Facial Recognition of Olivetti Dataset using Optimized CNN Model

Jungwoo Park jup006@ucsd.edu
Department of Cognitive Science
University of California San Diego
San Diego, CA 92122, USA

Sarah Borsotto sborsott@ucsd.edu
Department of Cognitive Science
University of California San Diego
San Diego, CA 92122, USA

Editor:

Abstract

Our study explores the application of Convolutional Neural Networks (CNN) to the facial recognition task using the Olivetti Faces dataset, which consists of 400 images across 40 classes. Inspired by the LeNet-5 CNN model built by Zhiming Xie et al. in 2019, we developed a new ResNet18 architecture using PyTorch. After achieving higher accuracy with our model, we experimented with different combinations of optimizers and activation functions. We discovered that the ResNet model paired with the Adam optimizer and Leaky ReLU activation function achieves the highest test accuracy, alongside one of the smallest losses observed. Our findings suggest that careful manipulation of activation functions and optimizers, paired with a suitable modern architecture, can significantly enhance the performance of facial recognition models.

1 Introduction

The main goal of facial recognition technology is to confirm an individual's identity based on their facial characteristics. Facial recognition is applied within various different fields, including security and law enforcement for surveillance and access control, user authentication in mobile devices, personalization in social media and advertising, and healthcare for patient identification and monitoring. This process relies on algorithms that can effectively extract distinct facial features from photos or videos, compare these features to other samples, and identify the most likely candidates. Techniques such as principal component analysis (PCA) and Support Vector Machine classifier (SVM), Histogram of Gradients (HOG) and Relevance Vector Machine (RVM), independent component analysis (ICA), and local binary patterns (LBP) have greatly improved facial recognition technology in the past. However, the most notable recent advancement has stemmed from deep learning systems, particularly convolutional neural networks (CNNs), which efficiently combine the task of feature extraction and classification into one model (K B & J, 2020).

CNN's have garnered attention for their immense computational capabilities in object recognition and high accuracy outcomes across a wide range of datasets. Despite achieving

a high degree of precision in controlled environments, these models face significant limitations in uncontrolled settings. Such challenges include pose variation, where changes in subject orientation hinder recognition, illumination variation caused by unpredictable lighting conditions, and occlusion, where facial features are obscured by objects such as hats, sunglasses, or scarves. These factors critically impact the model's ability to perform a facial recognition task (Ali et al., 2020). In an effort to address these concerns, we carefully selected a dataset designed to encompass a variety of image modifications. Our choice led us to the Olivetti dataset, which includes images captured under diverse circumstances, including different lighting conditions, facial expressions (such as open/closed eyes, smiling/not smiling), presence or absence of glasses, and subjects positioned upright with some tolerance for minor side movements.

Zhiming Xie et al. (2019) constructed a CNN model based on the LeNet-5 architecture specifically for this dataset. The study investigated the network's behavior when altering the number of neurons in the hidden layer and varying the number of feature maps in convolutional layers 1 and 2. Through extensive experimentation, the optimal CNN model identified was 36-76-1024, achieving a recognition rate close to 100 percent. Yet, their implementation relied on outdated packages, namely Keras and TensorFlow. In a more modern study, Almabdy and Elrefaei (2019) assessed the performance of pre-trained convolutional neural network models, specifically AlexNet and ResNet-50, for facial recognition tasks. They found strong results with accuracy ranging between 94 percent to 100 percent across eight databases, suggesting potential applicability to the Olivetti dataset. While these outcomes are impressive, we aim to develop a customized model tailored for our dataset by leveraging the newer ResNet architecture instead of applying pre-trained models, thereby addressing a gap in the current literature.

2 Method

A Convolutional Neural Network (CNN) is a class of deep neural networks used extensively in Pattern Recognition and Artificial Intelligence. In these fields, convolution operates as a "template" and is a linear operation involving a filter that is applied uniformly across all locations. This filter is learned during training. The design of CNNs necessitates consideration of several key components: Softmax loss for multiclass classification; convolution, where weights are automatically learned but sizes are manually specified; activation functions, including options such as sigmoid, tanh, and ReLU, which are manually selected; pooling methods, such as max, average, stochastic, and mixed, also manually specified; padding, to fill out boundary pixels/cells; and stride, the gap in convolution, which is manually adjusted. CNNs have proven to be powerful tools for tasks such as face recognition.

Our team selected ResNet as our architecture of choice. Short for Residual Network, ResNet is a specialized CNN designed to enhance learning speed and efficiency by incorporating residual blocks with skip connections. These connections allow the input from one layer to bypass some layers and be added to the output of a subsequent layer. This approach facilitates the backpropagation of gradients and effectively addresses the vanishing gradient problem, making ResNet's variable depth highly beneficial for computer vision tasks, including face recognition.

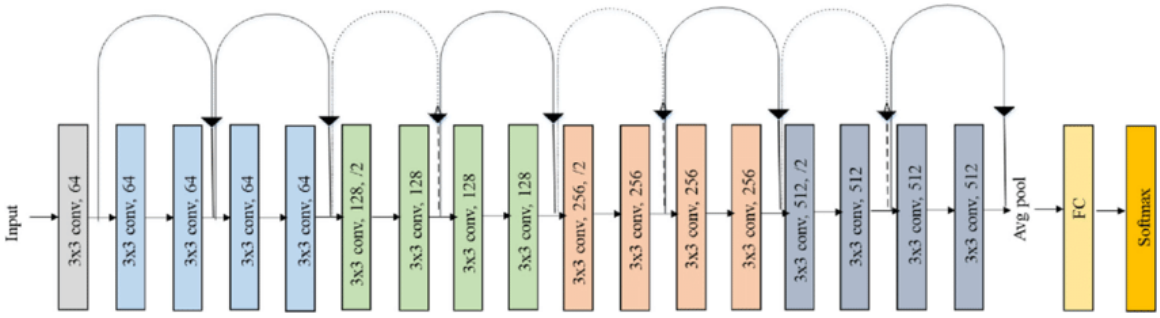

Figure 1: ResNet 18

2.1 Network Architecture

Based on our dataset, we set our ResNet depth to 18 layers. Each BasicBlock contains convolutional layers with 3x3 kernels. Batch normalization is applied after each convolutional layer to stabilize and accelerate training. We used ReLU as the activation function and applied dropout after the first activation function to prevent overfitting by randomly dropping units along with their connections during training.

For the ResNet architecture, we constructed four main layers of the network using the "make_layer" function, with each layer having a varying number of blocks and channels. The stride is set to 2 for layers 2, 3, and 4, enabling downsampling and increasing the receptive field with each subsequent layer. After the last layer, we performed average pooling to reduce the spatial dimension to 1x1, thus focusing solely on significant features for recognition. The output is then flattened and passed through a fully connected linear layer to produce the final classification output, corresponding to the 40 classes in our Olivetti Faces dataset.

During training, we made sure to utilize a GPU, leveraging CUDA for acceleration. The training spanned 40 epochs, each consisting of training and evaluation phases. For optimization and loss calculation, we used Adam, which facilitates quicker convergence by adapting the learning rate for each parameter, and cross-entropy loss, which is effective for multi-class classification tasks like face recognition.

2.2 Convolutional Layer

A convolutional layer in a CNN model extracts features from input data, such as images, by applying filters that detect patterns like edges or textures, helping the network learn representations relevant to the task at hand. Each additional layer can capture increasingly complex features by building upon previous representations. This hierarchical approach allows the network to learn intricate patterns and relationships within the data. However, adding too many layers may induce the vanishing gradient problem. As the number of layers in a neural network increases, the gradients of the loss function used to update the weights of the model approach zero, leading to a "disappearing" gradient. This can be seen in the activation function, such as sigmoid, where larger values are linked with smaller slopes (Wang, 2019). When the gradients are too small, the network is unable to learn effectively, resulting in poorer performance. ResNets are able to combat this issue through "skip connections," permitting the network to train numerous layers by reusing activations

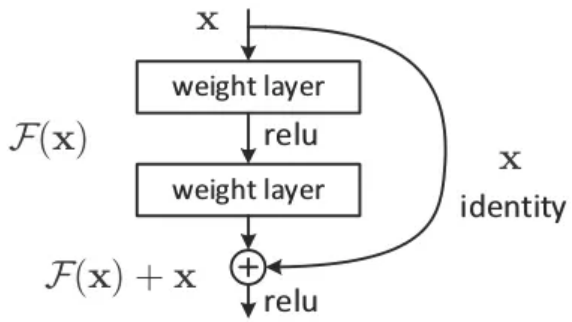

Figure 2: Residual Block in ResNet's Convolutional Layer

from previous layers. ResNet's convolutional layer incorporates residual connections directly to earlier layers. In this approach, illustrated in Figure 1, the residual connection adds the original value at the beginning of the block, denoted as x, to the output of the block (F(x) + x). By bypassing activation functions that could dampen derivatives, this residual connection maintains a higher overall derivative for the block.

2.3 Pooling Layer

The pooling layer spatially downsamples the input volume independently in each depth slice. There are two main objectives of pooling. The first is to achieve invariance and robustness to local image content changes. The second is to reduce the image size for dimensionality reduction. In our model, we utilized average pooling. Although max pooling, which focuses on the most prominent features, is more common, average pooling tends to retain more background information and fewer feature-specific details by comparison. This method ensures that the network pays attention to overall facial traits rather than potentially being distracted by specific facial features that might not be as useful for recognition. For comparatively smaller datasets, avoiding overfitting is crucial, and average pooling excels by representing larger regions. Lastly, its even distribution ensures a smooth gradient during backpropagation, leading to better convergence during training.

2.4 Activation Function

The activation function in a CNN model introduces non-linearities to the network, allowing it to learn and represent complex patterns and relationships within the data. They determine whether or not an artificial neuron should be activated based on the weighted sum of its inputs. These functions are an essential part of the network's learning process; without nonlinearity, we would simply be training a single perceptron model. ReLU (Rectified Linear Unit) is a widely used activation function in neural networks, known for its simplicity and effectiveness, as well as its ability to prevent the vanishing gradient problem. Mathematically, ReLU(x) = max(0, x), meaning it outputs the input value if it's positive, and 0 otherwise (Sharma, 2023). This piecewise linear nature of ReLU makes it computationally efficient to compute and differentiate, leading to impressive results. We will be exploring other activation functions in our experiment, such as LeakyReLU and TanH.

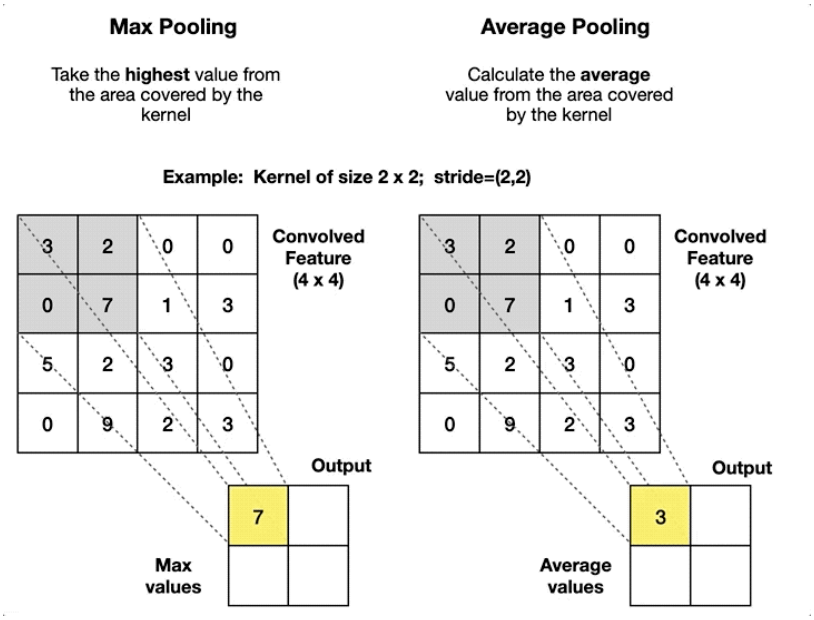

Figure 3: Max vs. Avg Pooling

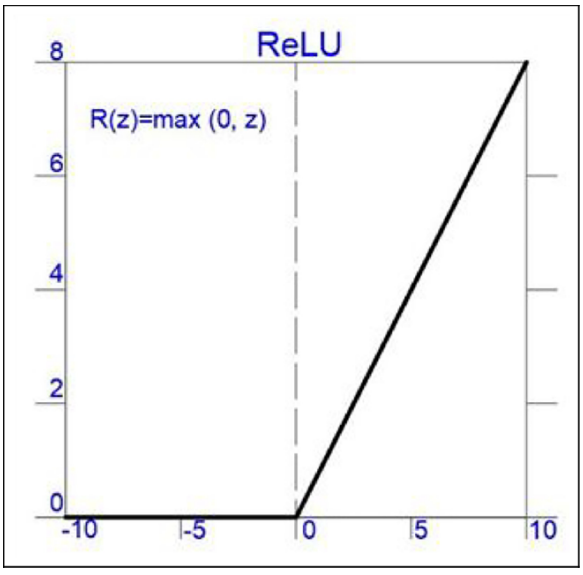

Figure 4: Relu Activation Function

2.5 Optimizer Function

The optimizer's role is to adjust the network's weights to minimize the loss function. It utilizes the gradients calculated during backpropagation to update each parameter in a direction that will reduce the loss. We utilized Adam, also known as Adaptive Moment Estimation, which merges the best characteristics of the AdaGrad and RMSProp algorithms. Adam is highly favored for its efficiency in terms of computational resource requirements and minimal memory consumption. It facilitates faster and more effective convergence compared to simpler algorithms like Stochastic Gradient Descent, especially for complex and varying loss functions encountered in tasks like face recognition.

3 Experiment

The experiment aims to investigate the impact of different activation functions and optimization methods on the performance of a ResNet CNN model. Six different combinations are tested: ReLU with Adam (Model 1), Leaky ReLU with Adam (Model 2), Tanh with Adam (Model 3), ReLU with Stochastic Gradient Descent (SGD) (Model 4), Leaky ReLU with SGD (Model 5), and Tanh with SGD (Model 6). Each combination is trained and tested on the same training and test datasets.

During training, the model's loss, training accuracy, and test accuracy are monitored over multiple epochs. By comparing these metrics across various combinations, the experiment seeks to identify which activation function and optimization method yield the best performance in terms of minimizing loss and maximizing accuracy. We also compare the overall test accuracy for each trial. Ultimately, the goal is to determine the most effective combination for the given ResNet CNN model architecture.

3.1 Dataset

The dataset comprises a collection of face images captured at AT&T Laboratories Cambridge between April 1992 and April 1994 (Pedregosa et al., 2011). Utilizing sklearn, we accessed the Olivetti faces dataset, which consists of 400 images distributed among 40 classes and ten images per class representing different individuals. The photos were captured against a uniform dark background, with variations in lighting and facial expressions across subjects and images. The resulting grayscale images were cropped to 64x64 pixels and normalized between [0,1] for easier usage. The dataset is organized into four categories: data, images, target, and description. The "data" array (400, 4096) contains flattened face images, each originally sized 64 x 64 pixels, reshaped into vectors suitable for CNN models. The "images" array holds individual face images corresponding to the 40 subjects, while the "target" array contains the true labels of the individual, ranging from 0 to 39, associated with each image. In order to effectively prepare the data for the CNN model, we split the data into training and test sets of size 300 to 100 respectively, reshaped the image data into (-1, 1, 64, 64) [N, C, H, W], converted the numpy arrays into PyTorch tensors, and loaded the data into data loaders. We provide an illustration of all 40 people in Figure 4 and an example of all ten samples for five people in Figure 4.

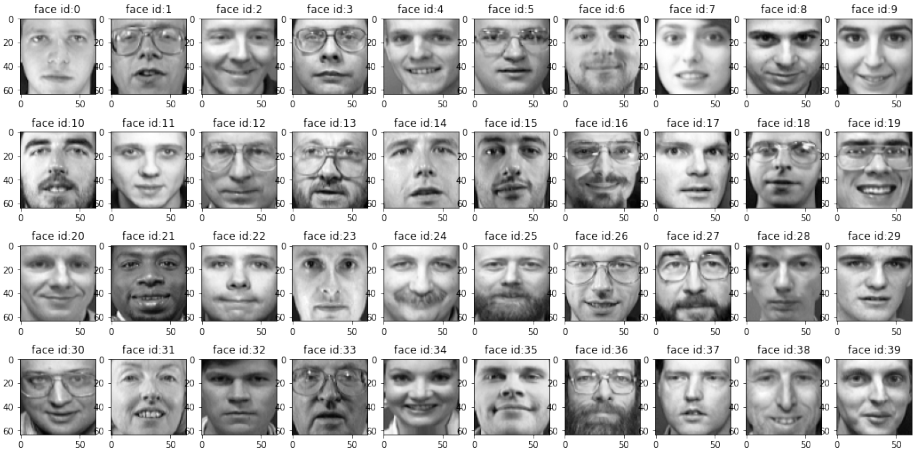

Figure 5: Images of all 40 People from the Dataset

Figure 6: All Ten Images of 5 Selected People from the Dataset

3.2 Baseline CNN Model

For our ResNet model tailored to the Olivetti Faces dataset, we opted for an 18-layer configuration, utilizing BasicBlocks that incorporate 3x3 kernel convolutional layers. After convolution, batch normalization is employed to ensure training stability and accelerate the process, paired with ReLU activation functions. To mitigate overfitting, dropout is strategically implemented after the initial activation, randomly omitting units and their connections during the training phase. The architecture is structured into four primary layers through the "make_layer" function, varying in block count and channel capacity, with layers 2 through 4 employing a stride of 2 for downsampling and expanding receptive fields. Following the final layer, average pooling reduces spatial dimensions to a singular focus, emphasizing essential recognition features. This processed output is then linearized and fed through a fully connected layer, culminating in classification across the 40 distinct classes of the dataset. Throughout the 40-epoch training regimen, which includes both training and evaluation phases, we leverage GPU acceleration, specifically CUDA, to enhance performance. The Adam optimizer and cross-entropy loss function are chosen for their efficiency in hastening convergence and their effectiveness in multi-class classification challenges, respectively, perfectly aligning with our facial recognition task.

3.2.1 Baseline Results

For our baseline model, which pairs Adam with ReLU, we achieved a training accuracy of 100%, indicating that the model perfectly learned the training dataset and managed to correctly classify all faces. Initially, we were concerned about potential overfitting; however, our concerns were alleviated when we observed a testing accuracy of 96%, closely aligning with the training accuracy. This performance suggests that the model can generalize well to unseen data, and the features learned during training are robust and effective for classifying the 40 classes. Furthermore, with a loss of 0.0010013706400059164—a very low number—it is evident that the model's predictions are exceedingly close to the actual labels, with minimal errors. This outcome implies that Adam is well-suited for navigating the loss landscape and identifying optimal weights for this model.

3.3 Hyper-parameters

We focus our attention on two types of hyperparameters: activation functions and optimization functions. Activation functions determine the non-linear transformations applied to neuron outputs, enabling the network to capture complex patterns and relationships in the data. Meanwhile, optimization functions guide the parameter updates during training, influencing convergence speed and final performance. The functions we are interested in exploring are ReLU, LeakyReLU, TanH, Adam optimizer, and Stochastic Gradient Descent optimizer.

ReLU - As described in the methods section, ReLU is a widely used activation function that introduces non-linearity by returning the input value for positive inputs and zero for negative inputs. It reduces the vanishing gradient problem and accelerates convergences.

LeakyReLU - Leaky ReLU is a variant of ReLU that addresses the "dying ReLU" problem, where neurons can become inactive during training. It introduces a small slope

Combination
ReLU with Adam
Leaky ReLU with Adam
Tanh with Adam
ReLU with SGD
Leaky ReLU with SGD
Tanh with SGD

Figure 7: Hyperparameter Combinations

for negative inputs, preventing the zero gradient issue and enabling the flow of gradients even for negative values (Shah, 2024).

Tanh - Tanh is another common activation function that transforms input values into a range of [-1, 1]. It is similar to the sigmoid function but centered at zero. The advantage of using tanh instead of sigmoid is that it maps negative inputs into strongly negative outputs and maps near-zero inputs into near-zero outputs, meaning that the tanh graph leads to more sensitive derivatives (Sharma, 2022).

Adam Optimizer - Adam is an adaptive optimization algorithm that combines the advantages of both AdaGrad and RMSProp. It dynamically adjusts learning rates for each parameter, corrects for initialization bias, stabilizes the training processes, and produces fast convergence rates (Vishwakarma, 2023).

Stochastic Gradient Descent Optimizer - This is a classic optimization algorithm that updates model parameters based on the gradients computed from a random subset of the training data. It converges to a minimum by iteratively adjusting parameter values in the direction of steepest descent (Stojiljković, 2023).

We presume that the ReLU and Adam optimizer combination will have the best performance as they are the most popular functions used for current CNN models.

3.4 Training Process

For the ResNet model we developed using PyTorch, we experimented with different combinations of optimizers and activation functions. As optimizers, we selected Adam and Stochastic Gradient Descent (SGD). Regarding activation functions, we chose ReLU, Leaky ReLU, and Tanh.

3.4.1 Results of Adjusted CNN Model

Contrary to our expectations, Adam paired with Leaky ReLU achieved almost the same loss as Adam with ReLU but demonstrated a higher test accuracy of 97%. This superior performance with Leaky ReLU suggests that its adjustment for negative inputs enables the

	Combination	Final Loss	Final Train Accuracy	Final Test Accuracy
0	ReLU with Adam	0.001001	100.000000	96.0
1	Leaky ReLU with Adam	0.001095	100.000000	97.0
2	Tanh with Adam	0.005366	100.000000	82.0
3	ReLU with SGD	2.542081	42.666667	21.0
4	Leaky ReLU with SGD	2.441295	52.666667	34.0
5	Tanh with SGD	2.916726	30.333333	13.0

Figure 8: Final performance

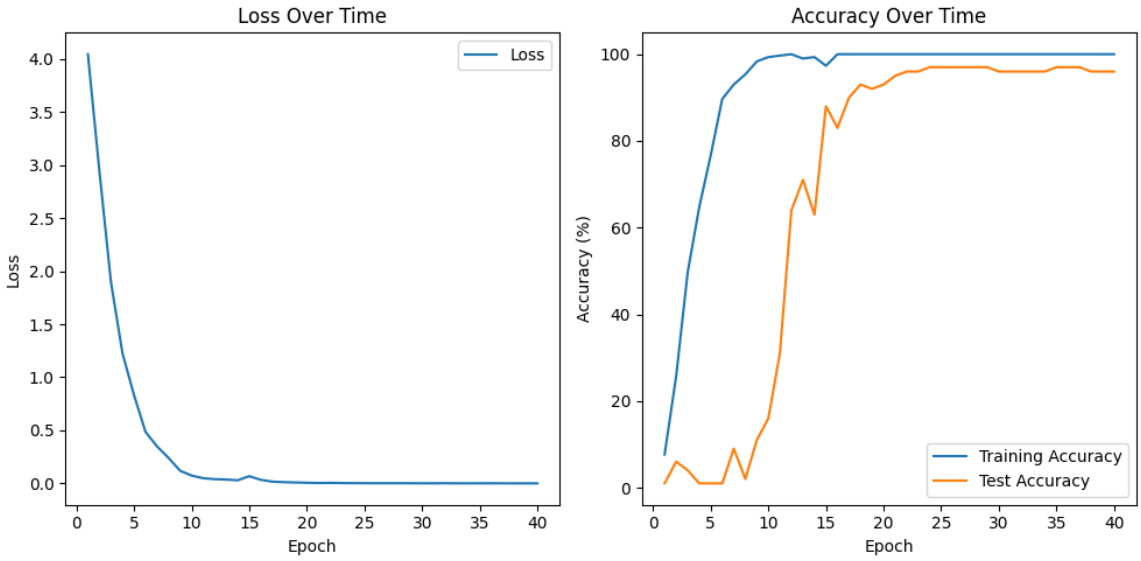

Figure 9: Model1 Performance

network to learn more robust features that better generalize to unseen data. This could imply that Leaky ReLU allows our model to maintain improved gradient flow in situations where ReLU might result in zero gradients, thus facilitating a more nuanced understanding of the Olivetti dataset.

The poorer performance of SGD compared to Adam could be attributed to SGD's fixed learning rate, which may get stuck in saddle points or struggle with the complexity and poorer handling of sparse gradients.

4 Conclusion

In summary, our study explored facial recognition using a customized ResNet18 CNN model on the Olivetti Faces dataset. Our adoption of the ResNet architecture proved beneficial, by addressing the vanishing gradient problem, we were able to implement more layers to support the network's learning. The experiment investigated the influence of different opti-

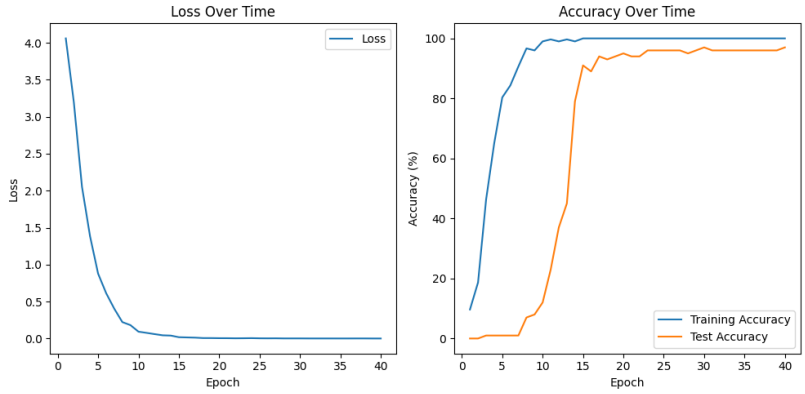

Figure 10: Model2 Performance

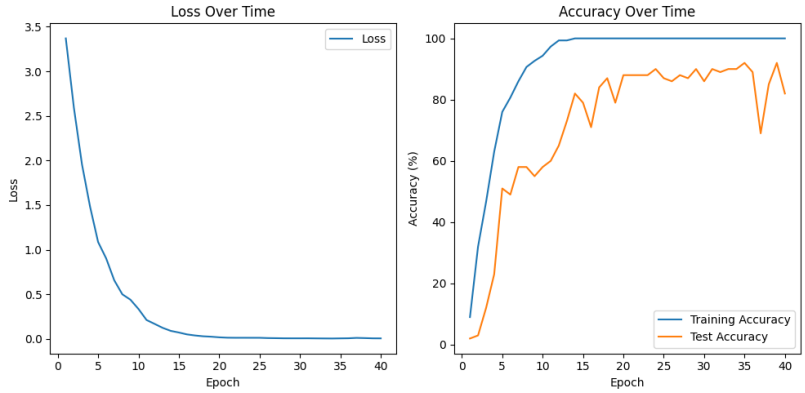

Figure 11: Model3 Performance

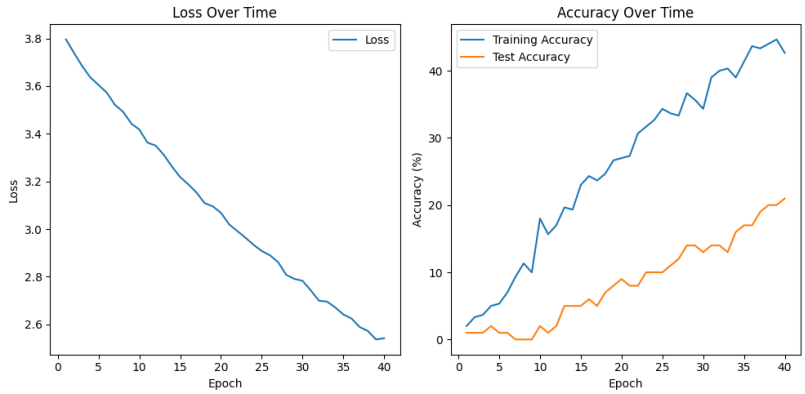

Figure 12: Model4 Performance

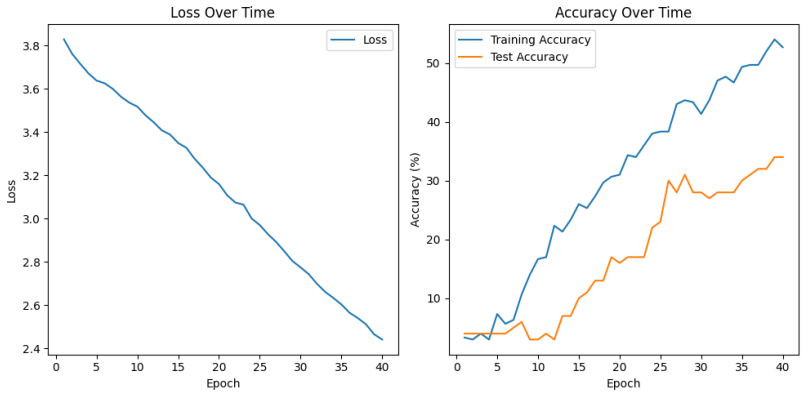

Figure 13: Model5 Performance

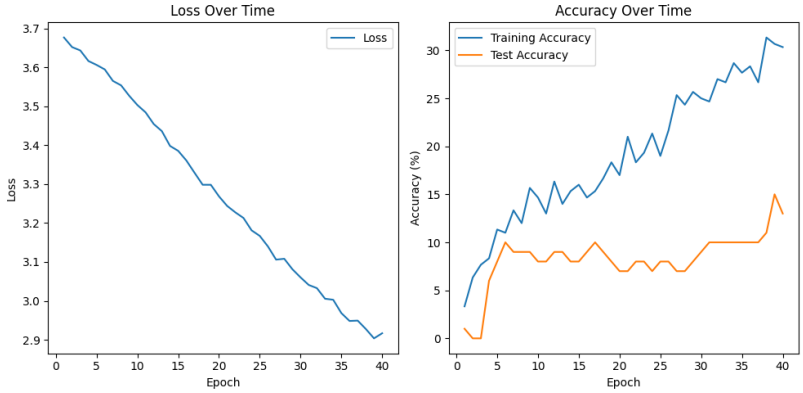

Figure 14: Model6 Performance

mizer and activation functions on model performance. Among the six combinations tested, pairing the Adam optimizer with Leaky ReLU activation function yielded the highest test accuracy and lowest loss. The adjustment for negative inputs in Leaky ReLU facilitated improved gradient flow, enabling the model to learn more robust features and generalize better to unseen data. On the other hand, the poorer performance of the Stochastic Gradient Descent optimizer underscores the importance of adaptive learning rates when working with complex datasets. These findings emphasize the critical role of activation functions and optimizers in improving facial recognition models' performance. A model's rate of success can vary significantly based on the hyperparameters chosen for the model, as subtle changes in parameters can have a profound impact on the model's overall effectiveness and efficiency. Therefore, conducting comprehensive testing on a model and exploring various options is crucial, especially for real-world applications.

References

[1] Ali, W. et al. (2020) 'Classical and modern face recognition approaches: A complete review', Multimedia Tools and Applications, 80(3), pp. 4825–4880. doi:10.1007/s11042-020-09850-1.

[2] Almabdy, S. and Elrefaei, L. (2019) 'Deep convolutional neural network-based approaches for face recognition', Applied Sciences, 9(20), p. 4397. doi:10.3390/app9204397.

[3] K B, P. and J, M. (2020) 'Design and evaluation of a real-time face recognition system using convolutional neural networks', Procedia Computer Science, 171, pp. 1651–1659. doi:10.1016/j.procs.2020.04.177.

[4] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

[5] Shah, S. (2024) What is leaky relu?, Educative. Available at: `https://www.educative.io/answers/what-is-leaky-relu` (Accessed: 22 March 2024).

[6] Sharma, S. (2022) Activation functions in neural networks, Medium. Available at: `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6` (Accessed: 22 March 2024).

[7] Sharma, T. (2023) Detailed explanation of residual network(resnet50) CNN model., Medium. Available at: `https://medium.com/@sharma.tanish096/detailed-explanation-of-residual-network-resnet50-cnn-model-106e0ab9fa9e` (Accessed: 22 March 2024).

[8] Stojiljković, M. (2023) Stochastic gradient descent algorithm with python and NumPy, Real Python. Available at: `https://realpython.com/gradient-descent-algorithmpython` (Accessed: 22 March 2024).

[9] Vishwakarma, N. (2023) What is Adam Optimizer?, Analytics Vidhya. Available at: `https://www.analyticsvidhya.com/blog/2023/09/what-is-adam-optimizer/` (Accessed: 22 March 2024).

[10] Wang, C.-F. (2019) The vanishing gradient problem, Medium. Available at: `https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484` (Accessed: 22 March 2024).

[11] Xie, Z., Li, J. and Shi, H. (2019) 'A face recognition method based on CNN', Journal of Physics: Conference Series, 1395(1), p. 012006. doi:10.1088/1742-6596/1395/1/012006. ()