B3GAT1 | 0.78 | SIRPG | 0.706 | CD1B | 0.752 | XPA | 0.794 |
| CDH1 | 0.744 | EPHA10 | 0.703 | BLM | 0.745 | BRCA2 | 0.79 |
| NCR1 | 0.734 | DPP4 | 0.697 | FH | 0.744 | TSC2 | 0.764 |
| ETV1 | 0.732 | CD58 | 0.694 | ZXDA | 0.736 | FOXL2 | 0.752 |
| PDCD1 | 0.726 | PDGFA | 0.685 | ELF4 | 0.735 | HNF1A | 0.751 |
| FASLG | 0.717 | AKAP9 | 0.683 | RB1 | 0.733 | TP53 | 0.751 |
| CD74 | 0.707 | SDHB | 0.671 | LIFR | 0.728 | MLH1 | 0.732 |
| KLRC1 | 0.704 | CD101 | 0.669 | ANKK1 | 0.727 | BRIP1 | 0.731 |
| CD109 | 0.704 | PLAU | 0.651 | TNFAIP3 | 0.725 | FANCG | 0.724 |
| PF4 | 0.693 | ENPEP | 0.651 | GATA3 | 0.722 | BMPR1A | 0.714 |
| INSL3 | 0.692 | RPN1 | 0.648 | ZNF180 | 0.72 | APC | 0.693 |
| VHL | 0.687 | ATM | 0.643 | CLK3 | 0.714 | KLRB1 | 0.693 |
| CDKN2A | 0.684 | ALCAM | 0.638 | PHKG1 | 0.711 | NUAK1 | 0.683 |
| HMGA1 | 0.683 | GDF2 | 0.637 | ARNT | 0.704 | PALB2 | 0.676 |
| GAST | 0.672 | GDF15 | 0.632 | IL36A | 0.703 | SETD2 | 0.674 |
| ERCC2 | 0.671 | VEGFA | 0.629 | FASLG | 0.697 | CDH1 | 0.671 |
| CMTM5 | 0.662 | BCR | 0.627 | TPM3 | 0.695 | TRIB3 | 0.668 |
| HOXC9 | 0.654 | RET | 0.626 | PRKAA1 | 0.692 | BUB1B | 0.665 |
| CIITA | 0.648 | IL2RA | 0.621 | SOX9 | 0.69 | NSD1 | 0.661 |
| TSSK4 | 0.648 | PRKG1 | 0.621 | RNASEL | 0.69 | ERCC5 | 0.651 |
| LASP1 | 0.645 | AFF3 | 0.617 | ATM | 0.681 | EZH2 | 0.651 |
| TCF12 | 0.636 | MIA | 0.614 | CD1E | 0.68 | AURKB | 0.651 |
| IL3RA | 0.632 | CEBPA | 0.614 | CXCL2 | 0.679 | RBM15 | 0.649 |
| CD7 | 0.63 | TRIM33 | 0.611 | CCL13 | 0.677 | TFAP4 | 0.649 |
| CTLA4 | 0.628 | CDCP1 | 0.61 | TRIO | 0.675 | CR1 | 0.638 |
| KIR2DL4 | 0.625 | EGFR | 0.608 | L1CAM | 0.674 | DDB2 | 0.638 |
| EGFR | 0.615 | BLM | 0.608 | ARHGEF12 | 0.671 | CDC7 | 0.629 |
| TFRC | 0.611 | BUB1B | 0.608 | WNK4 | 0.666 | FBXW7 | 0.627 |
| PIK3R1 | 0.61 | TWF2 | 0.605 | MLLT10 | 0.663 | ALPK3 | 0.626 |
| NTF3 | 0.609 | ERCC2 | 0.604 | TLK2 | 0.662 | HMMR | 0.617 |
| IL2RG | 0.609 | MYCL | 0.602 | SETD2 | 0.66 | CDK15 | 0.616 |
| BLK | 0.609 | MUC1 | 0.6 | ERCC2 | 0.656 | INHBE | 0.614 |
| SUFU | 0.609 | | | ADM | 0.645 | FANCC | 0.613 |
| DKK1 | 0.605 | | | MLLT6 | 0.643 | GRK7 | 0.608 |
| BLM | 0.602 | | | TCL1A | 0.642 | MYC | 0.606 |
| | | | | NKX2-1 | 0.641 | ATIC | 0.606 |
| | | | | IL10RB | 0.636 | MSH2 | 0.605 |
| | | | | STK11 | 0.635 | IGF2R | 0.604 |
| | | | | NDP | 0.632 | | |
| | | | | SMARCB1 | 0.627 | | |
| | | | | DKK1 | 0.625 | | |
| | | | | CD47 | 0.624 | | |
| | | | | NF1 | 0.623 | | |
| | | | | ATOH1 | 0.623 | | |
| | | | | IRX1 | 0.621 | | |
| | | | | KDM5C | 0.619 | | |
| | | | | MIF | 0.615 | | |
| | | | | CCN1 | 0.614 | | |
| | | | | DPRXP6 | 0.613 | | |
| | | | | BIRC3 | 0.607 | | |
| | | | | FAS | 0.604 | | |

Table 3.3: Average normalized absolute IG score of all patients from five different cancer cohorts. The highest average normalized absolute IG score across five cross validation models is referred to as the "Absmax". These IG scores are used to demonstrate the significance of genetics of different cancer cohorts. Many cancer oncogenes are on the high-impact list.

| Gene | Absmax | Her2Max | LumAMax | LumBMax | BasalMax | Maxsubtype |
|---|---|---|---|---|---|---|
| PDGFRA | 0.407 | 0.962 | 0.79 | 0.565 | 1 | 1 |
| GUCY2D | 0.414 | 1 | 0.588 | 0.716 | 0.797 | 1 |
| BUB1B | 0.449 | 0.716 | 1 | 1 | 1 | 1 |
| CLTC | 0.462 | 0.803 | 0.66 | 1 | 0.838 | 1 |
| SDHD | 0.48 | 0.873 | 1 | 1 | 0.946 | 1 |
| SDHAF2 | 0.555 | 1 | 1 | 0.787 | 1 | 1 |
| ERCC5 | 0.665 | 1 | 1 | 1 | 1 | 1 |
| SAA1 | 0.686 | 1 | 0.586 | 0.568 | 0.743 | 1 |
| EP300 | 1 | 1 | 1 | 1 | 1 | 1 |
| BLM | 0.375 | 0.837 | 0.998 | 0.895 | 0.789 | 0.998 |
| GATA3 | 1 | 0.632 | 0.948 | 0.83 | 0.981 | 0.981 |
| EDN1 | 0.494 | 0.777 | 0.638 | 0.943 | 0.763 | 0.943 |
| SMARCB1 | 0.735 | 0.941 | 0.939 | 0.743 | 0.816 | 0.941 |
| CCL25 | 0.522 | 0.934 | 0.506 | 0.586 | 0.43 | 0.934 |
| PALB2 | 0.237 | 0.565 | 0.927 | 0.719 | 0.634 | 0.927 |
| CSF3 | 0.266 | 0.693 | 0.477 | 0.919 | 0.74 | 0.919 |
| SDHB | 0.347 | 0.604 | 0.906 | 0.717 | 0.598 | 0.906 |
| AMER1 | 0.556 | 0.904 | 0.637 | 0.629 | 0.825 | 0.904 |
| PIK3R1 | 0.366 | 0.896 | 0.58 | 0.618 | 0.565 | 0.896 |
| ITGB4 | 0.4 | 0.551 | 0.447 | 0.885 | 0.483 | 0.885 |
| CDK12 | 0.213 | 0.866 | 0.577 | 0.509 | 0.694 | 0.866 |
| ITGA2B | 0.167 | 0.757 | 0.509 | 0.63 | 0.861 | 0.861 |
| MEN1 | 0.537 | 0.858 | 0.517 | 0.606 | 0.472 | 0.858 |
| BRCA2 | 0.422 | 0.475 | 0.853 | 0.627 | 0.583 | 0.853 |
| NBN | 0.247 | 0.847 | 0.55 | 0.586 | 0.476 | 0.847 |
| ERBB2 | 0.386 | 0.767 | 0.592 | 0.838 | 0.579 | 0.838 |
| RARA | 0.303 | 0.823 | 0.752 | 0.801 | 0.808 | 0.823 |
| SIGLEC5 | 0.524 | 0.667 | 0.451 | 0.589 | 0.822 | 0.822 |
| FCGR2A | 1 | 0.596 | 0.478 | 0.812 | 0.548 | 0.812 |
| BRCA1 | 0.327 | 0.808 | 0.622 | 0.733 | 0.666 | 0.808 |
| FGFR3 | 0.354 | 0.589 | 0.628 | 0.805 | 0.66 | 0.805 |
| LHFPL6 | 0.441 | 0.607 | 0.488 | 0.801 | 0.52 | 0.801 |
| NFIB | 0.321 | 0.584 | 0.556 | 0.673 | 0.795 | 0.795 |
| MS4A1 | 0.503 | 0.747 | 0.513 | 0.795 | 0.627 | 0.795 |
| RET | 0.285 | 0.499 | 0.495 | 0.789 | 0.517 | 0.789 |
| MAP2K6 | 0.361 | 0.785 | 0.58 | 0.577 | 0.459 | 0.785 |
| CD5 | 0.414 | 0.625 | 0.476 | 0.784 | 0.509 | 0.784 |
| AKT3 | 0.344 | 0.371 | 0.777 | 0.607 | 0.569 | 0.777 |
| RECQL4 | 0.521 | 0.733 | 0.72 | 0.674 | 0.768 | 0.768 |
| FNBP1 | 0.282 | 0.763 | 0.561 | 0.529 | 0.757 | 0.763 |
| RIPK3 | 0.163 | 0.6 | 0.675 | 0.471 | 0.751 | 0.751 |
| TGFA | 0.404 | 0.657 | 0.446 | 0.547 | 0.74 | 0.74 |
| SMO | 0.28 | 0.739 | 0.505 | 0.63 | 0.734 | 0.739 |
| GAS7 | 0.31 | 0.727 | 0.499 | 0.547 | 0.681 | 0.727 |
| FSHB | 0.688 | 0.677 | 0.489 | 0.725 | 0.648 | 0.725 |
| BRD4 | 0.566 | 0.601 | 0.399 | 0.723 | 0.395 | 0.723 |
| ICAM2 | 0.095 | 0.71 | 0.435 | 0.511 | 0.484 | 0.71 |
| FANCA | 0.618 | 0.7 | 0.473 | 0.655 | 0.564 | 0.7 |
| FANCF | 0.49 | 0.375 | 0.698 | 0.505 | 0.543 | 0.698 |
| MYLK3 | 0.385 | 0.627 | 0.539 | 0.4 | 0.688 | 0.688 |
| HNRNPA2B1 | 0.395 | 0.683 | 0.446 | 0.502 | 0.624 | 0.683 |
| ATIC | 0.395 | 0.681 | 0.419 | 0.474 | 0.598 | 0.681 |
| PTK2B | 0.182 | 0.604 | 0.514 | 0.397 | 0.68 | 0.68 |
| STK11 | 0.277 | 0.403 | 0.665 | 0.68 | 0.497 | 0.68 |
| APLN | 0.437 | 0.68 | 0.257 | 0.432 | 0.404 | 0.68 |
| CEACAM5 | 0.47 | 0.494 | 0.405 | 0.673 | 0.447 | 0.673 |
| GAST | 0.312 | 0.586 | 0.357 | 0.672 | 0.437 | 0.672 |
| JAK2 | 0.314 | 0.469 | 0.492 | 0.587 | 0.672 | 0.672 |
| TGFB2 | 0.13 | 0.664 | 0.363 | 0.624 | 0.506 | 0.664 |
| THRA | 0.161 | 0.664 | 0.375 | 0.405 | 0.392 | 0.664 |
| CXCL1 | 0.254 | 0.662 | 0.281 | 0.419 | 0.338 | 0.662 |
| CD274 | 0.272 | 0.662 | 0.335 | 0.484 | 0.395 | 0.662 |
| MN1 | 0.592 | 0.537 | 0.385 | 0.659 | 0.393 | 0.659 |
| SIGLEC1 | 0.583 | 0.563 | 0.439 | 0.653 | 0.459 | 0.653 |
| KSR1 | 0.167 | 0.554 | 0.491 | 0.395 | 0.648 | 0.648 |
| CLTCL1 | 0.446 | 0.529 | 0.384 | 0.648 | 0.387 | 0.648 |
| AURKA | 0.156 | 0.544 | 0.531 | 0.467 | 0.643 | 0.643 |
| FGF7 | 0.488 | 0.643 | 0.436 | 0.504 | 0.556 | 0.643 |
| BRIP1 | 0.323 | 0.629 | 0.64 | 0.619 | 0.626 | 0.64 |
| PTGFRN | 0.575 | 0.373 | 0.426 | 0.638 | 0.426 | 0.638 |
| MRTFA | 0.242 | 0.574 | 0.303 | 0.609 | 0.635 | 0.635 |
| CD52 | 0.423 | 0.342 | 0.409 | 0.634 | 0.348 | 0.634 |
| JUN | 0.216 | 0.478 | 0.484 | 0.629 | 0.469 | 0.629 |
| CBLB | 0.421 | 0.373 | 0.627 | 0.494 | 0.518 | 0.627 |
| CD58 | 0.404 | 0.475 | 0.353 | 0.626 | 0.419 | 0.626 |
| PTEN | 0.803 | 0.366 | 0.622 | 0.479 | 0.494 | 0.622 |
| SCG2 | 0.461 | 0.411 | 0.338 | 0.615 | 0.45 | 0.615 |
| EDN3 | 0.137 | 0.614 | 0.464 | 0.564 | 0.475 | 0.614 |
| LILRB5 | 0.806 | 0.408 | 0.391 | 0.613 | 0.356 | 0.613 |
| FANCD2 | 0.202 | 0.612 | 0.523 | 0.495 | 0.591 | 0.612 |

Table 3.4: Average absolute IG score of all patients from 4 different BRCA cancer subtypes: Her2, LumA, LumB, Basal. The highest average normalized absolute IG score across five cross validation models among all BRCA patients is referred to as the "Absmax", the highest average normalized absolute IG score across five cross validation models among all BRCA patients from a specific subtype is referred to as the "SubtypeMax". Maxsubtype indicating the maximum of 4 "SubtypeMax". These IG scores are used to demonstrate the significance of BRCA subtype specific genetics. This table was sorted by the Maxsubtype.

| Gene | Absmax | Maxsubtype | Difference | Gene | Absmax | Maxsubtype | Difference |
|---|---|---|---|---|---|---|---|
| ITGA2B | 0.167 | 0.861 | 0.694 | PALB2 | 0.237 | 0.927 | 0.69 |
| CDK12 | 0.213 | 0.866 | 0.653 | CSF3 | 0.266 | 0.919 | 0.653 |
| BLM | 0.375 | 0.998 | 0.623 | ICAM2 | 0.095 | 0.71 | 0.615 |
| NBN | 0.247 | 0.847 | 0.6 | PDGFRA | 0.407 | 1 | 0.593 |
| RIPK3 | 0.163 | 0.751 | 0.588 | GUCY2D | 0.414 | 1 | 0.586 |
| SDHB | 0.347 | 0.906 | 0.559 | BUB1B | 0.449 | 1 | 0.551 |
| CLTC | 0.462 | 1 | 0.538 | TGFB2 | 0.13 | 0.664 | 0.534 |
| PIK3R1 | 0.366 | 0.896 | 0.53 | RARA | 0.303 | 0.823 | 0.52 |
| SDHD | 0.48 | 1 | 0.52 | RET | 0.285 | 0.789 | 0.504 |
| THRA | 0.161 | 0.664 | 0.503 | PTK2B | 0.182 | 0.68 | 0.498 |
| AURKA | 0.156 | 0.643 | 0.487 | ITGB4 | 0.4 | 0.885 | 0.485 |
| FNBP1 | 0.282 | 0.763 | 0.481 | BRCA1 | 0.327 | 0.808 | 0.481 |
| KSR1 | 0.167 | 0.648 | 0.481 | EDN3 | 0.137 | 0.614 | 0.477 |
| NFIB | 0.321 | 0.795 | 0.474 | MSI2 | 0.105 | 0.574 | 0.469 |
| SMO | 0.28 | 0.739 | 0.459 | ERBB2 | 0.386 | 0.838 | 0.452 |
| FGFR3 | 0.354 | 0.805 | 0.451 | POU2AF1 | 0.13 | 0.581 | 0.451 |
| EDN1 | 0.494 | 0.943 | 0.449 | SDHAF2 | 0.555 | 1 | 0.445 |
| C3 | 0.134 | 0.576 | 0.442 | AKT3 | 0.344 | 0.777 | 0.433 |
| BRCA2 | 0.422 | 0.853 | 0.431 | MAP2K6 | 0.361 | 0.785 | 0.424 |
| GAS7 | 0.31 | 0.727 | 0.417 | JUN | 0.216 | 0.629 | 0.413 |
| CCL25 | 0.522 | 0.934 | 0.412 | PNOC | 0.145 | 0.556 | 0.411 |
| HSPB8 | 0.085 | 0.495 | 0.41 | FANCD2 | 0.202 | 0.612 | 0.41 |
| CXCL1 | 0.254 | 0.662 | 0.408 | STK11 | 0.277 | 0.68 | 0.403 |

Table 3.5: Comparison of Average Absolute IG Scores Across BRCA Cancer Subtypes (Positive difference only). The table lists genes with their corresponding "Absmax" scores, which represent the highest average normalized absolute IG score across all BRCA patients, and the "Maxsubtype" scores, indicating the maximum of the four subtype-specific scores. The "Difference" column illustrates the difference between the "Absmax" and "Maxsubtype" scores. Genes with a larger difference may have a more pronounced role in specific breast cancer subtypes. This table is sorted by the "Difference" in descending order to highlight genes of potential subtype-specific significance.

# Chapter 4

# Domain adaptation of foundation models for image classification in digital pathology

## 4.1 Abstract

The accuracy of classifying images in digital pathology is essential for correct diagnosis and has major implications for patient outcomes. Nevertheless, traditional machine learning models often struggle to move from general images to the specialized domain of pathological images (Razzak *et al.,* 2018). We present the LoRA Vision Transformer (LoRA-ViT), a novel adaptation of existing models that is especially adept at classifying images in digital pathology. By using Low-Rank Adaptation (LoRA) techniques, LoRA-ViT can be easily adjusted to the unique features and distribution of pathological data without needing to be extensively retrained or tuned to a specific dataset. This approach helps reduce the domain shift issue, refine the model's focus and take advantage of the pre-trained representations. The adaptability and computational efficiency of LoRA-ViT stand to offer substantial advancements in automated pathology, ultimately supporting the broader integration of artificial intelligence in diagnostic medicine.

## 4.2 Introduction

### 4.2.1 Digital pathology

Pathology, the cornerstone of precise diagnosis in healthcare, has traditionally been grounded in the examination of biopsy samples on glass slides (Acs *et al.,* 2020). This practice, while effective, is constrained by the need for physical presence for examination (Jahn *et al.,* 2020). Digital pathology represents a transformative approach, leveraging digital technology to usher in a new era of accurate diagnosis in the field.

The advent of digital pathology brought significant advancements in imaging technology, including high-speed scanners capable of capturing detailed images of histological slides, greatly expanded access to information and analysis opportunities (Wright *et al.,* 2013).Central to digital pathology is its ability to transcend the physical barriers of slide examination. Pathologists can now examine, interpret, and analyze digitized slide images with enhanced clarity and precision (Aeffner *et al.,* 2019). The benefits of this digital shift are manifold. It not only improves diagnostic accuracy through superior image quality but also integrates quantitative analysis tools, augmenting the expertise of pathologists.

### 4.2.2 Deep learning in digital pathology

Deep learning is poised to revolutionize the domain of histopathology. Traditionally, pathologists expert have been responsible for the meticulous task of examining stained tissue specimens under a microscope, often to diagnose serious conditions such as cancer (Spanhol *et al.,* 2015). This manual inspection, while thorough, can be extremely time-consuming and sometimes subjective. The emergence of new deep learning technologies has opened up the possibility of quickly and automatically analyzing and interpreting large amounts of digitized whole slide images (WSIs). These advances not only have the potential to streamline traditional processes, significantly reducing the amount of time pathologists spend manually examining slides, but also add a quantitative element of accuracy. For instance, rather than relying on a pathologist's qualitative assessment, it is now feasible to use deep learning-driven approaches to quantify specific metrics, like the number of tumor-infiltrating lymphocytes across entire WSIs with unparalleled accuracy (Klauschen *et al.,* 2018).

This transformative shift towards automated histopathology is attributed to the consis-

tent advancements in deep learning techniques. Recent deep learning methodologies have obtained exemplary results in various histopathology analysis tasks (Niazi *et al.,* 2019, Bera *et al.,* 2019, Van der Laak *et al.,* 2021). More impressively, some of these state-of-the-art machine learning models have achieved performance metrics comparable to, and in some cases exceeding, experienced human pathologists in specific diagnostic tasks (Xiaoxuan *et al.,* 2019). As the field progresses, it is anticipated that integration of deep learning with histopathological practices will further enhance the precision, efficiency and reliability of diagnoses, setting new standards for patient care and medical research.

### 4.2.3 Challenges: generalizability

In the past decade, deep learning has witnessed significant advances in digital pathology, both in refining algorithmic accuracy and pioneering innovative methodologies. However, the journey to fully integrate these deep learning methods into clinical settings is riddled with multifaceted challenges.

The most significant challenge is the generalizability of deep learning digital pathology algorithms to clinical practice. Although the size of datasets used to create deep learning digital pathology algorithms has grown significantly in recent years, they still do not reflect the data typically encountered in clinical settings (Abels *et al.,* 2019). The data employed in academic studies often oversimplify the rich variations seen in practice. Additionally, factors such as different patient demographics across different institutions or geographical areas can lead to subtle biases that are not adequately addressed in current deep learning digital pathology algorithms.

Therefore, deep learning digital pathology algorithms perform optimally on data from the source(s) they were trained on, but not as well on data from other sources. For instance, when a prostate cancer detection model was applied to whole slide images (WSIs) from the same dataset used for algorithm construction, but rescanned on a different WSI scanner, the model's performance dropped by 5.84% (Campanella *et al.,* 2019). Other studies have also reported similar performance drops when using external test data (de Bel *et al.,* 2018, Y. Liu *et al.,* 2019).

### 4.2.4 Foundation models in computer vision

The evolution of computer vision has been marked by pivotal advancements, primarily spurred by the introduction and continual refinement of deep learning models. The main impetus for this transformation has been the evaluation of datasets such as ImageNet (J. Deng *et al.,* 2009), which has become known as the ultimate testing ground for the state of the art (SOTA) models. The ImageNet dataset, comprising a diverse range of 1000 common classes, provides an ideal grand set against which new models can be evaluated.

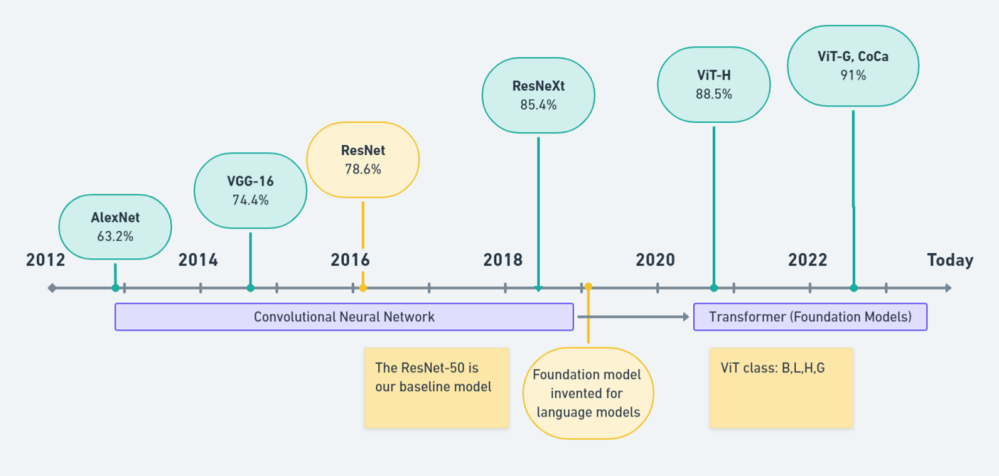



Figure 4.1: Computer Vision Benchmark on ImageNet (Top 1 accuracy / 1000 classes)

Figure 4.1 shows the different models and their accuracy on ImageNet (Top 1 accuracy / 1000 classes). In 2012, a major advancement was made with the debut of AlexNet (Krizhevsky *et al.,* 2012). This marked the onset of the era where convolution neural networks (CNNs) began to redefine the paradigms of image classification. Their inherent ability to hierarchically extract features from images set them apart, leading to superior performance. Subsequently, 2014 witnessed the evolution of Visual Geometry Groups Very Deep Convolutional Networks (VGG)(Simonyan & Zisserman, 2014), which expanded upon the layers introduced by AlexNet, enhancing depth and complexity. However, a monumental change occurred toward the end of 2015 when Kaiming He introduced ResNet (K. He *et al.,* 2016a). With its ingenious design of a bypass shortcut, ResNet effectively circumvented the degradation problems often associated with profoundly deep networks. Subsequently, this design has been the cornerstone of numerous successful networks. Its balanced combination of robust performance and simplicity makes it an ideal choice for training or fine-tuning

purposes.

As time went on, models became more complex and accurate. ResNeXt (S. Xie *et al.,* 2017) was a demonstration of this, further developing the basic elements of ResNet. However, it was in 2019 that the field underwent a major transformation. The advent of large-scale pretrained foundation models, especially those ingrained in transformer architecture, revolutionized the field. Originally conceived for natural language processing (NLP), the potential of these models in computer vision began to unfold. Up until 2020, CNNs were the predominant choice for computer vision tasks. However, the landscape began to change with the emergence of the Vision Transformer (ViT) towards the end of 2020. Drawing inspiration from the success of transformers in NLP, ViT showcased the potential of these architectures to decipher visual data (Dosovitskiy *et al.,* 2020). This marked the beginning of transformer-based foundation models that overshadowed their CNN counterparts in terms of performance. Subsequent models, like ViT-G (indicating "Giant"), escalated the number of parameters. Similarly, hybrid models such as CoCa (J. Yu *et al.,* 2022) amalgamated convolution and transformer architectures, thus defining themselves as the contemporary state of the art models.

Recently, Meta AI has made a groundbreaking advancement in the field of computer vision with the introduction of the Segment Anything Model (SAM) (Kirillov *et al.,* 2023). This model combines deep learning and extensive dataset training to enable accurate segmentation of any object in an image. Prompts can be a single point, a group of points (including a full mask), a bounding box, or text. The model is expected to generate a valid segmentation mask even when the prompt is ambiguous. What makes SAM unique is its capacity for *zero-shot generalization*, which means that it does not require additional training when presented with new objects. This is made possible by its training alongside the three-stage development of the SA-1B dataset with of over 11 million images and more than a billion masks. To generate this dataset, firstly human annotators improved the masks created by SAM from public datasets with manual adjustments; then, they increased the variety of objects by dealing with SAM's less certain mask predictions, leading to the ultimate segmentation created by SAM through grid-distributed points across pictures, only selecting the most reliable results. Potential applications of SAM are far-reaching, as demonstrated

by its ability to precisely delineate any desired object with a single click or to autonomously sample points to segment multiple entities, as seen with the Brooklyn Bridge in New York City (Figure 4.2).

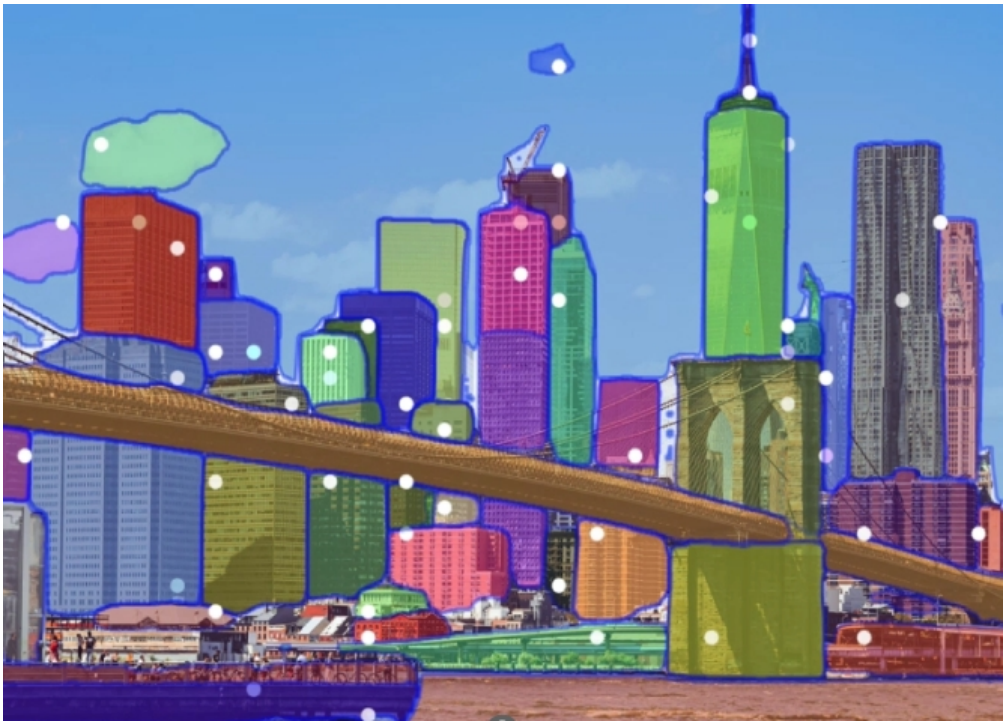


Figure 4.2: Segment Anything Model demo with Brooklyn Bridge (Kirillov *et al.,* 2023).

Reflection on the potential cross-disciplinary applications of cutting-edge advancements is warranted. How can these sophisticated tools be effectively incorporated into the biology and medical field? In (Mazurowski *et al.,* 2023) the authors assessed SAM across 19 medical imaging datasets, demonstrating inconsistent performance across different tasks. This implies that while SAM has potential in medical image segmentation, its application must be carefully considered, with a lot of room for improvement.

### 4.2.5 Challenges in foundation model domain adaptation

In the medical domain and digital pathology analysis, adapting foundation models presents a distinct set of challenges. The main problem is caused by the specialized nature of the medical images we are examining, which is significantly different from the regular images that traditional computer vision activities manage. This specialization results in significantly smaller datasets—typically ranging from several hundreds to a few thousand im-

ages—compared to the multi-million image datasets like the 11 million images used to train models such as SAM. The lack of medical images is not only evident in their quantity, but also in the quality of data needed. Each image in our datasets contains a wealth of important information, and labeling them requires specialized knowledge, which is in stark contrast to the ability of companies such as Meta to use less costly labor for annotation tasks in more general areas.

Medical applications usually require extremely high accuracy. Currently, people use simpler, yet reliable models, such as ResNet50, which are trained for each particular dataset and issue. One question naturally occurs: why can't we exploit the capacities of large-scale, pre-trained base models?

The tremendous size of pretrained foundation models makes them resource-intensive, a critical consideration given the often constrained resources in medical research settings. Even with the traditional training method, the performance of these models often fails to meet the precision required in medical applications. In essence, while the foundation models hold promise, the journey to effectively adapt them to medical imagery is fraught with obstacles that require innovative solutions to bridge the gap between general-purpose vision models and the nuanced needs of medical image analysis. Therefore, in this section we investigate methods to adapt these foundational models into the biology and medical domains.

## 4.3 Method

### 4.3.1 Vision Transformer

Motivated by the success of the Transformer in the field of natural language processing, some researchers have investigated whether similar models can learn useful representations for images. Vision Transformer (ViT) (Dosovitskiy *et al.,* 2020) is a pure transformer that is applied directly to sequences of image patches for image classification tasks. It follows the original design of the Transformer as closely as possible. Figure 4.3 illustrates the ViT framework.

To process 2D images, the image $X \in \mathbb{R}^{h \times w \times c}$ is reshaped into a series of flattened 2D patches $X_p \in \mathbb{R}^{n \times \left(p^2 \cdot c\right)}$, where $c$ is the number of channels. The resolution of the original image is $(h, w)$, while the resolution of each image patch is $(p, p)$. The effective sequence

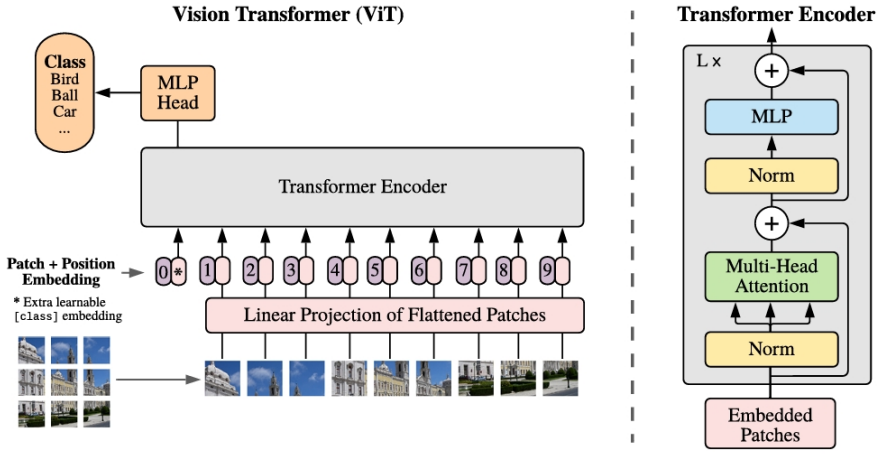



Figure 4.3: The framework of ViT (Dosovitskiy *et al.*, 2020). The image was divided into patches of a fixed size, ordered, then embedded linearly, and position embeddings were added. The sequence of vectors that resulted was then fed to a regular Transformer encoder. To classify, the usual approach of adding a trainable "classification token" to the sequence was used.

length for the transformer is $n = hw/p^2$. A trainable linear fully connected layer is used to transform each patch embedding size to $d$, and the output of this projection is referred to as patch embeddings.

A trainable classification layer is applied to the sequence of embedding patches, similar to the classification token of BERT (Devlin *et al.*, 2018). This embedding serves as the image representation, and position embeddings are added to the patch embeddings to preserve positional information. ViT only utilizes the standard transformer encoder, with an MLP layer following its output. In most computer vision applications, ViT is pre-trained on large datasets and then the MLP layer is fine-tuned for downstream tasks with smaller data.

### 4.3.2 LoRA

The evolution of large-scale transformer-based networks has led to a significant increase in the number of parameters, posing challenges in model tuning and application. One promising solution to this problem is the Low-Rank Adaptation (LoRA) technique (E. J. Hu *et al.*, 2021), which offers a pathway to fine-tune massive foundational models efficiently. LoRA works by updating a bypass layer within the network instead of the whole parameter matrix, which makes the fine-tuning process more efficient. The bypass layer serves as a shortcut for parameter updates, potentially reducing the number of trainable parameters to as little as

1/10000 of the original model. The LoRA method can be selectively applied to specific weight matrices within a neural network, such as the four weight matrices in the Transformer's self-attention module ($W_q, W_k, W_v, W_o$) and the two in the MLP module, to reduce trainable parameters.

Research (Aghajanyan *et al.*, 2020) on the "intrinsic dimensionality" of pretrained language models suggests that after fine-tuning on downstream tasks, these models have a low intrinsic rank in their weight matrices. This implies that high-dimensional representations of large models may contain redundancy. LoRA therefore hypothesizes that the changes to the weights during adaptation have a low "intrinsic rank".

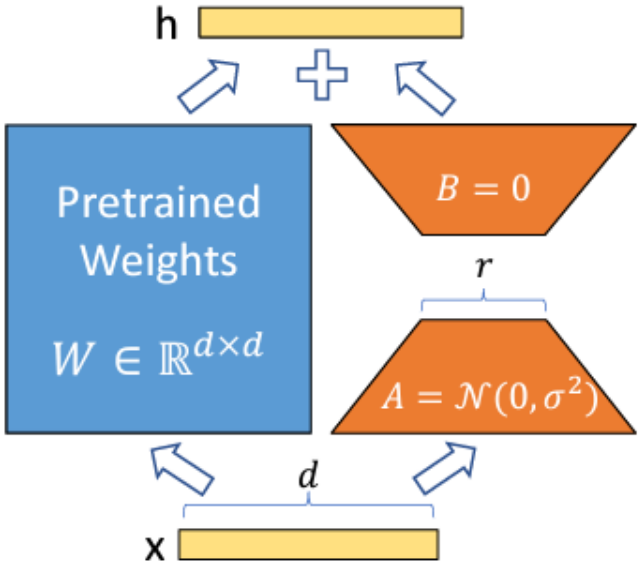



Figure 4.4: The LoRA layer. Only A and B will be trained.

The LoRA method constrains the update of the pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$ by employing a low-rank decomposition $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and the rank $r \ll \min(d, k)$. In this approach, $W_0$ remains static during training, not subject to gradient updates, while $A$ and $B$ are designated as trainable parameters. The LoRA method applies both $W_0$ and $\Delta W = BA$ to the same input, and their output vectors are summed coordinate-wise. For an input vector $x$ and output $h = W_0 x$, the modified forward pass that LoRA employs is articulated as:

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

The reparametrization technique of LoRA is illustrated in Figure 4.4. Initially, $A$ is randomly assigned a Gaussian distribution and $B$ is set to zero, making $\Delta W = BA$ start from zero. During the forward pass, LoRA scales $\Delta W x$ by $\frac{\alpha}{r}$, with $\alpha$ as a constant scaling factor and $r$ the rank. The rank is a hyper parameter of the LoRA method.

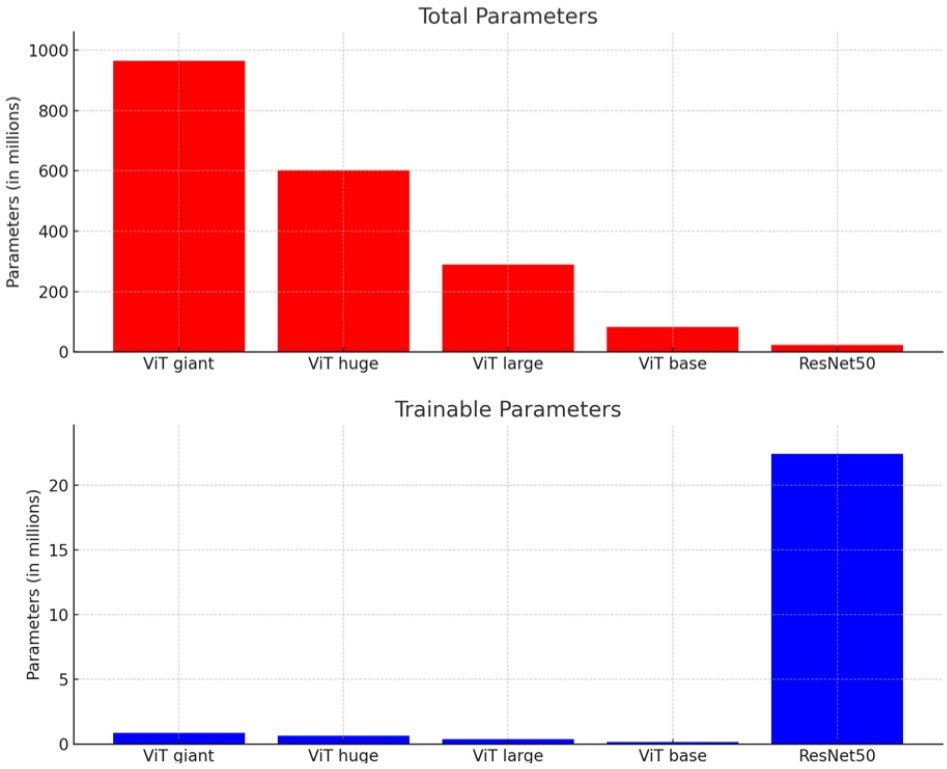



Figure 4.5: Comparison between transformer models tuned with and without LoRA and ResNet

The Low-Rank Adaptation (LoRA) technique offers a significant reduction in the number of trainable parameters and GPU memory requirements for Vision Transformer (ViT) models. The structure of LoRA layer applied in ViT (Vision Transformer) is illustrated in Figure 4.6. When employing LoRA, the Vision Transformer models exhibit a dramatic decrease in trainable parameters, as shown in Figure 4.5:

- **ViT-Giant:** Trainable parameters are reduced to 0.868 million from a total of 965.228 million.

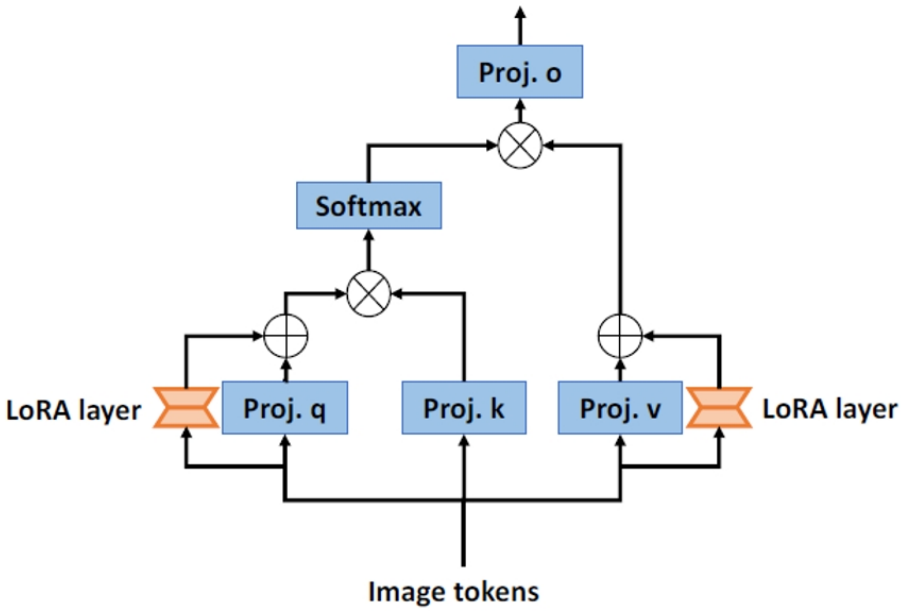



Figure 4.6: The LoRA layer in ViT transformer layer. The LoRA layer was applied to the $q$ (query) and $v$ (value) projection layer of each of the transformer block in the ViT (excluding $k$ (key) layer). "Proj.q", "Proj.k", "Proj.v", "Proj.o" represent the projection layer of $q, k, v$ and $o$, respectively. Plus means element wise add, and multiply means matrix production.

- **ViT-Huge:** Trainable parameters are reduced to 0.633 million from a total of 602.179 million.
- **ViT-Large:** Trainable parameters are reduced to 0.382 million from a total of 289.517 million.
- **ViT-Base:** Trainable parameters are reduced to 0.146 million from a total of 81.970 million.

The traditional methods (Figure **??** shown in red) make the basic model unmanageable due to the large number of trainable parameters. However, the LoRA-adapted model (depicted in blue) reduces the trainable parameters significantly. In comparison, the ResNet50 model has 22.431 million trainable parameters, which is much higher than any LoRA-ViT model. Without the use of LoRA, fine-tuning large ViT models would be impractical unless very small batch sizes were used, which could have a negative effect on the training dynamics and model performance. Therefore, the application of LoRA brings these benefits:

1. **Reduction of Trainable Parameters:** By significantly reducing the number of train-

able parameters, LoRA ensures that training large foundational models becomes feasible, even on high-performance computing (HPC) systems with resource constraints.

2. **Transferability Across Tasks:** LoRA's adaptability allows the transfer of weights of a foundation model between various tasks. This transferability ensures that only the lightweight LoRA layer needs to be stored for each specific task, rather than the entire model, leading to substantial storage savings.

3. **GPU and Training Efficiency:** LoRA reduces the necessary GPU resources for fine-tuning foundational models, which, coupled with the method's potential to speed up training, presents a cost-effective solution for model adaptation.

4. **Scalability with Limited Data:** LoRA is particularly advantageous in specialized fields such as digital pathology where extensive datasets may not be available. It enables effective fine-tuning on smaller datasets, which is often a limiting factor.

5. **Inference Efficiency:** The design of LoRA ensures that there is no additional inference latency, making it an attractive option for deploying models in real-world applications.

### 4.3.3 Data and model training

To assess the capability of transformer-based foundation models against the established ResNet50 benchmark, a cell classification task is conducted using a dataset consisting of 54,679 images. An illustration of the images in the data set can be found in Figure 4.7. These images are stratified across six unique cell type classes with the following distribution:

- Epithelial: 8,658 images
- Lymphocyte: 7,629 images
- Neutrophil: 2,991 images
- Plasma: 10,401 images
- Stroma: 10,000 images

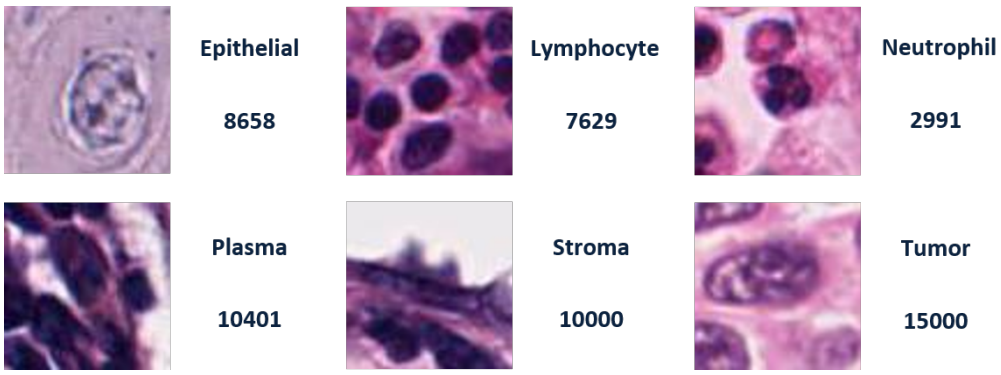


Figure 4.7: Sample and numbers of pictures for cell type classes: Epithelial, Lymphocyte, Neutrophil, Plasma, Stroma, Tumor.

- Tumor: 15,000 images

We used ResNet-50 (K. He *et al.*, 2016a) as our baseline model in the experiment. We observed that the performance was the same whether the model was pre-trained with ImageNet weights or trained from scratch. For the ViT models of different sizes, we start from the pre-trained weights obtained from Huggingface (Face, 2023), and apply the LoRA technique with different rank hyperparameters. The performance of the models is evaluated through three distinct experimental configurations. Each of these configurations has different train-validation-test splits, which are used to simulate various application scenarios.

1. **Experiment 1:** Comprises a standard distribution of 60% for training, 20% for validation, and 20% for testing.

2. **Experiment 2:** Utilizes a skewed split with 16% for training, 24% for validation, and a predominant 60% for testing, to assess model robustness with minimal training data.

3. **Experiment 3:** Employs an extreme case with approximately 2.7% (240 images) per class for training, a large 72% for validation, and 25% for testing, challenging models with a scarce training dataset.

In each experiment, the same data division is kept in order to guarantee that the outcomes from different models and training conditions can be compared. The training procedure involves updating the model weights with the training set and selecting the most effective model based on the validation set. In all of the training sets, data augmentations such

as random resizing and cropping, as well as random flipping, were applied. The ultimate measure of performance, the test accuracy, reflects the best model's capability as determined by the validation process.

We train all models using a single V100 GPU in a high-performance computing environment to ensure consistent computational conditions. After trail and error, the learning rate has been set to 0.001 for all models, and a cosine anneal learing rate schedule was applied. We pay particular attention to the LoRA method as it has the potential to reduce the number of trainable parameters, which could lead to more efficient and adaptable training of the model in data-limited situations. The results of these experiments will be essential in determining whether parameter reduction and approximation in the LoRA layer affects model accuracy or help to create a more effective and scalable training process.

## 4.4 Results

Here, we present the results of three experiments with different batch size and rank settings of LoRA-ViT compared to the baseline model ResNet 50 (which was pretrained on ImageNet and fine tuned on the train dataset). Tables 4.1, 4.2, and 4.3 display the results of these experiments.

In the first experiment, we observed that LoRA-ViT consistently outperformed the ResNet 50 across various configurations. Notably, the ViT base model achieved a higher accuracy with the same inferencing time as the ResNet model, and required significantly fewer epochs to reach comparable accuracy levels, indicating faster training speeds. For example, to reach 97. 5% accuracy in the validation set, the ViT base model with a batch size of 16 and rank 4 took 164 epochs and 7.3 hours, while with a batch size of 128 and the same rank, it took only 68 epochs and 3.4 hours, as shown in Table 1.

The second experiment further established the superiority of LoRA-ViT when training data was limited. With only 24% of the training data, the test accuracy gap widened to approximately 2%. The ViT base model with a batch size of 128 and rank 4 reached 96% accuracy in 74 epochs, taking 1.8 hours, which was more efficient compared to the baseline model, which could not reach 96% accuracy, as detailed in Table 2.

Experiment three demonstrated the robustness of LoRA-ViT under extreme data scarcity,

using only 2.7% of the training data. Despite a general performance decline, the test accuracy of the foundation model was 8% higher than that of the baseline model. The ViT base model reached a test accuracy of 93.4%, while the ResNet50 only achieved 85. 4%, highlighting the superior generalizability of the foundation model.

These results confirm the hypothesis that foundation models, due to their specialized pre-training, learn more effective vision representations than the ResNet50. Furthermore, LoRA effectively leverages the pretrained knowledge of the foundation model, requiring similar training effort as traditional fine-tuning methods for baseline models. Additional observations included that the training time for the ViT base model was on the same scale or faster than the baseline model. To achieve parity with ResNet50's performance, LoRA-ViT only needed about half an hour. The selection of hyperparameters such as rank and batch size proved crucial; a rank of 4 was sufficient for the ViT base model, while larger models required a reduction in rank due to GPU constraints. A minimum batch size of 64 was necessary to ensure convergence for models larger than the ViT base. These findings underscore the importance of optimizing training configurations to fully exploit the capabilities of LoRA-ViT.

| Model | Batch Size | Rank | Epoch to reach 97.5% Acc on validation | Time(hr) to reach 97.5% Acc on validation | Test Acc | Infer Time (s) |
|---|---|---|---|---|---|---|
| ViT – base | 16 | 4 | 164 | 7.3 | 97.8% | 34 |
| ViT – base | 128 | 4 | 68 | 3.4 | 97.7% | 34 |
| ViT – base | 32 | 8 | 90 | 4.5 | 97.9% | 34 |
| ViT – large | 32 | 4 | 82 | 8.7 | 98.0% | 48 |
| ViT - huge | 32 | 4 | Cannot reach. Need more GPU resource | | 96.8% | 86 |
| ResNet50 | 32 | N/A | Cannot reach 97.5%. | | 96.7% | 37 |

Table 4.1: The performance of Vision Transformers (ViT) with Low-Rank Adaptation (LoRA) and a ResNet50 model across various configurations in experiment 1. Test Acc means accuracy on the test set of the best model. Infer Time (s) means for one batch of data, how many seconds the model will spend on inference stage.

## 4.5 Conclusion

The conclusion of this research demonstrates the efficacy of the Low-Rank Adaptation (LoRA) technique in fine-tuning large-scale transformer-based vision models. Through three experimental conditions, LoRA-ViT models consistently outperformed the baseline ResNet 50 model, particularly in scenarios with limited training data. The reduction in epochs and training time required to reach high accuracy levels underlines the efficiency of LoRA in

| Model | Batch Size | Rank | Epoch to reach 96% Acc on validation | Time(hr) to reach 96% Acc on validation | Test Acc | Infer Time (s) |
|---|---|---|---|---|---|---|
| ViT – base | 32 | 4 | 139 | 3.5 | 96.4% | 44 |
| ViT – base | 128 | 4 | 74 | 1.8 | 96.4% | 44 |
| ViT – base | 128 | 8 | 113 | 2.7 | 96.6% | 45 |
| ViT – large | 64 | 4 | 135 | 6 | 96.5% | 60 |
| ViT – huge | 32 | 4 | Model cannot converge. Need more GPU resource (on one V100) | | | |
| ResNet50 | 32 | N/A | Cannot reach 96%. | | 94.5% | 48 |

Table 4.2: The performance of Vision Transformers (ViT) with Low-Rank Adaptation (LoRA) and a ResNet50 model across various configurations in experiment 2. Test Acc means accuracy on the test set of the best model. Infer Time (s) means for one batch of data, how many seconds the model will spend on inference stage.

| Model | Batch Size | Rank | Max Train Acc | Max Val Acc | Test accuracy |
|---|---|---|---|---|---|
| ViT – base | 64 | 4 | 98.3% | 92.3% | 93.4% |
| ViT – large | 64 | 4 | 98.1% | 92.5% | 93.5% |
| ViT – huge | 64 | 2 | 98.4% | 92.3% | 93.4% |
| ResNet50 | 64 | N/A | 97.2% | 83.6% | 85.4% |

Table 4.3: The performance of Vision Transformers (ViT) with Low-Rank Adaptation (LoRA) and a ResNet50 model across various configurations in experiment 3.

leveraging pre-trained foundational models. Crucially, the LoRA technique's ability to maintain or even enhance model accuracy while dramatically decreasing the number of trainable parameters addresses significant computational and storage challenges, offering a scalable and resource-efficient solution for deploying advanced vision models. Our findings advocate for the adaptation of LoRA in future vision tasks, especially in fields constrained by data scarcity or computational resources, promising a new horizon in the practical application of state-of-the-art vision models.

# Chapter 5

# Conclusion and Discussion

Deep learning has become a crucial asset in the biomedical field and has revolutionized our prediction capabilities in problems arising in biological systems and disease modeling. Despite these advances, the primary challenge of designing and training deep learning models that can effectively manage the complex and diverse nature of biomedical data is barely starting to be explored. The aim is not just to use these models for analyzing large and complex datasets but also to ensure they are capable of interpreting biological variability. Overcoming issues such as data heterogeneity, the necessity for interpretable models, and computational limitations is essential in harnessing the full potential of deep learning in this realm. A complete integration of deep learning into the medical field will allow us to enhance the precision and effectiveness of medical treatments and deepen our basic understanding of biological systems.

In my PhD thesis, I have explored the role of deep learning in biology by addressing some very challenging and complex questions. Through the development and implementation of advanced deep learning models, I have sought to use the growing volume of biomedical data to obtain new insights. This thesis navigates the integration of deep learning into life sciences, carefully strategizing to meet specific domain challenges while optimizing the analysis of highly heterogeneous datasets.

In Chapter 2, we delve into the development of Scratch-AID, a deep learning-based system engineered to automate the detection of mouse scratching behaviors, specifically following the administration of chloroquine. This innovation emerges as a response to the

limitations inherent in manual video analysis, such as labor intensity and susceptibility to human error. At its core, Scratch-AID employs a convolution recurrent neural network, meticulously trained on a diverse dataset of video recordings, annotating mouse scratching behaviors induced by chloroquine. This training encompasses a range of environmental conditions to ensure robustness and accuracy. The system's efficacy is highlighted by its impressive recall rate of 97.6% and a precision rate of 96.9%, signifying its potential as a reliable alternative to traditional manual methods in both pharmacological screenings and behavioral studies. This case study not only underscores the potential of deep learning in enhancing the accuracy of behavioral analysis but also highlights the challenges in ensuring comprehensive training and continuous model refinement. Our research demonstrates the methodology for designing a robust deep learning model to meet the intricate requirements of biological applications.

In Chapter 3, we delve deeper into the critical issue of cancer patient survival, presenting the Genomic-Guided Hierarchical Attention Multi-Scale Pathology Transformer (GG-HAMPT). This framework, motivated by the urgent need to improve survival prediction accuracy in oncology, builds upon and extends the foundations laid by the Multimodal Co-Attention Transformer (MCAT)(R. J. Chen *et al.,* 2021). Our contribution lies in the novel multi pathology level early data fusion strategy that GG-HAMPT employs. Leveraging the co-attention mechanism, a concept originally from the domain of Visual Question Answering (VQA), the model innovatively integrates genomic data with three-level representations of Whole Slide Images (WSIs) of cancer tissues. This integration, which is a significant advancement from the MCAT approach, allows for a more dynamic and nuanced analysis of the pathological data in relation to the genomic information of patients. The GG-HAMPT's ability to weigh pathology information with respect to corresponding genomic data significantly enhances its predictive capacity for patient survival outcomes, using multimodal data including gigapixel WSIs. The model notably surpasses other existing weakly supervised methods in performance. Looking forward, this pioneering approach in GG-HAMPT paves the way for more sophisticated, data-integrated models in the field of oncology, potentially transforming survival prediction and can help discover new genes that may be overlook from previous study.

In the final section of my research, I ventured into exploring the domain adaptation of pre-trained models in digital pathology image classification, particularly focusing on the development of the low rank adapatation of Vision Transformer (LoRA-ViT). The motivation behind this initiative stemmed from the hypothesis that large-scale models possess the capability to learn more robust feature representations in computer vision, potentially offering greater robustness in diverse applications. However, these large models, due to their extensive parameter sets, pose challenges in fine-tuning, requiring significant computational power and time for training. Our objective was to investigate methodologies to efficiently and effectively train large models in digital pathology tasks, aiming to outperform traditional models like ResNet. The LoRA-ViT demonstrated exceptional efficiency and accuracy, notably surpassing the traditional ResNet 50 model in a 6-class cell classification task. Conclusively, the LoRA-ViT proved an impressive adaptability to digital pathology data. This adaptability is achieved without the necessity for extensive retraining or specific dataset tuning, effectively mitigating the domain shift issue. Such adaptability and speed position LoRA-ViT as a substantial advancement in automated pathology, supporting the broader integration of artificial intelligence in diagnostic medicine and potentially revolutionizing the field.

Domain adaptation in deep learning means applying a model trained on one set of data (source domain) to a different, but related set (target domain). The core of domain adaptation is the ability to transfer knowledge from one domain to another, addressing the challenge of data heterogeneity and scarcity, which are common in biomedical applications. This is particularly pertinent in areas such as medical imaging, genomics, and drug discovery, where models trained on large datasets (like public repositories) must be adapted to work effectively on specific, often smaller datasets from hospitals or labs. The success of domain adaptation hinges on the model's ability to generalize well from the source domain to the target domain, capturing underlying patterns and features that are invariant across domains. This involves techniques like feature alignment, transfer learning, and fine-tuning, ensuring that the model, initially trained on a broad dataset, remains accurate and reliable when applied to more specialized, domain-specific data.

In my research, I have undertaken three projects that each contribute to this field. The first project involved training deep learning models from scratch in the experimental animal

domain, which is pivotal in understanding the development of models tailored to unique datasets. The second project centered on designing complex, multi-scale models for cancer multimodal data, including imagery, highlighting the necessity of developing specialized models to manage the complexities of multimodal biomedical data. The third project focused on the domain adaptation method LoRA for pre-trained large models. This method is essential for adapting extensive models to specific biomedical data, thereby enhancing the performance of deep learning in the biomedical area. Collectively, these studies significantly contribute to the advancement of deep learning in the biomedical sector, particularly in the customization of deep learning models for specific biomedical data and the utilization of appropriate methodologies.

# Bibliography


1. Abadi, M. *et al. TensorFlow: a system for Large-Scale machine learning* in *12th USENIX symposium on operating systems design and implementation (OSDI 16)* (), 265–283. ISBN: 1931971331.
2. Abels, E. *et al.* Computational pathology definitions, best practices, and recommendations for regulatory guidance: a white paper from the Digital Pathology Association. *The Journal of pathology* **249,** 286–294 (2019).
3. Acs, B., Rantalainen, M. & Hartman, J. Artificial intelligence as the next step towards precision pathology. *Journal of internal medicine* **288,** 62–81 (2020).
4. Aeffner, F. *et al.* Introduction to digital image analysis in whole-slide imaging: a white paper from the digital pathology association. *Journal of pathology informatics* **10,** 9 (2019).
5. Aghajanyan, A., Zettlemoyer, L. & Gupta, S. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255* (2020).
6. Ahmad, F. S., Luo, Y., Wehbe, R. M., Thomas, J. D. & Shah, S. J. Advances in machine learning approaches to heart failure with preserved ejection fraction. *Heart Failure Clinics* **18,** 287–300 (2022).
7. Alipanahi, B., Delong, A., Weirauch, M. T. & Frey, B. J. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature biotechnology* **33,** 831–838 (2015).
8. Antolini, L., Boracchi, P. & Biganzoli, E. A time-dependent discrimination index for survival data. *Statistics in medicine* **24,** 3927–3944 (2005).
9. Antoniou, A. C. *et al.* Breast-cancer risk in families with mutations in PALB2. *New England Journal of Medicine* **371,** 497–506 (2014).
10. Aronszajn, N. Theory of reproducing kernels. *Transactions of the American mathematical society* **68,** 337–404 (1950).

11. Arora, A. *et al.* Transcriptomic and protein expression analysis reveals clinicopathological significance of bloom syndrome helicase (BLM) in breast cancer. *Molecular cancer therapeutics* **14,** 1057–1065 (2015).

12. Baltrusaitis, T., Ahuja, C. & Morency, L. P. Multimodal machine learning: A survey and taxonomy. *IEEE T. Pattern Anal.* **41.** https://doi.org/10.1109/TPAMI.2018.2798607 (2019).

13. Beattie, K. *et al.* TRPC3 Antagonizes Pruritus in a Mouse Contact Dermatitis Model. *Journal of Investigative Dermatology* **142,** 1136–1144. ISSN: 0022-202X (2022).

14. Bengio, Y., Simard, P. & Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* **5,** 157–166 (1994).

15. Bera, K., Schalper, K. A., Rimm, D. L., Velcheti, V. & Madabhushi, A. Artificial intelligence in digital pathology—new tools for diagnosis and precision oncology. *Nature reviews Clinical oncology* **16,** 703–715 (2019).

16. Bishop, C. M. Training with noise is equivalent to Tikhonov regularization. *Neural computation* **7,** 108–116 (1995).

17. Blasch, E. *et al.* Machine learning/artificial intelligence for sensor data fusion–opportunities and challenges. *IEEE Aerospace and Electronic Systems Magazine* **36,** 80–93 (2021).

18. Bohnslav, J. P. *et al.* DeepEthogram, a machine learning pipeline for supervised behavior classification from raw pixels. *Elife* **10.** ISSN: 2050-084X (Electronic) 2050-084X (Linking). https://www.ncbi.nlm.nih.gov/pubmed/34473051 (2021).

19. Bradski, G. The openCV library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer* **25,** 120–123. ISSN: 1044-789X (2000).

20. Bray, F. Global cancer statistics 2018: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA-Cancer J. Clin.* **68.** https://doi.org/10.3322/caac.21492 (2018).

21. Brown, T. *et al.* Language models are few-shot learners. *Advances in neural information processing systems* **33,** 1877–1901 (2020).

22. Campanella, G. *et al.* Clinical-grade computational pathology using weakly supervised deep learning on whole slide images. *Nature medicine* **25,** 1301–1309 (2019).

23. Campbell, F. W. & Kulikowski, J. J. Orientational selectivity of the human visual system. *The Journal of physiology* **187,** 437–445 (1966).

24. Castanedo, F. *et al.* A review of data fusion techniques. *The scientific world journal* **2013** (2013).

25. Cevikbas, F. & Lerner, E. A. Physiology and pathophysiology of itch. *Physiological reviews* **100,** 945–982. ISSN: 0031-9333 (2020).

26. Chang, J.-W. *et al.* Wild-type p53 upregulates an early onset breast cancer-associated gene GAS7 to suppress metastasis via GAS7–CYFIP1-mediated signaling pathway. *Oncogene* **37,** 4137–4150 (2018).

27. Chaudhary, K., Poirion, O. B., Lu, L. & Garmire, L. X. Deep Learning–Based Multi-Omics Integration Robustly Predicts Survival in Liver CancerUsing Deep Learning to Predict Liver Cancer Prognosis. *Clinical Cancer Research* **24,** 1248–1259 (2018).

28. Cheerla, A. & Gevaert, O. Deep learning with multimodal representation for pan-cancer prognosis prediction. *Bioinformatics* **35.** https://doi.org/10.1093/bioinformatics/btz342 (2019).

29. Chen, H., Qi, X., Yu, L. & Heng, P.-A. *DCAN: deep contour-aware networks for accurate gland segmentation* in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2016), 2487–2496.

30. Chen, H., Cong, X., *et al.* Intratumoral delivery of CCL25 enhances immunotherapy against triple-negative breast cancer by recruiting CCR9+ T cells. *Science advances* **6,** eaax4690 (2020).

31. Chen, L. *et al.* Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems* **34,** 15084–15097 (2021).

32. Chen, R. J. *et al. Multimodal Co-Attention Transformer for Survival Prediction in Gigapixel Whole Slide Images* in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (Oct. 2021), 4015–4025.

33. Cheon, S. The accuracy of clinicians predictions of survival in advanced cancer: A review. *Ann. Palliat. Med.* **5.** https://doi.org/10.3978/j.issn.2224-5820.2015.08.04 (2016).

34. Chetlur, S. *et al.* cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759* (2014).

35. Chew, N. J. *et al.* FGFR3 signaling and function in triple negative breast cancer. *Cell Communication and Signaling* **18,** 1–17 (2020).

36. Chikontwe, P., Kim, M., Nam, S. J., Go, H. & Park, S. H. *Multiple instance learning with center embeddings for histopathology classification* in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part V 23* (2020), 519–528.

37. Ching, T. *et al.* Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface* **15,** 20170387 (2018).

38. Cho, K., Van Merriënboer, B., Bahdanau, D. & Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

39. Colubri, A. *et al.* Machine-learning prognostic models from the 2014–16 Ebola outbreak: data-harmonization challenges, validation strategies, and mHealth applications. *EClinicalMedicine* **11,** 54–64 (2019).

40. Cox, D. R. Regression models and life-tables. *J. R. Stat. Soc.* **34.** https://doi.org/10.1111/j.2517-6161.1972.tb00899.x (1972).

41. Cui, C. *et al.* Deep multi-modal fusion of image and non-image data in disease diagnosis and prognosis: a review. *Progress in Biomedical Engineering* (2023).

42. Darmani, N. & Pandya, D. Involvement of other neurotransmitters in behaviors induced by the cannabinoid CB1 receptor antagonist SR 141716A in naive mice. *Journal of neural transmission* **107,** 931–945. ISSN: 1435-1463 (2000).

43. De Bel, T., Hermsen, M., Kers, J., van der Laak, J. & Litjens, G. Stain-transforming cycle-consistent generative adversarial networks for improved segmentation of renal histopathology (2018).

44. Deng, J. *et al. Imagenet: A large-scale hierarchical image database* in *2009 IEEE conference on computer vision and pattern recognition* (2009), 248–255.

45. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

46. Dhungel, N., Carneiro, G. & Bradley, A. P. A deep learning approach for the analysis of masses in mammograms with minimal user intervention. *Medical image analysis* **37,** 114–128 (2017).

47. Dimitriou, N., Arandjelović, O. & Caie, P. D. Deep learning for whole slide image analysis: an overview. *Frontiers in medicine* **6,** 264 (2019).

48. Doi, A. *et al.* Enhanced expression of retinoic acid receptor alpha (RARA) induces epithelial-to-mesenchymal transition and disruption of mammary acinar structures. *Molecular oncology* **9,** 355–364 (2015).

49. Dosovitskiy, A. *et al.* An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

50. Elliott, P., G'Sell, M., Snyder, L. M., Ross, S. E. & Ventura, V. Automated acoustic detection of mouse scratching. *PLoS One* **12,** e0179662. ISSN: 1932-6203 (Electronic) 1932-6203 (Linking). https://www.ncbi.nlm.nih.gov/pubmed/28678797 (2017).

51. Elman, J. L. Finding structure in time. *Cognitive science* **14,** 179–211 (1990).

52. Erhan, D., Bengio, Y., Courville, A. & Vincent, P. Visualizing higher-layer features of a deep network. *University of Montreal* **1341,** 1 (2009).

53. Eser, U. & Churchman, L. S. FIDDLE: An integrative deep learning framework for functional genomic data inference. *Biorxiv,* 081380 (2016).

54. Esteva, A. A guide to deep learning in healthcare. *Nat. Med.* **25.** https://doi.org/10.1038/s41591-018-0316-z (2019).

55. Face, H. *The AI community building the future* Accessed: 2023-11-06. 2023. https://huggingface.co/.

56. Faraggi, D. & Simon, R. A neural network model for survival data. *Stat. Med.* **14.** https://doi.org/10.1002/sim.4780140108 (1995).

57. Farahani, N., Parwani, A. V. & Pantanowitz, L. Whole slide imaging in pathology: advantages, limitations, and emerging perspectives. *Pathology and Laboratory Medicine International,* 23–33 (2015).

58. Fotso, S. Deep neural networks for survival analysis based on a multi-task framework. *arXiv preprint arXiv:1801.05512* (2018).

59. Frank, I. *et al.* An outcome prediction model for patients with clear cell renal cell carcinoma treated with radical nephrectomy based on tumor stage, size, grade and necrosis: the SSIGN score. *The Journal of urology* **168,** 2395–2400 (2002).

60. Gampenrieder, S. *et al.* Endothelin-1 genetic polymorphism as predictive marker for bevacizumab in metastatic breast cancer. *The Pharmacogenomics Journal* **17,** 344–350 (2017).

61. Giunchiglia, E., Nemchenko, A. & van der Schaar, M. *Rnn-surv: A deep recurrent model for survival analysis* in *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27* (2018), 23–32.

62. Grant, S. W., Hickey, G. L. & Head, S. J. Statistical primer: Multivariable regression considerations and pitfalls. *Eur. J. Cardio-Thorac.* **55.** https://doi.org/10.1093/ejcts/ezy403 (2019).

63. Graves, A. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).

64. Gu, J. *et al.* Recent advances in convolutional neural networks. *Pattern recognition* **77,** 354–377. ISSN: 0031-3203 (2018).

65. Gulati, A. *et al.* Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100* (2020).

66. Gulli, A. & Pal, S. *Deep learning with Keras* ISBN: 1787129039 (Packt Publishing Ltd, 2017).

67. Guo, L. *et al.* A high-risk luminal A dominant breast cancer subtype with increased mobility. *Breast Cancer Research and Treatment* **175,** 459–472 (2019).

68. Guo, W., Wang, J. & Wang, S. Deep multimodal representation learning: A survey. *IEEE Access* **7.** https://doi.org/10.1109/ACCESS.2019.2916887 (2019).

69. Han, L. *et al.* A subpopulation of nociceptors specifically linked to itch. *Nat Neurosci* **16,** 174–82. ISSN: 1546-1726 (Electronic) 1097-6256 (Linking). https://www.ncbi.nlm.nih.gov/pubmed/23263443 (2013).

70. He, K., Zhang, X., Ren, S. & Sun, J. *Deep residual learning for image recognition* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 770–778.

71. He, K., Zhang, X., Ren, S. & Sun, J. *Deep residual learning for image recognition* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 770–778.

72. He, K., Zhang, X., Ren, S. & Sun, J. *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification* in *Proceedings of the IEEE international conference on computer vision* (2015), 1026–1034.

73. Hong, J., Buddenkotte, J., Berger, T. G. & Steinhoff, M. *Management of itch in atopic dermatitis* in *Seminars in cutaneous medicine and surgery* **30** (NIH Public Access, 2011), 71.

74. Hosseini, M. S. *et al. Atlas of digital pathology: A generalized hierarchical histological tissue type-annotated database for deep learning* in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), 11747–11756.

75. Hou, L. *et al. Patch-based convolutional neural network for whole slide tissue image classification* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), 2424–2433.

76. Hu, E. J. *et al.* Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

77. Huang, S.-C., Pareek, A., Zamanian, R., Banerjee, I. & Lungren, M. P. Multimodal fusion with deep neural networks for leveraging CT imaging and electronic health record: a case-study in pulmonary embolism detection. *Scientific reports* **10,** 22147 (2020).

78. Hui, D. Prognostication in advanced cancer: Update and directions for future research. *Support. Care Cancer* **27.** https://doi.org/10.1007/s00520-019-04727-y (2019).

79. Iglovikov, V. I., Rakhlin, A., Kalinin, A. A. & Shvets, A. A. in *Deep learning in medical image analysis and multimodal learning for clinical decision support* 300–308 (Springer, 2018).

80. Ikoma, A., Steinhoff, M., Ständer, S., Yosipovitch, G. & Schmelz, M. The neurobiology of itch. *Nature reviews neuroscience* **7,** 535–547. ISSN: 1471-0048 (2006).

81. Ilse, M., Tomczak, J. & Welling, M. *Attention-based deep multiple instance learning* in *International conference on machine learning* (2018), 2127–2136.

82. Ioffe, S. & Szegedy, C. *Batch normalization: Accelerating deep network training by reducing internal covariate shift* in *International conference on machine learning* (2015), 448–456.

83. Jahn, S. W., Plass, M. & Moinfar, F. Digital pathology: advantages, limitations and emerging perspectives. *Journal of clinical medicine* **9,** 3697 (2020).

84. Jansson, S. *et al.* The PDGF pathway in breast cancer is linked to tumour aggressiveness, triple-negative subtype and early recurrence. *Breast cancer research and treatment* **169,** 231–241 (2018).

85. Jha, A., Gazzara, M. R. & Barash, Y. Integrative deep models for alternative splicing. *Bioinformatics* **33,** i274–i282 (2017).

86. Al-Juboori, S. I. *et al.* PYK2 promotes HER2-positive breast cancer invasion. *Journal of Experimental & Clinical Cancer Research* **38,** 1–14 (2019).

87. Jumper, J. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* **596,** 583–589 (2021).

88. Katzman, J. L. DeepSurv: Personalized treatment recommender system using a Cox proportional hazards deep neural network. *BMC Med. Res. Methodol.* https://doi.org/10.1186/s12874-018-0482-1 (2018).

89. Kelley, D. R., Snoek, J. & Rinn, J. L. Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome research* **26,** 990–999 (2016).

90. Kim, S., Kim, D. H., Jung, W.-H. & Koo, J. S. Succinate dehydrogenase expression in breast cancer. *Springerplus* **2,** 1–12 (2013).

91. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

92. Kinreich, S. *et al.* Predicting risk for Alcohol Use Disorder using longitudinal data with multimodal biomarkers and family history: a machine learning study. *Molecular psychiatry* **26,** 1133–1141 (2021).

93. Kirby, J. C. *et al.* PheKB: a catalog and workflow for creating electronic phenotype algorithms for transportability. *Journal of the American Medical Informatics Association* **23,** 1046–1052 (2016).

94. Kirillov, A. *et al.* Segment anything. *arXiv preprint arXiv:2304.02643* (2023).

95. Klauschen, F. *et al. Scoring of tumor-infiltrating lymphocytes: From visual estimation to machine learning* in *Seminars in cancer biology* **52** (2018), 151–157.

96. Kline, A. *et al.* Multimodal machine learning in precision health: A scoping review. *npj Digital Medicine* **5,** 171 (2022).

97. Kobayashi, K. *et al.* Automated detection of mouse scratching behaviour using convolutional recurrent neural network. *Sci Rep* **11,** 658. ISSN: 2045-2322 (Electronic) 2045-2322 (Linking). https://www.ncbi.nlm.nih.gov/pubmed/33436724 (2021).

98. Kohane, I. S. *et al.* What every reader should know about studies using electronic health record data but may be afraid to ask. *Journal of medical Internet research* **23,** e22219 (2021).

99. Koo, P. K. & Eddy, S. R. Representation learning of genomic sequence motifs with convolutional neural networks. *PLoS computational biology* **15,** e1007560 (2019).

100. Koo, P. K., Qian, S., Kaplun, G., Volf, V. & Kalimeris, D. Robust neural networks are more interpretable for genomics. *bioRxiv,* 657437 (2019).

101. Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V. & Fotiadis, D. I. Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal* **13,** 8–17 (2015).

102. Koutsouleris, N. *et al.* Multimodal machine learning workflows for prediction of psychosis in patients with clinical high-risk syndromes and recent-onset depression. *JAMA psychiatry* **78,** 195–209 (2021).

103. Koyuncu, D., Sharma, U., Goka, E. T. & Lippman, M. E. Spindle assembly checkpoint gene BUB1B is essential in breast cancer cell survival. *Breast cancer research and treatment* **185,** 331–341 (2021).

104. Kremer, A. E., Mettang, T. & Weisshaar, E. Non-dermatological challenges of chronic itch. *Acta Dermato-Venereologica* **100,** 21–26. ISSN: 1651-2057 (2020).

105. Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* **25** (2012).

106. Krogh, A. & Hertz, J. A simple weight decay can improve generalization. *Advances in neural information processing systems* **4** (1991).

107. Kvamme, H., Borgan, O. & Scheel, I. Time-to-event prediction with neural networks and Cox regression. *J. Mach. Learn. Res.* **20** (2019).

108. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521.** https://doi.org/10.1038/nature14539 (2015).

109. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *nature* **521,** 436–444 (2015).

110. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86,** 2278–2324 (1998).

111. LeCun, Y., Jackel, L., *et al. Comparison of learning algorithms for handwritten digit recognition* in *International conference on artificial neural networks* **60** (1995), 53–60.

112. Lee, C., Yoon, J. & Schaar, M. Dynamic-DeepHit: A deep learning approach for dynamic survival analysis with competing risks based on longitudinal data. *IEEE T. Bio-Med. Eng.* https://doi.org/10.1109/TBME.2019.2909027 (2019).

113. Lee, C., Zame, W., Yoon, J. & Van Der Schaar, M. *Deephit: A deep learning approach to survival analysis with competing risks* in *Proceedings of the AAAI conference on artificial intelligence* **32** (2018).

114. Li, M. *et al.* RNA-binding protein MSI2 isoforms expression and regulation in progression of triple-negative breast cancer. *Journal of Experimental & Clinical Cancer Research* **39,** 1–18 (2020).

115. Li, S., Shi, H., Sui, D., Hao, A. & Qin, H. *A Novel Pathological Images and Genomic Data Fusion Framework for Breast Cancer Survival Prediction* in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)* (2020), 1384–1387.

116. Liang, M., Li, Z., Chen, T. & Zeng, J. Integrative data analysis of multi-platform cancer data with a multimodal deep learning approach. *IEEE/ACM transactions on computational biology and bioinformatics* **12,** 928–937 (2014).

117. Liang, S. *et al.* CDK12: a potent target and biomarker for human cancer therapy. *Cells* **9,** 1483 (2020).

118. Liberzon, A. *et al.* The molecular signatures database hallmark gene set collection. *Cell systems* **1,** 417–425 (2015).

119. Lin, M., Chen, Q. & Yan, S. Network in network. *arXiv preprint arXiv:1312.4400* (2013).

120. Lin, P. *et al.* A radiogenomics signature for predicting the clinical outcome of bladder urothelial carcinoma. *European Radiology* **30,** 547–557 (2020).

121. Liu, Q., Tang, Z., *et al.* Sensory neuron-specific GPCR Mrgprs are itch receptors mediating chloroquine-induced pruritus. *Cell* **139,** 1353–65. ISSN: 1097-4172 (Electronic) 0092-8674 (Linking). https://www.ncbi.nlm.nih.gov/pubmed/20004959 (2009).

122. Liu, Q. & Dong, X. The role of the Mrgpr receptor family in itch. *Pharmacology of itch,* 71–88 (2015).

123. Liu, Y. *et al.* Artificial intelligence–based breast cancer nodal metastasis detection: Insights into the black box for pathologists. *Archives of pathology & laboratory medicine* **143,** 859–868 (2019).

124. Liu, Z. *et al. Swin transformer: Hierarchical vision transformer using shifted windows* in *Proceedings of the IEEE/CVF international conference on computer vision* (2021), 10012–10022.

125. Loew, E. R., MacMILLAN, R. & Kaiser, M. E. The anti-histamine properties of Benadryl, -dimethylaminoethyl benzhydryl ether hydrochloride. *Journal of Pharmacology and Experimental Therapeutics* **86,** 229–238. ISSN: 0022-3565 (1946).

126. Lu, J., Yang, J., Batra, D. & Parikh, D. Hierarchical question-image co-attention for visual question answering. *Advances in neural information processing systems* **29** (2016).

127. Lu, S., Simin, K., Khan, A. & Mercurio, A. M. Analysis of integrin $\beta 4$ expression in human breast cancer: association with basal-like tumors and prognostic significance. *Clinical cancer research* **14,** 1050–1058 (2008).

128. Lundberg, S. & Lee, S.-I. An unexpected unity among methods for interpreting model predictions. *arXiv preprint arXiv:1611.07478* (2016).

129. Luo, Y., Ahmad, F. S. & Shah, S. J. Tensor factorization for precision medicine in heart failure with preserved ejection fraction. *Journal of cardiovascular translational research* **10,** 305–312 (2017).

130. Mariotto, A. B. Cancer survival: An overview of measures, uses, and interpretation. *JNCI Monogr.* **145–186.** https://doi.org/10.1093/jncimonographs/lgu024 (2014).

131. Marvin, M. & Seymour, A. P. Perceptrons. *Cambridge, MA: MIT Press* **6,** 318–362 (1969).

132. Mathelier, A. *et al.* JASPAR 2016: a major expansion and update of the open-access database of transcription factor binding profiles. *Nucleic acids research* **44,** D110–D115 (2016).

133. Mathis, A. *et al.* DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nat Neurosci* **21,** 1281–1289. ISSN: 1546-1726 (Electronic) 1097-6256 (Linking). https://www.ncbi.nlm.nih.gov/pubmed/30127430 (2018).

134. Matterne, U., Strassner, T., Apfelbacher, C. J., Diepgen, T. L. & Weisshaar, E. Measuring the prevalence of chronic itch in the general population: development and validation of a questionnaire for use in large-scale studies. *Acta Derm Venereol* **89,** 250–6. ISSN: 0001-5555 (Print) 0001-5555 (Linking). https://www.ncbi.nlm.nih.gov/pubmed/19479120 (2009).

135. Mazurowski, M. A. *et al.* Segment anything model for medical image analysis: an experimental study. *Medical Image Analysis* **89,** 102918 (2023).

136. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* **5,** 115–133 (1943).

137. Min, X., Chen, N., Chen, T. & Jiang, R. *DeepEnhancer: predicting enhancers by convolutional neural networks* in *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (2016), 637–644.

138. Mo, Y. J., Kim, J., Kim, J.-K., Mohaisen, A. & Lee, W. *Performance of deep learning computation with TensorFlow software library in GPU-capable multi-core computing platforms* in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)* (2017), 240–242.

139. Mobadersany, P. *et al.* Predicting cancer outcomes from histology and genomics using convolutional networks. *Proceedings of the National Academy of Sciences* **115,** E2970–E2979 (2018).

140. Moon, H.-G. *et al.* NFIB is a potential target for estrogen receptor-negative breast cancers. *Molecular oncology* **5,** 538–544 (2011).

141. Morita, T. *et al.* HTR7 Mediates Serotonergic Acute and Chronic Itch. *Neuron* **87,** 124–38. ISSN: 1097-4199 (Electronic) 0896-6273 (Linking). https://www.ncbi.nlm.nih.gov/pubmed/26074006 (2015).

142. Mu, D. *et al.* A central neural circuit for itch sensation. *Science* **357,** 695–699. ISSN: 1095-9203 (Electronic) 0036-8075 (Linking). https://www.ncbi.nlm.nih.gov/pubmed/28818946 (2017).

143. Nair, V. & Hinton, G. E. *Rectified linear units improve restricted boltzmann machines* in *Proceedings of the 27th international conference on machine learning (ICML-10)* (2010), 807–814.

144. Nakanishi, C. *et al.* Germline mutation of the LKB1/STK11 gene with loss of the normal allele in an aggressive breast cancer of Peutz-Jeghers syndrome. *Oncology* **67,** 476–479 (2005).

145. Ngiam, J. *et al. Multimodal deep learning* in *ICML* (2011).

146. Niazi, M. K. K., Parwani, A. V. & Gurcan, M. N. Digital pathology and artificial intelligence. *The lancet oncology* **20,** e253–e261 (2019).

147. Nie, D., Zhang, H., Adeli, E., Liu, L. & Shen, D. *3D deep learning for multi-modal imaging-guided survival time prediction of brain tumor patients* in *International conference on medical image computing and computer-assisted intervention* (2016), 212–220.

148. Norgeot, B., Glicksberg, B. S. & Butte, A. J. A call for deep-learning healthcare. *Nat. Med.* **25.** https://doi.org/10.1038/s41591-018-0320-3 (2019).

149. Novakovsky, G., Dexter, N., Libbrecht, M. W., Wasserman, W. W. & Mostafavi, S. Obtaining genetics insights from deep learning via explainable artificial intelligence. *Nature Reviews Genetics* **24,** 125–137 (2023).

150. OpenAI. GPT-4 Technical Report. *ArXiv* **abs/2303.08774.** https://api.semanticscholar.org/CorpusID:257532815 (2023).

151. Pan, J.-K. *et al.* ICAM2 initiates trans-blood-CSF barrier migration and stemness properties in leptomeningeal metastasis of triple-negative breast cancer. *Oncogene,* 1–13 (2023).

152. Paratala, B. S. *et al.* RET rearrangements are actionable alterations in breast cancer. *Nature communications* **9,** 4821 (2018).

153. Park, I. *et al.* Machine-Learning Based Automatic and Real-time Detection of Mouse Scratching Behaviors. *Exp Neurobiol* **28,** 54–61. ISSN: 1226-2560 (Print) 1226-2560 (Linking). https://www.ncbi.nlm.nih.gov/pubmed/30853824 (2019).

154. Pascual, J. & Turner, N. Targeting the PI3-kinase pathway in triple-negative breast cancer. *Annals of Oncology* **30,** 1051–1060 (2019).

155. Paszke, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* **32** (2019).

156. Peng, Y. *et al.* NegBio: a high-performance tool for negation and uncertainty detection in radiology reports. *AMIA Summits on Translational Science Proceedings* **2018,** 188 (2018).

157. Piccolella, M. *et al.* The small heat shock protein B8 (HSPB8) modulates proliferation and migration of breast cancer cells. *Oncotarget* **8,** 10400 (2017).

158. Powers, D. M. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061* (2020).

159. Qi, Z., Khorram, S. & Li, F. *Visualizing Deep Networks by Optimizing with Integrated Gradients.* in *CVPR Workshops* **2** (2019), 1–4.

160. Qu, L., Fu, K., Yang, J., Shimada, S. G. & LaMotte, R. H. CXCR3 chemokine receptor signaling mediates itch in experimental allergic contact dermatitis. *Pain* **156,** 1737 (2015).

161. Quang, D. & Xie, X. DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic acids research* **44,** e107–e107 (2016).

162. Rakhlin, A., Shvets, A., Iglovikov, V. & Kalinin, A. A. *Deep convolutional neural networks for breast cancer histology image analysis* in *international conference image analysis and recognition* (2018), 737–744.

163. Rasmussen, L. V., Brandt, P. S., *et al. Considerations for improving the portability of electronic health record-based phenotype algorithms* in *AMIA Annual Symposium Proceedings* **2019** (2019), 755.

164. Rasmussen, L. V., Hoell, C., *et al.* Solutions for unexpected challenges encountered when integrating research genomics results into the EHR. *ACI Open* **4,** e132–e135 (2020).

165. Razzak, M. I., Naz, S. & Zaib, A. Deep learning for medical image processing: Overview, challenges and the future. *Classification in BioApps: Automation of Decision Making,* 323–350 (2018).

166. Ren, K. *et al. Deep recurrent survival analysis* in *Proceedings of the AAAI Conference on Artificial Intelligence* **33** (2019), 4798–4805.

167. Rennert, G. *et al.* Clinical outcomes of breast cancer in carriers of BRCA1 and BRCA2 mutations. *New England Journal of Medicine* **357,** 115–123 (2007).

168. Ribeiro, M. T., Singh, S. & Guestrin, C. *" Why should i trust you?" Explaining the predictions of any classifier* in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (2016), 1135–1144.

169. Riesenhuber, M. & Poggio, T. Hierarchical models of object recognition in cortex. *Nature neuroscience* **2,** 1019–1025 (1999).

170. Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* **65,** 386 (1958).

171. Roth, H. R. *et al.* Improving computer-aided detection using convolutional neural networks and random view aggregation. *IEEE transactions on medical imaging* **35,** 1170–1181 (2015).

172. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *nature* **323,** 533–536 (1986).

173. Russakovsky, O. *et al.* Imagenet large scale visual recognition challenge. *International journal of computer vision* **115,** 211–252 (2015).

174. Sakamoto, N., Haraguchi, T., Kobayashi, K., Miyazaki, Y. & Murata, T. Automated scratching detection system for black mouse using deep learning. *Frontiers in physiology,* 1466. ISSN: 1664-042X (2022).

175. Saltz, J. *et al.* Spatial organization and molecular correlation of tumor-infiltrating lymphocytes using deep learning on pathology images. *Cell reports* **23,** 181–193 (2018).

176. Salzberg, S. L. *C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993* 1994.

177. Sandsveden, M., Borgquist, S., Rosendahl, A. H. & Manjer, J. Low thyroid hormone receptor alpha-2 (THR$\alpha$-2) tumor expression is associated with unfavorable tumor characteristics and high breast cancer mortality. *Breast Cancer Research* **23,** 1–12 (2021).

178. El-Sappagh, S., Alonso, J. M., Islam, S. R., Sultan, A. M. & Kwak, K. S. A multilayer multimodal detection and prediction model based on explainable artificial intelligence for Alzheimer's disease. *Scientific reports* **11,** 2660 (2021).

179. Schaller, R. R. Moore's law: past, present and future. *IEEE spectrum* **34,** 52–59 (1997).

180. Schettini, F. & Prat, A. Dissecting the biological heterogeneity of HER2-positive breast cancer. *The Breast* **59,** 339–350 (2021).

181. Schmidhuber, J., Hochreiter, S., *et al.* Long short-term memory. *Neural Comput* **9,** 1735–1780 (1997).

182. Schölkopf, B. & Smola, A. J. *Learning with kernels: support vector machines, regularization, optimization, and beyond* (MIT press, 2002).

183. Selvaraju, R. R. *et al. Grad-cam: Visual explanations from deep networks via gradient-based localization* in *Proceedings of the IEEE international conference on computer vision* (2017), 618–626.

184. Selvaraju, R. R. *et al. Grad-cam: Visual explanations from deep networks via gradient-based localization* in *Proceedings of the IEEE international conference on computer vision* (2017), 618–626.

185. Sengupta, S., de Oliviera, K. A., Jin, L. & Clarke, R. Dysregulated TCA cycle is associated with endocrine therapy resistance in ER+ breast cancer cells. *Cancer Research* **82,** 2327–2327 (2022).

186. Shapley, L. *A value for n-person games In Contributions to the theory of games Annals of Mathematics, vol. 2* 1953.

187. Shimada, S. G. & LaMotte, R. H. Behavioral differentiation between itch and pain in mouse. *Pain* **139,** 681–687. ISSN: 0304-3959 (2008).

188. Simonyan, K., Vedaldi, A. & Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).

189. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

190. Solinski, H. J. *et al.* Nppb Neurons Are Sensors of Mast Cell-Induced Itch. *Cell Rep* **26,** 3561–3573 e4. ISSN: 2211-1247 (Electronic). https://www.ncbi.nlm.nih.gov/pubmed/30917312 (2019).

191. Spanhol, F. A., Oliveira, L. S., Petitjean, C. & Heutte, L. A dataset for breast cancer histopathological image classification. *Ieee transactions on biomedical engineering* **63,** 1455–1462 (2015).

192. Staff, S., Isola, J., Jumppanen, M. & Tanner, M. Aurora-A gene is frequently amplified in basal-like breast cancer. *Oncology reports* **23,** 307–312 (2010).

193. Stander, S. *et al.* Clinical classification of itch: a position paper of the International Forum for the Study of Itch. *ACTA DERMATOVENEREOLOGICA-STOCKHOLM-* **87,** 291. ISSN: 0001-5555 (2007).

194. Stoll, G. *et al.* Pro-necrotic molecules impact local immunosurveillance in human breast cancer. *Oncoimmunology* **6,** e1299302 (2017).

195. Sun, Y. G. & Chen, Z. F. A gastrin-releasing peptide receptor mediates the itch sensation in the spinal cord. *Nature* **448,** 700–3. ISSN: 1476-4687 (Electronic) 0028-0836 (Linking). https://www.ncbi.nlm.nih.gov/pubmed/17653196 (2007).

196. Sundararajan, M., Taly, A. & Yan, Q. *Axiomatic attribution for deep networks* in *International conference on machine learning* (2017), 3319–3328.

197. Thalor, A., Joon, H. K., Singh, G., Roy, S. & Gupta, D. Machine learning assisted analysis of breast cancer gene expression profiles reveals novel potential prognostic biomarkers for triple-negative breast cancer. *Computational and structural biotechnology journal* **20,** 1618–1631 (2022).

198. Thorpe, S., Fize, D. & Marlot, C. Speed of processing in the human visual system. *nature* **381,** 520–522 (1996).

199. Thurmond, R. L., Gelfand, E. W. & Dunford, P. J. The role of histamine H1 and H4 receptors in allergic inflammation: the search for new antihistamines. *Nature reviews Drug discovery* **7,** 41–53. ISSN: 1474-1784 (2008).

200. Tomczak, K., Czerwińska, P. & Wiznerowicz, M. Review The Cancer Genome Atlas (TCGA): an immeasurable source of knowledge. *Contemporary Oncology/Współczesna Onkologia* **2015,** 68–77 (2015).

201. Uzunoglu, H. *et al.* Association of the nibrin gene (NBN) variants with breast cancer. *Biomedical reports* **4,** 369–373 (2016).

202. Vale-Silva, L. A. & Rohr, K. Long-term cancer survival prediction using multimodal deep learning. *Scientific Reports* **11,** 13505 (2021).

203. Van Der Groep, P. *et al.* Loss of expression of FANCD2 protein in sporadic and hereditary breast cancer. *Breast cancer research and treatment* **107,** 41–47 (2008).

204. Van der Laak, J., Litjens, G. & Ciompi, F. Deep learning in histopathology: the path to the clinic. *Nature medicine* **27,** 775–784 (2021).

205. Vaswani, A. *et al.* Attention is all you need. *Advances in neural information processing systems* **30** (2017).

206. Vleugel, M. M., Greijer, A. E., Bos, R., van der Wall, E. & van Diest, P. J. c-Jun activation is associated with proliferation and angiogenesis in invasive breast cancer. *Human pathology* **37,** 668–674 (2006).

207. Wahba, G. *Spline models for observational data* (SIAM, 1990).

208. Walter, C. Kryder's law. *Scientific American* **293,** 32–33 (2005).

209. Wang, D., Khosla, A., Gargeya, R., Irshad, H. & Beck, A. H. Deep learning for identifying metastatic breast cancer. *arXiv preprint arXiv:1606.05718* (2016).

210. Wang, H., Xing, F., Su, H., Stromberg, A. & Yang, L. Novel image markers for non-small cell lung cancer classification and survival prediction. *BMC bioinformatics* **15,** 1–12 (2014).

211. Wang, S., Yao, J., Xu, Z. & Huang, J. *Subtype cell detection with an accelerated deep convolution neural network* in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19* (2016), 640–648.

212. Wang, X. *et al. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 2097–2106.

213. Weisshaar, E. & Dalgard, F. Epidemiology of itch: adding to the burden of skin morbidity. *Acta Derm Venereol* **89,** 339–50. ISSN: 1651-2057 (Electronic) 0001-5555 (Linking). https://www.ncbi.nlm.nih.gov/pubmed/19688144 (2009).

214. Wiesmann, F. *et al.* Frequent loss of endothelin-3 (EDN3) expression due to epigenetic inactivation in human breast cancer. *Breast Cancer Research* **11,** 1–18 (2009).

215. Wilkinson, M. D. *et al.* The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data* **3,** 1–9 (2016).

216. Wimalasena, N. K. *et al.* Dissecting the precise nature of itch-evoked scratching. *Neuron* **109,** 3075–3087 e2. ISSN: 1097-4199 (Electronic) 0896-6273 (Linking). https://www.ncbi.nlm.nih.gov/pubmed/34411514 (2021).

217. Wright, A. M. *et al.* Digital slide imaging in cervicovaginal cytology: a pilot study. *Archives of pathology and laboratory medicine* **137,** 618–624 (2013).

218. Wulczyn, E. *et al.* Deep learning-based survival prediction for multiple cancer types using histopathology images. *PloS one* **15,** e0233678 (2020).

219. Xiaoxuan, L. *et al.* A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis. *The lancet digital health* **1,** e271–e297 (2019).

220. Xie, S., Girshick, R., Dollár, P., Tu, Z. & He, K. *Aggregated residual transformations for deep neural networks* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 1492–1500.

221. Xiong, J. *et al.* Multimodal machine learning using visual fields and peripapillary circular OCT scans in detection of glaucomatous optic neuropathy. *Ophthalmology* **129,** 171–180 (2022).

222. Xu, Y. *et al.* Comprehensive Analysis of Necroptosis-Related Genes as Prognostic Factors and Immunological Biomarkers in Breast Cancer. *Journal of Personalized Medicine* **13,** 44 (2022).

223. Yang, C. *et al.* CXCL1 stimulates migration and invasion in ER-negative breast cancer cells via activation of the ERK/MMP2/9 signaling axis. *International journal of oncology* **55,** 684–696 (2019).

224. Yang, L. *et al.* Deep learning based multimodal progression modeling for Alzheimer's disease. *Statistics in Biopharmaceutical Research* **13,** 337–343 (2021).

225. Yao, J., Wang, S., Zhu, X. & Huang, J. *Imaging biomarker discovery for lung cancer survival prediction* in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19* (2016), 649–657.

226. Yao, J., Zhu, X. & Huang, J. *Deep multi-instance learning for survival prediction from whole slide images* in *Medical Image Computing and Computer Assisted Intervention–*

*MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part I 22* (2019), 496–504.

227. Ye, M. *et al.* Interfering with CSE1L/CAS inhibits tumour growth via C3 in triple-negative breast cancer. *Cell proliferation* **55,** e13226 (2022).

228. Yoon, H.-J., Ramanathan, A. & Tourassi, G. *Multi-task deep neural networks for automated extraction of primary site and laterality information from cancer pathology reports* in *INNS Conference on Big Data* (2016), 195–204.

229. Yosinski, J., Clune, J., Bengio, Y. & Lipson, H. How transferable are features in deep neural networks? *Advances in neural information processing systems* **27** (2014).

230. Yosipovitch, G., Rosen, J. D. & Hashimoto, T. Itch: From mechanism to (novel) therapeutic approaches. *J Allergy Clin Immunol* **142,** 1375–1390. ISSN: 1097-6825 (Electronic) 0091-6749 (Linking). https://www.ncbi.nlm.nih.gov/pubmed/30409247 (2018).

231. Yosipovitch, G. & Fleischer, A. B. Itch associated with skin disease. *American journal of clinical dermatology* **4,** 617–622. ISSN: 1179-1888 (2003).

232. Yu, H., Wangensteen, K., Deng, T., Li, Y. & Luo, W. MRGPRX4 in Cholestatic Pruritus. *Semin Liver Dis* **41,** 358–367. ISSN: 1098-8971 (Electronic) 0272-8087 (Linking). https://www.ncbi.nlm.nih.gov/pubmed/34161994 (2021).

233. Yu, J. *et al.* Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917* (2022).

234. Zadeh, S. G. & Schmid, M. Bias in cross-entropy-based training of deep survival networks. *IEEE transactions on pattern analysis and machine intelligence* **43,** 3126–3137 (2020).

235. Zaheer, M. *et al.* Deep sets. *Advances in neural information processing systems* **30** (2017).

236. Zeiler, M. D., Krishnan, D., Taylor, G. W. & Fergus, R. *Deconvolutional networks* in *2010 IEEE Computer Society Conference on computer vision and pattern recognition* (2010), 2528–2535.

237. Zeng, T., Li, R., Mukkamala, R., Ye, J. & Ji, S. Deep convolutional neural networks for annotating gene expression patterns in the mouse brain. *BMC bioinformatics* **16,** 1–10 (2015).

238. Zhang, A., Lipton, Z. C., Li, M. & Smola, A. J. *Dive into Deep Learning* https://D2L.ai (Cambridge University Press, 2023).

239. Zhang, H., Stebbing, J. & Giamas, G. The many-faced KSR1: a tumor suppressor in breast cancer. *Oncoscience* **2,** 669 (2015).

240. Zhang, W. *et al.* Deep model based transfer and multi-task learning for biological image analysis. *IEEE transactions on Big Data* **6,** 322–333 (2016).

241. Zheng, T. *et al.* A machine learning-based framework to identify type 2 diabetes through electronic health records. *International journal of medical informatics* **97,** 120–127 (2017).

242. Zhong, Y. *et al. Characterizing design patterns of EHR-driven phenotype extraction algorithms* in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (2018), 1143–1146.

243. Zhou, C. *et al.* TGFB2-AS1 inhibits triple-negative breast cancer progression via interaction with SMARCA4 and regulating its targets TGFB2 and SOX2. *Proceedings of the National Academy of Sciences* **119,** e2117988119 (2022).

244. Zhou, J. & Troyanskaya, O. G. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods* **12,** 931–934 (2015).

245. Zhou, X. *et al.* Prognostic biomarkers related to breast cancer recurrence identified based on Logit model analysis. *World journal of surgical oncology* **18,** 1–10 (2020).

246. Zhu, W. & Xie, X. Adversarial deep structural networks for mammographic mass segmentation. *Biorxiv,* 095786 (2016).

247. Zhu, X., Yao, J. & Huang, J. *Deep convolutional neural network for survival analysis with pathological images* in *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (2016), 544–547.

248. Zong, B. *et al. Deep autoencoding gaussian mixture model for unsupervised anomaly detection* in *International conference on learning representations* (2018).

249. Zou, J. *et al.* A primer on deep learning in genomics. *Nature genetics* **51,** 12–18 (2019).