Development of a Multi-Scale Model for Enhanced Performance by Using a Modified Soft-Margin SoftMax Loss and its Application for Facial Expression Recognition

Armin Nabaei

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

April 2024

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis is prepared.

By: Armin Nabaei

Entitled: Enhancing Development of a Multi-Scale Model for Enhanced Performance by Using a Modified Soft-Margin SoftMax Loss and its Application for Facial Expression Recognition

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Electrical and Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect for originality and quality.

Signed by the final examining committee:

_____ Chair

Dr. Wei-Ping Zhu

_____ Internal Examiner

Dr. Habib Benali

_____ Internal Examiner

Dr. Wei-Ping Zhu

_____ Internal Examiner

Dr. M. Zahangir Kabir

Approved by: _____

Dr. M. Zahangir Kabir

Graduate Program Director

_____2024: _____

Dr. Mourad Debbabi, Interim Dean,

Gina Cody School of Engineering and Computer Science

Abstract

Development of a Multi-Scale Model for Enhanced Performance by Using a Modified Soft-Margin SoftMax Loss and its Application for Facial Expression Recognition

Armin Nabaei

This thesis introduces an improved Convolutional Neural Network (CNN) model that selectively excludes irrelevant elements from input vectors to extract distinct features. The model employs Convolution (Conv) to encode data information and Deconvolution (Deconv) layers to reconstruct the spatial dimensions of feature maps and utilizing shortcut connections to exploit applicable sparsity and capture comprehensive and detailed information. This research addresses the limitations of the traditional SoftMax Loss function, which tends to overfit by wrongly classifying due to shortage of discriminability through integrating a regularization technique to the conventional Cross-Entropy loss and introducing an adaptive-margin to the standard SoftMax function. The adjusted SoftMax Loss is designed to enhance the separation of different embedding vectors and tighten clusters of similar ones. This adjustment results in boosting both the diversity between different classes and the similarity within the same class. These modifications aim to elevate the proposed model's accuracy, and evaluations are benchmarked against existing methods using well-known datasets such as CIFAR10 , MNIST, and SVHN. Lastly, the application of the model is fine-tunned in the domain of facial expression recognition (FER). This work asserts the model's advanced capabilities over state-of-the-art methods in FER and demonstrated its effectiveness through superior accuracy on three established datasets: FER-2013, RAF-DB, and CK+.

Acknowledgment

I want to express my sincere thanks to my previous supervisors, Dr. M.N.S. Swamy, and Dr. M. Omair Ahmad, for their immense support and guidance. This work could not have been realized without their constant encouragement. I also extend my thanks to Dr.William E. Lynch for his excellent guidance in revising and structuring the content of thesis. Their contributions have been invaluable, and while circumstances prevented our continued collaboration to the end, their impact on this work remains. I am deeply grateful for the opportunity to develop my ideas under their guidance. Also, I like to extend my gratitude to Dr. Yousef R. Shayan, Dr. M. Zahangir Kabir, and Dr. Geoffrey Dover for substantial support and assistance in completing this thesis. I would like to acknowledge my family for their love and support throughout my entire educational life. Above all, I give my heartfelt thanks to the Creator for giving me the strength to successfully complete my work.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

List of Symbols

Chapter 1

Introduction

The exploration into convolutional networks (ConvNets) reveals a strategy to improve high-resolution detail in image analysis. By innovatively linking layers, ConvNets encourage feature reuse and oppose the diminishing gradient issue. Convolution layers, which downsampling images, can lead to a loss of resolution. The superiority of Convolutional neural network (CNN) for Image classification is well established [1], in view of ability to generate automatic features depending on the application. However, its capability to generate automatic features heavily relies on the architecture of CNN. Although deeper networks can improve accuracy, they also amplify computational demands and risk of overfitting, particularly when training data is scarce.

In this domain, the loss function used to train the network aim to balance between grouping similar items and separating different ones [2]. Also, regularization techniques like shrinking insignificant weights and assigning differential weights for misclassifications have been employed to mitigate overfitting risks. In the following, we conduct a brief review on challenges, objectives in field of image classification, and then, describe the structure of the thesis.

1.1 Motivation

Existing convolutional neural network (ConvNet) models [2], [3], [4], [5] for categorizing images often use a pathway that captures the context of the image as a feature extractor. However, the process of convolution can sometimes lose details due to the limited receptive field of the convolutions. To fix this, a model can be designed to include an expanding pathway with deconvolution layers named transposed layers that works alongside the original pathway [6]. By

doing so, it retains both the detailed and broader information, which helps in identifying complex and discriminable features.

Current Image datasets often contain a mix of 'hard' and 'easy' samples. Hard samples are those that are not easily classified and often fall close to the decision boundaries, typically due to a lack of detailed annotations, leading to under-learning. 'Easy' samples, on the other hand, may have too much repetitive information, pushing the model to become overfitted to the training data [2]. Furthermore, the standard SoftMax loss function can become less effective when faced with extremely high or low input values or when there's a significant difference between input values [7]. Notably, performance of a loss function efficiently is limited due to its poor capability to encourage in generating distinct features [8]. Improving how the loss function performs can make the optimization process better. Since, the goal of optimization is to work with a convex loss curvature to find the point of least error, a process known as convergence. But common loss functions often lead to complex, irregular loss landscapes, causing standard optimizers to struggle with poor convergence [9]. Recent enhancements to the SoftMax loss function attempt to lower the risks of divergence and overfitting by introducing regularizations and penalties [8].

1.2 Objective

This thesis introduces a deep convolutional neural network (DCNN) designed to classify objects in natural settings and recognize facial expressions. The architecture of the proposed network utilizes efficiently connected layers and optimally chosen kernels to extract detailed and relevant features. The main aim is to create a deep learning model that can generate a compact yet effective representation of data. It uses mid-level information and an auxiliary classifier [1] to improve the network's ability to distinguish between different categories.

Typical loss functions might not learn complex categories well and could lead to overlearning simpler ones. Adding extra elements to the cross-entropy loss can improve learning for tough categories and build noise tolerance to prevent overfitting [10]. In addressing this challenge, the thesis explores a modified cross-entropy loss that penalizes wrong predictions with learnable hyperparameters. The use of regularization technique in loss function during training reduces the risk of overfitting, improves the flow of gradients, enhances the network's capacity to represent complex information. The thesis examines an enhanced cross-entropy loss that uses learnable

hyperparameters to penalize incorrect predictions. Also, setting the right margin in softmax loss [11] helps with generalization by pushing the model to learn distinct features. An adjustable margin controls how discriminative the features are and has shown to be more effective, especially in image classification domain where clear separation between classes is crucial for performance. These modifications assist the model in learning features that distinguish more. With these methods, the network gets better at distinguishing between classes and grouping similar ones closely. It also handles variations in input data well and in the end, leads to more accurate results.

1.3 Discussion

The work develops a lightweight, precise model designed to tackle challenges related to information degradation and the scarcity of annotated samples in facial expression recognition. The model emphasizes generalization and enhances localization for the classification task. These aspects are fulfilled through the development of a convolutional network capable of multi-scale feature abstraction with incorporating shortcut connections which aids in enhancing gradient back-propagation and stabilizing training. Model components are structured to integrate key aspects of proven vision networks, essential for effective feature extraction and generation from limited training data. This is crucial for a model's ability to select beneficial elements from the feature vector, consequently boosting discriminability and facilitating the learning of complex features. The model's effectiveness is demonstrated through benchmarking against recognized networks on datasets like CIFAR10, MNIST, and SVHN, confirming its reliable performance.

Improvement of optimization by creating a smoother loss curvature [12] is facilitated through the incorporation of additional terms into the loss function, acting as a regularization technique. This improvement is powered by the integration of a loss function combining standard cross-entropy with a refined focal loss [13], transforming fixed constants into adaptive parameters. Experiments evaluation with traditional loss functions illustrates superior capabilities in mitigating overfitting and achieving higher accuracy. However, the standard loss function's limited capacity in promoting feature discriminability requires further refinement. Therefore, modifications to SoftMax are employed to force the model towards greater feature discrimination, thus refining loss performance. Additionally, for the classification task, the selection of an appropriate margin to define class boundaries within the SoftMax function is vital in enhancing generalization.

This work expands the range of feature representation by introducing an adaptive margin [14], in the SoftMax function as a learnable parameter. This adjustment separates dissimilar features while grouping similar ones, enhancing the classifier's ability to distinguish features more clearly. The improved SoftMax's performance evaluation, when compared to the conventional SoftMax, highlights its ability to raise classifier accuracy. Also, sample classification visualization [15] demonstrates that this method separates data points of different classes and groups those of the same class more effectively than the standard SoftMax Loss function, which improves the network's generalization from the training data.

Finally, the framework's performance evaluation against state-of-the-art models in facial expression recognition on datasets like FER-2013, RAF-DB, and CK+ shows its capability that It not only learns efficient features well but also correctly classifies hard samples from classes with few examples, proving its ability as a robust vision network.

1.4 Organization of the Thesis

This thesis is organized as follows:

• Chapter 2 reviews the literature, discussing widely-used network models in vision, improvements in SoftMax loss functions, and concludes with current facial expression learning (FEL) methods.
• Chapter 3 gives a detailed examination of the convolutional neural network (CNN) components used in classification, exploring layers, connections. It explores how combining different levels of detail helps to effectively extract useful information. and tackles training and optimization techniques to prevent vanishing gradients. This chapter details tuning training hyperparameters and evaluates the model’s performance on image classification datasets like MNIST, CIFAR10, and SVHN.
• Chapter 4 focuses on improving the model accuracy discussed in Chapter 3. It expands on changes to the Cross-Entropy (CE) Loss, the implementation of regularization to learn from challenging samples, and an innovative method for adjusting margins in the SoftMax function for clearer class distinction. The performance of these proposed changes is compared with traditional algorithms on the MNIST and CIFAR10 datasets.

• Chapter 5 offers concluding insights on applying the methods described earlier to facial expression recognition (FER) task. testing the proposed framework on recognized FER datasets: FER2013, RAF-DB, and CK+. It includes a comparison with existing FER methods.

• Chapter 6 concludes the work completed and points out opportunities for future research that build on this thesis.

Chapter 2

Literature Review

2.1 Brief Overview of Convolutional Networks for Image Classification

The architecture of Convolutional Neural Networks (CNNs) is characterized by interchanging layers of convolutions and subsampling, progressively increasing in complexity within the network's structure. As the network deepens, the number of feature maps increases while the spatial resolution diminishes. Each activation unit within a given layer may form multiple connections to the feature maps of its preceding layer.

Image processing serves as a foundational step in Computer Vision, primarily focused on the extraction of fundamental image characteristics such as edges, corners, and application of various filters. Unlike image processing, which operates on raw images without the need for prior knowledge, Computer Vision aims to produce meaningful descriptions from images [16]. The significant impact of Convolutional Networks on image recognition was highlighted by Yann LeCun in 1998 [1]. The state-of-the-art performance on image classification tasks was further advanced on CIFAR10 and NORB databases, as presented in "Improving neural networks by preventing co-adaptation of feature detectors" by G. E. Hinton [17], Subsequently, the superior generalization capabilities of CNNs for unseen tasks in Vision paradigms were showcased in "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition" by J. Donahue et al. [18]. Despite numerous explanations of ConvNets dominance in Image Processing over handcrafted methods, CNNs often remain puzzling, with no transparent understanding of their operational mechanisms or the specifics of their performance enhancements [19]. Real-world

databases are typically noisy, which complicates the use of standard ConvNets due to their need for inductive bias. Co-adaptation of feature detectors refers to a scenario where certain neurons become highly dependent, leading to significant output variations in response to random input distortions. Regularization strategies [20], [21], [22] such as the implementation of dropout in networks during training have been proposed to mitigate this issue and prevent overfitting by randomly omitting units in each layer and assigning probabilities to training cases [17].

Among vision networks, three prominent models share architectural inspirations within the ConvNet paradigm. DenseNet [23] architectures enhance the performance of ResNet [24] through strategic modifications, leveraging the strengths of its earlier model. Similarly, the foundational design of the plain ResNet model draws inspiration from the VGG [2] network's structure. The inception [25] architecture, a neural network framework codenamed for its innovative structure, leverages the Hebbian principle and multi-scale processing to dynamically define the model. This approach enables a multiplicative expansion of network resources internally. The inception concept, influenced by the 'Network in Network' paper [26], aims to augment the representational capability of the network, a feature proven beneficial as demonstrated in the ILSVRC 2014 classification challenge.

2.1.1 Utilizing Inception for Improved Regularization

In the work "Rethinking the Inception Architecture for Computer Vision" [24], the authors explore ways to make networks larger in a computationally efficient manner. They employ factorized convolutions and strong regularization to this end. The paper highlights several key principles for effective network scaling such as:

- The network's structure should avoid unnecessary complexity by cutting redundant correlations without over-compressing. A cyclic graph is used for guiding information flow and gradually reduces the input size until the final output. This method ensures that higher-dimensional information is kept available for processing, aiding in the separation of features.
- For spatial aggregation, lower-dimensional representations are preferred as they can speed up learning. This is based on the idea that reducing dimensions increases unit correlations, which helps in preserving essential information.

- Balancing the network's depth and width is crucial. The computational resources should be distributed carefully to enhance the network's quality without excessive demands on processing. Efficient convolutions are key to reducing computational demands and the number of parameters. Smart dimensionality reduction can lead to more unique parameters, saving memory and increasing the network's capacity. To avoid the high computational cost of large convolutional filters, the paper suggests using two smaller filters in sequence. For example, replacing one 5×5 filter with two 3×3 filters can yield the same results with fewer parameters and improved weight sharing.

For processing medium-sized features, the paper proposes an approach where a larger convolutional layer is replaced with two layers: one that captures features across the channel and another that blends them spatially. This sequence is computationally efficient and effective for spatial feature processing.

The auxiliary classifier introduced in [25], to improve convergence and combat vanishing gradients in very deep neural networks also promotes stable learning, as argued in [27]. This paper defines the auxiliary classifier as regularizer in which a batch normalization layer or dropout is utilized. The integration of traditional image recognition frameworks with residual connections has demonstrated substantial advancements, as shown in the ILSVR 2015 challenge. With Inception-V3, the implementation of residual connections within the Inception architecture was explored, resulting in moderate improvements. Subsequently, Inception-V4 presented the advantages of residual connections for faster and more reliable training. While Inception architectures offer extensive tunability, there is a tendency to be cautious in layer and filter expansion to maintain training stability. The novel Inception modules introduced are characterized by increased depth and diverse filter mapping techniques to the inputs, which generate a variety of feature maps. The innovation in the latest Inception iteration lies in its cost-efficient design, employing lightweight mappings through 1×1 convolutions after each convolutional layer. This design fuses two distinct sampling methods and applies batch normalization only on a top primary model rather than after each Residual-Inception module. This approach is considerate of GPU memory constraints and seeks to minimize information degradation [28]. The Inception V3 study [24] highlights the computational load of larger spatial filters (e.g., 5×5 or 7×7), noting that a 5×5 convolution requires approximately three times more operations than a 3×3 convolution. The pursuit of reduced computational demand without sacrificing logical capability has led to the

proposition of replacing a 5×5 kernel with two successive 3×3 kernels, thereby maintaining the network's performance while navigating the input activation grid more efficiently (refer to Figure 2.1).

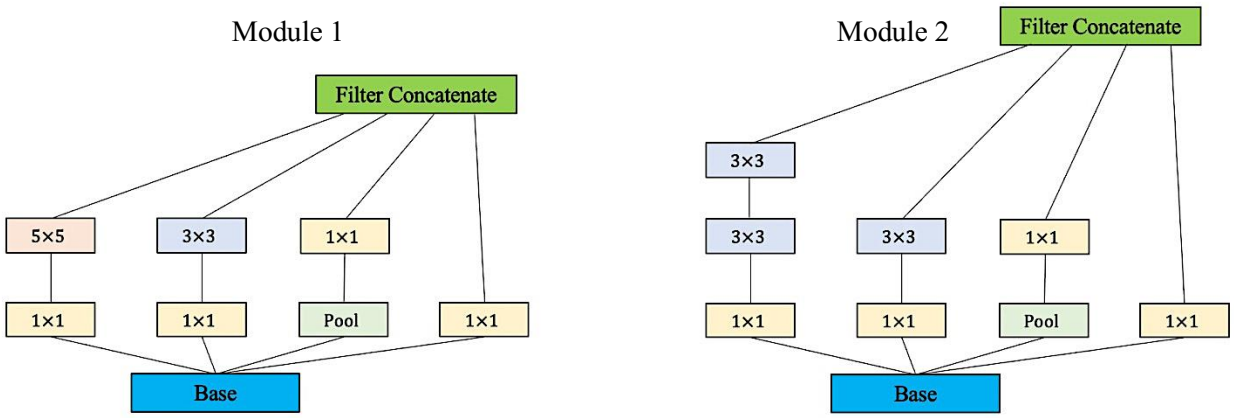


Figure 2. 1 Module 1: The original Inception module (left) [1]. Module 2: An enhanced Inception module with increased efficiency, where each 5x5 convolution has been substituted by two 3x3 convolutions.

The Inception framework, as detailed in [1], is engineered by methodically stacking a series of nine identical modules, referred to as Module 1, which are detailed in Figure 2.1. This repetitive stacking is a deliberate design choice, enhancing the network's ability to learn more complex features at various scales without a significant increase in computational cost. Each instance of Module 1 is a carefully arranged composition of convolutional layers, pooling layers, and activation functions, working together to expand the network's depth and width.

Given the network's considerable depth, ensuring effective backpropagation of gradients through all layers was challenging. A compelling observation is that shallower networks' impressive performance on this task indicates that the features generated by the network's middle layers are highly discriminative. Introducing auxiliary classifiers to these intermediate layers could promote discriminative learning in the earlier stages of the classifier, enhance the backpropagation of the gradient signal, and offer extra regularization.

The most notable example of this approach, GoogLeNet which is shown in figure 2.2. The network utilizes rectified linear activation for all convolutions, including those within the Inception modules (module 1 in Figure 2.1). The receptive field size for the network is 224×224 pixels, processing RGB color channels. Designed for computational efficiency, the network has a depth

of 22 layers, considering only those with parameters, or 27 layers when including pooling layers. The total number of distinct building blocks forming the network approximates 100. Transitioning from fully connected layers to global average pooling has enhanced the top-1 accuracy by roughly 0.6%, yet maintaining dropout proved crucial even after eliminating fully connected layers.

The challenge of effectively propagating gradients throughout the network's substantial depth was addressed. The notable performance of shallower networks suggests that features from the network's mid-layers are particularly discriminative. To promote early-stage discrimination within the classifier, enhance gradient backpropagation, and provide added regularization, they proposed adding auxiliary classifiers to these middle layers. Auxiliary classifiers are essentially smaller convolutional networks positioned upon the output of Inception modules. During training, their loss is factored into the network's total loss with a reduced weight (auxiliary classifiers' losses contribute 0.3 to the overall loss). These auxiliary networks are removed during inference.

A notable experimental approach includes setting a lower positive bound as a penalty for deviations in predicted labels or Label-Smoothing Regularization (LSR), which amplifies the Cross-Entropy (CE) loss by accounting for all potential predictions. Gradient clipping to a threshold of 2.0 and applying Batch Normalization (BN) exclusively to fully connected layers, rather than convolutional layers, have proven effective in stabilizing the training process. Model performance is assessed using a running average of parameters evaluated against the validation set.

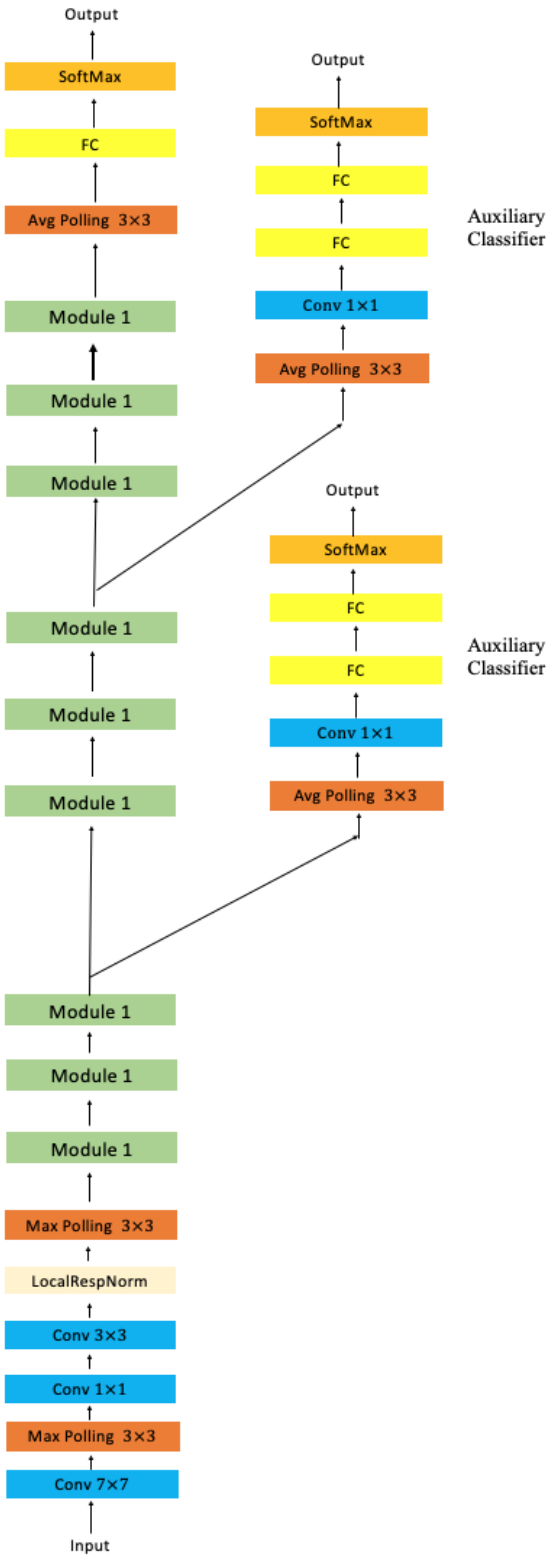


Figure 2. 2 Schematic of the GoogLeNet (Inception) Network. Layer parameters are labeled as 'layer name' (receptive field size). The RELU activation function is omitted for simplicity. 'Conv' refers to convolutional layers, and 'FC' to fully connected layers. Details of module 1 can be found in Figure 2.1

2.1.2 Enhancing Feature Propagation Using DenseNet

In Densely Connected Convolutional Networks (DenseNet) [4], recognized for their proficiency in extracting features across a range of computer vision tasks, each layer is connected to every other in a feed-forward fashion. This design means that each layer has access to the feature maps of all preceding layers. DenseNet's architecture was developed to mitigate the vanishing gradient problem, boost feature propagation, and facilitate feature reuse. A deep DenseNet is shown with three dense block structures. DenseNets, in contrast to the deep or wide structures typically seen, harness the network's potential by reusing features, leading to compact models that are straightforward to train and remarkably efficient in terms of parameters. The practice of merging feature maps from various layers diversifies the inputs for the following layers, enhancing the model's efficiency. This strategy is a key point of divergence from ResNets, which do not employ such concatenation. Even when compared with Inception networks, which similarly concatenate features from diverse layers, DenseNets stand out for their simplicity and greater efficiency. The underlying principle of DenseNet's efficiency is that each layer accesses into the preceding layers' feature maps within its block, essentially pooling the network's entire "collective knowledge". This can be compared to maintaining a global state of the network through the feature maps.

As illustrated in Figure 2.3, the architecture of DenseNet features three dense blocks, each comprising an identical number of layers. Transition layers that perform convolution and pooling separate these dense blocks. The transition layers are composed of a batch normalization layer, a 1×1 convolutional layer, and a subsequent 2×2 average pooling layer. Introducing a 1×1 convolution as a bottleneck layer before each 3×3 convolution serves to limit the number of input feature-maps, refining the process. A notable distinction of DenseNet from other architectures is its capacity to operate efficiently with narrower layers.

In Figure 2.4, the concept of direct connections from any layer to all succeeding layers is introduced. This configuration is schematically depicted in Figure 2.3, showing how DenseNet's dense connectivity pattern earns its name. In this setup H (L^{th}), the layer receives the feature maps from all preceding layers, unifying the network's knowledge at each node. This dense connectivity is central to what defines the architecture as a Dense Convolutional Network (DenseNet).

The function H(.) represents a composite function including three sequential operations: batch normalization (BN), rectified linear activation (ReLU), and a 3 × 3 convolution (Conv).

Consequently, the L^{th} layer receives the feature-maps of all preceding layers, x0, . . . , x−1, as input:

x = H ([x0, x1, . . . , x−1]). If each function H produces k feature maps, it follows that the L^{th} layer has k0 + k × (L−1) input feature-maps, where k0 is the number of channels in the input layer. Within each dense block, layers inherit concatenated outputs from all previous layers as an input tensor, while down-sampling is applied between dense blocks during training. The dense block is created by densely convolutional layers connected. This connection is made after two convolutions, which means each dense block consists of eight convolutional layers to increase feature reuse and compensate for resolution loss abilities. Each layer is connected to all earlier layers for better feature extraction.

A key distinction of DenseNet from other architectures is its adoption of narrow layers. For instance, a growth rate of 12 channels per layer allows feature maps to serve as a network's global state, accessible throughout the network. DenseNet also emphasizes model compactness, especially within transition layers. Concluding dense blocks, global average pooling is executed instead of fully connected layers before the SoftMax classifier. DenseNet's feature fusion is characterized by weight sharing within each dense block, with the subsequent blocks assigning minimal weight to outputs to address feature redundancy, as evidenced in the DenseNet-BC experiments. The classifier connected to the preceding dense block leverages weights from the entire block, focusing on the final feature maps for higher-level feature production. DenseNet accomplishes comparable performance to other networks with fewer parameters and computational requirements, similar to executing deep supervision implicitly through short connections (identity mapping), promoting feature reuse. DenseNet is scalable to hundreds of layers without encountering optimization challenges.

Dense U-Net [3], framework adds a newly designed dense block through U-Net architecture to introduce deep U-Net architecture as the first work. Each dense module comprises two convolution layers (3 × 3) and a max-pooling layer with RelU as the rectified function. For the left-hand crafting path with four steps and five dense blocks to capture semantic contextual information, four stages and five dense blocks from the expansive right-hand path, including up-sampling and two convolution layers followed by the Relu activation function to reconstruct the high-resolution images.

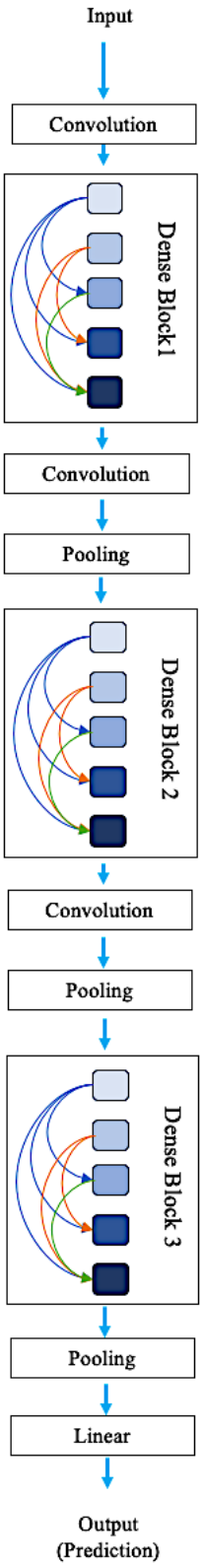


Figure 2. 3 DenseNet Structure [4] Featuring Three Dense Blocks. Transition Layers Between Blocks Adjust Feature-Map Sizes and Include Batch Normalization (BN), 1×1 Convolution (Conv), and 2×2 Average Pooling.

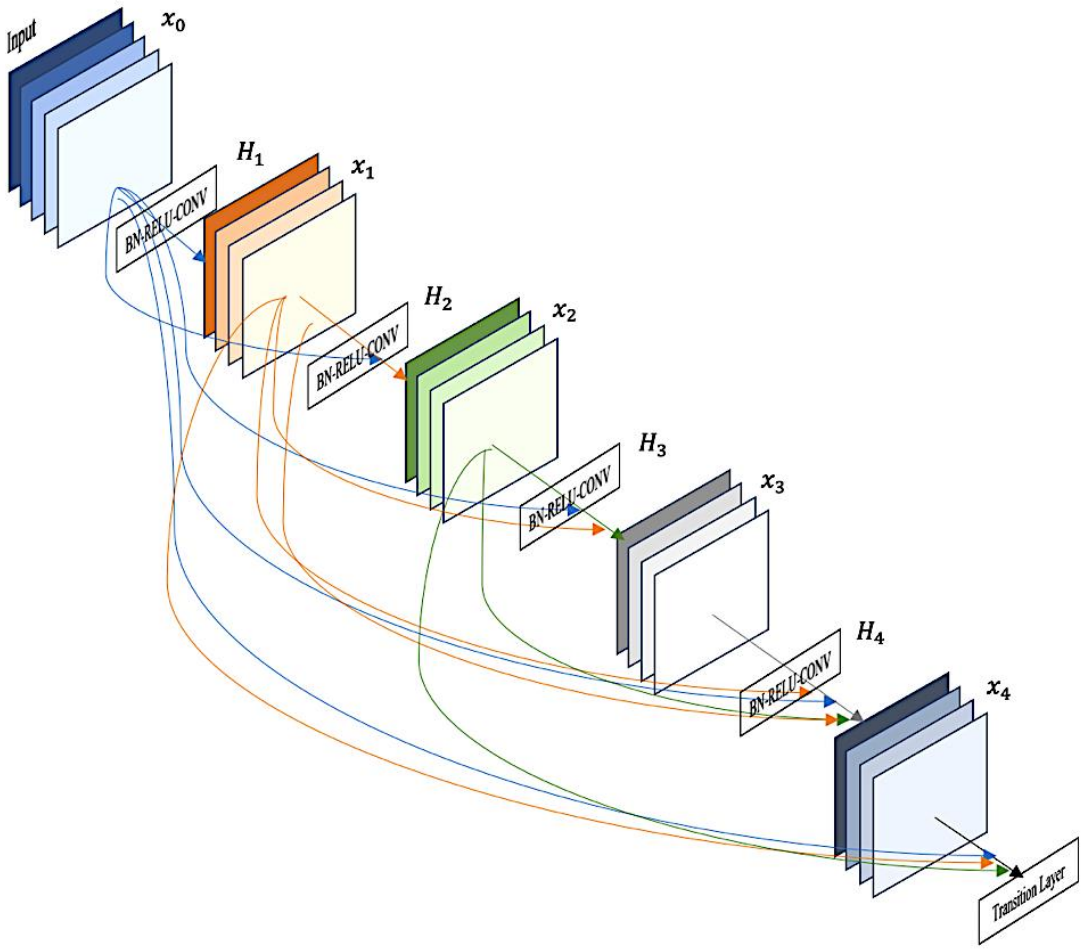


Figure 2. 4 Diagram of a Dense Block in DenseNet Architecture Showing Layer Connectivity and Growth Rate, each dense block is created by densely convolutional layers connected.

2.1.3 Advancing Deep CNNs with Residual Learning

The enhancement of image recognition architectures through increased depth has been extensively studied, particularly focusing on the role of residual modules [4], [5]. Research has also delved into the impact of shortcut connections on reducing the need for a large number of parameters [4], [29]. Highway networks, which feature shortcut connections with learnable gate functions, were introduced as an innovation in this space [30], [31]. Residual connections, which link the input directly to the output when dimensions align, were incorporated to mitigate information degradation in deep models. This adoption of residual learning at each layer serves to counteract the potential performance drop associated with increased model depth. Three variations of identity shortcut connections are explored: without residual learning, with moderate projection shortcuts, and with full projection shortcuts; the latter is omitted to alleviate computational. In Figure 2.5, they explore the implementation of deeper bottleneck architectures within ResNet-50, ResNet-101, and ResNet-152. These architectures refine the foundational building block by incorporating a stack of three layers instead of the conventional two. The structure of these three layers consists of 1×1, 3×3, and again 1×1 convolutions. The primary role of the 1×1 layers is to first reduce and subsequently increase (or restore) the dimensions. This strategic reduction and expansion enable the central 3×3 layer to operate as a bottleneck, processing data with smaller input/output dimensions. Such a bottleneck design, characterized by the sequence of three layers, is specifically customized for deeper models to manage each residual function efficiently. Crucially, this design facilitates the maintenance of residual blocks' complexity, especially through the final 1×1 convolution layer, which expands the dimensions back to their original size. Within the spectrum of ResNet architectures, two main types of shortcuts are utilized to enhance the efficiency and functionality of these networks. The first type, known as projected shortcuts, employs a 1×1 filter to increase the dimensionality where necessary. The second type, identity shortcuts, provides parameter-free connections. These connections are vital for an effective bottleneck design, ensuring that the network can learn more complex functions without a proportional increase in computational complexity. Comparatively, while all three models ResNet-50, ResNet-101, and ResNet-152 leverage this bottleneck architecture, the depth of their networks varies, with ResNet-152 being the deepest. This variation in depth allows for a sophisticate exploration of how increasing the number of layers impacts the network's ability to learn more complex patterns without significant increases in resource requirements. The adoption of bottleneck designs across

these models illustrates a strategic approach to enhancing deep learning networks' efficiency and performance, illustrating the balance between depth and computational feasibility.

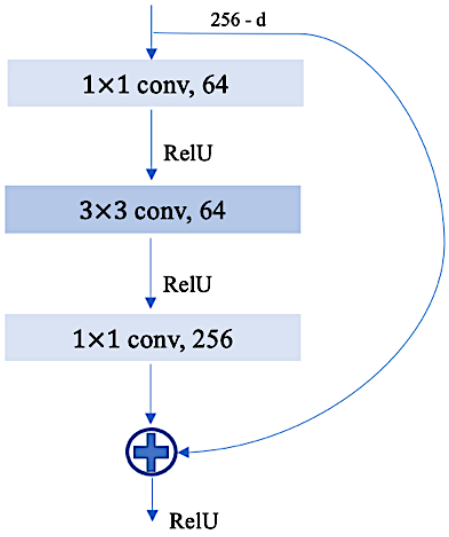


Figure 2. 5 Residual Learning: Implementation of 'Bottleneck' Layers in ResNet-50, ResNet-101, and ResNet-152 [32].

A key characteristic of the shortcut connections employed in these networks is the use of identity mapping. This is achieved by concatenating input and output of the same dimension, supplemented with zero padding where necessary, a technique that does not introduce any additional parameters [32]. Identity mappings within this context are realized through shortcut connections that employ element-wise addition, a design choice that introduces neither additional parameters nor computational complexity. A residual block is thus composed of nonlinear layers, coupled with identity mapping, where the input and output dimensions are compatible.

The comparison between two models for ImageNet, as tested by ResNet, is illustrated in Figure 2.6, focusing on the Plain Network and its residual peer. The Plain Network, shown in the middle of Figure 2.6, draws inspiration from the VGG nets, represented on the left of the same figure. This model utilizes 3×3 filters across its convolutional layers, following to two principal design guidelines: (i) maintaining a consistent number of filters for layers with identical output feature map sizes and (ii) doubling the number of filters whenever the feature map size is reduced by half, thereby keeping the time complexity per layer unchanged. Downsampling is achieved directly

through convolutional layers with a stride of 2, concluding in a global average pooling layer, followed by a 1000-way fully connected layer with softmax. Notably, the Plain Network comprises 34 weighted layers and is characterized by a simpler structure and fewer filters compared to the VGG nets. Specifically, the 34-layer baseline model requires 3.6 billion FLOPs, significantly less than the 19.6 billion FLOPs of the VGG-19, accounting for just 18% of its computational complexity. Building upon the Plain Network, the study introduces shortcut connections, as shown on the right of Figure 2.8, transforming it into a residual network. These shortcuts, which do not add any extra parameters when the input and output dimensions match, facilitate direct identity mapping. For dimensionality mismatches, two approaches are considered: (A) identity mapping with zero-padding for increased dimensions, adding no additional parameters, and (B) dimension matching through 1×1 convolutions. These shortcuts, especially effective when crossing feature maps of varying sizes, employ a stride of 2 to maintain efficiency. Despite the Plain Network's lower complexity and reduced FLOP count relative to VGG19, the introduction of parameter-free shortcut connections between pairs of 3×3 filters elevate it to the more advanced residual model. This adaptation, coupled with the systematic use of batch normalization (BN) after each convolution and before activation, without incorporating Dropout in any ResNet variant, underscores a strategic architectural enhancement. This design choice notably excludes Dropout across all depths of ResNet models (18, 34, 50, 101, 152, or 1202 layers), acknowledging that deeper networks might face slower convergence. The ResNet architecture addresses the degradation challenge, promoting higher accuracy with increased depth. Moreover, it optimizes the training process by ensuring faster convergence at the start, highlighting its superior performance and efficiency compared to its earlier versions.

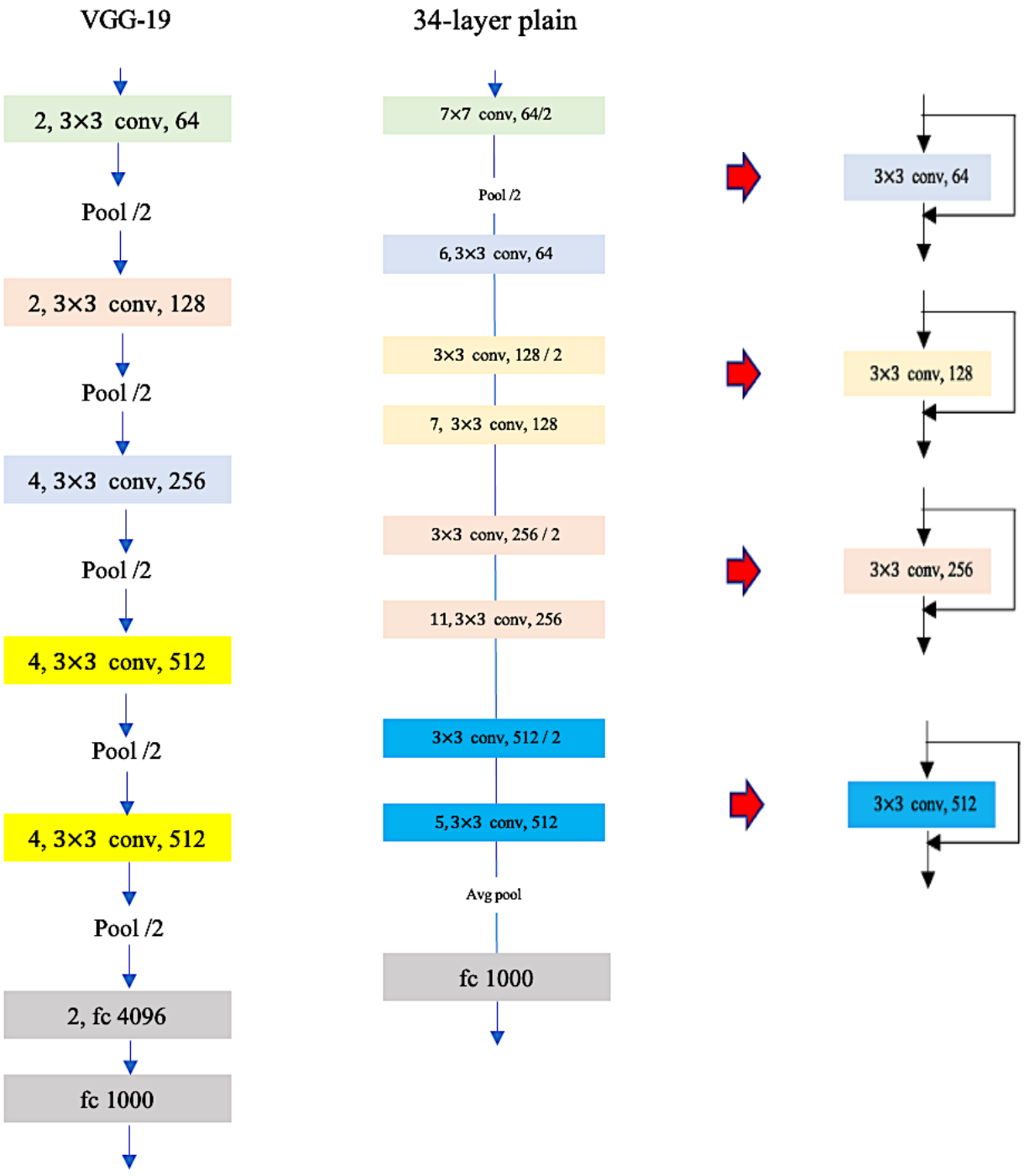


Figure 2. 6 Comparison of Network Architectures [24]: Left VGG-19 Model as a Design Reference , Middle : Plain Network with 34 Layers Without Shortcut Connections, right: Residual Network with 34 Layers Featuring Shortcut Connections.

2.1.4 Changing Deep CNN Architectures with U-Net

The U-Net architecture [3], introduced in 2015, has gained prominence for its efficacy in image segmentation tasks, particularly within the medical imaging domain. Its usage continues to grow due to the flexibility and success of various U-Net adaptations. The fundamental architecture of U-Net is divided into two principal pathways:

The first pathway is known as the contraction or analysis path, operating similar to a conventional convolutional network to encode and compress information, facilitating classification. The second pathway, the expansion or synthesis path, is comprised of transposed convolution layers that progressively reconstruct the resolution of the output. This architectural innovation enhances the precise localization required for detailed segmentation tasks, with the U-shaped design proving instrumental in context-based learning for image segmentation.

The architecture of the network, depicted in Figure 2.7, is composed of two main segments: a contracting path to the left and an expansive path to the right. The contracting path mirrors a conventional convolutional network design, involving a series of two 3x3 convolutions without padding, succeeded by a ReLU activation and a 2x2 max pooling with a stride of 2 for downsampling. With each downsampling, the feature channels are doubled, enhancing the network's ability to capture complex features. Conversely, the expansive path begins with upsampling the feature map, followed by a 2x2 “Transposed Convolution” that decreases the feature channels by half. This is immediately followed by merging it with a matched, size-adjusted feature map from the contracting path, and then applying two additional 3x3 convolutions, each succeeded by a ReLU. This merging, known as "Skip Connection," allows the network to recover spatial information lost during downsampling. The need for cropping arises to compensate for the spatial loss at each convolutional step. The final layer uses a 1x1 convolution to translate the 64-component feature vector into the required number of output classes. Altogether, the network comprises 23 convolutional layers. The first segment of the U-Net behaves similarly to the VGG network, where the image's spatial resolution is reduced while the channel depth increases. In the expansive path, this process is reversed: spatial resolution is amplified as the number of channels diminishes. The expansive path also maintains a high number of feature channels, which is crucial for conveying context information up to the higher resolution layers. Consequently, the expansive path roughly mirrors the contracting path, completing the U-shaped structure of the architecture.

This symmetry ensures that the network efficiently produces high-resolution outputs, making it adept at tasks requiring detailed localization and segmentation.

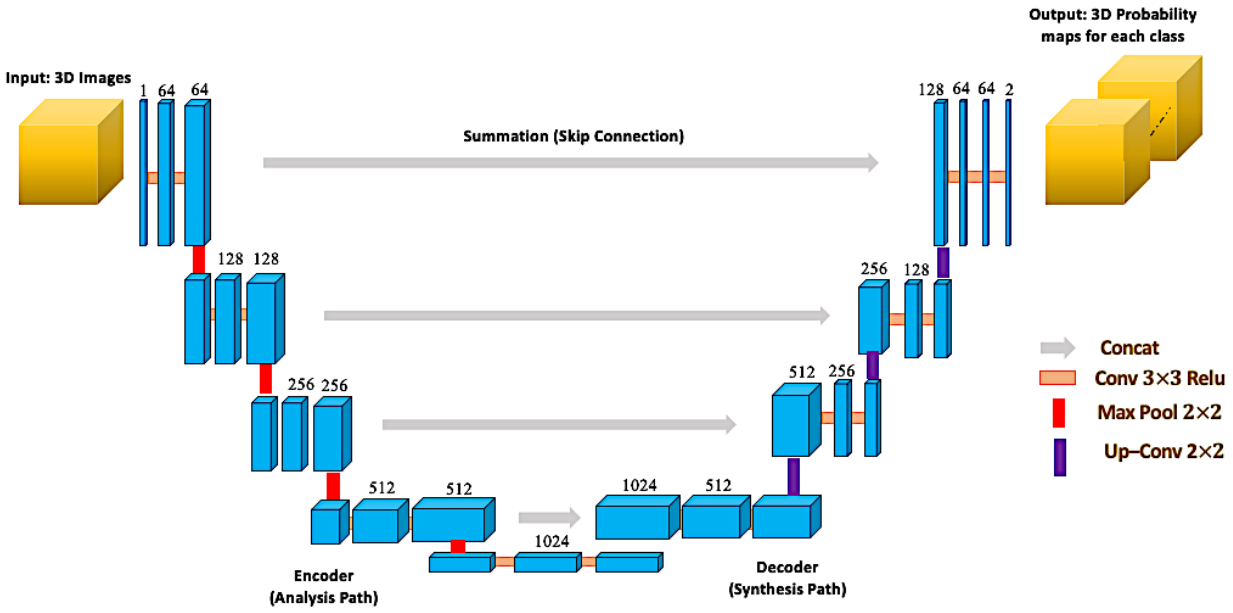


Figure 2. 7 U-net [6] architecture with 32x32 pixels at the lowest resolution. Each brown box represents a multi-channel feature map, with the number of channels indicated above the box.

The Standard U-Net [6] architecture is notably efficient when trained on a limited number of samples. It was originally enhanced by incorporating data augmentation and a loss penalty function to develop its performance. The architecture is distinguished by its dual-path design: an encoding path that captures context and a symmetric decoding path that facilitates precise localization of relevant information. To achieve high-resolution localized outputs, the results from the encoding path layers are merged with the up-sampling segments of the decoding path. This integration strengthens a more accurate learning of segmentation information from the initial convolutional layers.

Attention U-Net [33], extension develops of vanilla U-Net with an attention module to surpass irrelevant features and automatically learn to highlight salient regions with minimal extra computation while increasing prediction accuracy. The attention block on top of U-Net is a self-attention gating module utilized in CNN to specify more attention to local regions. U-Net has two main advantages: the efficient use of GPU memory and extracting features from scaled images

provide good performance in image segmentation. The purpose of solving difficulty in the prediction of small objects with high variation for results, attention architectures consist of the addition of scaled input and original.

The Inception U-Net architecture [34], merges the principles of Google's Inception [25], with the foundational U-Net [6], substituting each convolutional layer with an Inception module. Additionally, a hybrid pooling layer namely Hartley spectral pooling which is adept at preserving the spectral structure of features for enhanced discriminability is implemented to retain more spatial information [35]. This model amplifies depth through the U-Net structure and widens via the Inception module. The architecture is outlined by a contraction path and a subsequent expansion path, which is carefully optimized with a weighted objective function customized for segmentation loss. This function is a combination of three distinct loss functions: binary cross-entropy (BCE), Dice coefficient loss (DCL), and Intersection over Union (IoU), each contributing to the model's learning efficacy.

The Deep Residual U-Net [36], architecture employs residual skip connections to streamline the training of deep networks. These connections enable efficient information flow, allowing for the training of a network with fewer parameters while avoiding the issue of degradation. They simplify the complex process of training image segmentation models that capture fine-grained details and retain low-level information. Researchers further enhance the model's performance by employing fine-tuning through transfer learning and extensive data augmentation to offset any potential information loss. The network's architecture is sectioned into three key areas: the encoder, which compresses the input data into a representative form; the decoder, which reconstructs the image representation for semantic segmentation; and an intermediate bridge connecting the two. Each module within the network maintains identity mapping, ensuring a direct connection between input and output.

U-Net++ [37], is structured with an encoder and decoder, similar to its earlier version. What sets it apart is the reconfiguration of skip connections, which, unlike the direct connections in the standard U-Net, are densely interconnected. These enhanced skip pathways facilitate gradient flow, making the network more receptive to training. Each paired encoder-decoder block within U-Net++ incorporates a trio of convolutional layers. The design intent is to streamline optimization by ensuring that the encoder and decoder receive feature maps of like semantic quality. U-Net++ employs deep supervision, allowing for precise and speed up model training.

2.1.5 Optimizing Deep CNN Architectures Using VGG

Very Deep Convolutional Networks for Large-Scale Image Recognition network, codenamed VGG [2], explained the significance of depth in visual representation frameworks, showcasing models with 16 and 19 layers. In the 2014 ImageNet challenge, VGG achieved top ranks, underscoring the vital role of increased depth in enhancing accuracy. The network uses a consistent 3×3 kernel size for filters, demonstrating robust generalization across various datasets for localization and classification tasks. A key aspect of the ConvNet architecture that VGG highlights is the improvement in performance through a progressive increase in depth, adding more convolutional and MaxPooling layers. During experimentation, image inputs of size 224×224 were preprocessed with VGG normalization before passing through convolutional layers with 3×3 kernel filters and minimal receptive fields, optimizing for precise information capture from different image regions. Some variants of the model employ 1×1 convolutional layers as linear transformers of input channels, paired with non-linear ReLU activation to amplify the decision function's non-linearity without altering the conv layers' receptive fields. Padding with a stride of 1 is implemented to maintain spatial resolution post-convolution. The number of channels in the convolutional layers starts at 64 and doubles incrementally, resulting in a width of 512, followed by MaxPooling. The architecture concludes with three Fully Connected (FC) layers, choosing for narrower widths compared to shallower models, as noted by Sermanent et al., 2014 [38].

The VGG ConvNets configurations are outlined in Table 2.1, with a straightforward preprocessing step: every pixel has the mean RGB value, derived from the training set, subtracted from it. The images experience a series of convolutional (conv) layers, employing 3 × 3 filters. In certain configurations, 1 × 1 convolution filters are also applied, effectively performing a linear transformation of the input channels, followed by a non-linear activation function. The stride of the convolutions is consistently set at 1 pixel, with spatial padding maintained to ensure the o riginal spatial resolution remains preserved after the convolutions this requires a padding of 1 pixel for the 3 × 3 conv layers. Spatial pooling is executed by five max-pooling layers distributed among the conv layers, although not every conv layer is immediately followed by max-pooling. These max-pooling layers operate over a 2 × 2 pixel window with a stride of 2, serving to reduce spatial dimensions and allowing for the assumption of stronger feature representations at reduced computational costs. The architecture's depth varies, leading to a series of conv layers succeeded by three fully connected (FC) layers. The first two FC layers each have 4096 channels, while the

third FC layer outputs a number of channels corresponding to the number of classes. A softmax layer concludes the sequence. The design of the FC layers remains uniform across all models. All hidden layers utilize the ReLU activation function to introduce non-linearity, enhancing the network's ability to learn complex patterns. It is noteworthy that Local Response Normalisation (LRN) is not employed in these networks with the exception of one as it does not boost performance on the ILSVRC dataset. Additionally, LRN tends to increase both memory demands and computational time. The tailored design of VGG ConvNets, with its focused convolutional strategy and absence of LRN, showcases an efficient structure fine-tuned for robust image classification performance without unnecessary computational overhead.

Table 2. 1 Variations of VGG ConvNet architectures are presented column-wise. Configuration depth progresses from left (A) to right (E), with additional layers highlighted in bold. Convolutional layer specifications are described by 'conv-receptive field size-number of channels'. The ReLU activation function is omitted for simplicity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight Layers	11 weight Layers	13 weight Layers	16 weight Layers	16 weight Layers	19 weight Layers
Input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128	conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

2.2 Enhancing Classification with Modified Cross-Entropy Loss Functions

2.2.1 Cross-Entropy Loss

2.2.1.1 Highlighting Cross-Entropy in Neural Network Training

The loss function serves as the guiding force in the network training process, optimizing weight adjustments during backpropagation by measuring the gap between predictions and actual targets. This metric guides the iterative refinement of the network's weights, driving the loss towards a minimum for improved model precision. Traditionally, sigmoid activations paired with sum squared error were standard. However, current practices prefer SoftMax activation for the output layer coupled with cross-entropy loss for classification tasks. As depicted in Figure 2.8, cross-entropy (shown by the red curve) offers a superior loss trajectory over the traditional approach (indicated by the blue line), especially apparent during the initial training phase when predictions are most inaccurate. This improvement not only speeds up early learning phases but also accelerates convergence, enhancing efficiency of the optimization process within backpropagation and cutting down on the time required for gradient calculations. This passage focuses on the evolution from traditional sigmoid and sum squared error to modern SoftMax and cross-entropy methods, highlighting the benefits in terms of learning efficiency and speed of convergence.

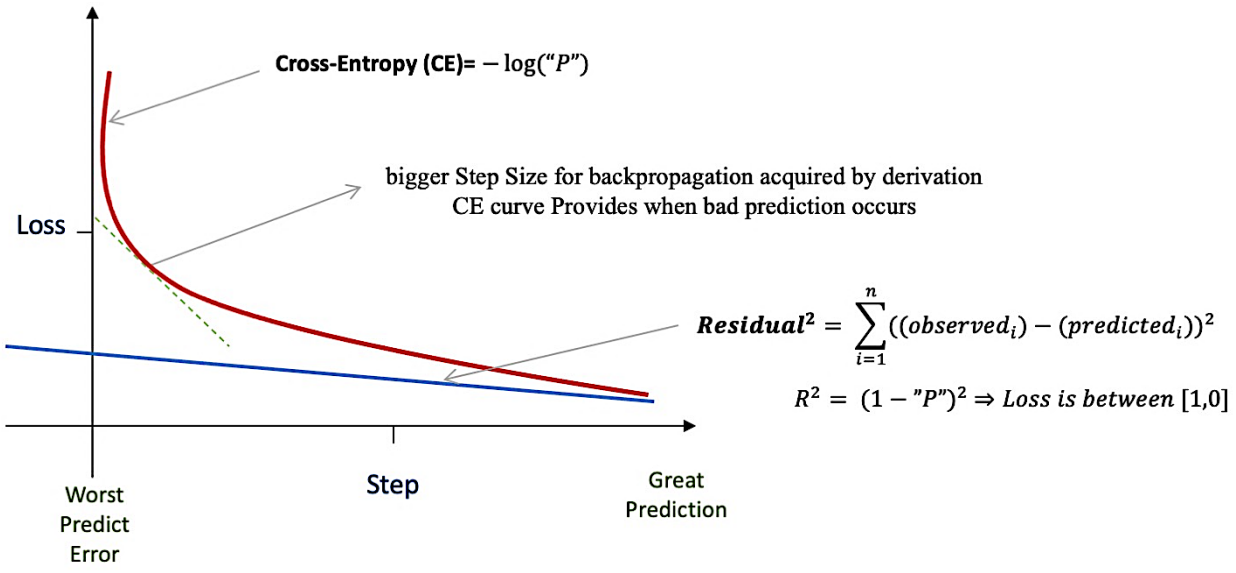


Figure 2. 8 displays the loss curves for cross entropy (red) and mean square error (blue) during model training [2].

If we see the neural network as a function $f(x;\theta)$, the outputs don't describe y distribution directly. This act is done by w (parameters). So, we can represent the loss function as $-\log P(\,y\,;w(x)\,)$, and logarithm in loss is according to maximum likelihood suggestion. This loss function provides an incrementally condition for optimization procedure to learn variance [7]. Total prediction error of a model is written by:

2.1

$$\mathrm{E}_{\mathrm{x,y,D}}\,[\,(\mathrm{h}_{\mathrm{D}}(\mathrm{x})-\mathrm{y})^2]=\;\mathrm{E}_{\mathrm{x,D}}\left[\mathrm{h}_{\mathrm{D}}(\mathrm{x})-\bar{\mathrm{h}}\,(\mathrm{x}))^2\right]+\;\mathrm{E}_{\mathrm{x}}\left[\left(\bar{\mathrm{h}}(\mathrm{x})-\;\bar{\mathrm{y}}(\mathrm{x})\right)^2\right]+\;\mathrm{E}_{\mathrm{x,y}}\left[(\bar{\mathrm{y}}(\mathrm{x})-\mathrm{y})^2\right]$$

Given a dataset $D=\{\,(x_1,y_1),\ldots\,,(x_n,y_n)\}$, and (x,y) pairs are data points for regression. Here, x is the input, y is the true label, $\hat{y}$ is the predicted label, h_D is the model hypothesis, and $\hat{h}$ is a weighted average of functions. $E_{x,y,D}$ represents the expected error , $E_{x,D}$ is variance, E_x is bias, and $E_{x,y}$ is noise. Understanding prediction errors, specifically bias and variance, is crucial for evaluating a model's performance. This balance affects model accuracy and helps prevent overfitting and underfitting.

- Bias: The difference between the model's average predictions and actual values. High-bias models ignore training data, leading to significant errors on both training and test datasets.
- Variance: The inconsistency of model predictions for a specific data point, showing data spread. High-variance models closely follow training data but fail to generalize, performing well on training sets but poorly on test sets.
- Irreducible Error: The noise inherent in data that cannot be eliminated, regardless of model complexity.

Models with few parameters might exhibit high bias and low variance, suggesting a simplicity that fails to capture complex patterns. Conversely, models with numerous parameters may show low bias but high variance, fitting the training data too closely at the expense of generalizability. The key lies in finding a balance between bias and variance to minimize overall error. This balance, depicted in Figure 2.9, avoids the extremes of overfitting and underfitting. Reaching to the sweet spot (shown in Figure 2.9) ensures that the model is neither too simple nor overly complex, optimizing for both accuracy and generalizability.

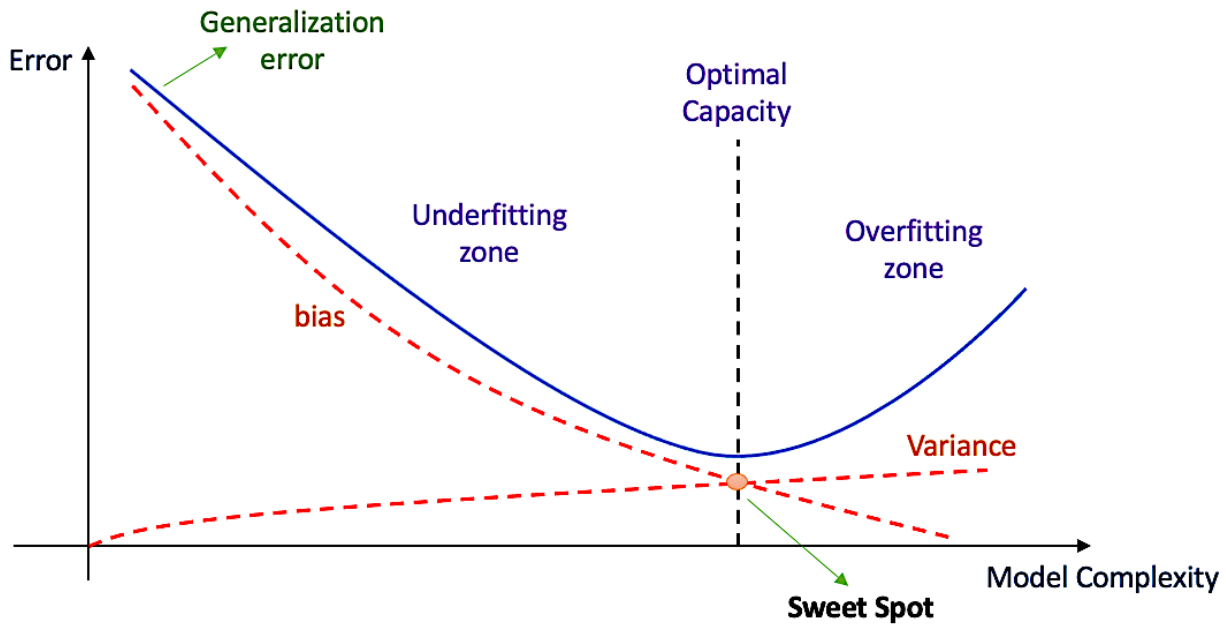


Figure 2. 9 Bias-Variance Trade-off in Machine Learning. This figure depicts the balancing act between bias and variance [2].

Deep neural networks have achieved significant success in classification tasks, particularly when dealing with limited annotated training data. In response to the challenge of noisy data, researchers have proposed noise-robust loss functions. These functions aim to adapt and modify traditional loss functions for better performance under noisy conditions. Examples include the ramp loss [39] and unhinge loss [40], which incorporate L2 regularization to derive a convex loss function, enhancing robustness against symmetric label noise (SLN) [41]. Recent efforts have focused on further improving noise-robust loss functions, with particular emphasis on Mean Square Error (MSE) and Cross Entropy (CE), aiming to refine their effectiveness in noisy environments.

The paper titled 'The generalized cross-entropy loss for training deep neural networks with noisy labels' [2] introduces a generalized categorical cross-entropy (CCE) loss. This approach outperforms the traditional Mean Square Error (MSE) as a baseline noisy-robust cost-function, particularly in handling noisy labels and complex datasets. CCE is effective for dealing with both closed-set and open-set noisy data samples. 'Open set' refers to scenarios where incorrect labels do not match any ground truth classes in the dataset. Conversely, 'closed set' describes situations where both incorrect and correct labels are from the known classes present in the training data. Compared to MSE, cross-entropy loss, which is based on logarithmic error calculation, offers several advantages. As a variation of Kullback-Leibler divergence, cross-entropy loss focuses on

minimizing the discrepancy between the probability estimation (p) and the ground truth (q), along with an additive constant. This method proves more efficient in refining model accuracy under conditions of label noise and dataset complexity.

$$\text{Cross Entropy} = -< \log(\mathrm{P_i}), \mathrm{L_i} > \quad 2.2$$

The inner product is denoted by <>, where P_i represents the prediction for the i_{th} sample, and L_i is the one-hot encoded reference label. The goal of cross-entropy is to measure the discrepancy between the predicted probability distribution of outcomes and the actual distribution for a specific class of the model, known as the loss function. As predictions increasingly deviate from the true values, the loss value in cross-entropy escalates exponentially. Essentially, cross-entropy quantifies this deviation as the negative log-likelihood, summing up the correct log probabilities. This process is particularly relevant for comparing two sub-gaussian probabilities, for which the Bernoulli distribution is commonly applied [42].

Traditionally, classification tasks endeavor to find the most accurate weights for the model by employing backpropagation to adjust for errors in misclassified patterns. This minimization of the error function aims to improve the model's predictions but can lead to overfitting if weights become too large, or 'saturated.' Hence, the significance of weight magnitude in achieving model generalization is underscored, highlighting its importance over the basic count of hidden units in preventing overfitting [10].

2.2.1.2 Improving Cross-Entropy Performance Using Regularization Techniques

Conventional cross-entropy struggles to learn complex classes and tends to overfit simpler ones [2]. Introducing additional terms into cross-entropy can enhance learning for challenging classes and add noise tolerance to reduce overfitting risks. Regularization techniques, such as weight shrinkage, help prevent overfitting by controlling prediction density. If a model's weights are too variable, it may overfit by closely mirroring the training set's output distribution. Cross-entropy regularization provides various methods to modify learning approaches, preventing the loss function from accruing extreme negative rewards [7]. In this sense, learning is the design of a function, not only selecting parameters.

Neural units in a saturated state generate minimal gradients. As a result, using mean absolute error (MAE) or mean squared error (MSE) can lead to unsatisfactory outcomes. In contrast, the cross-entropy (CE) loss function exhibits an advantage over MAE and MSE. This benefit persists even when it's unnecessary to predict an entire distribution over the target variable y for each feature set x [7]. The CE cost function focuses solely on the probability assigned to the correct class, which often results in more efficient training, especially for classification problems. The gradient calculation for the Cross-Entropy (CE) loss is defined as follows:

$$\sum_{i=1}^{n} \frac{\partial L\,(f(x_i;\theta), y_i)}{\partial \theta} = \left\{ \sum_{i=1}^{n} -\frac{1}{fy_i(x_i:\theta)} \nabla_\theta\, f_{y_i\,(x_i:\theta)} \right. \tag{2.3}$$

Here, θ represents the weights associated with the input samples x, n denotes the number of neurons in the layer, ∂ stands for the derivative of the loss with respect to the weights, and $fy_i(x_i:\theta)$ symbolizes the predicted probability of the correct class for the input x. In the context of CE with a SoftMax output layer, aligned or matching samples receive a higher emphasis through an implicit weighting in the gradient update, represented by either a larger value of $fy_i(x_i:\theta)$ or a smaller reciprocal $1/f_{y_i}(x_i;\theta)$. This method implicitly adjusts the learning process to pay more attention to samples that are already well-predicted, smoothing the path towards convergence.

Mean Square Error (MSE) treats each training sample equally, leading to a model that is robust against outliers and noise. However, this equality means that MSE may require longer training times, as it doesn't inherently prioritize challenging samples. This can make the learning process slower because there's no built-in mechanism to focus on those samples that would benefit most from correction. On the other hand, Cross-Entropy (CE) naturally puts more focus on the difficult samples during training, which can be advantageous when working with clean, well-labeled data. This focus can lead to faster learning on those samples that are furthest from their target predictions. Yet, this same characteristic might result in overfitting when the data contains noisy labels, as the model might learn the noise instead of the underlying pattern. The issue with using CE becomes apparent at the start of the training phase. Initially, the SoftMax output leads to many samples receiving very small gradient updates. If these samples are disregarded because their contribution to the update is minimal, the effective number of samples that the model learns from

drops significantly. This can greatly affect the model's ability to learn a broad range of patterns, particularly those from less frequent or more complex samples.

To avoid this issue, the current formula introduced:

$$arg_\theta = \sum_{i=1}^{n} L_{trunc}\ (f(x_i;\theta), y_i) = arg_\theta min \sum_{i=1}^{n} v_i L_q(f(x_i{:}\theta), y_i) + (1 - v_i)L_q(k) \quad 2.4$$

Where $L_q(k)$ is threshold (penalty as a regularization term), $v_i = 0$, if $f_{y_i}(x_i) \leq k$ (the SoftMax output probability is under the threshold). Optimizing the above loss is the same as optimizing this composition:

$$\begin{aligned} arg_\theta min &= \sum_{i=1}^{n} V_i L_q\ (f(x_i;\theta), y_i) - v_i L_q(k) \\ &= arg_{\theta, w[o,1]^n}\ min \sum_{i=1}^{n} w_i L_q(f(x_i{:}\theta), y_i) - L_q(k) \sum_{i=1}^{n} w_i \end{aligned} \quad 2.5$$

The process of updating the weight vector w can be similar to the concept of pruning. In this context, pruning occurs at each step of the iterative training process. This is accomplished by evaluating the function $f(x_i; \theta^{(t)})$, which represents the model's prediction for the input x_i at the t^{th} iteration with the current parameters θ. During an iteration, only those samples for which the loss value falls below a predetermined threshold are maintained for the weight update [2]. This selective process effectively focuses the model's learning efforts on the most informative samples those that are not yet well-predicted while discarding the rest. Consequently, this targeted update can lead to more efficient training, as the model is not overburdened with adjustments due to samples that are already in close agreement with the model's prediction. This method can be particularly advantageous when dealing with large datasets, as it helps in speeding up the training by reducing computational overhead.

2.2.1.3 Enhancing Cross-Entropy Performance for Robust Learning in Imbalanced Datasets

A notable regularization adaptation in CE is the Real-World Weighted Cross-Entropy (RWWCE) [43], which allocates distinct weights for false positives (incorrect predictions) and false negatives (omitted correct labels), enhancing the learning process.

$$L = -\frac{1}{M}\sum_{m=1}^{M}[\, W_{f_n} \times y \log\big(h_\theta(x)\big) + W_{f_p} \times (1-y) \times \log(1-h_\theta(x_m))] \qquad 2.6$$

where M represents the number of training samples, W_{f_n} is the weight assigned to the cost of false negatives, and W_{f_p} is the weight assigned to the cost of false positives. The variable y denotes the target label, x the input data for training, and h_θ the hypothesis function parameterized by the neural network weights. This model demonstrates robust performance on imbalanced datasets and, importantly, the experiments indicate that this version could result in lower real-world costs. The weights for this binary loss model are established through a specified matrix. Adjustments to cross-entropy help tackle optimization obstacles. The landmark study titled "Modified Cross-Entropy Method for Classification of Events in NILM Systems" [44] adds a new modification by setting a cut-off point and penalizing log errors that cross this limit. This adjusted version of cross-entropy loss deals well with samples that carry the same weight, which is especially useful for classes in datasets where some types of data are rare. It also updates how samples are weighed, giving more importance to reliable, noise-free data. This is especially vital in tasks like recognizing facial expressions, where distinguishing between true and mistaken labels is crucial. By introducing these penalties, the method shifts focus, giving an edge to accurate, clean samples, which can lead to better, more reliable recognition performance.

In instances where the disparity between the predicted probability and the target value is less than a predefined threshold, Cross-Entropy (CE) assigns greater weight compared to mean absolute error (MAE), enhancing the model's ability to learn from hard samples. CE's weighting scheme prioritizes clean data, yet it may lead to overfitting in deep convolutional neural networks (DCNNs). On the contrary, MAE assigns lesser weight to noisy samples, offering robustness against noisy labels, but its weighting approach diminishes learning efficiency, making it less effective for DCNNs in learning label distributions. To address these limitations, an improved cross-entropy that incorporates the noise robustness of MAE, along with the weighted scheme of

CE, is proposed in [45], [46]. This enhanced cross-entropy formulation motivates the model to exhibit consistent behavior during both training and testing phases. Specifically, it reduces bias, thereby making the model less prone to input variations, by employing a schedule sampling weight for the hyperparameter within the cross-entropy formula. As training progresses, this parameter increases in value, highlighting the importance of negative predictions associated with hard samples, as empirically validated in [47]. DCNNs naturally tend to initially learn from easy samples, gradually progressing to hard samples. The modified cross-entropy [134] with a weight scheduling parameter is outlined as:

$$L = -[(1-\alpha_i).\sum_{t=1}^{n}\log P_\theta(y_t|y<t,x) + \alpha_i .\sum_{t=1}^{n}\log P_\theta(\hat{y}_t|y<t,x)] \quad 2.7$$

In this work the standard cross-entropy (CE) loss is used across training methods, functions as the core of the learning process, guiding the model when predictions $\hat{y}_t$ match the actual targets y_t. However, the formula introduces a model-dependent version of CE for additional flexibility. Specifically, when the predicted class $\hat{y}_t$ aligns with the true class y_t, the mixed CE simplifies to the standard CE. Conversely, when $\hat{y}_t$ does not equal y_t, the mixed CE adjusts to accommodate the discrepancies. This dual behavior allows mixed CE to act adaptively during scheduled sampling, reinforcing the model's predictions when accurate and seeking to correct them when they diverge. This approach seeks to find a balance between strengthening correct predictions and addressing and correcting errors, tailoring the learning process to the specific needs and dynamics of the training data. Where hyperparameter α is: $\alpha_i = m.\frac{i}{total\_iter} \implies 1 \le i \le total_{iter}$, $m = 0.5$.

In a comprehensive framework that contains both active and passive components, the scaled version of the loss function is expressed as:

$$\mathrm{L_{overall}} = \alpha.\mathrm{L_{Active}} + \beta.\mathrm{L_{passive}} \quad 2.8$$

Here, α and β serve as balancing parameters. This dual-component term is notably characterized by its robustness [49].

A dynamic weighting scheme for the loss function, functioning as a self-adjusting weight algorithm, enables the model to refine parameters when encountering minority class samples that excessively affect classification. Misclassification error often arises from uniform weight distribution across class samples. Therefore, during training, hard samples with incorrectly high-confidence predictions are assigned increased weight. This contrasts with a static weighting approach [50], which, while capable of balancing minority and majority classes, does not differentiate between hard and easy samples as effectively as focal loss [13] exploited. The class weight parameters are dynamically learned and self-adjust based on predicted probabilities [51]. Dynamically Weighted Balanced (DWB) Loss defined by the following formula:

$$L_{DWB} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} w_j^{(1-P_{ij})} y_{ij} \log(P_{ij}) - P_{ij}(1-P_{ij}) \quad 2.9$$

where w_j is the class weight of class j, y_{ij} is the j th element of one-hot encoded label of instance x_i and P_{ij} is the predicted probability of the class j of instance x_i. Here, W represents a hyperparameter that can be optimized through cross-validation or can be set by determining an appropriate log ratio. This log ratio is designed to smooth the weights, inversely proportional to class frequency and is defined as follows:

$$w_j = \log\left(\frac{\max(n_j \mid j \in c)}{n_j}\right) + 1 \quad 2.10$$

Within the framework, which is describing, the penalty for incorrectly classifying a sample into class j and class frequency n_j is amplified by a factor corresponding to that class's weight w_j. This means that if a class j has an assigned cost weight of w_j, any misclassification of that class is penalized w_j times more heavily compared to misclassifications of the majority class, which carries a weight of 1. This weighted penalty system prioritizes the accurate classification of minority classes, which is often crucial in imbalanced datasets where certain classes are underrepresented. By assigning greater penalties to these classes, the model is motivated to learn features specific to them more completely, thereby aiming to reduce bias towards the majority class. Selecting adaptable weights within the cross-entropy (CE) loss significantly influences the performance of deep neural networks (DNNs), conferring robustness to noisy labels and

accelerating the learning process. The enhanced categorical cross-entropy [52] dynamically adjusts weights by employing an exponential term within the weighting scheme, tailored to the predicted probability distribution.

The learning progress differs significantly when dealing with clean data as opposed to noisy labels, effectively categorizing the training data into easy and complex (hard) samples. Classes filled with simple samples might fit the training data too well, leading to overfitting. Conversely, classes with complex samples often don't fit enough, raising the risk of underfitting. To address this, techniques that improve robustness are crucial. An example is the technique introduced in "Learning from Noisy Labeled Data Using Symmetric Cross-Entropy Loss for Image Classification" [53], which puts forward the Symmetric Cross-Entropy (SCE) method. This alternative to the traditional loss function uses a ratio of samples adjusted based on the level of noise in the data, a step that's particularly important early in training. This adjustment helps the model focus on learning from the simple samples at the beginning, thereby balancing the learning process across different types of data. The sample ratio is defined as follows:

$$\lambda(e) = 1 - \min\left\{\frac{correct\ epoch}{max\ epoch}\ \tau, \tau\right\} \text{ where } \tau \text{ is noise ratio.} \quad 2.11$$

2.2.2 SoftMax Function

2.2.2.1 Understanding SoftMax and its Advantages in Classification

The standard cross-entropy is represented by the following formula:

$$L = \sum q(k|x)\ \log P(k|x) \quad 2.12$$

Where $P(k|x) \approx \frac{e^{zk}}{\sum_{j=1}^{k} e^{zj}}$ is a SoftMax as an activation function for the last layer of neural networks in classification domain. In SoftMax, lower logits are higher entropy, so the prediction has more uncertainty. The efficiency of loss functions is traditionally blocked by their limited ability to promote discriminability during feature generation. A classifier is required to produce a conditional probability distribution, where the goal is to align the argmax of the predicted

probability vector closely with the correct class label. This is the essential function of a SoftMax classifier. However, enhancements to the SoftMax function that require a model to yield more distinct features have led to improved loss function performance. This subject has collected considerable interest in research circles over recent years.

SoftMax function is applicable at any moment, there is a need of a probability distribution over number of distinct categories. It's an extension of the sigmoid function, scaling up from binary to multi-class classification. Before normalization, the log probabilities from the linear layer's output ($z = W^T \times h + b$) are well-suited for gradient optimization. This scaling translates into a Bernoulli distribution, modulated by the parameters of the SoftMax function, which extends to the multinomial distribution, also known as the categorical distribution. The logarithm in the log-likelihood shields against negative exponentials that may hinder gradient updates, a phenomenon known as vanishing gradients. The definition of SoftMax within the log-likelihood leverages the natural property of logarithms to offset the exponential:

$$\text{Log SoftMax}(z)_i = z_i - \log \sum_j \exp(z_j) \tag{2.13}$$

Here, the input term always directly influences the loss function, avoiding saturation during differentiation. By maximizing the log-likelihood, the first term is raised while minimizing the second, giving us insight into the negative log-likelihood's heavy penalties on the most active incorrect predictions. When a function has a correct answer, this term contributes minimally to the overall training loss and is dominated by the costs of incorrectly classified samples. This model variant remains numerically stable, shrinking numerical errors even when z oscillations to extremely high or low values. SoftMax tends towards 1 with large gaps between the highest and other values or zeroes out when the highest isn't the absolute maximum, indicating overconfidence in an incorrect prediction. Here, the loss function must adjust, appropriately penalizing errors to temper the overall impact on the loss [7].

It's important to understand how SoftMax output, when compared to a one-hot vector, can be made more accurate through regularization techniques. Regularization, particularly using entropy, introduces new information for the model to learn, thereby enabling the identification of more distinctive feature vectors. This process is critical as it allows the model to enhance its ability to

differentiate between classes effectively. The integration of cross-entropy loss with the SoftMax classifier, commonly referred to as SoftMax-Loss [54], is a crucial step. This combination leverages the strengths of both components: while SoftMax transforms the model's outputs into probabilities, cross-entropy loss measures the difference between the predicted probabilities and the actual distribution. Together, they work collaboratively to refine the model's predictions, making the SoftMax-Loss function a cornerstone of effective model training [8]. (see figure 2.10)

$$\text{Softmax Loss (Multi Class Logistic Loss)} = \text{Softmax Activation} + \text{Cross Entropy}$$

Figure 2. 10 Optimizing a model via the cost function for classification tasks [8].

2.2.2.2 Improving Classification with Margin-Based SoftMax Variants

In classification domain, the creation of distinct decision boundaries between classes is essential. Difficulty often occurs when predicted outputs float near these boundaries. To mitigate this, researchers have developed multiple variations of the traditional SoftMax function. These novel adaptations propose unique takes on the concept of margin, which serves to widen the gap between classes, thereby enhancing class separability. Moreover, they aim to tighten the cluster of data points within each class, strengthening the coherence of samples sharing the same label. This dual strategy of expanding inter-class margins while compacting intra-class distances greatly contributes to a model's discriminative power.

Literatures on margin loss frequently explores these SoftMax variants. The common thread among these enhancements is the incorporation of a margin that is directly embedded into the SoftMax function. This integration fundamentally modifies the decision landscape, amplifying the model's ability to locate and sustain accurate class divisions. Such strategic modifications to the SoftMax function have shown to significantly elevate classification accuracy, offering a more maintain framework for complex, multi-class categorization tasks. For context and as a foundational concept for subsequent content, the SoftMax function [11] is expressed as follows:

$$L_{SoftMax} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{w_{y_i}^T f(x_i)+b_i}}{\sum_{j=1}^{c} e^{w_j^T f(x_i)+b_j}} \quad 2.14$$

Where N is the training batch size, C is the number of classes, x_i denotes the i_{th} input of samples, $f(x_i)$ is the corresponding output of last FC layer, and y is the corresponding label, w is weight and b is bias [55]. For simplicity, bias taken zero, and use x_i instead of $f(x_i)$.

Angular SoftMax (A-softmax) [11] loss to promote Standard SoftMax is introduced for face verification, consider A-SoftMax for a binary case, let's start by discussing its application in a scenario with two classes, though this analysis can easily be extended to cases involving multiple classes. The softmax loss function, in a two-class scenario, yields Resultant probabilities which guide the assignment of predicted labels: a sample is classified into class 1 if p1 ≥ p2, and into class 2 otherwise. The decision boundary in this context is determined by the expression $(W_1^T - W_2^T)\,x = 0$. This equation can be simplified to $(||W_1||\cos\theta_1 - ||W_2||\cos\theta_2)\,||x|| = 0$, where θ_1 and θ_2 represent the angles between the input vector x and the weight vectors W_1 and W_2, respectively. Specifically, for A-SoftMax, the bias term 'b' is omitted, transforming the standard SoftMax to the following expression:

$$L_{SoftMax} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{||x_i||\cos(\theta_{y_i,i})}}{\sum_{j=1}^{c} e^{||x_i||\cos(\theta_{j,i})}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{||x_i||\phi(\theta_{y_i,i})}}{Z} \quad 2.15$$

Where $z = e^{s(\cos(\theta_{y_i,i})-m)} + \sum_{j=1,j\neq i}^{c} e^{s(\cos(\theta_{j,i}))}$, The angle θ between the weight vector w and the feature vector x is critical for positioning the learned embedding distribution within an angular framework. To enhance the traditional softmax approach, A-softmax introduces two key modifications [56] aimed at improving class separability and feature discrimination:

- **Normalization and bias zeroing**: The first adjustment involves normalizing the weight vectors ($||W_1|| = ||W_2|| = 1$) and setting the biases to zero ($b1 = b2 = 0$). This transformation shifts the decision boundary from being linear to angular, defined by $\cos\theta_1 - \cos\theta_2 = 0$. While this adjustment promotes a more geometrically intuitive

separation between classes, it does not necessarily ensure that the features learned by the model are discriminative.

- **Incorporation of angular margin**: To address the issue of discriminative feature learning, the introduction of an angular margin is proposed. By incorporating an integer margin parameter m (where $m \geq 2$), the model is encouraged to learn features that are not only separable but also clearly distinguishable. The decision rule becomes based on whether $\cos(m\theta_1) - \cos(m\theta_2) > 0$ or $\cos(m\theta_2) - \cos(m\theta_1) > 0$, assigning samples to class 1 or class 2 respectively. Essentially, the model is pushed to ensure that the cosine of the angle between the input and the weight vectors of the correct class, multiplied by the margin m, is greater than the cosine of the angle to the incorrect class's weight vector.

By comparing A-softmax to the traditional softmax loss, the key differences and advantages become clear. A-SoftMax's modifications aim to enhance model performance by:

1. **Promoting feature discriminability**: Through the angular margin, features are learned in a way that emphasizes inter-class differences and intra-class similarities.
2. **Shifting to angular decision boundaries**: By normalizing weight vectors and removing biases, the decision criteria move from linear distances to angular separations, which can be more naturally aligned with how humans distinguish between categories.

These enhancements offered by A-softmax are designed to improve the robustness and accuracy of classification models, particularly in complex, high-dimensional spaces where traditional method may struggle to differentiate between closely situated or overlapping classes.

In the context of A-Softmax loss, for example to evaluate a sample ' x ' that could be part of either class 1 or class 2, which are represented by the weight vectors W_1 and W_2, respectively. The fundamental classification rule imposed by A-Softmax loss dictates that for sample ' x ' to be classified as belonging to class 1, $\cos(\theta_1) > \cos(\theta_2)$. This condition can be equivalently expressed by the inequality $m\theta_1 < \theta_2$. This means that the angle formed by ' x ' and the weight vector of class 1, when scaled by the factor 'm', should be less than the angle formed by ' x ' and the weight vector of class 2. By introducing the multiplicative factor ' m ', the A-Softmax loss effectively enforces a stricter angular criterion for class determination, enhancing the model's ability to distinguish and categorize samples more distinctly [56].

A straightforward definition exists for choosing the angular parameter in additive margin SoftMax [57]. This method of applying additive hyperparameter (m) is more interpretable compared to A-SoftMax loss, and It's given by:

$$\psi(\theta) = \cos\theta - m \quad \rightarrow \quad \text{where } x = \cos\theta \Rightarrow \psi(x) = x - m = \frac{w_{y_i}^T f_i}{\left||w_{y_i}|\right| ||f_i||} \qquad 2.16$$

The parameter m is used for the purpose of learning an angular margin between different feature vectors and $\frac{w_{y_i}^T f_i}{\left||w_{y_i}|\right| ||f_i||}$ means normalization of input x under transformation will be happen.

In the training phase, the hyperparameter labeled m plays a crucial role in utilizing input features. For L-SoftMax loss [58], the margin is determined by the similarity between two feature vectors, which is measured by the cosine of the angle between them. This margin is adjusted by the hyperparameter m, known as a hard angle margin. Here, a hard margin implies that the margin parameter must be a positive integer to expand the Taylor series. This adjustment improves an angular decision margin between classes. Although it requires both forward and backward computation to decide on the angle, it achieves a clearer separation of classes compared to the original softmax. The paper "The Soft-Margin Softmax for Deep Classification" [57] builds upon the fundamental idea of using a soft margin, represented as a real-number distance. This approach simplifies the forward computation and eliminates the need for backward computation. These methods have proven effective in preventing network divergence, focusing instead on compacting similar samples. To illustrate the concepts of hard and soft margins, consider the standard SoftMax framework used in binary classification for a sample x which is correctly classified to class 1 (as opposed to class 2). As the training progresses, the weight allocated to class 1 surpasses that assigned to the input features corresponding with class 2. This process is mathematically represented by the following equation:

$$\left||w_1|\right| \left||x|\right| \cos(\theta_1) \quad > \quad \left||w_2|\right| \left||x|\right| \cos(\theta_2) \qquad 2.17$$

The concept of introducing a hard margin (angular margin) by L-Softmax loss is implemented by formulating an intermediate term that facilitates the classification of samples close to the

decision boundary, effectively distinguishing class 1 from class 2. The mathematical expression of this concept is represented as follows:

$$||w_1||\,||x||\cos(\theta_1) \geq ||w_1||\,||x||\cos(\alpha\theta_1) > ||w_1||\,||x||\cos(\theta_2) \quad 2.18$$

From a different perspective, the angle in L-SoftMax is m times smaller than the angle in the original SoftMax. Consequently, this reduces the angle between the learned feature and w_1. This observation is true for every class. Essentially, L-SoftMax loss constrains the acceptable angle for each class. The complex computations required for both backward and forward propagation by L-SoftMax is made more manageable with Soft-Margin Softmax. This is achieved by reformulating the angle equation to introduce a soft margin, as described below:

$$w_1^T x \geq w_1^T x - m > w_2^T x \quad 2.19$$

In the equation, m represents the distance margin as a real positive number, hence the Soft-Margin SoftMax is expressed as follows:

$$s_i = \frac{e^{w_{y_i}^T x_i - m}}{e^{w_{y_i}^T x_i - m} + \sum_{j \neq y_i}^{k} e^{w_j^T x_i}} \quad 2.20$$

Hence, the Soft Margin SoftMax loss is formulated as follows:

$$L_i = -\log\left(\frac{e^{w_{y_i}^T x_i - m}}{e^{w_{y_i}^T x_i - m + \sum_{j \neq y_i}^{k} e^{w_j^T x_i}}}\right) \quad 2.21$$

"Margin Matters: Towards More Discriminative Deep Neural Network Embeddings for Speaker Recognition" [59] explained specifies three fundamental modifications to standard SoftMax loss functions: A-SoftMax [60], AM-SoftMax [57], and AAM-SoftMax [61]. In A-SoftMax, criteria for the magnitude of the angular margin must be:

$$\emptyset(\theta_{y_i,i}) = \cos(m\theta_{y_i,i}) \leq \cos(\theta_{y_i,i}) \quad 2.22$$

The AM-SoftMax function replaces the multiplicative angular margin used in A-SoftMax loss, which proved to be inefficient for gradient-based optimization methods. Instead, it introduces an additive margin alongside input normalization to control feature vector magnitudes effectively. The key enhancement in defining $\varphi(\theta_{y_i,i})$is:

$$\varphi(\theta_{y_i,i}) = \cos(\theta_{y_i.i}) - m \quad 2.23$$

The AAM-SoftMax (Additive Angular SoftMax) approach is predicated on the concept of arc displacement of an input vector within a hyper spherical space. This method involves the utilization of the following formula:

$$\varphi(\theta_{y_i,i}) = \cos(\theta_{y_i.i} + m) \quad 2.24$$

The formulation could be in what is termed the AAM-SoftMax, defined as follows:

$$L_{AAM-SoftMax} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i}+m))}}{z} \quad 2.25$$

The scale factor 's' is strategically employed to mitigate the issue of vanishing gradients, as outlined in [59]. The specific formulation is as follows:

$$Z = e^{s(\cos(\theta_{y_i}+m))} + \sum_{j=1,j\neq y_i}^{c} e^{s(\cos(\theta_j))} \quad 2.26$$

The AM-SoftMax introduces a constant margin to enhance class separability, but this does not account for the varying angles between sample features and their corresponding class centroids. DAM-SoftMax [14] proposes a dynamic margin for SoftMax, where smaller value of $\cos\theta$ between a sample and its class centroid are penalized with a larger margin. This approach promotes greater compactness within the class feature space. The underlying principle of this technique is visualized to explain the conceptual reasoning.

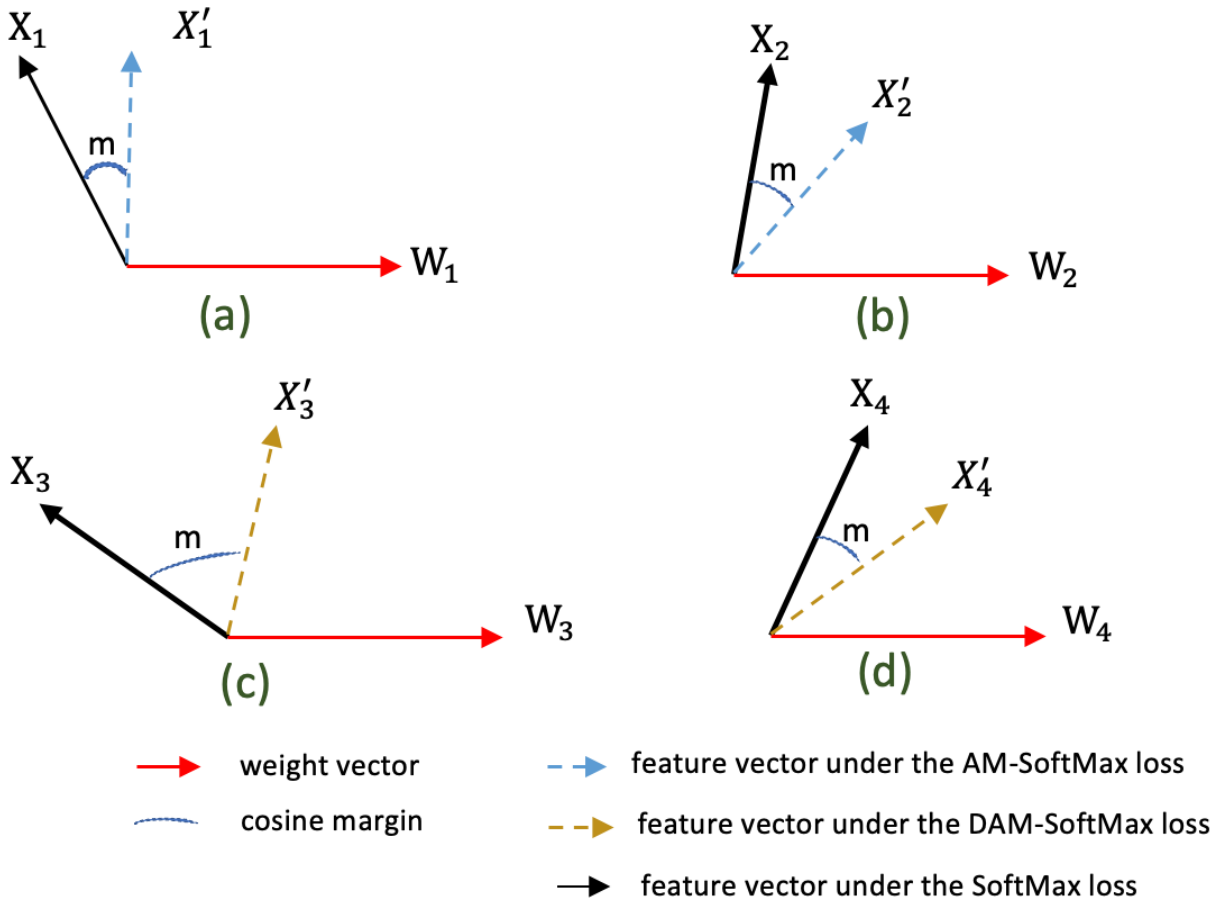


Figure 2. 11 Figures (a) and (b) depict shifts by a constant margin in AM-SoftMax. In contrast, (c) and (d) show transformations by a dynamic margin in DAM-SoftMax, assuming smaller angles indicate greater sample distances from their respective classes. Specifically, in (c), the feature vector (black line) is distant from the class (red line), resulting in a larger additive angle for this training sample compared to the sample in (d) [14].

In AM-SoftMax, constants (a) and (b) in Figure 2.11 are adjusted by a fixed margin, whereas in DAM-SoftMax, (c) and (d) in figure 2.11 are adjusted dynamically. As depicted in (c), the feature vector, represented by the black line, is significantly distant from its class centroid, denoted by the red line. Consequently, a more substantial additive angular margin is allocated for this particular training instance relative to the one in (d). To achieve the objective of learning discriminative features on the hypersphere, which involves both the norms of the neural network output and the angle contributing to the SoftMax posterior , the following computation by Large Margin Cosine Loss (LMCL) for Deep Face Recognition [62] is recommended:

2.27

$$L_{softmax} = -\frac{1}{N}\sum_{i=1}^{N} -\log p_i = -\frac{1}{N}\sum_{i=1}^{N} -\log \frac{e^{f_{y_i}}}{\sum_{j=1}^{c} e^{f_j}} \Rightarrow f_j = w_j^T x = \left||w_j|\right| \, ||x|| \cos\theta_j$$

In the cosine variant of the margin SoftMax function, the verification process compares angles rather than magnitudes between the prediction and ground truth which requires more computation burden. Large Margin Cosine Loss for Deep Face Recognition introduces a margin in the cosine space rather than in the angular space. The norm of the feature vector $||w||$ is constrained to a constant value. Consequently, the posterior probability during training is dependent upon the cosine of the angle rather than the norm, as expressed by:

$$L_{ns} = \frac{1}{N}\sum_{i} -\log\frac{e^{s\cos(\theta_{y_i,i})}}{\sum_j e^{s\cos(\theta_{j,i})}} \qquad 2.28$$

This formulation considers the feature vector f(x) and the weight vector (w) with angle θ for a given sample in a two-class scenario (c_1, c_2), as depicted in the following schematic representation:

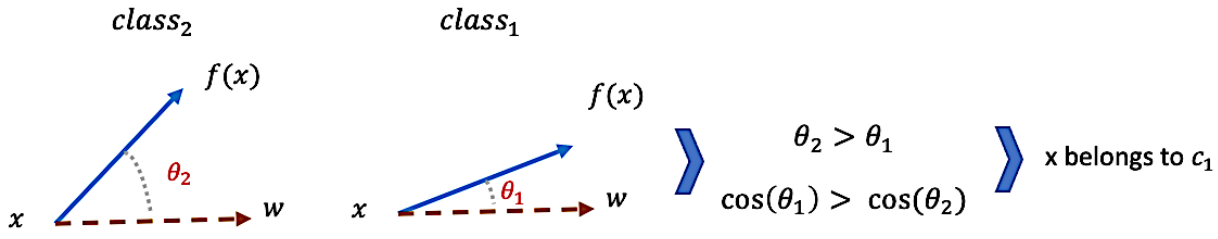


Figure 2. 12 feature vector f(x) and weight (w) for class sample C [62].

Thus, a sample x is classified into class c_1, if $\cos(\theta_1) > \cos(\theta_2)$. The corresponding margin is defined as follows:

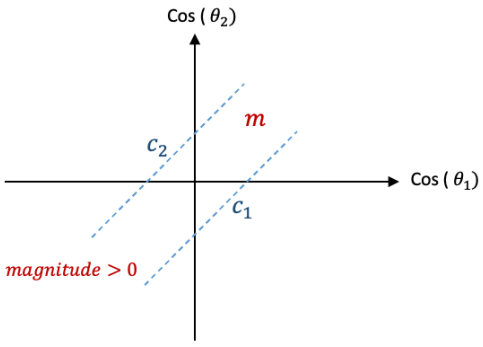


Figure 2. 13 Visualization of sample classification using the cosine variant of the SoftMax function (LMCL) [62].

For a correct classification into class c_1, the criterion is $\cos(\theta_1) - \mathrm{m} > \cos(\theta_2)$,and conversely, for class c_2, the criterion is $\cos(\theta_2) - \mathrm{m} > \cos(\theta_1)$:

$$L_{lms} = \frac{1}{N}\sum -\log\frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)} + \sum_{j\neq y_i} e^{s(\cos(\theta_{j,i})}} \quad 2.29$$

$$c_1 : \cos(\theta_1) \geq \cos(\theta_2) + m$$

$$c_2 : \cos(\theta_2) \geq \cos(\theta_1) + m$$

The concept of Double Additive Margin SoftMax loss (DAMS), as proposed in [63], employs dual additive margin criteria to amplify both intra-class compactness and inter-class separation. The formula is as follows:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s.(\cos(\theta_{y_i})-m)}}{e^{s.(\cos(\theta_{y_i})-m)} + \sum_{c=1,c\neq y_i}^{C} e^{s.(\cos(\theta_c)+m)}} \quad 2.30$$

Where $\mathrm{g}(\theta_c) = \cos(\theta_c) + \mathrm{m}$, the geometric interpretation on the hypersphere manifold for classifying a sample to class 1 should adhere to the proved AM-SoftMax loss rule, which states $\cos(\theta_1) - \mathrm{m} < \cos(\theta_2)$ and implies $\cos(\theta_1) - \cos(\theta_2) < \mathrm{m}$. Conversely, in DAM-SoftMax, the inequality is modified to:

$$\cos(\theta_1) - \mathrm{m} < \cos(\theta_2) + \mathrm{m} \text{ leading to } \cos(\theta_1) - \cos(\theta_2) < 2\mathrm{m} \quad 2.31$$

The hyperparameter selection method introduced by this technique results in greater inter-class margins and reduced intra-class variances compared to AM-SoftMax. From the classification perspective, selecting an optimal value for 'm' enhances generalization by promoting discriminative feature learning on the hypersphere, ensuring that all weight vectors corresponding to a class are closely clustered. A wise choice of 'm' improves robustness to noisy data. However, 'm' should be carefully calibrated; if it is too large, it may restrict convergence, leading to a decline in overall performance, as discussed in [62].

2.3 Brief Overview of Automated Facial Expression Recognition Methods

The early neural network models for facial expression recognition, as introduced by Ekman and Friesen and later implemented by Lisetti et al., adopted a two-step process involving the detection of seven core emotions, followed by classification via a Multi-layer Perceptron (MLP) or RBF. Further progress led to the development of hybrid models [64] that combined wavelet transforms and neural networks [65], where low-dimensional features were transformed into higher dimensions using wavelet Karhunen-Loeve transforms. Attention-based hybrid methods [66], incorporating both Neural Networks and Recurrent Neural Networks (RNNs), were designed to account for the temporal dynamics in feature extraction, while others relied on a 2D Discrete Cosine Transform (2DDCT) approach for feature detection and classification using a constructively designed feedforward network with a single hidden layer [67].

The initiation of Convolutional Neural Network (CNN) applications for facial expression recognition marked a significant departure from hand-crafted feature extraction methods. Pioneered during the FER-Workshop in 2013 [68], CNNs demonstrated a remarkable decrease in generalization error, proving superior in analyzing unseen data. While traditional feature extraction models were replaced by CNN's automated feature learning capabilities, integrating hand-designed modules as auxiliary branches further enhanced the framework's ability to extract nuanced and informative features. “Local Learning with Deep and Handcrafted Features for Facial Expression Recognition” [69] introduces a SIFT module alongside a CNN branch, capturing vector features for each sample using a bag-of-visual-words algorithm, then features are classified via a traditional local-learning algorithm, such as k-means. The innovation in this approach lies in its utilization of multiple parallel branches, incorporating spatial information through a pyramid model for multi-scale inputs, which provides a comprehensive view of the input data. Additionally, this work distinguishes itself by segmenting the input into patches or sub-regions, enhancing local feature extraction. An SVM-Linear classifier is employed within the local-learning framework, transitioning from linear to non-linear classification by designating individual SVM classifiers to closely related test samples within different classes, a technique referred to as one-versus-rest.

Current research emphasizes the significance of assigning appropriate weights as a form of learning uncertainty, as described in [70]. This perspective deviates from the Gaussian distribution assumption, considering the inherent noise in facial images that can lead to misclassification. The proposed uncertainty model is equipped for multi-task learning and accommodates multi-scaled

input images, which are pre-processed to enhance low-resolution images. A supplementary module external to the main ConvNet body is recommended to prompt the model to generate optimal weights. A novel loss function is introduced, tailored to address the challenge of learning from hard versus easily discernible features.

The primary goal of FER models splits into two fundamental objectives: the extraction of discriminative features and the activation of the model to produce informative features that increase the differences between classes in the data representation area. This is achieved through the introduction of auxiliary loss functions that serve as regularization mechanisms. Notable research employs the center loss function, which utilizes the mean and standard deviation of specific classes to reduce the Euclidean distance of intra-class features to their respective class means. By adhering to this principle, the model is refined to discard noisy features that fall outside of class clusters, thus focusing on relevant feature extraction [71]. An auxiliary loss in "Learn from All: Erasing Attention Consistency for Noisy Label Facial Expression Recognition" [72] used Consistency loss for the helper branch with randomly different augmented input for memorizing semantic meaning, helps the overall loss focus on the key part of features, not the whole of them. Another aspect of using augmentation techniques is creating real-life sample input by methods like randomly erasing parts. Because in real life, most FER datasets include low-resolution hard samples. So, researchers prefer to utilize in-the-wild databases by the assumption of models for a noisy condition.

As mentioned, one of the main topics for researchers in the FER domain is the need for sufficient clean training samples. So many methodologies have been proposed to overcome this issue. One is related to augmentation techniques to produce new noisy samples by introducing functions like randomly erasing parts of each input. It means, occluded images deliberately are created to enforce the model to learn noisy samples; then, the risk of overfitting can be reduced [73].

One of the biggest challenges in Facial Expression Recognition (FER) is handling low-resolution images. Utilizing inputs at multiple scales allows for learning of local features and a comprehensive understanding of the feature map's structure. As mentioned in [74], a pyramid-scaled input enhances an ensemble model's ability to interpret images This includes a super-resolution module that addresses the challenge of low-resolution images. It upscales images using traditional algorithms to make up for their lack of precision. Additionally, the system employs a label smoothing function that provides more information about similar classes. For instance, it

reduces the weight of correct labels for 'fear' expressions, which are often confused with 'disgust' and 'surprise' in predictions.

One significant aspect of using a Deep Metric Learning (DML) method for automated facial expression recognition is encouraging the network to improve the discrimination of embedding feature vectors to enhance variation for between-class samples and learn semantically meaningful features in embedding space, which leads to robustness against within-class variations. As mentioned in "Ad-Corre: Adaptive Correlation-Based Loss for Facial Expression Recognition in the Wild" [74], an auxiliary loss is proposed to guide the network to generate higher correlation for intra-class feature vectors and less correlated feature vectors for inter-class samples. So, by penalizing the input pairs, the model can extract more discriminative features and break the symmetric feature vectors of an input image. The focus is on generating discriminative feature vectors to handle intra-class variability and inter-class similarities. By introducing regularization components that produce varied mean vectors for different labels, the model is prompted to generate consistent feature vectors for similar labels, enhancing its discriminative power.

FaceNet2ExpNet: Regularizing a Deep Face Recognition Net for Expression Recognition by H. Ding [75] discusses the risk of overfitting due to the scarcity of annotated training samples. To mitigate this issue, an auxiliary loss function derived from output distributions is employed to reduce the distance between embedding vectors and the mean values of their respective classes, thereby diminishing intra-class variation. The loss function is formulated following the Maximum Likelihood Estimation process, with the aim of harnessing the low-entropy, highly expressive features of facial images. Subsequently, the model is fine-tuned with cross-entropy in an independent training phase, benefiting from label supervision to bolster its discriminative efficacy. The discussion also covers using combined, multi-task learning networks in FER domain. These models face performance constraints due to the costly and time-consuming process of annotating labels, including landmarks and Action Units (AUs). However, the development of such models is crucial for advancing the field and providing robust solutions to the challenges presented by real-world facial expression data.

Some works demonstrate that suitable optimization techniques in training can help the network to diminish problems of vanishing and exploding gradients by using some methods for decreasing learning rate linearity or exponentiality. An adaptive learning rate adjusts the learning rate based on local gradient value, which guides the model to find optima by correctly updating weights.

During the hyperparameter tuning phase, the model learns to identify key features. If the learning rate is too high, the loss function may not converge properly, resulting in divergence or oscillation around the minima. On the other hand, a low learning rate might cause the model to get stuck in non-optimal. Additionally, to improve the model's learning ability and achieve better classification accuracy, a mix of validation and training datasets is used, especially when there are only a few images available for training [71], [76], [77], [78].

Chapter 3

Proposed Neural Network Architecture

3.1 Introduction

The content of this chapter was showcased at the IVADO 2024 [79], and ICCSIT 2022 [80] conferences. It received the second-place award at the 6th Annual Graduate Student Research (GSR) Conference in 2022 [81].

In the field of computer vision, the search for more advanced neural network architectures is constant, driven by the goal of achieving higher accuracy and efficiency in tasks ranging from image classification to object recognition. This work introduces an architecture influenced by U-Net, VGG, and Inception modules, designed for classification tasks. The reason for this standard vision networks property integration is to utilize U-Net's excellent multi-scale feature mapping capability with the VGG's ability to extract high-level features using small receptive fields and the Inception framework's skill in dealing with auxiliary classifier as regularizer and introducing batch normalization in improvement of training stability, thereby creating a robust and flexible model. This introduction prepares for a comprehensive review of the proposed architecture, its application, and the practical assessment of how it performs compared to current standard networks in demonstrating robust generalization across widely used datasets for localization and classification tasks.

This chapter presents a model tailored for image categorization, beginning with an overview of the proposed architecture. Subsequently, the details of each individual component within this framework are described. The model's precision in image classification has been validated using

datasets, including MNIST, CIFAR10, and SVHN and the model's effectiveness is evaluated to a range of recognized classification CNN models.

This chapter introduces a model called EmoSynthNet, designed to efficiently extract information when trained on limited data. This approach addresses the issue of overfitting in deep convolutional networks, which typically require large amounts of annotated data. Insufficient data can block a model's ability to generalize, requiring methods to extend its applicability to wider domains.

Expanding the size of a network introduces more parameters, increasing the risk of overfitting. To oppose this, innovative techniques are explored such as reusing feature maps, optimizing computational resources, and processing information across various scales. These features are then combined, allowing subsequent layers to utilize abstract features from multiple scales. Additionally, by propagating contextual information to higher resolution layers through an expansive path, the model's performance is further enhanced as outlined in this chapter.

3.2 Discussion of EmoSynthNet Framework

The architecture of proposed model ' EmoSynthNet' is designed with a dual-branch approach, each focusing on distinct computational criteria and perspectives. The following sections will delve into the specifics of their construction, comparing how each branch contributes to the overall performance and efficiency of the model.

3.2.1 Discussion of The Main Branch

EmoSynthNet's main branch, depicted in Figure 3.1, takes inspiration from the VGG19 architecture by Zisserman and Simonyan [2]. This represents the first significant exploration into the impact of depth within convolutional networks, a concept pioneered by VGG to improve the detection of fine details in images. By refining the convolutional pathway, VGG19 aimed to simplify complexity and reduce the number of parameters, effectively lowering the likelihood of overfitting, especially in situations where data is limited. The approach in EmoSynthNet extends this methodology, aiming to further enhance image analysis capabilities while maintaining a balance between model complexity and performance.

VGG underlines the importance of depth in computer vision models by presenting two distinct architectures with 16 and 19 layers and represents a significant advance in the domain of visual representation. A key feature of VGG is its use of smaller receptive fields, scaling down from AlexNet's [82] 11x11 to a more compact 3x3. This strategic choice enhances the model's ability to incorporate more non-linear activation layers, which in turn mitigates the risk of overfitting. Additionally, VGG maintains a consistent kernel size of 3x3 throughout its architecture and utilizes 1x1 convolutions as linear transformations for channel-wise feature integration and improve computational efficiency.

The proposed model (EmoSynthNet) , reflecting the efficient high-resolution image processing of VGGNet, adopts a deep CNN framework.

The EmoSynthNet's architecture reveals in a main branch as shown in Figure 3.1, where the main branch is divided into two distinct segments: an analytical segment consisting of blocks 1 through 4, and a synthetic segment including block 5 alone.

The analytical section is marked by the use of 3x3 kernel sizes, recognized for their computational efficiency, and 1x1 kernels that serve as linear transformers of input channels, enhancing the decision function's non-linearity without modifying the convolution layers' receptive fields. A 1 x 1 convolution, often termed as a 'network in network' layer, has unique capabilities. It acts as a flexible tool within a convolutional neural network. By applying this convolution, the dimensionality of the input can be reduced, creating a compressed representation that is more manageable for the network. This is particularly useful when there is a need to mitigate the computational load without losing critical information. Beyond just reduction, 1 x 1 convolutions can also increase the network's depth while keeping the spatial dimensions Unchanged. This allows the model to learn more complex features without expanding the size of the feature maps. It's like having a mini neural network for each pixel that can process all the information from the channels at that pixel. Furthermore, by introducing non-linearity, 1 x 1 convolutions can enhance the decision-making capabilities of the network. When stacking these layers, they effectively become a multi-layer perceptron for each pixel location. This can be especially beneficial when dealing with tasks that require understanding of fine-grained details, as it allows the network to make more sophisticated decisions based on local pixel information [26]. [82]

Unlike VGG19, EmoSynthNet introduces shortcut connections between 3×3 filter pairs . This, along with Batch Normalization after each convolution and prior to activation layer [83], and strategic Dropout integration, marks a significant architectural advancement. To maintain the integrity of the features and facilitate the reuse of feature maps, skip connections are strategically integrated within blocks 1 and 3. The Residual Network (ResNet) architecture [32], introduced and utilized shortcut connections to transmit information smoothly, reduce training time and improve accuracy, without increasing complexity or the model's parameter count that identify and amplify relevant features, suppressing the unnecessary.

In the process from block 1 to block 4, each convolutional layer concludes with an Exponential Linear Unit (ELU) activation to prevent vanishing gradient and improve dying ReLU problem for faster convergence and better generalization. This layer is immediately followed by a Batch Normalization (BN) layer, ensuring consistency and stability in the network's learning process. Notably, as progress through the blocks, there is a progressive doubling in the number of channels, beginning from 32 in the first block and escalating to a robust 512 channels block 4, reflecting an intentional design to enhance the network's capacity for feature extraction. EmoSynthNet's block 2 has four conv layers with 128 channels, while block 1 has four conv layers with 64 channels, except the initial one with 32 channels, block 3 has four conv layers with 256 channels, and block 4 has three conv layers with 512 channels.

In Figure 3.1, the VGG19 architecture is illustrated on the left, blocks 1 and 2 characterized by its use of two convolutional layers followed by a MaxPooling layer while blocks 3 to 5 characterized by its use of four convolutional layers followed by a MaxPooling layer. This design choice aims to reduce overfitting and expanded the field of view by merging contextual details and disregarding precise location data. On the right, EmoSynthNet's customization is depicted, where it improves upon the VGG19 concept by implementing consecutive 3×3 convolutional layers and substituting the standard MaxPooling with Average Pooling. This modification is critical for preserving the data's seasonality and trends while maintaining residual information. This arrangement helps balance consideration of low-level features with network complexity.

The EmoSynthNet main branch combines two influential frameworks in vision tasks, specifically taken from idea of U-Net [6] framework. U-Net is architected for multi-scale feature mapping, utilizing an encoder to concentrate high-level complexities into refined features, combined with a decoder for high-resolution mapping. The common issue of vanishing gradients

is addressed through skip connections, bridging contraction steps with expansion paths, thereby preserving detail-rich context. U-Net's design facilitates feature extraction from minimal data sets.

While convolution layers reduce image resolution by downsampling, a synthetic segment opposes this effect by employing transposed layers to restore detail. The addition of transposed layers after the analytic segment enhances the network's effectiveness with limited data, improving the precision of localization. The synthesis segment features four layers designed for dimension expansion: two transposed convolutional layers and two custom upsampling layers. Transposed convolutions increase the feature maps' spatial dimensions using adjustable parameters. In contrast, upsamplers enlarge spatial dimensions by a factor of two without relying on learnable parameters. The incremental approach of upsampling using smaller kernels significantly cuts down on the number of parameters, offering a more efficient alternative to wide-ranging upsampling that typically requires medium to large-sized kernels.

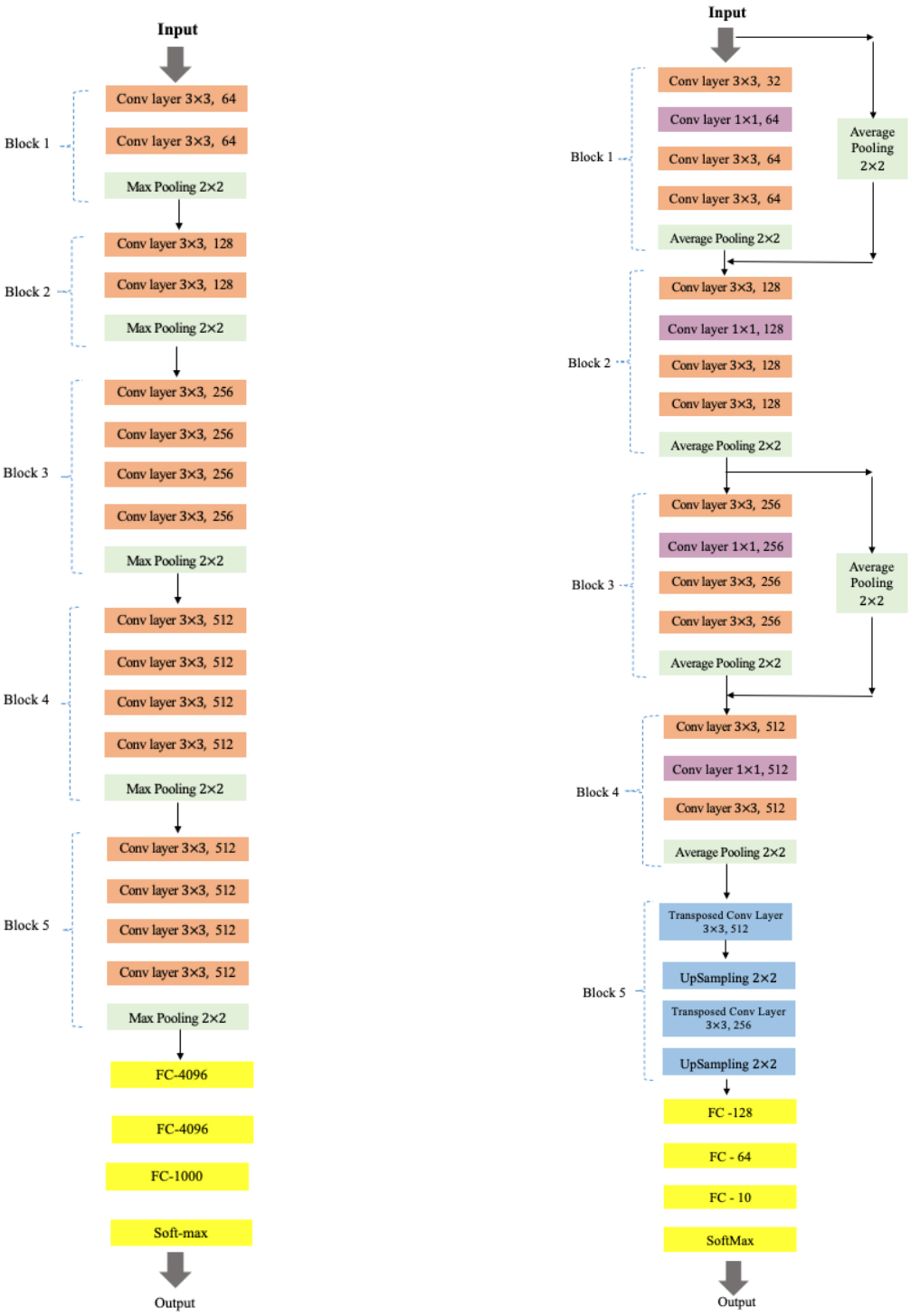


Figure 3. 1 Left: Schematic Diagram of the VGG19 with 19 layers, right: Schematic of the main branch of the EmoSynthNet with 20 layers.

3.2.2 Discussion of The Side Branch

Adding depth to the network often improves accuracy but can also increase computational load and the risk of overfitting, especially in data-constrained environments. Therefore, pairing an auxiliary classifier with the main classifier can sharpen early-stage feature discrimination and boost training stability.

The concept of an auxiliary structure in EmoSynthNet design as shown in Figure 3.2 (right image) is inspired by the inception module [25], as presented in Figure 3.2 (left image). In the specific case of Inception, an auxiliary classifier was introduced as a regularizer to aid in stabilizing training. The inception module's design, aimed at expanding filter banks, served as the foundation for the auxiliary module proposed in this work. This module has been modified for EmoSynthNet's auxiliary modules in terms of the type and number of layers and incorporates short connections within the module itself. The approach integrates additional branches within the network to strengthen feature extraction capabilities and improve overall model robustness by providing multiple scales of observation within the same layer.

Inception's module architecture is distinguished by its scalability, adaptability, and efficiency in the field of computer vision. It provides a framework that leverages sparse connections for the majority of computations, coupled with aggregated features. The architecture incorporates a sequence of stacked convolutions and, in certain instances, max pooling with a stride of two. This design allows the model to process visual information at multiple scales simultaneously, enabling it to abstract features across scales and reinforce network learning. The improved gradient back-propagation across the layers is a direct result of this multi-scale feature abstraction.

The EmoSynthNet side branch with 14 layers is composed of a pair of auxiliary modules, as presented In Figure 3.2 (right image), followed by an average pooling layer. Each auxiliary module with seven convolutional layers, surpassing the inception module (left image) by one layer. These modules are organized into three sub-branches, integrating a max-pooling layer with a stride of 1, compared to the inception's stride of 2. A key distinction lies in the decision to minimize the width and extend the depth of each module and use of short connections. In place of the inception module's four branches, the EmoSynthNet's auxiliary modules utilize three branches, which include two sequential convolution layers of 3×3 and 5×5, enhancing the network's ability to capture complex features at various scales.

The EmoSynthNet auxiliary pathway's design strategically merges the outputs from the three sub-branches. It also utilizes skip connections from the second convolutional layers of the first and third sub-branches, with the goal of expanding the feature map collection. To balance the spatial dimensions of the side branch output, a 7×7 kernel size pooling layer is employed. This adjustment is crucial for allowing a smooth concatenation with the output from Block 5's first transposed convolution layer, as detailed in Figure 3.1.

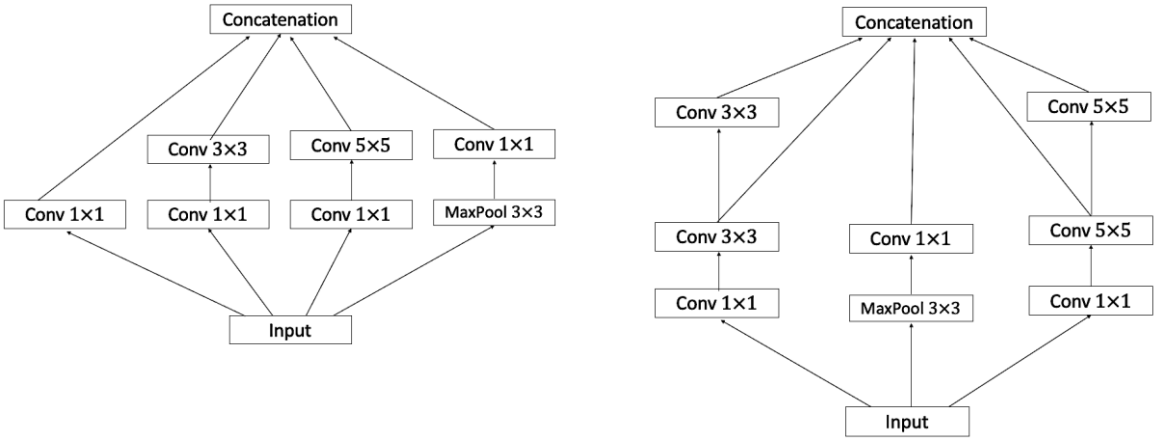


Figure 3. 2 Structure of the Inception Module (Left) and the Auxiliary Module of the Proposed Framework (Right).

3.2.3 Design and Analysis of EmoSynthNet Framework

The Inception U-Net architecture [34] combines the strengths of Google's Inception model [25] with the robust structure of U-Net [6], replacing traditional convolutional layers with Inception modules. By doing so, the model achieves greater depth using U-Net's proven framework and increased width from the Inception modules' capacity to handle multi-scale information. The resulting architecture of the EmoSynthNet framework features a main branch akin to U-Net for depth and a side branch for width that draws on the principles of the Inception modules (as detailed in section 3.2.2). Additionally, the Deep Residual U-Net [36] architecture incorporates residual connections that ease the training of deep networks by facilitating a smoother flow of information. This concept has guided the incorporation of short connections throughout the framework, enhancing the efficient propagation of gradients and improving learning in deep network structures.

The proposed model's architecture (EmoSynthNet), delineated in Figure 3.3, is structured into a primary branch as shown in Figure 3.1 Following VGG's paradigm, and a side branch as shown in Figure 3.2. Following Inception's paradigm,

EmoSynthNet's main branch includes a total of 34 layers, the primary branch with 20 layers retains spatial resolution from the input across all maps within blocks 1 to 4. EmoSynthNet's side branch includes a total of 14 layers for multi-scale processing.

The diagram of the proposed model is shown in Figure 3.3, which consists of a main branch and a side branch (Figure 3.2). The main branch of EmoSynthNet has two parts: an analysis segment consisting of blocks 1 to 4 and a synthesis segment consisting of block 5. The side branch consists of two auxiliary modules and an average pooling layer. The main branch of the network has 20 layers. In blocks 1 to 4, the spatial resolution of all the maps produced by the various convolutional layers is the same as that of the input to the block. Blocks 1 and 3 use skip connections. The analysis segment of EmoSynthNet uses kernel size 3×3. Each convolutional layer of blocks 1 to 4 is followed by an exponential linear unit (ELU) and a batch normalization (BN) layer. In the analysis segment, the number of channels doubles after each block, starting from 64 at the output of the first block to 512 at the output of the fourth block. The synthesis segment has a total of 4 layers, of which two are transposed convolutional layers, and the other two are upsamplers. Transposed convolutional layer increases the spatial size of the feature maps through learnable parameters. The upsampler increases the spatial size of each feature map from $(n \times n)$ to $(2n \times 2n)$ without using any learnable parameters. The benefit of upsampling in stages using small-sized kernels is that it reduces the number of parameters by a significant amount compared to that in multi-scale upsampling using kernels of medium to large size. The skip connections allow effective information transfer from the analysis segment to the synthesis segment without an increase in the number of parameters.

The side branch displayed in Figure 3.3 is composed of two auxiliary modules having the same structure, as shown in Figure 3.2. Each auxiliary module consists of 7 convolutional layers followed by a rectified linear unit (ReLU) divided into 3 sub-branches with one maxpooling layer. The first sub-branch includes a 1×1 convolution layer with 64 filters followed by two successive 5×5 convolution each with 96 filters. The second sub-branch consists of a max-pooling layer followed by a 1×1 convolution layer with 64 filters. The third sub-branch has the same structure as the first sub-branch except that the first convolution layer has 128 filters, while the second and

third convolution layers use 192 filters using kernels of size 3×3 instead of 5×5. The outputs of the three sub-branches, along with the outputs from the second convolution layers of the first and third sub-branches, are concatenated with the aim of increasing the numbers of feature maps. Then, a pooling layer with a kernel of size 7×7 is used to adjust the spatial size of the output of the side branch so that it can be concatenated with the output of the first transposed convolution layer of Block 5.

Assuming the size of the input image to the network to be $m \times m$, it's seen from Figure 3.3 that the input to the first transposed convolution layer in Block 5 is of size $^{m}/_{16} \times {}^{m}/_{16}$. The size of the output of the first transposed layer using a kernel of size 3×3, stride 1 and without padding is $(^{m}/_{16} + 2) \times (^{m}/_{16} + 2)$. After concatenation with the output of the side branch, the output has 1408 feature maps (512 maps from the first transposed convolution layer and 896 maps from the side branch), each of spatial size $(^{m}/_{16} + 2) \times (^{m}/_{16} + 2)$. Then, the upsampling layer increases the spatial size of the feature map to $(^{m}/_{8} + 4) \times (^{m}/_{8} + 4)$. This process of the transposed layer is repeated using 256 filters, each using a kernel size of 3×3. The output of this transposed layer is upsampled using a 2×2 kernel size. Thus, the output of block5 has 256 maps, each of the spatial size $(^{m}/_{4} + 12) \times (^{m}/_{4} + 12)$.

Block 6 consists of a flattening layer followed by three dense layers containing 128, 64, and 10 neurons, respectively and a SoftMax activation layer. The output of block 5 is fed to the flattening layer that produces a one-dimensional feature vector of size $256 * (^{m}/_{4} + 12)^2$. Thus, the numbers of the outputs from the three dense layers are respectively 128, 64, and 10. A SoftMax activation function completes the sequence, producing a probability distribution over the different classes, effectively indicating the model's prediction for class membership.

The model's structure is optimized to ensure an effective balance in resource distribution across computational dimensions. This includes parallel computation capabilities, shared weights, and a significant reduction in dimensionality, leading to more efficient memory utilization. Additionally, the utility of residual connections is explored.

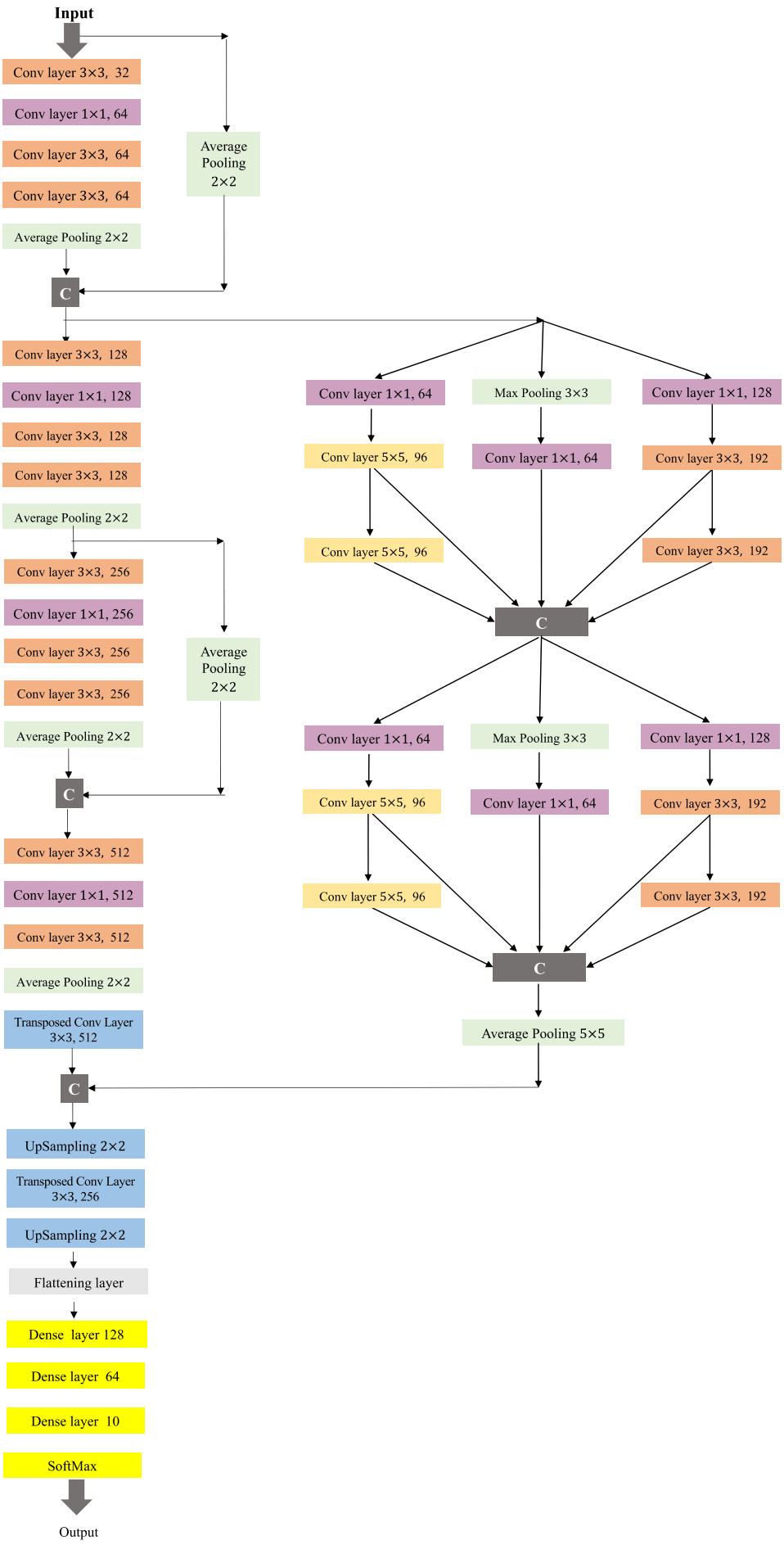


Figure 3. 3 Schematic representation of EmoSynthNet architecture consists of main and side branches.

3.3 Experimental Results

In this section, the classification accuracy of the EmoSynthNet is evaluated through simulations. Three widely used object recognition datasets: MNIST, CIFAR10, and SVHN are employed to benchmark the model's performance. Comparative analysis is carried out between EmoSynthNet against established networks results documented in published literature, and against vision networks deployed in this work, indicating the efficacy of the EmoSynthNet in classification task.

3.3.1 Datasets

MNIST

The MNIST dataset, distinguished for its straightforward structure and small image sizes, serves as a benchmark for assessing image classification algorithms. It comprises 70,000 grayscale images of handwritten digits from 0 to 9, arrayed across 10 classes, with each image sized at 28×28 pixels. The dataset is split into a training set of 60,000 images and a testing set of 10,000 images, necessitating minimal pre-processing for effective classifier training. Figure 2.3(a) presents a sample from the MNIST dataset.

CIFAR10

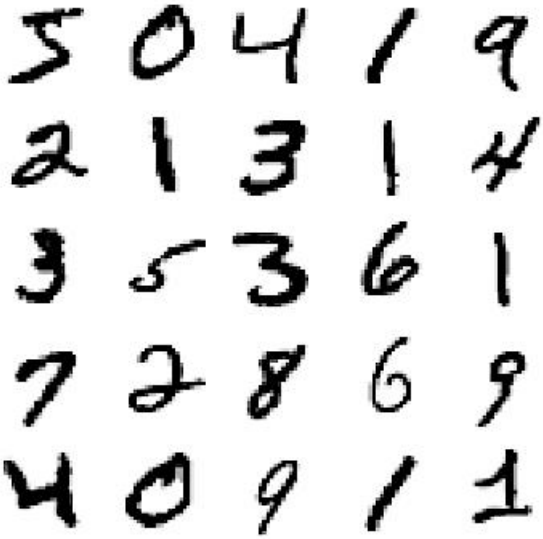
CIFAR10, known for its diversity and real-world complexity, consists of 60,000 color images categorized into 10 distinct classes. Each 32×32 pixel image is part of a collection where 50,000 constitute the training set and 10,000 make up the testing set. This dataset ensures mutual exclusivity among classes, avoiding overlap between similar categories such as 'automobile' and 'truck' or 'dog' and 'cat,' as illustrated in Figure 2.3(b). The unnamed dataset presents an array of more intricate and varied classes when compared to CIFAR10, encompassing both fine-grained and broader categories.

SVHN

The Street View House Numbers (SVHN) [84] dataset, known for its application in natural scene image recognition, is a pivotal resource in machine learning and object recognition research. This dataset comprises labeled digits captured from street-level photographs, presented in both cropped

and original formats. Notably, SVHN facilitates the development of advanced algorithms by offering a real-world challenge with a straightforward setup, minimizing the need for extensive data preprocessing and formatting. The dataset includes images printed digits (from 0 to 9) and divided to a substantial training set contains 73257 images, test set 26032 and the extra set 531131, each sized at 32x32 pixels. These images are systematically categorized into a training set, a testing set, and an additional set derived from house numbers visible in Google Street View images. This categorization is designed to support a wide range of experiments, focusing on practical, yet complex, real-world problems. For instance, the experiments highlighted in Figure 3.10 utilize the cropped version of the dataset, specifically extracted from street house numbers, to demonstrate the application of various machine learning models. This approach underscores the dataset's relevance and versatility in tackling real-world challenges in image recognition.

MNIST	Original	Use in experiment
Training set		60000
Test set		10000
Classes		10



SVHN	Original	Use in experiment
Training set		73257
Test set		26032
Classes		10

CIFAR10	Original	Use in experiment
Training set		50000
Test set		10000
Classes		10

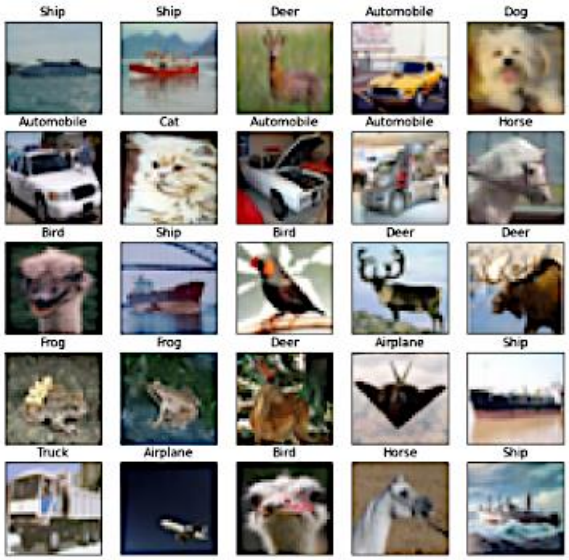


Figure 3. 4 Sample images from datasets: MNIST (Top), SVHN (Down left), CIFAR10 (Down right) .

3.3.2 Training Methodology Analysis

The network is trained using backpropagation (BP), which updates the weight of the network to acquire an optimal solution by calculating the gradient of the loss function relative to the weights of the network. This process is repeated until the loss function reaches a minimum value. During training, each batch consisting of 64 images is fed to the network. Hence, the network sees shuffled images in small numbers during the process of updating the weights, thus reducing the computational demand. The training dataset was augmented by a factor of 0.2 using zooming, shearing, and flipping. The number of training epochs is limited to 100 in experiments. The training is performed in an end-to-end manner, using the Adam optimizer with 0.9 momentum. The model was observed to converge to an optimal solution efficiently for a learning rate of 0.001 and decreased by a factor of 0.1 every time the validation error is not decreasing after every 5 epochs. A weight class function that assigns more loss weight for classes with fewer labels is applied. A two-scale training is implemented wherein the training starts with a scale setting of 1/255, and then, switching to a scale value of 1/512 in the later part of training to speed up the training similar to VGG models training. Simulations are carried out using Google Collaboratory Pro, with GPU and High-Ram runtime configuration settings.

Figure 3.5 (left column) maps out the final 20 epochs, showing model accuracy and validation loss on the MNIST (Figure 3.5 (a)) and CIFAR10 (Figure 3.5 (c)) datasets. These outcomes arise after fine-tuning hyperparameters to enhance the model's capacity to generalize. Additional tests involved implementing regularization techniques, confirming the consistency of these improvement trends across datasets.

In Figure 3.5 (right column) validation loss for MNIST (Figure 3.5 (b)) and CIFAR 10 (Figure 3.5 (d)) has been shown. On MNIST, the loss plateaus after the 10th epoch, and a similar trend is noted on CIFAR 10, where the validation loss stops to decline after the 14th epoch. Despite this, the training loss continues to diminish across all epochs for both datasets. This pattern is mirrored in the accuracy plots for MNIST in Figure 3.5 (a) and CIFAR10 in Figure 3.5 (a), with no substantial improvements beyond epochs 10 and 14, respectively. These patterns are classic signs of overfitting, suggesting that the model isn't learning new features beyond these points.

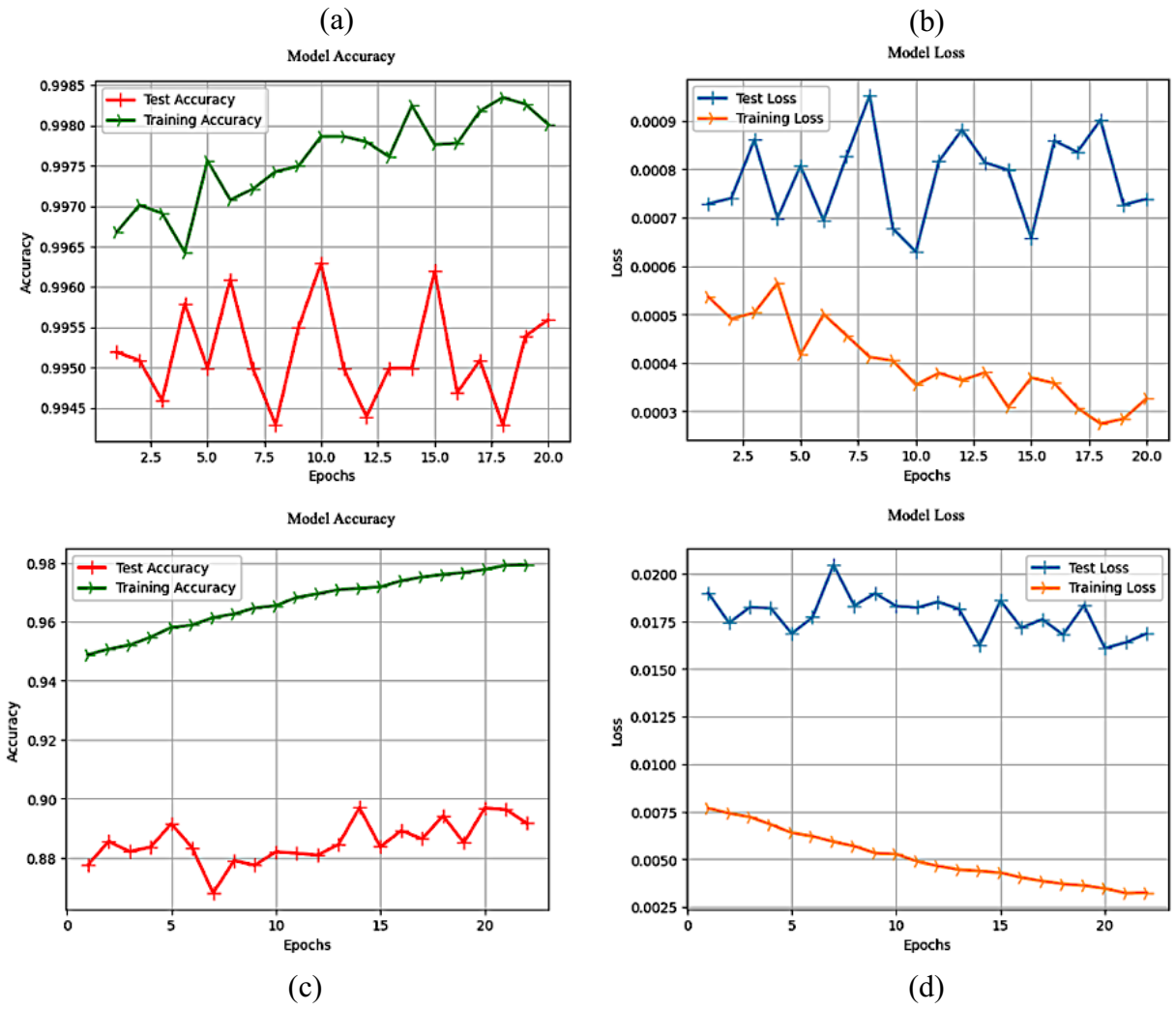


Figure 3. 5 Training / Validation accuracy and loss of the EmoSynthNet of Figure 2.1 implementation on MNIST (a) and the accuracy pruned over 99%, CIFAR10 (b) and the accuracy pruned over 85%.

The confusion matrices for MNIST and CIFAR10, as shown in Figure 3.6, illuminated on classification performance using Cross-Entropy loss and Adam. For CIFAR10, a considerable pattern appears with 'cat' and 'dog' classes often confused, revealing misclassification rates of 8.9% FN (False Negative means missed correct labels (Type I error)) and 9.4% FP (False Positive means Incorrectly predicted positive cases (Type II error)). respectively. This may highlight the difficulty in differentiating between similar features of noisy samples. The confusion matrix for CIFAR10 also indicates that certain classes with similar shapes and textures are prone to higher misclassification rates. For example, 'cats' being mistaken for 'frogs' at a rate of 6.4% FP which points to some visual similarities between these categories that the model struggles to analyze. In contrast, the MNIST dataset shows minimal confusion, with the largest misclassification being

only 0.001% between the numerals such as ‘0’ and ‘8’, ‘0’ and ‘2’, ‘0’ and ‘7’. This demonstrates the model's exceptional accuracy in distinguishing clear, distinct digit patterns for number zero which may be caused due to noisy samples for class zero or lack of ability to extract sufficient features for this specific class. For MNIST, despite the overall high accuracy, subtle confusions suggest that the model may benefit from additional regularization [20] to refine its precision for these similarly shaped numerals. Addressing the misclassifications as indicated by the confusion matrix is crucial when applying regularization techniques, such as penalizing the loss function or implementing dropout. In the case of MNIST, the error range is tiny, maintaining above 99% accuracy most of the time. CIFAR10's errors, however, stem from misidentification between classes with similar features. Although errors in MNIST are small, they highlight the necessity of regularization strategies to effectively distinguish hard-to-separate samples with shared characteristics.

This comparison underlines the importance of customizing regularization techniques to specific challenges presented by each dataset. The application of dropout and penalty adjustments to the loss function can be informed by these findings. Similarly, adjusting the loss penalty for CIFAR10 could help the model learn more robust features for classes with high semantic similarity but distinct characteristics, improving the distinction between classes like 'cat' and 'dog'. These customized regularization techniques aim to enhance the model's generalization capabilities, thereby reducing the error rates and improving the accuracy across both datasets.

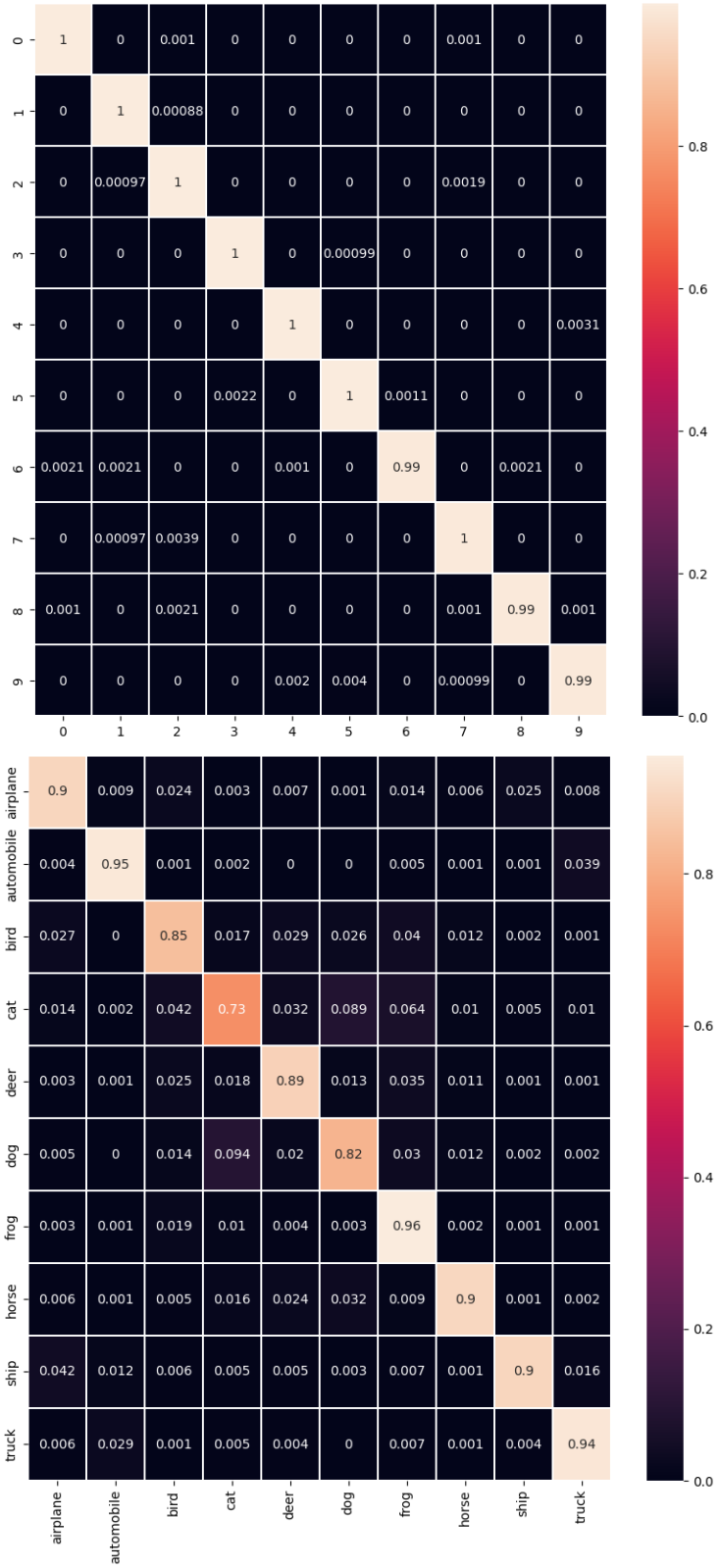


Figure 3. 6 Confusion matrix for the EmoSynthNet implementation on MNIST (Up) and CIFAR10 (Down).

Grad-CAM, or Gradient-Weighted Class Activation Mapping [85], leverages the gradients flowing into the final convolutional layer to generate a localization map, emphasizing pivotal regions within the image for predicting a particular class. The strength of Grad-CAM lies in its ability to identify class-specific regions within non-attention CNN models, all without necessitating alterations to the network architecture or re-training the model. Grad-CAM enhances the interpretability of a classifier [86], [87], [88] by attributing significance to individual neurons. It computes the gradients of the output category, denoted as y^c (prior to the SoftMax operation), with respect to the feature map activations, A^k, of the convolutional layer. These convolutional layers inherently encapsulate class-specific semantic information, similar to 'object parts.' By globally averaging the gradients across the width and height (i, j) dimensions, it specifies the importance weights, α_k^c, for the neurons, according to the formula:

$$\alpha_k^c = \underbrace{\frac{1}{Z}\sum_i \sum_j}_{\text{Global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{Gradient via backprop}} \tag{3.1}$$

These importance weights, α_k^c, effectively linearize the deep network's decision-making process, highlighting the significance of specific feature maps for the target class. A linear combination of these weights with the activation maps yields a coarse heatmap that aligns with the dimensions of the final convolutional feature map.

Figure 3.7 illustrates the Grad-CAM visual explanations for the EmoSynthNet demonstrates that the neurons highlighted by Grad-CAM as crucial for the class-discriminative localization are vital for the identification of the object class, independent of bounding box annotations. Given that the datasets utilized for training contain images with a single object, the model exhibits amplified precision in recognizing images with individual objects, particularly those classes featured within the training datasets, such as dogs, cats, and vehicles. Convolutional layers are adept at preserving spatial details that fully connected layers might overlook. Therefore, we expect the concluding convolutional layers to offer an optimal balance of high-level semantics and complex spatial data. These layers' neurons detect semantic, class specific features within an image, such as distinct object parts. Grad-CAM exploits the gradients flowing into the CNN's final convolutional layer to

allocate significance to each neuron for specific decision-making processes. This method has broad applicability, enabling the interpretation of activations across any layer in a deep network.

Backpropagating the signal for the class of interest to the relevant rectified convolutional feature maps yields the initial Grad-CAM localization, visualized as a blue heatmap. This heatmap outlines the relative 'importance' of different regions in making a classification decision, showing where the model focuses its attention. However, upon examination, particularly in Figure 3.7's first and fourth rows, the heatmap, while highlighting the recognized object a car or a hand isn't exceptionally precise. The highlighted area includes parts of the object and the surrounding environment. It indicates that the model detects a car or hand, but doesn't clarify which specific features confirm the classification. This is where deeper layers come into play, surpassing simple edge detection of earlier layers. They begin to understand and emphasize the distinct shapes that define objects, such as the details of a car's design or the curvature of a hand. These defining features start to emerge, marked by vibrant green and yellow shades, against the context of less relevant areas shaded in red.

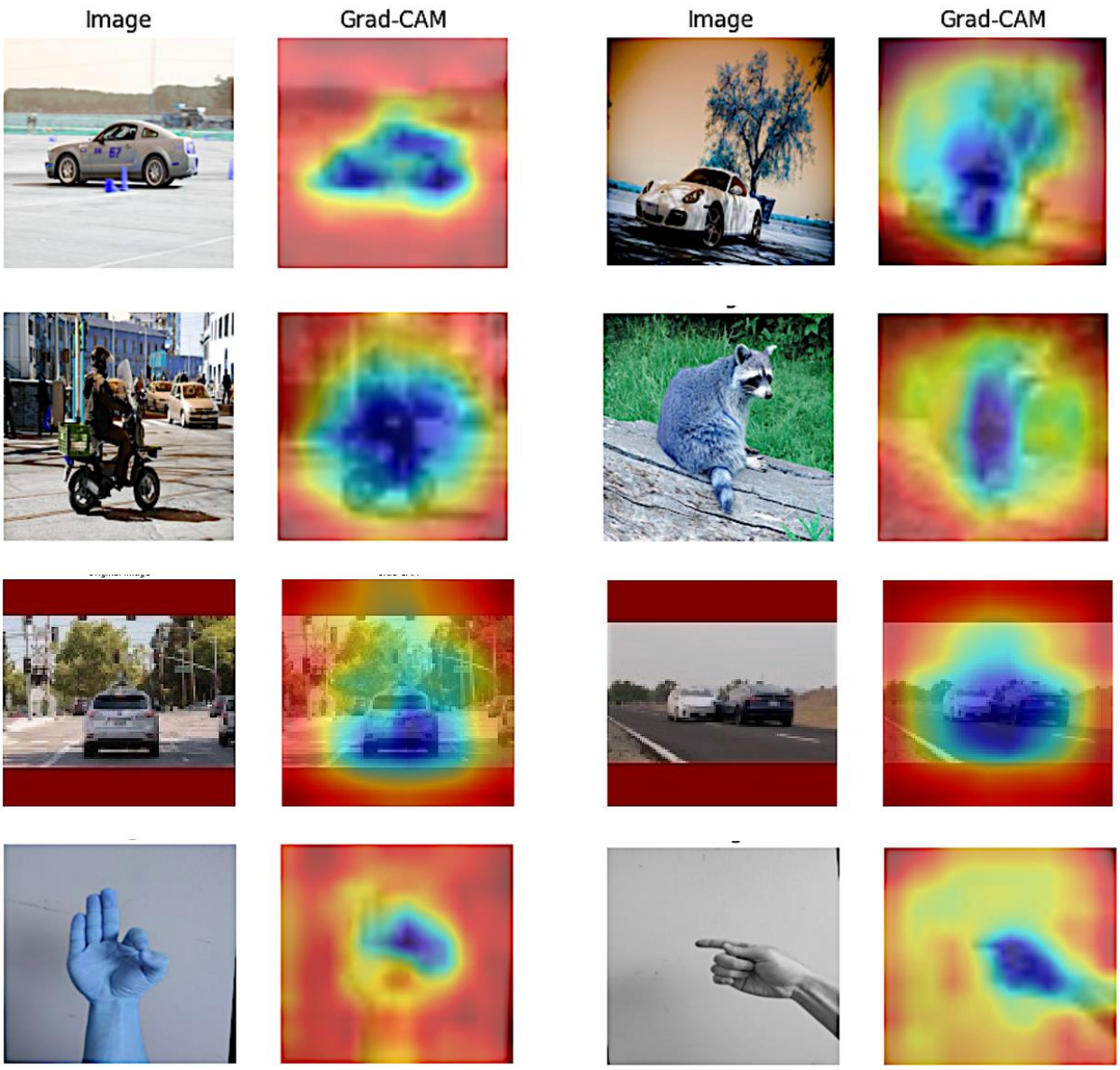


Figure 3. 7 From left to right, each two columns display the qualitative results for the Original Image, and the Grad-CAM Visualization, respectively, accompanied by their confidence levels. Grad-CAM effectively localizes the categories of the original objects.

3.3.3 Comparative Performance Analysis

3.3.3.1 Evaluating Results with Contemporary Vision Networks

Table 2.1 shows the classification accuracy of the EmoSynthNet and contrasts it with seven existing vision frameworks, each employing a unique network performance methodology. Uniformly, these models leverage the ADAM and CE for error reduction. The table's fourth and

fifth columns detail the classification performance on the MNIST and CIFAR10 datasets, correspondingly. Notably, the best, second-best, and third-best accuracy results are denoted in red, blue, and green fonts. Table 2.1 shows that the EmoSynthNet surpasses all comparative methods in accuracy. Specifically, the proposed architecture achieves a 1.13% improvement over the enhanced InceptionV3 + SwAT model on the C10 dataset and a 0.05% increase compared to the VGG19 + SwAT model on the MNIST dataset. The superiority of EmoSynthNet is further emphasized by its accuracy increments: by 0.02% over the second-best method, which incorporates Cross-Entropy (CE) loss, and by 0.05% over the third-best on MNIST. For the CIFAR10 dataset, the enhancements are even more clear, with a 0.35% and 1.13% accuracy raise over the second and third-ranked methods, respectively.

These results are especially significant considering they were attained irrespective of the choice between Cross Entropy (CE) and Mean Squared Error (MSE) loss functions. Such outcomes suggest that while certain regularization approaches may be effective for particular architectures and datasets, their effectiveness is not widely applicable. The EmoSynthNet's architecture, therefore, demonstrates a comprehensive adaptability and effectiveness across two widely used object recognition datasets.

Table 3. 1 Comparative analysis of classification accuracy among various methods on MNIST and CIFAR-10 benchmarks: The most effective method is emphasized in bold red, followed by the second most effective in blue, and the third in green.

Methods	Loss	Optimizer	MNIST	CIFAR10
DMoCo-V1 (2021) [89],	CE	Adam	97.6%	85.0
ResNet-26 (2023) [90],	CE	SGD	97.2%	91.1%
Inceptionv3 (2023) [91],	MSE	Adam	99.51%	90.32%
ResNet50 (2023) [91],	MSE	Adam	98.83%	81.76%
DensNet121 (2023) [91],	MSE	Adam	99.36%	84.92%
VGG19 (2023) [91],	MSE	Adam	99.58%	83.98%
ResNet50 (2022) [92],	MSE	Adam	99.56%	86.21%
EmoSynthNet,	MSE	Adam	**99.63%**	89.69%
EmoSynthNet,	CE	Adam	99.61%	**91.45%**

3.3.3.2 Comparing Performance with Existing Vision Networks

Table 3.2 highlights EmoSynthNet's computational efficiency with greater FLOPS (approximately 1.12 Giga Floating Point Operations Per Second (FLOPS) computed as a unite of speed in computation) in comparison to established vision networks like MobileNetV2, EfficientNetB0, ResNet50, VGG19, and DenseNet121. EmoSynthNet presents a compact design with fewer parameters (~34 million) and a compact memory footprint (~82 MB). Its structure, 34 layers deep, ranks just after VGG19, signifies a more efficient network that doesn't sacrifice processing speed.

Table 3. 2 Details of experimented comparative CNN models

Methods	Parameters (Millions)	FLOPS (Billion)	Approx. Size (in MB)	Depth
MobileNetV2 [93],	4	0.006	14	105
EfficientNetB0 [88],	5	0.008	29	132
ResNet50 [92],	26	0.078	98	107
VGG19 [91],	144	0.417	549	19
DensNet121 [94],	8	0.057	33	242
EmoSynthNet,	**21**	**1.120**	**82**	**34**

Performance evaluations on CIFAR10 and SVHN datasets, detailed in Tables 3.3 and 3.4, showcase EmoSynthNet's enhanced ability to learn features effectively with smaller datasets relative to standard classification networks deployed in this work. In unseen test data, EmoSynthNet demonstrates robust generalization, and aspect highlighted through superior precision, recall, and F1-score metrics against conventional vision architectures.

Precision, recall and F1-score metrics calculated in Tables 3.3, 3,4 are essential for assessing classification performance beside accuracy, and are explained as following:

- **Precision**: The ratio of correctly predicted positive instances to all predicted positive instances, highlighting accuracy. A high precision indicates few false positives, essential where such errors are costly.
- **Recall**: Known as sensitivity, recall measures the fraction of correctly predicted positives from all actual positives, reflecting the model's ability to capture relevant instances. High recall signifies minimal false negatives.
- **F1-Score**: Harmonizing precision and recall, the F1-Score provides a balanced metric, particularly beneficial in unbalanced class distributions.

Figures 3.8 and 3.10 visualize EmoSynthNet's comparative accuracy advantages in training SVHN,CIFAR-10 respectively, and Figures 3.9 and 3.11 visualize EmoSynthNet's comparative accuracy advantages in validation SVHN,CIFAR-10 respectively. The simulation has been implemented with an experimental setup for SVHN mirroring CIFAR-10: 50 training epochs, batch size of 96, an initial learning rate of 0.001 adjusted lack training progress. The Cross-Entropy loss function, along with Adam optimizer, pushed EmoSynthNet to the forefront 86.32% test accuracy on CIFAR-10 and 95.75% test accuracy on SVHN, surpassing VGG19 and DenseNet121 which followed closely behind. The comparable performance of EmoSynthNet is remarkable, considering its lower parameter count, reduced memory requirements, and greater number of FLOPS, as demonstrated in Table 3.2. The upcoming strategy of loss in chapters 4 will be applied to EmoSynthNet, focusing on amplifying discriminative feature identification to elevate classification accuracy for more complex datasets on chapter 5.

Table 3. 3 Comparative analysis among existing vision networks vs the EmoSynthNet and functions on CIFAR-10 dataset. The most effective method is emphasized in bold.

			CIFAR10				
Methods	Optimizer	Loss	Accuracy	Error	Precision	Recall	F1-Score
MobileNetV2,	Adam	CE	77.23%	1.20	78.30%	76.65%	0.77
EfficientNetB0,	Adam	CE	81.26%	0.93	82.24%	80.87%	0.81
ResNet50,	Adam	CE	82.59%	1.10	82.91%	82.21%	0.82
DensNet121,	Adam	CE	85.68%	0.85	86.00%	85.34%	0.85
VGG19,	Adam	CE	86.03%	1.05	86.35%	85.78%	0.86
EmoSynthNet,	**Adam**	**CE**	**86.32%**	**0.67**	**87.03%**	**85.99%**	**0.86**

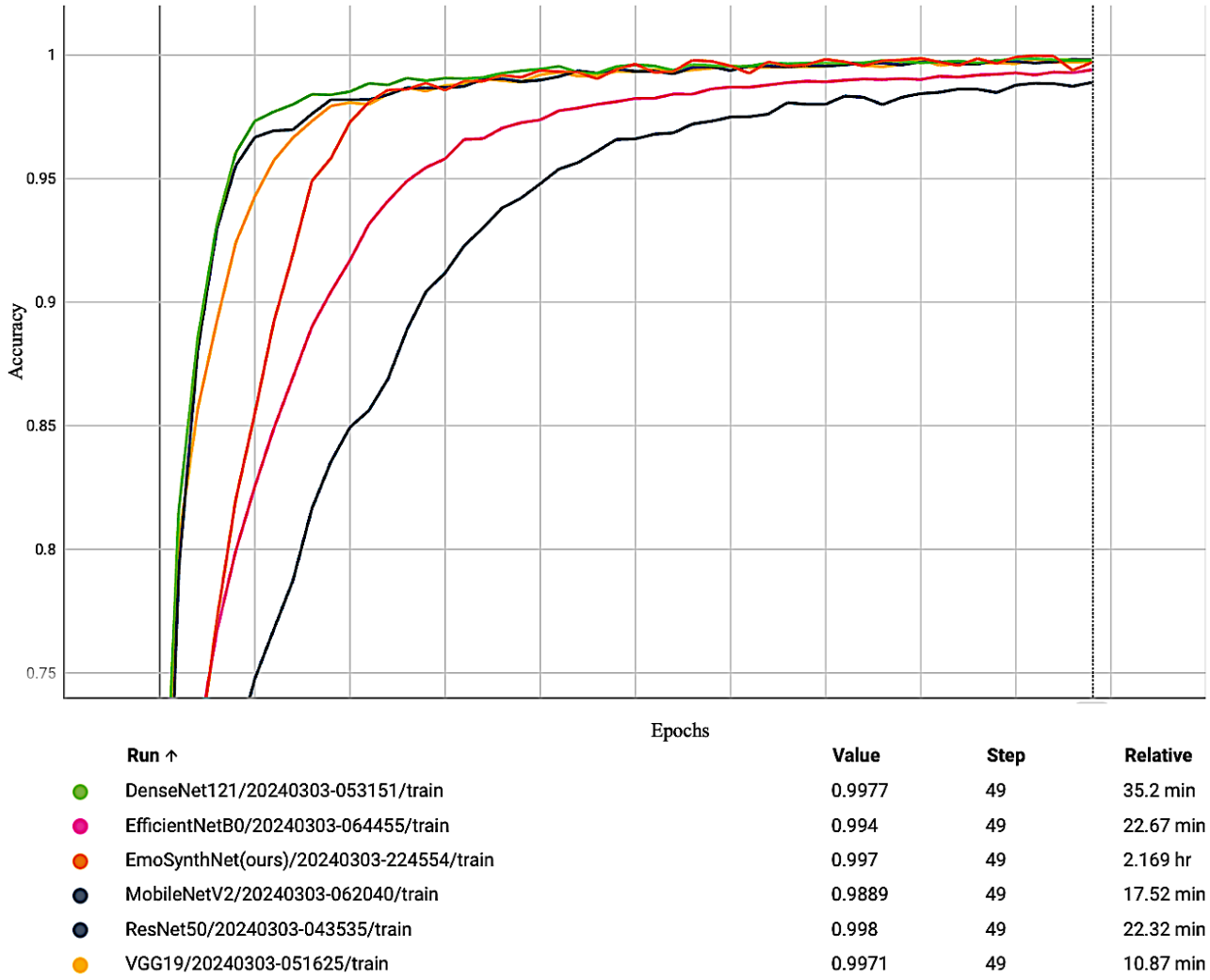
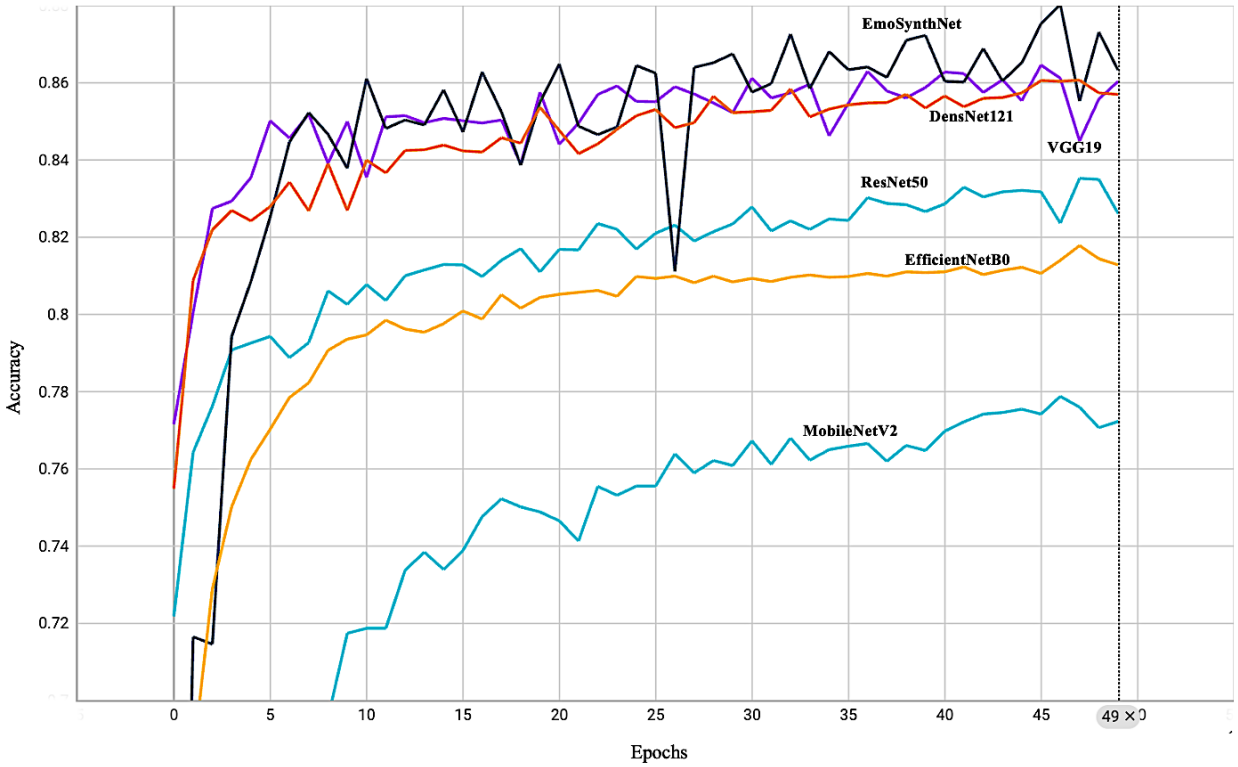


Figure 3. 8 Training accuracy comparison plots of the EmoSynthNet against existing vision networks on CIFAR10.



Run ↑	Value	Step	Relative
DenseNet121/20240303-053151/validation	0.857	49	35.2 min
EfficientNetB0/20240303-064455/validation	0.8128	49	22.67 min
EmoSynthNet(ours)/20240303-224554/validation	0.8633	49	2.169 hr
MobileNetV2/20240303-062040/validation	0.7722	49	17.52 min
ResNet50/20240303-043535/validation	0.8261	49	22.32 min
VGG19/20240303-051625/validation	0.8604	49	10.87 min

Figure 3. 9 Validation accuracy comparison plots of the EmoSynthNet against existing vision networks on CIFAR10.

Table 3. 4 Comparative analysis among existing vision networks vs the EmoSynthNet and functions on SVHN dataset. The most effective method is emphasized in bold.

			SVHN				
Methods	Optimizer	Loss	Accuracy	Error	Precision	Recall	F1-Score
MobileNetV2,	Adam	CE	90.32%	0.54	90.80%	90.11%	0.90
EfficientNetB0,	Adam	CE	91.50%	0.41	92.09%	91.20%	0.91
ResNet50,	Adam	CE	92.95%	0.46	93.21%	92.84%	0.93
VGG19,	Adam	CE	94.14%	0.55	95.18%	93.53%	0.94
DensNet121,	Adam	CE	94.68%	0.34	94.83%	94.56%	0.94
EmoSynthNet,	**Adam**	**CE**	**95.75%**	**0.33**	**96.01%**	**95.66%**	**0.95**

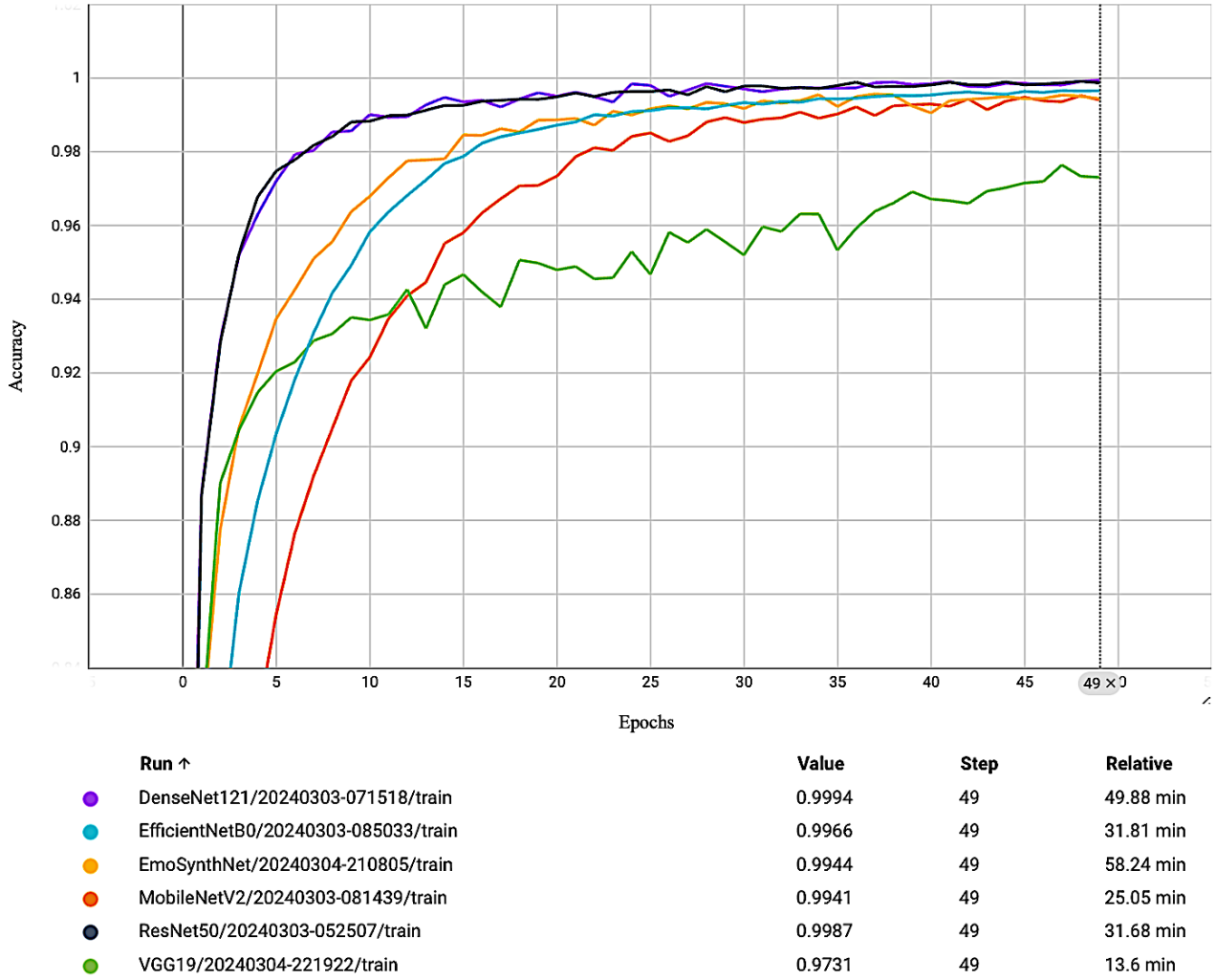
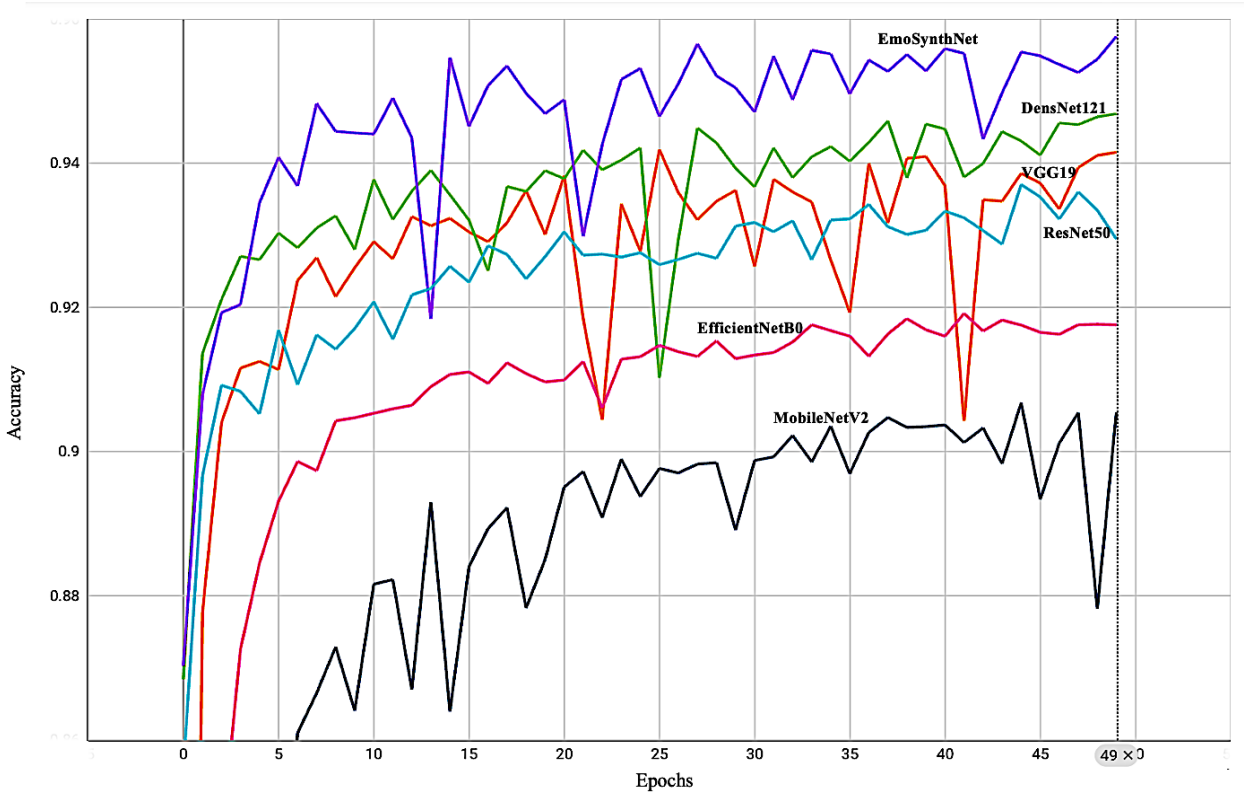


Figure 3. 10 Training accuracy comparison plots of the EmoSynthNet against existing vision networks on SVHN.



Run	Value	Step	Relative
DenseNet121/20240303-071518/validation	0.9468	49	49.88 min
EfficientNetB0/20240303-085033/validation	0.9176	49	31.81 min
EmoSynthNet/20240304-210805/validation	0.9576	49	58.24 min
MobileNetV2/20240303-081439/validation	0.9054	49	25.05 min
ResNet50/20240303-052507/validation	0.9295	49	31.68 min
VGG19/20240304-221922/validation	0.9415	49	13.6 min

Figure 3. 11 Validation accuracy comparison plots of the EmoSynthNet against existing vision networks on SVHN.

3.4 Summary

This chapter examines the proposed model's performance, highlighting its superior accuracy compared to existing vision models with Cross Entropy (CE) loss and Mean Squared Error (MSE) loss. The network architecture and training process are outlined, highlighting how a synchronized structure and precise hyperparameter tuning enhance the model's training efficiency. Results show that the proposed model performs in a comparable manner to existing vision networks due to its capability to extract distinct features effectively. Next chapter will discuss enhancing the proposed model's accuracy (EmoSynthNet) by employing the proposed SoftMax loss (MAM-SoftMax), designed to mitigate the challenge of limited sample availability in classification task. Then, chapter 5 will present the application of this framework for facial expression recognition task.

Chapter 4

SoftMax-Loss Function

4.1 Introduction

The loss function plays a vital role in adjusting weights during the backpropagation stage of the neural network's training process. It measures the difference between the expected outcome and the actual prediction, guiding weight adjustments through the calculation of gradients. Typically, this process is repeated multiple times to refine the weights, with the goal of minimization and improving accuracy. Standard loss functions can lead to underlearning of complex classes and overfitting of simpler ones. Including extra terms in cross-entropy loss can help learn difficult classes better and add noise tolerance to prevent overfitting (as outlined in Section 2.2.1.2).

Chapter 4 discusses the reasoning behind the approach to cross-entropy and SoftMax, offering a clear explanation of their roles in feature learning, and advanced methods that enhance the traditional backpropagation process, refining the network's learning strategies. Here, methods to optimize the penalty application in cross-entropy function and to strategically select hyperparameters that project samples effectively in the embedding space are presented. This method enables the networks to generalize more effectively from training data.

In this chapter, a regularization technique within the loss function, alongside a method for selecting margins in SoftMax functions are introduced. These are designed to amplify the dispersion between different classes while maintaining consistency within the same class (Section 2.2.1.3 details the use of advanced regularization algorithms in cross-entropy to create unique

features. Following this, Section 2.2.2.2 investigates the techniques for choosing margins in softmax. These techniques aim to improve the separation between different classes while keeping instances of the same class closer together. This discussion highlights the significance of these methods in enhancing the ability to distinguish between features more effectively). Then, an assessment between the performance of proposed functions against existing approaches and simulation of the geometric interpolation of embedded learned features are demonstrated. This comparative analysis aims to illustrate the effectiveness of the proposed methods in improving the learning and generalization capacity of neural networks. The aim is to secure uniform model efficiency throughout the training and testing stages, focusing on producing unique feature vectors that are distinctive. This consistency is crucial to ensure that the model not only learns effectively but also generalizes well to new, unseen data, which is a hallmark of a robust machine learning model.

4.2 Proposed Softmax-Loss Function

4.2.1 Symmetric Cross-Entropy (SCE) Loss Function

This section explains the role of loss functions in driving a model towards its highest confidence level. The goal is to reach a state of certainty in predictions, ensuring the model performs equally well during training and testing. This represents an even distribution of current knowledge and the ability to generalize (see Section 2.2.1.1).

In machine learning, entropy is a measure of a system's uncertainty, while cross-entropy assesses the amount of information present, emphasizing its robustness against irregular data [8]. This strength comes from regularization methods like Tikhonov regularization [95], which is also known as L2 norm, enhancing the model's performance. Using regularization in cross-entropy loss is beneficial for detailed classifications of classes that are rich in information, and it helps prevent the model from simply memorizing large amounts of data due to its ability to generalize well. Cross-entropy (CE) often struggles with noisy data in multi-class classification problems. New variants of CE have been introduced that apply stricter penalties to difficult samples, improving the handling of noisy data early in training. This strategy reduces the risk of overfitting and recognizes that Gaussian distribution assumptions don't always fit individual, noisy data (refer to Section 2.2.1.2).

The main advancement comes from modifying CE to respond better, affecting the size of gradient steps, and leading to quicker convergence and sharper separation of classes. The aim isn't to reach zero loss but to produce a vector whose argmax corresponds to the correct label. The suggested method in this work (Symmetric Cross-Entropy (SCE)) adds new penalty terms as hyperparameters to address confidence errors, taken from the concept of Focal loss (FL) [13], to combat underlearning. An extra function combined with CE measures the gap between wrong predictions and the truth, adjusting weight updates in the logarithmic space to balance the loss. The primary focus of the proposed loss (SCE) is enhancing its ability to set boundaries based on specific criteria to simplify and improve the optimization process. This enables the extraction of feature vectors with minimal variability. Moreover, the scale parameter is dynamically adapted based on the magnitude of errors, demanding more focus on misclassifications early on training and as training progresses computed error is diminished better. This technique is designed to amplify the loss curve, resulting in steeper gradients during optimization, which is particularly beneficial for handling data contamination and outliers. learning incorrect predictions is proposed as a weighted term to tackling difficult samples.

The cross-entropy (CE) loss for binary classification is defined as:

$$\mathrm{CE} = -[\, y_t \log(y_p) + (1 - y_t)(1 - y_p)] \qquad 4.1$$

Where y_p is the model's prediction, y_t is a ground truth. To make things easier to determine easy/hard samples in correct predictions for cross-entropy (CE), CE is written as: "$-\log y_p$ ".

To handle classes that are not equally represented, a weight factor is presented. This is a hyperparameter, alpha (α), that ranges between 0 and 1. For the more common class is set to α, ($\mathrm{CE} = -\alpha \log y_p$) and for the less common class, the weight is set to 1−α ($\mathrm{CE} = -(1 - \alpha) \log y_{1-p}$). This helps to balance the classes. While the factor α adjusts the weight given to positive and negative examples, it does not consider differentiate between easy/hard examples. FL loss addresses this by modifying the loss function, reducing the focus on simple examples, and concentrating more on difficult ones during training. It introduced a modulating term of : $(1 - y_p)^{\gamma}$ with tunable parameter gamma (γ>0) as a learnable hyperparameter to penalize the relative loss for poor classification samples (hard examples) and reduce the relative loss for well-

classified samples . Also, a scaling parameter α is introduced as a trade-off variable. FC formula for a batch of N samples is written as follows:

$$FC = -\frac{1}{N}\sum_{i=1}^{N} \alpha\,(1 - y_p)^{\gamma} \log(y_p) \tag{4.2}$$

To enhance the loss function efficiently, integrating of applying two adaptive hyperparameters is designed to maximize the distinction between different classes. This approach applies customized regularization terms to incorrect predictions, thereby helping the model to concentrate on learning from imbalanced datasets. A tunable parameter is proposed for the scaling of loss weights (referred to as alpha in FC loss). This term is exceeded in power of square for the time where the difference between prediction and correct label is negligible and prevent vanishing gradient, which calculates the discrepancy between the model's predictions and the actual ground truth across a batch of N samples. The formulation alpha is as follows:

$$\text{alpha} = \frac{1}{N}\sum_{i=1}^{N} (y_t - y_p)^2 \tag{4.3}$$

where y_t is the true label for the i-th sample, and y_p is the corresponding prediction by the model. By dynamically adjusting alpha, the model is encouraged to refine its accuracy, particularly for those samples that are more sensitive to get correctly classified. The intent behind this modification is to not only address the immediate challenge of hard samples but also to equip the learning process with a more insightful focus, thereby enhancing overall model performance. The learnable value alpha represents a critical adjustment. This dynamic element is essential in modulating the influence of loss for samples that are classified with a high degree of certainty, enabling a focus on samples that require additional learning and minimizing the impact from those that are well-identified.

Additionally, the grounded weight parameter by L2 norm , is playing a key concept for the SCE-Loss. Here, it is defined through a tensor operation by evaluating the correlation between incorrect predictions and model predictions. It measures the closeness of predicted outcomes to incorrect classes, and as confidence in predictions increases, the weight parameter decreases, reducing the

loss influence from accurately classified samples. This process ensures that the learning is directed towards enhancing the understanding of more subtle or difficult samples. Incorporating the parameters ‘alpha’ and ‘weight’ into the learning mechanism significantly improves the model's learning efficiency as shown in results section 4.3, strengthening accuracy and generalization capabilities.

$$\text{weight} = ||(1 - y_t) - y_p||_2 \tag{4.4}$$

To summarize the enhancements made, the refined SCE loss is expressed as shown below.

$$\text{SCE Loss} = \text{alpha}(\text{ weight } * \log(y_p)) \tag{4.5}$$

This formulation integrates the aforementioned learnable elements, ‘alpha’ and ‘weight’, to adjust the contribution of each sample to the overall loss. It carefully adjusts the impact based on classification certainty, thus prioritizing learning from those samples that have not yet been learned. The complete expression for the proposed loss function provides a sophisticated approach to optimizing the classification process, ensuring that the model is not only accurate but also robust against overfitting and underlearning. Therefore, the total loss with joining cross-entropy loss (LCE), is defined as:

$$\text{SCE Loss } = \propto L_{CE} + \beta L_{SCE} \tag{4.6}$$

$\propto$ and β serve as balancing parameters to fine-tune the impact of each component. In the experiments conducted for this work, these factors were set to $\propto$= 0.5, and β= 0.5. This configuration of the loss function aims to enhance the learning process, leading to better generalization and discrimination in the model's performance [49] .

4.2.2 Maximized Adaptable Margin SoftMax (MAM-SoftMax) Function

As outlined in sections 2.2.2.1, metric-based learning methods have become a standard in classification tasks where data is scarce. Multiple logistic function with codename SoftMax, first time proposed by Bridle [96] to replace squared error criterion with scoring relative entropy for classifier based on maximum likelihood (ML). Standard convolution outputs can be perceived as a Gaussian mixture distribution [97].The challenge lies in characterizing the model through density estimation (construction of an estimate) based on the maximum likelihood estimation (MLE) criterion for neural component. For binary classification, a sigmoid output suffices, but for multiple classes, the normalized exponential function better known as the SoftMax function steps in to manage the model's performance. Whenever, the aim is to represent a probability distribution over different classes, then utilizing the SoftMax function which is a generalization of the sigmoid function can be a well alternative. The name 'SoftMax' can be somewhat misleading, as it implies a connection to the argmax function, yet operates differently. Unlike argmax, which produces a non-continuous, one-hot vector that isn't differentiable, SoftMax yields a continuous and differentiable output. This distinction comes from the 'Soft' aspect of SoftMax, indicating a smoother, differentiable version of argmax. Therefore, a more descriptive name might have been 'Soft Argmax' to more accurately reflect its nature and functionality, diverging from the traditional naming convention of SoftMax [7]. Think of the SoftMax function as coordinating a race among participants, where it's crucial to maintain everyone's motivation by applying both encouragement and penalization. SoftMax (as detailed in section 2.2.2.1), exponentiates and normalizes the unnormalized log probability of linear layers outputs. and defined as:

$$\text{SoftMax}\,(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}\text{, in which } z_i = \log P(y = i|x) \tag{4.7}$$

According to logistic regression [98], exponential function in SoftMax works well in training with utilizing maximum Log-Likelihood. It’s because of the log nature to undo the impact of exp. Otherwise, the term fails in learning the arguments of exp and becomes a negative value. SoftMax tends to approach 1 when there's a large gap between the highest value and the others, indicating a high probability for an incorrect prediction due to overconfidence or it approaches 0 when the highest value isn't significantly larger, indicating vanishing gradient. To address it, the loss

function must effectively adjust the error by appropriately penalizing the impact of each participant's contribution to the total cost [7].

Conventional SoftMax probabilities, created through logarithmic and exponential calculations, are vulnerable to numerical underflow and overflow. This is mitigated by the Logsumexp operation, which utilizes the normalization property. The essence of SoftMax is about maximizing information entropy, represented in probability, to decrease the unpredictability of one output based on another variable.

To enhance feature discrimination, it's crucial to secure the distribution against noise within and between classes. Prior to this adjustment, feature vectors are revised in SoftMax through enhanced functions that exploit innovations in margin expansion. Improving compacting intra-class relations while expanding inter-class separations proposes a decision space instead of boundary decision creates a new gate of working on imbalance database for research in SoftMax. Specifically, facial expression recognition field which suffer of shortage and high noisy datasets [99].

As detailed in Section 2.2.2.2, the use of a multiplicative angular margin in softmax has shown limitations for gradient-based learning. An additive margin, combined with input normalization to control the feature vector sizes, has proven more effective. This strategy relies on adjusting the arc of an input vector within a spherical space. For classification, choosing the right margin for softmax can improve generalization by encouraging the learning of distinct features, ensuring that each class's weight vectors are tightly grouped. Properly setting boundary distances can enhance the model's resilience to noisy data. However, margins must be set with attention which could lead to an excessive condition that is beyond what is desirable performance and restrict the learning process. A larger or dynamic margin in softmax [54], [58] can lead to more compact features within a class. These considerations inspired this section to suggest an adaptive margin for soft margin softmax function, which adjusts flexibly to improve class separability and model robustness. The constant hyperparameters in margin based SoftMax functions shift values within the exponential space. This assumption led to the introduction of various margin schemes, such as cosine, angular, soft margin, additive soft margin, double additive, etc., designed to adjust outliers towards a more interpretable standard deviation range effectively centering data closer to the normal distribution's peak.

The proposed SoftMax “ Maximum Adaptive Margin SoftMax (MAM-SoftMax)” is tailored for datasets with an irregular distribution due to imbalanced datasets, aiming to provide a more

significant margin for less-represented classes for facial expression. The design of MAM-SoftMax is centered on the classification of challenging samples that float near the decision boundary. The utilized concept of MAM-SoftMax is derived to maximize the margin [63]. The margin hyperparameter is optimized in MAM-SoftMax by adjusting as a scalable learnable parameter, which depends on the class distribution's mean and variance. This hyperparameter serves as a measure for the effective number of neighboring feature vectors sharing the same class, bonded by an adhesive boundary decision parameter named 'm'. The methodology involves prior and posterior normalization to prompt underflow and overflow issues by managing the embedding space for feature vector shifts. Despite the risk of saturation in SoftMax, faster convergence is achieved by permitting the use of a learnable parameter, deriving from the inherent scale invariance of the SoftMax function (referred to section 2.2.2.2).

The conventional SoftMax can be written as follows [100]:

$$L_{SoftMax} = -\log\left(\frac{e^{w_{y_i}^T x_i}}{\sum_j^K e^{w_{y_i}^T x_i}}\right) = -\log\left(\frac{e^{\left||w_{y_i}|\right|\ ||x_i||\cos(\theta_{y_i})}}{\sum_j^K e^{\left||w_{y_i}|\right|\ ||x_i||\cos(\theta_j)}}\right) \quad 4.8$$

In which x_i denotes the feature sample, y_i is the target, w is the weight parameter of the last fully connected layer, which is named classifier, K is the total number of classes, and θ is the angle between inner product of w_j and x_i. θ can be scaled by a constraint with the purpose of encouraging SoftMax to learn more discriminative features. As detailed in section 2.2.2.2, there is a simpler definition on choosing the angular parameter in additive margin SoftMax (AM-SoftMax) [57] . The method of applying additive hyperparameter 'm' is more interpretable compared to A-SoftMax [56] and It's given by:

$$\psi(\theta) = \cos\theta - m \quad \rightarrow \quad \text{where } x = \cos\theta \Rightarrow \psi(x) = x - m = \frac{w_{y_i}^T f_i}{\left||w_{y_i}|\right|\ ||f_i||} \quad 4.9$$

This enhancement empirically proved to avoid network divergence to aim more compactness for corresponding samples [57]. The basic intuition of the above idea is that in standard SoftMax, sample x is classified correctly to class 1 for a binary classification with two classes (class1, and

class 2) when during training the assigned weight corresponding to class 1 is larger than assigned weight to input features related to class 2 [100]. This formula is given by:

$$||w_1||\,||x||\,\cos(\theta_1) \quad > \quad ||w_2||\,||x||\,\cos(\theta_2) \tag{4.10}$$

This concept can involve incorporating a margin by introducing an intermediate variable as mentioned in L-SoftMax [58], that helps to differentiates between class1 and class2 by identifying samples located near the decision boundary. This effectively. The implementation is demonstrated as follows:

$$||w_1||\,||x||\,\cos(\theta_1) \;\geq\; ||w_1||\,||x||\,\cos(\alpha\theta_1) \;>\; ||w_1||\,||x||\,\cos(\theta_2) \tag{4.11}$$

As detailed in section 2.2.2.2, α chosen as a positive integer value (named hard margin) and due to complex computation through backward and forward propagation, the formula with angels is addressed in additive form to introduce the soft margin equation by:

$$w_1^T x \;\geq\; w_1^T x - m \;>\; w_2^T x \tag{4.12}$$

Where m is a distant margin and solely needs forward computation, thus the SM-SoftMax [100] can be defined as:

$$L_i = -\log\left(\frac{e^{w_{y_i}^T x_i - m}}{e^{w_{y_i}^T x_i - m + \sum_{j\neq i}^{k} e^{w_j^T x_j}}}\right) \tag{4.13}$$

The use of a learnable parameter (detailed in section 2.2.2.2) is justified by the characteristic of SoftMax being invariant to adding a scale to its input [61]. The objective in MAM-SoftMax is to create a reliable method for tuning and learning hyperparameter 'm' as a margin distant real positive number. In cases where a network performs multi-class classification, it produces a multi-variant Gaussian distribution of results [101]. The means of the distributions match the different classes in the learned feature space and are used in margin computation. This is a key factor in achieving quicker convergence. The solution, which presented in SM-SoftMax loss, is enhanced, and introduced the margin parameter named ‘M’, allowing for choosing variable margin distances

between classes. Unlike the AM-SoftMax [57] that uses a fixed or integer-only margin, MAM-SoftMax allows 'm' to take on any real positive value, offering greater flexibility and precision in class separation. Hence, ‘M’ values are adaptable and developed based on the greatest variation observed in the data distribution of each class. By adapting ‘M’ to reflect the most significant distribution changes, the learning model is able to tune the class boundaries for improving accuracy and generalization. The proposed method for hyperparameter ‘M’ can be formulated as follows:

$$M_v \leftarrow \sum \text{ArgMax}\,(x) + \in \tag{4.14}$$

$$M_l \leftarrow \log M_v$$

$$M_c \leftarrow \text{Cliping}\ (\ 0 < M_l \leq 4)$$

$$M_U \leftarrow \text{Mean}\,(x) + M_c$$

In this case, x represents the input, M_v denotes the upper limit of the input range, and ‘$\in$’ represents for the initial value which is set to default value 0.2. M_l is the value of normalized margin through logarithmic terms, while M_c is the clipped margin value that prevents the gradient from reaching extreme highs or lows. Then, The adjustable hyperparameter, known as the margin, is tuned according to the central mean of each data sample's distribution in logarithmic space to achieve the widest range M_U [63]. Here, M_c denotes the initial margin value, and M_U represents the maximum variation applied to the mean of each distribution in a multi-class classification task Thus, for data from class 1, the margin selection formula can be expressed as:

$$W_1^T x \ \geq w_1^T x - \ m \gg w_1^T x - \ M_U \ \geq \ w_2^T x \tag{4.15}$$

Finally, the normalized MAM-SoftMax output is defined as follows:

$$\text{MAM} - \text{SoftMax} = -\log\left(\frac{e^{w_{y_i}^T x_i - M_U}}{e^{w_{y_i}^T x_i - M_U} + \sum_{j \neq i}^{k} e^{w_j^T x_j}}\right) \tag{4.16}$$

This approach incorporates both prior and posterior normalization to prevent underflow and overflow, managing the embedding space for adjusted feature vectors effectively. Normalization techniques serve a dual purpose: they not only speed up convergence but also solve the saturation problem in functions like SoftMax, which happens with very high or low values. Evaluations in next section on CIFAR-10, MNIST, and RAF-DB show MAM-SoftMax loss boosts accuracy and speed up convergence over traditional methods by pulling similar data points together and pushing different ones further apart. In the following, Figure 4.1 demonstrates algorithm of proposed loss and SoftMax functions.

SCE Loss Algorithm

Input (y_{true}, y_{pred})

$weight \leftarrow ||(1 - y_{true}) - y_{pred}||_2$
(contribution of wrong prediction in scaling as a regularization term)

$alpha \leftarrow \frac{1}{N} \sum_{i=1}^{N} (y_{true} - y_{pred})^2$

$y_p \leftarrow -\sum log_{10} y_p$: (log normalization)

$L_{proposed} \leftarrow < alpha, (< weight, y_p >) >$
(<> *denotes multiplication*)

Return $L_{proposed}$

$Loss_{SCE} = \alpha L_{cce} + \beta L_{Proposed}$
(α& β default values are 0.5)

Output $Loss_{SCE}$

MAM-SoftMax algorithm

Input (x)

$m \leftarrow \sum_i \text{argmax}(x_i) + \text{Initialized value}$ (defult ≈ 0.2)

Clipping $m \leftarrow 0 > m \geq 4$

$\exp m \leftarrow \sum \text{Mean}(\exp x_i) + \log_{10} m$

$\exp(x_i - m) \leftarrow \exp x * \exp(-m)$

$S_{MAM} \approx$ Proposed SoftMax $\leftarrow -\log\left(\frac{\exp(x_i - m)}{\sum_{j \neq i} \exp x_j + \exp(x_i - m)}\right)$

Return S_{MAM}

Figure 4. 1 Algorithms of SCE-Loss function (Up), and MAM-SoftMax (Down)

4.3 Experimental Results

Table 4.1 describes the performance of the EmoSynthNet with 3 loss functions (MSE, CCE, and the MAM-SoftMax Loss). From the table, it's seen, boosting accuracy by using MAM-SoftMax and achieving a 3%~4% improvement over the original SoftMax loss and MSE. On CIFAR10, MAM-SoftMax has achieved notable improvement with 93.63% accuracy for EmoSynthNet model proposed in Chapter 3 (EmoSynthNet). In Table 4.1, the bold-faced values in red font indicate the results from the best-performing methods, and those in blue and green fonts indicate the results from the second-best and third-best performing methods. From the results, the following observations resulted; the EmoSynthNet (supervised by MAM-SoftMax loss) outperforms the conventional function, by improving the best accuracy performance of CE loss on the validation set (93.63% by MAM-SoftMax loss vs. 91.54% by CE loss on the C10 dataset). This result demonstrates that MAM-SoftMax loss can promote the discriminative power of deep-learned features. Second, compared to MSE loss, the MAM-SoftMax loss achieves better performance (93.63% by MAM-SoftMax loss vs. 89.69% by MSE loss on the C10 dataset). The implemented idea in the proposed MAM-SoftMax for choosing an adaptable margin with the maximum possible value is taken from hinge loss in choosing the maximum decision boundary for SVM classifier [102], and escalating wrong penalization idea of the loss functions is taken from focal loss, which reduces well-classified example loss values.

Table 4.1 Improvement in accuracy of the EmoSynthNet using the MAM-SoftMax loss function. The bold-faced values in red font indicate the results from the best-performing methods, and those in blue and green fonts indicate the results from the second-best and third-best performing methods, respectively.

Methods	Loss	Optimizer	MNIST	CIFAR10
EmoSynthNet	RMSE/MSE	Adam	99.63	89.69
EmoSynthNet	SoftMax+ CE Loss	Adam	99.61	91.54
EmoSynthNet	MAM-SoftMax Loss	Adam	**99.66**	**93.63**

Figures 4.2 illustrates a side-by-side comparison of the proposed SoftMax Loss (MAM-SoftMax and SCE Loss) with conventional SoftMax Loss functions, experimented on CIFAR-10 dataset. It is clearly seen that maximizing the margin in softmax reduces the impact of samples close to the decision boundary. Additionally, using regularization in cross-entropy (CE) loss leads to more

distinct embeddings. Therefore, the MAM-SoftMax loss enhances the traditional SoftMax and CE Loss by providing greater computational efficiency and more stable training. The result in Figure 4.2 points out that MAM-SoftMax loss surpasses state-of-the-art SoftMax functions by improving feature discriminability, compressing intra-class variations, and expanding inter-class separation for more effective classification, and contributing to smoother loss curvature resulting higher accuracy. The following section continues with more assessments of MAM-SoftMax Loss's performance. Figure 4.3 presents accuracy and loss metrics from the network's performance proposed in [103] on MNIST dataset. These figures validate performance of MAM-SoftMax using SCE-Loss function against the conventional SoftMax and CE Loss functions. The result in Figure 4.3, in particular, demonstrate the proposed functions improves performance of image classification in terms of accuracy through creating a smoother loss curve, which simplifies reaching better convergence in the optimization process. The model evaluated here, referenced from "Striving for Simplicity" by J.T Springenberg et al. [103], was trained using the Adam optimizer with a learning rate of 0.001 over 30 epochs.

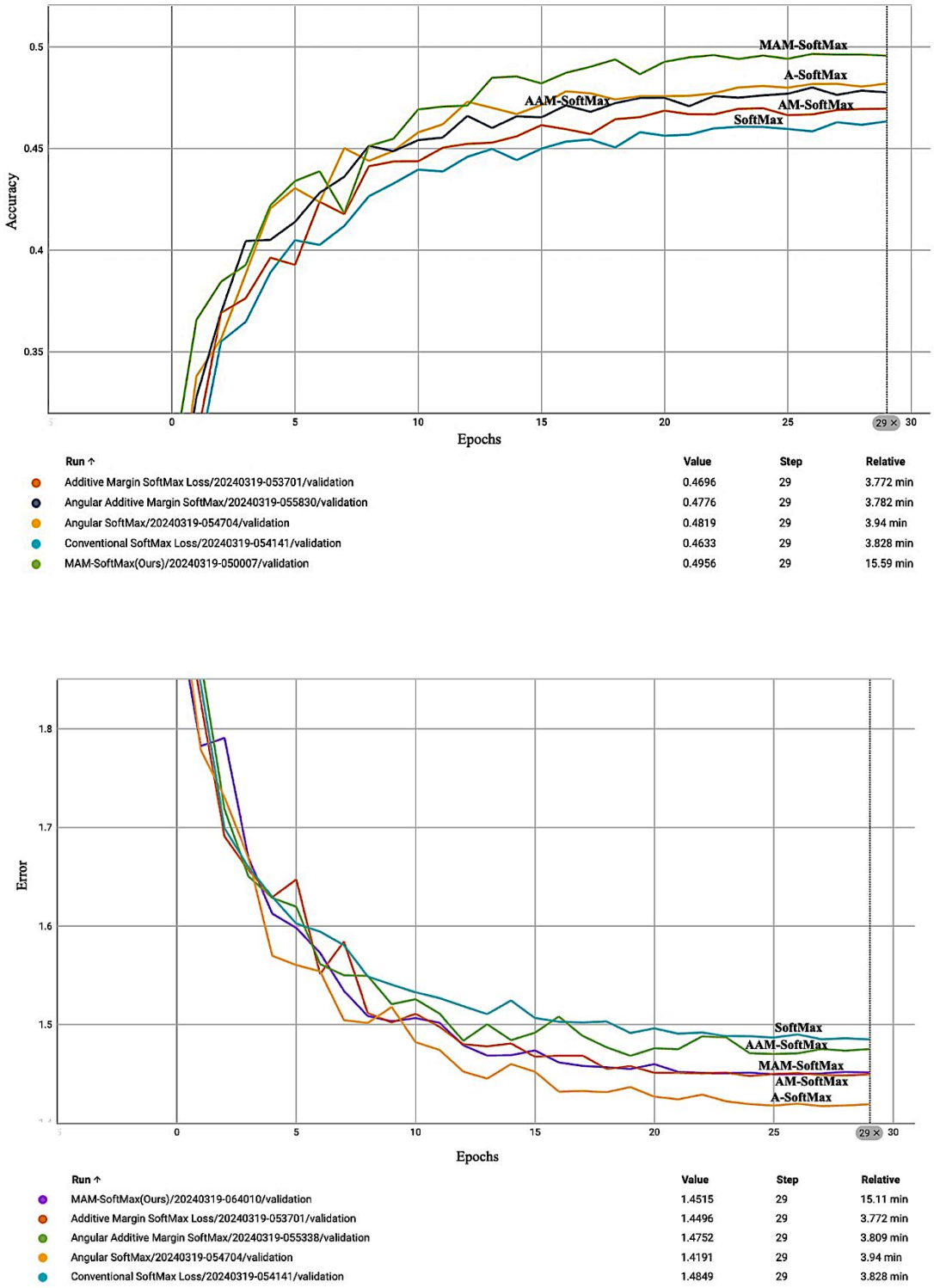
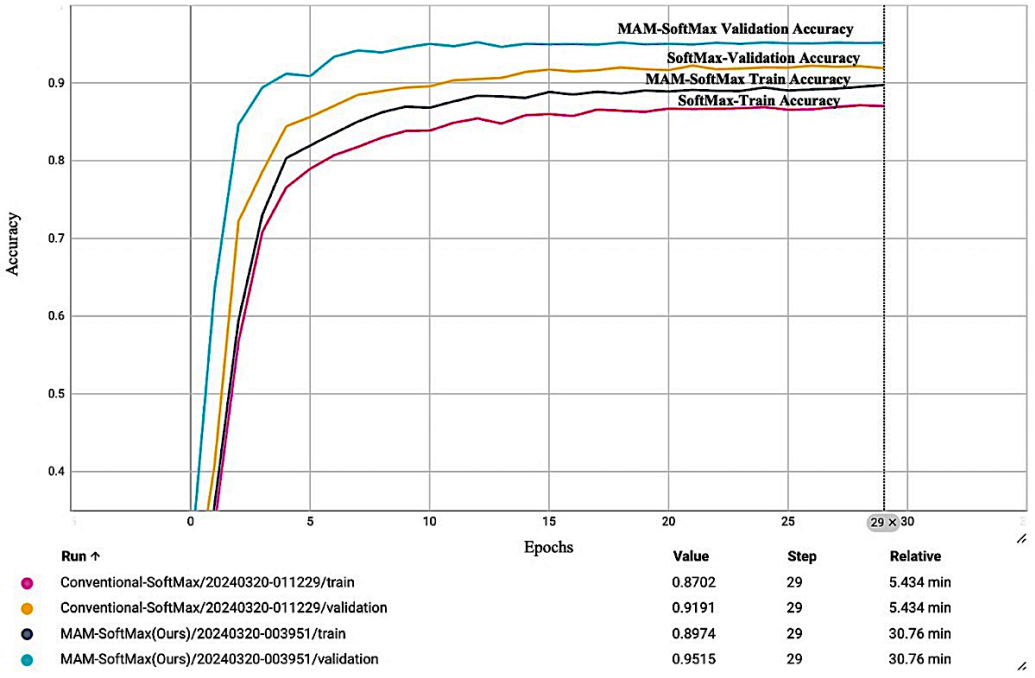
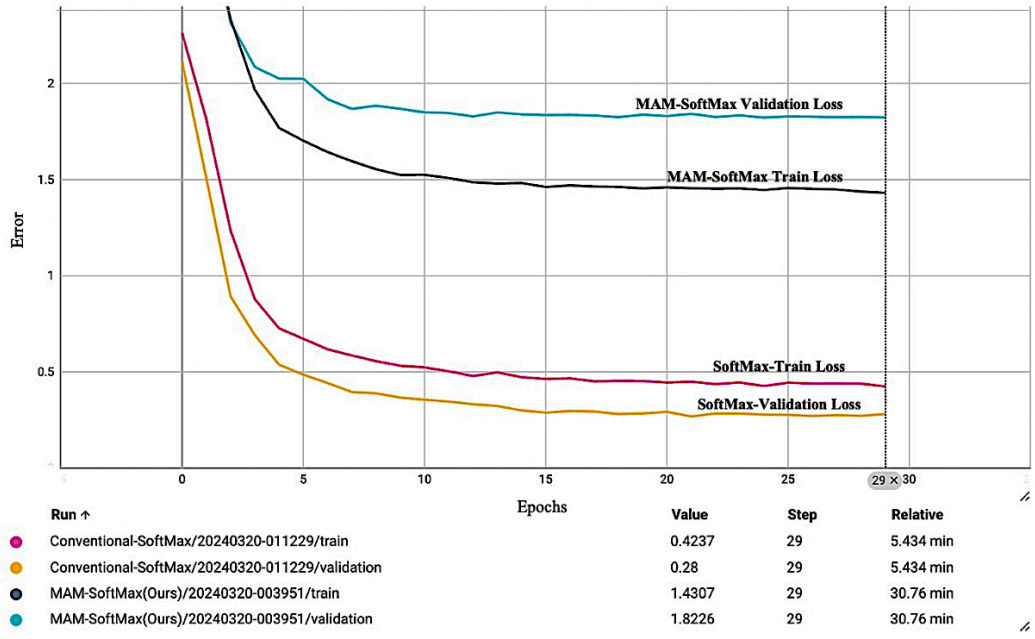


Figure 4. 2 Accuracy (Top) and Loss (Down) Analysis : Proposed SoftMax Algorithm (MAM-SoftMax) Versus 3 existing SoftMax Functions.



Run	Value	Step	Relative
Conventional-SoftMax/20240320-011229/train	0.8702	29	5.434 min
Conventional-SoftMax/20240320-011229/validation	0.9191	29	5.434 min
MAM-SoftMax(Ours)/20240320-003951/train	0.8974	29	30.76 min
MAM-SoftMax(Ours)/20240320-003951/validation	0.9515	29	30.76 min



Run	Value	Step	Relative
Conventional-SoftMax/20240320-011229/train	0.4237	29	5.434 min
Conventional-SoftMax/20240320-011229/validation	0.28	29	5.434 min
MAM-SoftMax(Ours)/20240320-003951/train	1.4307	29	30.76 min
MAM-SoftMax(Ours)/20240320-003951/validation	1.8226	29	30.76 min

Figure 4. 3 Comparative Evaluation: accuracy (Up), loss (Down) plots using MAM-SoftMax and Conventional SoftMax-Loss on MNIST dataset.

The study of how loss landscapes affect generalization is carried out using various visualization techniques with filter normalization, as shown in "Visualizing the Loss Landscape of Neural Nets"[9]. This research examines how networks with shortcut connections lead to flatter loss landscapes and investigates the influence of training parameters on the characteristics of minimizers.

The 1D linear interpolation approach to investigate the paths taken by different minimization methods, as implemented by Goodfellow et al. [104], has several limitations. Primarily, it struggles to effectively represent the complexities of non-convex shapes through one-dimensional plots. Alternatively, one can utilize Contour Plots and Random Directions. This technique requires selecting a central point θ* on the graph and two directional vectors, δ and η. For the one-dimensional case, function is plotted as $f(\alpha) = L(\theta^\star + \alpha\delta)$ while for the two-dimensional scenario, the plot is:

$$f(\alpha, \beta) = L(\theta^\star + \alpha\delta + \beta\eta) \qquad 4.21$$

This approach [12] has been employed in Figure 4.4 to demonstrate how distinct optimization algorithms discover various local minima within the two-dimensional projected area. It is important to note that each plot's center aligns with a minimizer and the two axes characterize two random directions.

Figures 4.4 illustrates the contour of the loss functions shaped by SCE Loss function following a process of dimensionality reduction for simplicity. The character of the loss landscape is a key factor in model generalization broad. Smooth valleys in this landscape often lead to lower generalization errors. Conversely, rough, and steep areas can result in higher errors and slow down learning. The SCE Loss results in a landscape with fewer sharp valleys and disordered regions, promoting more consistent gradient information and preventing overfitting during optimization.

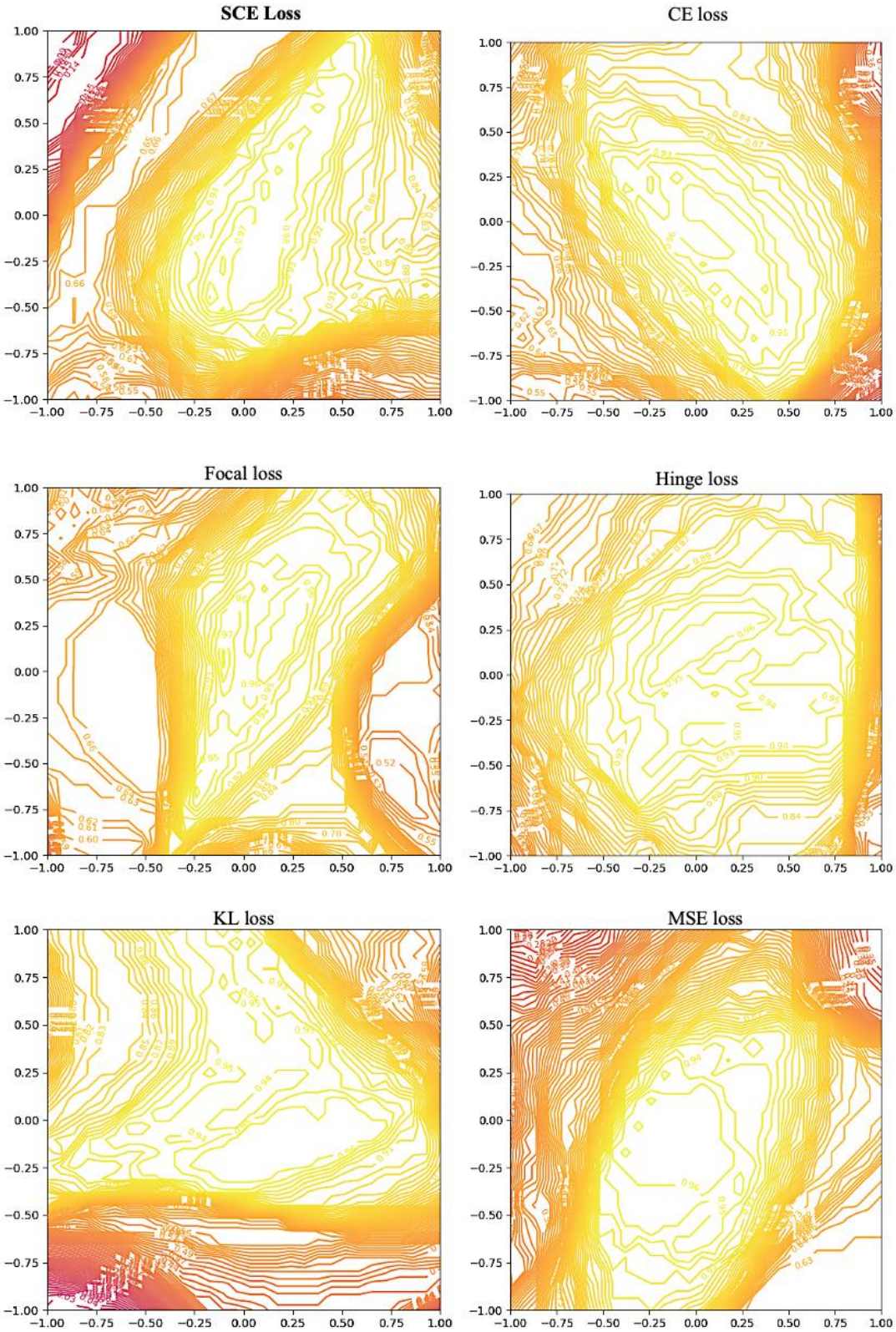


Figure 4. 4 Loss contour visualization: comparative 2d representation of conventional loss functions and MAM-SoftMax Loss.

The low-dimensional representation may give the impression of convexity, but this doesn't necessarily hold in higher dimensions. Instead, this observation suggests that the positive curvatures are common; in more technical terms, the mean curvature, or the average of the eigenvalues, is positive. These calculations are performed in Figure 4.5 by using an implicitly Lanczos method, which relies only on Hessian-vector products (HVPs) computed via automatic differentiation, without needing the Hessian's full representation or its factorization [9] . The process involves determining the smallest and largest eigenvalues of the Hessian, λ_{min} and λ_{max}. The ratio $\left|{\lambda_{min}}/{\lambda_{max}}\right|$ across the loss landscapes, with the same reference point and random directions. Figure 4.5 highlights the differences in the loss geometry between the SCE Loss function and the conventional ones in which blue areas denote regions that are more convex, while yellow signifies areas with considerable negative curvature. Convex areas in the plot of the SCE Loss actually correlate with zones having negligible negative eigenvalues, meaning no substantial non-convex characteristics are missed. Conversely, some regions near other minimizers exhibit clear negative curvatures. Large negative curvatures in disordered zones could be areas of concern in the optimization landscape. The findings emphasize SCE Loss function can improve performance of classification task, demonstrating its ability to better navigate the feature space for enhanced class distinction.

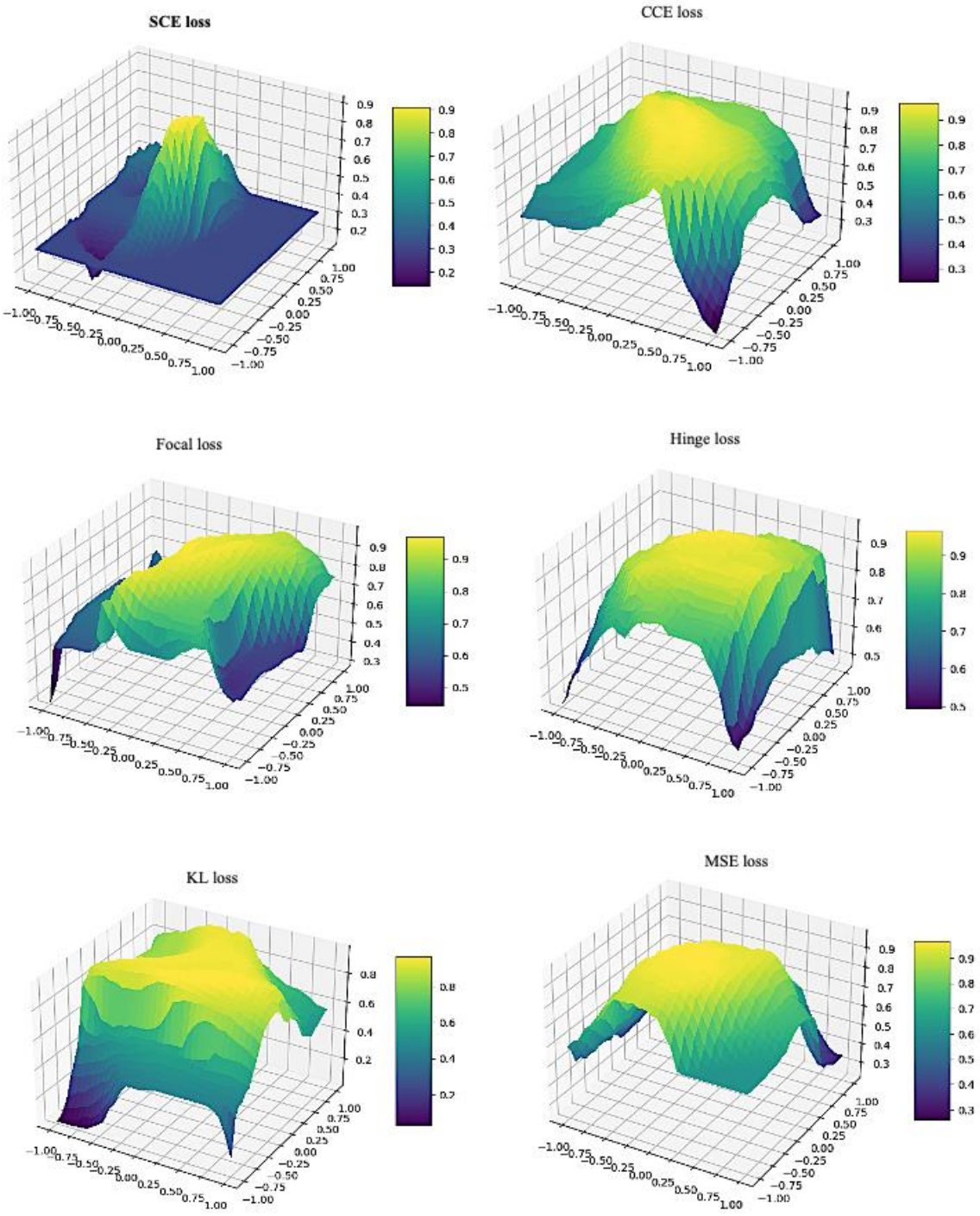


Figure 4. 5 Comparative Loss Surfaces: Proposed SoftMax Loss vs. Conventional Loss Functions.

Automatic dimensionality reduction serves as a crucial tool in machine learning, simplifying both pre-processing and the visualization of complex, high-dimensional data. It involves projecting complex vectors into a more manageable, lower-dimensional space. A key technique in this process is t-Distributed Stochastic Neighbor Embedding (t-SNE) [15], a probabilistic method renowned for its efficacy in visualizing high-dimensional data through a dimensionality-reduction lens. This approach is optimized through the Barnes-Hut approximation, allowing for efficient computation. t-SNE establishes an embedding concept rooted in probable neighbors, working to maintain the integrity of local neighborhoods. It centers each data point within a Gaussian distribution in the high-dimensional space and leverages the resulting pairwise dissimilarities to infer a probability distribution that captures the essence of neighboring relationships. The general goal of t-SNE embedding is to closely approximate this probabilistic landscape, a task at which t-SNE and its derivatives succeed, even when applied to vast datasets comprising millions of instances. The innovation of MAM-SoftMax Loss lies in its ability to expand additive angular margins, which translates into an adaptable distinction between the features of different classes, surpassing the conventional SoftMax's performance. As demonstrated in Figure 4.6, the proposed SoftMax encourages a tighter, more integrated clustering of features within the same class while ensuring a distinct separation from features of other classes. Take, for instance, the distinctive separation achieved among the dog, cat, deer, and horse classes, each learning unique features that anchor them in distinct regions of the embedding space. This contrasts with the standard SoftMax, where classes may converge too closely due to inadequate feature discrimination. The result includes an obvious boost in classification accuracy on CIFAR10 with the MAM-SoftMax function, highlighting its enhanced ability to discriminate learned features.

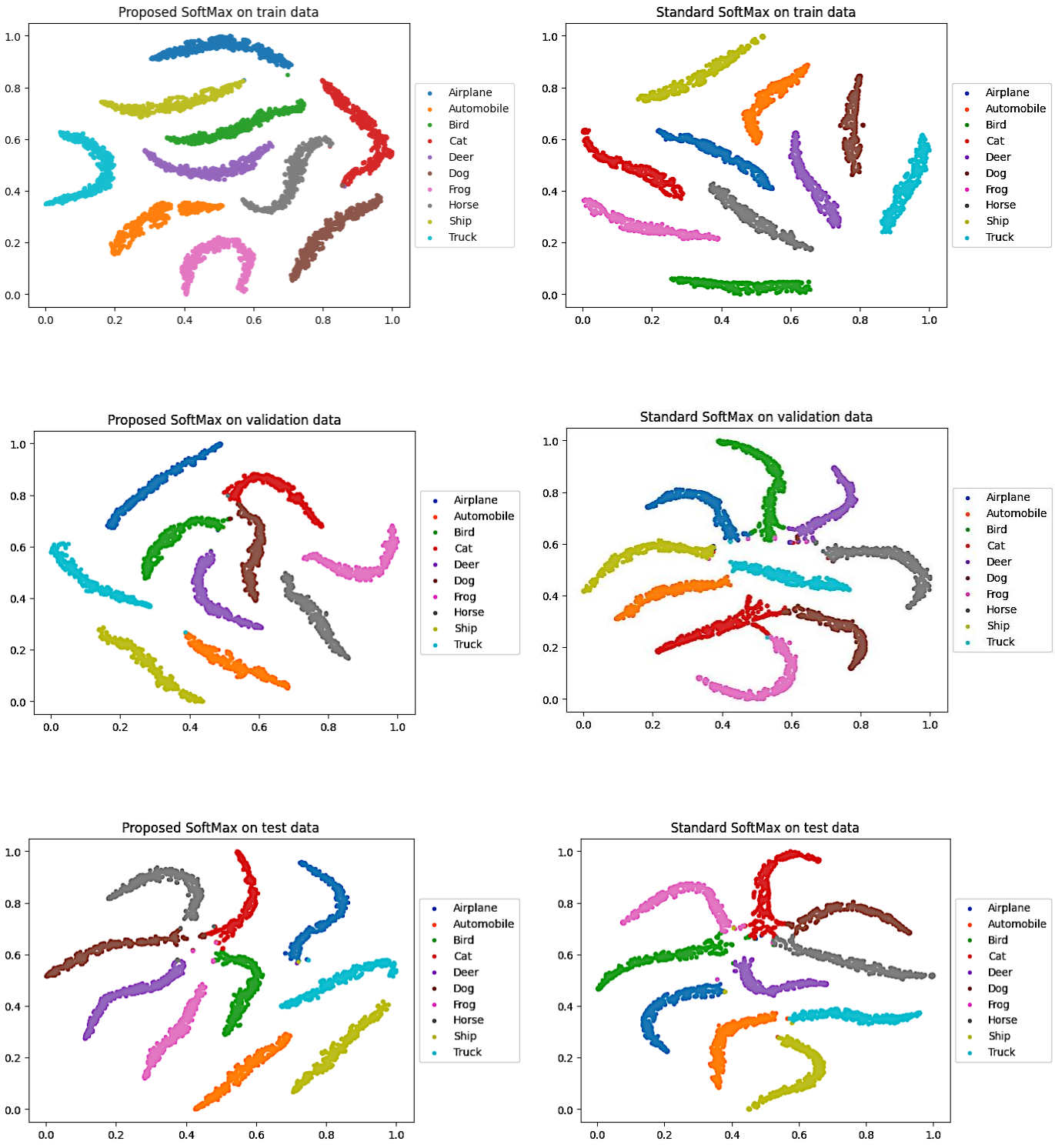


Figure 4. 6 Feature Space Visualization (t-SNE) on CIFAR10: Comparing Standard (Right Column), and Proposed SoftMax (Left Column).

4.4 Summary

In this chapter, an improved SoftMax Loss was introduced, featuring an adaptable margin controlled by a learnable hyperparameter for an optimal margin maximization. The enhanced SoftMax Loss function was evaluated across diverse datasets, showcasing better performance when compared to traditional functions. Notably, the expanded margin in MAM-SoftMax has a geometrically design, contributing to its clear interpretability. Furthermore, the findings suggest that SCE-Loss function enhanced with regularization technique can be effective for better convergence and more distinct features. The results confirmed that the proposed approach can beat existing algorithms in image classification task. The next chapter will deploy and evaluate the application of proposed framework for facial expression recognition against state-of-the-art models, covering seven distinct basic emotion categories consist of surprise, fear, disgust, happiness, sadness, anger, and neutral.

Chapter 5

Application of EmoSynthNet to Facial Expression Recognition

5.1 Introduction

In recent years, facial expression recognition has become a key area of research in computer vision, due to the goal of equipping computers with the ability to understand human emotions [105].Facial Expression Learning (FEL) is a prominent field in human-computer interaction, with applications in healthcare, education, smart robotics, and more [106]. The number of large-scale benchmark databases for Facial Expression Learning (FEL) has grown in recent years, leading to significant improvements in the recognition accuracy of Convolutional Neural Network (CNN) methods [107], [108], [109]. Despite impressive progress, FEL remains challenging due to a few reasons: (1) Limited ability of convolutional networks to capture the global context of input images. (2) Similarity between classes, where different expressions may look alike. (3) Variation within classes, as images of the same expression may differ due to factors like background, gender, or age. (4) Sensitivity to scale, as images taken in natural settings vary in size and resolution, which can affect the performance of deep learning networks [110].

Correctly identifying an emotion from a static facial image is still a big challenge. This is because some emotions look similar, and FER datasets often have uneven and inaccurate emotion labels [111].The main goal of facial expression recognition (FER) when looking at a single facial image is to identify basic emotions such as happiness, sadness, and neutrality [72] [112]. The primary aim of Facial Expression Recognition (FER) systems is to detect distinctive features among a scarcity of labeled samples while minimizing the overfitting.

Many of the best current methods start by training deep learning models on clearer datasets to get a good understanding of general facial features. Then, they adjust these models for specific FER tasks [113]. Beyond Convolutional Neural Networks (CNNs), other models like Graph Neural Networks (GNNs) and some Hybrid Vision Transformers (Hybrid ViT's) have also been exploited for FER, showing prominent results. A Hybrid ViT combines a pre-trained CNN with multiple Transformer blocks [114], [115]

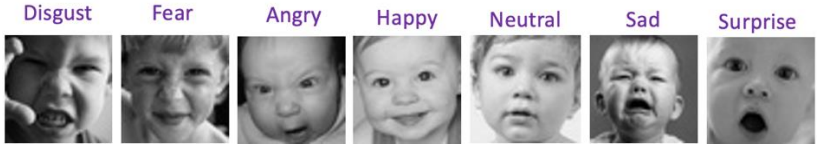
Deep Metric Learning (DML) methods play a vital role in automated FER by promoting feature distinction between classes and consistency within classes. FER models try to develop informative features that increase differences between classes and shorten variations within the same class. Then, the use of auxiliary loss functions as regularization, or main label smoothing techniques, can reduce the gap between the model's output prediction and the true class values, thereby mitigating within-class variations [74], [116], [117], [118].

In this chapter, the efficiency of the EmoSynthNet with advanced SoftMax Loss for FER task is showcased. The model is adeptly designed to detect emotions through a Deep Convolutional Neural Network (DCNN) framework, inputting of static face emotion images and aiming for fine-grained classification despite limited clean data. The comparison between state-of-the-art deep learning methods and the proposed framework on three widely used facial expression datasets shown in tables 5.1, 5.2, 5.3 demonstrates the superior performance of EmoSynthNet in term of accuracy. The behavior of the model in terms of loss, accuracy, and confusion matrix during training and validation is shown in Figures 5.2, 5.3, 5.5, and 5.6, respectively.

5.2 Experimental Results

Facial Expression Recognition (FER) on FER2013:

The FER2013 [119] dataset includes both lab-controlled and in-the-wild RGB facial images. The images are labeled with 7 basic expressions: 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, and 6=Neutral. The Disgust label has the fewest images, about 600, while the other labels have nearly 5,000 images each.



The FER-2013 dataset is a collection of 35,887 facial expression images. These images, gathered by Pierre Luc Carrier and Aaron Courville through a Google image search, were published for the ICML 2013 challenge under the same name. The dataset includes a mix of laboratory-controlled and naturalistic 'in-the-wild' images, depicting six basic emotional states plus a neutral expression. They have been standardized to 48x48 pixel grayscale images. However, the dataset is known to have a labeling accuracy of about 68% ±5%, as indicated by Ian J. Goodfellow, which points to a significant mislabeling challenge.

Comprising 28709 train and 7178 test images across seven emotion categories [120] (‘Angry’, ‘Disgust’, ‘Fear’, ‘Happy’, ‘Sad’, ‘Surprise’, Neutral’). For FER-2013, most methods incorporate fine-tuning of transfer learning models or combine convolutional models with handcrafted features, often using classic classifiers like SVM or KNN to address the labeling misclassification [120], [121], [122].

As shown in the comparative Table 5.1, the accuracy results from most competitors in the FER-2013 challenge are still modest. Based on results observation, the effectiveness of purely CNN-based models on this dataset appears to remain limited. The proposed model, EmoSynthNet, clearly performs better than all existing SOTA network architectures on the FER-2013 dataset in terms of accuracy. The method achieved top-1 accuracy on the FER-2013 dataset, surpassing the state-of-the-art results (∼17%) compared to the second work “Ensemble ResmaskingNet with 6 other CNNs by Pham et al.”, with a recorded test-accuracy of 93.79%. This improvement

highlights the strength of the EmoSynthNet model, enhanced by the proposed loss functions, in extracting highly informative features from a limited amount of data.

Table 5. 1 FER-2013 test set accuracy comparison

	Model	Accuracy (%)
1	**EmoSynthNet,**	**93.79**
2	PAtt-Lite [123],	92.50
3	Ensemble ResmaskingNet with 6 other CNNs [119],	76.82
4	EmoNetXt [124],	76.12
5	Segmentation VGG-19 [125],	75.97
6	Local Learning Deep+BOW [69],	75.42
7	LHC-Net [121],	74.42
8	LHC-NetC [121],	74.28
9	Residual Masking Network [119],	74.14
10	ResNet18 With Tricks [126],	73.70
11	VGGNet [73],	73.28
12	CNNs and BOWN + global SVM [122],	73.25
13	ResNet50 [127],	73.20
14	SE-Net50 [127],	72.50
15	CNN Hyperparameter Optimisation [78],	72.16
16	Ad-Corre [74],	72.03

Figure 5.1 offers a visual comparison of classification outcomes using standard and proposed SoftMax-Loss functions. It is a comparison of the performance between MAM-SoftMax and the standard SoftMax loss demonstrates the efficacy of the proposed approach in improving inter-class separation and intra-class compactness, as shown through the geometric interpolation on the hypersphere manifold, the standard softmax shows the difficulty of detection for some samples belong to certain classes (class 0 "angry", class 1 "disgust", class 2 "fear") where placed in center

of embedding space because there are limited samples and their emotion patterns look similar, making it tough to classify them apart. However, the proposed SoftMax-Loss overcomes these problems by using margin and regularization techniques.

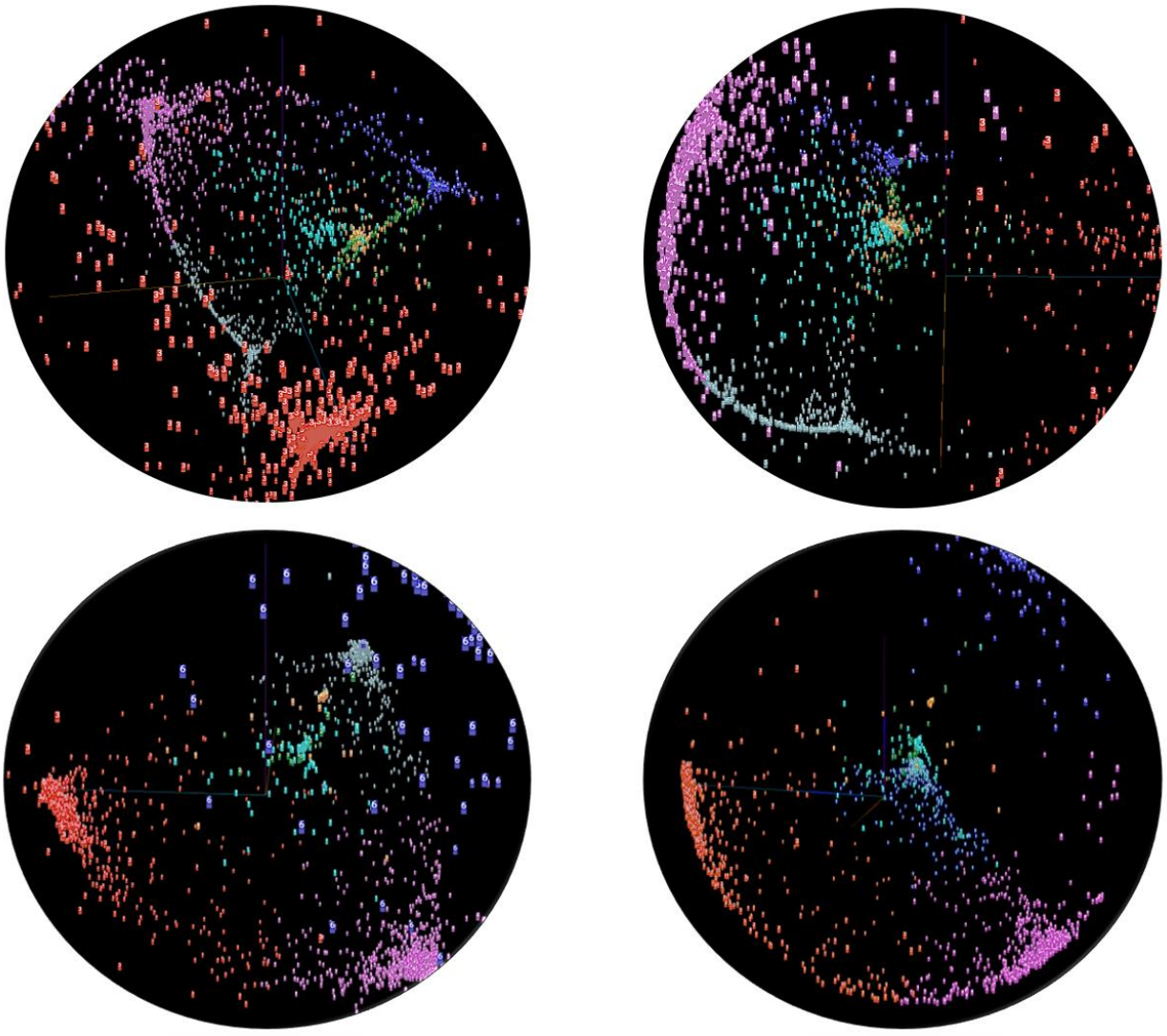


Figure 5. 1 Feature space visualization of classified test sample of FER-2013 dataset: Conventional SoftMax Loss (Top), MAM-SoftMax Loss (Bottom).

Additionally Figure 5.2, highlights the loss, and accuracy metrics observed during testing and confusion matrix. The confusion matrix (Figure 5.3) for FER-2013 shows that classes “0” and “1”,

which have similar looks, are more likely to be confused. So, for the FER-2013 dataset, there's a clear pattern where the "Angry" and "Disgust" classes get mixed up often. For example, 11% of the time, samples that are actually "Disgust" are predicted as "Angry." This shows it's hard to tell the difference between similar emotions, especially when there aren't many examples.

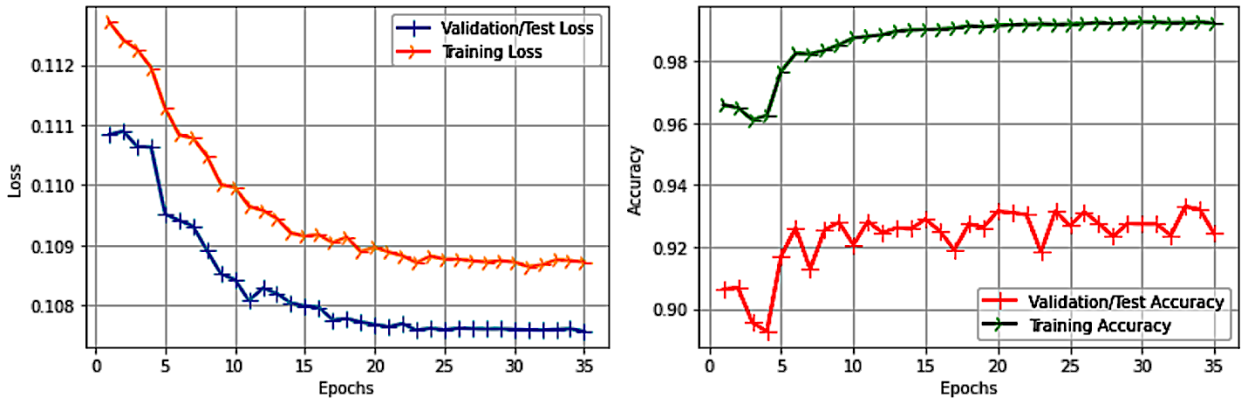


Figure 5. 2 Metric visualization of accuracy, and loss on FER-2013 test Dataset

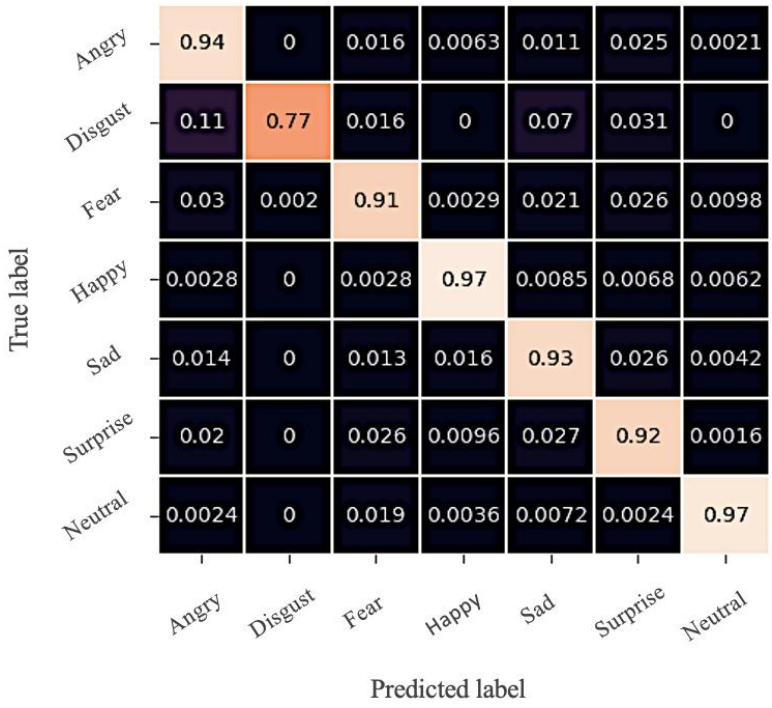


Figure 5. 3 Confusion matrix for the EmoSynthNet evaluated on the FER-2013 test dataset.

Facial Expression Recognition (FER) on RAF-DB :

The Pyramid with Super Resolution for in-the-wild Facial Expression Recognition paper [128] discussed the RAF-DB, a comprehensive facial expression dataset with nearly 30.000 images. This experiment exploits the basic emotion subset of the database including 12,271 training samples and 3,068 test samples [123], varying in color, size, and consists of both aligned and original photos. The dataset is broken down into a primary set of seven basic emotions [129] of: 'Surprise', 'Fear', 'Disgust', 'Happiness', 'Sadness', 'Anger', 'Neutral', and a secondary set with 12 compound expressions, annotated by expert human coders.

Due to its imbalanced nature, it introduces a Prior Distribution Label Smoothing (PDLS) loss function, which uses prior class label knowledge to group similar emotions together while distinguishing between different ones. This approach led to achieving outstanding validation accuracy on the RAF-DB. Preprocessing techniques such as random cropping and grayscale conversion have been applied. This preprocessing was aimed at reducing computational demands and improving model output efficiency.

The table 5.2 compares the accuracy of the EmoSynthNet model with other models on the RAF-DB dataset. EmoSynthNet shows the best performance on this benchmark. It did better than the DDAMFN [1] as the second-best performance by 0.59%, making EmoSynthNet the top method for FER tasks.

The visualization in Figure 5.4 illustrates the classification effectiveness using both standard and proposed SoftMax-Loss, showing clear differentiation in mitigating variation within the classes.

Also, Figure 5.4 shows that the MAM-SoftMax loss function, unlike the standard softmax loss, greatly improves EmoSynthNet's ability to classify rare samples close to decision boundaries that are usually hard for the classifier to identify.

Figures 5.5 shows the accuracy and loss measurements, along with a normalized confusion matrix. These figures prove the ability of the EmoSynthNet as a facial expression recognition method to work well with new data. The confusion matrix Figures 5.5 and visualization of Figure 5.4 show how the proposed loss function (MAM-SoftMax) puts in place harder penalties to get better at telling emotions apart by special attention to the 'Fear' class , which is often mixed up with 'Anger'. The matrix shows that 16% of 'Fear' samples are confused with 'Sad' class, since poor distinctive features , and 18% of 'Disgust' samples are mistaken for 'Neutral' due to not enough training data to learn distinctive features for 'Disgust'.

Table 5. 2 RAF-DB test set accuracy comparison

	Model	**Accuracy (%)**
1	**EmoSynthNet**	**91.94**
2	DDAMFN [130],	91.35
3	ViT_base + MAE [131],	91.07
4	TransFER [132],	90.91
5	EAC [72],	90.35
6	DAN [129],	89.70
7	RUL (ResNet-18) [70],	88.98
8	PSR (VGG-16) [128],	88.98
9	MViT [133],	88.62
10	EfficientFace [87],	88.36
11	MA-Net [72],	88.36
12	DACL (ResNet-18) [116],	87.78
13	MixAugment [134],	87.54
14	SCN [135],	87.03
15	RAN [136],	86.90
16	gACNN [137]	85.07

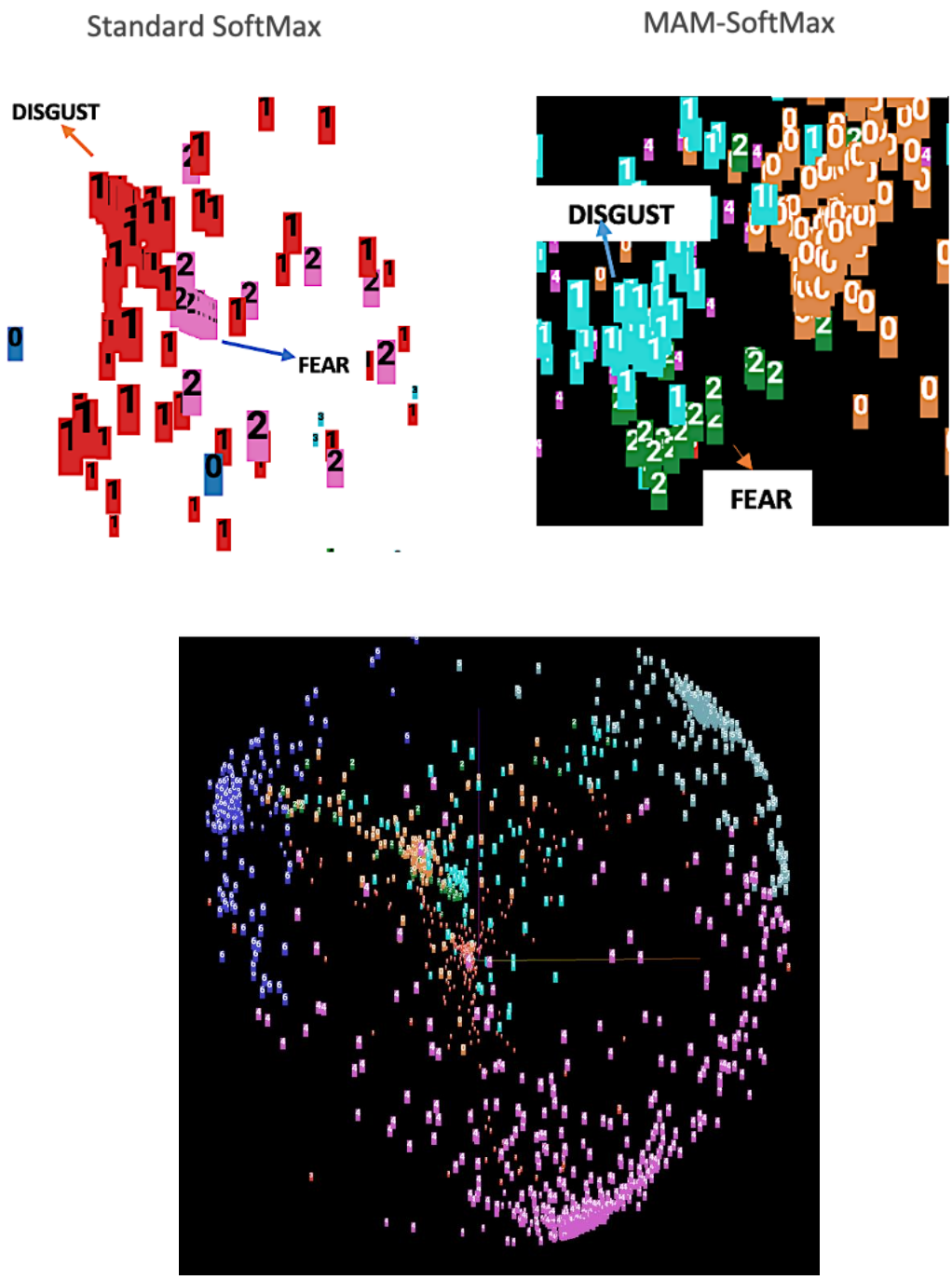


Figure 5. 4 Feature space visualization of how hard samples are classified on the RAF-DB dataset by Standard SoftMax and MAM-SoftMax (Top), and feature space visualization of test samples RAF-DB dataset classified using MAM-SoftMax (Bottom).

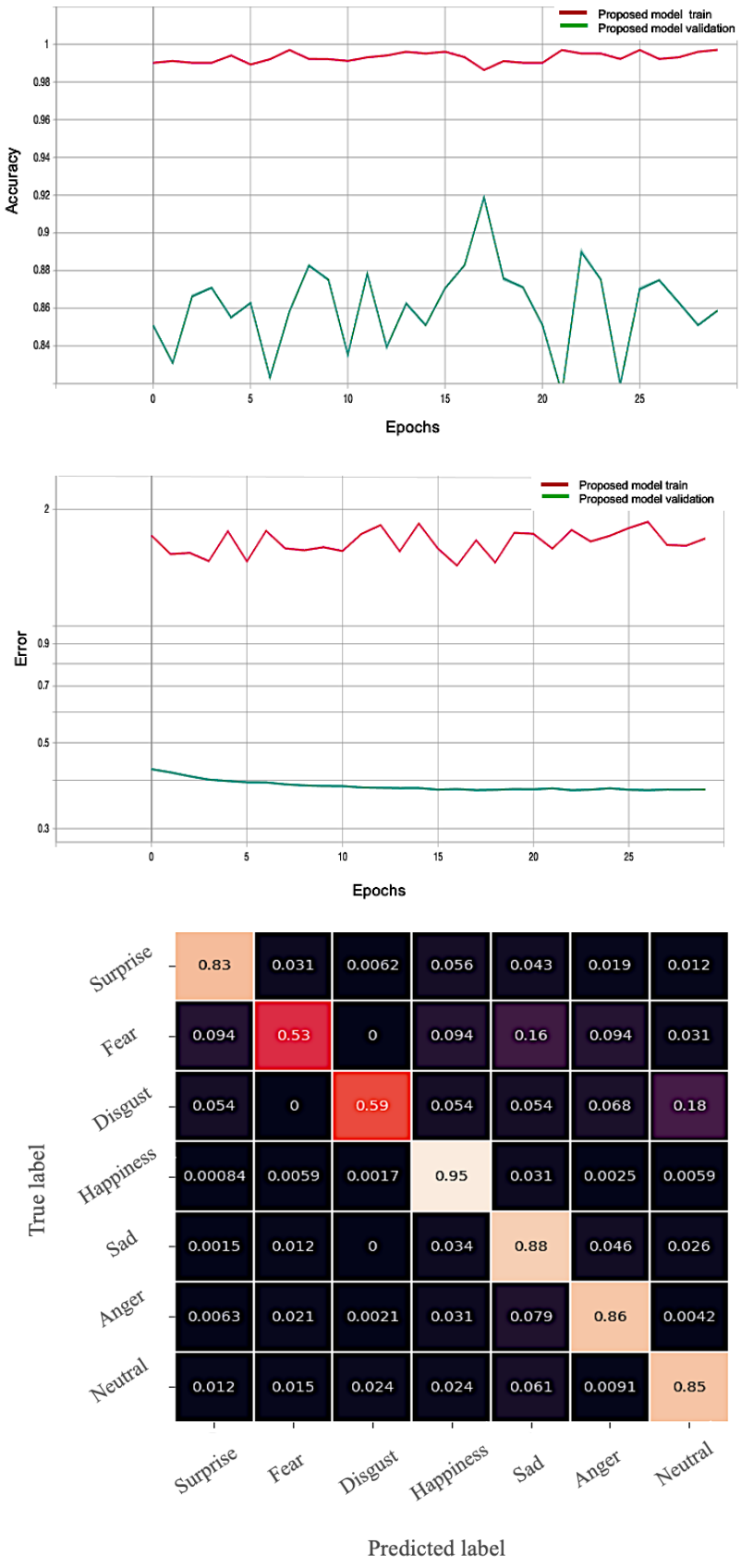


Figure 5. 5 Metric visualization of accuracy, and loss for RAF-DB Dataset (Top, and Middle respectively).

Facial Expression Recognition (FER) on CK+:

The Cohn-Kanade dataset, also known as CK+ [107], includes a collection of 593 image sequences of emotional expressions from 123 subjects, detailed in a resolution of 640×490 and 640×480 pixels [138]. Each sequence transitions from a neutral state to the peak of an expressed emotion. However, only 327 sequences have annotations for one of the 7 basic emotions ('Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', Neutral') [120].

Classes of 'Contempt' and 'Neutral' get merged [123], reducing the number of basic emotion categories to seven. For the purposes of this thesis, the last two frames from each annotated sequence were selected to enhance the training and testing datasets, with the initial frame representing a neutral expression [111].

As part of the preprocessing, the originally colored images were converted to grayscale and resized to a manageable dimension of 48×48 pixels, simplifying computational demands. Previous studies employing the CK+ dataset have often fine-tunned pre-trained models under supervision to mitigate data scarcity [139]. This experiment leverages the learned weights from the FER-2013 and RAF-DB facial expression databases to improve the quality of learned features. EmoSynthNet using MAM-SoftMax compresses the labels of similar classes and amplifies discriminative features, enhancing the model's ability to distinguish subtle nuances among uncertainty.

The methodology deployed has achieved a new benchmark as shown in Table 5.3, attaining highest accuracy of 100% using a 10-fold cross-validation protocol [138], surpassing previous SOTA results. Figure 5.6 shows the accuracy, loss, and a confusion matrix, comparing how the standard and the new SoftMax-Loss classify data. Figure 5.7 presents a geometric interpolation

that displays the effectiveness of classification with MAM-SoftMax, emphasizing EmoSynthNet model's performance improve.

Table 5. 3 CK+ test set accuracy comparison

	Model	Accuracy (%)
1	**EmoSynthNet**	**100.00**
2	PAtt-Lite [123],	100.00
3	ViT +SE [138],	99.8
4	FAN [75],	99.7
5	FDRL [106],	99.54
6	FN2EN [75],	98.6
7	ST network [140],	98.50
8	Nonlinear eval on SL+SSL Puzzling (B0) [139],	98.23
9	DeepEmotion [141],	98
10	IF-GAN [142],	97.52
11	SCAN-CCI [142],	97.31
12	PPDN [143],	97.3
13	ST-RNN [141],	97.2
14	DTAGN [144],	97.2
15	pACNN [145],	97.03
16	gACNN [145],	96.40

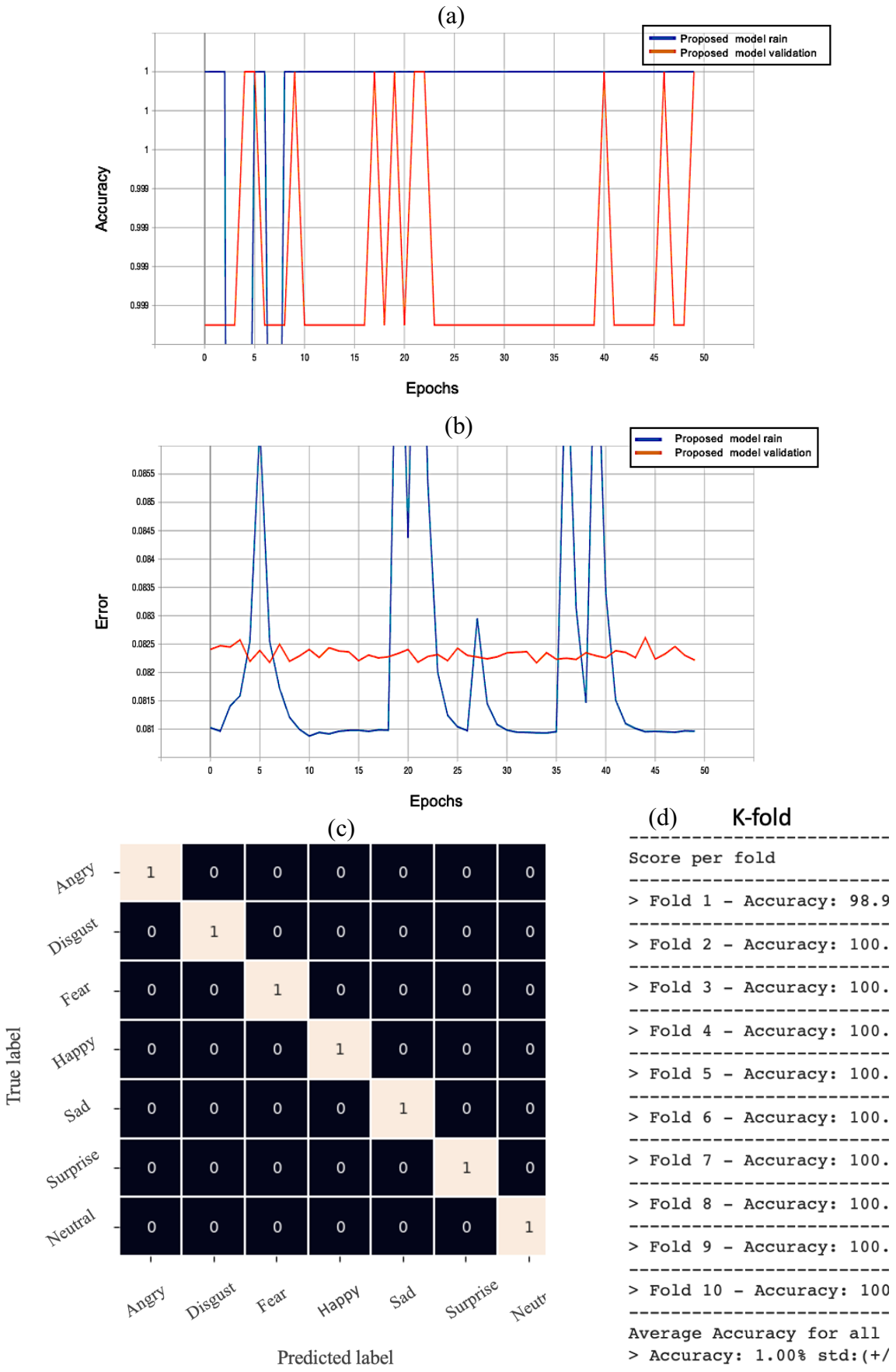


Figure 5. 6 Metric visualization of accuracy, and loss for CK+ Dataset (a), and (b) respectively, Confusion matrix for the EmoSynthNet evaluated on test dataset (c), and Confusion Matrix Analysis (d).

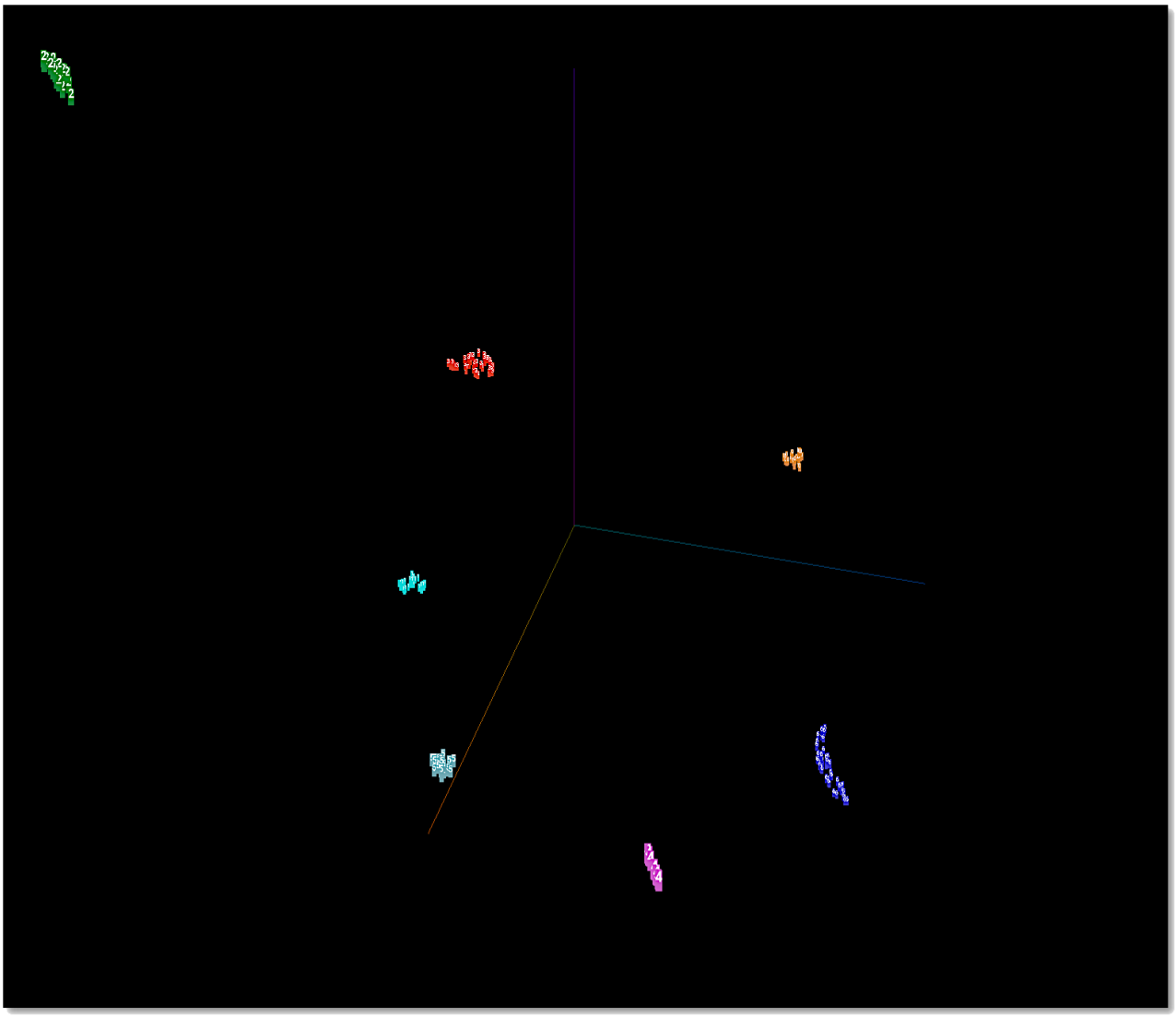


Figure 5. 7 Feature space visualization of how test samples are classified on the CK+ dataset using MAM-SoftMax.

As detailed in section 3.3.2, Gradient-Weighted Class Activation Mapping (Grad-CAM) is a visualization tool applied to various CNN architectures, aiding in identifying key regions for concept prediction within images. Figure 5.8 illustrates the application of Grad-CAM on the final layer of EmoSynthNet, which highlights important parts for the classification decision. Although Grad-CAM effectively indicates significant image regions, it doesn't capture fine details as precisely as pixel-level techniques like Guided Backpropagation, which filters out negative gradients. The distinction between various emotional states as interpreted by the network remains somewhat non-transparent when relying only on heatmaps. By integrating Grad-CAM with

Guided Backpropagation through element-wise multiplication, a more refined visualization called Guided Grad-CAM is produced. This method sharpens the visualization, identifying specific facial features like eyes, nose, and mouth that are crucial for emotion classification. Guided Grad-CAM, developed by Selvaraj et al [85], combines the concepts of Gradient backpropagation [103], and Grad-CAM [85]. This hybrid technique showcases the most powerful class-specific units for classification decisions, providing a detailed explanation of how CNNs detect facial emotions [125], [138] , essentially 'learning' through a vocabulary of discriminative units similar to words, each corresponding to class-specific elements.

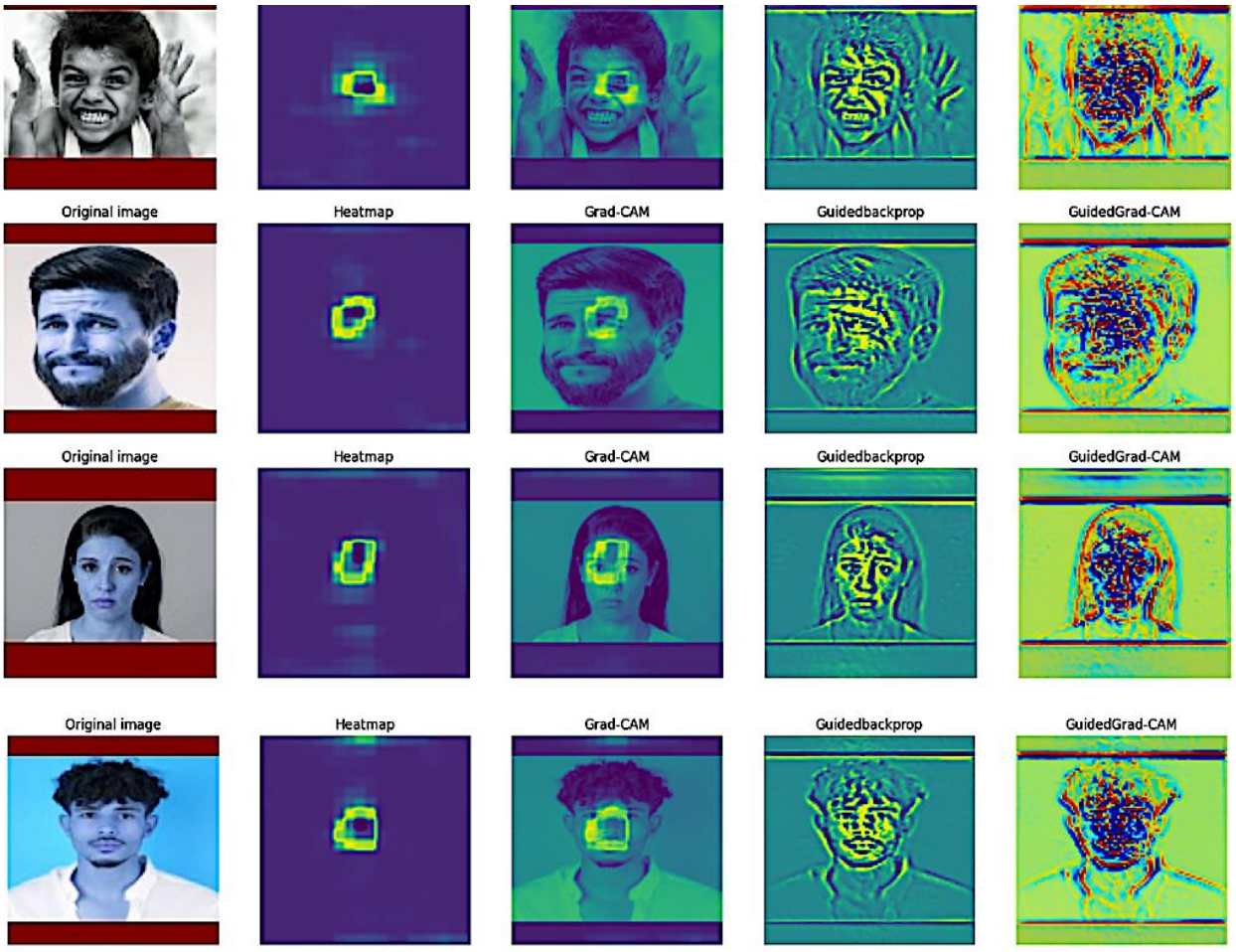


Figure 5. 8 Facial Emotion Visualization with Grad-CAM: Coarse Localization and Enhanced Detailing via Guided Grad-CAM Methodology.

5.3 Summary

The final chapter presents the validation of the developed image classification model (EmoSynthNet) for FER application across three distinct datasets: FER-2013, RAF-DB, and CK+. These datasets incorporate a diverse range of facial images, both captured in controlled environments and taken from the wild, presenting various age groups and demographics in both color and grayscale formats. This chapter begins with a detailed description of the datasets utilized and the preprocessing techniques employed to prepare the data for the experimental trials. Following this, the study compares the performance of the introduced model against state-of-the-art FER methodologies, utilizing standard evaluation metrics to assess image classification efficacy. The findings indicate that the EmoSynthNet, equipped with the newly designed SoftMax-Loss and is able to surpasses existing deep learning methods in facial expression learning (FEL).

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis introduces a model inspired by U-Net, Inception, and VGG architectures, designed to process input data effectively by focusing on meaningful features. It leverages an analysis path to generate a latent space and a synthesis path to improve data handling efficiency and localization precision. Enhanced by an auxiliary classifier, the model adeptly processes images across multiple scale feature maps, aiming to select a subset of beneficial feature vector elements. This selection process introduced in chapter 3 and experimented in image classification datasets is pivotal for enhancing discriminability and learning complex features.

Another part of this these delves into refining the traditional backpropagation process and learning strategies of the network by introducing a regularization technique to modify the Cross-Entropy function and implement an adaptive Soft-Margin SoftMax function. This modification is intended to maximize the separation of dissimilar feature representations while bringing similar ones closer together, addressing the limitations of standard SoftMax Loss function. This function typically struggles with underlearning complex classes and overfitting simpler ones. Also, by incorporating additional terms into the cross-entropy loss, the aim was to improve learning for difficult classes and introduce noise tolerance in SoftMax Loss to mitigate risk of overfitting. This approach improves training stability, ensuring consistent model performance throughout training and testing phases.

The ultimate goal of this thesis is to encourage a learning embedding space where distinct features are emphasized, facilitating the generalization of complex features. By fine-tuning the proposed method for facial expression learning (FEL),a balanced model capable of performing

robustly across various image classification, face recognition was achieved. This model represents a promising approach in developing deep learning methods that can adapt to a broad spectrum of data efficiently.

6.2 Future Work

This thesis explored how to handle limited training samples and complex feature learning to reduce risk of overfitting. In the future, the method will be tested as a multi-task learning approach with more detailed facial benchmarks and in areas like head pose estimation to improve generalization. Tuning the weights of attentive feature extractors, like the Vision Transformer (ViT), whether as a main component or an add-on, is crucial for results. Finding the right regularization and optimization techniques is also a challenge, as is adapting loss functions for more stable training in tough situations with noisy data. This is particularly true when data is scarce. One promising solution could be creating a new loss function that dynamically assigns weights [146]. This method would help with learning different quantities that have varied scales and units in classification and regression.

Future work includes selecting the best architecture for data classification in different fields such as healthcare, audio, and textures. Since various tasks affect the model differently, choosing the right architecture is key and could lead to different outcomes [147]. To address the issue of data scarcity, some types of CNN-based Generative Adversarial Networks (GANs) will be explored to create new facial expressions. The urgent need for more research and development is emphasized to enhance the accuracy and reliability of automated facial expression recognition in real-time situations, even with limited resources.

Conferences

[1] A.Nabaei, M.O. Ahmad and M.N.S. Swamy, “Enhanced Multi-Scale Feature Extraction using a Deep Convolutional Neural Network for Face Emotion Detection”, presented at the 6th Annual Graduate Students Research (GSR)., Montreal, Canada, Mar. 2022, [Online]. Available: https://www.iccsit.org/2022.html.

[2] A.Nabaei, M.O. Ahmad and M.N.S. Swamy, ‘‘Enhanced Multi-Scale Feature Extraction Using a Deep Convolutional Neural Network for Face Emotion Detection,’’ presented at the 15th International Conference on Computer Science and Information Technology (ICCSIT)., Tokyo, Japan, Oct. 2022, [Online]. Available: https://www.iccsit.org/2022.html.

[3] A.Nabaei, William E. Lynch (Admin Supervisor), “Optimizing Facial Expression Recognition: A Novel Multi-Scale Model Employing Enhanced Soft-Margin SoftMax Loss and Adam Optimizer Modifications”, presented at the IVADO Digital Futures., Montreal, Canada, Mar. 2024, [Online].Available: https://ivado.ca/en/events/ivado-digital-futures-2024/.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Ha, "Gradient-Based Learning Applied to Document Recognition," p. 46, 1998.

[2] "Generalized cross entropy loss for training deep neural networks with noisy labels | Proceedings of the 32nd International Conference on Neural Information Processing Systems." Accessed: Apr. 23, 2024. [Online]. Available: https://dl.acm.org/doi/10.5555/3327546.3327555

[3] "U-Net and Its Variants for Medical Image Segmentation: A Review of Theory and Applications | IEEE Journals & Magazine | IEEE Xplore." Accessed: Dec. 19, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9446143

[4] "(4) (PDF) A Multigrid Tutorial, 2nd Edition." Accessed: Dec. 19, 2022. [Online]. Available: https://www.researchgate.net/publication/220690328_A_Multigrid_Tutorial_2nd_Edition

[5] R. Szeliski, "Fast surface interpolation using hierarchical basis functions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 6, pp. 513–528, Jun. 1990, doi: 10.1109/34.56188.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 234–241. doi: 10.1007/978-3-319-24574-4_28.

[7] "Deep Learning." Accessed: Jan. 18, 2023. [Online]. Available: https://www.deeplearningbook.org/

[8] A. Dubey, O. Gupta, R. Raskar, and N. Naik, "Maximum-Entropy Fine Grained Classification," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2018. Accessed: Mar. 19, 2024. [Online]. Available: https://papers.nips.cc/paper_files/paper/2018/hash/0c74b7f78409a4022a2c4c5a5ca3ee19-Abstract.html

[9] "Visualizing the loss landscape of neural nets | Proceedings of the 32nd International Conference on Neural Information Processing Systems." Accessed: Mar. 20, 2024. [Online]. Available: https://dl.acm.org/doi/10.5555/3327345.3327535

[10] “The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Jan. 18, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/661502

[11] “Angular Margin Softmax Loss and Its Variants for Double Compressed AMR Audio Detection | Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security.” Accessed: Feb. 01, 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3437880.3460414

[12] “[PDF] An Empirical Analysis of Deep Network Loss Surfaces | Semantic Scholar.” Accessed: Apr. 23, 2024. [Online]. Available: https://www.semanticscholar.org/paper/An-Empirical-Analysis-of-Deep-Network-Loss-Surfaces-Im-Tao/815ae2909fd7fc536afa9773228dab40872d5cb7

[13] “Focal Loss for Dense Object Detection | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Apr. 23, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/8417976

[14] D. Zhou *et al.*, “Dynamic Margin Softmax Loss for Speaker Verification,” in *Interspeech 2020*, ISCA, Oct. 2020, pp. 3800–3804. doi: 10.21437/Interspeech.2020-1106.

[15] G. Hinton and S. Roweis, “Stochastic Neighbor Embedding”.

[16] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, *A Guide to Convolutional Neural Networks for Computer Vision*. in Synthesis Lectures on Computer Vision. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-031-01821-3.

[17] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *CoRR*, vol. abs/1207.0580, 2012, Accessed: Apr. 23, 2024.

[18] J. Donahue *et al.*, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition,” in *Proceedings of the 31st International Conference on Machine Learning*, PMLR, Jan. 2014, pp. 647–655. Accessed: Mar. 05, 2024. [Online]. Available: https://proceedings. Generalized Cross Entropy Loss for Training Deep Neural mlr.press/v32/donahue14.html

[19] “Visualizing and Understanding Convolutional Networks | SpringerLink.” Accessed: Apr. 23, 2024. [Online]. Available: https://link-springer-com.lib-ezproxy.concordia.ca/chapter/10.1007/978-3-319-10590-1_53

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014.

[21] J. Kukacka, V. Golkov, and D. Cremers, “Regularization for Deep Learning: A Taxonomy,” *CoRR*, vol. abs/1710.10686, 2017, Accessed: Apr. 16, 2024.

[22] “Training with Noise is Equivalent to Tikhonov Regularization | MIT Press Journals & Magazine | IEEE Xplore.” Accessed: Apr. 16, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6796505

[23] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” presented at the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Jul. 2017, pp. 2261–2269. doi: 10.1109/CVPR.2017.243.

[24] “Deep Residual Learning for Image Recognition | IEEE Conference Publication | IEEE Xplore.” Accessed: Mar. 05, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/7780459

[25] C. Szegedy *et al.*, “Going deeper with convolutions,” presented at the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Jun. 2015, pp. 1–9. doi: 10.1109/CVPR.2015.7298594.

[26] “[PDF] Network In Network | Semantic Scholar.” Accessed: Apr. 23, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Network-In-Network-Lin-Chen/5e83ab70d0cbc003471e87ec306d27d9c80ecb16

[27] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, “Deeply-Supervised Nets,” in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, PMLR, Feb. 2015, pp. 562–570. Accessed: Apr. 23, 2024. [Online]. Available: https://proceedings.mlr.press/v38/lee15a.html

[28] “Inception-v4, inception-ResNet and the impact of residual connections on learning | Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence.” Accessed: Apr. 23, 2024. [Online]. Available: https://dl.acm.org/doi/10.5555/3298023.3298188

[29] “Pattern Recognition and Neural Networks.” Accessed: Dec. 19, 2022. [Online]. Available: https://www.cambridge.org/core/books/pattern-recognition-and-neural-networks/4E038249C9BAA06C8F4EE6F044D09C5C

[30] “(PDF) Highway Networks.” Accessed: Apr. 23, 2024. [Online]. Available: https://www.researchgate.net/publication/275897262_Highway_Networks

[31] “Training very deep networks | Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2.” Accessed: Apr. 23, 2024. [Online]. Available: https://dl.acm.org/doi/10.5555/2969442.2969505

[32] “Deep Residual Learning for Image Recognition | IEEE Conference Publication | IEEE Xplore.” Accessed: Mar. 05, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/7780459

[33] “[PDF] Attention U-Net: Learning Where to Look for the Pancreas | Semantic Scholar.” Accessed: Apr. 23, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Attention-U-Net%3A-Learning-Where-to-Look-for-the-Oktay-Schlemper/ae1c89817a3a239e5344293138bdd80293983460

[34] N. S. Punn and S. Agarwal, “Inception U-Net Architecture for Semantic Segmentation to Identify Nuclei in Microscopy Cell Images,” *ACM Trans. Multimed. Comput. Commun. Appl.*, vol. 16, no. 1, p. 12:1-12:15, Feb. 2020, doi: 10.1145/3376922.

[35] H. Zhang and J. Ma, “Hartley Spectral Pooling for Deep Learning.” Oct. 08, 2020. doi: 10.4208/csiam-am.2020.

[36] Z. Zhang, Q. Liu, and Y. Wang, “Road Extraction by Deep Residual U-Net,” *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 5, pp. 749–753, May 2018, doi: 10.1109/LGRS.2018.2802944.

[37] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, “UNet++: A Nested U-Net Architecture for Medical Image Segmentation,” in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, D. Stoyanov, Z. Taylor, G. Carneiro, T. Syeda-Mahmood, A. Martel, L. Maier-Hein, J. M. R. S. Tavares, A. Bradley, J. P. Papa, V. Belagiannis, J. C. Nascimento, Z. Lu, S. Conjeti, M. Moradi, H. Greenspan, and A. Madabhushi, Eds., Cham: Springer International Publishing, 2018, pp. 3–11. doi: 10.1007/978-3-030-00889-5_1.

[38] “[PDF] OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks | Semantic Scholar.” Accessed: Apr. 23, 2024. [Online]. Available: https://www.semanticscholar.org/paper/OverFeat%3A-Integrated-Recognition%2C-Localization-and-Sermanet-Eigen/1109b663453e78a59e4f66446d71720ac58cec25

[39] “Ramp Loss Linear Programming Support Vector Machine.” Accessed: Mar. 17, 2024. [Online]. Available: https://www.jmlr.org/papers/v15/huang14a.html

[40] “On the dynamics under the unhinged loss and beyond | The Journal of Machine Learning Research.” Accessed: Mar. 17, 2024. [Online]. Available: https://dl.acm.org/doi/10.5555/3648699.3649075

[41] B. van Rooyen, A. Menon, and R. C. Williamson, “Learning with Symmetric Label Noise: The Importance of Being Unhinged,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2015. Accessed: Jan. 18, 2023. [Online]. Available: https://papers.nips.cc/paper/2015/hash/45c48cce2e2d7fbdea1afc51c7c6ad26-Abstract.html

[42] C. Bishop, “Neural networks for pattern recognition,” 1995. Accessed: Jan. 18, 2023. [Online]. Available: https://www.semanticscholar.org/paper/Neural-networks-for-pattern-recognition-Bishop/b9b1b1654ce0eea729c4160bfedcbb3246460b1d

[43] “The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Jan. 31, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/8943952

[44] R. Machlev, Y. Levron, and Y. Beck, “Modified Cross-Entropy Method for Classification of Events in NILM Systems,” *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 4962–4973, Sep. 2019, doi: 10.1109/TSG.2018.2871620.

[45] B. Shan and Y. Fang, “A Cross Entropy Based Deep Neural Network Model for Road Extraction from Satellite Images,” *Entropy*, vol. 22, p. 535, May 2020, doi: 10.3390/e22050535.

[46] P. Li *et al.*, “An improved categorical cross entropy for remote sensing image classification based on noisy labels,” *Expert Syst. Appl.*, vol. 205, p. 117296, Nov. 2022, doi: 10.1016/j.eswa.2022.117296.

[47] “Normalized loss functions for deep learning with noisy labels | Proceedings of the 37th International Conference on Machine Learning.” Accessed: Apr. 23, 2024. [Online]. Available: https://dl.acm.org/doi/abs/10.5555/3524938.3525545

[48] "[PDF] Mixed Cross Entropy Loss for Neural Machine Translation | Semantic Scholar." Accessed: Apr. 23, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Mixed-Cross-Entropy-Loss-for-Neural-Machine-Li-Lu/e8c7a43c7505d4990f4596b1ecc35f01e365a59e

[49] "Dynamically Weighted Balanced Loss: Class Imbalanced Learning and Confidence Calibration of Deep Neural Networks | IEEE Journals & Magazine | IEEE Xplore." Accessed: Jan. 31, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9324926

[50] P. Goyal, "Shallow SegNet with bilinear interpolation and weighted cross-entropy loss for Semantic segmentation of brain tissue," in *2022 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, Mar. 2022, pp. 361–365. doi: 10.1109/SPICES52834.2022.9774193.

[51] K. R. M. Fernando and C. P. Tsokos, "Dynamically Weighted Balanced Loss: Class Imbalanced Learning and Confidence Calibration of Deep Neural Networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 7, pp. 2940–2951, Jul. 2022, doi: 10.1109/TNNLS.2020.3047335.

[52] "Improved Categorical Cross-Entropy Loss for Training Deep Neural Networks with Noisy Labels | SpringerLink." Accessed: Jan. 31, 2023. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-88013-2_7

[53] H. Takeda, S. Yoshida, and M. Muneyasu, "Learning from Noisy Labeled Data Using Symmetric Cross-Entropy Loss for Image Classification," in *2020 IEEE 9th Global Conference on Consumer Electronics (GCCE)*, Oct. 2020, pp. 709–711. doi: 10.1109/GCCE50665.2020.9291873.

[54] "Large-margin softmax loss for convolutional neural networks | Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48." Accessed: Apr. 06, 2024. [Online]. Available: https://dl.acm.org/doi/10.5555/3045390.3045445

[55] "[PDF] L2-constrained Softmax Loss for Discriminative Face Verification | Semantic Scholar." Accessed: Apr. 06, 2024. [Online]. Available: https://www.semanticscholar.org/paper/L2-constrained-Softmax-Loss-for-Discriminative-Face-Ranjan-Castillo/57a807f65e61bbba0ea3ba02572cc5843ecef2b6

[56] “SphereFace: Deep Hypersphere Embedding for Face Recognition | IEEE Conference Publication | IEEE Xplore.” Accessed: Mar. 18, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/8100196

[57] “Additive Margin Softmax for Face Verification | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Feb. 01, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/8331118

[58] T. Kobayashi, “Large Margin In Softmax Cross-Entropy Loss,” presented at the British Machine Vision Conference, 2019. Accessed: Feb. 01, 2023. [Online]. Available: https://www.semanticscholar.org/paper/Large-Margin-In-Softmax-Cross-Entropy-Loss-Kobayashi/0c88506b5e6091906e656dfeb9ee1060946d4aab

[59] X. Xiang, S. Wang, H. Huang, Y. Qian, and K. Yu, “Margin Matters: Towards More Discriminative Deep Neural Network Embeddings for Speaker Recognition,” *2019 Asia-Pac. Signal Inf. Process. Assoc. Annu. Summit Conf. APSIPA ASC*, pp. 1652–1656, Nov. 2019, doi: 10.1109/APSIPAASC47483.2019.9023039.

[60] F. Gao, B. Li, L. Chen, Z. Shang, X. Wei, and C. He, “A softmax classifier for high-precision classification of ultrasonic similar signals,” *Ultrasonics*, vol. 112, p. 106344, Apr. 2021, doi: 10.1016/j.ultras.2020.106344.

[61] “ArcFace: Additive Angular Margin Loss for Deep Face Recognition | IEEE Conference Publication | IEEE Xplore.” Accessed: Feb. 19, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/8953658

[62] H. Wang *et al.*, “CosFace: Large Margin Cosine Loss for Deep Face Recognition,” presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5265–5274. Accessed: Feb. 01, 2023. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Wang_CosFace_Large_Margin_CVPR_2018_paper.html

[63] S. Zhou, C. Chen, G. Han, and X. Hou, “Double Additive Margin Softmax Loss for Face Recognition,” *Appl. Sci.*, vol. 10, p. 60, Dec. 2019, doi: 10.3390/app10010060.

[64] H. Wang, H. Huang, Y. Hu, M. Anderson, P. Rollins, and F. Makedon, “Emotion detection via discriminative kernel method,” in *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments*, in PETRA ’10. New York, NY,

USA: Association for Computing Machinery, Jun. 2010, pp. 1–7. doi: 10.1145/1839294.1839303.

[65] F. Chen, Z. Wang, Z. Xu, J. Xiao, and G. Wang, "Facial Expression Recognition Using Wavelet Transform and Neural Network Ensemble," in *2008 Second International Symposium on Intelligent Information Technology Application*, Dec. 2008, pp. 871–875. doi: 10.1109/IITA.2008.24.

[66] N. Jain, S. Kumar, A. Kumar, P. Shamsolmoali, and M. Zareapoor, "Hybrid deep neural networks for face emotion recognition," *Pattern Recognit. Lett.*, vol. 115, pp. 101–106, Nov. 2018, doi: 10.1016/j.patrec.2018.04.010.

[67] L. Ma and K. Khorasani, "Facial expression recognition using constructive feedforward neural networks," *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 34, no. 3, pp. 1588–1595, Jun. 2004, doi: 10.1109/TSMCB.2004.825930.

[68] "Papers with Code - FER2013 Benchmark (Facial Expression Recognition)." Accessed: Feb. 13, 2023. [Online]. Available: https://paperswithcode.com/sota/facial-expression-recognition-on-fer2013

[69] M.-I. Georgescu, R. T. Ionescu, and M. Popescu, "Local Learning with Deep and Handcrafted Features for Facial Expression Recognition," *IEEE Access*, vol. 7, pp. 64827–64836, 2019, doi: 10.1109/ACCESS.2019.2917266.

[70] Y. Zhang, C. Wang, and W. Deng, "Relative Uncertainty Learning for Facial Expression Recognition," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2021, pp. 17616–17627. Accessed: Nov. 13, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/9332c513ef44b682e9347822c2e457ac-Abstract.html

[71] "Facial Expression Recognition in the Wild via Deep Attentive Center Loss | IEEE Conference Publication | IEEE Xplore." Accessed: Nov. 13, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9423267

[72] "Learn from All: Erasing Attention Consistency for Noisy Label Facial Expression Recognition | Computer Vision – ECCV 2022." Accessed: Apr. 23, 2024. [Online]. Available: https://dl.acm.org/doi/abs/10.1007/978-3-031-19809-0_24

[73] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random Erasing Data Augmentation," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-*

Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, AAAI Press, 2020, pp. 13001–13008. doi: 10.1609/AAAI.V34I07.7000.

[74] A. P. Fard and M. H. Mahoor, "Ad-Corre: Adaptive Correlation-Based Loss for Facial Expression Recognition in the Wild," *IEEE Access*, vol. 10, pp. 26756–26768, 2022, doi: 10.1109/ACCESS.2022.3156598.

[75] H. Ding, S. K. Zhou, and R. Chellappa, "FaceNet2ExpNet: Regularizing a Deep Face Recognition Net for Expression Recognition," presented at the 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), IEEE Computer Society, May 2017, pp. 118–126. doi: 10.1109/FG.2017.23.

[76] "Classifying emotions and engagement in online learning based on a single facial expression recognition neural network | IEEE Journals & Magazine | IEEE Xplore." Accessed: Nov. 14, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9815154

[77] "[PDF] Facial Emotion Recognition: State of the Art Performance on FER2013 | Semantic Scholar." Accessed: Apr. 23, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Facial-Emotion-Recognition%3A-State-of-the-Art-on-Khaireddin-Chen/fac01fc5a47e92f54c7f719bf3c65d36e00c95ac

[78] "Convolutional Neural Network Hyperparameters optimization for Facial Emotion Recognition | IEEE Conference Publication | IEEE Xplore." Accessed: Nov. 14, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9425073

[79] "IVADO Digital Futures 2024 | IVADO." Accessed: Feb. 29, 2024. [Online]. Available: https://ivado.ca/en/events/ivado-digital-futures-2024/

[80] "17th ICCSIT 2024 | Computer Science and Information Technology." Accessed: Mar. 11, 2024. [Online]. Available: https://www.iccsit.org/2022.html

[81] "GSR 2022." Accessed: Mar. 11, 2024. [Online]. Available: https://sites.google.com/view/ece-gsr-2022/home

[82] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*,

Curran Associates, Inc., 2012. Accessed: Dec. 31, 2022. [Online]. Available: https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[83] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, in ICML'15. Lille, France: JMLR.org, Jul. 2015, pp. 448–456.

[84] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng, "Reading Digits in Natural Images with Unsupervised Feature Learning," 2011. Accessed: Feb. 28, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Reading-Digits-in-Natural-Images-with-Unsupervised-Netzer-Wang/02227c94dd41fe0b439e050d377b0beb5d427cda

[85] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 336–359, Feb. 2020, doi: 10.1007/s11263-019-01228-7.

[86] "Learning Deep Global Multi-Scale and Local Attention Features for Facial Expression Recognition in the Wild | IEEE Journals & Magazine | IEEE Xplore." Accessed: Feb. 28, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9474949

[87] Z. Zhao, Q. Liu, and F. Zhou, "Robust Lightweight Facial Expression Recognition Network with Label Distribution Training," *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 4, Art. no. 4, May 2021, doi: 10.1609/aaai.v35i4.16465.

[88] "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." Accessed: Mar. 12, 2024. [Online]. Available: https://www.researchgate.net/publication/333444574_EfficientNet_Rethinking_Model_Scaling_for_Convolutional_Neural_Networks

[89] C. Feichtenhofer, H. Fan, B. Xiong, R. Girshick, and K. He, "A Large-Scale Study on Unsupervised Spatiotemporal Representation Learning," *2021 IEEECVF Conf. Comput. Vis. Pattern Recognit. CVPR*, pp. 3298–3308, Jun. 2021, doi: 10.1109/CVPR46437.2021.00331.

[90] "Trainable Activations for Image Classification[v1] | Preprints.org." Accessed: Jan. 16, 2024. [Online]. Available: https://www.preprints.org/manuscript/202301.0463/v1

[91] S. Verma, A. Chug, and A. P. Singh, “Revisiting activation functions: empirical evaluation for image understanding and classification,” *Multimed. Tools Appl.*, Jul. 2023, doi: 10.1007/s11042-023-16159-2.

[92] P. J. P. and A. Sethi, *WaveMix: Resource-efficient Token Mixing for Images*. 2022.

[93] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” presented at the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Jun. 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.

[94] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” presented at the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Jul. 2017, pp. 2261–2269. doi: 10.1109/CVPR.2017.243.

[95] “Tikhonov Regularization and Total Least Squares | SIAM Journal on Matrix Analysis and Applications.” Accessed: Mar. 19, 2024. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/S0895479897326432?journalCode=sjmael

[96] J. Bridle, “Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters,” in *Advances in Neural Information Processing Systems*, Morgan-Kaufmann, 1989. Accessed: Jan. 18, 2023. [Online]. Available: https://proceedings.neurips.cc/paper/1989/hash/0336dcbab05b9d5ad24f4333c7658a0e-Abstract.html

[97] “[PDF] Gaussian Mixture Convolution Networks | Semantic Scholar.” Accessed: Apr. 23, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Gaussian-Mixture-Convolution-Networks-Celarek-Hermosilla/ce108373d3458c2f27524483221aaad772c3edf3

[98] “An Introduction to Logistic Regression Analysis and Reporting: The Journal of Educational Research: Vol 96, No 1.” Accessed: Mar. 20, 2024. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00220670209598786

[99] E. Trentin, “Maximum-Likelihood Estimation of Neural Mixture Densities: Model, Algorithm, and Preliminary Experimental Evaluation,” 2018, pp. 178–189. doi: 10.1007/978-3-319-99978-4_14.

[100] "Soft-Margin Softmax for Deep Classification | SpringerLink." Accessed: Feb. 01, 2023. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-70096-0_43

[101] "Large scale multi-output multi-class classification using Gaussian processes | Machine Learning." Accessed: Mar. 20, 2024. [Online]. Available: https://link-springer-com.lib-ezproxy.concordia.ca/article/10.1007/s10994-022-06289-3

[102] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. in Springer Series in Statistics. New York, NY: Springer, 2009. doi: 10.1007/978-0-387-84858-7.

[103] "[PDF] Striving for Simplicity: The All Convolutional Net | Semantic Scholar." Accessed: Apr. 23, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Striving-for-Simplicity%3A-The-All-Convolutional-Net-Springenberg Dosovitskiy/33af9298e5399269a12d4b9901492fe406af62b4

[104] "[PDF] Qualitatively characterizing neural network optimization problems | Semantic Scholar." Accessed: Mar. 20, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Qualitatively-characterizing-neural-network-Goodfellow-Vinyals/4d4d09ae8f6a11547441f7fee36405758102a801

[105] "[PDF] ARBEx: Attentive Feature Extraction with Reliability Balancing for Robust Facial Expression Learning | Semantic Scholar." Accessed: Apr. 23, 2024. [Online]. Available: https://www.semanticscholar.org/paper/ARBEx%3A-Attentive-Feature-Extraction-with-Balancing-Wasi-vSerbetar/b468a27c1ea7692d4de00ff5a948f3bfeaa63a60

[106] D. Ruan, Y. Yan, S. Lai, Z. Chai, C. Shen, and H. Wang, "Feature Decomposition and Reconstruction Learning for Effective Facial Expression Recognition," *2021 IEEECVF Conf. Comput. Vis. Pattern Recognit. CVPR*, pp. 7656–7665, Jun. 2021, doi: 10.1109/CVPR46437.2021.00757.

[107] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, Jun. 2010, pp. 94–101. doi: 10.1109/CVPRW.2010.5543262.

[108] S. Li, W. Deng, and J. Du, "Reliable Crowdsourcing and Deep Locality-Preserving Learning for Expression Recognition in the Wild," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2584–2593. doi: 10.1109/CVPR.2017.277.

[109] S. Chen, J. Wang, Y. Chen, Z. Shi, X. Geng, and Y. Rui, "Label Distribution Learning on Auxiliary Label Space Graphs for Facial Expression Recognition," presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13984–13993. Accessed: Apr. 07, 2024. [Online]. Available: https://openaccess.thecvf.com/content_CVPR_2020/html/Chen_Label_Distribution_Learning_on_Auxiliary_Label_Space_Graphs_for_Facial_CVPR_2020_paper.html

[110] "Pyramid With Super Resolution for In-the-Wild Facial Expression Recognition | IEEE Journals & Magazine | IEEE Xplore." Accessed: Apr. 07, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/9143068

[111] D. Zeng, Z. Lin, X. Yan, Y. Liu, F. Wang, and B. Tang, "Face2Exp: Combating Data Biases for Facial Expression Recognition," presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 20291–20300. Accessed: Apr. 07, 2024. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/html/Zeng_Face2Exp_Combating_Data_Biases_for_Facial_Expression_Recognition_CVPR_2022_paper.html

[112] C. L. Lisetti and D. E. Rumelhart, "Facial Expression Recognition," p. 5.

[113] "[PDF] MViT: Mask Vision Transformer for Facial Expression Recognition in the wild | Semantic Scholar." Accessed: Apr. 07, 2024. [Online]. Available: https://www.semanticscholar.org/paper/MViT%3A-Mask-Vision-Transformer-for-Facial-Expression-Li-Sui/070c917ab1a4d6b924a9613ca18443f260d8d5be

[114] "[PDF] TransFER: Learning Relation-aware Facial Expression Representations with Transformers | Semantic Scholar." Accessed: Apr. 07, 2024. [Online]. Available: https://www.semanticscholar.org/paper/TransFER%3A-Learning-Relation-aware-Facial-Expression-Xue-Wang/ebf221bf7260e2d27b243b15909d89196f62f39b

[115] A. Vaswani *et al.*, "Attention is All you Need," presented at the Neural Information Processing Systems, Jun. 2017. Accessed: Apr. 07, 2024. [Online]. Available:

https://www.semanticscholar.org/paper/Attention-is-All-you-Need-Vaswani-Shazeer/204e3073870fae3d05bcbc2f6a8e263d9b72e776

[116] A. H. Farzaneh and X. Qi, "Facial Expression Recognition in the Wild via Deep Attentive Center Loss," in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2021, pp. 2401–2410. doi: 10.1109/WACV48630.2021.00245.

[117] "Relative Uncertainty Learning for Facial Expression Recognition." Accessed: Apr. 07, 2024. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/9332c513ef44b682e9347822c2e457ac-Abstract.html

[118] "[PDF] Deep-Emotion: Facial Expression Recognition Using Attentional Convolutional Network | Semantic Scholar." Accessed: Apr. 07, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Deep-Emotion%3A-Facial-Expression-Recognition-Using-Minaee-Abdolrashidi/361b59d962068d47cac30c6fabbf1b8a91efcb2a

[119] I. J. Goodfellow *et al.*, "Challenges in Representation Learning: A Report on Three Machine Learning Contests," in *Neural Information Processing*, M. Lee, A. Hirose, Z.-G. Hou, and R. M. Kil, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, pp. 117–124. doi: 10.1007/978-3-642-42051-1_16.

[120] L. Pham, T. H. Vu, and T. A. Tran, "Facial Expression Recognition Using Residual Masking Network," in *2020 25th International Conference on Pattern Recognition (ICPR)*, Jan. 2021, pp. 4513–4519. doi: 10.1109/ICPR48806.2021.9411919.

[121] "[PDF] Local Multi-Head Channel Self-Attention for Facial Expression Recognition | Semantic Scholar." Accessed: Apr. 23, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Local-Multi-Head-Channel-Self-Attention-for-Facial-Pecoraro-Basile/cde1045b1e9f3b3939ae54df6ae8a2e3079b6d15

[122] "Local Learning With Deep and Handcrafted Features for Facial Expression Recognition | IEEE Journals & Magazine | IEEE Xplore." Accessed: Apr. 07, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/8716652

[123] "[PDF] PAtt-Lite: Lightweight Patch and Attention MobileNet for Challenging Facial Expression Recognition | Semantic Scholar." Accessed: Apr. 23, 2024. [Online]. Available: https://www.semanticscholar.org/paper/PAtt-Lite%3A-Lightweight-Patch-and-Attention-for-Ngwe-Lim/b0a3332bc56133fef59eba242f44add54fc2af9b

[124] "EmoNeXt: an Adapted ConvNeXt for Facial Emotion Recognition | IEEE Conference Publication | IEEE Xplore." Accessed: Apr. 07, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10337732

[125] S. Vignesh, M. Savithadevi, M. Sridevi, and R. Sridhar, "A novel facial emotion recognition model using segmentation VGG-19 architecture," *Int. J. Inf. Technol.*, vol. 15, no. 4, pp. 1777–1787, Apr. 2023, doi: 10.1007/s41870-023-01184-z.

[126] "Fer2013 Recognition - ResNet18 With Tricks | Papers With Code." Accessed: Aug. 14, 2023. [Online]. Available: https://paperswithcode.com/paper/fer2013-recognition-resnet18-with-tricks

[127] "Facial Expression Recognition via Deep Learning | IEEE Conference Publication | IEEE Xplore." Accessed: Apr. 07, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/8308363

[128] H. Vo, G.-S. Lee, H.-J. Yang, and S. H. Kim, "Pyramid with Super Resolution for In-The-Wild Facial Expression Recognition," *IEEE Access*, vol. PP, pp. 1–1, Jul. 2020, doi: 10.1109/ACCESS.2020.3010018.

[129] Z. Wen, W. Lin, T. Wang, and G. Xu, "Distract Your Attention: Multi-Head Cross Attention Network for Facial Expression Recognition," *Biomim. Basel Switz.*, vol. 8, no. 2, p. 199, May 2023, doi: 10.3390/biomimetics8020199.

[130] S. Zhang, Y. Zhang, Y. Zhang, Y. Wang, and Z. Song, "A Dual-Direction Attention Mixed Feature Network for Facial Expression Recognition," *Electronics*, vol. 12, no. 17, Art. no. 17, Jan. 2023, doi: 10.3390/electronics12173595.

[131] J. Li, J. Nie, D. Guo, R. Hong, and M. Wang, "Emotion Separation and Recognition from a Facial Expression by Generating the Poker Face with Vision Transformers." arXiv, Jun. 09, 2023. doi: 10.48550/arXiv.2207.11081.

[132] F. Xue, Q. Wang, and G. Guo, "TransFER: Learning Relation-aware Facial Expression Representations with Transformers," *2021 IEEECVF Int. Conf. Comput. Vis. ICCV*, pp. 3581–3590, Oct. 2021, doi: 10.1109/ICCV48922.2021.00358.

[133] "MViT: Mask Vision Transformer for Facial Expression Recognition in the wild." Accessed: Apr. 08, 2024. [Online]. Available: https://www.researchgate.net/publication/352244399_MViT_Mask_Vision_Transformer_for_Facial_Expression_Recognition_in_the_wild

[134] A. Psaroudakis and D. Kollias, "MixAugment & Mixup: Augmentation Methods for Facial Expression Recognition," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 2366–2374. doi: 10.1109/CVPRW56347.2022.00264.

[135] "[PDF] Suppressing Uncertainties for Large-Scale Facial Expression Recognition | Semantic Scholar." Accessed: Apr. 08, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Suppressing-Uncertainties-for-Large-Scale-Facial-Wang-Peng/92006a84036f9e37e11c773c19141ad8a6675e39

[136] K. Wang, X. Peng, J. Yang, D. Meng, and Y. Qiao, "Region Attention Networks for Pose and Occlusion Robust Facial Expression Recognition," *IEEE Trans. Image Process.*, vol. 29, pp. 4057–4069, 2020, doi: 10.1109/TIP.2019.2956143.

[137] "[PDF] Occlusion Aware Facial Expression Recognition Using CNN With Attention Mechanism | Semantic Scholar." Accessed: Apr. 08, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Occlusion-Aware-Facial-Expression-Recognition-Using-Li-Zeng/b5bb7e12a15b57b4d307e742da127a74596d0c7c

[138] "[PDF] Learning Vision Transformer with Squeeze and Excitation for Facial Expression Recognition | Semantic Scholar." Accessed: Apr. 23, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Learning-Vision-Transformer-with-Squeeze-and-for-Aouayeb-Hamidouche/7109de2f4c522af71098e99563c90db62c19ba55

[139] "Using Self-Supervised Auxiliary Tasks to Improve Fine-Grained Facial Representation | Semantic Scholar." Accessed: Aug. 10, 2023. [Online]. Available: https://www.semanticscholar.org/paper/Using-Self-Supervised-Auxiliary-Tasks-to-Improve-Pourmirzaei-Esmaili/8c34613a9c123246802e079fa0feb0709f090087

[140] "Facial Expression Recognition Based on Deep Evolutional Spatial-Temporal Networks - PubMed." Accessed: Apr. 08, 2024. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/28371777/

[141] "Spatial-Temporal Recurrent Neural Network for Emotion Recognition - PubMed." Accessed: Oct. 18, 2023. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/29994572/

[142] "(PDF) Identity-Free Facial Expression Recognition Using Conditional Generative Adversarial Network." Accessed: Apr. 08, 2024. [Online]. Available:

https://www.researchgate.net/publication/351736453_IdentityFree_Facial_Expression_Recognition_Using_Conditional_Generative_Adversarial_Network

[143] "Peak-Piloted Deep Network for Facial Expression Recognition | SpringerLink." Accessed: Apr. 23, 2024. [Online]. Available: https://link-springer-com.lib-ezproxy.concordia.ca/chapter/10.1007/978-3-319-46475-6_27

[144] H. Jung, S. Lee, J. Yim, S. Park, and J. Kim, "Joint Fine-Tuning in Deep Neural Networks for Facial Expression Recognition," presented at the Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2983–2991. Accessed: Oct. 18, 2023. [Online]. Available: https://openaccess.thecvf.com/content_iccv_2015/html/Jung_Joint_Fine-Tuning_in_ICCV_2015_paper.html

[145] "Occlusion Aware Facial Expression Recognition Using CNN With Attention Mechanism | IEEE Journals & Magazine | IEEE Xplore." Accessed: Apr. 08, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/8576656

[146] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 7482–7491. doi: 10.1109/CVPR.2018.00781.

[147] "[PDF] Multi-Task Learning with Deep Neural Networks: A Survey | Semantic Scholar." Accessed: Apr. 08, 2024. [Online]. Available: https://www.semanticscholar.org/paper/Multi-Task-Learning-with-Deep-Neural-Networks%3A-A-Crawshaw/74f23063ca77f5b1caa3770a5957ae5fc565843e