Task-Driven Adaptation of Deep Learning Architectures

By

SIYU QI

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical and Computer Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Zhi Ding, Chair

Bernard C. Levy

Zhaojun Bai

Committee in Charge

2023

Siyu Qi
914462899

Abstract

Task-Driven Adaptation of Deep Learning Architectures

Deep learning (DL) is one of the widespread frameworks for solving problems from both thriving areas such as image recognition and long-standing areas such as salinity level estimation for water planning. Applying DL neural network models to establish rules from automatic data analysis and unsupervised feature extraction and generalize to unknown data, domain knowledge and human experience can be fused to adapt the existing models for specific DL tasks to obtain boosted performance and better interpretability. This dissertation aims to address some existing obstacles in practice with a focus on two themes: image compression and recognition in band-limited networks and water salinity modeling in Sacramento-San Joaquin Delta (Delta), California. When considering the deployment of learning-based image classifiers in distributed wireless Internet of Things (IoT) systems like remote camera deployment, effective feature extraction is critical for efficient bandwidth utilization. In the first part of this dissertation, we develop task-aware image compression codecs for edge nodes in the IoT systems.

Massive deployment of low cost IoT devices in various networked artificial intelligence must overcome limited computation and storage capacities of sensor terminals, thereby motivating studies on developing image codec to efficiently encode source images for transporting over bandwidth-constrained network links to cloud nodes responsible for complex computations. However, traditional standardized codecs such as JPEG were designed for human end users based on subjective tests, not for machine learning. Under limited storage and transport bandwidth, we aim to adapt the popular JPEG codecs for joint image compression and classification. Our novel end-to-end deep learning framework can optimize widely deployed JPEG codecs to improve classification accuracy over current JPEG settings. This integrative framework simplifies training and classification by directly leveraging the stored or received JPEG images in the frequency domain during learning to bypass the unnecessary step of image reconstruction.

On the other hand, neural-network-based image compression codecs, which usually provide a more promising performance, also play a critical role in remote camera applications. Yet, there exists several practical challenges in distributed DL over band-limited channels. Specifically, many IoT systems consist of sensor nodes for raw data collection and encoding, and servers for learning and inference tasks. Adaptation of DL over band-limited network data links has only been scantly addressed. The second challenge is the need for pre-deployed encoders being compatible with flexible decoders that can be upgraded or retrained. The third challenge is the robustness against erroneous training labels. Addressing these three challenges, we attach a side branch to the vanilla auto-encoder models and develop a hierarchical learning strategy to guide the encoder via this side path. Experimental results show that our hierarchically-trained models can improve link spectrum efficiency without performance loss, reduce storage and computational complexity, and achieve robustness against training label corruption.

Next, we identify another important challenge which is how to effectively train such distributed models when the training samples undergo some distortive transformations and the connecting channels have limited rate/capacity. Our goal is to optimize DL model such that the encoder latent requires low channel bandwidth while still delivers transform-invariant feature information for high classification accuracy. This work proposes a three-step joint learning strategy to guide encoders to extract features that are compact, discriminative, and amenable to common augmentations/transformations. We optimize latent dimension through an initial screening phase before end-to-end (E2E) training. To obtain an adjustable bit rate via a single pre-deployed encoder, we apply entropy-based quantization and/or manual truncation on the latent representations. The proposed trained models also exhibit robustness to such latent quantization and truncation.

In the second part of this dissertation, we turn to the DL applications where training data is insufficient. Reliability of the DL models usually comes with the pre-requisite of massive annotated training data. For example, the generalization capability of the DL models discussed above relies on tens of thousands of training samples. However, acquisition of task-specific annotated data can be costly in terms of experimental resource, human labor and user privacy, which calls for the few-shot learning (FSL) paradigm where models learns data representations effectively from limited number of samples. In addition, accurate label information may conflict with the intrinsic features in data, hence become misleading when training the embedding extractors. To alleviate model dependence on labeled data and address the common overfitting problem of FSL in computer vision, again, we integrate the side path in the encoder to ensure linear discriminative embeddings extraction. Moreover, to mitigate the disagreement between categorical labels from the classifier end and underlying patterns from the encoder side, we propose to incorporate coarse-grained instead of fine-grained labels into the embedding regularizer term. The proposed regularizer reduces overfitting and improves test accuracy over E2E CE training or its fine-grained version, especially for deeper models which are more likely to overfit. This regularizer works better when there is less manual intervention and more randomness in coarse label assignment, which in turn supports our statement that the inherent discriminative characteristics in data may not be well detected via the straightforward E2E label-based training.

In the third part of this dissertation, we shift to a conventional field of water salinity modeling. Domain-specific architectures of multi-layer perceptron (MLP) artificial neural networks (ANNs) have been developed as computer emulators for a commonly used process model, the Delta Simulation Model II (DSM2), for fast salinity level estimation at key monitoring stations in the Delta. However, achieving promising prediction results and fast inference speed at the same time can be challenging with an insufficient amount of training samples and/or the inevitable measurement noise in the observed dataset. To begin with, we propose three major enhancements to the existing ANN architecture for purposes of training time reduction, estimation error reduction and better feature extraction. Particularly, we design a novel multi-task ANN architecture with shared hidden layers for joint salinity estimation at multiple stations, achieving a reduction of 90% training and inference time. As another major structural redesign, we replace pre-determined pre-processing on input data by a trainable convolutional layer. We further enhance the multi-task ANN design and training for salinity forecasting. These enhancements substantially improve the efficiency and expand the capacity of the current salinity modeling ANNs in the Delta. Our enhanced ANN design methodologies have the potential for incorporation into the current modeling practice and provide more robust and timely information to guide water resource planning and management in the Delta.

The enhanced ANN is able to produce adequate estimation accuracy on DSM2 simulated data, but the performance degrades when being applied to field observations due to data insufficiency and noise in measurements. For further performance gain and inference acceleration, we develop novel DL models by attaching a residual shortcut path of recurrent neural network (RNN) layers to the vanilla MLP ANN architecture, called the "Res-RNNs". The proposed Res-RNNs can capture spatial variations with the main MLP path and handle temporal information with the assistance of the RNN side path, hence provides better performance than MLP models. Our work demonstrates the feasibility of DL-based models in supplementing the existing operational simulators in providing more accurate and real-time estimates of salinity to inform water management decision-making.

Overall, this dissertation reveals the potential of adapting existing DL model architectures for downstream tasks to achieve interpretable, robust and timely results in both the rising area of learning-based image recognition and the classical area of water modeling.

Acknowledgments

There are many who helped me over the course of my researching and writing this dissertation and I would like to take a moment to thank them. First and foremost, I would like to thank my esteemed advisor, Prof. Zhi Ding, for his invaluable inspiration, guidance and tutorship during the journey of my PhD degree. My research work along with this dissertation would not have been possible without his keen insights and constant support. He led me through the way of becoming an independent researcher, a responsible writer and a confident person.

I am deeply indebted to the members of my dissertation committee, Prof. Bernard Levy and Prof. Zhaojun Bai. They generously give their time and expertise to offer me precious advice toward improving my work. In particular, Prof. Zhaojun Bai offered me with constructive suggestions, without which I would not have been able to complete my research. I would also like to extend my deepest gratitude to Prof. Khaled Abdel-Ghaffar and Dr. Tariq N. Kadir, who devoted their time to serving as a member of my qualifying exam committee.

I would like to recognize the assistance that I received from all of my collaborators from the California Department of Water Resources. The completion of my dissertation would not have been possible without their support and help. I also had great pleasure of working with all my lab mates at the Broadband Radio Access Technology Lab (BRAT-Lab), especially (in no particular order), Taha Bouchoucha, Qinwen Deng, Mohammadamin Farajzadehjalali, Carlos Feres, Lahiru D. Chamain Hewa Gamage, Chih-Ho Hsu, Vincent Cong Vinh Huynh Shusen Jing Yu-Chien Lin, Xintong Ling, Zhenyu Liu, Molly Mo, Suchinthaka Prabhashwara Wanninayake Wanninayaka Mudiyanselage, Mason del Rosario, Dylan Shadduck, Igor Sheremet, Hui-Ying Siao, Jonathan Tivald, Weiwei Wang, Qing Wei, Achintha Achintha Wijesinghe, Ge Yao, Tianhang Yu, and Songyang Zhang. I very much appreciate their thought-provoking discussions and helpful technical ideas in the lab and the cherished time we spent together in social settings.

Particularly helpful to me during this time, were my cats Juice and Sisi, who brought me with countless joy, emotional support and spiritual comfort. The love Juice gave me in return for my care improved the quality of my life, especially over the pandemic. I would also like to acknowledge the support from all my friends in Davis.

Lastly, I would like to express my deepest appreciation to my parents, Yadong Qi and Huimin Zhao, for their unwavering support and constant encouragement. They provided me with endless love throughout my life, which cannot be overestimated. I must express my very profound gratitude to them for giving me the strength to chase my dreams.

The work invested into this dissertation was supported in part by the National Science Foundation under Grants No. 1824553 and No. 2002937.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Recent advances in deep learning (DL) together with the support of higher computing capabilities enables researchers to apply this growing technology to solve various complex problems among human societies in both emerging fields like data-driven image compression and classification, as well as traditional fields like salinity modeling in water supply systems. Brief diagrams of DL applications are provided below in Figs. 1.1 and 1.2.

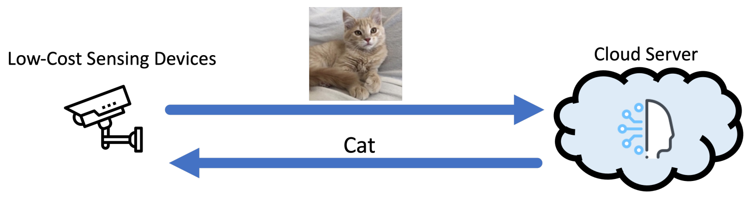


Figure 1.1: An example of a cloud-based image classification framework.

An illustration of DL-assisted cloud-based image classification technique can be found in Fig. 1.1, where low-cost sensing devices, such as surveillance cameras, capture and compress image data before sending them to a resource-rich cloud server for classification. The cloud server then decompresses and categorizes the received information and transmit the label back to the edge nodes. As existing traditional classifiers are designed for uncompressed high-quality images, proper architecture adaptation shall be applied in this scenario.

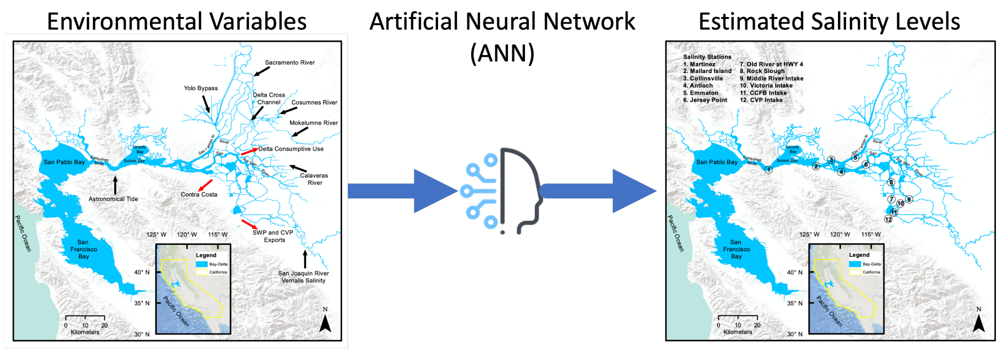


Figure 1.2: An example of a salinity estimation framework.

The diagram of a typical salinity estimation process is outlined in Fig. 1.2, where an artificial neural network (ANN) produces salinity level estimations at locations of interest based on a set of measured environmental variables. For both cases, DL has shown significant potential in approximating and reducing the dimension of complex datasets into highly compact and informative subspaces [3–9]. Hence, it has been adopted as a promising approach for big-data-based feature extraction. In order to attain rapid accurate estimation, certain enhancements can be introduced to the previous ANN architecture.

In this dissertation, we aim at solving the problems of developing effective learned feature extraction applications for both the advanced field of image compression and classification, as well as the classical field of water supply management.

1.1.1 Joint Image Compression and Classification Paradigms

In the era of Internet of Things (IoT) and cloud computing, many practical image processing applications [10–12] rely on widely deployed low-cost cameras and sensors for data collection before transmitting sensor data to powerful cloud or edge servers that host pre-trained deep classifiers. With recent developments in remote sensing technology, the sensing devices can capture higher quality image data, which is advantageous to cloud-based remote sensing image applications. However, the huge amount of information in these images poses a challenge for data transmission as most network links usually are severely band-limited and must prioritize heavy data traffic. As a result, image compression techniques become a vital topic for efficient and effective utilization of limited network bandwidth and storage resources.

Conventional Image Compression Codecs

Conventional image compression approaches can be employed for remote sensing images. JPEG [13] is one of such popular standards for lossy image compression, widely used to conserve bandwidth in source data transmission and storage. An overview of a JPEG-embedded joint image compression and cloud-based classification framework is depicted in Fig. 1.3.

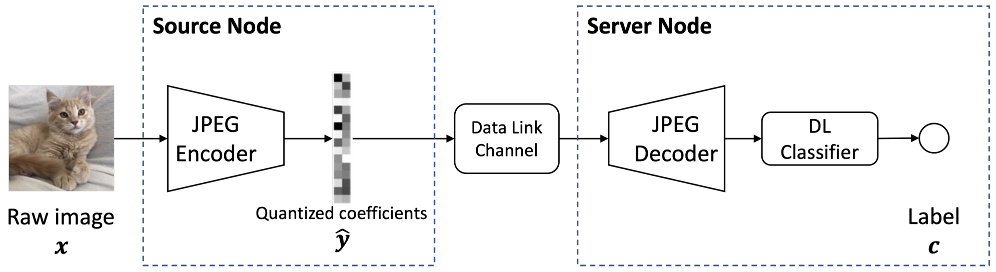


Figure 1.3: Overall framework of the joint image compression and classification technique.

At the source device end, the JPEG encoding process includes discrete cosine transform (DCT) and frequency-dependent quantization. The quantized integer coefficients $\hat{\mathbf{y}}$ are encoded via run-length encoding (RLE) and Huffman coding. Due to RLE, total bit rate of an image cannot be predicted straightforwardly [14–16]. By retaining more low-frequency data, which human visual systems are more sensitive to, the JPEG encoding achieves substantial image compression ratio with little human perception quality sacrifice. These encoded bits are then transmitted over a data link channel of limited capacity (bit rate) before decoding and recovery for various applications at the server node.

Deep-Learning-Based Image Compression Models

With recent research progress, deep learning has become a major tool in multimedia applications such as joint image compression and classification in the IoT systems. In present days, auto-encoders (AEs) [3–5] have been introduced into data compression tasks with encouraging performance.

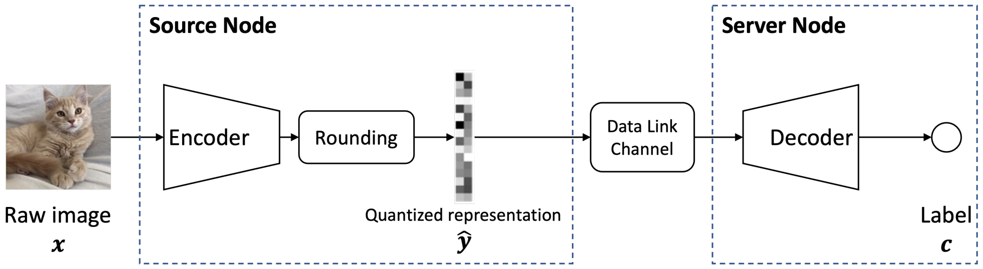


Figure 1.4: Overall framework of the joint image compression and classification technique with an auto-encoder.

An AE is a specific type of neural networks that is popular for data encoding/compression and decoding. As shown in Fig. 1.4, an AE consists of an encoder for the source node, which is responsible for compressing the input into a low-dimensional meaningful representation, and a decoder for the server node, which maps the extracted latent representation to the desired output for a specified deep learning task, such as reconstruction, segmentation or classification [6–8]. The rapid growth and broad deployment of low-cost sensing devices have inspired the develeopment of multiple variants of lightweight and efficient convolutional AEs [17–22] for networked learning frameworks.

1.1.2 Image Classification with Limited Data

Benefiting from the emergence of high speed computing devices, well-designed models and rich datasets, DL-based classification frameworks are able to overcome human intelligence in terms of durance, accuracy and processing speed. For instance, the joint image compression and classification models in Section 1.1.1 learns to preserve key features and accurately classify the images. Yet, these data-intensive DL applications usually require learning from a sufficient amount of labeled dataset to adapt to accommodate unseen data. Acquiring such training data may be costly, if not impossible, for the sake of data-gathering labor, data-labeling expertise, user privacy and safety. Straightforward approaches, such as fine-tuning models based on the limited task-specific data, can lead to severe overfitting [23]. In contrast, humans can recognize new categories at a high accuracy [24] with only a few instances utilizing their past experience. Consequently, researchers have devoted efforts in this new sub-area, Few-Shot Learning (FSL), to tackle the problem of learning the underlying pattern from a limited amount of training samples. Inspired by human development theory, the first and most obvious solution to FSL is to gain experience from other similar tasks, known as "meta-learning" or "learning to learn" [25]. Classic meta-learning usually involves two learning stages. Firstly, a meta-learner rapidly learns from several individual tasks. In the second stage, a learner acquires generic experience across the similar tasks and generalizes the accrued knowledge using task-specific information.

As one of the meta-learning approaches that has made significant progress in FSL, metric-learning-based methods [23, 26] learn discriminative instance embedding functions that map images to a common feature space, where they get classified based on a distance metric. An overall diagram is provided in Fig. 1.5. In particular, Matching Networks [26] learn with a weighted nearest-neighbor objective and measure the normalized cosine similarity between the unlabeled images (the query set) with each labeled sample (the support set), while Prototypical Networks (ProtoNets) [23] perform nearest neighbor classification by measuring the negative distance between the query image and the class prototypes (instead of every sample in the support set). Recent variants of ProtoNets [27–29] enhance its performance with a task-adaptive metric [27, 29] or an image-to-class local metric [28].

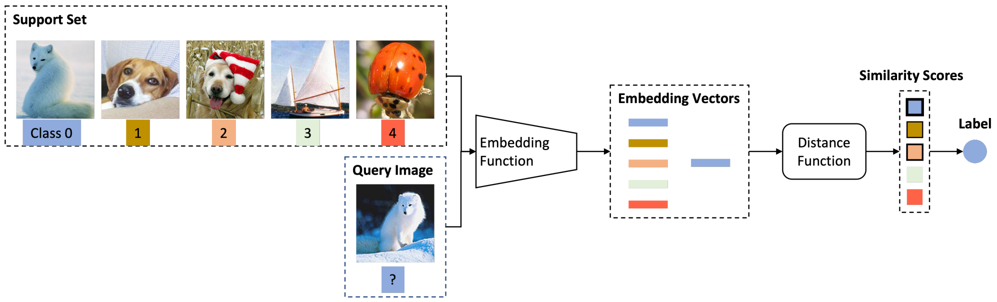


Figure 1.5: Example setup of a metric-learning based FSL technique.

1.1.3 Water Salinity Modeling

Deep learning has also been proved to be an advanced big data analysis tool to solve complex meteorological and hydrological phenomena in traditional fields such as water demand [30], leak detection [31], water system control [32], and modeling salinity in groundwater [33], soil [34], rivers [35], oceans [36], and estuarine environments [37, 38]. Salinity management is the keystone of water resources management in estuarine environments due to the underlying biological significance and inherently high variations in space and time of salinity [39]. Understanding these variations and predicting their patterns under different potential future scenarios lay the foundation for informed water management decision-making. This is especially true for areas with great ecological, social, and economic importance including the Sacramento–San Joaquin Delta (Delta) in California, United States.

1.2 Motivation

1.2.1 Optimizing Image Compression Algorithms for Band-Limited Networks

In this dissertation, without loss of generosity, we target at image compression codec optimization for the image classification task, while the proposed methods can be directly applied for other tasks. In a networked learning scenario, there is a strong need to conserve network bandwidth and local storage for remote image classification. Targeting human users, the parameters in JPEG configuration are selected according to visualization subjective tests. However, in CNN-based image classifications, naïve adoption of the lossy JPEG image encoding, designed primarily for human visualization needs, can lead to unexpected accuracy loss because the traditional CNN models are agnostic of the compression distortion. To tackle this issue, this work is motivated by the obvious and important question in distributed AI: *How to optimally (re)configure standardized JPEG for image compression to improve DL-based image classification.* Classical lossy image compression codecs are usually designed for human vision. But images compressed by these algorithms can be harder when being analyzed by a deep neural network (DNN). Direct application of traditional image classifiers, implemented for high-quality raw images, to compressed images in this scenario usually leads to sub-optimal results. Hence, proper architecture adaptation, such as a trainable DL model, is required for joint image compression and classification that can optimize the widely used JPEG codec for higher classification accuracy under the standard JPEG settings.

Complex computer vision tasks have attracted immense research efforts. Due to the huge amount of high-dimensional training data, DL-based image compression codecs are naturally more suitable than handcrafted codecs for digital image analysis and/or the downstream image classification task. When being embedded in the IoT systems consisting of cloud servers and edge nodes, the multilayer structure of DL neural networks makes them naturally suitable to be distributed separately. However, it is challenging to directly deploy traditional DL architectures on edge nodes due to their limited storage space, power supply and computational resources.

As a result, we extend the scope from the conventional manually-designed image codecs to the more flexible DNN-based trainable AE codecs. Traditional AEs are trained in an end-to-end (E2E) manner to learn an *unsupervised* mapping from the input to the latent domain, which may not be optimal for a band-limited IoT scenario, where bandwidth efficiency is an essential concern. To be specific, in distributed DL frameworks over practical rate-limited channels, the extracted latent representations to be transmitted should be both discriminative for classification and compact for compression. Moreover, the cost to upgrade an encoder can be high once it has been embedded on source nodes, so the encoder should be compatible with various decoders that may be trained separately for different objectives. Besides, as errors in training data, such as inaccurate labeling, are inevitable in realistic applications, the developed AE need to be robust against certain level of training label corruptions. The straightforward E2E training with a single cost function, typically the cross-entropy (CE) function, without regularization in the latent domain cannot address the three challenges mentioned above.

Additionally, the original data collected by IoT sensing devices usually undergo various transformations, including translation, mirroring and rotation, color jitter [40–42], etc. And the encoders on edge node shall be capable of learning the key invariant features from these distorted image data. Lastly, as the bandwidth limit may change in the IoT systems, it is also essential that the encoder can provide various compression ratios for choice. Lastly, channels can be disrupted when transmitting data and the learned latent representations are expected to be robust to such random corruptions. Although E2E training strategy empirically produces adequate performance with a black box classifier, it does not handle the five challenges existing in real-time networked learning frameworks described above.

1.2.2 Problems of Embedding Regularization in Few-Shot Learning

In the area of metric-learning based methods for FSL, how to extract discriminative embeddings that are universal for both seen and unseen classes remains the key question. Intuitively, embeddings of the same classes shall be close to each other while those of different classes shall locate far away in the feature domain. It has been proposed in [29] that introducing adaptable set-to-set functions to ProtoNets can promote such effects, alleviate overfitting and achieve performance gain. Moreover, it has been shown in [43] that integrating an auxiliary loss term in [29] to ensure the compactness and discriminability of embedding can further reduce overfitting and show accuracy improvement with no additional computational or storage cost at inference. However, there remains several problems when applying such learning guidance. Firstly, the embedding regularization term may disagree with the classification objective during training [44], leading to a local minima where neither embedding is well-regularized nor classifier is well-trained. The conflict will likely result in a relatively marginal accuracy gain, which can be observed in [43]. Secondly, as neural networks only pay attention to the task-specific patterns and ignore unrelated ones [45], the meta-information shared across tasks may get missed if it is not strongly correlated with the class labels. For example, if a learner is trained on images of cats and dogs, it may overfit to distinguishing animals and have a hard time transferring the knowledge to recognizing fish. Hence, such strict regularization through fine label information can be less optimal for cross-task generalization in FSL. Thirdly, hand-crafted fine-grained labels are expensive even in FSL problems as they usually require expertise to implement while coarse-grained labels are more accessible. To mitigate the three problems, we consider coarse-grained labeling information for embedding regularization as a relaxation for fine labels.

Data augmentation is another line of methods to prevent overfitting in FSL, where the supervised information in training set gets enriched using prior knowledge [46]. For example, one can introduce variance using manually-designed transformations such as noise, flipping, cropping and rotation [40–42]. Designing the augmentation rules requires intensive domain knowledge and they are data-specific and hard to be broadcast to other datasets. In contrast, our coarse-label-guided FSL approach is applicable to any existing dataset or task. As an exploratory research, we also investigate and compare the effects of integrating different augmentation rules and/or our proposed coarse-label-guided approach to the FSL backbone.

1.2.3 Enhancing Salinity Estimation Models

Previous works [2, 47–49] have investigated applying ANNs in salinity modeling in the Delta. However, these previous works introduce training a single-task learning multi-layer perceptron (MLP) [50] ANN for each monitoring station of interest with the same set of input variables, which is less efficient in terms of both training time and inference speed. Therefore, more efficient architectures and/or training strategies shall be investigated to improve the processing speed while maintaining a valid performance.

The recurrent neural network (RNN) is a type of ANN model well-suited for handling sequential data. Previous studies provide evidence that RNNs can be applied to the time series data in salinity modeling [51–54]. Specifically, the Long Short-Term Memory (LSTM) [55] and Gated Recurrent Unit (GRU) [56] architectures have shown special potential in simulating variables by keeping a memory of their predictors [49]. However, these RNN models have two major disadvantages. Firstly, in order to keep track of sufficient memory, they often contains more parameters than non-recurrent models. Secondly, the iterative computations in RNNs process in an autoregressive manner and cannot run in parallel, which naturally reduces the training and inference speed of RNNs. These facts motivate us to seek a modification in the architecture to reduce the complexity of these powerful RNN models for faster training and inference of the specific task.

1.3 Objectives and Approach

In the first part of this dissertation, we focus on developing low-cost image compression algorithms for deployment on edge devices.

1.3.1 Learning-Based Joint Image Compression and Classification

Conventional Image Compression Codecs

To improve the widely-used JPEG standard for image transmission at the cloud, the DL-based model should take both the image compression performance and the neural network's classification accuracy into consideration during training. We formulate this dual-objective problem as a constrained optimization problem which maximizes classification accuracy subject to a compression ratio constraint. We fuse the trainable JPEG compression blocks and JPEG decoding blocks together with the trainable CNN classifier in our end-to-end learning model. By incorporating the traditional JPEG codec in a neural network model and making certain parameters of it trainable, we can jointly optimize/configure the JPEG codec and the DL model to reduce the classification accuracy degradation caused by lossy compression at a given bandwidth, compared with the original JPEG compression algorithm.

Deep-Learning-Based Image Compression Models

In this dissertation, we propose to tackle the challenges by leveraging a multi-phase hierarchical learning concept by an information-theoretic principle of Maximal Coding Rate Reduction (MCR^2) [44] through a side branch attached to the AE. With the proposed training strategy, we assign the sub-task of efficient feature extraction to the encoder and the sub-task of classification to the decoder and train them separately. In addition, to make the encoder of the AE less complex and more feasible for cyber-deployment on sensing devices, we propose a encoder pruning strategy based on Fisher's linear discriminative analysis (LDA) [57]. Experimental results show that the AE models trained by our training strategy provide robustness to filter pruning compared with E2E CE training.

Inspired by hierarchical training, we suggest to incorporate self-supervised learning in this training strategy designed above and propose a three-step hierarchical training method. We observe that with the self-supervision-enhanced training strategy, the AE models manage to maintain a satisfying accuracy under multiple transformations or augmentations tested, yielding a better overall rate-accuracy performance with entropy-based quantization applied and are more robust to random dropout in latent representations, potentially caused by channel disruptions, than baseline E2E CE-trained models.

1.3.2 Weakly-Supervised Few-Shot Learning

In the second part of the dissertation, we target at alleviating overfitting, improving model generalization capability, reducing data-collection labor cost for FSL. Particularly, following [43], we add a side branch and apply an MCR^2 [44]-guided embedding regularization term to the cost function of the enhanced ProtoNet in [29] to achieve the dual objectives of efficient embedding extraction and accurate classification. In this work, we assign coarse labels, based on either a random or a hand-crafted hierarchy, to the training set and only use the coarse-grained grouping information in MCR^2 loss for embedding guidance. Our experimental outcomes suggest that such weakly-supervised MCR^2 guidance is as powerful as vanilla supervised version in [43] on shallower and thinner models. On deeper and wider models, which are more prone to overfitting, weakly-supervised MCR^2 guidance can lead to further accuracy improvement than [43]. Additionally, we explore the effectiveness of common data augmentation methods versus our proposed method to better understand how the principal of MCR^2 leverage the discriminability in embedding mapping and classification accuracy. Overall, applying data augmentation alleviates overfitting at a cost of degradation in both training and test accuracies. Random rotation, which may confuse the neural networks as the images' internal patterns are completely manipulated pixel-wisely, appear to interfere the classifier's capability of extracting underlying discriminative features the most. For this reason, the proposed weakly-supervised MCR^2 guidance strategy does not establish improvements under image rotation either. However, with other augmentation methods including grayscale conversion, color jitter and flipping, our weakly-supervised MCR^2 guidance strategy shows the potential of reducing the performance drop caused by augmentation.

1.3.3 Deep Learning on Time Series Data for Delta Modeling

In the third part of the dissertation, we aim at enhancing current models as well as developing novel architectures for salinity estimation in the Delta.

Enhancing Multilayer Perceptron (MLPs) for Salinity Modeling

Motivated by the fact that salinity levels at multiple monitoring stations are affected by the same hydrological measurements within the same regional ecosystem, we propose that these domain-specific ANNs can be enhanced by viewing salinity estimations at monitoring stations as an integrated multi-task learning (MTL) problem. In this way, the key knowledge extracted by intermediate layers can be shared by developing and training a single MTL ANN that generates salinity estimations for all stations simultaneously. The proposed MTL ANN is much more efficient and outperforms baseline single-task learning ANNs for most stations.

Enhancing Recurrent Neural Networks (RNNs) for Salinity Modeling

Encouraged by the great success of ResNets [58] in image recognition, we adapt the idea of residual learning to overcome the two aforementioned obstacles. In this dissertation, we devise the novel Res-LSTM and Res-GRU models which are less complex than the vanilla LSTM or GRU models tested, where the baseline MLP ANN developed in our previous work [59] is considered as the main branch and a simplified LSTM or GRU layer is attached as a shortcut path. Experimental results reveal that the Res-RNN models produce better salinity estimation and forecasting performance than the baseline MLP ANNs while introducing only a mild increase to computational complexity.

Chapter 2

End-to-End Optimization of JPEG Standard for Rate-Limited Image Classification

2.1 Introduction

The rise of machine learning has benefited significantly from the tremendous success of deep learning neural networks in image classification and recognition. In recent years, deep convolutional neural networks (CNNs) have demonstrated successes in learning tasks such as image classification and recognition, owing to their capability of extracting image features among adjacent pixels. The emergence of residual network (ResNet)[58] further enhanced image classification without introducing extra computational complexity. In many practical AI applications involving widely deployed low cost cameras, the DL tasks are carried out on cloud servers remotely, relying on low cost sensing devices for data capturing and transmission. As most (RF) network links usually are severely band limited and must prioritize heavy data traffic, image compression techniques are vital for efficient and effective utilization of limited network bandwidth and storage resources. JPEG[13] is a highly popular codec standard for lossy image compression, widely used to conserve bandwidth in source data transmission and storage. Fig. 2.1 captures the JPEG encoding process including discrete cosine transform (DCT), quantization, and entropy encoding. JPEG partitions each image into 8×8 non-overlapping blocks before DCT. To compress the image size and conserve bandwidth, high DCT bands are less critical to visual perception and are assigned coarser quantization. These quantized integer DCT coefficients are then encoded via a combination of run-length encoding (RLE) and Huffman coding. Due to the RLE, total bit rate of an image cannot be predicted straightforwardly[14–16]. The JPEG encoding achieves substantial image compression ratio with little human perception sacrifice. These encoded bits are then transmitted over a channel of limited capacity (bit rate) before decoding and image recovery for various visual applications.

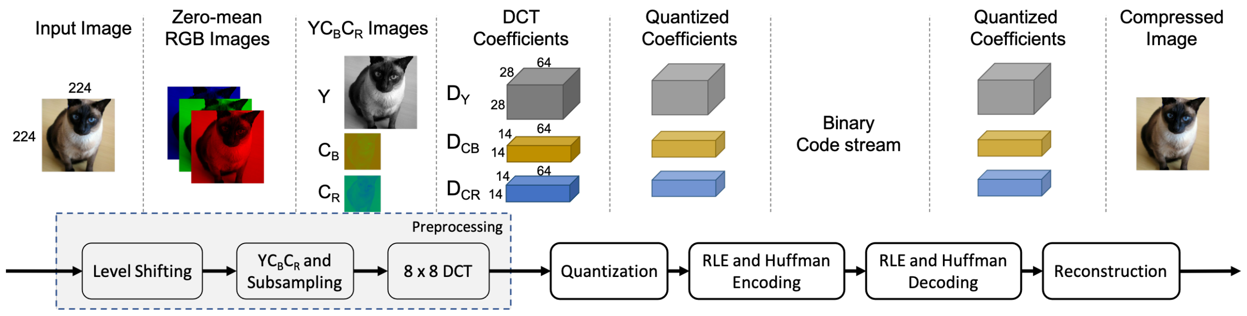


Figure 2.1: Brief Overview of JPEG Standard.

We note, however, in AI-related applications such as DL image classifications, naïve adoption of JPEG image encoding, designed primarily for human subjects, can lead to unexpected accuracy loss because the traditional CNN learning models are agnostic of the lossy JPEG processing. Our tests in Fig. 2.2 illustrate a steep decline of classification accuracy by CNN with growing JPEG compression. Clearly, JPEG compression is not designed or optimized for classification based on deep learning. Still, its widespread availability provides a strong incentive to optimize JPEG code configuration for deep learning tasks instead of its abandonment.

Our objective in this work is to develop a specialized deep learning structure that can

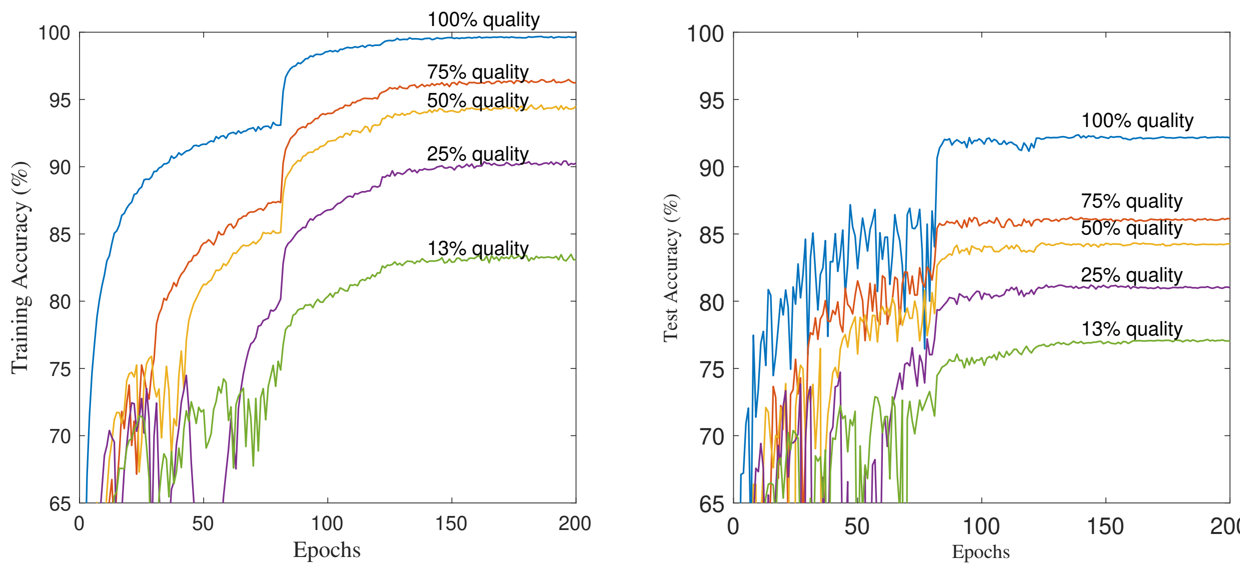


Figure 2.2: Training accuracy (left) and test accuracy (right) on 13%, 25%, 50% and 100% JPEG image quality CIFAR-10 with ResNet-50. Lower quality images correspond to higher compression ratio and lower classification accuracy.

effectively perform the joint task of image compression and classification. This dual-objective problem can be formulated as a constrained optimization which maximizes classification accuracy subject to a set degree of compression. We shall generalize the existing deep learning architectures to jointly optimize the JPEG image codec and the classifier. Specifically, We propose to incorporate trainable JPEG compression blocks and reconstruction blocks that are fully JPEG compliant into our CNN architecture. Our proposed integrative JPEG framework is capable of configuring suitable JPEG encoding parameters to achieve high classification accuracy for a given compression ratio.

We demonstrate how this end-to-end deep learning architecture can optimize JPEG codec to improve classification accuracy in comparison with the baseline JPEG settings. On high resolution images such as the ImageNet dataset, this joint framework simplifies training and classification by directly leveraging the stored or received JPEG images in the DCT domain without the unnecessary spatial reconstruction. Tests on CIFAR-10, CIFAR-100 and ImageNet datasets demonstrate improved validation accuracy for limited image sizes.

We organize the rest of the chapter as follows. Section 2.2 introduces the basics of

JPEG codec and summarizes related works. Section 2.3 proposes the novel end-to-end DL architecture. We provide experimental results in Section 2.4, before concluding and discussing potential future directions in Section 2.5.

2.2 JPEG Codec and Learning over JPEG

2.2.1 JPEG Codec in View of Deep Learning

In JPEG compression with 4:2:0 chroma subsampling, an RGB source image is first converted to YCbCr color space through linear transformations defined as:

$$\begin{aligned} Y &= \phantom{128 - {}} 0.299\, R + 0.587\, G + 0.114\, B \\ C_B &= 128 - 0.169\, R - 0.331\, G + 0.5 \quad B \\ C_R &= 128 + 0.5 \quad R - 0.419\, G + 0.081\, B \end{aligned}$$

Here R, G, B and Y, C_B, C_R represent a pixel value of the corresponding color channel. For further compression, the two chrominance channels (Cb and Cr) are subsampled by 2 both vertically and horizontally. After subsampling, each of the 3 YCbCr channels is split into non-overlapping 8×8 blocks before applying blockwise DCT.

The 2-dimensional (2-D) DCT of an image block $\boldsymbol{I}$ of size $N \times N$ with entries $\boldsymbol{I}(k, l)$ is defined by $N \times N$ block $\boldsymbol{F}$. Let $0 \leq m \leq N-1$, $0 \leq n \leq N-1$. The entries of $\boldsymbol{F}$ are

$$\begin{aligned} &F(m+1, n+1) = \\ &\quad \alpha_m \alpha_n \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} I(k,l) \cos \frac{(2k+1)m\pi}{2N} \cos \frac{(2l+1)n\pi}{2N}, \end{aligned}$$

where $\alpha_m = \sqrt{(2-\delta[m])/N}$. The DCT matrix D entries are

$$D(m+1, n+1) = \begin{cases} N^{-1/2}, & \text{if } m = 0, \\ \dfrac{2}{\sqrt{N}} \cos \dfrac{(2n+1)m\pi}{2N}, & \text{otherwise.} \end{cases}$$

The 2-D DCT and the 2-D Inverse DCT (IDCT) can be written, respectively, as matrix multiplication:

$$\boldsymbol{F} = \boldsymbol{D}\boldsymbol{I}\boldsymbol{D}^T \quad \text{and} \quad \boldsymbol{I} = \boldsymbol{D}^T\boldsymbol{F}\boldsymbol{D}.$$

The DCT representations of images in the frequency domain is capable of compacting image features with a small number of DCT coefficients which can be used for compression with little perceptible loss.

For compression, each block of the 8×8 frequency-domain coefficients $\boldsymbol{F}$ is quantized using pre-defined quantization matrices, or "Q-tables" $\boldsymbol{Q}$ with entries $\boldsymbol{Q}(j,k)$ at JPEG encoder to obtain quantized block $\boldsymbol{F}_q$ whose entries are $\boldsymbol{F}_q(j,k) = \text{round}[\boldsymbol{F}(j,k)/\boldsymbol{Q}(j,k)]$. The decoder reconstructs from the compressed block $\boldsymbol{F}_q$ and the Q-table to form Hadamard product $\boldsymbol{F}_q = \boldsymbol{F} \circ \boldsymbol{Q}$. Decoder would then use IDCT to recover spatial RGB images. Parameters in $\boldsymbol{Q}$ can be adjusted to achieve different compression levels and visual effects. JPEG standard provides two Q-tables to adjust compression loss, one for the Y channel and another for CB and CR channels.

In networked image applications, training using full resolution images would make little practical sense and would be prone to accuracy loss because only codec compressed image data are available at the cloud/edge processing node. Thus, deep learning networks should directly use compressed DCT coefficients as inputs for both training and inference instead of full resolution RGB images for training. Image classification (labeling) directly based on DCT coefficients can further reduce decoder computation during both training and inference by skipping the IDCT and potentially achieve better robustness under dynamic levels of JPEG compression.

Importantly, ResNets that were successfully developed for recognition of fully reconstructed JPEG images tend to exhibit performance loss if they are directly used on image data in DCT domain. Motivated by the need to improve image processing performance in networked environments under channel bandwidth and storage constraints, this work investigates deep learning architecture designs suitable for optimizing standard compliant JPEG configurations to achieve high classification accuracy and low bandwidth consumption by directly applying DCT input data. Our joint optimization of the JPEG configuration is achieved by optimizing both the JPEG Q-table parameters and the deep learning classifier to achieve end-to-end deep learning framework spanning from the IoT source encoder to the cloud classifier. Our experiments include tests on the lower resolution CIFAR-10, CIFAR-100 and Tiny ImageNet datasets, as well as the high resolution ImageNet[60] dataset.

2.2.2 Related Works

In terms of compression for bandwidth and storage conservation, DL architectures such as auto-encoders have been effectively trained[61–63] from end to end for lossy image compression with little degradation of image classification accuracy or perceptual quality. Previous works [1, 64] have revealed that training DL models on the DCT coefficients is possible and can benefit from the sparsity of DCT coefficients. Specifically, the authors of [1] presented a faster CNN structure with a modified ResNet-50 architecture, which learns directly in the DCT domain. In another work [65], the authors developed a joint compression and classification network model based on JPEG2000 encoding. These papers are among a number of evidences that suggest the viability of DL-based optimization of image codecs specifically for end-to-end joint image compression and classification.

Previous studies [16,66–68] have also recognized the importance of Q-tables in traditional JPEG codecs and seek to optimize them for DL-based image classification. [66,67] propose to design JPEG Q-tables based on the importance of DCT coefficients, evaluated by the relative frequency [67] or the standard deviation [66] of the coefficients. Our work is most relevant to [16,68]. Both [16,68] offer end-to-end task-aware DL models that aim to estimate a set of optimized Q-tables for each input image, where one or more entropy estimation models are pre-trained to predict the bandwidth of each image. In contrast, targeting low cost sensing nodes, our proposed model learns a single set of Q-tables for all images during training, which can be pre-configured and incorporated within the JPEG codec after training for inference tasks. This reduces the required computational power at the sensing nodes. Moreover, our proposed training model tunes the Q-tables using a regularization term in the loss function and does not need a separate entropy estimation model.

To the best of the author's knowledge, there has not been any published work on JPEG Q-table optimization for distributed by targeting low-cost sensing devices. Since JPEG continues to be a commonly used image coding methods in massive number of low-cost devices, we focus our investigation on the rate-accuracy trade-off to address the critical issue for their widespread applications in distributed learning environment.

2.3 Joint Compression-Classification DL Architectures

2.3.1 Pre-processing

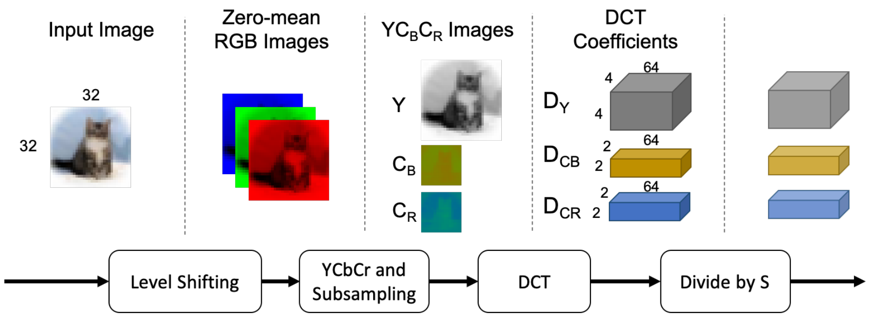


Figure 2.3: Pre-processing of input images. Image sample is from CIFAR-10 dataset and original image size is 32 $\times$ 32 pixels.

Fig. 2.3 illustrates the standardized JPEG encoding and processing steps. First, the captured image pixel values are level-adjusted by 128 before converting to YCbCr color space. Next, pixels in Cb and Cr channels are down-sampled by 2 both horizontally and vertically. All three channel data undergo 8×8 block-wise DCT. Each 8×8 DCT coefficient block is scaled down by a constant $1/S$ and re-organized into a 64×1 vector. This scaling by $1/S$ can be equivalently performed by scaling the Q-tables up by S in the JPEG encoding standard. We stack all the resulting 64×1 image vectors as inputs to the DL neural network. This process is easily amenable to augmentation in RGB color space before DCT transformation into frequency domain.

2.3.2 ResNet for CIFAR-10

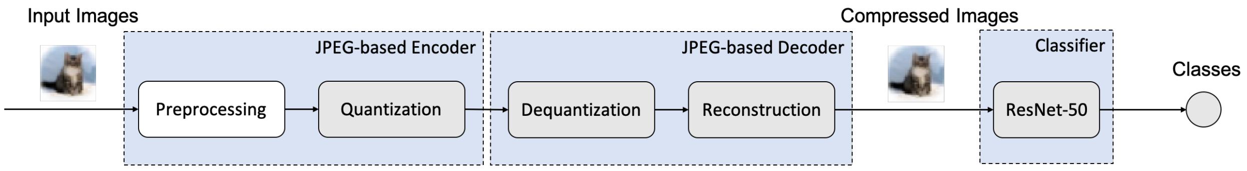


Figure 2.4: Proposed architecture for CIFAR-10, whose layers are marked as gray blocks. The detailed structure in ResNet-50 is not showed here.

Compression Layers

As illustrated in Fig. 2.4, the first hidden layer in our neural network is a trainable compression layer, similar to the "quan block" in [65]. As discussed earlier, there are two 8×8 quantization tables in JPEG standard suggested for luminance and chrominance channels, respectively. However, as stated in Annex A.3.4 and Annex B.2.2 of [13], JPEG is flexible enough to allow three distinct custom Q-tables $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$, and $\boldsymbol{Q}_3$, respectively, to each of the three YCbCr color channels, leading to 192 trainable parameters in this layer. Same as in [65], to further simplify computations, we change the quantization parameters into element-wise reciprocal of the Q-table entries in $\boldsymbol{Q}_i$ according to

$$\boldsymbol{q}_i(j,k) = \boldsymbol{Q}_i(j,k)^{-1},$$
$$\text{for } i \in \{1,\ 2,\ 3\} \text{ and } j,\ k \in \{1,\ 2, ...,\ 8\}.$$

The element-wise reciprocal matrices $\boldsymbol{q}_i$, referred to as the "compression kernels", convert element-wise division to multiplication and allow neural network to discard DCT coefficients by setting the corresponding entry $\boldsymbol{q}_i(j,k) = 0$. Smaller $\boldsymbol{q}_i$ values lead to smaller range of quantized DCT coefficients and consequently generates fewer encoded bits.

Note that the quantization layer includes a non-differentiable rounding operation$a(\boldsymbol{F}) = \text{round}(\boldsymbol{F})$, which cannot be used in a gradient-based training framework, as its activation function. To facilitate gradient based training, we mitigate this problem by substituting a smooth approximation function $\hat{a}(\boldsymbol{F}) = \boldsymbol{F}$ for the rounding function in the backward pass of backpropagation following [69].

$$\frac{d}{d\boldsymbol{z}}[\boldsymbol{z}] := \frac{d}{d\boldsymbol{z}}\boldsymbol{z}.$$

Note that we replace only the derivative and in forward pass, the rounding function operates as usual. Together, the pre-processing and the quantization layer form a JPEG-based encoder.

The dequantization layer only needs to multiply the quantized coefficients element-wise by their respective Q-table matrices $\boldsymbol{Q}_i$, $i = 1,\ 2,\ 3$ already determined in the quantization layer of the encoder. Hence, these parameters in dequantization layer are not trained and require no activation function. Together, the quantization and dequantization layers jointly form a pair of "compression layers". ·

Reconstruction Layer

As described in Section 2.2, the reconstruction layer contains 64 fixed parameters of the 8×8 DCT matrix $\boldsymbol{D}$. This layer performs IDCT via $\boldsymbol{I} = \boldsymbol{D}^T \boldsymbol{F}_q \boldsymbol{D}$ for each quantized DCT coefficient block $\boldsymbol{F}_q$ and rearranges the reconstructed blocks $\boldsymbol{I}$. Cb and Cr channels are upsampled via bilinear interpolation. Without activation function, this layer directly outputs spatial-domain YCbCr images. Together, the dequantization layer and the reconstruction layer form the JPEG decoder. This architecture allows the CNN to operate on reconstructed images on GPU and save CPU computations.

Loss Function during Training

To jointly reconfigure the JPEG parameters in compression layers and to optimize the deep learning classifier, we design the following loss function during training:

$$\mathcal{L} = \mathcal{L}_{\text{CLA}} + \lambda \mathcal{L}_{\text{Quan}},$$

where $\mathcal{L}_{\text{Quan}}$ quantifies the quantization loss and $\mathcal{L}_{\text{CLA}}$ is the loss term from the deep learning classifier, including the cross-entropy classification loss and the parameter regularization terms. Without loss of generality, we adopt ResNet [58] as the classifier.

Since there exists no easy and standard metric to quantify the JPEG compression level (or channel bandwidth), which relies on run-length encoding and Huffman coding, we propose the following surrogate loss function

$$\mathcal{L}_{\text{Quan}} = \sum_{i=1}^{3} \sum_{j=1}^{8} \sum_{k=1}^{8} q_i^2(j,k).$$

designed to control the power of quantization parameters. λ is a tunable regularization parameter to adjust compression level. A larger hyper-parameter λ leads to smaller $q_i(j,k)$ and consequently smaller range of DCT coefficients, thereby achieving higher compression ratio.

2.3.3 Wide ResNet (WRN) for CIFAR-100 and Tiny ImageNet

For CIFAR-100 and Tiny ImageNet, we propose the WRN model of Fig. 2.5. Following JPEG standard, our image pre-processing steps include level shifting, color transformation, subsampling and DCT. In the pre-processing layer, the color transformation coefficients can also be trained under simple constraints to ensure invertibility.

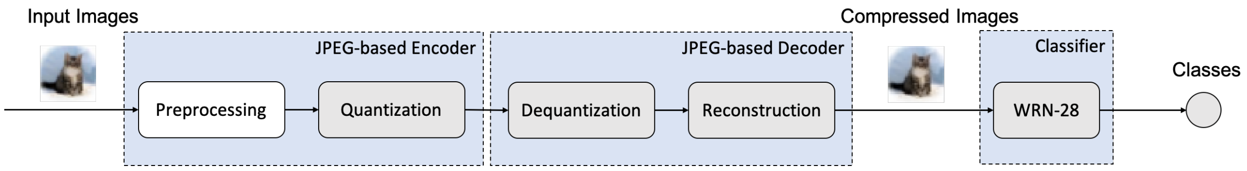


Figure 2.5: Proposed architecture for CIFAR-100 and Tiny ImageNet. The detailed structure in WRN-28 is omitted here.

Classifier

WRNs[70] achieve impressive classification performance on the CIFAR-100[71] and Tiny ImageNet[72] datasets. Without loss of generality, we adopt a 28-layer WRN as the classifier. For CIFAR-100, we set the convolutional layer width multiplier $k = 10$, same as that used in [70]. For Tiny ImageNet, we set $k = 1$ to further simplify training for very low cost IoT devices.

Loss Function During Training

However, the gradient of quantization layer activation function $a(z)$ can be large when z is close to an integer value. This large gradient value may lead to convergence issues, especially with datasets containing as many as 100 and 200 categories. To mitigate this potential problem, we replace the activation function by an accurate rounding function and manually set its gradient to 1 during back-propagation. Moreover, we supplement the loss function

$$\mathcal{L}_{\text{Quan}} = \sum_{i=1}^{3}\sum_{j=1}^{8}\sum_{k=1}^{8} q_i^2(j,k) + \lambda_1 \sum_{i=1}^{3}\sum_{j=1}^{8}\sum_{k=1}^{8} |q_i(j,k)|.$$

by adding a second term to promote sparsity in the quantization kernel with a weight of λ_1.

With the same base loss function term $\mathcal{L} = \mathcal{L}_{\text{CLA}} + \lambda\mathcal{L}_{\text{Quan}}$, we propose the following surrogate function as the penalty term:

$$\mathcal{L}_{\text{Quan}} = \sum_{i=1}^{3}\sum_{j=1}^{8}\sum_{k=1}^{8} \max(\boldsymbol{q}_i^2(j,k) - c, 0) \\ + \lambda_1 \sum_{i=1}^{3}\sum_{j=1}^{8}\sum_{k=1}^{8} |\boldsymbol{q}_i(j,k)|.$$

where c and λ_1 are tunable hyper-parameters. The ℓ_1 loss term promotes sparsity whereas the ℓ_2 loss term regulates the compression kernels. The hyper-parameter c acts as a constraint on the squared magnitude of q_i, shall be appropriately selected based on the values of λ and λ_1. Larger λ and λ_1 and a smaller c leads to higher compression ratio and lower classification accuracy. We propose the current form of the surrogate penalty function after testing both logarithm and sigmoid functions without witnessing performance benefits.

2.3.4 Modified ResNet for ImageNet

For ImageNet[60], we adopt the same pre-processing steps and quantization layer from 2.3.3 and utilize the Deconvolution-RFA architecture in [1] as the classifier. As demonstrated in Fig. 2.6, the quantized DCT coefficients of Cb and Cr channels are augmented to the same spatial size as Y channel by two separate transposed convolutional layers. Then the three channels are concatenated to form the input of the deconvolution-RFA model. Considering the higher complexity of this model, we suggest that a single ℓ_2 regularization for compression kernels be sufficient for optimizing the quantization parameters, with quantization loss term of

$$\mathcal{L}_{\text{Quan}} = \sum_{i=1}^{3}\sum_{j=1}^{8}\sum_{k=1}^{8} \boldsymbol{q}_i^2(j,k).$$

2.3.5 Implementation

We test the learning framework in Fig. 2.4 with CIFAR-10 and test the framework in Fig. 2.5 with CIFAR-100 and Tiny ImageNet datasets.The source images in CIFAR-100 and CIFAR-

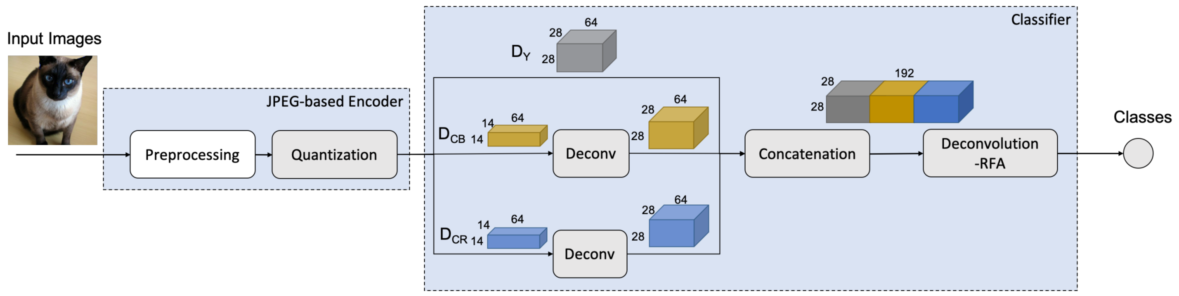


Figure 2.6: Architecture of our proposed model for ImageNet. The detailed structure of Deconvolution-RFA[1] is omitted here.

100 datasets are of 32×32 pixels, while images in Tiny ImageNet are of 64×64 pixels. After reconstruction, the reconstructed images as inputs to ResNet-50 or WRN-28, are also of size 32×32 or 64×64.

Input data are first transformed from RGB to a different color space by the color conversion, then processed by the compression layers. The two compression layers share the 192 trainable parameters in the compression kernels, which are all initialized to 1. The randomly-initialized ResNet-50 or WRN-28 classifiers are trained jointly with the parameters in compression kernels, as well as the color transformation coefficients if needed, using Adam optimizer[73] with a batch size of 100. The training proceeds alternatively: the classifier trains for 2 epochs while JPEG-based layers are frozen, followed by the color transformation and compression layers train for 1 epoch while freezing the classifier. The training takes 150 such alternations. For ResNet-50 with CIFAR-10, the learning rate starts from 0.001, and is scaled by 0.1, 0.01, 0.001, and 0.0005, at epoch 80, 120, 160, and 180, respectively. The training takes 200 epochs. For WRN-28 with CIFAR-100 or Tiny ImageNet, the learning rate starts from 0.05, and is scaled by 0.1 and 0.01, at epoch 100 and 200, respectively.

We implement the higher complexity learning framework in Fig. 2.6 with ImageNet dataset in which the images are of 224×224 pixels. Similarly, color transformation coefficients are initialized with the JPEG standard and all 192 quantization parameters are initialized to 1. The color transformation coefficients, compression kernels, and Deconvolution-RFA classifier are trained from end-to-end by using Adam optimizer with a batch size of 32. The learning rate of color transformation coefficients and compression kernels starts from 1×10^{-8} while that for other parameters starts from 0.001. Both learning rates are scaled by 0.1, 0.01, and 0.001, at epoch 30, 60 and 80, respectively. The training takes 90 epochs.

2.3.6 Customize Huffman Coding

Annex K of JPEG standard [13] offers a set of default Huffman tables suitable for most visual applications. However, with trained Q-tables in our model, the original JPEG Huffman tables may yield lower compression ratio because the quantized DCT coefficients are impacted by the new Q-tables. Thus, we also customize Huffman tables for better bandwidth saving. For each set of learned Q-tables, we randomly choose 50k images from the corresponding training set and generate Huffman tables accordingly. The bandwidth of the validation data set is measured using learned Q-tables and the generated Huffman tables, combining the principles of RLE and Huffman coding.

2.4 Experiments

Our experiments are conducted on Keras [74] and TensorFlow backend [75]. We utilize the two metrics to evaluate perceptual quality of images: peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) index. We test the proposed joint compression and classification (JCC) frameworks on four datasets: (a) CIFAR-10 dataset based on 50k training images and 10k test images belonging to 10 categories; (b) CIFAR-100 dataset based on 50k training images and 10k test images belonging to 100 categories; (c) Tiny ImageNet dataset based on 100k training images and 10k validation images belonging to 200 categories; (d) ImageNet dataset based on approximately 1.3M training images and 50k test images belonging to 1000 categories.

2.4.1 JPEG Standard Baseline

We first present baseline results, in which images are compressed using standard JPEG algorithm with 4:2:0 chroma subsampling. We initialize compression kernels $\boldsymbol{q}_i$, $i = 1,\ 2,\ 3$ using the Q-tables given in JPEG standard. In this baseline scenario, we only adapt the classifier parameters during training by fixing. the parameters in pre-processing, quantization, and dequantization.

For CIFAR-10 and CIFAR-100, we consider 11 and 9 different JPEG image qualities between 12.5% and 100%, respectively. For Tiny ImageNet, we select 4 different image qualities between 10% and 80%. For ImageNet, our experiments consider 5 different image qualities between 12.5% and 100%. The classification results are shown in Figs. 2.7, 2.8 and 2.9. These baseline results reveal that the classification accuracy correlates positively with image qualities and average image bandwidth (rate).

2.4.2 Joint Compression and Classification (JCC)

In the case of JCC training, we freeze the color transformation coefficients to train and optimize the compression kernels and the classifier. For CIFAR-10 dataset, using $S = 10$ defined in Section 2.3.1, we select 10 values of λ between 1×10^{-7} and 1.6×10^{-1}. Additionally, we use $S = 1,\ \lambda = 1 \times 10^{-9}$, and $N = 1024$ to achieve the size of 1.82 KB per image, which is close to 100% quality JPEG images. The JCC classification results under trainable compression are given in Fig. 2.7. Compared with the baseline JPEG, for image sizes between 0.85 and 1.3 KB, our JCC model clearly achieves higher classification accuracy while using similar image bandwidth as the baseline. On the other hand, when the available image size becomes too small (e.g., below 0.85KB) or too large (e.g., above 1.3 KB), we achieve similar classification performance as the JPEG baseline. Overall, the PSNR and SSIM of JCC-optimized compression kernels are higher than JPEG standard ones at the bandwidth between 0.9 KB and 1.2KB per image, and are similar otherwise.

For CIFAR-100, using $S = 1$, we selected 8 values of λ from 1×10^{-6} to 5 with $\lambda_1 = 1$

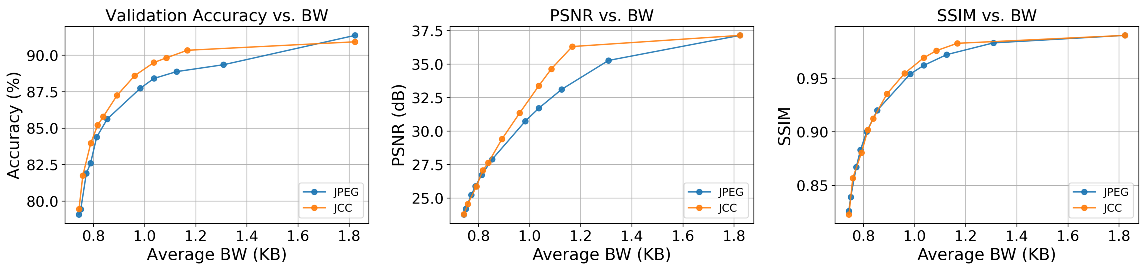


Figure 2.7: Comparison between JCC and JPEG standard on CIFAR-10 dataset with ResNet-50 Model, with respect to classification accuracy, PSNR and SSIM versus average bandwidth.

and $c = 0.01/\lambda$. The classification accuracy, PSNR and SSIM results can be found in Fig. 2.8. Compared with the JPEG baseline, JCC achieves clear improvement of up to 2.4% in accuracy at bandwidth between 0.75 and 1.5 KB per image. Meanwhile, the PSNR and SSIM of JCC-compressed images are similar to JPEG-compressed images.

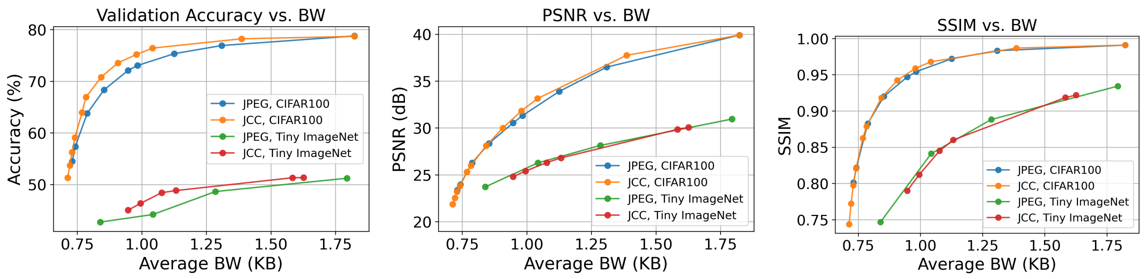


Figure 2.8: Comparison between JCC and JPEG standard on CIFAR-100 and Tiny ImageNet datasets with WRN-28 Model, with respect to classification accuracy, PSNR and SSIM versus average bandwidth.

For Tiny ImageNet, we select 6 values of c from 5×10^{-3} to 0.8. As shown in Fig. 2.8, when comparing with the JPEG baseline, we observe accuracy gain of up to 4% by the proposed JCC model at low bandwidth between 0.9 and 1.6 KB per image while maintaining similar visual quality. Overall, the PSNR and SSIM of JCC-optimized and JPEG standard quantization tables are similar.

For ImageNet, we consider 9 values of λ between 25 and 100. The resulting top-5 classification accuracy, PSNR, and SSIM are given in Fig. 2.9. For encoding rates below 11 KB per image, the JCC model outperforms the baseline by up to 3.7% in terms of classification accuracy. For bandwidths above 11 KB per image, the classification accuracy difference between JPEG and JCC is quite insignificant. Furthermore, PSNR and SSIM of JCC-compressed images outperform those of standard JPEG encoded images.

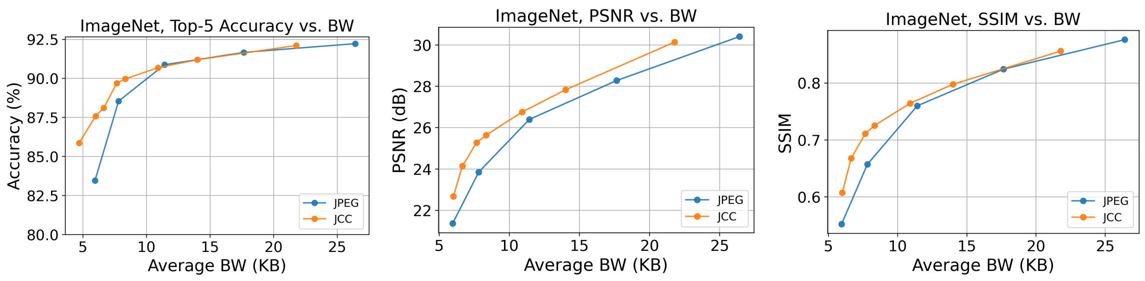


Figure 2.9: Comparison between JCC and JPEG baseline on ImageNet dataset with Deconvolution-RFA Model, with respect to top-5 classification accuracy, PSNR and SSIM versus average bandwidth.

From these experimental results, we observe that the JCC model can effectively optimize the JPEG compression kernels for better rate-accuracy trade-off, especially at moderately image bit rates. It is intuitive that the performance edge of JCC diminishes for very high image sizes because most image features can be preserved when given sufficient number of bits. JCC and JPEG no longer need to delicately balance the rate-accuracy trade-off. Similarly, for very low image sizes, very few bits can be used to encode vital information in DCT coefficients. Hence, the encoders have less flexibility to further optimize the rate-accuracy trade-off, thereby making it difficult for even the JCC model to find better parameter settings.

2.4.3 Ablation Study: JCC and Color Transformation

In JPEG standard, images in RGB color space are first converted to YCbCr space before being compressed, through the linear transformation defined as:

$$\begin{aligned} Y &= \phantom{128 - {}} 0.299\,R + 0.587\,G + 0.114\,B \\ C_B &= 128 - 0.169\,R - 0.331\,G + 0.5\,B \\ C_R &= 128 + 0.5\,R - 0.419\,G + 0.081\,B \end{aligned}$$

where Y is the luminance component and Cb and Cr are chrominance components. The YCbCr color space is more suitable for efficient compression because the Y channel carries the most detailed luminance information while Cb and Cr channels carry less information and can be compressed via down-sampling. Theoretically, the coefficients in the color space conversion equations impact the compression ratio and can be tuned as long as the conversion is invertible. To ensure the conversion is invertible, we set the parameter optimization constraints as:

$$\begin{aligned} &\theta_{R,Y} + \theta_{G,Y} + \theta_{B,Y} = 1 \\ &\theta_{\alpha,Y} \geq 0 \text{ for } \alpha \in \{R, G, B\} \\ &\theta_{B,C_B} = 0.5 \\ &\theta_{R,C_B} + \theta_{G,C_B} = -0.5 \\ &\theta_{R,C_B} \leq 0, \theta_{G,C_B} \leq 0 \\ &\theta_{R,C_R} = 0.5 \\ &\theta_{G,C_R} + \theta_{B,C_R} = -0.5 \\ &\theta_{G,C_R} \leq 0, \theta_{B,C_R} \leq 0 \end{aligned}$$

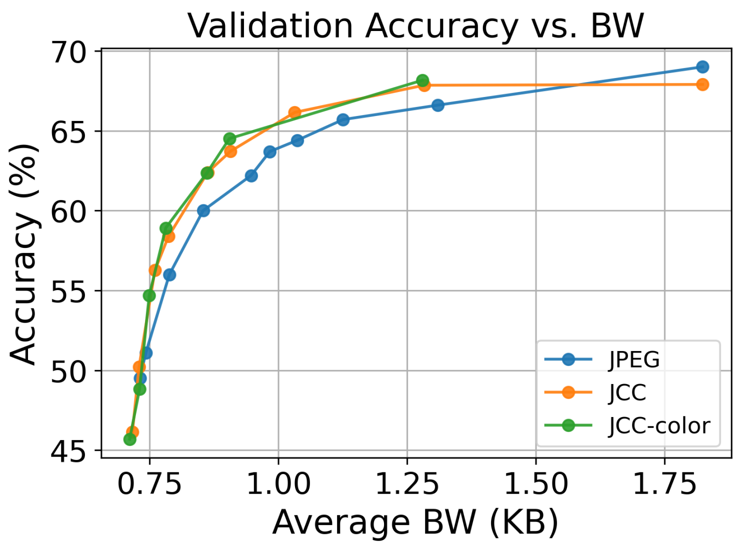


Figure 2.10: Comparison between JCC, JCC-color and JPEG baseline on CIFAR-100 dataset with WRN-28 Model, $k = 1$ with respect to classification accuracy versus average bandwidth.

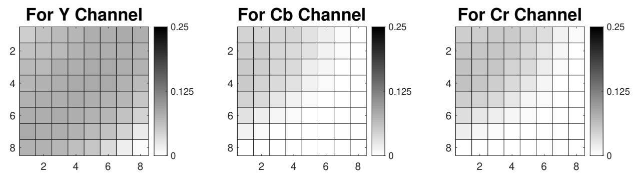
where $\theta_{\alpha,\beta}$ refers to the coefficient connecting color channel α and color channel β.

We examine the performance with WRN-28 model with CIFAR-100 dataset. We initialize the color conversion coefficients according to the conversion defined in JPEG standard. When training the color conversion coefficients, we remove the linear YCbCr to RGB conversion in the Reconstruction layer in frameworks presented in Fig. 2.5. During training, we alternately freeze two sets of parameters among the compression layers, the classifier and the color conversion coefficients, and train the third one. The training takes 150 such alternations.

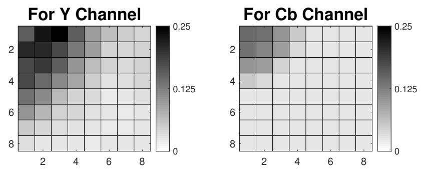
We show the results in Fig. 2.10 and use "JCC-color" to refer to the scenario where color transformation coefficients, compression kernels and classifier are all trainable.

Considering JCC and color transformation (JCC-color) results, tuning the color conversion coefficients does not establish obvious performance gain over JCC, except for 0.78 KB/image and 0.90 KB/image where we observe an accuracy improvement of 0.5% and 0.8%, respectively. Theoretically, invertible color space transformation does not lead to information loss and can be subsumed by first dense-layer in the neural network. In fact, we observe that the resulting color transform rarely changes from the initial values. Hence, there is not evidence that the existing JPEG color transformation needs further optimization.

2.4.4 Further Analysis and Discussions



(a) JCC optimized compression kernels based on CIFAR-100 for Y (left), CB (middle) and CR (right) channels, with $\lambda = 0.001, \lambda_1 = 1$ and $c = 10$.



(b) Compression kernels for Y (left), CB and CR (right) channels in JPEG standard.

Figure 2.11

Generally, our experimental results suggest that the proposed trainable JCC model can extract critical features in the DCT domain for classification among categories of CIFAR, Tiny ImageNet, and ImageNet datasets. Furthermore, the perceptual quality of images are preserved. As an illustration, Fig. 2.11 compares two sets of compression kernels under the same color scale with which WRN-28 achieves similar classification accuracy on CIFAR-100 dataset. Fig. 2.11a displays learned compression kernels for Y, CB and CR channels that yields 75.20% classification accuracy at an average rate of 0.979 KB/image. Fig. 2.11b displays the standard JPEG compression kernels yielding 73.05% accuracy and average rate of 0.983 KB/image. Qualitatively, darker grids imply low compression or higher importance of the corresponding DCT coefficient. Clearly, both encoders favor lower frequency bands. It is apparent that there are longer consecutive 0's in the zig-zag order in the end-to-end compression kernels and consequently, the correspondingly compressed DCT coefficients require fewer bits via RLE. Furthermore, the trainable model learns to discard some higher frequency DCT components as they are less critical to classification accuracy.

Together, these experimental results reveal that the proposed trainable model can further enhance the standardized JPEG codec for cloud based image classification by learning optimal quantization parameters. Practically, the proposed joint training can be easily implemented in pre-installed JPEG encoders of low-cost devices by using software updates to include more Q-table entries for different JPEG encoding sizes.

2.5 Conclusions

We present an end-to-end deep learning (DL) architecture to jointly optimize JPEG image compression and classification for low cost sensors in distributed learning systems. Results on CIFAR-100, Tiny ImageNet and ImageNet datasets demonstrate that the proposed end-to-end DL framework can be easily trained and implemented for better image compression and classification performance without perceptual quality loss in networked applications. Optimized Q-tables can be readily incorporated within deployed JPEG codecs in practice. For high resolution ImageNet datasets, we can bypass image reconstruction and learning directly in the DCT domain to further lower computation cost.

Our results show that DL models should actively take practical constraints into consideration, such as storage and network bandwidth. Future works may explore the broad appeal of this end-to-end learning principle in other bandwidth-constrained distributed DL tasks such as object detection, segmentation, and tracking.

Chapter 3

Hierarchical Optimization of Deep Learning Models for Rate-Limited Image Classification

In this chapter, we first propose two spectrum-domain DL models for joint learning-based image compression and classification in a rate-limited framework and design a hierarchical learning strategy to train the proposed models. We evaluate the rate-accuracy performance by truncating the latent representations. In addition, we test their compatibility with training label corruptions, imbalanced datasets and various re-trained decoders. Then, we combine the principle of unsupervised learning in training and prove that the robustness against various image distortions/transformations is enhanced.

Part of this chapter is reprinted, with permission, from [S. Qi, L. D. Chamain, Z. Ding. "Hierarchical Training for Distributed Deep Learning over Band-limited Networks" in *IEEE ICIP*, 2022] and followup modifications for final publication.

3.1 Deep-Learning-Based Image Compression for Transmission and Classification

With important recent advances, deep learning has become a major tool in media applications such as compression and classification in Internet of Things (IoT) systems. The rapid growth and broad deployment of low-cost sensing devices have fueled the development of lightweight and efficient convolutional neural networks (CNNs) [17–22] for offline classification on devices with low computation power. Furthermore, it has also stimulated a paradigm shift toward distributed learning that relies on networked cooperation of source and server nodes. Within such networked learning framework, cost, power, and memory efficient encoding nodes are implemented to compress and transmit data to cloud/server nodes for subsequent deep learning and analysis. Sensor (camera) data can be efficiently transformed into lower-dimensional latent representations using auto-encoders (AEs) [3–5], which are typically optimized together with a decoder/classifier at servers in an end-to-end (E2E) training process. In this chapter, we focus on AEs consisting of an encoder at the source node before the channel and a classifier [6–8] at the channel output end, which turn out to be one of the foremost feasible solutions for bandwidth-limited image compression and classification. Representative variants of such AE architectures include variational AE [6], stacked AE [7], sparse AE [8], hierarchical sparse AE [76], etc. However, these AEs, trained by a rate loss accompanied by a weighted classification loss in an end-to-end (E2E) manner, which may lead to a local minimum that neither brings down the coding rate nor optimizes the accuracy without careful manual tuning [77] and face three major challenges to be deployed in a networked environment.

Firstly, in networked deep learning, bandwidth efficiency can be just as important as the overall accuracy. It is therefore vital to tackle the important problem of lowering the encoding rate of latent feature representations without severely compromising the data processing accuracy. Moreover, after deployment of encoders on source devices, the server nodes may directly channel the data obtained by encoders to separately-trained decoders [78–80]. Consequently, the encoders that are jointly optimized with one decoder may exhibit degraded performance with another reconfigured decoder. Hence, the reconfiguration flexibility with decoders is an essential characteristic of these encoders embedded on source devices. Another major issue under supervised deep learning is its reliance on labeled data for training. In real-life applications, however, errors in manual data annotation, inaccuracy in automatic label extraction process, or data poisoning attacks [81, 82] are commonplace that lead to unavoidable erroneous data labels [83]. As a result, supervised learning tends to suffer serious performance degradation [84–86] due to corrupted data labels (or mislabels). In addition, emerging real-world image datasets usually exhibit imbalanced class distributions, which may largely limit the generalization capability of trained models and lead to poor inference performance on under-represented classes due to overfitting. For this reason, achieving robustness to certain level of corrupted data labels or imbalanced dataset is critical to reliable supervised learning models.

In the first section of this chapter, we tackle the three aforementioned practical considerations that are vital to achieving bandwidth efficiency and robustness for distributed learning. Our solutions leverage a dual-phase hierarchical learning concept by integrating a recently proposed information theoretic deep learning principle of Maximal Coding Rate Reduction (MCR^2) with demonstrated robustness to label corruption and interpretability.

In training deep learning neural networks for image classification, cross-entropy (CE) loss function has been particularly effective. Targeting the difference between probability distribution of model outputs and the true distribution, the CE loss function is equivalently a negative likelihood function. Despite its successes, CE-based neural network training does not address the two major obstacles of limited bandwidth in distributed learning systems or model robustness to label corruption.

The recent work of [44] has provided a particularly important tool for deep learning by projecting input data to latent representations in low-dimensional subspaces that are inter-class discriminative and in-class compressive based on the principle of MCR^2. In addition to offering better interpretability, MCR^2-based classifiers have also demonstrated stronger robustness against label noise. Furthermore, the lower-dimensional latent representations of MCR^2 can potentially provide valuable insight on deriving deep learning models subject to bandwidth constraints.

To this end, this work presents two MCR^2-guided CNN models for offline and cloud-based image classification, respectively. Our first contribution is a modified lightweight ResNet-18 [58] hierarchically trained by the MCR^2 loss and the CE loss. It can be pruned to further reduce the computational complexity with little loss in accuracy, making it an offline image classifier compatible for edge devices. Our second contribution is the design of an auto-encoder learning architecture for cloud-based classification. Leveraging the MCR^2 principle, we introduce a novel hierarchical learning strategy for robust and interpretable image classification through side-channel monitoring. Specifically, we guide deep learning network training with MCR^2 loss from an auxiliary side branch, together with CE objective function in the main input/output path. Moving beyond the traditional training based on loss function superposition, this MCR^2-guided hierarchical learning method achieves the dual objectives of efficient compression and accurate classification.

With the three obstacles solved by the proposed training strategy, there still exist some additional challenges and requirements to be addressed. Firstly, the AEs rely heavily on the sizable training data to learn. For robustness in real life, training images may undergo various distortive transformations, such as translation, mirroring, rotation, color jitter [40–42], etc. Therefore, encoders shall compress input data for preserving key information for classification invariant for distortions. We handle this challenge by extending the proposed training strategy with self-supervised learning (SSL) algorithm, under the guidance of the principle of MCR^2, incorporated with supervised learning.

As discussed earlier, in networked learning, bandwidth is an essential constraint to be considered by the encoders. As the transmission band limit can change in real-time, the encoders should provide the flexibility of various coding rates. We address this objective through two different approaches: entropy-based quantization and manual truncation. Entropy-based quantization can be easily applied to any pre-deployed encoder directly without further optimization. By adjusting a scalar, the average coding rate of the latent representations can be customized in accordance to the band limit. The other method, manual truncation, requires fine-tuning and updating the deployed encoders and decoders but can provide greater bandwidth reductions while preserving the classification accuracy compared with entropy-based quantization. These two methods can be used either individually or together on any pre-deployed AE.

The dimension of an AE's bottleneck layer is an essential parameter that impacts the compression ratio and classification accuracy. However, the optimal latent dimension or the intrinsic dimensionality of data, varies between image datasets [87–89]. For example, a dataset with more categories and larger image sizes usually consumes more bandwidth and needs a higher latent dimension. As a result, expertise and experience is required to determine the optimal latent dimension when implementing an AE. We tackle the third issue via an initial CE-training screening process.

In brief, we implement a supervised and self-supervised joint learning approach guided by the principle of MCR^2. This approach regularizes the encoder to extract linear discriminant representations (LDRs) that are in-class compact, between-class discriminative and consistent to various augmentations/transformations. In addition, we propose two methods to obtain continuous latent bit rate via a single AE model: entropy-based quantization and manual latent truncation. Moreover, we suggest adding an initial screening phase during the AE designing stage to select the optimal latent dimension.

3.2 Related Works

3.2.1 Auto-Encoders for Image Classification

Auto-encoders have been very successful in achieving efficient data compression and feature extraction. Recent works [3–5, 90] have proposed efficient E2E compression models based on auto-encoders for image classification. However, existing deep learning models are typically trained in an E2E manner through superposition of multiple objective functions without guidance or constraints on latent representations generated by the bottleneck layer. Another related work [7] has introduced the framework of compact and discriminative stacked auto-encoder (CDSAE), by imposing a diverse regularization and a local Fisher's discriminant regularization [57] on each auto-encoder layer such that a diverse and discriminative mapping from input data to a low-dimensional feature space can be learned. The CDSAE architecture is accompanied by a two-phase training approach where the first phase aims at low-dimensional feature extraction to emphasize in-class similarity and between-class diversity, whereas the second phase targets E2E joint training of feature extraction and subsequent classification.

Training Label Corruption For decades, numerous techniques have been explored to solve the training label corruption problem in image classifications [91, 92], such as data cleaning [93], probabilistic methods [94], loss function correction [95, 96] and model-based methods [97]. In our work, we guide the model such that it relies less on the accurate training labels but more on the intrinsic patterns of images themselves.

Imbalanced Data Distributions There are various methods developed to reduce the performance degradation on the minority classes. One common solution is to re-sample the training data to achieve a balanced distribution [98–100], either by over-sampling or generating synthetic data for the minority classes, under-sampling the majority classes [101–105], or a combination of both [106]. Popular synthetic sample generation approaches include

SMOTE [107, 108], ADASYN [109] and generative adversarial networks (GANs) [110, 111]. However, these methods either require expertise knowledge on the complete training data, extra effort to pre-process the dataset and/or a separate model/algorithm to generate synthetic training samples. Our work differs from these methods in that we do not manipulate training set.

3.2.2 Linear Discriminative Representations

In networked learning scenarios, one way to effectively transmit high-dimensional real-world data is to map such data to a low-dimensional subspaces, which shall be linear in the ideal case. For image classification specifically, we would like to seek a compact and discriminative kind of mapping. In the meantime, there exist certain criteria of the latents extracted for encoding: they should be representative of the original data and robust against noise including real-life data distortion or channel errors. The principle of MCR^2 and the objective function $\mathcal{L}_{\text{LDR}}$ proposed in [44] can be used to guide an AE to learn such mappings.

Mathematically, one can denote an image dataset consisting of N samples belonging to $[K]$ classes as $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \in \mathcal{R}^{D_{\text{in}} \times N}$, with D_{in} being the input dimension. Each image has a class label $c_i \in [K], i = 1, \ldots, N$. For conventional DL image classification, a deep classifier can learn to map the input $\boldsymbol{x}$ to its label c, by CE loss. The recent work of [44], introduced a novel loss function based on the principle of MCR^2 used to regularize deep classifiers to extract lower-dimensional latent features $\boldsymbol{z} \in \mathcal{R}^{D_z}$ that is both inter-class diverse and in-class discriminative. Works in [112] have shown that this objective can be achieved by minimizing the objective function defined in 3.1.

From an information-theoretic perspective, minimizing this loss function corresponds to maximizing the difference between the global and group-wise average coding rates of the dataset. As elaborated in [44], this MCR^2 principle generates robust, low dimensional, diverse, and discriminant latent features from the data for classification. We consider this principle to be consistent with the goal of optimizing latent dimension for transmission over rate constrained channel for distributed classification.

3.2.3 Auto-Encoders and Hierarchical Learning

AEs have been very successful in both feature extraction and compression. Recent works [3–5, 90] have proposed efficient AE compression models for image classification. However, existing DL models are typically trained in an E2E manner through naïve superposition of multiple objective functions without guidance or constraints on latent representations generated by the bottleneck layer. Another related work [7] introduced a compact and discriminative stacked AE (CDSAE), by imposing a diverse regularization and a local Fisher's discriminant regularization [57] on each AE layer such that a diverse and discriminative mapping from input data to a low-dimensional feature space can be learned.

Further, the authors of [7,77] have proposed that hierarchical training of AEs, by assigning different sub-tasks to different modules in the models can improve the overall rate-accuracy trade-off. The CDSAE architecture in [7] accompanied by a two-phase training approach, in which the first phase aims at low-dimensional feature extraction to emphasize in-class similarity and between-class diversity, whereas the second phase targets E2E joint training of feature extraction and subsequent classification. The work of [77] also suggests a Dual-Phase Hierarchical Learning (DuPHiL) strategy. The first phase of DuPHiL fixes the decoder to optimize the encoder by using the MCR^2 loss via a side path. This phase trains the encoder to map input images to a compact and discriminative latent space for efficient compression while preserving the necessary information for subsequent classification. The second phase freezes the encoder after phase one and trains the decoder by using the CE loss to learn accurate classification based on the LDRs generated by the encoder.

3.2.4 Spectrum-Domain Image Classifier

JPEG 2000 [113] is a standardized commercial image compression encoding algorithm based on multi-level 2-dimensional Discrete Wavelet Transform (2D DWT). For compression, JPEG

2000 transforms RGB images to YCbCr color space before applying 2D DWT based on CDF 9/7 mother wavelet. A level-1 DWT of each color channel generates 4 sub bands: LL band (a lowpass approximation of the original image), LH, HL and HH bands that capture increasingly high frequency features, respectively.

Since JPEG 2000 encoding is in the spectrum domain, spectrum-domain image classification can save the decoding computation [114] and improve inference speed without the need to recover RGB source images before classification. Furthermore, the deep learning classifiers at the edge servers can be further simplified by eliminating a few hidden layers to operate directly on image data in the DWT [6] or the discrete cosine transform (DCT) [115] domains. For this reason, our proposed lightweight or distributed deep learning models shall operate in the transform domain for edge devices.

3.2.5 The Information Theoretic Principle of MCR^2

In the basic classification problem, consider a set of N samples $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \in \mathcal{R}^{D_{\text{in}} \times N}$ and their class labels $\{c_1, \ldots, c_N\} \in [K]$, where D_{in} is the input data (image) size, N is number of samples in the dataset, and K is the number of classes. Typically a deep learning classifier is designed to find a direct mapping from the input vector $\boldsymbol{x} \in \mathcal{R}^{D_{\text{in}}}$ to its class label c, based on training. Mostly, various deep learning architectures have been developed according to experience and empirical tests.

Based on information theoretic foundation, the recent work of [44] suggested a new principle that drives a deep learning model to extract more diverse and discriminant lower-dimensional latent representations $\boldsymbol{z} \in \mathcal{R}^{D_z}$ from input before classification. The reduced D_z is the dimension of the latent representation $\boldsymbol{z}$. This objective can be achieved by enforcing coding rate reduction via the objective function:

$$\Delta R(\boldsymbol{Z}) \doteq R(\boldsymbol{Z}) - R_c(\boldsymbol{Z}, \boldsymbol{\Pi}) \tag{3.1}$$

where, according to [112],

$$R(\boldsymbol{Z}) = \frac{1}{2}\log\det\left(\boldsymbol{I} + \frac{D_{\text{in}}}{\epsilon^2 N}\boldsymbol{Z}\boldsymbol{Z}^T\right) \tag{3.2}$$

is the average number of bits required to encode a learned representation $\boldsymbol{z}_i$ from $\boldsymbol{Z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N\} \in \mathcal{R}^{D_z \times N}$ up to an precision bound of ϵ. When a known partition $\boldsymbol{\Pi} = \{\boldsymbol{\Pi}_j\}_1^K$ can group samples into classes, we can write "group-wise average bit rate" of $\boldsymbol{z}_i$ as

$$R_c(\boldsymbol{Z}, \boldsymbol{\Pi}) = \sum_{j=1}^{K} \frac{\text{tr}(\boldsymbol{\Pi}_j)}{2N}\log\det\left(\boldsymbol{I} + \frac{D_{\text{in}}}{\epsilon^2\text{tr}(\boldsymbol{\Pi}_j)}\boldsymbol{Z}\boldsymbol{\Pi}_j\boldsymbol{Z}^T\right) \tag{3.3}$$

bits per sample. $\boldsymbol{\Pi}_j$ is a diagonal matrix with entries "1" for samples that belong to the j-th class and "0" otherwise.

This objective function, known as maximal code rate reduction (MCR^2), is an information-theoretic measurement aimed at maximizing the coding rate difference between the global average bit rate and the group-wise average bit rate of the data set. As elaborated in [44], the MCR^2 principle generates more robust, diverse and discriminate latent features from the data for classification.

3.2.6 Linear Discriminative Analysis Based Pruning

It is well known that deep CNNs can often generate redundant intermediate features of low utility value. To effectively reduce computational complexity without significantly affecting the learning outcome, network pruning can simplify and accelerate CNNs in real-time mobile and edge applications. In the context of our work, pruning can reduce the computation complexity and link rate between the encoder and the classification servers.

Among various screening approaches for neuron/filter pruning, [116] suggests the L_1 norm of weights in CNN filters; [117] applies a particle filtering; [118] ranks the filters according to their effect on the cost. The works of [119–122] propose to rank the importance of a neuron/filter by evaluating their discriminant power. More specifically with respect to training based on the CE loss, the study of [120] proposes to add discrimination-aware loss to the learning model to strengthen discriminative power of intermediate layers.

3.3 Methods

3.3.1 Supervised Learning

MCR^2-Guided Spectrum-Domain Image Classifier

To improve image classification on resource-limited devices in practice, we adopt the modified ResNet proposed in [6] for wavelet-domain image classification, as shown in Fig. 3.1. Bypassing image reconstruction when processing JPEG-2000 encoded images, we also apply level-1 DWT to lower subband image sizes both horizontally and vertically by 2.

During training and inference, we apply the same transformations in preprocessing as used in JPEG 2000, including level shifting, color space conversion and level-1 DWT. Denote an original image size as $H \times W \times C$, where H and W respectively denote image height and width, and $C = 3$ denotes the 3 colors. After preprocessing, each input image $\boldsymbol{x}$ is reshaped to $H/2 \times W/2 \times 4C$. For each image, its DWT coefficients form a total of 12 subband channels with 1/4 of the original size. To adapt to the change in input receptive field dimension, the number of filters in the first convolutional layer grows from 16 to 64 whereas the last ResNet module is removed from the original ResNet classifier [58]. Here, we use the term "ResNet module"to refer to two stacked ResNet blocks, each of which contains two convolutional layers and a shortcut.

Discriminative Power Analysis In order to better visualize and interpret the classification models, we adopt the concept of "discriminative power" [119] of neurons/filters in each layer. Given N samples $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \in \mathcal{R}^{D_{\text{in}} \times N}$ and their corresponding class labels $\{c_1, \ldots, c_N\} \in [K]$, according to Fisher's linear discriminative analysis (LDA) [57],

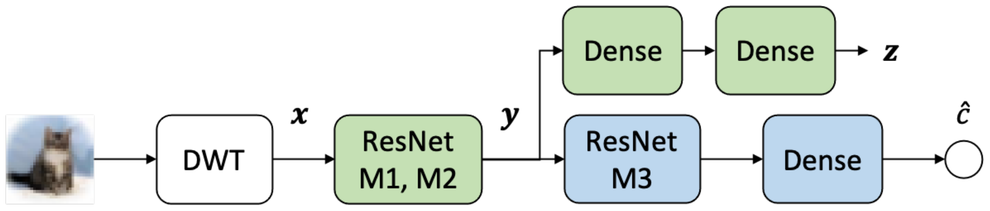


Figure 3.1: Architecture of the proposed wavelet-domain image classifier. "M" is short for "module". Green blocks are guided by MCR^2 loss and blue blocks are guided by CE loss.

the within-class scatter matrix $\boldsymbol{S}_w$ is defined as:

$$\boldsymbol{S}_w = \sum_{j=1}^{K} \sum_{\boldsymbol{x}\in\boldsymbol{\Pi}_j} (\boldsymbol{x} - \boldsymbol{m}_j)(\boldsymbol{x} - \boldsymbol{m}_j)^T \tag{3.4}$$

Here, $\boldsymbol{m}_j$ is the sample mean of $\boldsymbol{\Pi}_j$. The between-class scatter matrix $\boldsymbol{S}_b$ is defined as:

$$\boldsymbol{S}_b = \sum_{j=1}^{K} n_j (\boldsymbol{m} - \boldsymbol{m}_j)(\boldsymbol{m} - \boldsymbol{m}_j)^T \tag{3.5}$$

where n_j is the number of samples in $\boldsymbol{\Pi}_j$ and $\boldsymbol{m}$ is the global sample mean of the dataset. Following [119], we define the discriminative power D of each neuron/convolutional filter as:

$$D \doteq \text{trace}(\frac{\boldsymbol{S}_b}{\boldsymbol{S}_w}) \tag{3.6}$$

According to this definition, each neuron/filter has a discriminative power score. The neuron/filter with the highest score in a layer is the "best" neuron/filter of that layer. As a benchmark, we first detach the side branch in Fig. 3.1 and train the model using the CE loss on the CIFAR-10 dataset [123], reaching a 93.22% test accuracy. We plot the discriminative power of best filter per layer on both training and test set in Fig. 3.2. It can be observed that the discriminative power grows slowly from one layer to the next within the first two ResNet modules. This numerical analysis indicates that those layers tend to extract more general common features rather than discriminative features from the images.

For a lightweight CNN classifier that resides on an end device, such slow growth of discriminative power represents poor learning efficiency. To overcome this issue, we adopt the principle of MCR^2 at the front layers in our lightweight classifier in order to guide these layers to more quickly and more effectively extract the most discriminative image features. Based on the interpretability of MCR^2, this architecture is also more amenable to pruning.

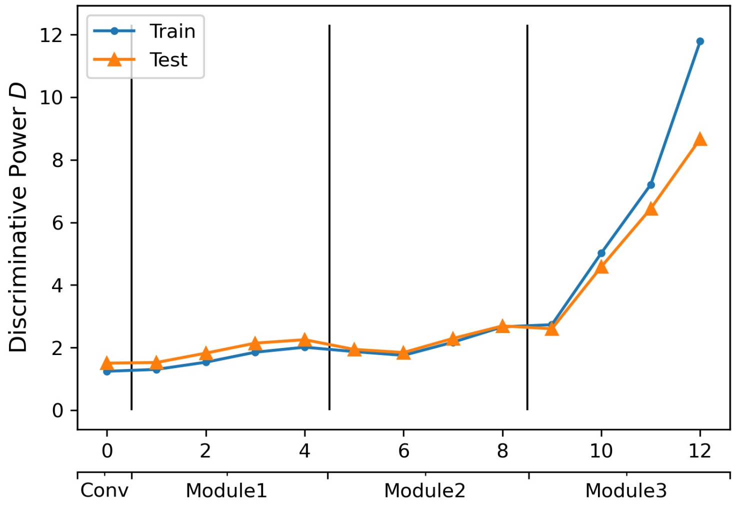


Figure 3.2: Layer-wise best filter's discriminative power in the CE-trained model.

Hierarchical Learning Our goal is to train the spectrum-domain CNN model according to two loss functions: the MCR^2 loss $\mathcal{L}_{\text{MCR2}} = -\Delta R(\boldsymbol{z})$ of Eq. (3.1) to control the latent encoding rate and the CE loss $\mathcal{L}_{\text{CE}}$ to minimize the classification discrepancy between the CNN output $\hat{c}$ and the true label c for each training image. A traditional training approach would be to superimpose the two losses to generate a sum loss function $\mathcal{L}_{CE} + \lambda \mathcal{L}_{\text{MCR2}}$ using a regularization variable λ. Such a naïve joint loss function, however, may lead to a convergence of a compromising local minimum that neither minimizes classification error, nor reduces the encoding rate. In fact, our preliminary studies confirm the poor outcome of such training method.

It is important to note that the principle of MCR^2 tend to generate latent representations that are feature-preserving, and of lower dimensions. In view of both important characteristics of MCR^2, we propose the concept of hierarchical learning in an effort to integrate the two loss functions. Specifically, our proposal is to divide the model training for image classification into two phases: (1) training of front modules (layers) to acquire diverse and discriminant features from inputs at lower dimensions; (2) training of ensuing later modules for accurate classification. Because of the feature-preserving characteristics of MCR^2, the first phase in training shall retain the key features necessary for accurate classification. At the same time, the compact nature of MCR^2 outputs makes it easier to identify candidates for pruning in order to reduce the overall complexity of the proposed lightweight image classifier.

Our dual phase training is different from [7], where a feature extractor is firstly trained before training the entire model from E2E by forming a CE loss regularized by a local discriminant regularization term and a diversity regularization term. We apply MCR^2 loss to only the front modules in phase 1 and apply the CE loss to only the later modules to implement classification in phase 1. This concept is shown in Fig. 3.1, where we build a side path in the learning model for applying the MCR^2 loss, which does not affect the subsequent classification training. This hierarchical learning strategy leverages the strength of MCR^2 and induces models more robust to pruning and potential data corruptions.

To summarize, each epoch of our hierarchical training consists of

Phase 1: Front modules, including ResNet modules 1 and 2 and the side branch, are updated by stochastic gradient descent to minimize $\mathcal{L}_{\text{MCR2}}$;

Phase 2: Blue modules, including ResNet module 3 and the ensuing dense layer, are updated by stochastic gradient descent to optimize $\mathcal{L}_{\text{CE}}$, while front modules are frozen from phase 1.

Feature Map Pruning: One benefit from the principle of MCR^2 is that the filters in the convolutional layers are more discriminate and naturally generates diverse and distinguishable intermediate features. As a result, it is simpler to apply LDA-based filter screening. Specifically, we adopt the neuron/filter screening strategy of [119] to reduce the computational complexity of the model. On the training set, we evaluate the discriminative power D of each filter in a convolutional layer and prune those filters with the lowest scores.

MCR^2-Guided Spectrum-Domain Auto-Encoder

We now consider the second deployment scenario that involves distributed auto-encoder/classifier where a source device is responsible for encoding image data for transmission to an edge server whereas the edge server is responsible for image classification.

When optimizing this distributed auto-encoder/classifier, the encoder node must efficiently pack useful features and discard redundant or less useful information. In other words, the learned representations should be compact within the same class while diverse between classes. In addition, the model needs to be robust to codeword mapping (i.e. quantization) error or label error due to data corruption.

Expanding the design of Fig. 3.1, we introduce a latent encoder consisting of a pooling layer and two dense layers before the rounding quantizer for dimension reduction on the intermediate feature $\boldsymbol{f}$ for transmission. At the edge server, the receiver begins with a latent decoder consisting of three transposed convolutional layers for latent recovery, as shown in Fig. 3.3.

The compact latent representations $\boldsymbol{y}$ are mapped (via rounding) into codeword $\hat{\boldsymbol{y}}$ before transmission to the receiver node over a communication data link. The "Dense" module in between encoder and decoder, consisting of one dense layer, is used as a side branch in phase 1 of hierarchical training with MCR^2 loss based on the latent $\mathbf{z}$. After hierarchical training, this model can be distributively deployed as an efficient encoder on remote mobile node and a decoder/classifier on the edge server for cloud-based classification.

Hierarchical Learning Similar to the lightweight classifier introduced in the previous section. We incorporate a single dense layer as a side branch for MCR^2 training in phase 1 to minimize the MCR^2 loss $\mathcal{L}_{\text{MCR2}}$. We apply the CE loss $\mathcal{L}_{\text{CE}}$ for optimization of the

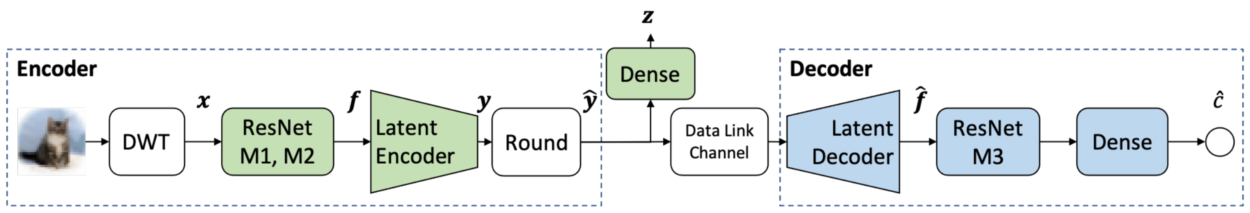


Figure 3.3: Architecture of the proposed wavelet-domain auto-encoder. “M” is short for “module”. Green blocks are guided by MCR^2 loss and blue blocks are guided by CE loss.

decoder/classifier only. To summarize, each epoch in our dual phase hierarchical training includes

Phase 1: Encoder modules, including ResNet modules 1 and 2, latent encoder, and the side channel dense layer, are jointly updated via stochastic gradient descent to minimize $\mathcal{L}_{MCR2}$;

Phase 2: Decoder modules, including the latent decoder, ResNet module 3 and the final dense layer, are jointly updated via stochastic gradient descent to optimize $\mathcal{L}_{CE}$, while Encoder modules are frozen after phase 1.

Feature Map Pruning In order to optimize the encoding rate in response to different link rate constraints, we further prune neurons from the bottleneck layer in latent encoder, which is equivalent to nullifying entries in the latent representation $\hat{\boldsymbol{y}}$. We adopt the same neuron screening strategy in [119]. Using training dataset, we evaluate the discriminative power D of each neuron in the bottleneck dense layer and prune those neurons of lowest scores.

Zero Gradient Recall that the rounding quantizer has zero derivative almost everywhere. During training, we adopt the method proposed in [124] to solve the zero gradient problem by adding a random uniform noise between -1 and 1 to the latent representations $\boldsymbol{y}$ as a relaxation of rounding. The reason of using a larger range of uniform noise between ± 1 instead of ± 0.5 in [124], is the recognition that MCR^2 tends to compact $\boldsymbol{y}$ toward zero, thereby leading to a greater quantization error effect on subsequent blocks. For this reason, larger additive noise is important to induce stronger robustness by the distributed learning model against quantization noise.

3.3.2 Hierarchical Training based on Joint Supervised and Self-Supervised Learning (SSL)

We adopt the same AE architecture proposed in [77] as shown in Fig. 3.3 and provide a simplified diagram in Fig. 3.4 for better understanding. The AE model consists of a pair of ResNet-based encoder and decoder as well as a lightweight side branch (for training only) from which the supervised and self-supervised LDR loss can guide the encoder.

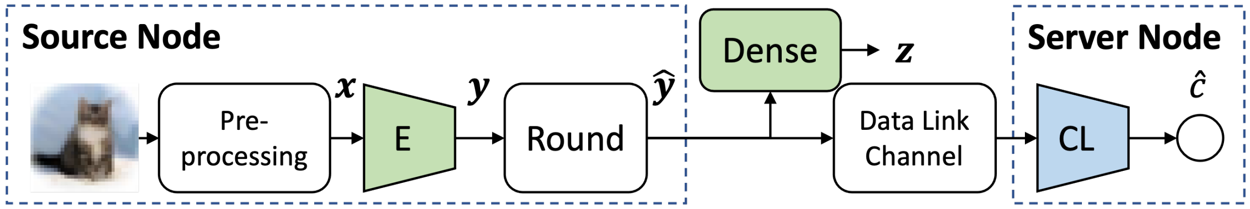


Figure 3.4: Architecture of adopted AE. “E” denotes “encoder” and “CL” denotes “classifier”.

When sensor node captures real-life images, distortions are common [40–42]. The encoder should learn to extract the underlying general information to account for such possible distortions with minimum loss. Self-supervised representation learning [125–130] can train the encoders to extract valuable features for a downstream task, such as image classification, and have demonstrated encouraging results. More specifically, the work of [44] suggests that SSL with MCR^2 principle is a promising approach to promoting LDRs' consistency under certain transformations/augmentations. Following [44], we also augment each image $\mathbf{x}_i$ in a mini-batch with n transformations randomly drawn from a collection $\mathcal{T}$ of augmentations with a know distribution $\mathcal{P}_{\mathcal{T}}$. These augmented images belong to the same class. We define the objective function of LDR-guided SSL as

$$\mathcal{L}_{\mathrm{LDR-SSL}} = -\Delta R(\boldsymbol{Z}) \doteq - R(\boldsymbol{Z}) + R_c(\boldsymbol{Z}, \boldsymbol{\Pi}^{\mathcal{T}}), \tag{3.7}$$

where $\boldsymbol{\Pi}^{\mathcal{T}}$ is the artificial self-labeled partition matrix.

However, model training by SSL algorithms alone has two main drawbacks. Firstly, since multiple augmentations generate a larger dataset, leading to potential overfitting of label-dependent but not feature-dependent information [128]. The convergence speed is slower. Secondly, SSL relies only on artificially-constructed labels instead of the categorical ground truth labels, which are available in our framework, and usually performs poorer than supervised learning [128,131]. Therefore, we propose to incorporate supervised learning with SSL, as a feature regularizer, to jointly balance training speed, rate-accuracy performance and LDRs' robustness to distortion. To summarize, our three-step hierarchical training includes

Step 1: Apply DuPHiL with Encoder loss $\mathcal{L}_{\mathrm{LDR}}$ and Decoder loss $\mathcal{L}_{CE}$, given the ground truth partition $\boldsymbol{\Pi}$.

Step 2: Apply DuPHiL with Encoder loss $\mathcal{L}_{\mathrm{LDR-SSL}}$ and Decoder loss $\mathcal{L}_{CE}$, given the artificially-constructed partition $\boldsymbol{\Pi}^{\mathcal{T}}$.

Step 3: Apply E2E training with loss $\mathcal{L}_{CE}$, while at a smaller learning rate than Phases 1 and 2.

Selecting Latent Dimensions Our tests show that, an initial quick CE-training phase can guide the selection of latent dimension from a finite set of values for our AE model. This initial screening consists of training separate AEs of various latent dimensions to minimize CE loss before selecting one with the highest accuracy.

Entropy-Based Latent Quantization Recent works [132] attempt to obtain different encoding data rates by training various AEs to manually adjusting a rate-accuracy trade-off parameter λ in the traditional joint-compression-and-classification loss function $\mathcal{L} = \mathcal{L}_R + \lambda\mathcal{L}_{CE}$. Such a naïve joint loss function, however, may lead to a convergence to a compromising local minimum that neither minimizes classification error, nor reduces the encoding rate. Furthermore, tuning λ can be time-consuming and costly. Alternately, several more complex frameworks, such as RNNs [133], trainable quantization modules together with conditional AEs [134] and asymmetric gain modules [135] have been developed to achieve continuous compression rate adaptation via a single model. Nevertheless, the long processing time and/or high computational complexity of these methods render them impractical for cloud-based DL in IoT systems. In our work, as the compact LDR $\hat{\mathbf{y}}$ naturally contains discriminative information for classification, we suggest to train only one AE and reduce the data rate of latent $\hat{\mathbf{y}}$ by directly adjusting the quantization step sizes.

Since different entries in latent representations may carry different levels of information (entropy), for each entry $\mathbf{y}_i$, we assign a quantization step size $\boldsymbol{q}_i$ to $\mathbf{y}_i$, where $i = 1,\ 2,\ \ldots,\ d_{\mathbf{y}}$, that is linearly proportional to average entropy $H(\mathbf{y}_i)$ on the training set for Gaussian $\mathbf{y}_i$. To be specific, $\boldsymbol{q}_i = s \times H(\mathbf{y}_i)$, where s is a scalar used to adjust the overall quantization level. We obtain the quantized latent $\hat{\mathbf{y}}_i$ for subsequent encoding and transmission by

$$\hat{\mathbf{y}}_i = \frac{\mathbf{y}_i}{\boldsymbol{q}_i} = \frac{\mathbf{y}_i}{s \times H(\mathbf{y}_i)}. \tag{3.8}$$

Clearly, smaller $\boldsymbol{q}_i$ values lead to smaller range of quantized latent values $\hat{\mathbf{y}}_i$, thereby generating fewer encoded bits (i.e. lower rate). Thus, those entries with higher entropy (i.e., higher variations) are quantized with finer resolution, while entries with lower entropy suffer less from quantization error.

Manual Truncation on LDRs Channel bandwidth can vary over time and the encoder needs to be updated accordingly in response. The proposed entropy-based quantization in

Section 3.3.2 is applied directly to encoder outputs without optimizing the encoder and can yield sub-optimal rate-accuracy performance with large quantization step sizes $\boldsymbol{q}_i$ (higher quantization error). Instead of re-training a new encoder from scratch, which is time-consuming and impractical, we fine-tune the existing model. As an alternate way to reduce the coding rate of latent representations while achieving a good rate-accuracy trade-off, we propose to truncate the bottleneck layer in the encoder, which is equivalent to manually truncating certain entries in $\hat{\boldsymbol{y}}$, followed by encoder fine-tuning. Since both the overall diversity and in-class compactness of latents are fortified by LDR-guided fine-tuning, they are expected to be more robust against such truncation.

3.4 Experimental Results

We train our AEs on CIFAR-10 and CIFAR-100 datasets, using a ResNet-18 and a ResNet-34 backbone architecture, respectively. Both CIFAR datasets consist of 50000 32×32 RGB training images and 10000 test images, while CIFAR-10 dataset has 10 classes and CIFAR-100 dataset has 100 classes. To begin, we pre-train the AE models, with side channel disabled, using CE loss for 200 epochs with a batch size of 200 and an initial learning rate of 0.1. The learning rate is multiplied by 0.1 at epochs 100, 150 and 180, respectively. Next, we update the pretrained models with either the dual-phase hierarchical learning strategy or the three-step joint learning strategy for 100 epochs. Note that MCR^2 loss requires a large batch size such that the distributions of the outputs $\boldsymbol{z}$ at the side channel is approximately Gaussian. We use a batch size of 1000 during the dual-phase or three-step training stage. Our experiments are conducted on a TITAN X GPU with 12 GB memory using the PyTorch library [136].

3.4.1 DuPHiL Performance Evaluation

In the dual-phase training process, the initial learning rate for MCR^2 loss in phase 1 is 0.001 for both proposed models. The initial learning rate for CE loss in phase 2 is 0.0025 for the lightweight image classifier and is 0.009 for the distributed auto-encoder/classifier. Both learning rates for phase 1 and 2 are reduced to 1/10 at epochs 25, 50 and 75, respectively. When generating random corrupted labels for training set, we use the random seed of 10.

MCR^2-Guided Lightweight Classifier

For comparison, we train three lightweight models of the same architecture in Fig. 3.1 based on (1) pure CE loss; (2) joint loss by superimposing gradients of the two cost functions to optimize weights in the entire model; (3) the proposed hierarchical learning strategy.We achieved similar test accuracy of 93.22%, 92.13% and 93.19%, respectively.

To visualize the effect of the three training methods, Fig. 3.5 compares their discriminative power D of the most discriminative ("best") filter in each convolutional layer. For all three models, we observe that the discriminative power generally grows between successive layers within each ResNet module. We do notice the mild drop of discriminative power between successive modules, which is possibly due to the reduction of feature map dimension and the doubling of filter numbers between modules. Such structural transition between modules leads to less information capturing by individual filter in the first layer of each succeeding module and subsequent drop of discriminative power.

As expected, since the CE loss does not impose any direct constraints such as the linear separability on the intermediate features, the resulting discriminative power rises slower than those from joint-loss or hierarchical training. Meanwhile, joint loss training naïvely superimposes the MCR^2 loss and CE loss and delivers lower accuracy compared with the CE-trained or hierarchically trained models. Remarkably, the discriminative power scores from hierarchical training rises rapidly in module 3, firmly establishes the benefit of focusing phase 1 training to achieve compact features before focusing phase 2 training on classification accuracy.

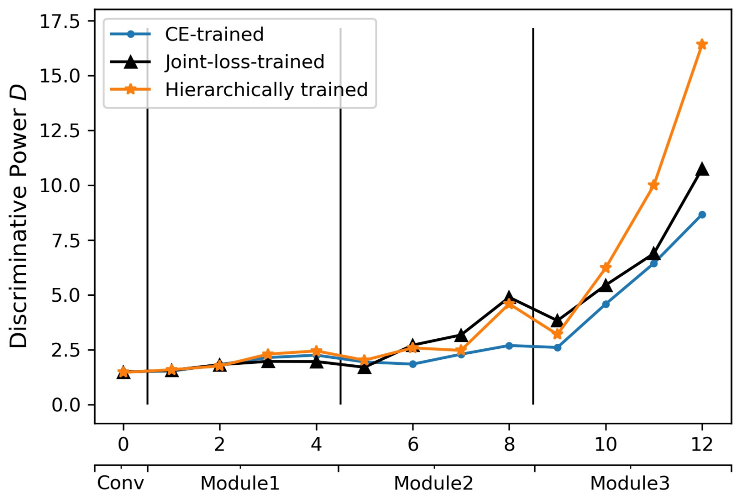


Figure 3.5: Discriminative power of the best filter in each convolutional layer, computed on test set.

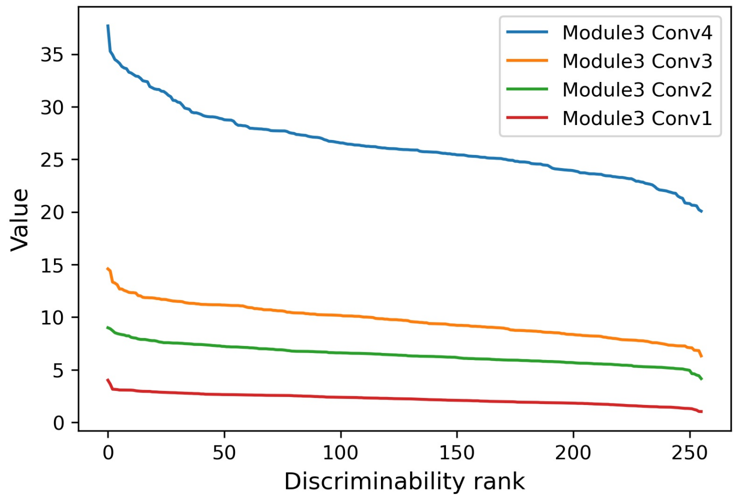


Figure 3.6: Discriminability ranks of convolution filters in the four layers in ResNet Module 3, computed on training set. There are 256 filters in each layer and their discriminability power scores are sorted from high to low.

To efficiently reduce computational complexity and memory, we prune the four convolutional layers in ResNet module 3 which account for the largest number of parameters. Since the best discriminative power scores in these layers are clearly higher as seen from Figs. 3.5

and 3.6, we can prune filters of lowest scores without seriously affecting the final classification accuracy. In Table 3.1, the first "Proposed Baseline" model refers to the network in Fig. 3.1 obtained from our hierarchical training method. It naturally provides the best test accuracy. We also present two sets of pruning results in Table 3.1 described below.

- For case "(A)", the numbers of convolutional filters in the four layers of ResNet module 3 are pruned from $\{256, 256, 256, 256\}$ to $\{166, 204, 128, 76\}$. "CE-Reduced-Size (A)" refers to a model retrained from scratch by CE loss after implementing filter reduction in ResNet module 3, matching the numbers of remaining filters in "Proposed-Pruning (A)" case. "CE-Pruning (A)" refers to the baseline model trained with CE loss, followed by the same screening and pruning process as "Proposed-Pruning (A)".
- For case "B", the only difference from case "A" is that the numbers of convolutional filters in the four layers are trimmed from $\{256, 256, 256, 256\}$ to $\{166, 166, 128, 76\}$ instead.

The numbers inside parentheses in Table 3.1 represent the accuracy reduction versus the "Proposed Baseline" in the first row. The results show that the pruned models from proposed hierarchical learning method achieve the classification accuracy of 92.95% and 92.65% for case A and case B, respectively, only 0.24% and 0.54% lower than the full model. At the same time, the number of network parameters are reduced by over 30% whereas the number of floating point operations (FLOPs) are lowered by over 12%. Compared against models with same computational complexity from traditional CE training ("CE-Reduced-Size"), which generates the best achievable accuracy through retraining given the complexity reduction, the proposed hierarchical learning and pruning exhibit accuracy loss of only 0.04% and 0.1%, respectively, for case A and case B. These comparisons show that MCR^2-guided hierarchically-trained models tend to deliver more separable features in intermediate layers and can be safely pruned by applying LDA-based neuron/filter screening with negligible performance loss.

Table 3.1: Pruning results

Model	Non-Zero Parameters	Test Accuracy	FLOPs
Proposed Baseline	2.92 M	93.19%	214 M
CE-Reduced-Size (A)	1.99 M (-31.8%)	92.99% (-0.20%)	187 M (-12.6%)
CE-Pruning (A)		90.86% (-2.33%)	
Proposed-Pruning (A) (new)		**92.95% (-0.24%)**	
CE-Reduced-Size (B)	1.90 M (-34.9%)	92.75% (-0.44%)	185 M (-13.6%)
CE-Pruning (B)		90.60 % (-2.59%)	
Proposed-Pruning (B) (new)		**92.65% (-0.54%)**	

Distributed Auto-Encoder Deployment

To test our second auto-encoder architecture of Fig. 3.3, we adopt a simple arithmetic encoder to convert the quantized latent vector $\hat{\boldsymbol{y}}$ into bitstreams for bit rate measurement. We obtain the approximated cumulative distribution functions (CDFs) by generating histogram of $\hat{\boldsymbol{y}}$ using the full training set. To further compress $\hat{\boldsymbol{y}}$, we evaluate the discriminative power of each entry in $\hat{\boldsymbol{y}}$ based on the training set and prune the least discriminating entries. As benchmarks, we also train the Encoder together with Decoder from E2E by minimizing the CE loss to generate the "CE-trained" model.

The rate-accuracy trade-off results are illustrated in Fig. 3.7. It is clear that, for classification, the input image can be compressed to a bit rate of around 0.3 bits-per-pixel (bpp) by using either hierarchical MCR^2 training or E2E CE training without accuracy loss. Fig. 3.7 shows that our proposed hierarchical learning can achieve similar rate-accuracy performance as CE-based E2E training. For both CE-trained and MCR^2-guided auto-encoders, pruning the output $\hat{\boldsymbol{y}}$ based on LDA followed by fine-tuning the last dense layer in the Decoder can achieve higher bandwidth efficiency by reducing encoding rate to 0.11 bpp with only less than 1% loss of accuracy.

It is interesting to observe that during phase 1 training, both the overall rate R and group-wise rate R_c would grow, implying that the MCR^2-guided encoder tends to encode latent representations into more bits, while ensuring the in-class compactness and between-class discrimination of latent vectors.

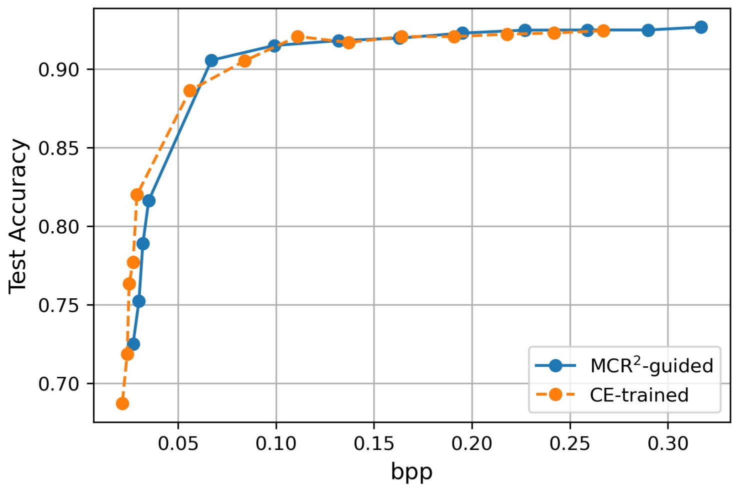


Figure 3.7: Rate-accuracy performance on test set of the proposed MCR^2-guided AE in comparison with a CE-trained AE.

Robustness against Corrupted Labels

The authors of [44] have demonstrated that deep models under MCR^2 can learn well despite the presence of some corrupted labels during training. To test our proposed model's robustness against label corruption, we train the auto-encoder with corrupted labels firstly by CE loss in an E2E manner, which are used as the baseline models. We then fine-tune the same model by applying the proposed hierarchical learning method. For classification evaluation, we use the correct ground truth labels.

Our experiments include label corruption ratio (CR) of 10%, 20% and 30%, respectively. We present the layer-wise discriminant power scores in Fig. 3.8. It is evident that MCR^2-guided learning models exhibit higher discriminative powers than CE-trained models when they are subject to the same level of label corruption. From Fig. 3.9 we can observe that with 10% training labels corrupted, the MCR^2-guided model can deliver up to 1% higher test accuracy than the CE-trained model at the same encoding data rate. Meanwhile, the MCR^2-guided model achieves robust learning even under 20% label corruption and clearly delivers higher accuracy at the same data rate than the corresponding benchmark CE model. Even with 30% random label corruption, our proposed learning model still yields a comparable rate-accuracy performance to the benchmark CE model with 20% label corruption. These results demonstrate the robustness of the proposed hierarchical learning against noisy training data by incorporating the MCR^2 principle.

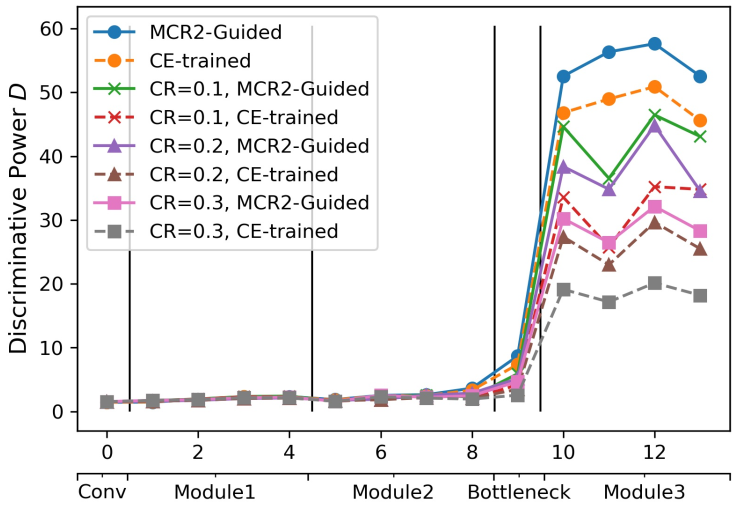


Figure 3.8: Layer-wise discriminative power of proposed distributed auto-encoder, computed on test set.

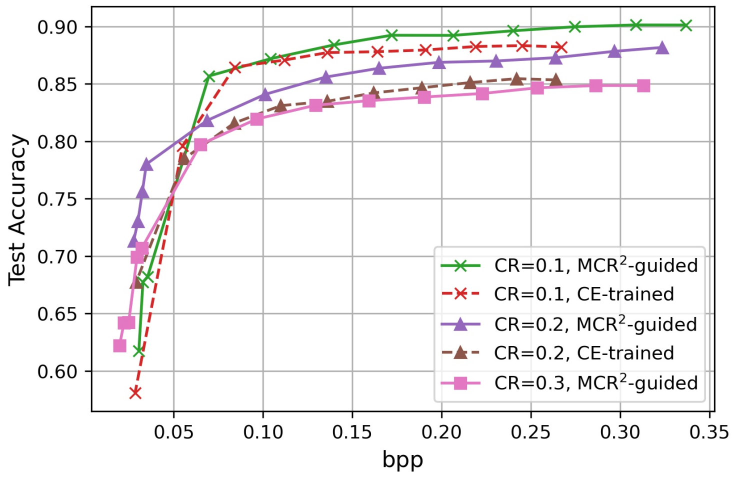


Figure 3.9: Selected test accuracy vs. bpp performance of proposed auto-encoders with corrupted training labels.

Effects on Imbalanced Datasets

To measure the effects of our proposed model and the DuPHiL training strategy on imbalanced training data, we generate artificial imbalanced CIFAR-10 and CIFAR-100 datasets by taking out training samples from some randomly selected classes to make them being under-represented. For CIFAR-10, which comes with 5000 training images per class, we pick two classes and manually remove 2500 or 4900 training images from them. For CIFAR-100, which comes with 500 training images per class, we randomly pick 5, 10 or 20 classes and remove 250 training images from these classes. Similar to the robustness tests against corrupted labels, we also pre-train the models using the artificial imbalanced training set by CE loss, whose results are considered as baselines, followed by fine-tuning process via our proposed DuPHiL strategy. We would like to highlight that in these experiments, the test sets, used in classification evaluation, are untouched and hence balanced. We present the classification accuracy on test sets in Tables 3.2 and 3.3.

Table 3.2: Accuracy performance of proposed model, trained on artificial imbalanced CIFAR-10 training set, tested on balanced CIFAR-10 test set.

Experiment 1: 2500 training images in each of the 2 minority classes

Training Strategy	2 Minority Classes	8 Majority Classes	Overall
CE (Baseline)	84.35%	93.19%	91.42%
DuPHiL	**86.2%**	**93.26%**	**91.85%**

Experiment 2: 100 training images in each of the 2 minority classes

Accuracy	2 Minority Classes	8 Majority Classes	Overall
CE (Baseline)	27.1%	93.29%	80.05%
DuPHiL	**33.75%**	**93.48%**	**81.53%**

It is obvious that on the imbalanced CIFAR-10 datasets with 2 under-represented classes, the MCR^2-guided model preserves the accuracy performance on the 8 majority classes and shows an accuracy improvement of 1.85% and 6.65% when 50% and 98% of samples are manually removed from 2 selected classes in the training set, respectively. For the imbalanced CIFAR-100 dataset, the MCR^2-guided model always outperforms baseline CE model on the majority classes but the gain becomes more marginal with more minority classes in the

Table 3.3: Accuracy performance of proposed model, trained on artificial imbalanced CIFAR-100 training set, tested on balanced CIFAR-100 test set.

Experiment 1: 250 training images in each of the 5 minority classes

Training Strategy	5 Minority Classes	95 Majority Classes	Overall
CE (Baseline)	68.6%	68.2%	68.22%
DuPHiL	**69.4%**	**68.65%**	**68.69%**

Experiment 2: 250 training images in each of the 10 minority classes

Training Strategy	10 Minority Classes	90 Majority Classes	Overall
CE (Baseline)	**69.7%**	68.1%	68.26%
DuPHiL	69.4%	**68.3%**	**68.41%**

Experiment 3: 250 training images in each of the 20 minority classes

Training Strategy	20 Minority Classes	80 Majority Classes	Overall
CE (Baseline)	**67.5%**	67.9%	**67.85%**
DuPHiL	66.8%	**68.1%**	**67.85%**

dataset. Moreover, the MCR^2-guided model exhibit better performance only when there are 5 minority classes, which is a relatively easier task than the other two cases. With more than 5 minority classes, it turns out that the CE models provides higher classification accuracy on minority classes and the gap between CE and MCR^2-guided models grows with more classes are manually processed to be under-represented. We claim that this is likely due to the fact that when there are more classes available in the dataset, the principle of MCR^2 demands more samples than CE loss to learn an optimal mapping that projects each classes into orthogonal subspaces.

Altogether, these results tell that the proposed DuPHiL strategy can be applied to alleviate the overfitting problem resulted from data imbalance for the datasets with a small amount of different categories.

Compatibility with Decoder Re-Training

To illustrate the general compatibility of our Encoders, we freeze the Encoders after CE or MCR^2-guided training, but train two new Decoders/classifiers from scratch, including: (1) a Decoder of the same architecture as in Fig. 3.3 but optimized with Kullback-Leibler Divergence (KL-D) [137] loss and (2) a linear support vector machine (SVM) [138]. We present the obtained results in Table 3.4. The results show that the encoder modules from DuPHiL continue to deliver robust performance. Using a linear SVM and the proposed decoder architecture trained by KL-D loss, we in fact observe up to 0.27% and 1.34% classification accuracy improvement on CIFAR-10 and CIFAR-100 datasets, respectively, over encoders from E2E training. Our results demonstrate the proposed MCR^2-guided Encoders are more flexible with various subsequent classifiers in comparison with E2E training.

Table 3.4: Accuracy performance of various classifiers based on fixed pre-trained Encoders.

Dataset	Classifier Model	Training Strategy	
		E2E (Baseline)	DuPHiL (Proposed)
CIFAR-10	As in Fig. 3.3 (CE loss)	92.64%	**92.77%**
	As in Fig. 3.3 (KL-D loss)	92.63%	**92.75%**
	Linear SVM	92.1%	**92.37%**
CIFAR-100	As in Fig. 3.3 (CE loss)	68.54%	**69.83%**
	As in Fig. 3.3 (KL-D loss)	68.45%	**69.79%**
	Linear SVM	64.69%	**65.37%**

3.4.2 Three-Step Joint Learning Performance Evaluation

First of all, to determine latent dimensions in the model design stage, we detach the side training branch to form an AE with a bottleneck layer dimension $d_{\mathbf{y}} \in \{32, 64, 128, 256, 512\}$ and pre-train each model using CE loss for 200 epochs. After this training step, we compare the rate-accuracy performances to choose the best latent dimension size accordingly and the selected model is labeled as the "CE-trained" (CE-T) baseline model. Using the pre-trained baseline model, we further apply our three-step joint training algorithm to generate the "LDR-fine-tuned" (LDR-FT) model.

Rate-Accuracy Performance

We present the rate-accuracy performance of models trained by the proposed three-step training strategy (LDR-FT) together with the DuPHiL models and baseline CE-T models, respectively, on CIFAR-10 with $d_{\mathbf{y}} \in \{32, 64, 128\}$ and CIFAR-100 with $d_{\mathbf{y}} \in \{64, 128, 256\}$ in Fig. 3.10. Assuming each individual entry in the quantized latent representations $\hat{\mathbf{y}}$ follows Normal distribution, the total entropy of $\hat{\mathbf{y}}$ is evaluated on the test set. To vary the entropy of each model, as discussed in Section 3.3.2, we assign a quantization step size $\mathbf{q}_i$ to each entry in $\mathbf{y}_i$ based on their entropy value and linearly scale the step size by a constant factor s.

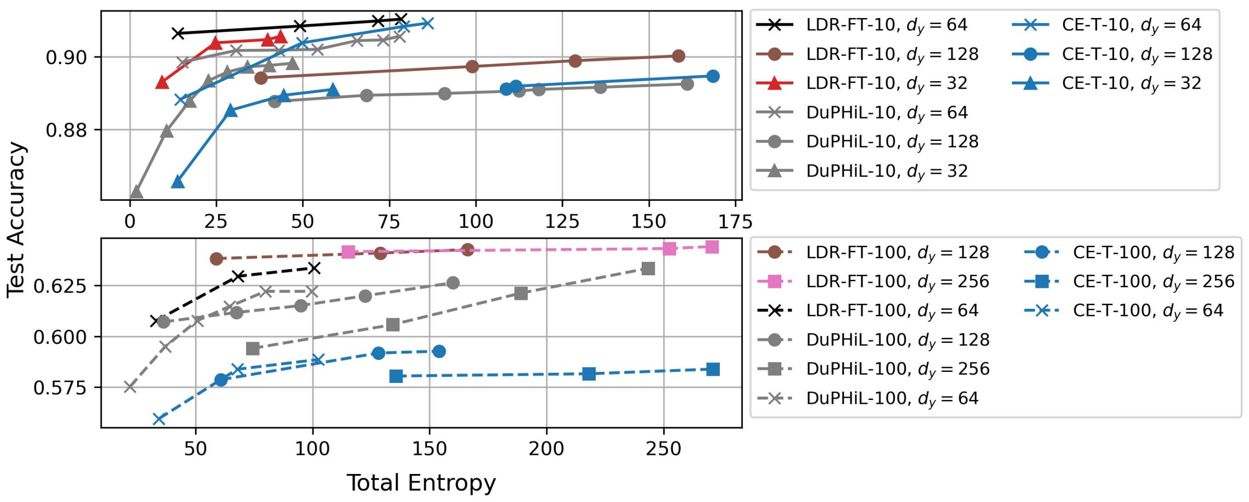


Figure 3.10: Accuracy of proposed LDR-FT versus DuPHiL and CE-T models on CIFAR-10 or CIFAR-100 test set.

It is obvious that the classification performance correlates positively with total entropy. As claimed in Section 3.4.1, targeting at model robustness and compatibility, DuPHiL does not achieve a significant performance gain over CE-T. Our LDR-FT models achieves a higher classification accuracy at the same total entropy compared with their corresponding DuPHiL or CE-T baseline models, yielding a better entropy-accuracy trade-off. For example, with $d_{\mathbf{y}} = 64$ as the optimized latent dimension according to our experiments, the test accuracy achieved by baseline CE-T is 90.92% whereas the LDR-FT model accuracy is is 0.11% higher at 91.03%. The LDR-FT model is also more robust to entropy-based quantization as its test accuracy drops by only 0.39% when the total entropy is reduced from 71.68 to 13.93, whereas the accuracy of the baseline CE-T model decreases by 2.09% when the total entropy is reduced from 86.02 to 14.64. Similarly, on CIFAR-100, at the same entropy, LDR-FT models usually outperform CE-T in test accuracy by over 3%. Further, the results also reveal unexpectedly that a higher latent dimension $d_{\mathbf{y}}$ does not always provide better classification accuracy, despite the ability to pack more information. We investigate and discuss this phenomenon in Section 3.4.3.

Robustness against Truncation

To illustrate the robustness against truncation when channel bandwidth is lowered, we manually truncate/nullify the last half latent entries in each of the same CE-T base models in Section 3.4.2, before fine-tuning them with either proposed training strategy (LDR-FT) or E2E CE training (CE-T). We compare the performance of LDR-FT versus CE-T models on CIFAR-10 with $d_{\mathbf{y}} \in \{32, 64, 128\}$ and CIFAR-100 with $d_{\mathbf{y}} \in \{64, 128, 256\}$ in Fig. 3.11.

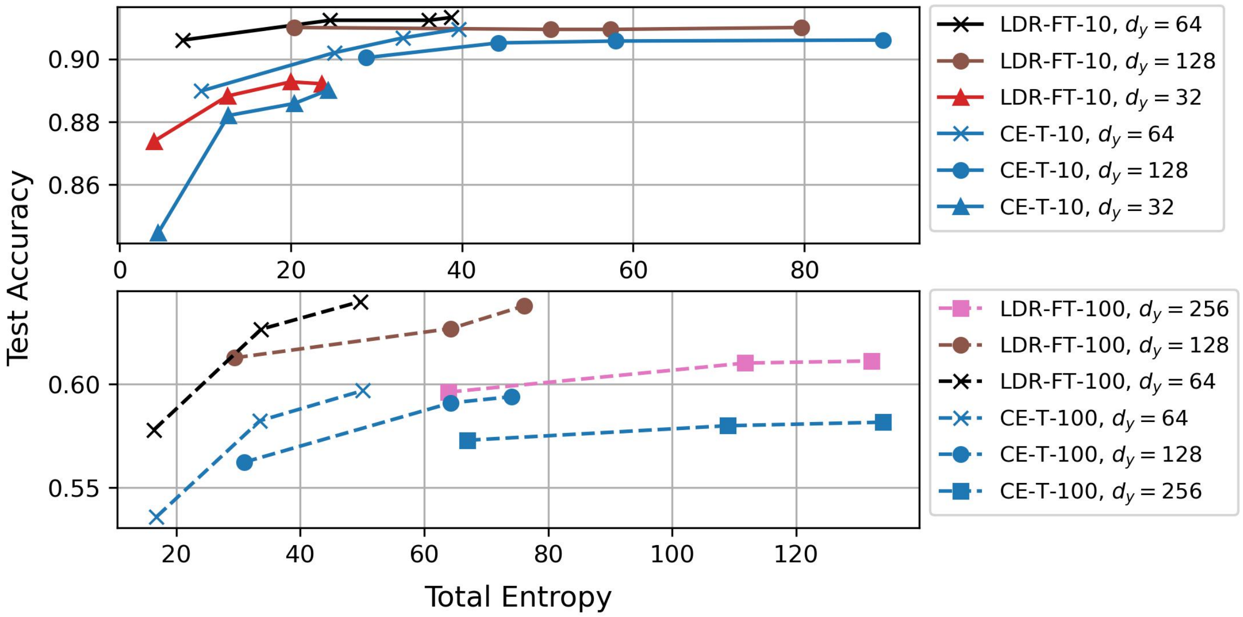


Figure 3.11: Accuracy of LDR-FT versus CE-T models (with manual truncation on latent) for CIFAR-10 or CIFAR-100.

As shown in Fig. 3.11, the truncated LDR-FT models outperform their CE-T counterparts in terms of the entropy-accuracy trade-off, indicating that the LDR-FT training strategy fortifies models to be more robust against such active truncation in response to the lower channel bandwidth. Moreover, comparing the performance of $d_{\mathbf{y}} = 128$ models on CIFAR-10 before and after manual truncation in Figs. 3.10 and 3.11, we can observe an accuracy improvement of approximately 1%, which again confirms our claim in Section 3.4.2 that higher latent dimension is not necessarily always the optimal choice.

Robustness against Distortive Transformations

To enhance model robustness against distortions, we augment the dataset by randomly selecting and applying one or more transformations among cropping, flipping, color jitter and uniform noise. After training, we randomly apply the same set of transformations to evaluate how well the models adapt to these deformations. We demonstrate the performance of LDR-FT versus CE-T models on the training set and the test set of CIFAR datasets in Figs. 3.12 and 3.13, respectively. Clearly, LDR-FT models achieve better accuracy on the

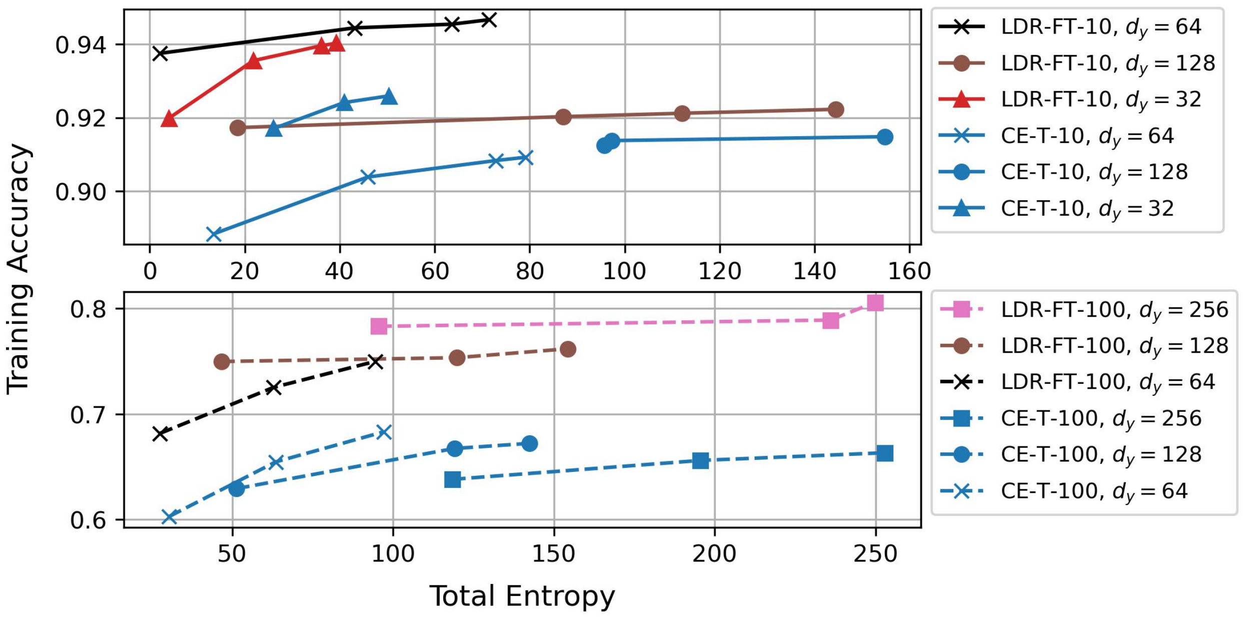


Figure 3.12: Training set accuracy of LDR-FT versus CE-T models (with images distorted) for CIFAR-10 or CIFAR-100.

distorted images than CE-T ones, implying that they learn to preserve the key information

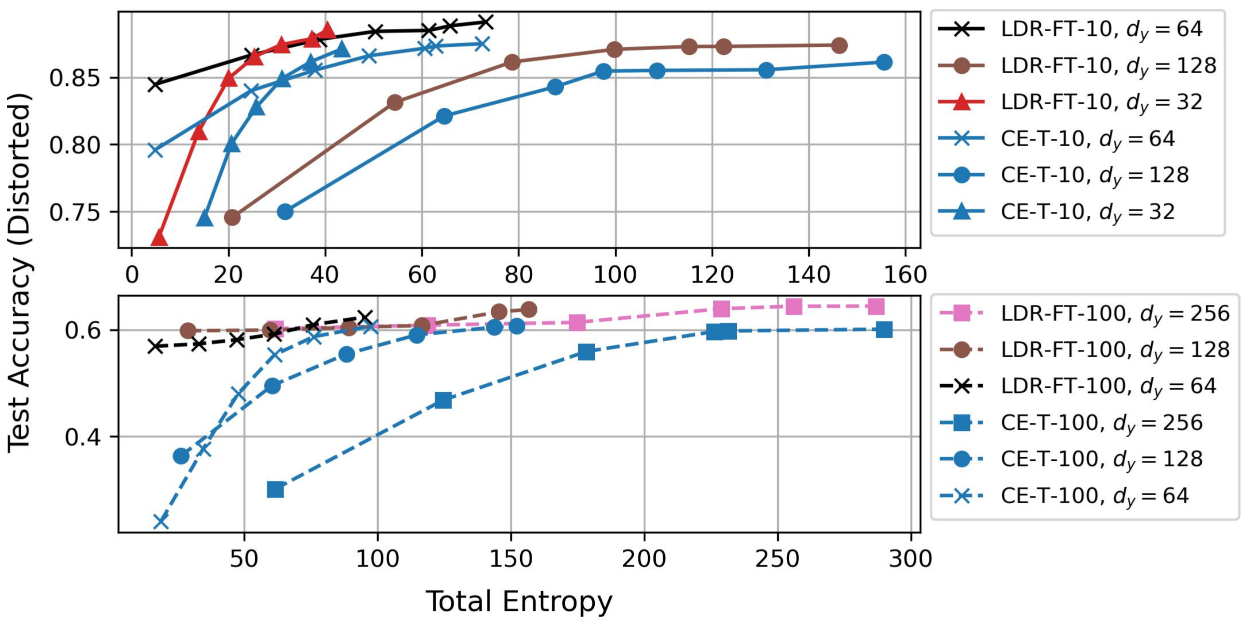


Figure 3.13: Test set accuracy of LDR-FT versus CE-T models (with images distorted) for CIFAR-10 or CIFAR-100.

for classification that is invariant toward image distortions as expected.

3.4.3 Impact of Latent Dimensions on Model Performance

Conceptually, with a higher latent dimension, we expect better classification accuracy as the latent representations are more informative. However, experiments show that this does not always hold.

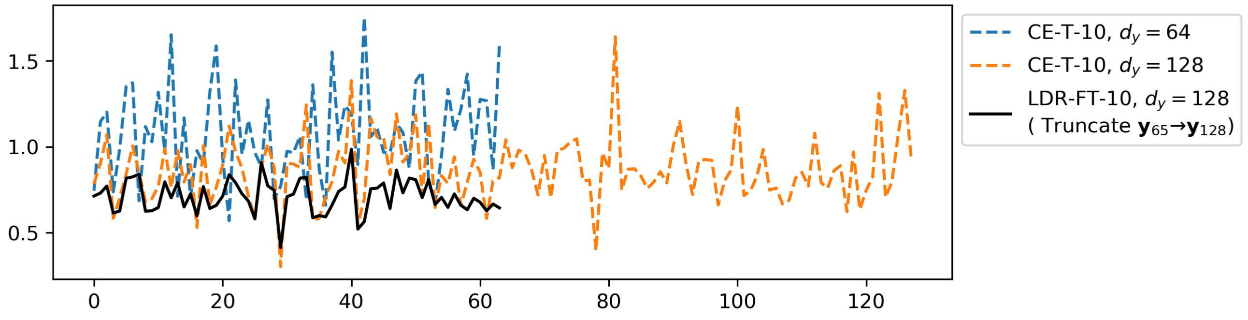


Figure 3.14: Standard deviation of entries in latent representation $\hat{\mathbf{Y}}$ computed on CIFAR-10 test set, with $d_{\mathbf{y}} = 64$ and 128.

To investigate the possible causes, without the loss of generosity, we examine the variations of entries in the learned latent representations for dimension sizes $d_{\mathbf{y}} = 64$ and 128, respectively, using the CIFAR-10 test set. As Fig. 3.14 shows, "CE-T" and "LDR-FT" represent CE-trained baseline and LDR-fine-tuned models, respectively. Comparing the two CE-T cases, the smaller latent dimension $d_{\mathbf{y}} = 64$ model learns to pack more information in the latent by increasing variations, making its latent more robust against quantization noise in comparison with $d_{\mathbf{y}} = 128$. Meanwhile, after applying manual truncation and fine-tuning "CE-T-10, $d_{\mathbf{y}} = 128$", the "LDR-FT-10" model learns to suppress latent variations, leading to an entropy reduction while preserving the most critical between-class discriminative information, thereby avoiding classification accuracy loss as can be seen in Fig. 3.10.

3.5 Summary

In this chapter, we study the training of AEs for distributed compression or classification in distributed learning environment. We first propose a novel hierarchical learning strategy to achieve the dual objectives of efficient discriminant feature extraction and accurate classification. Applying the information theoretic MCR^2 principle, we assign the sub-task of efficient feature extraction to front ResNet modules and the sub-task of classification to later modules. Instead of naïvely summing loss functions of each objective for E2E training, we train the front and later modules alternately to minimize their respective loss functions. By applying the MCR^2 loss to guide front modules to acquire in-class-compact and between-class-separable features before using the CE loss to optimize later modules for classification, our hierarchical learning not only achieves good accuracy as existing CE models but also provides robustness to LDA-based filter pruning and label corruption. Then, we suggest to incorporate self-supervised training, also guided by the information theoretic LDR criterion, to the proposed training algorithm such that the AEs can be amenable to common augmentations/transformations. In addition, to adjust transmission rate in response to channel bandwidth, we propose an entropy-based quantization method, which operates directly on any pre-trained encoders, and we propose a simple latent truncation in conjunction with encoder fine-tuning. Further, we investigate the impact of latent dimensions on the performance of AEs and suggest to optimize latent dimensions by using an initial screening process. Both proposed learning strategies can apply directly to various existing AE architectures.

Chapter 4

A Principled Hierarchical Approach for Few-Shot Deep Learning

In this chapter, we introduce a few-shot learning strategy with the guidance from the principle of MCR^2. This strategy is capable of navigating the DL models to extract more discriminant features from input data. With limited amount of training samples, this new method can reduce model overfitting to seen data and make the models generalize better to unseen data.

4.1 Few-Shot Image Classification

To enable supervised DL models to achieve impressive breakthroughs, availability of large quantities of labeled training data is usually an essential preliminary requirement. In cases where the accessibility of supervised information is limited, it can be more challenging for DL approaches to generalize to new tasks. Few-shot learning (FSL) [24, 26, 139] studies the potential of DL applications to such low-data regimes, and the work of [26] has been presumably considered as one of the main contributors to the solid progress made in FSL recently. The typical setting of an N-shot K-way few-shot image recognition problem involves learning classifiers to categorize image samples into K prediction classes, given only

N training samples from each class. During inference, a total of Q images are provided to be classified into the K prediction classes. The set of $N \times K$ training samples and the set of Q inference samples are defined as the "support" set and the "query" set, respectively. Existing FSL works can be divided into the three perspectives: data-level, parameter-level and algorithm-level approaches.

Straightforwardly, data-level approaches utilize prior knowledge to increase the number of available training samples, including data augmentation via manually-designed transformation rules [46] and synthetic data generation [140], etc. These methods are commonly seen as a pre-processing step in FSL tasks. Another line of approach in the parameter-level uses prior knowledge to constrain the complexity of DL classifiers as well as the mapped feature subspace. As the hypothesis space of classifiers becomes narrower, small number of training data may be sufficient and overfitting can thus be reduced. Typical parameter-level methods include parameter sharing from other tasks [141,142], embedding learning [23,143] and learning with external memory [144]. Lastly, the algorithm-level approaches aim at seeking the optimal hypothesis classifier. A popular method, known as "meta-learning" [145,146], trains models with batches of tasks, instead of samples, and optimizes classifiers with the expectation that future update steps based on data from other different tasks improves the generalization capability of models to the current task. Specifically, each task in meta-learning is comprised of a labeled dataset consisting of a support set, which provides task-specific information, and a query set, which evaluates generalization performance on this task. Note that the classes present in each task shall differ.

However, these previous works do not pose direct regularization on the discrimination capability of intermediate features extracted by the front layers in a DL model, which shall in theory benefit classification performance. As a result, the models tend to experience severe overfitting to the training set. In this work, we propose to alleviate the common overfitting problem, enhance the generalization capability of DL models and improve the classification accuracy on the unseen query set by guiding the feature extration layers to learn linear discriminant feature mappings via the principle of Maximal Coding Rate Reduction (MCR^2).

4.2 Related Works

4.2.1 Learning Discriminant Features for Few-Shot Learning

Some recent progress in few-shot image recognition features learning to extract discriminant embeddings from images [29, 147, 148] to avoid severe overfitting.

The Cross Attention Module (CAM) introduced in [147] enhances the discriminability of extracted features by learning to localize the most discriminative and representative regions in images and generate a weighted attention map in assistance for subsequent classification. Instead of proposing a new architecture, the authors of [148] design the unsupErvised discriminAnt Subspace lEarning (EASE) strategy, which quantifies the distance of representations in their subspace using a similarity metric and maximizes both the inter-class feature similarity and the intra-class feature dissimilarity to ensure the discriminability of learned representations. However, both of [147, 148] are conducted based on a transductive setting that is less practical, where all the inference images shall be present at once. Based on the ProtoNet [23] backbone, the few-shot embedding adaptation with Transformer (FEAT) model in [29] applies a set-to-set function to the learned embeddings to make them discriminative even for unseen classes. Attached after the instance embedding modules derived from seen classes, this learnable projection adjusts the mapping from learned embeddings to the class prototypes. These embeddings preserve the discriminative information, especially for the unseen ones in the query set, and will be used for downstream classification. Concretely, these previous studies have proved that promoting discriminability in the extracted prototypes in an unsupervised manner can result in superior performances compared with corresponding baseline approaches.

Different from these unsupervised methods, build upon the workflow developed in [29], the discriminability of learned latent representations for FSL can be advanced in a supervised manner [43] by means of bringing a surrogate latent regularization term to the loss function, that measures the difference between inter-class and intra-class coding rates of the projected prototypes. Quantified by an information-theory-based metric, known as MCR^2, the inter-class discriminability, intra-class compactness and linearity of prototypes can be enhanced during training. The benefits for the FEAT framework is twofold: inter-class discriminability pushes class-wise mean, or the clustering centers, of the latent embeddings further away from each other; intra-class compactness ensures the prototypes of the same class are projected closer to their corresponding class centers. For more details of the principle of MCR^2 and linear discriminative representations (LDRs), we refer the readers to Section 3.2.5 of Chapter 3.

4.2.2 Weakly-Supervised Few-Shot Learning

Although coarse labels are more affordable compared with fine-grain ones, directly training on coarse labels may lead to sub-optimal local minima for a DL task due to information deficiency [45]. Researchers have devoted many efforts to appropriately incorporate coarse labels to assist representation learning for FSL [149–151].

Under a setup where only coarse labels are available for training, the authors of [150] designs a new architecture: the Visual-Semantic Meta-Embedder (VSME). VSME bridges the gap between coarse and fine granularity by learning an optimal embedding mapping from the pseudo fine labels on the training set generated via clustering. Operating in the configuration where the base training set contains both coarse and fine labels, the Parent-Aware Self-training (PAS) representation learning approach implemented in [151] aims at tackling the fine-grain classification problem where query images come with coarse labels. Similar to VSME, this approach trains a teacher classifier for pseudo-label generation and a student model for representation learning. The teacher classifier learns from the fine-grained base data and is used to "pseudo-label" query images for the student model. Next, the student model learns from both the base data and the query data labelled by the teacher classifier and its feature extractor is used for downstream classification. In another work [149], the authors assume that a coarse-to-fine category graph is available and propose a two-stage training strategy including a level-wise cross-entropy training with both coarse and fine label information and a prototype propagation phase such that the prototypes for samples belonging to the same coarse class are merged in the feature space. With the additional information provided by the coarse labels, the FSL performance of their CNN prototype extractor and K-nearest neighbor (KNN) predictor is boosted.

The existing works mentioned above either rely on the E2E task-specific supervised learning, or focus on transferring the knowledge of classifying fine categories within a coarse group to unseen target groups. These methods are proposed with an anticipation that the models would automatically learn the optimal feature extraction by task-oriented training. But it remains unclear that whether these methods address the generalization problem in FSL. In contrast, we propose to leverage the auxiliary randomly-generated coarse-grained information via the LDR loss. By the guidance of this regularization term, the few-shot learner will be directed to obtain more linear discriminant representation mappings for unseen fine classes. We have observed that this enhancement can establish better performance than [43] in cases where the model experiences more severe overfitting.

4.3 Method: Weakly-Supervised LDR-Guided Few-Shot Learning via Coarse Labels

4.3.1 Learning Objectives

We note that [43] has proposed an effective way to incorporate LDR loss in the FSL framework to obtain inter-class discriminative and intra-class compressive prototypes and has achieved improved test accuracy. It has been shown that FSL can also benefit from unsupervised learning by randomly-generated labels accompanied by appropriate data augmentation [46]. However, it has been observed that the principle of MCR^2 tend to conflict with the conventional cross-entropy (CE) loss [44, 77] in the final convergence phase of models. As a result, if the LDR loss and CE loss are integrated through a linear combination, both training and test accuracy will oscillate after a certain number of training epochs. In this work, we adopt the model and workflow introduced in [43] and propose to lift the hard constraint from LDR loss by proving only coarse-grained groupings as a weak supervision instead of fine-grained ones.

Given the support set $\boldsymbol{X}_S$ and denote their corresponding latent representations as $\boldsymbol{Z}_S$, we define the weakly-supervised MCR^2 loss for a pre-defined error ϵ as:

$$\mathcal{L}_{\text{LDR-w}} = -\Delta R_{\text{w}}(\boldsymbol{Z}_S) \doteq -R(\boldsymbol{Z}_S) + R_c(\boldsymbol{Z}_S, \boldsymbol{\Pi}_{coarse}) \tag{4.1}$$

Given a query sample $\{\boldsymbol{x}_Q, \boldsymbol{c}_Q | \boldsymbol{z}_Q = f_\phi(\boldsymbol{x}_Q), \boldsymbol{x}_Q\}$ consisting of an image $\boldsymbol{x}_Q$ and a category label $\boldsymbol{c}_Q$, its classification loss can be computed by:

$$\mathcal{L}_{\text{cls}}(\boldsymbol{x}_Q, \boldsymbol{c}_Q | \boldsymbol{X}_S) = -\log \frac{\exp(-d(\boldsymbol{z}_Q, \boldsymbol{c}_Q))}{\sum_{j=1}^{K} \exp(-d(\boldsymbol{z}_Q, \boldsymbol{c}_j))}. \tag{4.2}$$

Together, we train the models with a weighted summation of the classification loss $\mathcal{L}_{\text{CE}}$ and the MCR^2 loss $\mathcal{L}_{\text{MCR2}-\text{w}}$:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{LDR-w}} \tag{4.3}$$

The weight parameter λ controls the trade-off between the two terms. Typically, a larger λ leads to relatively higher training and test accuracy but more severe overfitting. An overall diagram of the used framework as in [43] can be found in Fig. 4.1. Processed by the Encoder and the set-to-set mapping function, the support set is used to locate the centers, or the "prototypes", of each category in the embedding domain. Then, the predicted class label $\boldsymbol{x}_Q$ for the Query image $\boldsymbol{x}_Q$ is predicted by a soft nearest neighbor classifier in this common embedding space.

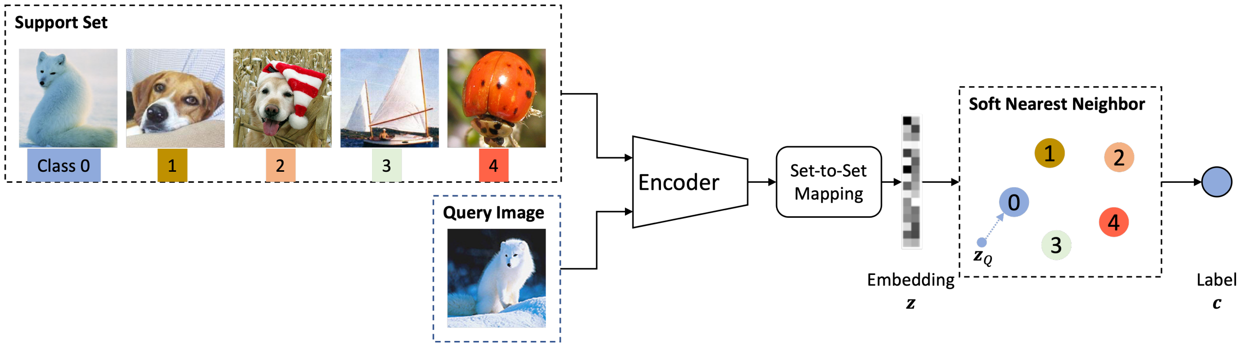


Figure 4.1: Diagram of the proposed FSL algorithm.

4.3.2 Instance Embedding Function Backbone

It is common practice to pick several empirically-designed neural network architectures that have produced promising performance after intensive trial and error as feature extraction (Encoder) backbones in FSL, such as the ConvNet and ResNet in [23, 43]. In this way, the architectures are treated as a non-interpretable black box and why they work for this specific FSL objective remains as an open question.

Concretely, it has been justified that optimizing a network to learn the low-dimensional LDR from input data in an iterative manner naturally constructs a ResNet-like structure in a recent work [152]. As a result, we believe that the ResNet is the most suitable instance embedding backbone in our work to optimize the classification and MCR^2 losses together. Actually, our experimental results show that the guidance of weakly-supervised MCR^2 loss promotes a negligible FSL performance gain, if not degraded, with ConvNet backbones, which support the statement above.

4.3.3 Coarse Label Generation

Previous works [149–151] utilize manually-assigned coarse labels for weak supervision in FSL. Such manual labeling usually merges similar objects together without considering the intrinsic patterns in the source images. As an example, both cats and lions are grouped in the “feline” class but the intrinsic patterns of their raw images, such as the background or the object sizes may have significant difference. As the training of models guided by the principle of MCR^2 relies more on these patterns instead of accurate label information, manual coarse labeling may not be the optimal choice for our weakly-supervised LDR-guided FSL framework. Thus, we consider three different ways to generate the coarse labels, including unconstrained random grouping, constrained random grouping and manual grouping. For unconstrained random grouping, we randomly split all the existing categories in training set into two even groups. For constrained random grouping, we first aggregate fine classes into various small “clusters” without overlapping, then divide these clusters into two coarse groups. Note that in the first stage of constrained random grouping, each small cluster contains a few similar fine classes that usually come with similar background and object size. For manual labeling, we directly separate fine classes into two coarse groups of live creatures and other objects. All the three coarse labeling strategies yields two coarse groups, each consisting of half of the fine classes.

4.4 Experimental Results

We provide the comparative performance of CE, hard guidance [43] and the proposed weak guidance methods with the three coarse labeling strategies in Tables 4.1 and 4.2. Particularly, Table 4.1 lists the test results without data augmentation while Table 4.1 shows the scenarios where we randomly apply one or more of four data augmentation techniques. In both tables, we refer to random coarse labeling, constrained coarse labeling and manual coarse labeling as “-R”, “-CR” and “-M” suffixes, respectively. For the instance embedding backbones, we adopt four variations of ResNet architecture, including the vanilla ResNet-12 and ResNet-18 as in [29], along with their modified versions: thinner ResNet-12 and wider ResNet-18. The numbers of filters in each layer of thinner ResNet-12 is half of ResNet-12.

Similarly, the numbers of filters in each layer of wider ResNet-18 is twice as much as ResNet-18. This allows us to explore the effectiveness of weak guidance toward LDR on different model complexities or overfitting potentials.

Table 4.1: Test accuracy (%) of proposed training method versus E2E CE training on the 1-shot 5-way FSL task, with no training data augmentation. Numbers in parentheses are accuracy changes with respect to corresponding CE cases.

Method \ Backbone	Thinner ResNet-12	ResNet-12	ResNet-18	Wider ResNet-18
CE	61.71	62.93	62.06	60.6
Hard-LDR [43]	61.98 (+0.27)	63.57 (-0.64)	62.86 (+0.80)	60.62 (+0.02)
Weak-LDR-R	61.75 (+0.04)	63.40 (+0.47)	62.92 (+0.86)	60.74 (+0.14)
Weak-LDR-CR	61.91 (+0.20)	63.33 (+0.40)	62.76 (+0.07)	60.75 (+0.15)
Weak-LDR-M	61.52 (-0.19)	63.45 (+0.52)	62.45 (+0.39)	60.79 (+0.19)

According to Table 4.1, the hard-LDR-guided [43] or the proposed weakly-supervised LDR-guided method almost consistently outperform their CE baseline when no data augmentation is applied. Particularly for ResNet-12 backbone, all three of our Weak-LDR strategies reports over 0.4% of accuracy gain over the CE baseline while Hard-LDR takes it down by 0.64%.

Overall, similar to the non-augmented results, Table 4.2 shows that a principled guidance on embedding is also helpful for downstream classification task when collaborating with data augmentation. When the models become more complex, the hard guidance starts to cause an accuracy degradation, indicating that the strict constraint from LDR loss term is not well adapted to the random distortions introduced by augmentation and is conflicting with the CE term. Meanwhile, our weakly-supervised LDR-guided method still produces robust performance improvement. Specifically, Weak-LDR-R delivers an accuracy gain of 0.59% over baseline CE while Hard-LDR brings down the accuracy by 0.53% with a wider ResNet-

18 backbone.

Table 4.2: Test accuracy (%) of proposed training method versus E2E CE training on the 1-shot 5-way FSL task, with grayscale conversion, color jitter, flipping and rotation as training data augmentation. Numbers in parentheses are accuracy changes with respect to corresponding CE cases.

Method \ Backbone	Thinner ResNet-12	ResNet-12	ResNet-18	Wider ResNet-18
CE	61.33	62.16	61.51	59.29
Hard-LDR [43]	61.42 (+0.09)	62.34 (+0.18)	61.32 (-0.19)	58.76 (-0.53)
Weak-LDR-R	61.28 (-0.05)	62.67 (+0.51)	61.55 (+0.04)	59.88 (+0.59)
Weak-LDR-CR	60.39 (-0.84)	62.62 (+0.46)	61.41 (+0.09)	59.79 (+0.50)
Weak-LDR-M	61.25 (-0.08)	62.58 (+0.42)	61.91 (+0.40)	59.24 (-0.05)

From both Tables 4.2 and 4.1, one interesting observation is that in general, Weak-LDR-R performs better than Weak-LDR-R, then followed by Weak-LDR-M, revealing that the embedding regularization term prefers label information that is less associated with human intelligence. This further confirms the statement that a learning process of LDR regularization loss relies less on accurate label information [44] hence a straightforward summation of CE loss and strict LDR guidance term can lead to accuracy degradation. However, it is also worth noting that although overfitting is alleviated and accuracy gets increased by the LDR guidance term, the test performance is still decreasing as model complexity grows due to overfitting. Neither data augmentation, nor fine/coarse LDR guidance provides a complete fix for this well known problem.

Consider the fact that some of the augmentation methods we adopt have more practical meanings than others, such as flipping and rotation, we evaluate the performance of every individual augmentation rule with the wider ResNet-18 backbone. "Grayscale" means 20% of the training images are randomly converted to grayscale; "Color Jitter" means the images' brightness, contrast and saturation are jittered by random factors between 0.8 and 1.2;

"Flipping" means 50% of the training images are randomly flipped horizontally; "Rotation" means each training image is randomly rotated by -45 to 45 degrees. Table 4.3 shows the classification accuracy of E2E CE baseline and the proposed weak-LDR-guided strategy, with one of the four augmentation method. Compared with the wider ResNet-18 results in Table 4.1, all four considered augmentation techniques worsens the test accuracy, this is likely because that the available information in a FSL task is highly insufficient and any additional perturbation can make the FSL task more difficult. Such negative impact is most obvious under rotation, which introduces an accuracy degradation of 3.8%, while those of others are relatively negligible (less than 0.7%). This matches our expectation as rotation is the only method that can completely change the intrinsic patterns of an image. For the same reason, the guidance of weakly-supervised LDR loss, which relies more on the internal characteristics of images than CE loss, does not demonstrate performance improvement when training images are rotated. In the meantime, the effects of the other three augmentation approaches with or without the guidance of weakly-supervised LDR-guided loss are similar. Overall, for FSL tasks, traditional manually-crafted augmentations shall be carefully selected since they can introduce unrealistic distortive noise, overcomplicate the task and result in an accuracy drop.

Table 4.3: Test accuracy (%) with the wider ResNet-18 backbone of proposed training method versus E2E CE training on the 1-shot 5-way FSL task, with one single training augmentation method applied. Numbers in parentheses are accuracy changes with respect to corresponding CE cases.

Method \ Augmentation	Grayscale	Color Jitter	Flipping	Rotation
CE	59.55	59.57	59.52	56.81
Weak-LDR-R	60.06 (+0.51)	59.83 (+0.26)	59.76 (+0.24)	56.64 (-0.17)

4.5 Summary

In this chapter, we expand the applications of the principle of MCR^2 to the FSL problem where the amount of training data is insufficient. Firstly, to address the common overfitting problem in FSL, we suggest to replace fine labels by coarse ones in the LDR regularization loss as a relaxation of the strict fine grouping constraint in the FSL pipeline proposed in [43]. By introducing this simple tweak to the baseline [43], our weakly-supervised LDR-guided FSL framework further reduces overfitting and shows accuracy improvement for more complex models which are more prone to overfitting. To make the exploratory study more comprehensive, we evaluate a total of three different coarse label generation methods: purely random, constrained random or manual. Interestingly, empirical evidence reveals that less human knowledge in the grouping information results in better weakly-supervised LDR-guided FSL performance. Upon these findings, we claim that it can lessen the severity of overfitting in FSL if the models are empowered to learn underlying structural meaning from data with less label bias. In addition, we look into another technique that is frequently used to alleviate overfitting: data augmentation. It is seen that data augmentation reduces overfitting at the cost of lower test accuracy in the adopted FSL framework. However, a weak guidance toward LDR can reduce the test accuracy degradation caused by data augmentation and provide potential for better model optimization by monitoring only the training performance. Additional experiments on single augmentation approach reveal that these straightforward methods, such as random rotation, may introduce unwanted and/or impractical distortion to the training set and make it harder for the classifier to extract useful information from such a limited dataset.

To conclude, we show that the weak guidance toward LDR is useful in understanding the underlying semantic features, is beneficial for the downstream tasks in practice and outperforms the end-to-end cross-entropy-based training or hand-crafted data augmentation in terms of reducing overfitting. Meanwhile, this method is compatible with any existing neural-network-based FSL learning frameworks and introduces no extra computational burden at inference.

Chapter 5

Artificial Neural Networks for Salinity Estimation and Forecasting in the Sacramento-San Joaquin Delta of California

In this chapter, we first enhance the existing domain-specific multi-layer perceptron (MLP) ANN architectures for salinity estimation in the Sacramento-San Joaquin Delta, California at key monitoring stations. We present a novel multi-task learning (MTL) ANN framework with shared hidden layers for joint salinity estimation at 12 salinity monitoring stations achieving a reduction of 90% training and inference time. Then, we replace the predetermined input data pre-processing by a trainable convolution layer, allowing more flexibility when transforming input data into lower-dimensional representations. Numerical tests based on a simulated dataset demonstrate the benefits of training time reduction, estimation error decrease, and better feature extraction performance.

With the MTL framework and inspired by the idea of residual learning in the well-know Residual Network (ResNet) [58] architecture, we further develop and apply two novel

DL models: a Residual Long-Short-Term Memory (Res-LSTM) network and a Residual Gated Recurrent Unit (Res-GRU) model to capture both the spatial and temporal variations of salinity and compare their efficacy against the baseline MLP ANN. In this phase, we expand to 23 monitoring stations in the Delta and train the models using historical salinity measurements. Results indicate that the proposed novel DL models generally outperform the baseline MLP model in simulating and predicting salinity on both daily and hourly scales at the salinity monitoring stations.

5.1 Background and Problem Formulation

Salinity management is the keystone of water resources management in estuarine environments due to the underlying biological significance and inherently high variations in space and time of salinity [39]. Understanding these variations and predicting variation patterns under different potential future scenarios lay the foundation for informed water management decision-making. This is especially true for areas with great ecological, social, and economic importance including the Delta. The Delta is the confluence of freshwater inflows from upstream rivers and saline tidal flows from the Pacific Ocean. Major streams like the Sacramento River, San Joaquin River, and eastside tributaries enter the Delta (Fig. 1.2) and the waters flow through the Delta in a complex network of intersecting channels which ultimately flow west out to the Pacific Ocean or are diverted for agricultural and municipal use inside and outside of the Delta. The salinity of water in the channels (concentration of salt measured, for example, in milligrams of salt per liter of stream water) determines the

Part of this chapter is reprinted, with permission, from [S. Qi, Z. Bai, Z. Ding, N. Jayasundara, M. He, P. Sandhu, S. Seneviratne and T. Kadir, "Enhanced Artificial Neural Networks for Salinity Estimation and Forecasting in the Sacramento-San Joaquin Delta of California" in *Journal of Water Resources Planning and Management*, Aug. 2021], [S. Qi, M. He, Z. Bai, Z. Ding, P. Sandhu, Y. Zhou, P. Namadi, B. Tom, R. Hoang and J. Anderson, "Multi-location Emulation of a Process-based Salinity Model using Machine Learning" in *MDPI Water*, 2022], [S. Qi, M. He, Z. Bai, Z. Ding, P. Sandhu, F. Chung, P. Namadi, Y. Zhou, R. Hoang, B. Tom, J. Anderson, D. M. Roh. "Novel Salinity Modeling using Deep Learning for the Sacramento-San Joaquin Delta of California", *MDPI Water*, 2022] and followup modifications for final publication..

suitability for fish and wildlife, growing crops (the Delta has approximately 420,000 acres of prime agricultural lands), and urban indoor/outdoor use. Water salinities in the Delta channels are affected by many factors including ocean tides, inflows to the Delta from inland rivers and streams, and agricultural activities/practices within the Delta. Also, human actions related to water usage, such as diverting to the Delta islands for agricultural and urban use or exports from the Delta through the State Water Project (SWP) and Central Valley Project (CVP) pumping plants, would also change flows and salinities through the mixing process. Freshwater flow releases from upstream reservoirs are managed to maintain Delta salinity at levels that support water supply and environmental needs. This requires estimates of salinity for various climate, flow, and operational conditions. To assist in the planning and management of the water resources in the Delta, the California Department of Water Resources (CDWR) has developed two key simulation models for use in planning studies: (1) CalSim, a water allocation model of the SWP and CVP systems [153], and (2) Delta Simulation Model 2 (DSM2), a hydrodynamics and water quality model [154], which is developed based upon the mathematical flow-salinity relationship model presented in [155]. We refer interested readers to an earlier paper [2] on detailed discussions of CalSim and DSM2 as tools used in water resource management and their functionalities. There are 12 key water quality monitoring stations in the Delta: Emmaton, Jersey Point, Collinsville, Rock Slough, Antioch, Mallard Island, Old River at HWY 4, Martinez, Middle River Intake, Victoria Intake, CVP Intake and Clifton Court Forebay (CCFB) Intake (see Fig. 1.2). However, running these models for long study periods under multiple scenarios can be computationally expensive.

Similar to the approach described in [2], we aim to improve salinity estimation by leveraging the seven hydrological, water quality and operation parameters, namely Northern Net flows (Sacramento River and East side Streams); San Joaquin river flows; Delta cross-channel gate operation; net Delta consumptive use; tidal energy; San Joaquin River inflow salinity at Vernalis; SWP and CVP exports via Banks pumping plant, Jones pumping plant, and

Contra Costa canal (see Fig. 1.2). We will estimate salinities at a number of measurement points which include Emmaton, Jersey Point, Collinsville and Rock Slough, among others. The input data are the (pre-processed) seven input variables and their definitions can be found in Table 5.1.

Table 5.1: Input Features to Proposed ML Models.

Index	**Input Feature Name**	**Definition**
1	Northern Flow	Sum of Sacramento, Yolo Bypass, Mokelumne River, Cosumnes River, and Calaveras River flows.
2	San Joaquin River Flow	San Joaquin River at Vernalis Flow.
3	Pumping	Sum of pumping from Banks Pumping Plant, Jones Pumping Plant, and Contra Costa Water District at Rock Slough, Old River, and Victoria Canal.
4	Delta Cross-Channel Gate Operation	Delta Cross-Channel Gate Openings.
5	Consumptive Use	Net Delta Consumptive use estimated by Delta Channel Depletion (DCD) and Suisun Marsh Channel Depletion (SMCD) models.
6	Martinez Tidal Energy	Tidal energy at Martinez, calculated as the daily maximum – the daily minimum astronomical tide at Martinez.
7	San Joaquin River EC	Electrical conductivity measured at San Joaquin River at Vernalis.
8	Sacramento River EC	Electrical conductivity measured at Sacramento River at Greens Landing.

Following [2], each of the seven variables is pre-processed via an empirical convolution process that converts the values of the input at the current day plus the antecedent 117 days into 18 values, including one value from each of the current day plus the most recent seven antecedent days along with 10 non-overlapping 11-day averages. Fig. 5.3 outline the pipeline to obtain the estimated salinity levels in [2]. The complete pipeline in mathematical notation is given in Fig. 5.1. For training and validation, we have access to monthly input data and daily salinity data covering water years 1941-2015. In California, each water year cycle runs from October 1 to September 30 of the following calendar year.

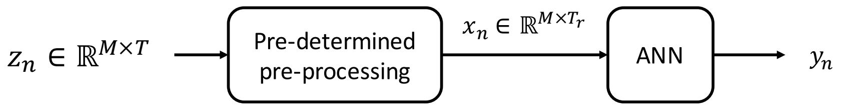

$z_n \in \mathbb{R}^{M \times T}$
Pre-determined pre-processing
$x_n \in \mathbb{R}^{M \times T_r}$
ANN
y_n

Figure 5.1: Pipeline for ANNs with mathematical notations according to [2]

There is a total of N data samples (or days) in the dataset. In our problem, we select $M = 7$ observation variables. Same as in CalSim [2], we pick $T = 118$ and $T_r = 18$ in the baseline case and pre-process the data as denoted in Fig. 5.2.

For input variable m on day n, we extract 8 daily values:

$$\boldsymbol{x}_{n,i}^{(m)} = \boldsymbol{z}_{n-i+1}^{(m)}, \tag{5.1}$$

where $i \in \{1, \dots, 8\}$. We also compute a total of 10 successive but non-overlapping 11-day

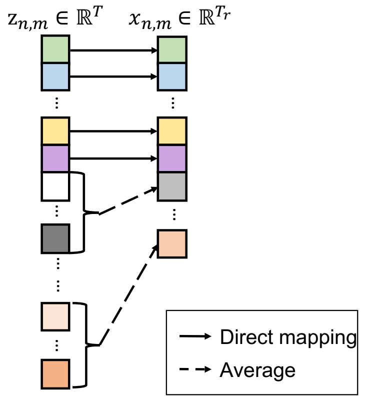


Figure 5.2: Pre-processing diagram

moving averages before the first daily data $\boldsymbol{x}_{n,i}^{(m)}$, $i \in \{1, \ldots, 8\}$ to be stored in

$$\boldsymbol{x}_{n,i+8}^{(m)} = \frac{1}{11} \sum_{j=1}^{11} \boldsymbol{z}_{n-11i-j+4}^{(m)}, \tag{5.2}$$

where $i \in \{1, \ldots, 10\}$. Altogether, for the M variables in each day n, we form $M \times T_r = 7 \times 18 = 126$ values as the $M \times T_r$ input matrix $\boldsymbol{x}_n$ to the ANNs.

Later for exploring a different ANN architecture to bypass this rather *ad hoc* pre-processing, we would form a trainable convolution layer instead of applying the above pre-determined pre-processing steps. In that case, those 118 daily values of each of the seven variables are directly provided to the convolution layer. The corresponding details will be described in Section 5.3.

The target outputs of ANNs are the salinity levels at one or more monitoring stations. Each STL ANN's output is salinity level at one single monitoring station, while each MTL ANN's outputs are salinity levels at all monitoring stations.

Different from the previous study [2], the current work randomly split 80% and 20% of this dataset for training and validation, respectively.

5.2 Multi-task Learning

Jayasundara et al. (2020), for the first time, have developed and applied individual MLP ANNs consisting of one input layer, two hidden layers, and one output layer, in simulating salinity based on seven variables in the Delta, including water control gate operations, water exports, tidal stage, as well as flow and salinity boundaries, to emulate DSM2 within CalSim 3, making runtimes much more practical. However, it is not efficient to train and inference those 12 separate ANNs. In the context of our objective for simultaneously estimating salinity levels at multiple monitoring stations based on the same set of inputs, we can view this problem as a special case of multi-task learning (MTL). This formulation is motivated by the fact that the salinities at the multiple monitoring stations are all affected by the same set of hydrological measurements within the same regional ecosystem.

MTL, in contrast to single-task learning (STL), is a machine learning strategy where multiple tasks sharing commonalities are solved simultaneously. As shown in [156–158], the domain-specific information contained in input data may allow one task to "eavesdrop" on features discovered for other related tasks and may lead the model to prefer some hypotheses over others. By leveraging the domain-specific information, MTL helps improve neural networks' efficacy and generalizability. One of the most commonly used MTL methods is known as *hard parameter sharing*, which is achieved by a joint architecture that requires multiple tasks to share some hidden layers while keeping several task-specific layers towards the end of model for each task [156]. The idea of hard parameter sharing has been applied to time series prediction such as rainfall amount prediction[159] and water quality forecasting[160]. We design the MTL ANN for simultaneous estimation of salinity at multiple monitoring stations and this new paradigm enables the ANN to better extract the underlying data features and generate better overall performance than the current STL model individually trained and optimized for each monitoring station [2].

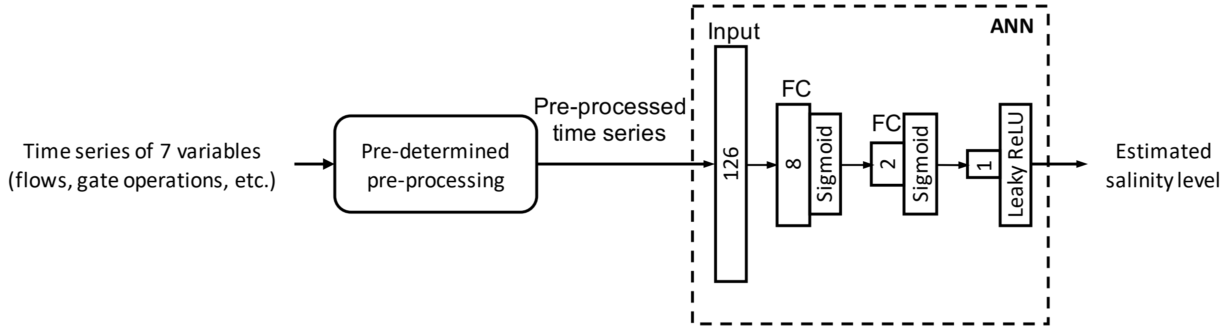

Time series of 7 variables
(flows, gate operations, etc.)
Pre-determined pre-processing
Pre-processed time series
ANN
Input
126
FC
8
Sigmoid
FC
2
Sigmoid
1
Leaky ReLU
Estimated salinity level

Figure 5.3: Complete pipeline for ANNs according to [2]

As described in [157], multiple inter-related tasks may be learned jointly by training a single ANN. The output layers shall include more neurons whereas the hidden layers are shared by the monitoring stations. These hidden layers together serve as a joint mechanism for feature extractions that can be used more consistently to generate salinity estimates at different monitoring stations. With MTL, an ANN can show better general performance over multiple disjoint single-task ANNs. As shown in Fig. 5.4, the MLP architecture proposed in [2] consists of two fully connected (FC) hidden layers and one output layer, with each layer containing 8 neurons, 2 neurons and 1 neuron, respectively.

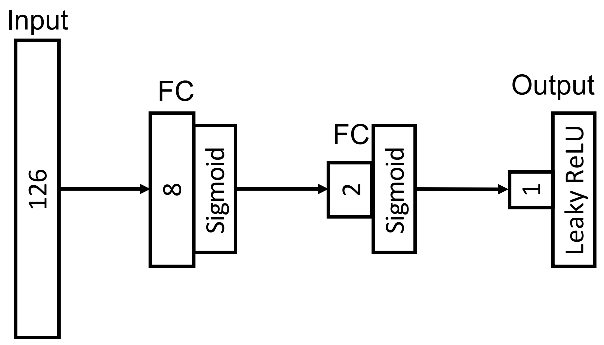


Figure 5.4: Architecture of a single-task ANN

Based on the model in previous successful STL ANNs in Fig. 5.4, we build the multi-task ANN architecture, which is an MLP network containing two hidden layers with sigmoid activation functions and one output layer with a Leaky ReLU [161] activation function. As illustrated in Fig. 5.5, we increase number of neurons by a factor of 12, which coincides with the number of monitoring stations in the first part of this chapter, for all layers to build the multi-task ANN, that is, the two hidden layers and output layer in multi-task ANN contain 96, 24 and 12 neurons respectively.

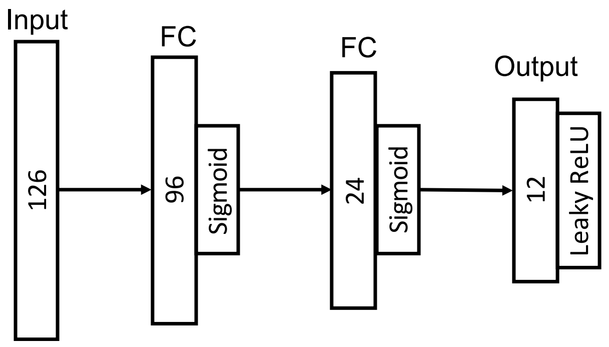


Figure 5.5: Architecture of a multi-task learning ANN

5.3 Trained Input Pre-processing via a Convolution Layer

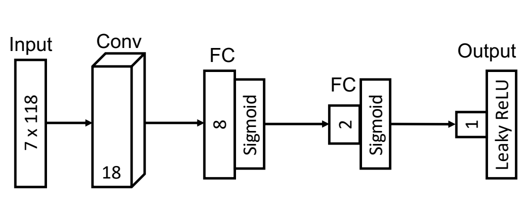
As discussed earlier, the authors of [2] utilized 8 newest daily values together with 10 non-overlapping moving averages of the daily values immediately before the 8 daily values as input data for salinity estimation (Fig. 5.2).

It should be recognized that the reported direct daily mappings and moving window averages are special cases of convolution processing, except that the existing pre-processing is not optimized through data training. Understanding the shortcomings of such a heuristic pre-processing, we propose instead to include a trainable convolution layer for data pre-processing in our novel ANN architecture. Mathematically, the convolution layer would implement the following data processing through the training weights $f_{j,i}^{(m)}$:

$$\boldsymbol{x}_{n,i}^{(m)} = \sum_{j=1}^{T} \boldsymbol{z}_{n-j+1}^{(m)} \times \boldsymbol{f}_{j,i}^{(m)}, \tag{5.3}$$

where $n \in \{1, \ldots, N\}$, $m \in \{1, \ldots, M\}$ and $i \in \{1, \ldots, T_r\}$. Clearly, by appropriately setting the convolution weights $f_{j,i}^{(m)}$, the convolution layer is capable of delivering daily value mapping and sliding window averaging. Moreover, this convolution layer is trainable in conjunction with the additional layers in the ANN. The inclusion of the convolution layer within the ANN allows the weights in this layer and other ANN layers be jointly optimized to achieve better overall performance.

By including the convolution layer, the two respective novel architectures of single-task and multi-task ANNs with a convolution layer are shown in Fig. 5.6. There are $T_r = 18$ filters in a convolution layer such that the convolution layers are able to extract at least the same 8 daily values and 10 average values in the pre-determined pre-processing. The complete pipeline with proposed convolution layer and the MTL ANN can be found in Fig. 5.7.



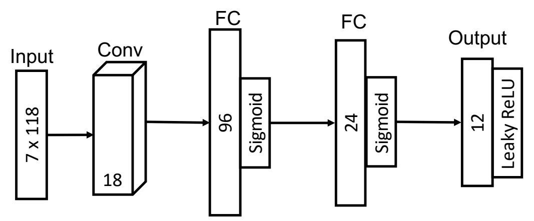


Figure 5.6: STL (left) and MTL (right) ANN architectures with a convolution layer.

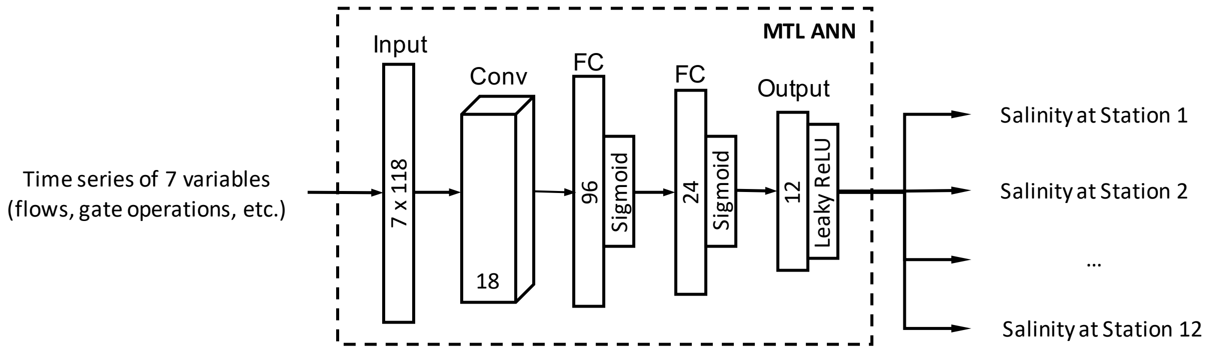


Figure 5.7: Complete pipeline for proposed MTL ANNs

5.3.1 Implementation

We implement the newly developed ANNs using the popular open source library, Tensorflow 2.2.0 [75], with Python 3.6.9. We conduct the experiments through web-browser on Google Colaboratory, which is a cloud-based Jupyter notebook environment with a Tesla T4 GPU. We normalize inputs and outputs to the range [0.1, 0.9] by linearly converting the i-th daily value of the k-th input variable in the n-th data sample $\boldsymbol{x}_{n,i}^{(m)}$ to

$$\widehat{\boldsymbol{x}}_{n,i}^{(m)} = \frac{\boldsymbol{x}_{n,i}^{(m)} - \left(\min_{k=1,\ldots,N} \boldsymbol{x}_{k,i}^{(m)}\right)}{\left(\max_{k=1,\ldots,N} \boldsymbol{x}_{k,i}^{(m)}\right) - \left(\min_{k=1,\ldots,N} \boldsymbol{x}_{k,i}^{(m)}\right)} \times 0.8 + 0.1. \tag{5.4}$$

We apply the same normalization to outputs $\boldsymbol{y}_n$ representing the salinity at a monitoring station on day n.

$$\widehat{\boldsymbol{y}}_n = \frac{\boldsymbol{y}_n - \left(\min_{k=1,\ldots,N} \boldsymbol{y}_k\right)}{\left(\max_{k=1,\ldots,N} \boldsymbol{y}_k\right) - \left(\min_{k=1,\ldots,N} \boldsymbol{y}_k\right)} \times 0.8 + 0.1. \tag{5.5}$$

The cost function used for training is the Mean Squared Error (MSE). For the LM optimizer, we adopt the same settings as [2], where the starting learning rate is 0.005 and decay factor is 10 and the training takes 150 epochs. For the Adam optimizer, the learning rate is determined using a grid search. The starting learning rate is 0.01, and it is scaled by 0.1, 0.01, 0.001 and 0.0005 at epochs 80, 120, 160 and 180, respectively, and the training takes 200 epochs.

5.4 Residual Recurrent Neural Network (Res-RNN) Architectures

Despite their scientific advances and practical values, the ML models introduced in the first part of this chapter generally have four limitations [162]. Firstly, they are applied in simulating salinity under different planning scenarios. The forecasting capability of the ML models is largely unexplored. Reliable and intelligent forecasting is one major practical application for water resources studies. Over the past decades, ML methods have gained more popularity in this area [163, 164], due to their ability to handle big data at different scales as well as their flexible structure to identify non-linear and complex relationships between input and output data.

Secondly, the ML models focus on daily or coarser temporal scales probably due to prohibitively expensive computing requirement associated with finer time scales. However, sub-daily scales (e.g., tidal scale, hourly scale) are also meaningful to water resources planning and management practices in the Delta. For instance, farmers may need to make water diversion schedules on when to pump water from Delta channels to irrigate their crop lands during a day. Understanding sub-daily variations of salinity would help inform their relevant decision-making to avoid diverting salty water that may have a detrimental effect on crops.

Thirdly, the ML models are trained using salinity simulations from a process-based model, DSM2. Simulated data are generally "noise-free" as they follow the physical laws embedded in the advection-dispersion governing equations hardwired in process-based models. This characteristic makes it straightforward for ML models to learn the underlying patterns or signals in simulated data. This limits the application of those ML models for certain applications including forecasting. To forecast the spatial and temporal variations of salinity in the near future, it would be ideal that the ML models are trained and tested using field observations directly so that they can be utilized to predict what would happen in the field. These field observations reflect the real-world salinity conditions containing information not captured by process-based models, which are, at most, simplified representations of reality.

We attempt to tackle these highlighted limitations by proposing two novel ML models: Res-LSTM and Res-GRU, which are less complex but more efficient compared to their vanilla versions (i.e., LSTM and GRU). In this part of our work, we utilize salinity observations as the target to train the ML models and assesses their performance on both daily and hourly time scales. Moreover, we explore the forecasting capability of the two proposed novel Res-RNN models.

5.4.1 Forecasting Setup

In our previous work [49, 162], we focused only on the investigation of same-day salinity estimation (i.e., the lead time is zero). In practice, forecasting near-term salinity is critical to informing real-time water management decision-making. In this work, we extend the scope of proposed Res-LSTM and Res-GRU models to salinity forecasting up to 14 days into the future (lead time equals 14 days). Specifically, one ML model is trained for each lead day. A total number of 14 Res-LSTM models and 14 Res-GRU models are developed.

For salinity forecasting on day t with a lead time of t_l, we perform the following pre-processing steps:

Step 1: We prepare model inputs the same way as discussed in Section 5.1, which consists of $\hat{x}_i^t, \ldots, \hat{x}_i^{(t-7)}$ $(1 \leq i \leq 8)$ and $\overline{\hat{x}_i^{(t-8)\to(t-18)}}, \ldots, \overline{\hat{x}_i^{(t-107)\to(t-117)}}$.

Step 2: We formulate the target output values by shifting the salinity values forward by t_l days, represented by $y_k^{t+t_l}, k = 1, 2, \ldots, 23$.

In this way, after training, the models shall be capable of predicting daily salinity levels at the 23 monitoring stations ahead of time by t_l days.

In the remaining of the paper, ML models trained with a lead time of zero ($t_l = 0$) are referred to as "salinity estimation" models, while models trained with a lead time of greater than or equal to 1 ($t_l \geq 1$) are referred to as "salinity forecasting" models. It is worth noting that forecasting models here differ from models applied in real-time forecasting operations which use forecast model inputs to drive the model and generate forecasts. The forecasting models developed for each lead time (i.e., day 1 through day 14 into the future) in the current study use purely historical data up to the lead time of zero.

5.4.2 Evaluation Metrics

The proposed models are trained with the Adam optimization algorithm [73] based on the widely used mean squared error (MSE) loss function. Four statistical evaluation metrics, consisting of square of the correlation coefficient (r^2), Bias, root mean standard deviation ratio (RSR), and Nash-Sutcliffe Efficiency coefficient (NSE), are employed to assess the ML model performance. Each of the four metrics evaluates modeled salinity performance from a different perspective: r^2 quantifies the strength of the linear relationship between modeled salinity and the target salinity; percent bias indicates whether the models over- or underestimate the salinity; RSR is a standardized representation of the root mean squared error (RMSE) between model outputs and targets; NSE compares the predictive capacity of the models with the global mean of target sequences. For r^2 and NSE, a value close to 1 indicates desirable model performance. For percent bias and RSR, a value close to 0 designates good model performance. Table 5.2 provides detailed descriptions and definitions of these five metrics. Here, S represents the salinity sequence, $\bar{S}$ indicates the overall average value of the salinity levels S, the subscripts ANN and $Observed$ indicate ANN-estimated and observed salinity, respectively.

Table 5.2: Study Metrics.

Name	Definition	Formula
MSE	Mean Squared Error	$\text{MSE} = \sum_{t=t_l+1}^{T} \left(S^t_{Observed} - S^t_{ANN}\right)^2$
r^2	Squared Correlation Coefficient	$r^2 = \left(\frac{\sum_{t=t_l+1}^{T} \lvert(S^t_{Observed} - \overline{S_{Observed}}) \times (S^t_{ANN} - \overline{S_{ANN}})\rvert}{T \times \sigma_{Observed} \times \sigma_{ANN}}\right)^2$
Bias	Percent Bias	$\text{Bias} = \frac{\sum_{t=t_l+1}^{T}(S^t_{ANN} - S^t_{Observed})}{\sum_{t=t_l+1}^{T} S^t_{Observed}} \times 100\%$
RSR	RMSE-observations standard deviation ratio	$\text{RSR} = \frac{\sqrt{\sum_{t=t_l+1}^{T} \left(S^t_{Observed} - S^t_{ANN}\right)^2}}{\sqrt{\sum_{t=t_l+1}^{T} \left(S^t_{Observed} - \overline{S_{Observed}}\right)^2}}$
NSE	Nash-Sutcliffe Efficiency coefficient	$\text{NSE} = 1 - \frac{\sum_{t=t_l+1}^{T} \left(S^t_{Observed} - S^t_{ANN}\right)^2}{\sum_{t=t_l+1}^{T} \left(S^t_{Observed} - \overline{S_{Observed}}\right)^2}$

5.4.3 Implementation

Our experiments are carried out on a public platform: the Google Colaboratory. Hyper-parameters such as batch size, learning rate and numbers of epochs may affect model performance. In a different manner, we use a constant small learning rate of 0.001 with the Adam optimizer [73] to train our models and stop training if the mean squared error (MSE) on the test set does not decrease for 50 epochs. In addition, to prevent overtraining, we limit the maximum number of epochs to 5000. In this way, we do not have to specifically optimize the learning rate or the number of epochs.

5.5 Results and Analysis

5.5.1 MTL ANNs

We evaluate the performance of the proposed MTL ANN models by calculating the unitless normalized mean square error (NMSE), which is computed on the normalized salinity outputs $\hat{y}_n$ based on the validation dataset. We compare the performance of several ANN architectures.

To begin, the basic model is a 3-layer STL ANN with pre-processed input data, consisting of two hidden layers and one output layer, as shown in Fig. 5.4. We train this baseline ANN using both the LM algorithm (STL-LM) and the Adam optimizer (STL-Adam), to illustrate the effects of optimizers. Both "STL-LM" and "STL-Adam" configurations are used as baseline results for comparison.

In our proposed ANNs based on the novel MTL strategy, we consider two different architectures: (a) a basic 3-layer MTL ANN with the pre-determined data pre-processing used in the baseline model using the Adam optimizer (3-MTL) for training; and (b) a 4-layer MTL ANN with a replacement of fixed data pre-processing by a trainable convolution layer. We consider two initializations for the trainable convolution layer parameters: random (4-MTL-R) and using the pre-determined pre-processing parameters (4-MTL-P) according to equations 5.1, 5.2 and 5.3.

Results from each of the five configurations are labeled, respectively, by "STL-LM", "STL-Adam", "3-MTL", "4-MTL-R", and "4-MTL-P". Table 5.3 presents the NMSE results of the five different ANN configurations. Correspondingly, Table 5.4 evaluates their respective training and inference time (complexity). From the performance comparison, we make the following observations.

- With pre-determined data processing, the LM algorithm outperforms the Adam optimizer in training STL ANN to generate lower NMSE values than STL-Adam and 3-MTL do at all study stations, as shown in Table 5.3. However, the LM algorithm

Table 5.3: Resulting NMSE$\times 10^4$ of different ANN architectures for salinity estimation. Both inputs and outputs of ANNs are normalized.

	STL-LM (baseline)	STL-Adam (baseline)	3-MTL	4-MTL-R	4-MTL-P
Optimizer	LM	Adam	Adam	Adam	Adam
Emmaton	3.2	9.03	10.03	4.25	**2.63**
Jersey Point	5.35	14.78	16.18	5.74	**3.28**
Collinsville	5.09	15.92	15.56	6.20	**3.86**
Rock Slough	5.34	13.33	17.95	6.55	**3.69**
Antioch	**1.84**	7.85	9.73	3.50	2.60
Mallard Island	**2.18**	8.25	9.59	3.28	2.68
Old River at HWY 4	5.01	18.99	21.03	5.27	**2.71**
Martinez	**0.61**	3.15	6.66	2.53	1.63
Middle River Intake	5.21	16.71	17.72	5.20	**2.66**
Victoria Intake	6.12	15.41	16.47	5.33	**2.88**
CVP Intake	5.11	21.32	18.95	6.97	**3.94**
CCFB Intake Gate	5.64	20.57	19.38	6.23	**3.32**

requires 8 times longer training time (complexity) when compared with both STL and MTL trained with the Adam optimizer as shown in Table 5.4.

- Using our newly proposed MTL architectures with a trainable convolution layer, training with the Adam optimizer can substantially improve the NMSE performance over STL. In particular, the 4-MTL-P results are distinctly better (with smaller NMSE values) when comparing with STL-Adam at all 12 stations. The 4-MTL-P scenario outperforms STL-LM at 9 out of the 12 stations.

- The proposed 4-MTL architecture not only improves the salinity estimation performance in providing generally lower NMSE values, but also requires much shorter training time (from 8.31 hours of STL-LM to 319 seconds of 4-MTL-P) as well as much faster inference (from 8.52 ms to 1.3 ms). Therefore, applying MTL to multi-station salinity estimation tasks can clearly improve training and inference efficiency.

- In 4-MTL, a trainable convolution layer significantly reduces NMSE as this data processing layer can learn to extract data features and adapt to wider MTL ANN architec-

Table 5.4: Training time and inference time of different ANN architectures

Model Information	Architecture		
	STL-LM	STL-Adam	4-MTL-P
Number of parameters	981	981	16962
Optimizer	LM	Adam	Adam
Training time (sec. per model)	2493	315	319
Inference time (ms. per sample)	0.71	0.71	1.3
Number of models needed	12	12	1
Total training time	8.31 hrs	1.05 hrs	**319 secs**
Total inference time for all 12 stations (ms. per day)	8.52	8.52	**1.3**

ture through training. Our pre-determined initialization helps reduce the probability of being trapped in a local minimum.

- From Table 5.3, Antioch, Mallard Island and Martinez are the 3 outliers in 4-MTL-P with slightly higher NMSE values than their counterparts from the STL-LM scenario. The reason is that stations located further west are more influenced by ocean tides of high salinity and are less effected by the input flows. Indeed, we can see from Fig. 1.2 that all these three stations are in the western part of the Delta.

5.5.2 Res-RNNs

This section presents the results of the two novel Res-RNN models together with four baseline models for comparison: MLP, ResNet, LSTM and GRU. Firstly, the training and testing performance of all six models is scrutinized in terms of skill metrics described in Section 5.4.2 as well as visual inspection of modeled salinity against the corresponding observations. Next, the forecasting capability of the two proposed novel models is examined. Finally, model performance is evaluated on the finer hourly time step.

5.5.3 Model Performance on the Daily Scale

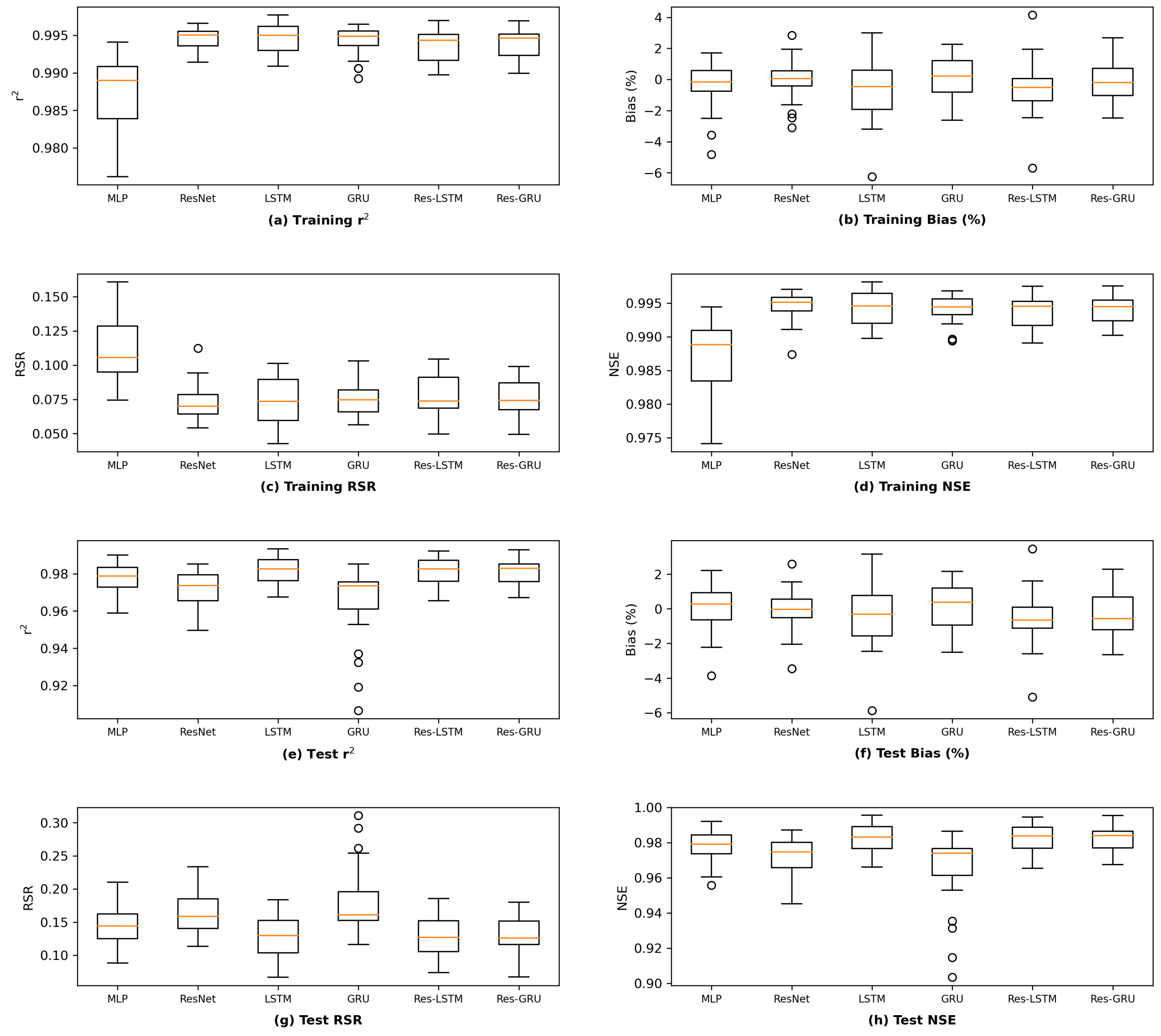


Figure 5.8: Comparison of six models on observed data at daily time step. Each box represents the interquartile range from the 25th to the 75th percentiles. The line inside each box represents the median value. The open circles represent outliers.

Fig. 5.8 illustrates the performance of the two exploratory ANNs, Res-LSTM and Res-GRU, in comparison with the original four basic networks, MLP, ResNet, LSTM, and GRU, in terms of four study metrics (r^2, Bias, RSR, and NSE). The figure reveals that the two new models (Res-LSTM and Res-GRU) have satisfactory performance. Specifically, the training results in Fig. 5.8(a–d) indicate that both Res-LSTM and Res-GRU outperform

MLP, while at similar level with ResNet, LSTM, and GRU. The former is most likely due to the learning compensation of the shortcut side branch from Res-LSTM and Res-GRU than MLP. The latter suggest that the new and simpler structures of Res-LSTM and Res-GRU could successfully achieve similar performance as their more complex counterparts LSTM and GRU. Meanwhile, the test results in Fig. 5.8(e-h) suggest that Res-LSTM and Res-GRU yield better or similar results as the four original models. All in all, Res-LSTM has slight edge over other models as it has slightly more desirable metrics. Simulations from Res-LSTM are further examined and compared with the observed salinity in different ways.

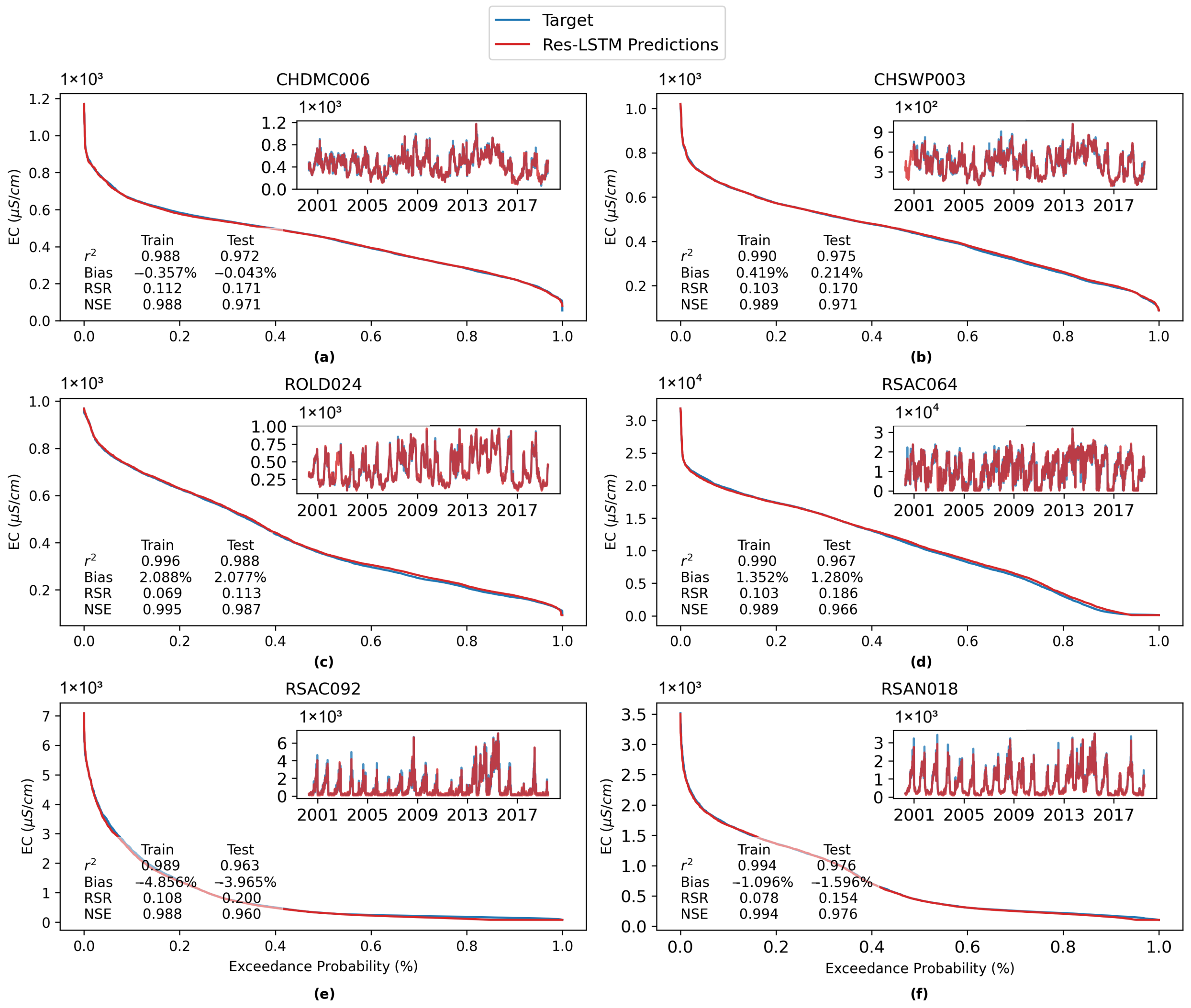


Figure 5.9: Exceedance probability plot and time series plot of Res-LSTM simulated versus observed salinity at daily time step.

Fig. 5.9 shows the corresponding exceedance probability curves and daily time series plots of Res-LSTM simulations comparing with the observed data at selected locations. Both types of plots demonstrate that the simulations mimic the target observed salinity very well, with the latter showing capture of temporal pattern and magnitude in general. Another notable pattern is that, despite the marginal discrepancies in the full salinity spectrum between the two models, time series plots reveal that Res-LSTM slightly underestimates high salinity, especially for RSAC092 and RSAN018.

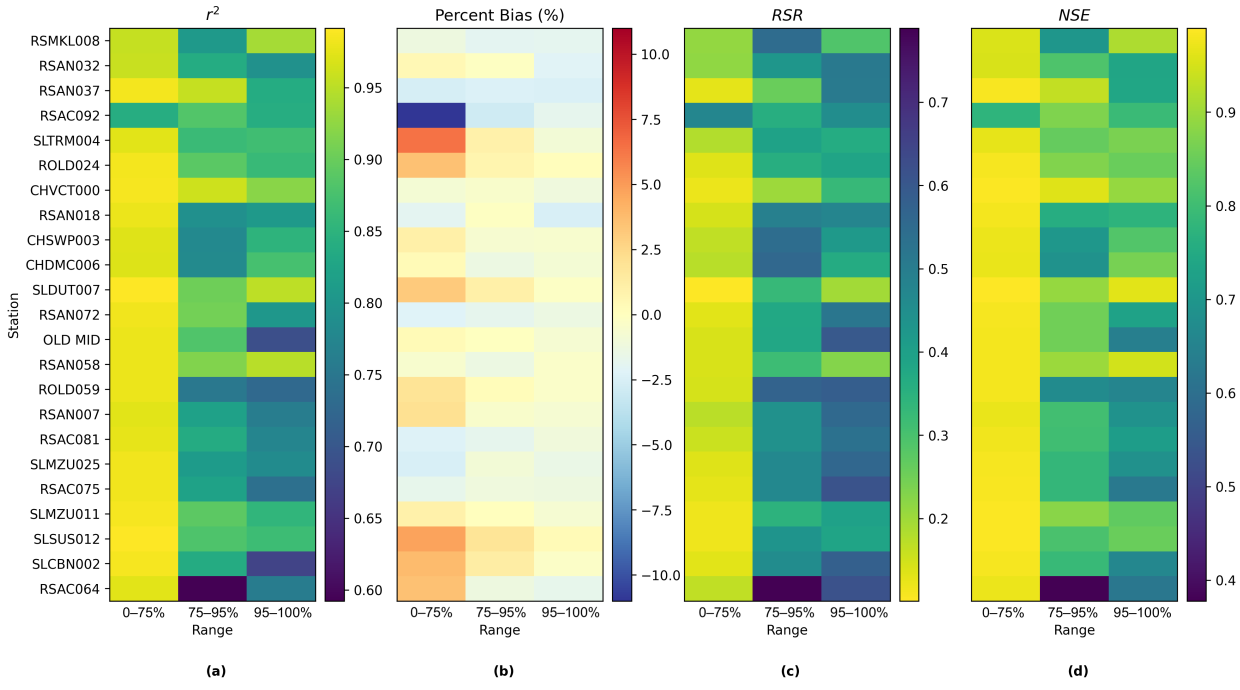


Figure 5.10: Heatmap showing Res-LSTM performance at different salinity ranges on the daily time step: low-middle range (lowest 75%), high range (75 to 95 percentile), and extreme high range (highest 5%) at the monitoring stations.

Fig. 5.10 shows the statistical metrics for each study location, calculated at three ranges, illustrating the performance of the Res-LSTM model compared to observed data on a daily time step. For metrics r^2, NSE, and RSR, "yellow" indicates satisfactory performance. For the percent bias metric, shaded blue and orange represent underestimation and overestimation, respectively. Overall, model performance is most satisfactory when the salinity is in the low-middle range (0-75%) and decreases with high (75-95%) and extremely high (95-100%)

salinity ranges. Several notable observations are further discussed below.

Performance at location RSAC092 (Sacramento River at Emmaton) is lower in the low salinity range but is consistent with the other locations in the high and extremely high ranges. Despite the departure from the other stations, overall r^2, NSE, and RSR for RSAC092 are acceptable. The Res-LSTM model underestimates salinity in the low-middle range, where the percent bias is -11%. This is because the Res-LSTM often estimates zero EC at Emmaton, but this generally does not occur in the observed data. At location RSAC064 (Port Chicago) r^2, NSE, and RSR are acceptable in the low-middle and extremely high ranges, but less satisfactory under the high range. The Res-LSTM is less able to capture the salinity variability at Port Chicago under the high range, but the percent bias is acceptable and consistent with other locations. At locations in the Suisun Marsh (SLMZU011, SLSUS012, SLCBN002), the Res-LSTM tends to overestimate salinity, especially in the low-middle range.

In short, the novel Res-LSTM and Res-GRU models can satisfactorily estimate salinity at the locations studied, while achieving similar or better performance compared with their more complex LSTM and GRU counterparts. Generally, performance is best at stations with lower median salinity and variability and degrades at stations with higher salinity and variability. The combination of a simpler architecture paired with comparably good performance to vanilla LSTM and GRU models indicate that the new models show promise in estimating Delta salinity on a daily time step.

5.5.4 Forecasting Performance

Fig. 5.11 compares the forecasting performance of the Res-LSTM model during the training (panels (a-d)) and test (panels (e-h)) runs, using box and whisker plots for four types of metrics consisting of r^2, percent bias, RSR, and NSE. Each plot includes one box and whisker for each lead time evaluated.

Generally speaking, model performance declines smoothly as lead time increases and all the metrics are within a reasonable range. Even with a lead time of 14 days, the Res-LSTM model predictions are satisfactory. For all lead times evaluated under training and testing, NSE is above 0.94, r^2 is above 0.95 and percent bias centers around zero percent, indicating excellent predictive performance without a tendency to systematically underestimate or overestimate.

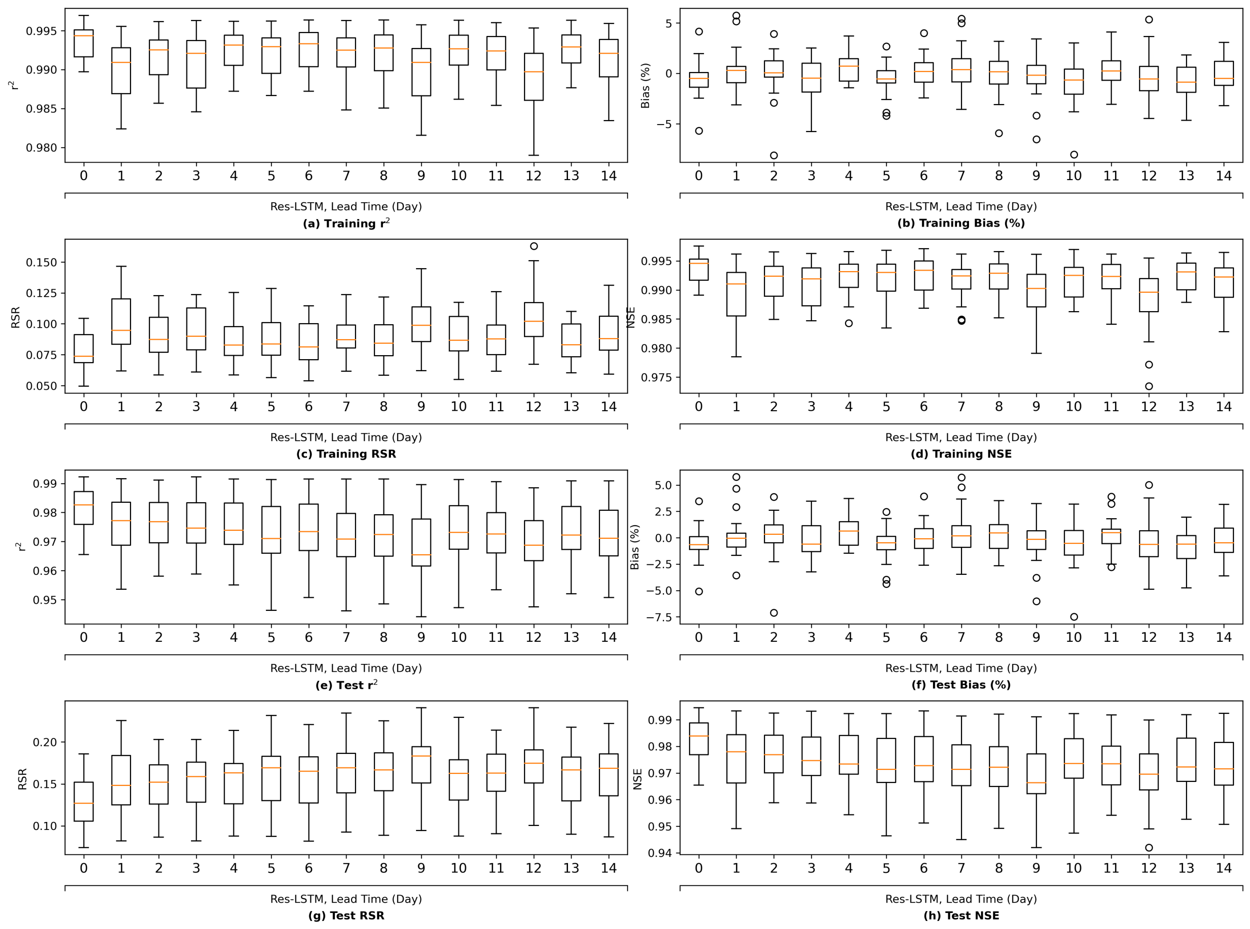


Figure 5.11: Salinity forecasting performance of Res-LSTM.

Fig. 5.12 shows the corresponding performance of Res-GRU models based on four criteria (r^2, bias, RSR, and NSE) in two rows. The first row (panels (a-d)) and the second row (panels (e-h)) display the performance of Res-GRU for training and test datasets, respectively. As a performance indicator for ML algorithms, results of the test dataset (panels (e-h) indicate that the nowcasting (forecasting with 0 lead time) model has the best performance, and the forecasting model's accuracy decreases when the lead time increases, which is reasonable for every forecasting model. However, the forecasting model with 6 and 12 days lead time does not follow this pattern, and the forecasting model with 6 days lead time provides the worst performance but still satisfactory (r^2 and NSE are high while RSR and bias are low). This suggests that historical data up to lead time 0 alone may not be ideal to forecast these two days for the Res-GRU model. Overall, all the metrics are within a reasonable range. For all lead times evaluated under training and testing, NSE is above 0.94, r^2 is above 0.94 indicating satisfactory performance overall. The percent bias metric indicates higher variability than the Res-LSTM predictions (Fig. 5.8) but does not show a clear systematic bias towards underestimation or overestimation.

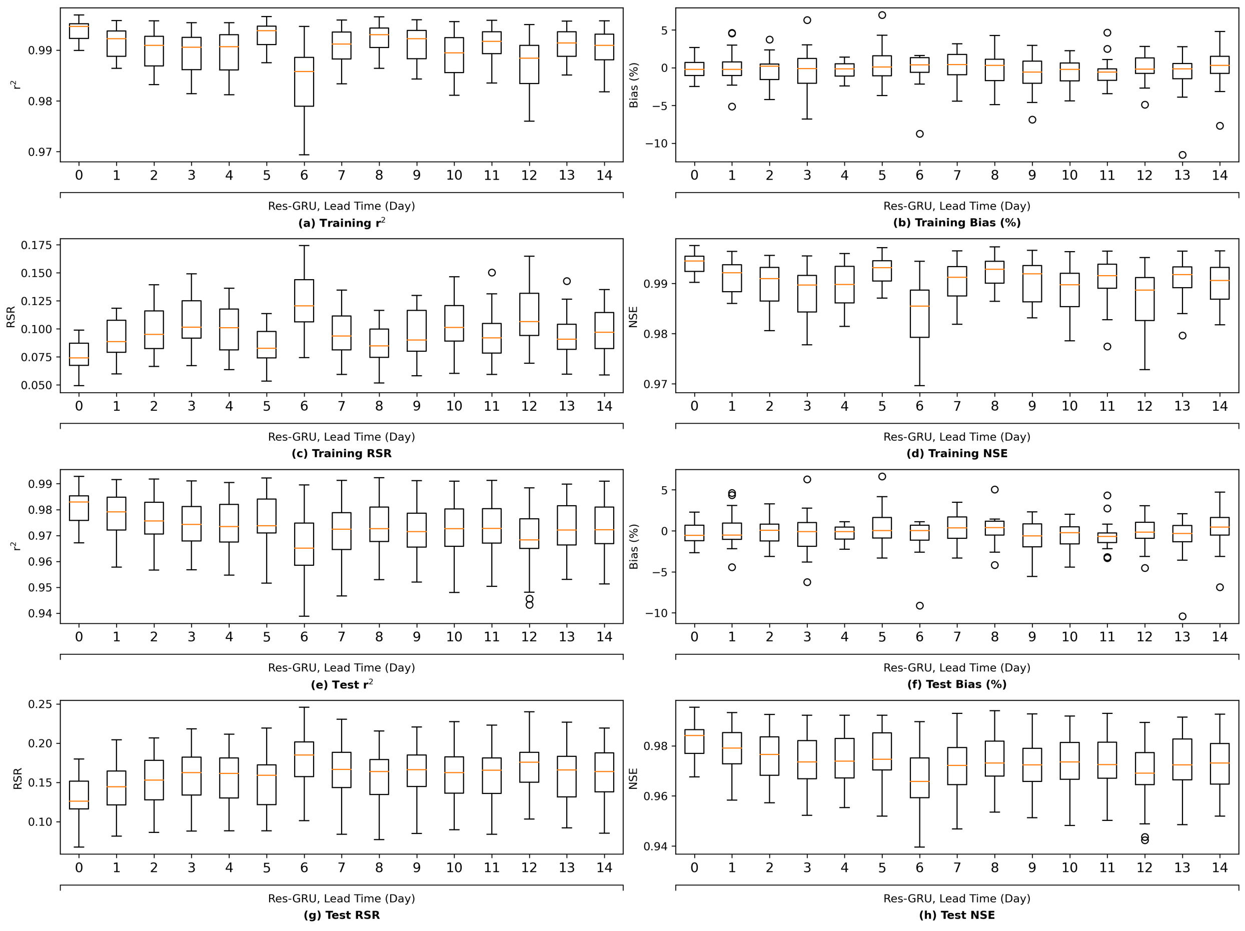


Figure 5.12: Salinity forecasting performance of Res-GRU.

All in all, for all lead times considered, Res-LSTM and Res-GRU can forecast salinity levels at all monitoring stations with satisfactory performance. Model performance generally decreases as the lead time increases.

5.5.5 Model Performance on the Hourly Scale

The results presented so far are all trained and tested using daily salinity data aggregated from the hourly observations of salinity. In this sub-section, the six ML models proposed are trained using the hourly observations directly though the input data supplied to the models are still on the daily scale. Fig. 5.13 compares the performance of these models during the training (panels (a-d)) and test (panels (e-h)) runs, using box and whisker plots for four types of metrics consisting of r^2, percent bias, RSR, and NSE. Each plot includes one box and whisker for each model evaluated. Based on these metrics, the Res-LSTM model generally outperforms all the other ML models tested during training and test runs. On average, Res-LSTM has the highest r^2 and NSE. It also has the lowest bias and RSR for both training and testing. The performance of Res-GRU is close to but not as ideal as that of Res-LSTM. In contrast, ResNet has slightly inferior performance compared to other ML models, followed by MLP. Compared to their counterparts on the daily scale (Fig. 5.8), the skill metrics r^2, RSR, and NSE are notably inferior, indicative of stronger performance on the daily (versus hourly) scale for all six models.

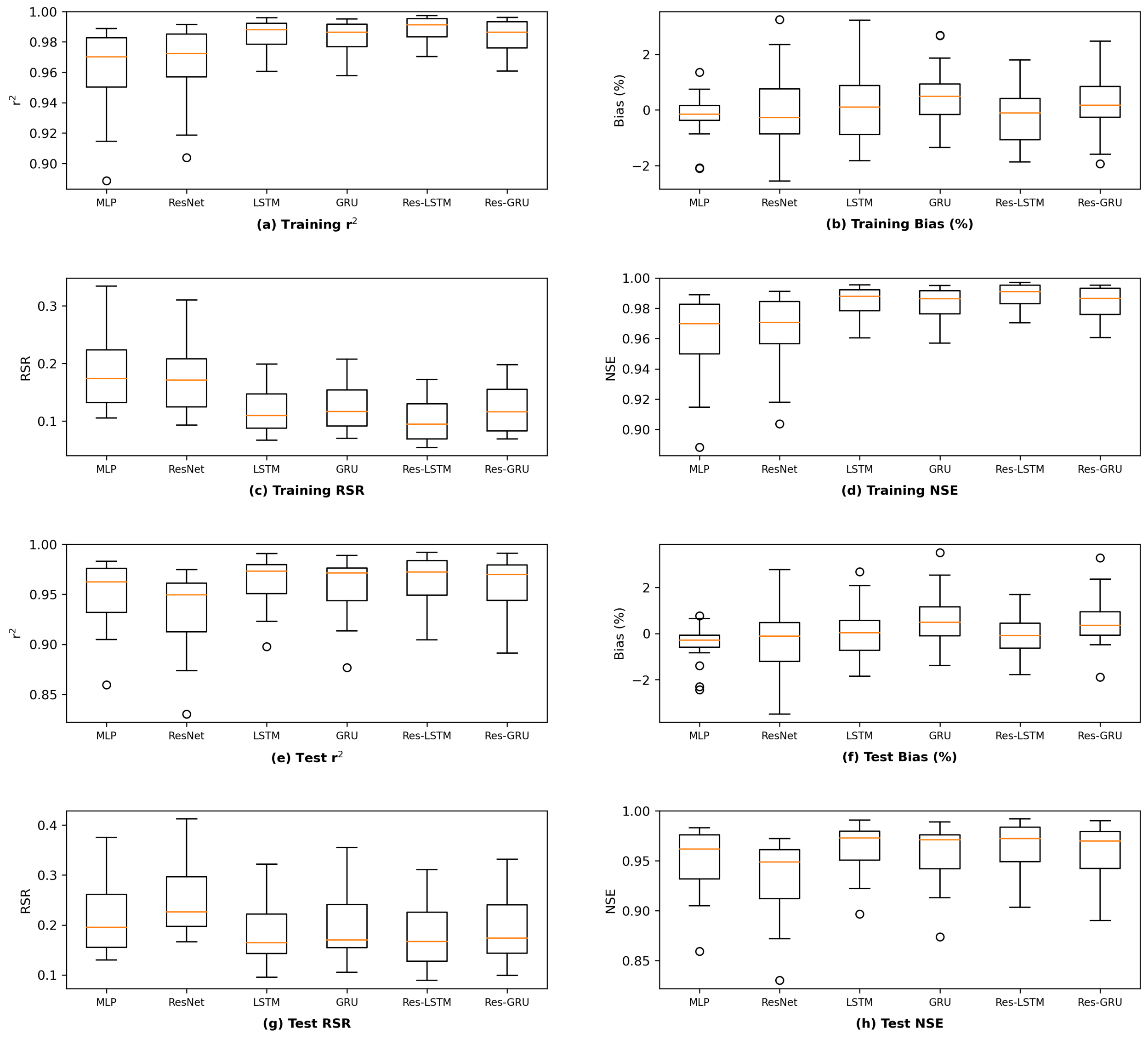


Figure 5.13: Comparison of six models on observed data at hourly time step with daily inputs.

Fig. 5.14 shows the corresponding exceedance probability curve and hourly time series plots to evaluate the performance of Res-LSTM at six selected locations in the Delta. In general, the differences between model simulations and the corresponding observations are marginal. However, the time series subplots indicate that the Res-LSTM models slightly underestimate the peak values at some of these specified locations. Nevertheless, the plots show remarkable similarity between models and observations, and the Res-LSTM model can skillfully capture the temporal pattern of observed salinity. Compared Res-LSTM performance on the daily scale (Fig. 5.9) versus the hourly scale (Fig. 5.14), the metrics associated with the daily scale are generally superior, Particularly, the r^2 and NSE are slightly higher while the RSR is generally lower on the daily scale. This is also observed for other models as illustrated in Figs. 5.8 and 5.13.

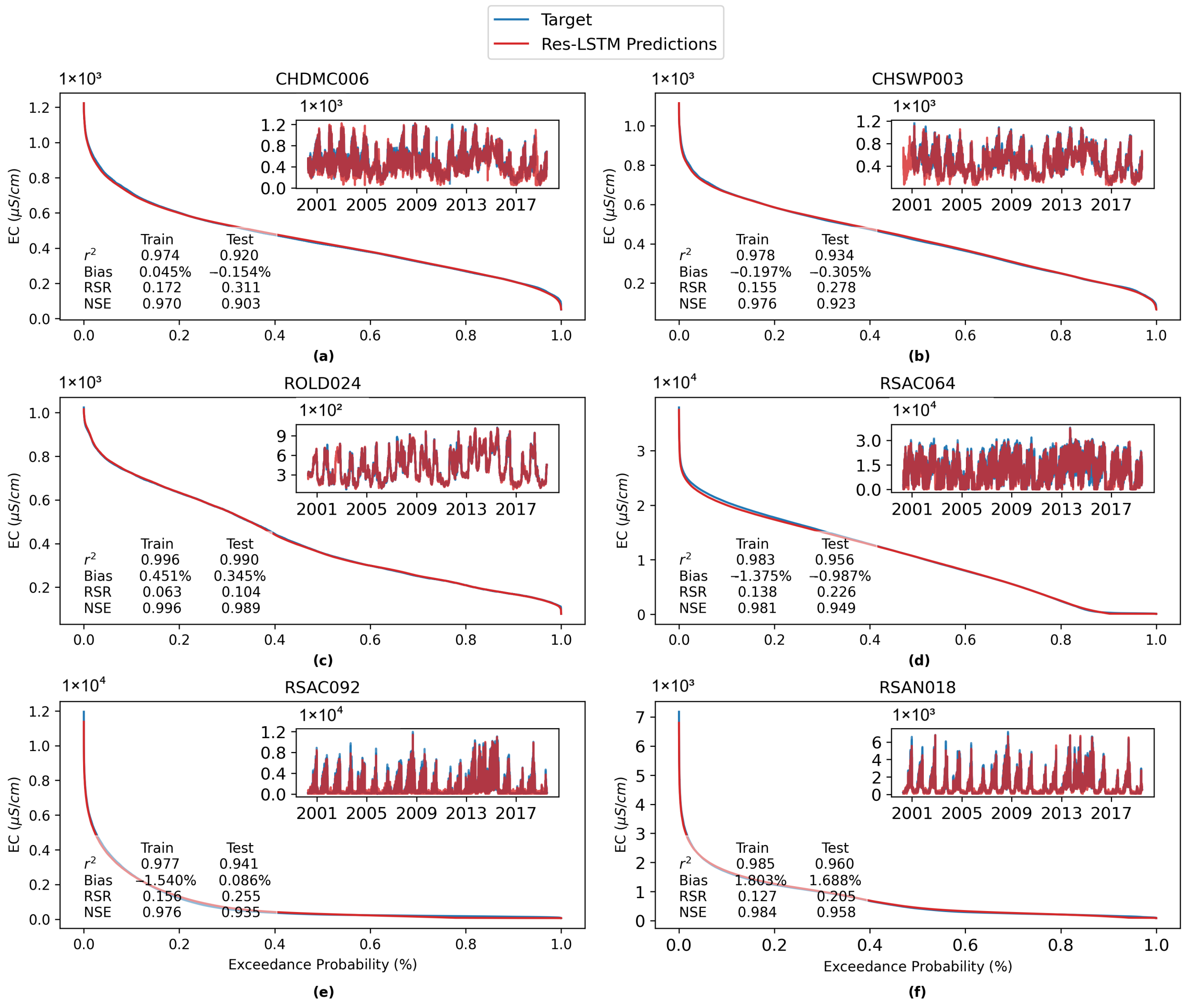


Figure 5.14: Exceedance probability plot and time series plot of Res-LSTM simulated versus observed salinity at daily time step with daily inputs.

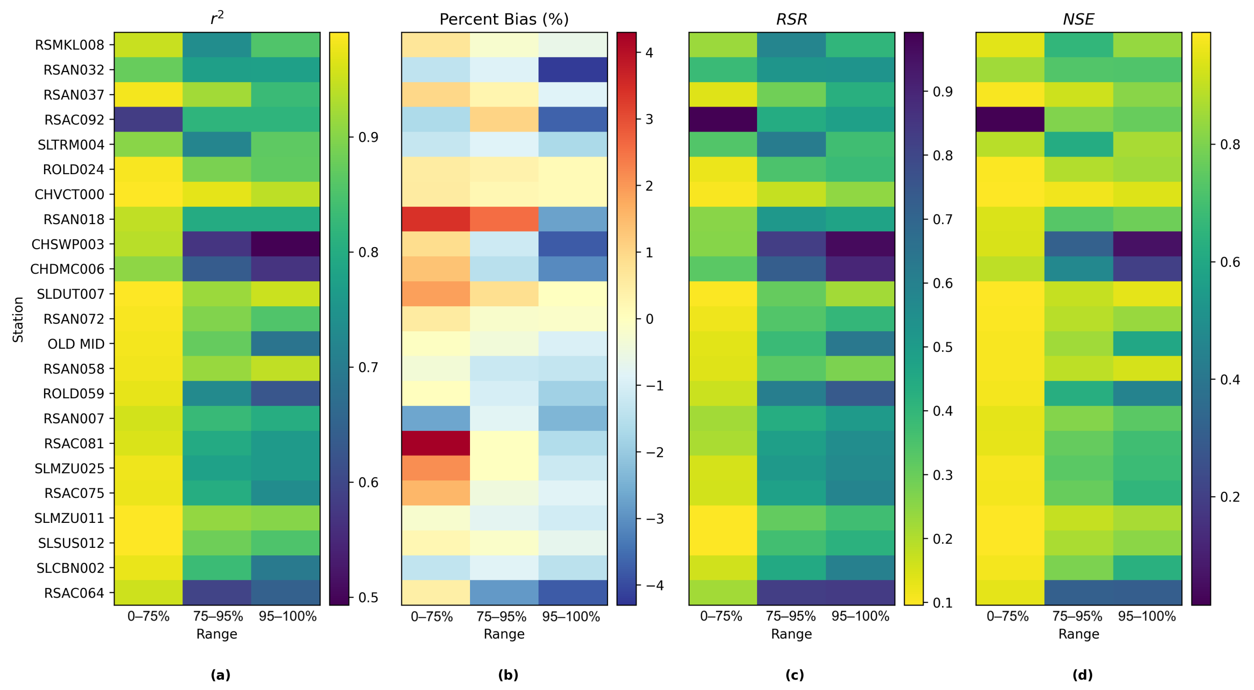


Figure 5.15: Heatmap showing Res-LSTM performance at different salinity ranges on the hourly time step: low-middle range (lowest 75%), high range (75 to 95 percentile), and extreme high range (highest 5%) at the monitoring stations.

As in Figs. 5.10 and 5.15 shows heatmaps which summarize the performance of the Res-LSTM model with hourly time steps using the statistical metrics r^2, percent bias, RSR, and NSE for each study location. In general, model performance is most satisfactory for salinities in the low-middle range across most stations, but lower for the high and extreme high ranges. Compared to the daily time step simulation results in Fig. 5.10,the metrics associated with the hourly time step are inferior for most locations.

In a nutshell, all six proposed models can achieve satisfactory performance at a finer hourly scale, and Res-LSTM slightly outperforms the other five. The differences between model simulations and the corresponding observations are small on average. The performance of Res-LSTM is highest in the low-middle range, but relatively lower for the high and extreme high ranges. Compared to their counterparts on the daily scale, the ML models on the hourly scale generally have slightly degraded performance.

5.6 Summary

In the first part of this chapter, we develop enhancements to the Delta salinity modeling ANNs for the purposes of training time reduction, estimation error reduction, and better feature extraction. The enhancements include structural redesign on two fronts: 1) incorporation of the MTL architecture and 2) addition of a convolution layer in input data pre-processing. The updated ANNs are further adapted to conduct salinity forecasting which is rarely investigated previously. The enhanced ANNs have the potential to be incorporated into the current modeling practice and provide more robust and timely information to guide water resources planning and management in the Delta.

Built upon the success of relevant previous studies that explored ML applications in salinity modeling in the Delta, we further develop two novel ML models, Res-LSTM and Res-GRU and apply them in both salinity simulation and prediction as well as on a finer hourly time scale that had never seen investigated before. Experimental results show that both novel architectures proposed can effectively simulate and predict salinity at all monitoring stations across the Delta with a moderate model complexity increase compared with the baseline MLP ANN. The effectiveness and efficiency of Res-RNNs make them viable supplements to operational process-based models in terms of providing salinity estimates to inform both real-time and long-term water management and planning practices.

Chapter 6

Conclusions and Future Work

In this chapter, we highlight the contributions of this dissertation and point out potential future research directions.

6.1 Summary

The goal of this dissertation is to adapt deep-learning-based approaches to complex problems under certain conditions in both emerging and traditional domains. Machine learning architectures and algorithms should actively take into consideration of practical limitations such as storage and communication bandwidth. Particularly, we investigate the image classification and salinity modeling problems. These approaches identify key information behind the input data, enable efficient feature extraction and improve the task performance.

6.2 Optimizing Image Compression for Classification

We explore both conventional and learning-based image compression codecs, which are originally implemented for better visual reconstruction quality. In a band-limited distributed learning setting, we embed the image codecs together with an existing classifier model, targeting at a downstream learning task: image classification. It is obvious that image classification requires less information than reconstruction, while these features shall preserve the discriminative characteristics. Inspired by the fact, we modify the architecture so that the encoder module can be guided to disregard unrelated information, cut down the transmission bit rate and increase the classification accuracy.

In Chapter 2, we propose an end-to-end image compression and classification framework integrated with the conventional JPEG encoding module in networked cloud application scenarios in the IoT systems. We claim that for lower resolution images and a shallow classifier, the quantization parameters can be easily optimized by our proposed framework and then deployed to edge devices. For large scales images, we further show that a deeper neural network is able to learn directly in the frequency domain, hence image reconstruction (JPEG decoder) can be bypassed and CPU computation load is reduced.

In Chapter 3, we move forward to a learning-based image compression codec, the auto-encoder (AE). Within the same distributed learning environment setup, some structural modifications in a traditional AE is required. With a side branch attached to the encoder during training, we guide the encoder for an in-class-compact and between-class-separable feature extraction via the principal of Maximal Coding Rate Reduction (MCR^2). During inference, this side branch will be removed hence there is no extra computational or storage cost.

We propose two hierarchical learning strategies that are compatible with the modified AE structure, where the dual tasks of compression and classification get split. We assign the sub-task of efficient feature extraction to the encoder and accurate classification to the decoder. Unlike common multi-objective learning frameworks where the objective function is the summation of various loss terms, our explicit task separation avoids the intensive tuning of trade-off hyper-parameters. In the first proposed dual-phase learning (DuPHiL) strategy, we alternate between encoder and decoder training. We observe as good performance as the baseline in this phase. Meanwhile, the encoders become more robust against model pruning, more tolerable to training label corruptions and more flexible with reconfigured decoders. In the second proposed three-step learning strategy, we further incorporate self-supervised learning as an additional phase to obtain AEs that are amenable to common image distortions. We also investigate the effects of proposed learning strategy by visualizing the discriminative power of each layer and discuss the impact of bottleneck layer dimension selection in AE-based codecs. It is worth to note that both strategies proposed in this chapter can be applied directly to existing AE architectures as a plug-and-play integration.

6.3 Weakly-Supervised Few-Shot Learning

In Chapter 4, we turn to the problem of Few-Shot Learning. With a similar structural modification as in Chapter 3, it has been proved in [29, 43] that by enhancing the between-class discriminability and within-class compactness of the embeddings extracted for subsequent classification can help addressing the unavoidable overfitting problem in FSL. Following [43], we adopt the principle of MCR^2 in training the encoder to promote such characteristics in latent embeddings but observed that the hard constraint may conflict with the label-dependent classification loss and lead to a sub-optimal local minima. As a soft relaxation, we replace the accurate grouping information in the regularization term by coarse groupings. Results show that the simple modification leads to further accuracy improvements and less overfitting, especially for complex models. In addition, we show that the weak guidance toward LDR works better when less human ingenuity is introduced in coarse labeling, which in turn proves our claim that the principle of MCR^2 relies less on accurate labeling information. Next, for exploratory purposes, we analyze the effects of the augmentation methods, including grayscale conversion, color jitter, flipping and rotation. Experimental results show that data augmentation is capable of reducing overfitting while sacrificing test accuracy. Among the four considered approaches, rotation has critical negative impact on test performance while others have only marginal negative effects. The accuracy degradation introduced by data augmentation can be alleviated when the proposed weakly-supervised LDR-guided training method is incorporated.

The proposed method is compatible with any existing neural-network-based FSL framework and brings no extra computational cost to the original model. Finally, it is worth noting that the test accuracy continues decreasing when the model's complexity rises, and no matter fine/coarse guidance toward LDR or data augmentation can completely overcome this overfitting problem.

6.4 Delta Flow-Salinity Modeling

In Chapter 5, we present an adapted framework to jointly learn input pre-processing and salinity estimation/forecasting. The task-driven enhancements include structural redesign on two fronts: 1) incorporation of the multi-task learning (MTL) idea and 2) addition of learnable input data pre-processing. We demonstrate that a three-layer MLP model with the proposed learning framework can faithfully emulate the operational process-based model DSM2 for simulation purposed. The estimation/forecasting performance can be further improved by substituting MLP backbone for a recurrent neural network (RNN) at the cost of training convergence time and inference time. To leverage the trade-offs between inference speed and estimation/forecast performance, we design a novel residual RNN (Res-RNN) architecture following the concepts of shortcut path and residual learning in ResNets [58], which can be directly utilized in the proposed enhanced framework. To build such a task-specific Res-RNN, we attach a shortcut path with a single "thin" RNN layer to the MLP model in order to capture the residuals missed by the main path. With a moderate model complexity increase, the proposed Res-RNNs can simulate/forecasting salinity faster than DSM2 and vanilla RNNs, emulate DSM2 more accurately than MLP or vanilla RNNs and outperforms DSM2 on historical data. Altogether, the flexible trainable time series processing, MTL framework and faster backbone models enable the finer resolution (hourly) real-time salinity modeling in the Delta. Overall, we demonstrate the effectiveness, efficiency and feasibility of DL-based models in supplementing the existing operational models in processing environmental variables and providing accurate real-time or long-term estimates of salinity to guide water management decision making.

6.5 Extensions

- **Reducing training data with DuPHiL strategy** We design a training strategy to train an encoder-decoder network in a distributed manner that is adaptable for a verity of applications. Relying less on the accurate grouping information, the proposed DuPHiL strategy lead to an accuracy gain compared with conventional end-to-end cross-entropy-based training strategy. In other words, theoretically, the DuPHiL strategy demands less information to obtain the same performance as the traditional methods. With that being said, the amount of available training data can be reduced and/or data augmentation can be adopted to introduce variety to image datasets.

- **Leveraging Latent Guidance and Learning Objective** We observe that the straightforward integration of two loss functions via linear combination either requires intensive hyper-parameter tuning or results in a sub-optimal local minima. In the first part of this dissertation, we propagate the regularization term via a side branch as a relaxation, but this also weakens the effectiveness of the guidance. An architecture redesign that takes the hierarchical objectives into account and leverages the latent guidance and subsequent objectives may be considered. Additionally, further works may also determine the broad appeal of this guided learning principle in other bandwidth constrained machine-learning applications.

- **Weakly-Supervised Few-Shot Learning with Coarse Labels Only** We show that a guidance on embeddings from weakly-supervised coarse labeling information contributes to a performance improvement for FSL tasks. As the classifier loss in our framework still relies on the fine labeling information, the amount of human efforts for data annotation is not alleviated. A practical enhancement to the framework is to consider only coarse labeling information in both classification loss and the embedding guidance loss. One possibility is to augment the coarse labels into pseudo fine ones with a separate coarse-to-fine classifier [165] or clustering [150]. Another interesting topic is to effectively select samples from the coarsely labeled dataset.

- **Incorporating additional information in salinity estimation** We use eight empirical variables as input features to the proposed salinity estimation models and achieve desirable performance at the selected monitoring locations. Other variables, including tidal energy, precipitation and wind speed, also influence the circulation and mixing of freshwater and sea water and thus affect the salinity level in the Delta. The impacts of considering these additional input features can be explored. Moreover, data augmentation is a technique to generate synthetic data for model training, which develops an enlarged and diversified dataset to better represent extreme conditions and possible future conditions. With the DSM2 salinity simulator, several modifications can be applied to the input variables, such as (1) scaling the magnitude of major boundary flows; (2) temporally shifting major boundary flows; and (3) changing operations of key Delta structures, such as operable gates. All of the above aim to reduce overfitting and thus improve the generalization ability of the trained neural networks.

References

[1] L. Gueguen, A. Sergeev, B. Kadlec, R. Liu, and J. Yosinski, "Faster neural networks straight from jpeg," in *Advances in Neural Information Processing Systems*, 2018, pp. 3937–3948.

[2] N. C. Jayasundara, S. A. Seneviratne, E. Reyes, and F. I. Chung, "Artificial neural network for Sacramento–San Joaquin Delta flow–salinity relationship for CalSim 3.0," *Journal of Water Resources Planning and Management*, vol. 146, no. 4, p. 04020015, 2020.

[3] M. Jankowski, D. Gündüz, and K. Mikolajczyk, "Joint device-edge inference over wireless links with pruning," in *IEEE 21st Intl. Workshop on Signal Processing Advances in Wireless Communications*. IEEE, 2020, pp. 1–5.

[4] S. Yao, J. Li, D. Liu, T. Wang, S. Liu, H. Shao, and T. Abdelzaher, "Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency," in *Proc. 18th Conference on Embedded Networked Sensor Systems*, 2020, pp. 476–488.

[5] J. Shao and J. Zhang, "Bottlenet++: An end-to-end approach for feature compression in device-edge co-inference systems," in *IEEE Intl. Conf. on Communications Workshops (ICC Workshops)*. IEEE, 2020, pp. 1–6.

[6] L. D. Chamain and Z. Ding, "Improving deep learning classification of jpeg2000 images over bandlimited networks," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 4062–4066.

[7] P. Zhou, J. Han, G. Cheng, and B. Zhang, "Learning compact and discriminative stacked autoencoder for hyperspectral image classification," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 57, no. 7, pp. 4823–4833, 2019.

[8] J. Chen, Z. Wu, J. Zhang, F. Li, W. Li, and Z. Wu, "Cross-covariance regularized autoencoders for nonredundant sparse feature representation," *Neurocomputing*, vol. 316, pp. 49–58, 2018.

[9] G. J. Bowden, H. R. Maier, and G. C. Dandy, "Input determination for neural network models in water resources applications. part 2. case study: forecasting salinity in a river," *Journal of Hydrology*, vol. 301, no. 1-4, pp. 93–107, 2005.

[10] S. Zhang, S. Cui, and Z. Ding, "Hypergraph-based image processing," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 216–220.

[11] S. Zhang, Q. Deng, and Z. Ding, "Image processing via multilayer graph spectra," *arXiv preprint arXiv:2108.13639*, 2021.

[12] A. Wijesinghe, S. Zhang, and Z. Ding, "Ps-fedgan: An efficient federated learning framework based on partially shared generative adversarial networks for data privacy," *arXiv preprint arXiv:2305.11437*, 2023.

[13] C. R. T.81, "Information technology - Digital compression and coding of continuous-tone still images - Requirements and guidelines." Standard, 1992.

[14] S.-W. Wu and A. Gersho, "Rate-constrained picture-adaptive quantization for JPEG baseline coders," in *Int. Conf. on Acoustics, Speech, and Signal Processing*. IEEE, 1993, pp. 389–392.

[15] E. Tuba, M. Tuba, D. Simian, and R. Jovanovic, "JPEG quantization table optimization by guided fireworks algorithm," in *Int. Workshop on Combinatorial Image Analysis*. Springer, 2017, pp. 294–307.

[16] X. Luo, H. Talebi, F. Yang, M. Elad, and P. Milanfar, "The rate-distortion-accuracy tradeoff: JPEG case study," *arXiv:2008.00605*, 2020.

[17] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[19] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proc. IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[21] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[22] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 116–131.

[23] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[24] B. Lake, R. Salakhutdinov, J. Gross, and J. Tenenbaum, "One shot learning of simple visual concepts," in *Proceedings of the annual meeting of the cognitive science society*, vol. 33, no. 33, 2011.

[25] S. Thrun and L. Pratt, *Learning to learn.* Springer Science & Business Media, 2012.

[26] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, 2016.

[27] B. Oreshkin, P. Rodríguez López, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," *Advances in neural information processing systems*, vol. 31, 2018.

[28] W. Li, L. Wang, J. Xu, J. Huo, Y. Gao, and J. Luo, "Revisiting local descriptor based image-to-class measure for few-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7260–7268.

[29] H.-J. Ye, H. Hu, D.-C. Zhan, and F. Sha, "Few-shot learning via embedding adaptation with set-to-set functions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8808–8817.

[30] M. H. Bata, R. Carriveau, and D. S.-K. Ting, "Short-term water demand forecasting using nonlinear autoregressive artificial neural networks," *Journal of Water Resources Planning and Management*, vol. 146, no. 3, p. 04020008, 2020.

[31] J. Bohorquez, B. Alexander, A. R. Simpson, and M. F. Lambert, "Leak detection and topology identification in pipelines using fluid transients and artificial neural networks," *Journal of Water Resources Planning and Management*, vol. 146, no. 6, p. 04020040, 2020.

[32] G. Hajgató, G. Paál, and B. Gyires-Tóth, "Deep reinforcement learning for real-time optimization of pumps in water distribution systems," *Journal of Water Resources Planning and Management*, vol. 146, no. 11, p. 04020079, 2020.

[33] P. Banerjee, V. Singh, K. Chatttopadhyay, P. Chandra, and B. Singh, "Artificial neural network model as a potential alternative for groundwater salinity forecasting," *Journal of Hydrology*, vol. 398, no. 3-4, pp. 212–220, 2011.

[34] H. Jiang, Y. Rusuli, T. Amuti, and Q. He, "Quantitative assessment of soil salinity using multi-source remote sensing data based on the support vector machine and artificial neural network," *International journal of remote sensing*, vol. 40, no. 1, pp. 284–306, 2019.

[35] J. M. Hunter, H. R. Maier, M. S. Gibbs, E. R. Foale, N. A. Grosvenor, N. P. Harders, and T. C. Kikuchi-Miller, "Framework for developing hybrid process-driven, artificial neural network and regression models for salinity prediction in river systems," *Hydrology and Earth System Sciences*, vol. 22, no. 5, pp. 2987–3006, 2018.

[36] S. Chen and C. Hu, "Estimating sea surface salinity in the northern gulf of mexico from satellite ocean color measurements," *Remote sensing of environment*, vol. 201, pp. 115–132, 2017.

[37] D. Le, W. Huang, and E. Johnson, "Neural network modeling of monthly salinity variations in oyster reef in apalachicola bay in response to freshwater inflow and winds," *Neural Computing and Applications*, vol. 31, no. 10, pp. 6249–6259, 2019.

[38] F. Zhou, B. Liu, and K. Duan, "Coupling wavelet transform and artificial neural network for forecasting estuarine salinity," *Journal of Hydrology*, vol. 588, p. 125127, 2020.

[39] M. Alber, "A conceptual model of estuarine freshwater inflow management," *Estuaries*, vol. 25, no. 6, pp. 1246–1261, 2002.

[40] Y. Rivenson, H. Ceylan Koydemir, H. Wang, Z. Wei, Z. Ren, H. Günaydın, Y. Zhang, Z. Göröcs, K. Liang, D. Tseng *et al.*, "Deep learning enhanced mobile-phone microscopy," *Acs Photonics*, vol. 5, no. 6, pp. 2354–2364, 2018.

[41] Y. Wang, S. M. A. Bashir, M. Khan, Q. Ullah, R. Wang, Y. Song, Z. Guo, and Y. Niu, "Remote sensing image super-resolution and object detection: Benchmark and state of the art," *Expert Systems with Applications*, p. 116793, 2022.

[42] L. Guo, G. Ju, B. Xu, X. Bai, Q. Meng, F. Jiang, and S. Xu, "Jitter-robust phase retrieval wavefront sensing algorithms," *Sensors*, vol. 22, no. 15, p. 5584, 2022.

[43] L. D. Chamain, S. Qi, A. Wijesinghe, and Z. Ding, "Linear discriminative representation (ldr)-guided zero-shot and few-shot learning."

[44] Y. Yu, K. H. R. Chan, C. You, C. Song, and Y. Ma, "Learning diverse and discriminative representations via the principle of maximal coding rate reduction," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[45] Y. Xu, Q. Qian, H. Li, R. Jin, and J. Hu, "Weakly supervised representation learning with coarse labels," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 593–10 601.

[46] A. Antoniou and A. Storkey, "Assume, augment and learn: Unsupervised few-shot meta-learning via random labels and data augmentation," *arXiv preprint arXiv:1902.09884*, 2019.

[47] L. Chen, S. B. Roy, and P. H. Hutton, "Emulation of a process-based estuarine hydrodynamic model," *Hydrological Sciences Journal*, vol. 63, no. 5, pp. 783–802, 2018.

[48] J. S. Rath, P. H. Hutton, L. Chen, and S. B. Roy, "A hybrid empirical-Bayesian artificial neural network model of salinity in the San Francisco Bay-Delta estuary," *Environmental Modelling & Software*, vol. 93, pp. 193–208, 2017.

[49] M. He, L. Zhong, P. Sandhu, and Y. Zhou, "Emulation of a process-based salinity generator for the sacramento–san joaquin delta of california via deep learning," *Water*, vol. 12, no. 8, p. 2088, 2020.

[50] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[51] B. Buongiorno Nardelli, "A deep learning network to retrieve ocean hydrographic profiles from combined satellite and in situ measurements," *Remote Sensing*, vol. 12, no. 19, p. 3151, 2020.

[52] T. Song, Z. Wang, P. Xie, N. Han, J. Jiang, and D. Xu, "A novel dual path gated recurrent unit model for sea surface salinity prediction," *Journal of Atmospheric and Oceanic Technology*, vol. 37, no. 2, pp. 317–325, 2020.

[53] D. Zhang, Q. Peng, J. Lin, D. Wang, X. Liu, and J. Zhuang, "Simulating reservoir operation using a recurrent neural network algorithm," *Water*, vol. 11, no. 4, p. 865, 2019.

[54] N. Thai-Nghe, N. Thanh-Hai, and N. Chi Ngon, "Deep learning approach for forecasting water quality in iot systems," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 8, pp. 686–693, 2020.

[55] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[56] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[57] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[58] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[59] S. Qi, Z. Bai, Z. Ding, N. Jayasundara, M. He, P. Sandhu, S. Seneviratne, and T. Kadir, "Enhanced artificial neural networks for salinity estimation and forecasting in the sacramento-san joaquin delta of california," *Journal of Water Resources Planning and Management*, vol. 147, no. 10, p. 04021069, 2021.

[60] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[61] S. Singh, S. Abu-El-Haija, N. Johnston, J. Ballé, A. Shrivastava, and G. Toderici, "End-to-end learning of compressible features," *arXiv preprint arXiv:2007.11797*, 2020.

[62] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," 2017.

[63] S. Diamond, V. Sitzmann, S. Boyd, G. Wetzstein, and F. Heide, "Dirty pixels: Optimizing image classification architectures for raw sensor data," 2017.

[64] X. Zou, X. Xu, C. Qing, and X. Xing, "High speed deep networks based on discrete cosine transformation," in *IEEE Int. Conf. on Image Proc.*, 2014, pp. 5921–5925.

[65] L. Chamain, Z. Ding, and S. Cheung, "Quannet: Joint image compression and classification over the channels with limited bandwidth," *IEEE Inter. Conf. on Multimedia and Expo*, 2019, in press.

[66] Z. Liu, T. Liu, W. Wen, L. Jiang, J. Xu, Y. Wang, and G. Quan, "DeepN-JPEG: A deep neural network favorable JPEG-based image compression framework," *Proc. 55th Annual Design Automation Conf.*, 2018.

[67] Z. Li, C. D. Sa, and A. Sampson, "Optimizing JPEG quantization for classification networks," *arXiv:2003.02874*, 2020.

[68] J. Choi and B. Han, "Task-aware quantization network for JPEG image compression," in *European Conf. on Computer Vision.* Springer, 2020, pp. 309–324.

[69] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," *arXiv:1703.00395*, 2017.

[70] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[71] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[72] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, 2015.

[73] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[74] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[75] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/

[76] X. Han, Y. Zhong, B. Zhao, and L. Zhang, "Unsupervised hierarchical convolutional sparse auto-encoder for high spatial resolution imagery scene classification," in *2015 11th International Conference on Natural Computation (ICNC).* IEEE, 2015, pp. 42–46.

[77] S. Qi, L. D. Chamain, and Z. Ding, "Hierarchical training for distributed deep learning based on multimedia data over band-limited networks," in *IEEE International Conference on Image Processing*, no. Bordeaux, France, 2022.

[78] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He, "Supervised representation learning with double encoding-layer autoencoder for transfer learning," *ACM Trans. Intelligent Systems and Technology*, vol. 9, no. 2, pp. 1–17, 2017.

[79] K. D. Yang and C. Uhler, "Multi-domain translation by learning uncoupled autoencoders," *arXiv preprint arXiv:1902.03515*, 2019.

[80] H.-h. Zhao and H. Liu, "Multiple classifiers fusion and cnn feature extraction for handwritten digits recognition," *Granular Computing*, vol. 5, no. 3, pp. 411–418, 2020.

[81] J. Steinhardt, P. W. Koh, and P. Liang, "Certified defenses for data poisoning attacks," in *Proc. 31st International Conf. on Neural Information Processing Systems*, 2017, pp. 3520–3532.

[82] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[83] D. Hendrycks, M. Mazeika, D. Wilson, and K. Gimpel, "Using trusted data to train deep networks on labels corrupted by severe noise," *arXiv preprint arXiv:1802.05300*, 2018.

[84] X. Zhu and X. Wu, "Class noise vs. attribute noise: A quantitative study," *Artificial intelligence review*, vol. 22, no. 3, pp. 177–210, 2004.

[85] M. Pechenizkiy, A. Tsymbal, S. Puuronen, and O. Pechenizkiy, "Class noise and supervised learning in medical domains: The effect of feature extraction," in *19th IEEE symposium on computer-based medical systems (CBMS'06)*. IEEE, 2006, pp. 708–713.

[86] D. F. Nettleton, A. Orriols-Puig, and A. Fornells, "A study of the effect of different types of noise on the precision of supervised learning techniques," *Artificial intelligence review*, vol. 33, no. 4, pp. 275–306, 2010.

[87] B. Ghojogh, M. N. Samad, S. A. Mashhadi, T. Kapoor, W. Ali, F. Karray, and M. Crowley, "Feature selection and feature extraction in pattern analysis: A literature review," *arXiv preprint arXiv:1905.02845*, 2019.

[88] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016.

[89] N. Bahadur and R. Paffenroth, "Dimension estimation using autoencoders," *arXiv preprint arXiv:1909.10702*, 2019.

[90] X. Ma, H. Wang, and J. Geng, "Spectral–spatial classification of hyperspectral image based on deep auto-encoder," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 9, pp. 4073–4085, 2016.

[91] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 5, pp. 845–869, 2013.

[92] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[93] J.-w. Sun, F.-y. Zhao, C.-j. Wang, and S.-f. Chen, "Identifying and correcting mislabeled training instances," in *Future generation communication and networking (FGCN 2007)*, vol. 1. IEEE, 2007, pp. 244–250.

[94] C. J. Pérez, F. J. Girón, J. Martín, M. Ruiz, and C. Rojano, "Misclassified multinomial data: a bayesian approach." *RACSAM*, vol. 101, no. 1, pp. 71–80, 2007.

[95] V. Mnih and G. E. Hinton, "Learning to label aerial images from noisy data," in *Proceedings of the 29th International conference on machine learning (ICML-12)*, 2012, pp. 567–574.

[96] G. Patrini, F. Nielsen, R. Nock, and M. Carioni, "Loss factorization, weakly supervised learning and label noise robustness," in *International conference on machine learning.* PMLR, 2016, pp. 708–717.

[97] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Asian conference on machine learning.* PMLR, 2011, pp. 97–112.

[98] M. S. Shelke, P. R. Deshmukh, and V. K. Shandilya, "A review on imbalanced data handling using undersampling and oversampling technique," *Int. J. Recent Trends Eng. Res*, vol. 3, no. 4, pp. 444–449, 2017.

[99] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural networks*, vol. 106, pp. 249–259, 2018.

[100] R. M. Pereira, Y. M. Costa, and C. N. Silla Jr, "Toward hierarchical classification of imbalanced data using random resampling algorithms," *Information Sciences*, vol. 578, pp. 344–363, 2021.

[101] I. Mani and I. Zhang, "knn approach to unbalanced data distributions: a case study involving information extraction," in *Proceedings of workshop on learning from imbalanced datasets*, vol. 126. ICML, 2003, pp. 1–7.

[102] S.-J. Yen and Y.-S. Lee, "Cluster-based under-sampling approaches for imbalanced data distributions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5718–5727, 2009.

[103] M. Koziarski, "Radial-based undersampling for imbalanced data classification," *Pattern Recognition*, vol. 102, p. 107262, 2020.

[104] B. Liu and G. Tsoumakas, "Dealing with class imbalance in classifier chains via random undersampling," *Knowledge-Based Systems*, vol. 192, p. 105292, 2020.

[105] M. S. E. Shahabadi, H. Tabrizchi, M. K. Rafsanjani, B. Gupta, and F. Palmieri, "A combination of clustering-based under-sampling with ensemble methods for solving imbalanced class problem in intelligent systems," *Technological Forecasting and Social Change*, vol. 169, p. 120796, 2021.

[106] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[107] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[108] J. Li, Q. Zhu, Q. Wu, and Z. Fan, "A novel oversampling technique for class-imbalanced learning based on smote and natural neighbors," *Information Sciences*, vol. 565, pp. 438–455, 2021.

[109] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. IEEE, 2008, pp. 1322–1328.

[110] S. K. Roy, J. M. Haut, M. E. Paoletti, S. R. Dubey, and A. Plaza, "Generative adversarial minority oversampling for spectral–spatial hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–15, 2021.

[111] M. Pavan Kumar and P. Jayagopal, "Multi-class imbalanced image classification using conditioned gans," *International Journal of Multimedia Information Retrieval*, vol. 10, no. 3, pp. 143–153, 2021.

[112] Y. Ma, H. Derksen, W. Hong, and J. Wright, "Segmentation of multivariate mixed data via lossy data coding and compression," *IEEE Trans. on Pattern analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1546–1562, 2007.

[113] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The jpeg 2000 still image compression standard," *IEEE Signal processing magazine*, vol. 18, no. 5, pp. 36–58, 2001.

[114] A. P. Byju, G. Sumbul, B. Demir, and L. Bruzzone, "Remote-sensing image scene classification with deep neural networks in jpeg 2000 compressed domain," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 59, no. 4, pp. 3458–3472, 2020.

[115] L. Gueguen, A. Sergeev, B. Kadlec, R. Liu, and J. Yosinski, "Faster neural networks straight from jpeg," *Advances in Neural Information Processing Systems*, vol. 31, pp. 3933–3944, 2018.

[116] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[117] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 13, no. 3, pp. 1–18, 2017.

[118] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[119] J. Zou, T. Rui, Y. Zhou, C. Yang, and S. Zhang, "Convolutional neural network simplification via feature map pruning," *Computers & Electrical Engineering*, vol. 70, pp. 950–958, 2018.

[120] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 883–894.

[121] Z. Hou and S.-Y. Kung, "A feature-map discriminant perspective for pruning deep neural networks," *arXiv preprint arXiv:2005.13796*, 2020.

[122] Q. Tian, T. Arbel, and J. J. Clark, "Task dependent deep lda pruning of neural networks," *Computer Vision and Image Understanding*, vol. 203, p. 103154, 2021.

[123] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[124] J. Ballé, V. Laparra, and E. Simoncelli, "End-to-end optimized image compression," 2017, 5th International Conference on Learning Representations, ICLR 2017.

[125] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[126] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[127] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3733–3742.

[128] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, "S4l: Self-supervised semi-supervised learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1476–1485.

[129] A. Kolesnikov, X. Zhai, and L. Beyer, "Revisiting self-supervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1920–1929.

[130] I. Misra and L. v. d. Maaten, "Self-supervised learning of pretext-invariant representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6707–6717.

[131] J.-C. Su, S. Maji, and B. Hariharan, "When does self-supervision improve few-shot learning?" in *European conference on computer vision.* Springer, 2020, pp. 645–666.

[132] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned image compression with discretized gaussian mixture likelihoods and attention modules," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7939–7948.

[133] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. J. Hwang, J. Shor, and G. Toderici, "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4385–4393.

[134] Y. Choi, M. El-Khamy, and J. Lee, "Variable rate deep image compression with a conditional autoencoder," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3146–3154.

[135] Z. Cui, J. Wang, S. Gao, T. Guo, Y. Feng, and B. Bai, "Asymmetric gained deep image compression with continuous rate adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 532–10 541.

[136] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[137] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[138] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[139] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.

[140] H. Gao, Z. Shou, A. Zareian, H. Zhang, and S.-F. Chang, "Low-shot learning via covariance-preserving adversarial augmentation networks," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[141] Y. Zhang, H. Tang, and K. Jia, "Fine-grained visual categorization using meta-learning optimization with sample selection of auxiliary data," in *Proceedings of the european conference on computer vision (ECCV)*, 2018, pp. 233–248.

[142] S. Motiian, Q. Jones, S. Iranmanesh, and G. Doretto, "Few-shot adversarial domain adaptation," *Advances in neural information processing systems*, vol. 30, 2017.

[143] M. Fink, "Object classification from a single example utilizing class relevance metrics," *Advances in neural information processing systems*, vol. 17, 2004.

[144] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, "End-to-end memory networks," *Advances in neural information processing systems*, vol. 28, 2015.

[145] S. Thrun and L. Pratt, "Learning to learn: Introduction and overview," in *Learning to learn*. Springer, 1998, pp. 3–17.

[146] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[147] R. Hou, H. Chang, B. Ma, S. Shan, and X. Chen, "Cross attention network for few-shot classification," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[148] H. Zhu and P. Koniusz, "Ease: Unsupervised discriminant subspace learning for transductive few-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9078–9088.

[149] L. Liu, T. Zhou, G. Long, J. Jiang, L. Yao, and C. Zhang, "Prototype propagation networks (ppn) for weakly-supervised few-shot learning on category graph," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 3015–3022.

[150] X. Xiang, Y. Tan, Q. Wan, J. Ma, A. Yuille, and G. D. Hager, "Coarse-to-fine incremental few-shot learning," in *European Conference on Computer Vision*. Springer, 2022, pp. 205–222.

[151] J. Yang, H. Yang, and L. Chen, "Towards cross-granularity few-shot learning: coarse-to-fine pseudo-labeling with visual-semantic meta-embedding," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 3005–3014.

[152] K. Chan, Y. Yu, C. You, H. Qi, J. Wright, and Y. Ma, "Redunet: A white-box deep network from the principle of maximizing rate reduction," *Journal of machine learning research*, vol. 23, no. 114, 2022.

[153] A. J. Draper, A. Munevar, S. K. Arora, E. Reyes, N. L. Parker, F. I. Chung, and L. E. Peterson, "CalSim: Generalized model for reservoir system analysis," *Journal of Water Resources Planning and Management*, vol. 130, no. 6, pp. 480–489, 2004.

[154] DWR-DSM2, "DSM2: Delta Simulation Model II," 2019. [Online]. Available: https://water.ca.gov/Library/Modeling-and-Analysis/Bay-Delta-Region-models-and-tools/Delta-Simulation-Model-II

[155] R. Denton and G. Sullivan, "Antecedent flow-salinity relations: Application to delta planning models," *Contra Costa Water District. Concord, California*, 1993.

[156] R. A. Caruana, "Multitask connectionist learning," in *In Proceedings of the 1993 Connectionist Models Summer School*. Citeseer, 1993.

[157] R. Caruana, "Learning many related tasks at the same time with backpropagation," in *Advances in neural information processing systems*, 1995, pp. 657–664.

[158] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[159] M. Qiu, P. Zhao, K. Zhang, J. Huang, X. Shi, X. Wang, and W. Chu, "A short-term rainfall prediction model using multi-task convolutional neural networks," in *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017, pp. 395–404.

[160] Y. Liu, Y. Liang, S. Liu, D. S. Rosenblum, and Y. Zheng, "Predicting urban water quality with ubiquitous data," *arXiv preprint arXiv:1610.09462*, 2016.

[161] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.

[162] S. Qi, M. He, Z. Bai, Z. Ding, P. Sandhu, Y. Zhou, P. Namadi, B. Tom, R. Hoang, and J. Anderson, "Multi-location emulation of a process-based salinity model using machine learning," *Water*, vol. 14, no. 13, p. 2030, 2022.

[163] M. Zounemat-Kermani, E. Matta, A. Cominola, X. Xia, Q. Zhang, Q. Liang, and R. Hinkelmann, "Neurocomputing in surface water hydrology and hydraulics: A review of two decades retrospective, current status and future prospects," *Journal of Hydrology*, vol. 588, p. 125085, 2020.

[164] A. Mosavi, P. Ozturk, and K.-w. Chau, "Flood prediction using machine learning models: Literature review," *Water*, vol. 10, no. 11, p. 1536, 2018.

[165] T. Pfister, J. Charles, and A. Zisserman, "Domain-adaptive discriminative one-shot learning of gestures," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*. Springer, 2014, pp. 814–829.