Journal of Computing and Electronic Information Management
ISSN: 2413-1660 | Vol. 16, No. 3, 2025

Comparative Analysis of Deep Learning Models for Handwritten Character Recognition Using the EMNIST Dataset

Hongfei Zhao

Henan University, Kaifeng, China

Abstract: This report presents a study where handwritten character recognition has been done using the EMNIST dataset that comprises 62 classes of letters and digits. The aim of this project is to classify the characters using three different deep learning models: Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), and CNN with Residual Connections. They were realized, deployed, and assessed to compare their outcomes. The findings portray that the CNN with Residual Connections archive the highest precision. This ability to capture complex features better than the MLP and traditional CNN models is the motivation for this approach. The easiest path in this work is finding a simple solution to the issues arising in the transcripts, such as overfitting and tuning hyperparameters. At the same time, recommendations for the resolution of these problems and prospective improvements are provided.

Keywords: Handwritten Recognition; EMNIST; Deep Learning; CNN; Residual Networks.

1. Introduction

Handwritten character recognition is a core computer vision issue that has numerous applications, for example, form automation, postal address identification, and document processing. Herein the EMNIST dataset, the expanded version of the original MNIST dataset, is utilized, which possesses 62 classes of handwritten characters where capital and small letters, as well as digits, are given. This dataset is not easy to distinguish since it consists of a great number of distinct classes and there is a considerable difference in individual handwriting styles.

The objective of this research work is to distinguish handwritten characters from the EMNIST dataset using three separate Deep Learning models: Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), and CNN with Residual Connections. These models have been chosen based on their past effectiveness in image data processing and because of their features-learning capability ability, which does not need additional feature engineering.

In the earlier handwritten character recognition techniques, machine learning methods such as SVM and k-NN were used and significantly depended on the features designed manually. Nevertheless, a new trend in machine learning, called deep learning, resulted in Convolutional Neural Networks as the most employed technique since such networks can deepen learning by recognizing inherent hierarchical features in the raw pixel data. Recently developed methods, such as the Residual Networks (ResNet), are enhancing the training of deeper structures by overcoming the vanishing gradient issue. As a result, they bring the way to more accurate and efficient learning in deep models.

In this study, the performance of the three models is implemented and compared on the EMNIST dataset. The simple MLP model used as baseline, followed by CNN and CNN with Residual Connections as a more advanced models that included sophisticated feature extraction procedures. The outcomes obtained from the recognition system indicate that the CNN with Residual Connections performed best, highlighting the strengths of deeper architectures and the fact that skip connections enhance learning performance. The difficulties met in the investigation, e.g., overfitting and hyperparameter tuning, are also dealt with, together with methods for improvement in ensuing studies.

2. Data Description and Preprocessing

2.1. Data Description

The dataset that is used to conduct this study is EMNIST by Class, which is the expanded version of MNIST dataset. Images in the EMNIST dataset are grayscale pictures with the resolution of 28 by 28 and contain 100,000 training and 20,000 testing samples. Such images, apart from being labeled in 62 classes, also consist of both upper-case and lower-case letters, as well as digits. Every image is characterized using a matrix where the pixel intensity values vary within the field from 0 to 255.

To better understand the dataset, we visualized several example images, shown below Fig.1

2.2. Data Exploration

Prior to training, the model was fed with the dataset and was analyzed by exploring its properties, such as distribution and characteristics. The class distribution plot showed an unbalanced situation where there are certain categories, particularly digits, which have more samples than others. This issue is likely to introduce bias to classifiers that over-represent the larger class in the second step even though it is not given any consideration in this project. Future research could address these approaches, for instance, class imposition or data augmentation, towards recognizing their role in attempting to resolve this imbalance.

The following Fig .2 bar plot shows the distribution of samples in the histogram: it contains frequency data for different classes.

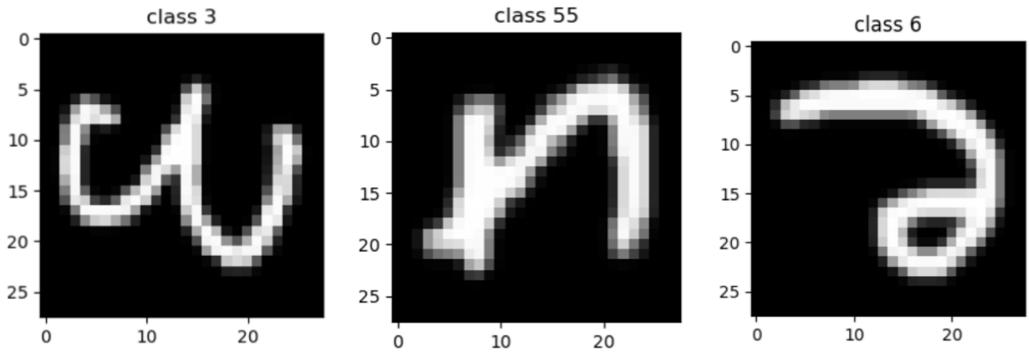

Figure 1. Sample visualization

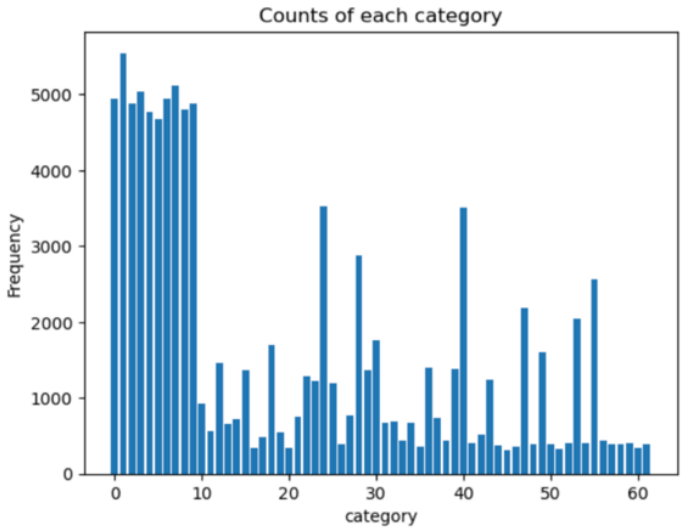

Figure 2. Distribution of samples

2.3. Preprocessing

To prepare the data for model training, the following preprocessing steps were applied:

1)Normalization: Normalization of data, namely image is the technique of putting the data in such a way to the model that it can be trained effectively. In this project, the Min-Max Normalization technique was used to fit the pixel values to the range [0, 1] based on the minimum and maximum values of the pixels. This technique made sure that the value of the pixels is normalized in the digital data distribution accordingly. Another simpler method, for example, which is often utilized for image data, involves dividing by 255, which is the highest possible value of the pixel (255 in case of grayscale images), where pixel values are divided directly. This method for achieving the same outcome (getting pixels into the range of [0, 1]) works without explicitly doing any min-max operations.

2)In the case of the EMNIST dataset, both Min-Max Normalization and division by 255 yield similar results because the pixel values are consistently between 0 and 255. For this project, Min-Max Normalization was applied to ensure flexibility in case of any potential variation in pixel values, though the simpler division by 255 would have been equally effective given the nature of the dataset.

3)Train -Validation Split: To ensure model performance and monitor for overfitting, the training data was further split into training and validation sets. A 90-10 split was applied, resulting in 90,000 training examples and 10,000 validation examples.

4)One-Hot Encoding: The categorical labels for the 62 classes were converted into one-hot encoded vectors. This step is necessary for training deep learning models, as it allows the models to output probabilities for each class during classification.

Reshaping for CNN Input: The form of the data used in Convolutional Neural Networks (CNNs) is specified. The EMNIST images were reshaped to include a channel dimension of 1, and the resultant shape was (28, 28, 1) per image. Consequently, CNN and CNN with Residual Connections can read the data properly.

2.4. Potential Challenges

Handwriting Variability: The sample is written in a way by a different set of individuals, which implies that huge proportions of the character styles are present in the sample. This might be a source of inaccuracy in understanding the general tendency of models, especially for the characters that have a high level of similarity in terms of their appearance (such as small l and capital I).

3. Methodology

3.1. Multilayer Perceptron (MLP):

3.1.1. Theory:

A multilayer Perceptron (MLP) model is an artificial network that consists of an input layer, an output layer, and multiple hidden layers. Each hidden layer is composed of multiple neurons and with activation functions.

3.1.2. Strengths and weaknesses:

The MLP model has a simple architecture and is easy to implement, making it suitable as a baseline for comparing

more complex models. Additionally, due to its simple structure, the MLP model has faster training speed and a smaller size, which can be useful in situations where computing resources are limited.

However, the MLP model has certain limitations when handling image data. Since each sample (a single image) becomes a row in the input matrix, the original image needs to be flattened into one-dimensional data for training. This means that the model discards the spatial relationships within the image data, which are often key components of the image's features.

Therefore, we use the MLP model as our baseline to compare the suitability of the other two models, CNN and CNN with residual connections, for this dataset.

3.1.3. Architecture and hyperparameters:

Table 1. Architecture of best MLP model. (After hyperparameter tuning).

Layer	Output Shape (n is batch size)	Param
Input layer	(n, 784)	0
Fully connected layer 1	(n, 200)	157,000
Fully connected layer 2	(n, 200)	40,200
Fully connected layer 3	(n, 200)	40,200
Dropout	(n, 200)	0
Output layer	(n, 62)	12,462

The total number of params for MLP model we used in this task is 249,862, the best learning rate is 0.001 and the best activation function for hidden layers is sigmoid.

As shown in the table above, the MLP model begins with an input layer that accepts data of shape (n, 784), where n is the batch size and 784 corresponds to a flattened image input. This is followed by three fully connected layers, each with 200 units, and the last layer has a dropout to prevent overfitting. The output layer produces predictions for 62 classes.

We used a fixed batch size of 64 and the Adam optimizer. The loss function consistently uses SoftMax combined with cross-entropy loss. The dropout rate was set to 0.3, training epochs are set to 10 and these hyperparameters were not included in the hyperparameter search process (due to time constraints).

For hyperparameter search, we used the RandomSearch class from the KerasTuner library [1]. This method performs random sampling during the search process. While it may not always find the optimal hyperparameter combination compared to grid search, it is more efficient and better suited for our high-dimensional hyperparameter space.

For the MLP model, we set up the hyperparameter search as follows:

1)Number of hidden layers: Search range is 1 to 3 layers, with a step size of 1. The number of hidden layers determines the complexity of the model. More complex model can capture more information among the data while increase the probability of overfitting.

2)Number of neurons in hidden layers: Search range is 50 to 200, with a step size of 50. Same as number of hidden layers, more neurons allow the model to learn more complex patterns in the data, but too many neurons can lead to overfitting, particularly with a small dataset. Too many layers or neurons can lead to longer training times and a higher risk of overfitting, so it's important to balance complexity and generalization ability by searching the best combination of number of hidden layers and neurons.

3)Activation function: Choose from ReLU, Sigmoid, and Tanh. Different activation functions determine the non-linear characteristics of hidden layer outputs, which affects the model's ability to learn non-linear patterns in the data. Though ReLU generally performs well due to its ability to deal with vanishing gradient. However, the vanishing gradient problem generally occurs in deeper models. Since our model has a simple structure, ReLU may not necessarily perform the best. Therefore, we also conducted a search for the activation function in this case.

4)Initial learning rate for Adam optimizer: Choose from 0.01, 0.005, and 0.0001. The learning rate determines the step size for parameter updates in each iteration. A high learning rate can cause the model to fail to converge, leading to oscillations or missing the optimal solution, while a low learning rate may slow down training or cause the model to get stuck in a local optimum.

3.2. Convolutional Neural Network (CNN):

3.2.1. Theory

A Convolutional Neural Network (CNN) is a class of deep learning models specifically designed for tasks involving grid-like data such as images. Unlike the MLP model, which flattens image data and discards spatial information, CNNs can preserve the spatial relationships between pixels using convolutional layers. These layers apply filters (kernels) to input images to extract features, such as edges, textures, and shapes, which are crucial for image recognition tasks.

3.2.2. Strengths and weaknesses:

CNNs have an advantage over MLPs when handling image data, as they maintain spatial relationships through convolutions and pooling, allowing the model to capture complex features more effectively. Additionally, CNNs require fewer parameters than MLPs for the same input size, as they reuse the same filter across the image through the convolutional process. This makes CNNs more efficient for large image datasets.

3.2.3. Architecture and hyperparameters:

Table 2. Architecture of best CNN model (After hyperparameter tuning).

Layer	Output Shape (n is batch size)	Param
Convolution layer 1	(n, 28, 28, 16)	160
Max pooling 1	(n, 14, 14, 16)	0
Convolution layer 2	(n, 14, 14, 64)	9,280

Table 3. Continued

Max pooling 2	(n, 7, 7, 64)	0
Convolution layer 3	(n, 7,7, 64)	36,928
Max pooling 3	(n, 3, 3, 64)	0
Flatten layer 1	(n, 576)	0
Fully connected layer 1	(n, 128)	73,856
Dropout	(n, 128)	0
Output layer	(n, 62)	7,998

The total number of parameters in the CNN model is 128,222 and the best learning rate is 0.001.

As shown in the table, CNN starts with three convolutional layers. The first convolutional layer has 16 filters, each of size 3x3, followed by a max-pooling layer to reduce the spatial dimensions. The second and third convolutional layers have 64 filters each, again followed by max-pooling layers to progressively reduce the dimensionality while preserving the

most important features. The feature maps are flattened and passed through two fully connected layers, with a dropout layer between them to help prevent overfitting.

We used the same hyperparameter search method as in the MLP model, focusing on key parameters that affect the CNN's performance:

1)Number of filters in each convolutional layer: We searched the number of filters from 16, 32 and 64 for the first layer, 32, 64 and 128 for the second layer and 64, 128 and 256 for the third layer. The number of filters determines the size of output channel. Fewer filters result in simpler models that may underfit the data, while more filters allow the model to capture more complex features but increase computational cost.

2)Initial learning rate for Adam optimizer: Choose from 0.01, 0.005, and 0.0001. This is as same as the MLP model.

3.3. Convolutional Neural Network (CNN) with Residual Connections:

3.3.1. Theory

A Convolutional Neural Network (CNN) with Residual Connections is an advanced deep learning model designed to solve the problem of vanishing gradients and improve the training of deeper networks. Residual connections add the input of a layer directly to the output of a deeper layer. We drew inspiration from the ResNet architecture [2] and implemented a simplified structure while retaining the core design of residual connections.

3.3.2. Strengths and weaknesses:

By using residual connections, the model can learn deeper representations without the risk of vanishing gradients. In addition, it simplifies the learning process, allowing the model to converge faster and more effectively, even in deeper networks. However, the inclusion of residual connections, combined with multiple convolutional layers, increases computational complexity, especially when dealing with large datasets or images.

3.3.3. Architecture and hyperparameters:

Table 4. Architecture of baseline CNN model with residual connections. (After hyperparameter tuning).

Layer	Output Shape (n is batch size)	Param
Convolution layer 1	(n, 28, 28, 32)	320
Max pooling 1	(n, 14, 14, 32)	0
Residual	(n, 14, 14, 64)	18,496
Max pooling 2	(n, 7, 7, 64)	0
Convolution layer 2	(n, 14, 14, 32)	9,248
Convolution layer 3	(n, 14, 14, 64)	2,112
Max pooling 3	(n, 7, 7, 64)	0
Add	(n, 7, 7, 64)	0
Max pooling 4	(n, 3, 3, 64)	0
Flatten layer 1	(n, 576)	0
Fully connected layer 1	(n, 128)	73,856
Dropout	(n, 128)	0
Output layer	(n, 62)	7,998

The total number of parameters in the CNN with residual connections model is 112,030 and the best learning rate is 0.001.

As shown in the table, the model starts with a convolutional layer with 32 filters (3x3) followed by max pooling to reduce the spatial dimensions. The residual connection is introduced after the first max-pooling operation, using a projection shortcut (1x1 convolution) to project the feature map dimensions to match the deeper layers (64 channels). This is followed by another max-pooling layer to further reduce spatial dimensions. The second and third convolutional layers continue processing, and the residual connection is applied after the third convolution. The final layers consist of max pooling, flattening the feature maps, and passing them through a fully connected layer with dropout to reduce overfitting.

The same hyperparameter search method was applied as in the previous CNN model, with key parameters being optimized as follows:

1)Number of filters in each convolutional layer: We searched for the number of filters from 16, 32, and 64 for the first layer, 64, 128, and 256 for the residual part, and 32, 64, and 128 for the second layer. The number of filters for the third convolutional layer is kept the same as the residual block to enable them to be added together.

2)Initial learning rate for Adam optimizer: Choose from 0.01, 0.005, and 0.0001. This is the same as the MLP model.

4. Results

4.1. Performance metrics:

The models were evaluated using several metrics, including accuracy, precision, recall, F1-score, training time, and number of parameters [3].

1)Accuracy: Represents the overall correctness of the model's predictions on the test dataset.

2)Training Time: The time required to train each model was recorded for comparison.

3)Number of Parameters: Total trainable parameters were noted to understand model complexity and memory requirements.

4)Interpretability: Each model's interpretability was discussed, especially regarding its capacity to capture spatial relationships in the input data.

Table 5. Result of different deep learning models.

Model	Accuracy	Precision	Recall	F1-Score	Time
MLP	0.81	0.79	0.81	0.79	265s
CNN	0.84	0.82	0.84	0.82	276s
ResNet	0.85	0.84	0.85	0.83	268s

Table 5 presents the performance outcomes of various deep learning models. To assess their effectiveness, we utilized evaluation metrics such as accuracy, precision, recall, and F1-score. Accuracy reflects how correct the model's predictions are overall, while precision measures the percentage of true positive predictions out of all predicted positives. Recall evaluates the model's ability to correctly identify actual positive instances, and the F1-score offers a balanced assessment by combining both precision and recall. Through a detailed analysis of these metrics, we can better understand the strengths and limitations of each model.

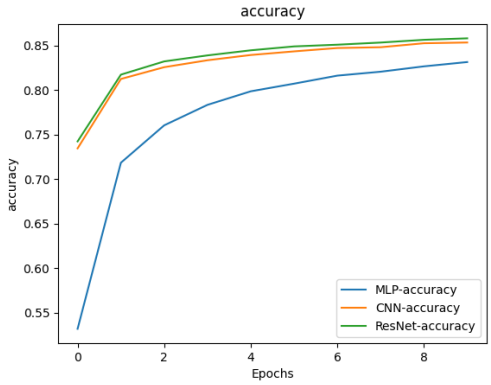

Figure 3. Accuracy of deep learning models

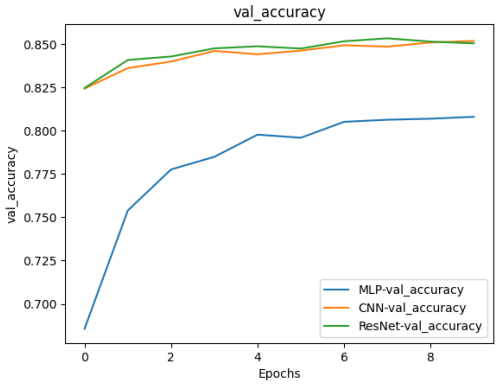

Figure 4. validation- accuracy of deep learning models

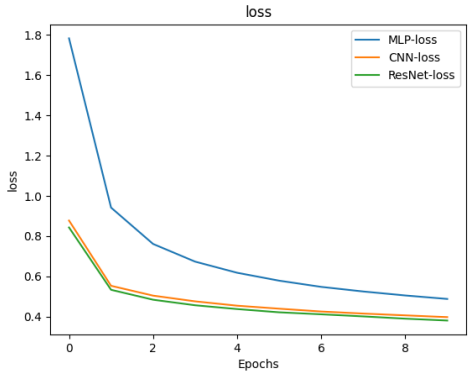

Figure 5. Loss of deep learning models

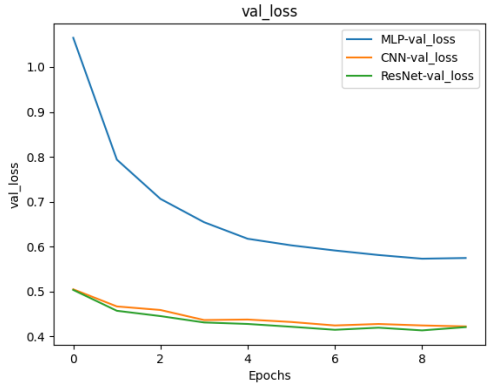

Figure 6. Validation- loss of deep learning models

Figure 3, the accuracy plot, visually displays the performance of various deep learning models in terms of accuracy throughout the training process. Figure 4, the validation accuracy plot, shows how accuracy evolves across multiple epochs on the validation set for each model. Figure 5 presents the loss plot, illustrating the loss values during training for the three models. Meanwhile, Figure 6, the validation loss plot, highlights the changes in loss values on the test set during evaluation. The loss function quantifies the difference between the predicted labels and the actual labels, reflecting the model's prediction errors.

4.2. Discussion

4.2.1. Accuracy and Loss

1)ResNet: This model achieved the highest accuracy (0.85) and exhibited the lowest loss, which indicates its superior learning capabilities due to deeper architecture and residual connections. Residual connections effectively mitigated the vanishing gradient problem, enabling the model to generalize well to the validation dataset.

2)CNN: The CNN model showed strong performance, achieving an accuracy of 0.84. However, some signs of overfitting were observed, as evidenced by fluctuating validation accuracy and loss in later epochs, suggesting the need for further regularization techniques.

3)MLP: The MLP model, despite being the simplest architecture, performed relatively well but lagged behind CNN and ResNet. Its inability to capture spatial relationships inherent in image data led to lower accuracy (0.81) and higher loss. However, the MLP was the fastest to train, making it suitable for scenarios with limited computational resources.

4.2.2. Model Complexity and Training Time:

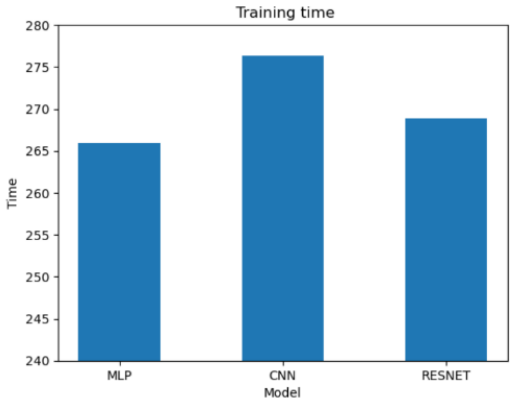

Figure 7. Training time of deep learning models

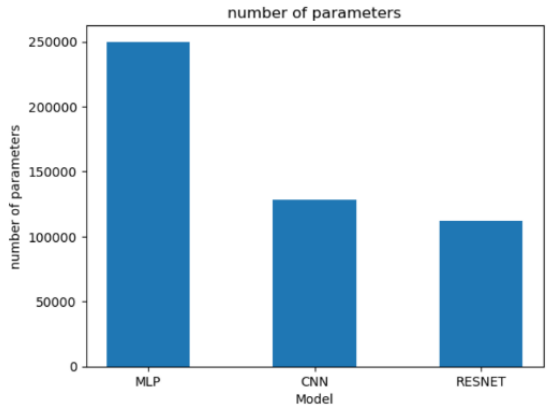

Figure 8. Number of parameters

The training time varied across models, with MLP being the fastest, followed by ResNet, and CNN taking the longest due to the high computational load of convolutional operations.

4.2.3. Number of Parameters:

1. The number of parameters and hyperparameters directly affected each model's complexity and generalization abilities. ResNet, with its higher parameter count and more hyperparameters, was more challenging to tune but ultimately achieved the best performance.
2. MLP was easier to tune due to its lower number of hyperparameters, but this simplicity limited its ability to generalize as well as CNN and ResNet.

4.2.4. Interpretability

1)MLP remains the most interpretable model, as it uses fully connected layers without complex feature extraction mechanisms.

2)CNN models are moderately interpretable due to the ability to visualize convolutional filters, but their complexity makes them harder to interpret at deeper layers.

3)ResNet, while the most complex, benefits from residual connections that allow for better understanding of how information flows through the network.

In this experiment, ResNet outperformed both MLP and CNN in terms of accuracy, loss, and generalization capabilities. Its deeper architecture, aided by residual connections, allowed it to effectively capture the complex features of the EMNIST dataset while minimizing loss. CNN, while performing well, showed some signs of overfitting and required the most time to train due to its complexity. MLP, though simpler and faster to train, lagged behind in accuracy and loss, demonstrating its limitations for image-based tasks.

5. Conclusion

Summary of Findings: This research has explored the performance of three neural architectures: MLP, CNN, and CNN with Residual connections to classify handwritten characters of the EMNIST dataset. The study showed that CNN with Residual Connections gave the maximum accuracy of 0.85, which was better than other models. The residual connections successfully fought with vanishing gradient problem; hence the model is able to learn low, middle, and high-level features. Moreover, the CNN model had good performance, but it had a few signs of overfitting. On the contrary, the MLP model, although it was simplistic, offered a good baseline.

Limitations: A few limitations were found that included computational issues during the study that limited the depth of analysis and exact definition of the hyperparameters. Apart from overfitting, which also appeared in the CNN model, the study also suggests the need for data augmentation and increasing the dropout techniques as other regularization approaches.

Future Work: Future projects could involve playing with even deeper architectures, improving hyperparameters using more advanced search algorithms, and using regularization techniques for avoiding overfitting. Further, we can improve models' robustness and generalizability if we use various datasets and include data augmentation techniques.

References

[1] T. O'Malley et al., KerasTuner. https://github.com/keras-team/keras-tuner, 2019. [Online]. Available: https://github.com/keras-team/keras-tuner

[2] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[3] Shiri F M, Perumal T, Mustapha N, et al. A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU[J]. arXiv preprint arXiv:2305.17473, 2023.

[4] Kadam S S, Adamuthe A C, Patil A B. CNN model for image classification on MNIST and fashion-MNIST dataset[J]. Journal of scientific research, 2020, 64(2): 374-384.