UNIVERSITY OF CALIFORNIA
RIVERSIDE

Understanding Tradeoffs Between Algorithm Complexity and Performance

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Computer Science

by

Abhijeet R Murthy

March 2023

Thesis Committee:

Dr. Bir Bhanu, Chairperson
Dr. Chinya V. Ravishankar
Dr. Ahmed Eldawy

The Thesis of Abhijeet R Murthy is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I am grateful to my advisor, without whose help, I would not have been here.

To my parents for all the support.

ABSTRACT OF THE THESIS

Understanding Tradeoffs Between Algorithm Complexity and Performance

by

Abhijeet R Murthy

Master of Science, Graduate Program in Computer Science
University of California, Riverside, March 2023
Dr. Bir Bhanu, Chairperson

The Minimum Description Length (MDL) principle asserts that the best model given some data is the one that minimizes the combined cost of describing the model and the misfit between the model and data with a goal to maximize regularity extraction for optimal data compression, prediction, and communication.

We utilize the MDL principle to comprehend the relationship between the complexity or size of the problem/representation with respect to the model's performance. Evaluating a large number of possibilities will enable us to compare model tradeoffs and select the model with the best performance but the least complexity.

The complexity of the problem can be determined by factors such as the size of the representation, number of lines of code, amount of processing time, and understanding the order of computation. In some approaches, the order of magnitude is defined as the number of basic computations performed by the model. In a deep learning model, it can be defined as the number of nodes that the data are propagated through. In the proposed approach we introduce a criterion function that accounts for system resources through the

use of number of FLOPs (floating-point operations) and allows an accurate representation of size of a model. This is combined with performance (error/accuracy) to estimate diverse tradeoffs.

We train, validate and evaluate model performance for multiple model parameters and hyperparameters. We use the criterion function based on the MDL principle to select the best model. We also use the criterion function to understand the tradeoffs of the model.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

There are several techniques for deep learning model selection, each with its own advantages and disadvantages. Some of the most commonly used techniques include grid search, random search, Bayesian optimization, evolutionary algorithms, and transfer learning.Grid search involves testing all possible combinations of hyperparameters, while random search randomly samples from a specified range of hyperparameters. Bayesian optimization constructs a probabilistic model of the hyperparameter space, and evolutionary algorithms use principles inspired by natural selection to search for the optimal parameter combination. Transfer learning involves fine-tuning a pre-trained model on a new task with a smaller set of learnable parameters. Regularization methods such as weight decay, dropout, and early stopping can also be used to improve model generalization performance. Ultimately, the choice of a model selection technique depends on the specific problem and available resources.

Minimum Description Length (MDL) is another optimization algorithm that can be used for model selection. It is based on the principle of Occam's razor, which states that the simplest explanation that fits the data is the most likely one. In MDL, the goal is to find the model that has the shortest description length while still fitting the data well.

The MDL criterion takes into account both the complexity of the model and the fit to the data, and it balances these factors to find the best model. The idea is that a good model should not only fit the data well, but also be simple and easy to understand.

Ultimately, the choice of a model selection technique depends on the specific problem and the available resources. By selecting the best model for a given task, we can achieve state-of-the-art performance on a wide range of applications in computer vision, natural language processing, and other fields.

1.2 Motivation

Model selection in deep learning is a crucial step in the development of a machine learning system, as it helps to determine the best model to use for a given problem. The motivation for model selection in deep learning can be summarized as follows:

Generalization: The ultimate goal of any machine learning model is to generalize well to new, unseen data. Model selection helps to ensure that the chosen model has learned the relevant patterns in the training data and is able to generalize to new data.

Overfitting: Deep neural networks have a large number of parameters, and if not controlled properly, can easily overfit the training data. Model selection helps to choose a model that has just the right amount of capacity to fit the data without overfitting.

Computational efficiency: Deep neural networks can be computationally expensive to train and run. Model selection can help to identify a model that is computationally efficient while still being accurate enough for the given problem.

Interpretability: In some cases, interpretability of the model is important, especially in applications such as healthcare, where understanding the reasoning behind the model's decision is crucial. Model selection can help to identify models that are more interpretable, such as decision trees or linear models.

Trade-off between performance and complexity: In many cases, there is a trade-off between model performance and complexity. Model selection can help to identify models that strike the right balance between the two, leading to more efficient and effective machine learning systems.

Overall, model selection is essential in deep learning to ensure that the chosen model can generalize well to new data, is not overfitting, is computationally efficient, and strikes the right balance between performance and complexity.

Chapter 2

Previous Approaches

RegNorm [9] is a deep learning regularization method that applies unsupervised attention across neural network layers. Inspired by neuroscience, RegNorm learns to attend to the most regular patterns in the data to reduce the impact of noisy or irregular patterns, leading to improved generalization performance. The principle of Minimum Description Length (MDL) can be applied to RegNorm to determine the optimal regularization strength that balances the model's complexity and its ability to fit the training data. By selecting the regularization strength that achieves the best tradeoff between model complexity and performance on the training data, RegNorm ensures that the model is regularized to the appropriate degree without overfitting or underfitting.

The paper "Enhancing multi-objective evolutionary neural architecture search with training-free Pareto local search"[12] proposes a method to enhance multi-objective evolutionary neural architecture search (MOENA) by incorporating a training-free Pareto local search (PLS) algorithm. The authors argue that PLS can help improve the efficiency

of MONEA by exploring and exploiting the search space in a more effective way. They evaluate their proposed method on three benchmark datasets (CIFAR-10, CIFAR-100, and ImageNet) and demonstrate that it outperforms state-of-the-art MONEA methods in terms of Pareto front diversity and convergence speed. Additionally, they show that their method can achieve better performance on the ImageNet dataset with significantly fewer search costs compared to the previous methods. The authors suggest that their method can be a valuable tool for automating the design of neural networks for a range of computer vision tasks.

The paper "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures" [8] describes a method for optimizing hyperparameters of deep learning models in high-dimensional spaces. The authors propose a new optimization algorithm, called Hyperband, which combines random search with a principled early-stopping scheme to efficiently search a large space of hyperparameters. The method is demonstrated on several computer vision tasks, achieving state-of-the-art results with significantly fewer computational resources compared to previous methods. The paper presents an extensive experimental evaluation and provides insights into the trade-offs between different optimization algorithms and hyperparameter settings.

The AME model [7] is a parallel hyper-parameter optimization approach that utilizes attention and memory mechanisms to improve optimization performance. The model's efficiency and effectiveness can be or has been evaluated using the Minimum Description Length (MDL) principle, which ensures that the best model is selected with the fewest number of evaluations. The AME model uses a memory mechanism to store previous eval-

uations and an attention mechanism to focus on promising regions of the hyper-parameter space, making it computationally efficient and able to handle large hyper-parameter spaces. By optimizing hyper-parameters with the fewest number of evaluations, the AME model can significantly reduce the computational cost of hyper-parameter optimization.

Chapter 3

Technical Approach

3.1 MDL Principle

A key principle in machine learning is the minimum description length (MDL) principle, which states that the best model is one that achieves the optimal balance between complexity and its ability to explain the data. This principle is essential not only in traditional machine learning but also in deep learning, a subset of machine learning that has become increasingly popular in recent years.In deep learning, the MDL principle helps determine the optimal complexity of a neural network model. This involves finding the smallest network that can explain the training data effectively without overfitting. Overfitting occurs when the model is too complex and fits the noise in the training data, leading to poor performance on new data. To address this issue, techniques like model compression are used, where the original model is simplified or reduced in size without losing too much accuracy.By balancing model complexity and accuracy, the MDL principle can guide the development of efficient and effective deep learning models. To achieve this balance, we

have developed a criterion function that enables us to evaluate the trade-off between model complexity and performance. This helps us select deep learning models that can generalize well to new data, making them suitable for deployment in resource-constrained devices or real-time applications.

3.2 FLOPs

FLOPs, or Floating-Point Operations, are a measure of the computational complexity of a mathematical operation. They are commonly used in the context of deep neural networks, which involve the manipulation of large arrays of floating-point numbers. FLOPs can help to estimate the amount of computation required to perform a specific operation, such as a convolution or matrix multiplication. By comparing the FLOPs of different models, we can assess their computational complexity and optimize their efficiency by reducing the number of FLOPs required. Generally, a higher number of FLOPs indicates a more computationally demanding model, which may require more processing power and memory to train and run.

3.3 Understanding the Tradeoff of time

Time can be influenced by factors outside the model's control: Factors such as network latency, hardware limitations, and other environmental factors can impact the time it takes to make predictions, even if the model is performing well. Time can be optimized at the expense of accuracy: If time is the primary metric being optimized, a model may sacrifice accuracy in order to make faster predictions. This can result in poor

performance and incorrect predictions. Time does not provide insight into the quality of predictions: Even if a model makes predictions quickly, if those predictions are inaccurate, the model is not performing well. Therefore, while time is an important consideration for deploying machine learning models, it should not be used as a primary metric for evaluating model performance. Instead, metrics such as accuracy, precision, recall, F1 score, and area under the receiver operating characteristic (ROC) curve should be used to assess model performance.

3.4 Criterion Functions

Criterion functions are an essential component of deep learning model selection, as they provide a way to evaluate the performance of a model and guide the optimization process during training. The choice of criterion function depends on the task at hand, and there are various types of criterion functions used in deep learning. For example, Mean Squared Error (MSE) is used for regression tasks, measuring the average squared difference between the predicted and true output values. Cross-Entropy Loss is used for classification tasks, measuring the difference between the predicted class probabilities and the true class labels. Binary Cross-Entropy Loss is a variant of the cross-entropy loss used for binary classification tasks, while Categorical Cross-Entropy Loss is used for multiclass classification tasks. KL Divergence is used in unsupervised learning and reinforcement learning, measuring the difference between the predicted and true probability distributions.

The choice of a criterion function can have a significant impact on the performance of a deep learning model. It is often used in conjunction with other techniques such as

regularization and optimization algorithms to improve the accuracy of the model. The goal is to minimize the value of the criterion function, which corresponds to maximizing the accuracy of the model on the training data.

The MDL criterion function formalizes this idea by using the length of the code or description of the model and the data to evaluate the goodness of fit of the model to the data. The MDL criterion function can be used for various purposes, such as feature selection, model selection, and hypothesis testing.

The criterion function used in Minimum Description Length (MDL) principle can be formulated as follows:

$$\text{Cost} = L(\text{model}) + L(\text{data} \mid \text{model}) \tag{3.1}$$

where $L(\text{model})$ is the complexity or size of the model, and $L(\text{data} \mid \text{model})$ is the misfit between the model and the data. The goal is to minimize the cost function to find the best model that fits the data well without overfitting.

This criterion function balances the tradeoff between the complexity of the model ($L(\text{model})$) and its ability to explain the data ($L(\text{data} \mid \text{model})$). The goal is to find the model that can describe the data with the least amount of code or description, without sacrificing the accuracy or goodness of fit to the data.

minimize, $MDL = -(\alpha_1 \log_2(FLOPs) + \alpha_2 \log_2(d) + \alpha_3 n_e \log_2, (N))$

where α_1 and α_2 are scaling factors for FLOPs and the embedding size d, respectively; FLOPs is the number of floating-point operations required by the model; d is the dimensionality of the learned embeddings; where n_e is the effective number of training examples, which accounts for data augmentation and other factors that increase the effective

size of the training set; and α_3 is the scaling factors for N which is the total number of images used for training. In this experiment the value of α_1 is 0.025, α_2 is 0.1, α_3 is 0.1.

This is used to understand the trade-off between model complexity (measured by FLOPs and embedding size) and the training set error. As the FLOPs and embedding size increase, the model becomes more complex and may be better able to capture the underlying patterns in the data. However, this comes at the cost of increased computational resources and training time. On the other hand, increasing the effective size of the training set can improve the generalization performance of the model by reducing overfitting. However, this may require more data and/or more data augmentation, which can also increase computational requirements.

3.5 Method Description

Proposed approach evaluates a model using a wide range of models and hyperparameters to find the best combination that produces the optimal performance. This can be done by testing a model on a variety of data sets and measuring its performance using different error metrics. This method is used to understand the contribution of each component to the overall performance of the model and to identify the most critical components in a system. This method can also be used to identify which components are responsible for overfitting or underfitting, allowing for better regularization techniques to be developed.

To better understand the tradeoffs between algorithm complexity, performance, and system resources, 2D and 3D visualizations can be created to visualize the relationships between these factors. For example, a 2D plot might show the relationship between

algorithm complexity and performance, while a 3D plot could show the relationship between algorithm complexity, performance, and embedding size. By evaluating the model with different combinations of parameters and visualizing the results, we can gain insights into how the model performs under different conditions. This can help us choose the best parameters for our model and optimize its performance.

Chapter 4

TestBed for Evaluation

4.1 Data

Our dataset contains indoor and outdoor media files of a large number of subjects. The indoor focuses on capturing the person in a controlled environment and the field is focused on replicating real-world scenarios for recognition The purpose of using the above dataset is to evaluate and develop person recognition algorithms that can handle the challenges associated with outdoor environments. The dataset contains a variety of images captured from different outdoor settings and perspectives, including distortions and occlusions. Therefore, it can be used to test the performance of existing person recognition algorithms in various outdoor environments and to develop new algorithms that can accurately recognize individuals under changing lighting and weather conditions, distance, and altitude. By using this dataset, researchers can improve the accuracy and robustness of person recognition systems and enhance their real-world applicability.

4.2 Data Preprocessing:

The HMR2.0 model requires a specific preprocessing of the input RGB image to ensure accurate predictions of human body shape and pose. For accurate predictions, the model needs an unobstructed view of the human body, which means that the human detected using the model should be placed in the center of the image with some background. This requires resizing and cropping the input image.

Size normalization is a crucial step in this preprocessing. The amount of background to include in the cropped image depends on the height of the individual. Shorter people require more background to be included in the cropped image, while taller people require less. Our method involves using the ground-truth information of the human, such as height or bounding box information, to determine the required amount of background.

4.3 Model Architecture

The goal of the model is to identify a person using a single image with the following properties: view independence, clothing independence, body articulation, and robustness to partial occlusion and distortion. View independence means that the model should recognize a person from any angle or orientation, regardless of the camera's position or angle. Clothing independence means that the model should identify a person regardless of what they are wearing. The model should also be body articulation invariant, able to recognize a person even if they are in different poses or positions. In addition, the model should be robust to partial occlusion and distortion. Partial occlusion means that the person's face or body may be partially obstructed by objects, such as hats, glasses, or other people, and the

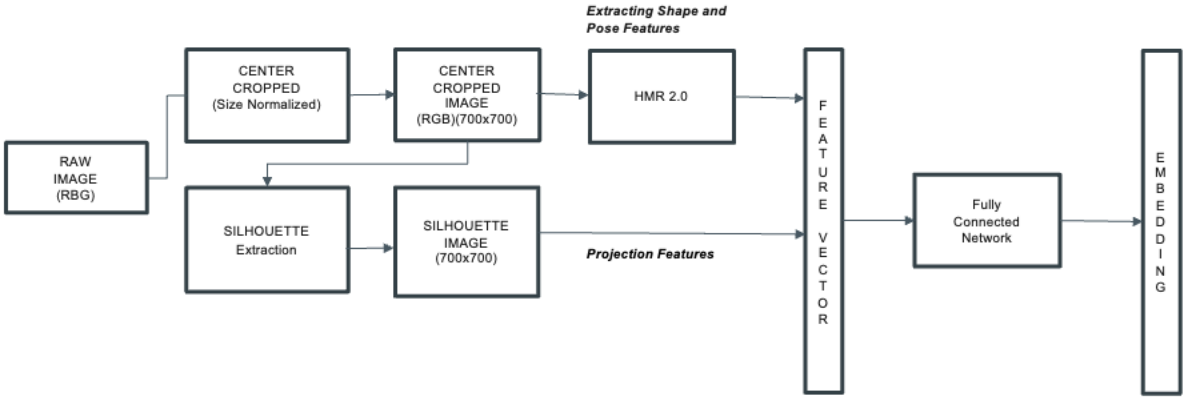

Figure 4.1: Model Architecture

model should still be able to identify the person. Distortion means that the image may be distorted or blurred, and the model should still be able to recognize the person.

4.3.1 HMR2.0

Our model uses HMR2.0 [13] to extract features from input images. HMR2.0 is a deep neural network that takes a single image as input and produces the 3D pose and shape of the human body as output. It consists of two stages: a pose stage and a shape stage. In the pose stage, the network uses a heatmap regression approach to estimate the 3D joint locations of the body. In the shape stage, the network refines the joint locations and estimates the 3D shape of the body using a template-based approach. This approach has been shown to be effective in a variety of applications, such as human motion tracking and analysis [1, 2].

4.3.2 Extracted Features

The HMR2.0 model was used to extract three key features from the images: projection, shape, and 3D joint and 2D distances. The projection feature is a 448x1 vector that captures the person's width by projecting the image onto the x and y axes. This feature encodes the overall shape of the person in the image, providing information that is important for recognizing individuals across different poses and viewpoints.

The shape feature is a 10x1 vector generated using HMR 2.0 and captures information about the person's body shape, such as their height, weight, and body proportions. This feature provides additional information that can aid in distinguishing individuals and improving the accuracy of the facial recognition model.

Finally, the 3D joint and 2D distances feature is a 21x1 vector that captures the distances between the joints of a person in the image. Joint locations are estimated using HMR 2.0 pose parameters, and 3D joint distances are calculated directly from these locations, while the 2D joint distances are obtained by projecting the 3D joint locations onto the image plane. This feature encodes information about the person's pose and orientation in the image, which is crucial for recognizing individuals across different poses and viewpoints.

Overall, the combination of these features provides a comprehensive representation of the person's shape, pose, and orientation, which can be used to accurately identify individuals in various settings and scenarios.

4.3.3 Dense Layer

A fully connected network, also referred to as a dense layer, is a type of neural network architecture that allows every neuron in one layer to connect to every neuron in the next layer. This design is widely used as a final layer in neural networks for tasks like classification and regression. Additionally, fully connected layers can be utilized as intermediate layers in deep neural networks, where they help to extract features from input data.

In our approach, we leverage a dense layer to generate embeddings for each input image, using the extracted features. These embeddings serve as representations of the input images and are used to provide match scores or rank-k accuracy. By utilizing the dense layer, we are able to capture the underlying patterns and features in the input data, enabling us to accurately identify individuals across different poses and viewpoints

4.4 Metrics for Evaluation

Commonly used metrics for evaluating facial recognition models include accuracy, false acceptance rate (FAR), false rejection rate (FRR), Receiver Operating Characteristic (ROC) curve, and Area Under the Curve (AUC). Accuracy measures the proportion of correctly identified objects, while FAR measures the proportion of incorrect matches and FRR measures the proportion of correct matches rejected by the system. The ROC curve helps to evaluate the trade-off between FAR and FRR, and the AUC provides an overall measure of performance. In our approach, we have used the Rank-n accuracy metric which measures the proportion of test samples where the correct object is among the top n pre-

dicted matches returned by the system. Meanwhile, the Cumulative Match Curve (CMC) plot is used to evaluate the performance of recognition systems in biometrics. It measures the percentage of correct matches in a ranked list of the top k matches, where the y-axis represents the percentage of correct matches up to rank k, and the x-axis represents the rank k.

A perfect recognition system would have a step function that reaches 100 Percentage accuracy at rank 1 in the CMC plot. The CMC plot can be used to compare the performance of different systems, identify the rank at which a system achieves a certain level of accuracy, and identify the point of saturation in the system. These metrics can be used individually or in combination to evaluate the performance of a recognition system, depending on the specific task and application.

Chapter 5

Experiment Results

5.1 Tradeoffs using Variable Embedding Size for Training

The size of the embedding generated by the fully connected network can have an impact on the performance of the model. Typically, larger embeddings can capture more information but can also increase the computational cost and may lead to overfitting if the model is not regularized properly.

One approach to determine the optimal embedding size is to experiment with different sizes and evaluate the model's performance on a validation set To perform the experiment, we can train multiple versions of the model with different embedding sizes and evaluate them on the validation set. We can then compare their performance and choose the embedding size that yields the best results.

We have a variable size data set in our experiment. 64 embedded size were used in the first experiment. In the second experiment, 128 embedded size were used, and in the third experiment, 256 embedded size were used. We have 60 training subjects. We have

15 validation subjects in all of the above experiments. We are testing all of the models on a new data set that has 20 subjects that the model has never seen before. The following experiments yielded the following results:

Embedding Size	**Rank 1**	**Rank 5**	**Rank 10**	**FLOPs**
64	12.88	44.09	71.5	4.27×10^8
128	14.90	47.77	72.57	1.07×10^9
256	15.54	51.73	75.89	2.99×10^9

Table 5.1: Experimental Accuracy using variable embedding size

Error 1	**Embedded Size**	**N**	**FLOPs**	**MDL Criterion**
87.12	64	22500	1.07×10^9	2.6577
85.10	128	43000	1.07×10^9	2.7597
84.46	256	65000	1.07×10^9	2.8869

Table 5.2: Experimental Error using variable embedded size

The experiment used a feature vector of length 479, which represented shape, 2D/3D distances, and projection features. The experiment explored the effect of different embedding sizes on the performance of the algorithm. The results showed that increasing the embedding size can help capture more information about the human face, but there is a tradeoff in terms of computational cost. The results also showed that using an embedding size of 128 can provide good performance without incurring excessive computational costs. Therefore, it may be worthwhile to explore and optimize the model with this embedding size further to achieve better performance while still maintaining computational efficiency.

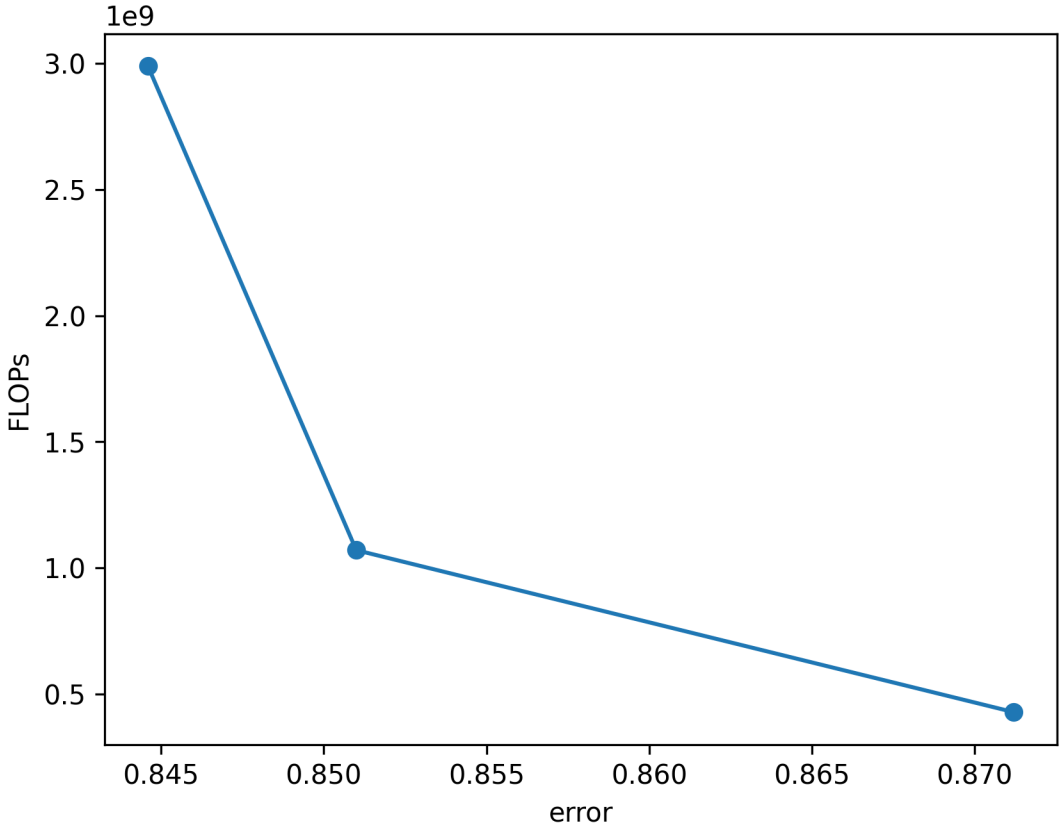

Figure 5.1: Experiment1 - Error Vs FLOPs

5.2 Tradeoffs for Feature Selection

Training the model by considering two different sets of features at a time can help determine the most influential feature set for accurate predictions. By comparing the performance of the model when trained with different combinations of features, we can understand which features have a greater impact on the model's ability to predict the shape and pose of the human body. For example, we can train the model using the projection feature and shape feature, then compare its performance to when trained using the shape feature and 3D joint and 2D distances feature, and so on. This can help identify the most important features and their relative importance for accurate predictions.

We are using Shape, Pose and Projection XY as the parameters for our experiment. The following experiments yielded the following results:

Features Used	**Rank 1**	**Rank 5**	**Rank 10**	**FLOPs**
Shape + 2D 3D distances	11.15	40.68	67.43	4.7×10^8
Shape + projection x,y	14.26	47.33	72.13	1.04×10^9
2D 3D distances + projection x,y	15.67	48.36	75.56	1.06×10^9
All features	14.90	47.77	72.57	1.07×10^9

Table 5.3: Experimental Accuracy using features

Features Used	**Error 1**	**N**	**FLOPs**	**MDL**
Shape + 2D 3D distances	87.12	43000	4.7×10^8	2.7878
Shape + projection x,y	85.74	43000	1.04×10^9	2.7685
2D 3D distances + projection x,y	84.33	43000	1.06×10^9	2.7475
All features	85.10	43000	1.07×10^9	2.7597

Table 5.4: Experimental Error using features

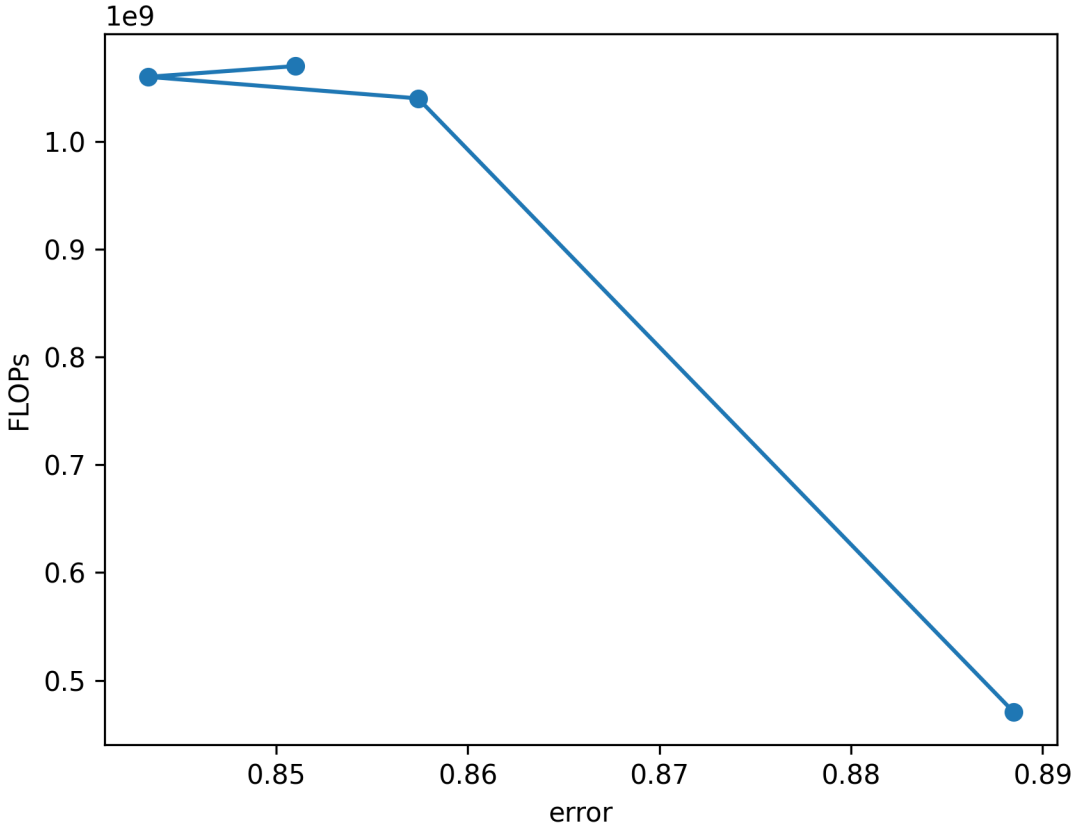

Figure 5.2: Experiment1 - Error Vs FLOPs

The experiment conducted highlights the importance of feature selection in achieving high accuracy in recognition tasks. The feature vector used included 10 shape features, 21 2D/3D distance features, and 448 projection features, which were analyzed in pairs to measure the model's accuracy. Interestingly, the results revealed that the combination of 2D/3D distance features and projection features provided the best accuracy. This indicates that these features play a critical role in ecognition, providing good discriminative power and allowing for the accurate identification of individuals. However, it was also found that the inclusion of shape features had a negative impact on model performance, suggesting that these features may not be as crucial for recognition or may not provide useful information for this task.

5.3 Tradeoffs using Variable Sizes of Data for Training

Deep learning models require a large amount of data for training, which can be difficult to obtain in certain scenarios where data collection is costly or limited. In such cases, using variable-sized data and different splits of data can help in determining if the deep learning model can be trained effectively using a lower number of samples. By evaluating the model's performance on a smaller number of samples, we can determine the optimal size of the training set, which can help in reducing the training time and computational resources required for training the model.

Using variable-sized data during training can also improve the robustness of the model. By training the model on data of varying sizes, the model can learn to handle different input sizes and accurately predict the shape and pose of the human body. This

can be especially useful in real-world scenarios where the input image sizes may vary due to differences in camera resolution, distance, and other factors. Overall, using variable-sized data and considering different splits of data can be a helpful strategy for optimizing the training process and improving the performance of deep learning models.

We have a variable size data set in our experiment. 30 data sets were used in the first experiment. In the second experiment, 60 data sets were used, and in the third experiment, 90 data sets were used. We have 15 validation subjects in all of the above experiments. We are testing all of the models on a new data set that has 20 subjects that the model has never seen before. The following experiments yielded the following results:

No.Subjects	**Rank 1**	**Rank 5**	**Rank 10**	**FLOPs**
30 Subjects	14.11	46.30	72.44	1.07×10^9
60 Subjects	14.90	47.77	72.57	1.07×10^9
90 Subjects	14.93	53.15	77.00	1.07×10^9

Table 5.5: Experimental Accuracy using variable training size

No.Subjects	**Error 1**	**N**	**FLOPs**	**MDL Criterion**
30 Subjects	85.89	22500	1.07×10^9	2.6916
60 Subjects	85.10	43000	1.07×10^9	2.7597
90 Subjects	85.07	65000	1.07×10^9	2.8099

Table 5.6: Experimental Error using variable training size

Increasing the number of training samples improves the model's ability to recognize people at rank 1, 5, and 10. This suggests that the model is capable of generalization and is learning to recognize individuals based on their features. The higher accuracy rates

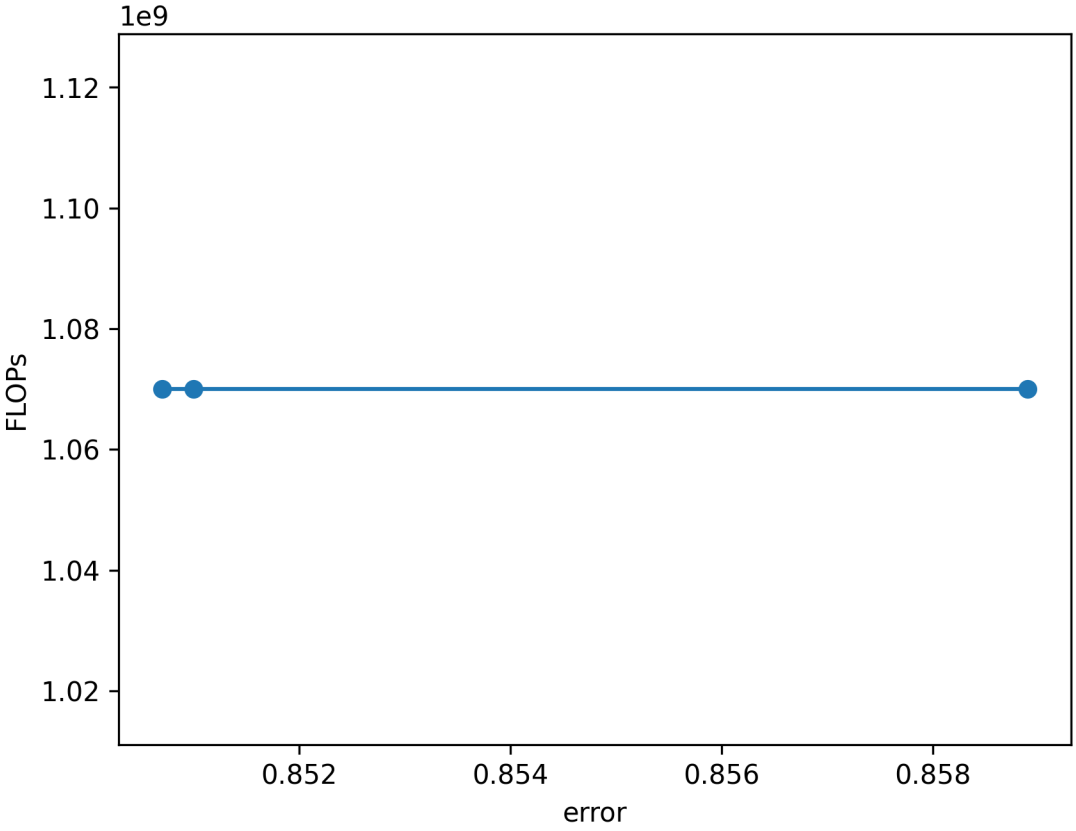

Figure 5.3: Experiment1 - Error Vs FLOPs

obtained with a larger number of training samples indicate that the model performs better when exposed to more diverse samples of faces. This supports the idea that the model is capable of recognizing a wider range of individuals when provided with more training data. Overall, the results of this experiment suggest that providing a model with a greater amount of training data can lead to improved performance in recognition tasks. Additionally, the more individuals that the model is trained to recognize, the better it becomes at accurately identifying people in general.

5.4 Discussion of Results

The discussion highlights some of the key factors that contribute to achieving high performance in recognition tasks. By carefully selecting the most important features, such as shape, pose, and 2D/3D projection features, and balancing model complexity with performance, it is possible to create models that generalize effectively and provide accurate recognition of individuals at different ranks.

In particular, the use of larger and more diverse training datasets can significantly improve model performance, providing a solid foundation for effective recognition. This can be further augmented by leveraging advanced techniques such as the minimum description length (MDL) criterion to guide model selection and avoid overfitting. It can be observed that the criterion function gives us a good representation of the model complexity and could be used as a metric to optimize the model.s

Overall, these findings point to the need for a nuanced and careful approach to recognition, one that balances the importance of different factors and leverages cutting-edge

techniques to achieve optimal performance. By continuing to refine our understanding of these factors and exploring new approaches to model optimization, we can continue to make significant progress toward creating more accurate and effective recognition systems.

Chapter 6

Conclusions

Deep learning has emerged as a powerful tool for various applications, including computer vision, natural language processing, and speech recognition. However, deep learning models can be computationally expensive, requiring significant computational resources to train and deploy. As such, it is important to consider the trade-offs between model complexity and accuracy in order to develop efficient and effective models.

To address this issue, we propose a formula for evaluating the quality of a deep learning model in terms of its accuracy and efficiency. This formula takes into account three main factors: the number of floating point operations required by the model (FLOPs), the embedding size, and the effective size of the training set. The FLOPs measure the computational cost of a model and are often used as a proxy for its complexity. The embedding size is the dimensionality of the learned representations in the model, and the effective size of the training set accounts for data augmentation and other factors that increase the effective size of the training data.

We examine how increasing the FLOPs of a model affects its performance according to this formula. Specifically, we consider the impact of FLOPs on the accuracy and efficiency of the model, and explore the trade-offs between model complexity and accuracy. By analyzing the formula, we show that increasing the FLOPs generally leads to better accuracy, but also increases the computational cost of the model. Therefore, there is a trade-off between model complexity and accuracy, and it is important to find the optimal balance between the two.

Overall, our work highlights the importance of considering model complexity when developing deep learning models. By using our formula, researchers and practitioners can evaluate the quality of their models in terms of both accuracy and efficiency, and make informed decisions about the trade-offs between model complexity and accuracy.

Chapter 7

Future Work

In our experiments, we have developed a test bench to evaluate the performance of various deep learning models and architectures, with a focus on achieving high accuracy while minimizing computational complexity. Moving forward, we plan to continue conducting experiments to optimize our recognition models, exploring changes to the architecture, training techniques, and validation methods to identify the most effective approach. We will also refine our evaluation bench to make it more scalable and flexible, allowing us to evaluate a wide range of model architectures and input data.

While FLOPs is a commonly used metric for measuring the size of deep learning models, it has limitations and other measures such as number of parameters, memory usage, and inference time should also be considered. Moreover, calculating accuracy and rank scores using only the mean result is not sufficient and we need to also take into account the standard deviation and variance to obtain a more accurate representation of the model's performance.

Another interesting approach would be to dynamically learn the weights associated with the MDL criterion function using techniques such as genetic algorithms to optimize model performance. By continuing to explore new approaches and techniques, we can make significant progress toward creating more effective and reliable recognition systems.

Bibliography

[1] Tianqi Chen, Itay Hubara, Zhe Xu, and Yoshua Bengio. Compressing deep neural networks with the hashing trick. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

[2] Xiaopeng Dong, Xiaolin Huang, Lei Yang, and Qionghai Dai. Attentive multi-task evaluation for hyper-parameter optimization. *IEEE Transactions on Image Processing*, 30:1368–1380, 2021.

[3] Ranasinghe M Gondara and Robert C Holte. Deep learning via minimum description length. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[4] Ranasinghe M Gondara and Robert C Holte. Sparse deep learning via stochastic compressive sensing. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2016.

[5] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[6] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7122–7131, 2018.

[7] Adam R Kosiorek, Alex Bewley, Ingmar Posner, and Stephen J Roberts. Learning deep architectures using information criteria to discover parsimonious models. In *Advances in Neural Information Processing Systems*, 2018.

[8] Liam Li, Kevin Jamieson, Genevieve DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2442–2451. JMLR. org, 2018.

[9] Xiyu Li, Zhuang Chen, Kun Zhang, Yi Yang, and Jun Zhang. Regnorm: Regularization of deep learning via unsupervised attention across neural network layers. *arXiv preprint arXiv:2004.05868*, 2020.

[10] Yuanqing Lin and Bir Bhanu. Genetic algorithm based feature selection for target detection in sar images. *Image and Vision Computing*, 21(7):591–608, 2003.

[11] Yuanqing Lin and Bir Bhanu. Object detection via feature synthesis using mdl-based genetic programming. *IEEE Transactions on Systems, Man and Cybernetics Part B*, 35(3):538–547, 2005.

[12] Chunxu Lu, Zhiguo Yang, Wenya Wang, Yue Liu, Zhe Lin, and Lei Zhang. Enhancing multi-objective evolutionary neural architecture search with training-free pareto local search. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3519–3528, 2019.

[13] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 459–468, 2018.

[14] Yiming Yang, Liqun Chen, and Yiqing Yu. Evaluating the optimization performance of augmented memory networks with minimum description length. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4999–5006, 2019.