Performance Analysis of Convolutional Neural Networks and Multi-Layer Perceptron Networks on the Kuzushiji-MNIST Dataset

Patrick Craig
ECE Department
University of Florida
Gainesville, FL, USA
pcraig1@ufl.edu

James Overmeyer
ECE Department
University of Florida
Gainesville, FL, USA
jovermeyer@ufl.edu

ROLES

Patrick Craig: Patrick developed the flexible model architecture for the MLP network to enable parameter searching with a dictionary across various MLP models.
James Overmeyer: James implemented foundational studies on the effects of initialization strategies for MLP and CNN architectures and evaluated the impact of data augmentation on CNN performance. He contributed to setting up and analyzing these factors to optimize model accuracy and robustness.

***Abstract*—Convolutional Neural Networks (CNNs) and Multi-Layer Perceptrons (MLPs) are prominent architectures in image classification. CNNs leverage spatial awareness to capture patterns in image data. CNNs can be optimized through systematic experimentation on number of layers, filter sizes, activation functions, pooling techniques, or batch normalization to enhance feature extraction. MLPs, are traditionally suited to flat data, but can perform well in image classification with the right hyperparameters. Layer depth, layer width, activation functions, and learning rates can be used to identify an optimal configuration for an MLP. In addition to these hyperparameters, dataset augmentations can be utilized to augment the models through image transforms and masking. Our study compares CNN and MLP performance with different hyperparameters and offers insights into how different data augmentation strategies can enhance the performance of these models to affect generalization and accuracy in these networks.**

I. INTRODUCTION

Convolutional Neural Networks (CNNs) have become a standard approach in image classification tasks due to their ability to capture spatial and hierarchical patterns in data, enabling effective feature extraction and classification. A critical aspect of developing a high-performing CNN involves fine-tuning the architecture and hyperparameters to achieve optimal generalization on unseen data. Commonly, this involves systematic experimentation with various configurations, such as the number of convolutional layers, filter sizes, activation functions, and pooling strategies, to identify the model setup that best captures the unique characteristics of the dataset.

Additionally, techniques like batch normalization and pooling are often explored to enhance model stability, improve convergence rates, and prevent overfitting. Batch normalization works by normalizing the inputs to each layer, typically by scaling and shifting the activations based on the mini-batch statistics during training. This normalization reduces internal covariate shift—the tendency for the distribution of each layer's inputs to change throughout training—which allows for faster convergence and greater model stability, even when using higher learning rates. By doing so, batch normalization helps prevent issues such as vanishing or exploding gradients, common in deep networks, thereby enhancing model performance and generalization [1]. Pooling techniques, such as max pooling and average pooling, are used to down-sample feature maps and reduce the spatial dimensions of data in CNNs. Max pooling selects the largest value from a feature map region, preserving prominent features, while average pooling computes the average value, providing a smoother and more generalized representation. Pooling not only reduces the computational cost by minimizing data size but also imparts a degree of translational invariance to the network, helping the model become less sensitive to the exact position of features in the image. This can improve performance by allowing the model to focus on relevant features rather than on exact spatial locations pooling approaches have been studied extensively leading to the development of more advanced pooling approaches like overlapping pooling [2]. Complemen-

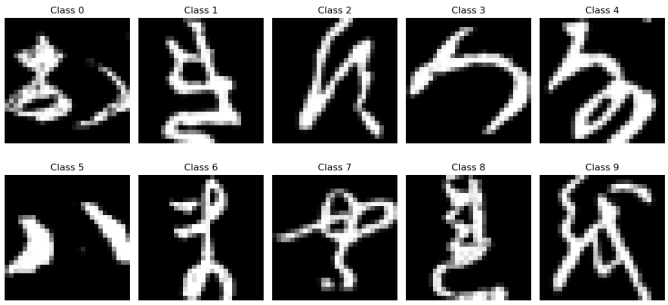


Fig. 1. Sample images for each class in the dataset

TABLE I
TEST PERFORMANCE OF AUGMENTATIONS ACROSS CNN AND MLP

Augmentation	CNN Test Accuracy	MLP Test Accuracy
Random Noise	96.47%	87.48%
Radial Noise	96.23%	88.25%
Flipping	92.59%	77.77%
Zooming	96.47%	86.28%
Masking	**96.93%**	**88.60%**
Random Noise + Zoom	**96.97%**	**88.43%**
Random Noise + Mask	82.53%	82.14%
None	96.70%	85.69%

tary data augmentation methods can further increase model robustness by diversifying the training data with modified samples, such as through transformations and noise injection. These transformations are quite simple and more complex augmentations such as image mixing, generative models, and feature augmentation have been studied extensively [3]. This combination of hyperparameter tuning and data augmentation ultimately enables CNNs to generalize better and achieve higher accuracy, even with limited or variable datasets. Here, we systematically evaluate these aspects to establish an optimized CNN configuration tailored to our classification task.

Multi-Layer Perceptrons (MLPs) are another foundational approach in machine learning, particularly useful in tasks where data can be represented in a flat, non-hierarchical format. Though traditionally less complex than Convolutional Neural Networks (CNNs), MLPs can still be powerful in classification tasks, especially for relatively simple datasets. MLPs operate by passing flattened input data through fully connected layers, where each neuron is connected to every neuron in the preceding layer, enabling comprehensive representation learning within the network. However, unlike CNNs, MLPs do not inherently capture spatial patterns in data, which can make them less suited for complex image tasks without extensive preprocessing or feature engineering.

Developing a high-performing MLP model requires careful tuning of architectural and training hyperparameters to achieve optimal classification results. Here, we systematically explore key configurations, such as the number of hidden layers (ranging from 0 to 3) and the number of neurons per layer (2000 for a single-layer model, 1000/1000 for two layers, or 500/1000/500 for three layers) [4]. Adjusting the number of layers and neurons allows us to investigate the balance between model complexity and generalization capacity; while deeper models with more neurons can potentially capture more complex patterns, they are also prone to overfitting, particularly on smaller datasets.

Another important aspect of MLP optimization is the choice of activation function, with rectified linear units (ReLU) and hyperbolic tangent (tanh) being the primary options under study. ReLU is widely preferred for its computational efficiency and ability to mitigate vanishing gradients, which aids in training deeper networks. However, tanh is often chosen for its ability to output both negative and positive values, which can result in better activation distribution and gradient flow for certain tasks. Both functions are evaluated to determine their impact on convergence and final model accuracy in our specific character classification context.

Additionally, learning rate tuning can play a crucial role in achieving effective convergence. We examine two initial learning rates, 0.001 and 0.1, to assess the influence of step size on model optimization and stability. A smaller learning rate generally results in more gradual convergence, reducing the risk of overshooting optima but requiring longer training times. Conversely, a larger learning rate can expedite convergence but may lead to unstable training or suboptimal performance. Through these systematic adjustments, we aim to identify the MLP configuration that offers the best balance of learning speed, model stability, and generalization.

To further enhance MLP performance and generalization, we examine the impact of data augmentation techniques on training. While data augmentation is often associated with CNNs, it can also benefit MLPs by enriching the dataset and reducing overfitting, especially in cases with limited data. Techniques such as noise injection, transformations, and feature-based augmentations introduce valuable variability, encouraging the MLP to learn more stable patterns rather than memorizing specific data points. By exploring these methods, we aim to improve the MLP's robustness and accuracy, ultimately providing insights into how data augmentation affects MLPs compared to CNNs in character classification.

II. METHODS

A. Training Implementation

Four different configurations of the CNN were tested, to evaluate the effects of batch normalization (True/False) and pooling (max/average). The implementation of pooling helps

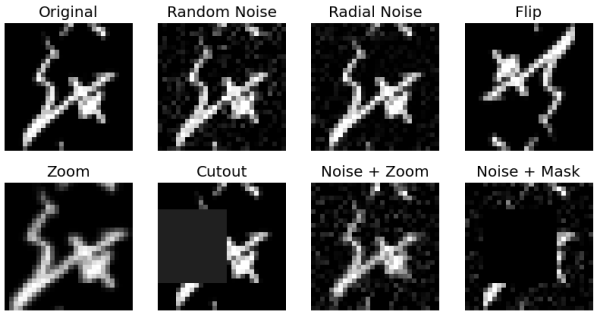


Fig. 2. Grayscale images of different augmentation approaches.

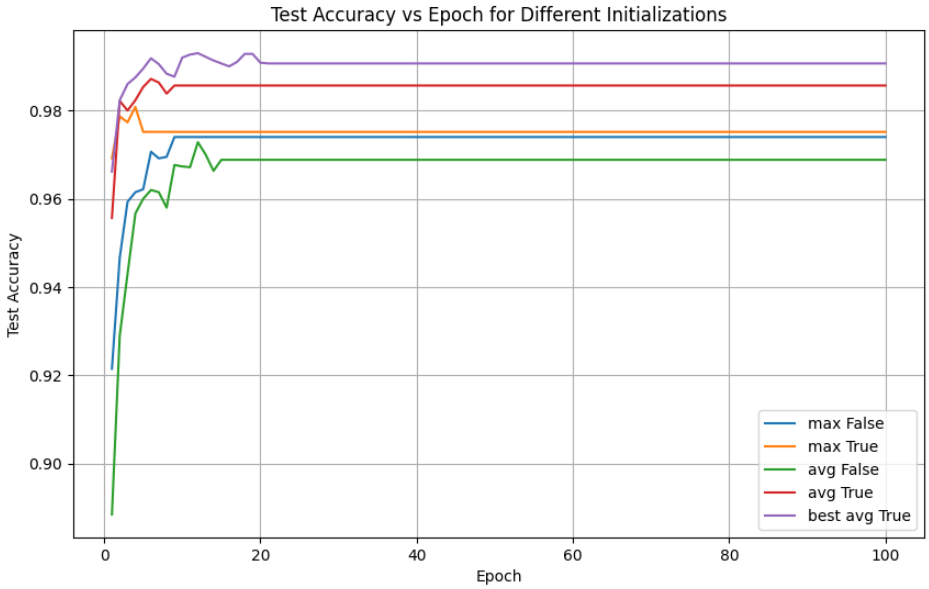


Fig. 3. Learning curves for different CNN implementations.

to reduce the spatial dimensions of the feature maps while retaining essential information, creating a more compact representation of the images. This approach can prevent overfitting by decreasing model complexity, allowing it to generalize better on unseen data. Overall the effect on performance of max and average pooling are not significantly different, but average pooling may show slightly enhance performance due to its ability to generalize. Meanwhile, the implementation of batch normalization standardizes the inputs to the network. This standardization assists the model with smoother training and better generalization. The initializations with batch normalization demonstrated greatly improved performance over the initializations without.

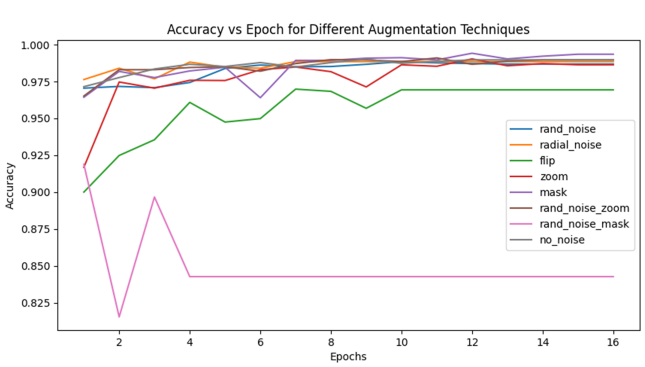
Additionally, twenty-four different configurations of the 2000 neuron MLP were tested. In addition to the layers and activation functions mentioned in the previous section, dropout rates of 0.0 and 0.05 and learning rates of 0.001 and 0.01 were tested to best evaluate model convergence and regularization levels. The splitting criteria also followed the CNN implementation above.

Prior to training the different model configurations, the training data was split into 90% training data and 10% validation data to help the models with generalization and prevent overfitting, this split was implemented using the sklearn library. Each configuration was trained for 20 epochs with a batch size of 32 and with early stopping after $3-5$ epochs of no improvement in the validation accuracy to prevent overfitting. The configurations were all compared through examination of their learning curves during training and their confusion matrix performances on the test set. The best performing configuration was then trained for 100 epochs with a batch size of 16 and with early stopping after 10 epochs of no improvement in the validation accuracy. The results were compiled as a confusion matrix and learning curve.

Radial noise, flipping, zooming, masking, and various combinations of these transforms were data augmentations for this implementation. The purpose of adding noise as random noise or radial noise is to enhance the generalization of the model and make it robust to small fluctuations in the dataset. The purpose of flipping is to increase the diversity of the dataset. The purpose of zooming is to train the model to recognize objects at different scales and to prevent overfitting by helping the model prevent overfitting by providing more perspectives of objects. Masking trains the model to make predictions even when parts of the object are missing making the model more robust. The best performing CNN and MLP configuration were utilized as the foundation for a study on the effects of different data augmentation techniques. In all, seven augmentation techniques were evaluated, random noise injection, radial noise injection, image flipping, zooming, masking, zooming with random noise, and no noise. Examples of these techniques are shown in Figure 2.

III. Results Across Convolutional Neural Networks

Preliminary experiments were employed to evaluate the best architecture for the CNN, and the best performing architecture that was investigated had 3 convolutional layers with 32, 64, and 64 filters, respectively. The CNN had 3 pooling layers, each with $3x3$ kernel sizes, ReLU activation, and 64 dense units. This CNN and the CNNs used throughout this paper were implemented using tensorflow keras.



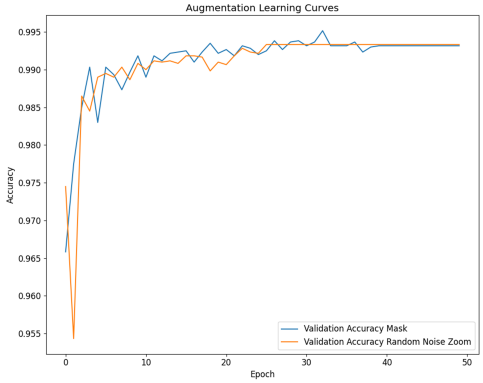


Fig. 4. Learning curves for different augmentation approaches.

A. *Learning Curves*

The learning curve illustrated in Figure 3 provides insight into the impact of pooling and batch normalization on the performance of the CNN. In preliminary testing, the combination of average pooling and batch normalization yielded the most favorable results, achieving a peak validation accuracy of 98.7%.

The best initialization was trained for 100 epochs with early stopping after 10 epochs of no improvement in the validation accuracy, and the training was stopped after 21 epochs with a maximum accuracy of 99.3% on the validation data, as can be confirmed from the learning curve. This model was evaluated on the test data set and achieved an accuracy of 97%. This represents a 2% improvement on the simple keras CNN presented in [5]. The confusion matrix of the best initialization is displayed in Figure 5.

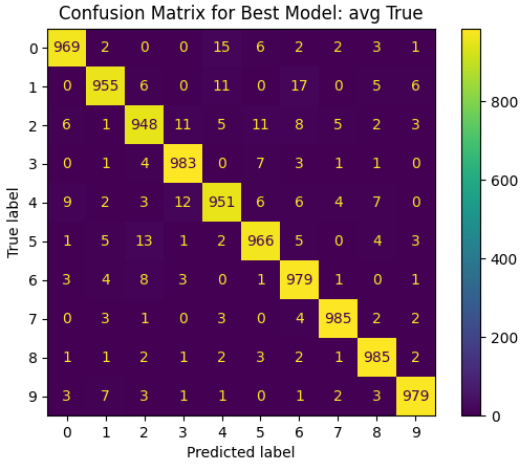


Fig. 5. Confusion matrix for best CNN implementation with average pooling and batch normalization.

Overall, none of the augmentation approaches demonstrated immediate massive improvements on the performance of the model, but due to time and processing constraints the study was not exhaustive. The learning curves can be found in Figure 4. Masking showed the best performance, achieving a validation accuracy of 99.35%. The performance of each augmentation approach on evaluating the test dataset is displayed in Table I. Both masking and random noise with zooming augmentations performed well, and were trained further for 50 epochs with early stopping after 10 epochs of no improvement and a smaller batch size of 16, and their learning curves are displayed in Figure 4. The masking augmentation achieved a test accuracy of 97.8% while the noise and zoom augmentation achieved a test accuracy of 97.28%. These accuracies represent 0.8% and 0.28% enhancements from the original model without any augmentations.

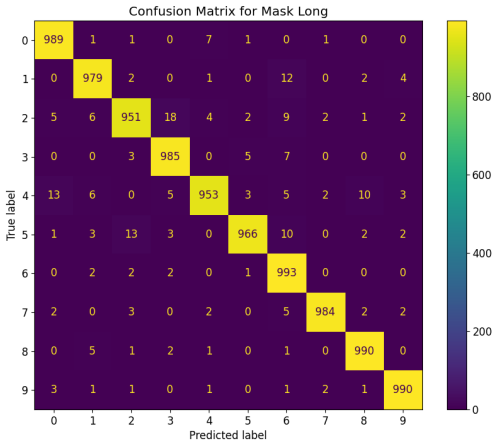


Fig. 6. Masking confusion matrix.

The confusion matrices of the masking and random noise with zooming augmentations can be found in Figures 6 and 11, respectively. Masking of random parts of the image was useful because it helped teach the model to make predictions based on incomplete information by making feature extractions from other regions. This augmentation approach encourages generalization and also simulates imperfections in the data. Masking is a common approach used for explaining black

box models, and one such example is random input sampling for explainability (RISE), where through random masking, the important features for classification can be identified [6]. The implementation of random zoom and noise as an augmentation helps the model become more robust to variations within the dataset. In general, it would be useful to study a larger assortment of augmentations and their various combinations to understand the best performing augmentation approach.

IV. Results across Multi-layer Perceptron Networks

A. Initial Search

In terms of selecting the best base model, of the 24 model implementations trained in the initial parameter search, the best model configuration was comprising of a 2 layer architecture with 1000 neurons per layer, a dropout rate of 0.5, learning rate of 0.001, and relu activation function achieved a model test accuracy of 92.48%. This was trained over 20 epochs with an early stopping criteria of 5 epochs. Over all the hyperparameter selecting models, most models implementing tanh had uncertain validation accuracy convergence to the best performing model weights. This is shown in Figure 7.

Most notably, the means of the validation accuracies were significantly influenced by the selection of the learning rate for the model. Models with a learning rate of 0.001 greatly outperformed models training on a 0.01 learning rate. This hints that the minima may have been overstepped in 0.01 learning rate architectures. In terms of other parameters, the models trained with relu activation functions, a single layer of 2000 neurons, and a dropout rate of 0.0 marginally outperformed their counterparts in the average. While marginal, these are essential changes that were implemented to increase the efficacy of the final model architecture. See Figure 10 for illustrations depicting this.

Model architecture types also effected the results, with implementations that are "wider" and containing more layers being slightly less accurate than architectures comprising 1 or 2 layers. Since MLPs can be interpreted as discriminative classifiers, it would make sense that having a deeper network may not be necessary to paritition the simplistic boundaries of the KMNIST dataset. The rationale for this is that each K-MNIST sample can be interpreted as a 3-D hyperplane, where the x and y pixel coordinates are the x and y-axis and the grayscale intensity value can be interpreted as the z-axis. This provides the task to the MLP to create discriminative boundaries within this 3-D space to draw many discriminative functions in this space. A wider network where there are less neurons per layer may not have enough discriminative functions to appropriately segment this space into distinct regions for the classifier to appropriately decide classes between regions.

B. Augmented Search

During the augmentation search, the base model achieved the highest accuracy on the validation set when trained with random noise+zoom (Figure 9), and masking augmentations applied to the test set. Specifically, the addition of random noise and masking to the MLP enhanced its robustness, likely by encouraging the model to focus on the essential features rather than noise or extraneous details in the data. This augmentation combination appears to improve the model's generalizability, suggesting that these perturbations introduce useful variance that aids in reducing overfitting while capturing complex patterns in the dataset. See Table I to compare these results across CNN and other MLP implementations.

C. Final Training Implementation

In the final training implementation, we focused on optimizing the performance of our models by leveraging the best base model alongside the most effective data augmentation strategies identified in the previous searches. Each of these augmented MLPs was subjected training across 100 epochs with early stopping criteria of 5 epochs. As seen in Figure 10, the best model with the random noise+zoom augmentation converged on a validation accuracy 95.6%, while conversely the model with the masking augmentation converged to 95.3%. However, in test, the final accuracies across the models were:

- Random Noise+zoom: 89.56%
- Masking: 89.15%

This showed that the noise+zoom augmentation slightly outperformed the masking augmentation.

Unfortunately, the results of the data augmentation strategies for the MLP were not as beneficial for the test set generalization like in the case in Section III. This showed a 3% loss in test generalization, and does not prove the be a viable way to increase the accuracy of an MLP.

V. Discussion

The results of this study highlight the effectiveness of different architectures and data augmentation techniques for Convolutional Neural Networks (CNNs) and Multi-Layer Perceptrons (MLPs) applied to the K-MNIST dataset. The CNN architecture, which included three convolutional layers and three pooling layers, achieved a peak validation accuracy of 99.3%. This performance improvement can be attributed to the use of batch normalization and average pooling, which likely facilitated stable training and enhanced convergence. The model reached a test accuracy of 97%, indicating that the architecture effectively captured key patterns in the data.

Among the augmentation techniques tested, masking proved to be the most effective, yielding a validation accuracy of 99.35%. This approach encouraged the model to learn to make predictions with incomplete information, promoting generalization. The model reached a test accuracy of 97.8%, a 0.8% improvement on the model without augmentation. The results suggest that effective training practices, including data augmentation, are critical for improving model robustness on unseen data.

For the MLP models, the optimal configuration a single-layer MLP with a learning rate of 0.001—achieved a test accuracy of 92.48%. The significant performance gap between learning rates of 0.001 and 0.01 indicates that selecting

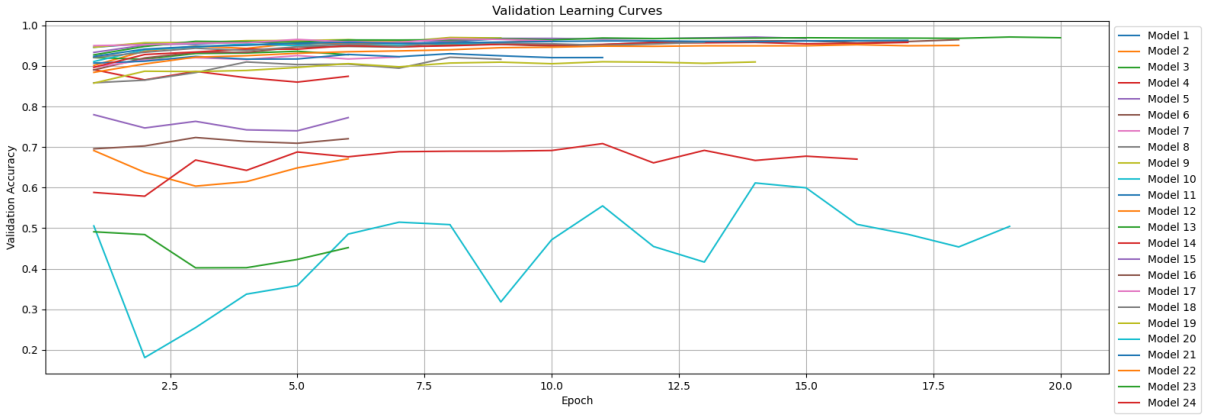


Fig. 7. This graph displays the learning curves of the various MLP architectures across the validation set.

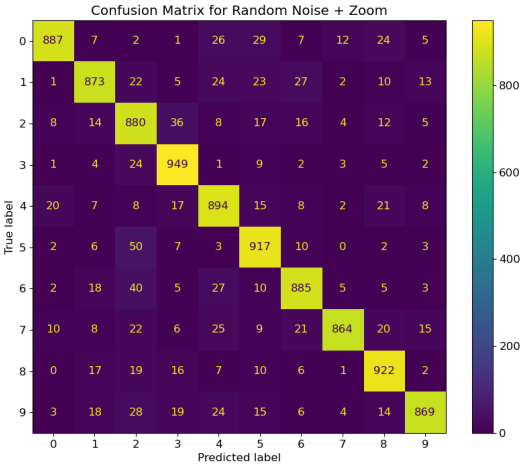


Fig. 8. MLP with random noise+zoom confusion matrix.

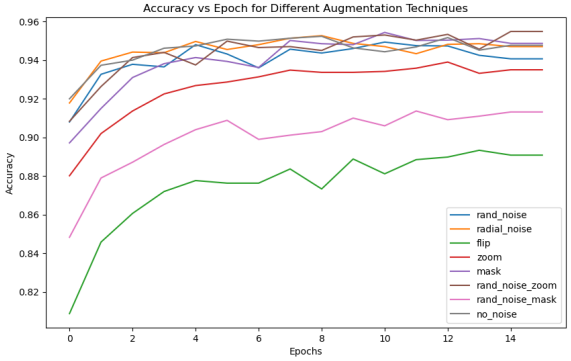


Fig. 9. MLP Augmentation validation accuracies

an appropriate learning rate is vital for model convergence. However, this peak accuracy was achieved with the base model and not the data augmentation steps we implemented across both CNNs and MLPs.

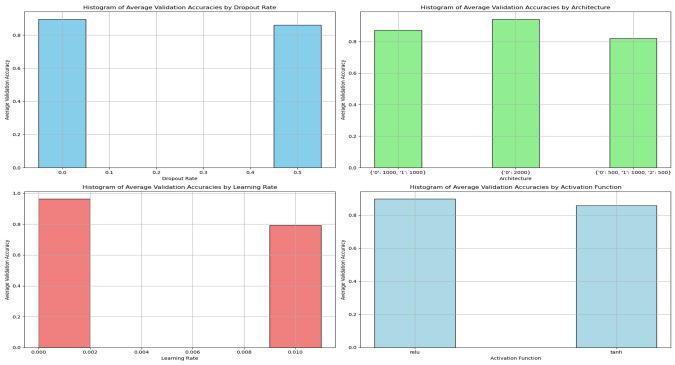
The application of data augmentation techniques, particularly for CNNs, demonstrated marginal benefits in model robustness. The combination of random noise and zoom augmentations, along with masking, improved the models' generalization capabilities by increasing the variance of the training data. The final results indicated that the random noise and zoom augmentation outperformed the masking strategy in terms of test accuracy, suggesting that simulating variability in the training data can enhance model performance. In summary, this study emphasizes the importance of architecture selection, hyperparameter tuning, and effective data augmentation in developing robust CNN and MLP models.

VI. Conclusions and Future Work

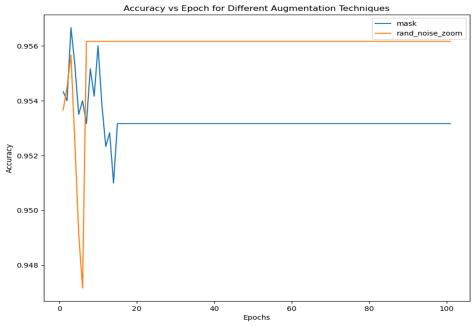
In this study, we systematically analyzed the performance of Convolutional Neural Networks (CNNs) and Multi-Layer Perceptrons (MLPs) on the Kuzushiji-MNIST dataset, highlighting the importance of hyperparameter choices and data augmentation techniques in enhancing model performance. Our experiments revealed that employing batch normalization and different pooling methods significantly improves the accuracy and robustness of CNNs, achieving a peak test accuracy of 97.8%.

One of the key insights from our findings is the effectiveness of data augmentation, especially the masking and noise injection techniques, in boosting model generalization. The masking approach helped the model to make predictions based on incomplete information, promoting a better understanding of relevant features within the dataset. While our research demonstrated promising results with the techniques explored, we acknowledge that further investigation is needed to comprehensively evaluate a wider array of augmentation strategies and their combinations.

Looking ahead, we are interested in expanding this work by incorporating Variational Autoencoders (VAEs) for synthetic data generation. VAEs have shown great potential in gener-



(a)



(b)

Fig. 10. Validation accuracies across MLP with masking and noise + zoom

ating diverse datasets, which could be beneficial for training robust machine learning models, especially in scenarios where data scarcity and lack of variance is an issue. Moreover, we could explore ensemble methods like bagging and boosting to further enhance our model performance. These techniques could help reduce variance and improve predictions by combining the outputs of multiple models, leading to more reliable and accurate classification outcomes.

In summary, our work establishes mediocre training results across CNNs and MLP, and assesses the capabilities for data augmentation across various CNN and MLP arcitectures. Future work will aim to utilize generative models for synthetic data generation and ensemble learning methods to achieve higher accuracies across these models.

REFERENCES

[1] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, May 2017. [Online]. Available: https://doi.org/10.1145/3065386

[3] S. Yang, W. Xiao, M. Zhang, S. Guo, J. Zhao, and F. Shen, "Image data augmentation for deep learning: A survey," 2023. [Online]. Available: https://arxiv.org/abs/2204.08610

[4] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural Computation*, vol. 22, no. 12, p. 3207–3220, Dec. 2010. [Online]. Available: http://dx.doi.org/10.1162/NECO$_{a}$0052

[5] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep learning for classical japanese literature," *CoRR*, vol. abs/1812.01718, 2018. [Online]. Available: http://arxiv.org/abs/1812.01718

[6] V. Petsiuk, A. Das, and K. Saenko, "RISE: randomized input sampling for explanation of black-box models," *CoRR*, vol. abs/1806.07421, 2018. [Online]. Available: http://arxiv.org/abs/1806.07421

APPENDIX

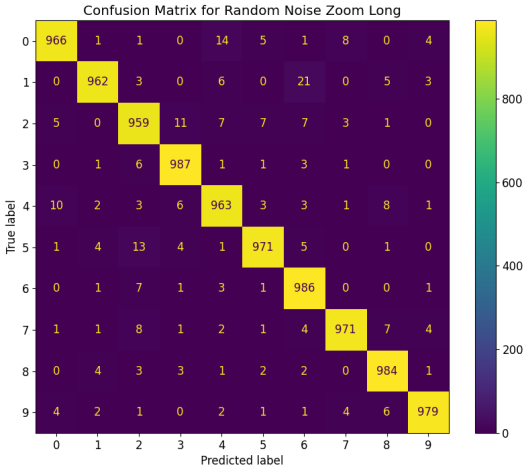


Fig. 11. Random noise with zoom confusion matrix.