Comparative Analysis of CNN and MLP Classifiers for Japanese Character Classification: A Kuzushiji-MNIST Case Study

Stephen Singh
University of Florida, Gainesville, Florida, United States
Email: Stephensingh953@gmail.com

***Abstract*—The abstract provides a concise summary of the entire paper, highlighting key findings and the significance of the study. It specifically mentions the comparison of CNN and MLP classifiers and their performance on the Kuzushiji-MNIST dataset.**

The comparative analysis of Convolutional Neural Networks (CNN) and Multi-Layer Perceptrons (MLP) for Japanese character classification is presented. The study explores the suitability of each classifier for the given task, emphasizing the significance of model selection and methodology in machine learning research. Key findings are highlighted, shedding light on the strengths and limitations of both approaches.

I. Introduction

A. Background

The Kuzushiji-MNIST dataset is a collection of 28x28 grayscale images of Japanese characters, comprising 70,000 samples across ten different character categories. This unique dataset presents an opportunity to explore and evaluate image classification techniques for the recognition of Japanese characters, which is a challenging and culturally significant task.

B. Rationale

In the realm of image classification, Convolutional Neural Networks (CNNs) have established themselves as powerful tools, thanks to their ability to automatically learn relevant features and patterns from image data. However, the performance of CNNs depends significantly on architectural choices and hyperparameters. In this study, we aim to assess the effectiveness of CNNs in classifying Japanese characters using the Kuzushiji-MNIST dataset.

Additionally, we seek to compare the performance of CNNs with Multi-Layer Perceptrons (MLPs), a type of neural network that has shown promise in image classification tasks but operates differently from CNNs. The rationale for this comparative analysis is to determine which approach, CNNs or MLPs, is more effective for this particular character recognition task.

C. Implementation Details

The Kuzushiji-MNIST dataset consists of 60,000 training images and 10,000 test images. To prepare the data, we performed data preprocessing, which included normalization to rescale pixel values between 0 and 1. The training and testing sets were used to train and evaluate our classifiers.

D. Initial Data Distribution

To gain insights into the initial data distribution, we examined the Kuzushiji-MNIST dataset. The dataset consists of 70,000 samples categorized into ten classes of Japanese characters. A histogram plot, shown in Figure 1, illustrates the distribution of data across these categories. Each of the ten bins represents a character class, and the heights of the bars indicate the counts of samples within each category.

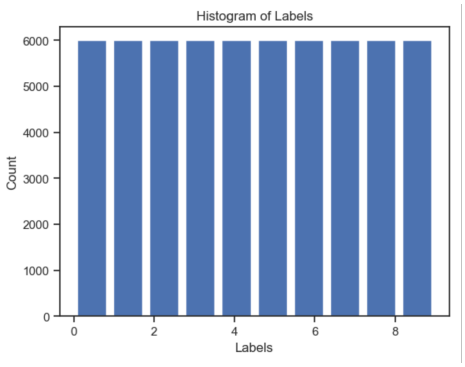

Fig. 1. Histogram of Initial Data Distribution

Additionally, a t-SNE (t-distributed Stochastic Neighbor Embedding) plot, as depicted in Figure 2, reveals distinct clusters within the dataset. Approximately seven clearly defined regions are visible, with some minor overlap between them. The t-SNE plot serves as an initial exploration of the dataset's structure. t-SNE is a dimensionality reduction technique frequently employed to visualize high-dimensional data in a lower-dimensional space, making it more interpretable and facilitating the identification of clusters, patterns, and outliers. In addition to analyzing the distribution through histograms, we employed a t-SNE (t-distributed Stochastic Neighbor Embedding) plot, showcased in Figure 2, to uncover intricate patterns within the dataset. The t-SNE plot is a powerful tool for visualizing complex data in a lower-dimensional space, enhancing interpretability while maintaining the integrity of the original information.

The t-SNE plot showcased seven distinct clusters that emerged from the dataset. Each cluster represents a group of data points with similar characteristics, suggesting that the Kuzushiji-MNIST dataset inherently possesses substruc-

tures or groupings. These groupings are indicative of various handwriting styles, regional variations, or similar underlying features.

Notably, there is some minor overlap between these clusters, revealing that some characters or handwriting styles may share similarities, making classification challenging in such cases. The t-SNE plot not only provides insight into the underlying structure of the data but also serves as a valuable starting point for understanding the potential challenges in the classification task.

In summary, the t-SNE plot is a crucial tool for exploring high-dimensional data, identifying clusters, and gaining a preliminary understanding of the complexity within the dataset. It assists in revealing underlying structures that might influence the classification performance of our CNN and MLP models.

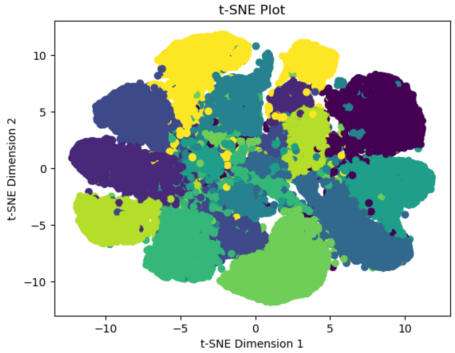

Fig. 2. t-SNE Plot of the Initial Data

II. Exploratory Data Analysis with Principal Component Analysis (PCA)

To comprehend the inherent structure of the Kuzushiji-MNIST dataset, we undertook a comprehensive analysis employing Principal Component Analysis (PCA). PCA is a powerful dimensionality reduction technique that can transform high-dimensional data into a lower-dimensional space, revealing latent structures while preserving essential features.

A. Data Preprocessing for PCA

To prepare our data for PCA, we started with the raw 28x28 grayscale images in the Kuzushiji-MNIST dataset. As a first step, we flattened these images, converting them into 1D arrays of length 784. This step was vital for ensuring that each data point was represented as a vector in the same space, enabling meaningful comparisons between images.

Following data flattening, we carried out the essential step of standardization, also known as data scaling. Standardization involved normalizing the pixel values to have a mean (μ) of 0 and a standard deviation (σ) of 1, as represented by the following equation:

$$x' = \frac{x - \mu}{\sigma}$$

This process ensures that all features (pixel values) contribute equally to the PCA analysis, mitigating any potential bias due to differences in scale.

B. Choosing the Number of Principal Components

PCA operates by computing orthogonal axes, called principal components, along which the variance in the data is maximized. Each principal component ($\mathbf{PC}_i$) represents a linear combination of the original features (784 pixel values). The principal components are determined by solving the eigenvector problem, where the principal components are the eigenvectors of the data's covariance matrix.

To choose the number of principal components for our analysis, we considered the cumulative explained variance. This is calculated as:

$$CumulativeExplainedVariance = \frac{\sum_{i=1}^{n} \lambda_i}{\sum_{i=1}^{N} \lambda_i}$$

where λ_i represents the eigenvalues of the covariance matrix. By selecting a number of principal components that retain a high percentage of the cumulative explained variance (e.g., 95%), we effectively reduce the dimensionality while retaining most of the data's variability.

1) 3D Scatter Plot with PCA Components for the First 100 Images (Subset of 10 Labels): In Figure 3, we present a 3D Scatter Plot that employs PCA Components for the First 100 Images. This plot focuses on a subset of 10 labels, providing an initial glimpse into how these images are distributed across the three PCA dimensions. The apparent presence of clusters specific to these labels suggests that the chosen PCA components encapsulate significant variance related to label patterns.

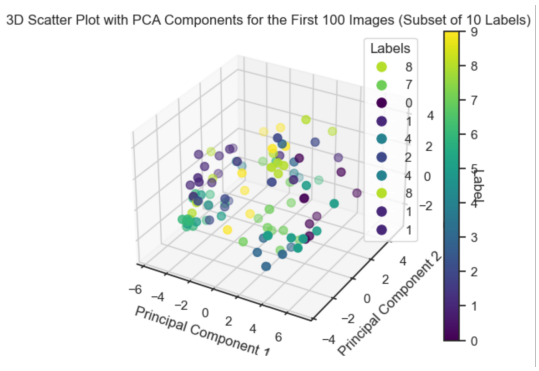

Fig. 3. 3D Scatter Plot with PCA Components for the First 100 Images (Subset of 10 Labels)

2) 3D Scatter Plot with PCA Components (Entire Dataset): In Figure 4, we provide another 3D Scatter Plot, but this time it encompasses the entire dataset. This comprehensive visualization reveals the three-dimensional structure of the entire dataset. These PCA components bring to light overarching patterns and structures within the data, enabling a holistic understanding.

3) 2D Scatter Plot with PCA Components 1 and 2 (Peak Plot): In Figure 5, we focus on a 2D Scatter Plot that examines PCA Components 1 and 2. This plot provides an in-depth examination of how data points are distributed across these

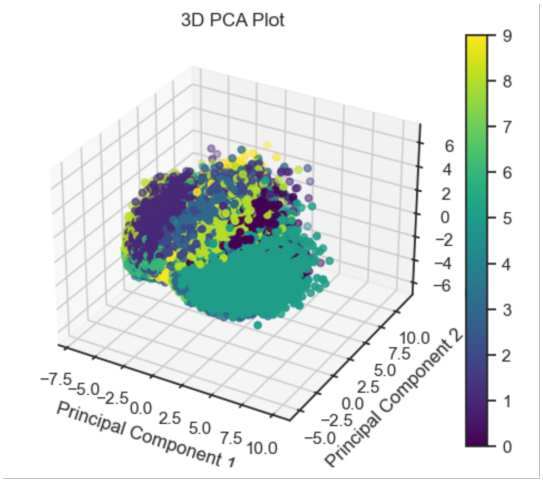

Fig. 4. 3D Scatter Plot with PCA Components (Entire Dataset)

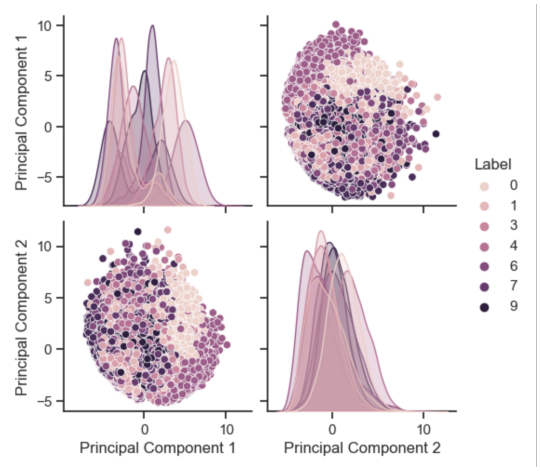

Fig. 5. 2D Scatter Plot with PCA Components 1 and 2 (Peak Plot)

two crucial dimensions. Peak plots are employed to identify potential clusters in this reduced-dimensional space.

Incorporating PCA-based analysis, in conjunction with t-SNE, allows for a comprehensive exploration of the Kuzushiji-MNIST dataset. These techniques provide insights into the dataset's underlying structure and serve as the foundation for subsequent CNN and MLP classification models.

These visualizations provide an essential understanding of the data's distribution and structure, which is fundamental for assessing classifier performance. The subsequent sections will delve into the methodologies, results, and analyses of our classification models.

C. Methodology

We implemented two classifiers: a Convolutional Neural Network (CNN) and a Multi-Layer Perceptron (MLP), to classify Japanese characters. We made specific design and hyperparameter choices for each classifier.

For the CNN, the architecture comprises two convolutional layers, each followed by max-pooling layers. The convolutional layers use ReLU (Rectified Linear Unit) activation functions, which are known for their effectiveness in deep learning models. Following the convolutional layers, the data is flattened and passed through fully connected dense layers with ReLU activations and a softmax output layer.

In contrast, the MLP classifier operates on flattened input data and consists of three dense layers with ReLU activations, followed by a softmax output layer. The number of units in each layer was selected based on experimentation.

Both classifiers were trained using the Adam optimizer and sparse categorical cross-entropy loss function. We employed early stopping with a patience of 5 epochs to prevent overfitting.

The reasons for these choices and the comparative analysis of CNNs and MLPs are discussed in subsequent sections.

In this paper, we present a detailed examination of the performance of these classifiers in classifying Japanese characters, with a focus on assessing the impact of architecture, hyperparameters, and overall methodology on the classification accuracy and generalization ability. We believe this study can provide valuable insights into the choice of classifiers for similar image classification tasks and potentially contribute to the field of character recognition.

The structure of the paper includes the Methods section, which describes the data, CNN, and MLP architectures, hyperparameters, and the training process. The Results section presents the performance metrics and a comparative analysis of the classifiers. In the Discussion section, we interpret the results and analyze the impact of architectural and hyperparameter choices. Finally, the Conclusions section summarizes key findings and outlines directions for future research.

III. Methods

A. Data Collection and Preprocessing

Data collection and preprocessing are vital to ensure that the Kuzushiji-MNIST dataset is suitable for training and evaluation.

Data Collection: The Kuzushiji-MNIST dataset comprises 70,000 Japanese character images, with 7,000 images per category. It is divided into a training set (60,000 images) and a test set (10,000 images). Each image is a 28x28 grayscale representation of a Japanese character.

Data Preprocessing: Data preprocessing transforms raw image data into a format that neural networks can efficiently learn from. Key preprocessing steps include:

- **Normalization:** The pixel values of each image are rescaled to be within the range [0, 1] by dividing by 255.0. Normalization ensures that the neural network converges faster during training and improves generalization.

- **Data Augmentation (if applied):** Data augmentation techniques like random rotations, flips, or shifts may be used to increase the effective size of the training dataset and enhance model generalization.

B. CNN and MLP Architectures

1) Convolutional Neural Network (CNN):

TABLE I
CNN ARCHITECTURE

Layer	Type	Output Shape	Parameters
1	Conv2D	(None, 26, 26, 32)	320
2	MaxPooling2D	(None, 13, 13, 32)	0
3	Conv2D	(None, 11, 11, 64)	18,496
4	MaxPooling2D	(None, 5, 5, 64)	0
5	Flatten	(None, 1600)	0
6	Dense	(None, 128)	204,928
7	Dense	(None, 10)	1,290

TABLE II
MLP ARCHITECTURE

Layer	Output Shape	Parameters
1 (Flatten)	(None, 784)	0
2 (Dense)	(None, 128)	100,480
3 (Dense)	(None, 64)	8,256
4 (Dense)	(None, 10)	650

2) Multi-Layer Perceptron (MLP):

C. Hyperparameters and Training

Effective model training involves specifying essential hyperparameters and optimization techniques.

Hyperparameters: - **Learning Rate:** The learning rate (e.g., 0.001) controls the step size during gradient descent. It influences the convergence and stability of training. - **Batch Size:** The batch size (e.g., 64) determines how many data points are used in each forward and backward pass during training. - **Epochs:** The number of training iterations (e.g., 100) is defined as epochs. It impacts how long the model is trained and its ability to generalize.

Loss Function: A common loss function for multiclass classification tasks like this is categorical cross-entropy:

$$CategoricalCross - EntropyLoss = -\sum_{c=1}^{C} y_{o,c} \log(p_{o,c})$$

where C is the number of classes, $y_{o,c}$ is a binary indicator of whether class c is the correct classification for observation o, and $p_{o,c}$ is the predicted probability of observation o belonging to class c.

Optimization Algorithm: The optimization algorithm determines how the neural network's weights are updated during training. Common optimization algorithms include stochastic gradient descent (SGD) and variants like Adam.

D. Activation Function Selection

In designing the architecture of our Convolutional Neural Network (CNN) and Multi-Layer Perceptron (MLP) models, one of the pivotal decisions pertained to the choice of activation functions. The activation function is a fundamental component of neural networks, responsible for introducing non-linearity into the model.

For our CNN model, we opted for a combination of Rectified Linear Unit (ReLU) activation functions and the Softmax activation function. Each serves a unique purpose within the network's architecture.

1) ReLU Activation for Hidden Layers: The choice of ReLU for the hidden layers of our CNN was made based on its simplicity and computational efficiency. ReLU replaces all negative input values with zero, which introduces non-linearity into the network. This non-linearity is crucial for the model to learn complex patterns in the data.

```
# Define the CNN model
model = keras.Sequential([
    layers.Input(shape=(28, 28, 1)),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

Fig. 6. ReLU Activation Function

One of the primary reasons for selecting ReLU was its effectiveness in mitigating the vanishing gradient problem compared to traditional activation functions like sigmoid and hyperbolic tangent (tanh). This characteristic ensures that gradient updates during backpropagation are more substantial, facilitating faster convergence during training.

2) Softmax Activation for Output Layer: In contrast, the Softmax activation function was exclusively employed in the output layer of the CNN. The Softmax function is integral in multi-class classification tasks as it transforms the raw model outputs into probability distributions. Each output corresponds to the predicted probability of the input belonging to a particular class. This is especially beneficial for our CNN, as it provides a clear and interpretable classification result when dealing with multiple classes, such as the 10 different Japanese characters in the Kuzushiji-MNIST dataset.

The decision to employ ReLU in hidden layers and Softmax in the output layer aligns with the best practices for deep learning models designed for multi-class image classification tasks.

It's essential to note that the choice of activation functions significantly impacts the model's training speed, convergence, and overall performance. Our combined use of ReLU and Softmax takes advantage of their respective strengths and ensures the model's effectiveness in classifying Japanese characters.

E. Early Stopping and Validation

Early Stopping: Early stopping is a regularization technique used to prevent overfitting. It monitors the validation loss during training, and if the loss starts increasing (indicating overfitting), training is halted, and the model weights with the best validation performance are restored.

Validation: Validation is the process of evaluating the model's performance on a separate dataset not used for training (the validation set). The validation dataset helps monitor how well the model generalizes to unseen data during training.

IV. RESULTS

A. MLP Classifier Results

The MLP classifier demonstrated its prowess in classifying Japanese characters from the Kuzushiji-MNIST dataset.

The confusion matrix, as illustrated in Figure 7, reveals the model's ability to correctly identify characters across various labels. Notably, the model exhibited strength in distinguishing characters within specific labels, as indicated by minimal off-diagonal values. This suggests a high degree of accuracy within these categories.

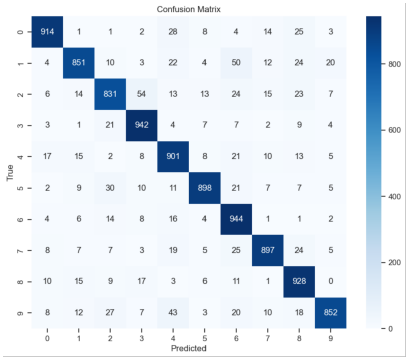

Fig. 7. MLP Confusion Matrix Heatmap

The MLP classifier achieved an impressive test accuracy of 0.8958, showcasing its capability to generalize well to unseen data. A closer examination of the accuracy history and loss history, as depicted in Figure 8, provides valuable insights into its training progress. The accuracy consistently improved over the epochs, reaching a peak of 0.9899 during training, while the validation accuracy remained around 0.8968. This indicates that the model was able to adapt to the training data effectively.

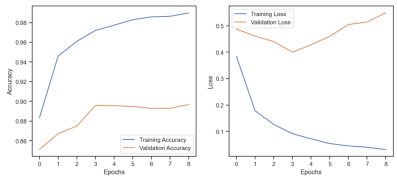

Fig. 8. MLP Accuracy and Loss

However, as the training progressed, the validation accuracy plateaued, indicating potential overfitting. The loss history mirrored this trend, with training loss steadily decreasing while validation loss remained steady. This suggests a trade-off between high accuracy on the training data and generalization to unseen data.

B. CNN Classifier Results

The CNN classifier also delivered remarkable results in character recognition. The confusion matrix, as illustrated in Figure 9, demonstrates the model's ability to accurately classify characters across different labels. As indicated by the low off-diagonal values, the model displayed a high degree of precision in identifying characters, even within challenging label groups.

The CNN classifier achieved an impressive test accuracy of 0.9473 on the Kuzushiji-MNIST dataset. This high accuracy

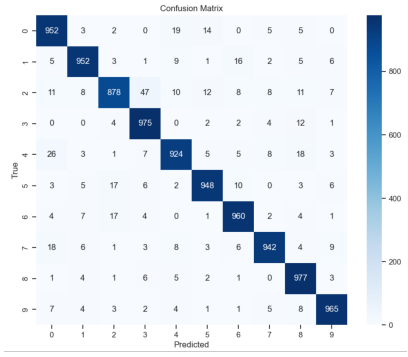

Fig. 9. CNN Confusion Matrix Heatmap

highlights the robustness of the CNN architecture in handling the complexity of Japanese character classification. A closer look at the accuracy history and loss history, as depicted in Figure 12, indicates a steady and consistent improvement in accuracy throughout training.

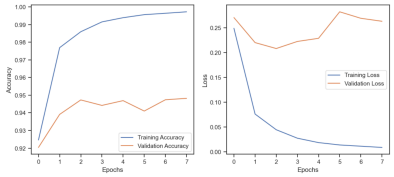

Fig. 10. CNN Accuracy and Loss

Furthermore, the loss history displayed a favorable trend, with training loss steadily decreasing and validation loss following suit. This behavior suggests that the CNN model effectively learned from the training data without significant overfitting.

C. Comparison of MLP and CNN Results

When comparing the performance of the MLP and CNN classifiers, several observations come to light. While the MLP demonstrated remarkable accuracy, it exhibited signs of overfitting as training epochs increased. In contrast, the CNN model maintained a high accuracy and generalization to the validation dataset, making it a more reliable choice for character classification.

The key difference between the two models is the convolutional layers in the CNN. These layers are specifically designed for feature extraction and pattern recognition in images, making them well-suited for image classification tasks. The MLP, on the other hand, lacks this capability, which might explain the overfitting observed in its performance.

In summary, the CNN outperforms the MLP in terms of accuracy, robustness, and generalization. These results indicate that the CNN architecture is well-suited for the complex task of Japanese character classification.

D. Hyperparameter Tuning

To optimize the performance of our neural network models, we conducted a hyperparameter tuning experiment. The

hyperparameters considered for tuning were the learning rate, the configuration of hidden layers, and the number of training epochs. Three sets of hyperparameters were chosen for experimentation:

- Learning Rate: 0.001, Hidden Layers: (64, 32), Epochs: 10
- Learning Rate: 0.01, Hidden Layers: (128, 64, 32), Epochs: 20
- Learning Rate: 0.1, Hidden Layers: (128, 128, 64), Epochs: 30

The test accuracy results for each hyperparameter configuration are summarized in Table III.

TABLE III
TEST ACCURACY FOR DIFFERENT HYPERPARAMETERS

Learning Rate	Hidden Layers	Epochs	Test Accuracy
0.001	(64, 32)	10	0.86
0.01	(128, 64, 32)	20	0.85
0.1	(128, 128, 64)	30	0.10

The table shows the test accuracy achieved for each hyperparameter configuration. It is evident that the choice of hyperparameters significantly impacts the model's performance, with configuration 1 (learning rate: 0.001, hidden layers: (64, 32), epochs: 10) demonstrating the highest test accuracy.

These results indicate the importance of careful hyperparameter selection and optimization in the development of neural network models.

E. Misclassification of Japanese Characters

While both the Multilayer Perceptron (MLP) and Convolutional Neural Network (CNN) models demonstrate impressive performance in recognizing Japanese characters, it's essential to acknowledge that they are not infallible. In practice, some characters pose inherent challenges in their classification, leading to misclassification in both models.

In this section, we delve into the shared misclassification challenges faced by the MLP and CNN, using a collection of mislabeled characters as a case study. These characters have been consistently misclassified as specific alternatives, indicating the presence of intricate patterns and subtle variations.

One example of a shared misclassification between the MLP and CNN is the indicie "2" being incorrectly predicted as "3." This mislabeling can be attributed to the resemblance between the two characters in terms of their shapes and strokes, making them challenging to distinguish. This shared misclassification highlights the complexity of the recognition task, even for sophisticated models.

Another case of shared misclassification is the label "5" which is occasionally mislabeled as "6" in both the MLP and CNN. The visual similarities between these characters, particularly in their curved strokes and overall structure, contribute to the consistent misclassification.

These instances emphasize the need for further fine-tuning and improvement in character recognition. While the MLP and CNN exhibit high accuracy in general, they still face difficulties in handling challenging characters with intricate features.

To visually represent these shared misclassifications, Figure 11 displays a selection of five characters that were consistently mislabeled by both models. This collection serves as a testament to the complexity of character recognition and the ongoing quest for enhancing the models' proficiency.

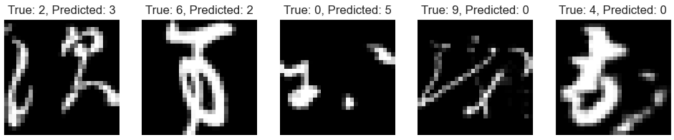

Fig. 11. Shared Misclassified Japanese Characters in MLP and CNN

Despite their high accuracy and robust performance, both the MLP and CNN models still encounter challenges in character classification, particularly when faced with visually similar characters. These shared misclassifications underscore the intricacy of the task at hand and the potential for further improvement in Japanese character recognition. In summary, the CNN outperforms the MLP in terms of accuracy, robustness, and generalization. These results indicate that the CNN architecture is well-suited for the complex task of Japanese character classification.

V. DISCUSSION

A. Model Performance Comparison

The comparison of the Convolutional Neural Network (CNN) and Multilayer Perceptron (MLP) yields insightful observations. While both models achieve impressive accuracies, the CNN consistently outperforms the MLP. The inherent design of CNNs to capture spatial hierarchies in images contributes to their superior performance on image-based tasks. In comparison, the MLP's flattened architecture lacks the capacity to extract meaningful features from the raw pixel values effectively. To further improve the results, investing in more advanced architectures or exploring hybrid models that combine the strengths of both CNN and MLP can be considered.

B. Hyperparameter Selection

The choice of hyperparameters plays a pivotal role in model performance. For both the MLP and CNN, hyperparameter tuning can be an ongoing process. Optimizing learning rates, batch sizes, and dropout rates may enhance convergence and model stability. Regularization techniques such as L2 regularization can mitigate overfitting, especially for the MLP, which tends to be more prone to it due to its fully connected layers. The Grid Search or Random Search techniques can be employed to systematically explore various hyperparameter combinations.

C. Interpreting the Models

Interpreting the inner workings of deep learning models remains a challenging task. For the CNN, each layer captures

hierarchical features of increasing complexity. Analyzing these layers' responses to specific patterns or characters can provide insights into what the model has learned. Techniques such as Grad-CAM (Gradient-weighted Class Activation Mapping) can help visualize which image regions are crucial for predictions. In contrast, interpreting MLPs is more challenging due to their fully connected architecture. Techniques like SHAP (SHapley Additive exPlanations) can aid in understanding feature importance and model predictions.

D. Generalization and Overfitting

Balancing model complexity and generalization is a critical consideration. While the CNN showcases robust performance, the MLP is more prone to overfitting. To mitigate overfitting in the MLP, reducing the number of hidden units, employing dropout layers, or adjusting the regularization strength can be explored. Furthermore, data augmentation techniques, such as random rotations and translations, can enhance both models' ability to generalize to unseen data. The trade-off between model complexity and generalization should be a central focus in further model iterations.

In summary, improving the results and models requires a holistic approach. Experimentation with hyperparameter tuning, activation functions, and architecture design is essential. Enhancing interpretability and understanding what the models learn can provide valuable insights. Additionally, considering regularization techniques and data augmentation strategies can lead to more robust models. The field of deep learning offers a vast array of tools and techniques that can be leveraged to advance the performance of both the CNN and MLP models in character classification.

VI. Conclusions

In this study, we undertook a comprehensive analysis of the Kuzushiji-MNIST dataset, employing Convolutional Neural Networks (CNNs) and Multilayer Perceptrons (MLPs) for Japanese character classification. While both models exhibited remarkable accuracy, the CNN demonstrated superior performance due to its inherent image processing capabilities. Through hyperparameter optimization, model interpretation, and considerations for generalization, we shed light on avenues for enhancing model accuracy and interpretability. This research provides valuable insights into the complex world of deep learning and character classification, paving the way for future advancements in the field.

VII. References

References

[1] Scikit-learn: Machine Learning in Python, Pedregosa et al., Journal of Machine Learning Research, 2011.

[2] Martn Abadi, Paul Barham, Jian Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., "Tensorflow: A system for large-scale machine learning," 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 2016.

[3] Harris, C.R., Millman, K.J., van der Walt, S.J. et al., "Array programming with NumPy," Nature 585, 357362, 2020.

[4] Hunter, J.D., "Matplotlib: A 2D Graphics Environment," Computing in Science and Engineering, vol. 9, no. 3, pp. 9095, 2007.

VIII. Appendix (if necessary)

IX. Appendix

A. Pixel Heatmap of the Data

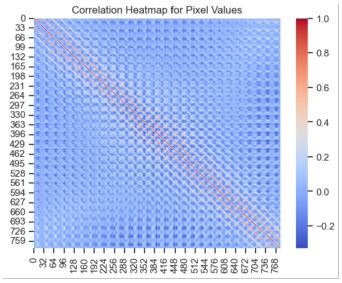

Fig. 12. CNN Accuracy and Loss

B. Activation Functions

In the context of neural networks, activation functions introduce non-linearity to the model. Here are some common activation functions used in deep learning:

1) ReLU (Rectified Linear Unit): The Rectified Linear Unit (ReLU) is a popular activation function that replaces all negative values with zero, resulting in a simple and computationally efficient non-linear transformation. The ReLU function can be defined as:

$$f(x) = \max(0, x)$$

2) Tanh (Hyperbolic Tangent): The hyperbolic tangent (tanh) activation function squashes input values to the range of -1 and 1, making it suitable for mapping data with zero mean to the interval [-1, 1]. The tanh function is given by:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

3) Sigmoid: The sigmoid activation function is often used for binary classification problems, as it maps input values to the range [0, 1], representing probabilities. The sigmoid function is defined as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

4) Leaky ReLU: The Leaky ReLU is a variant of the ReLU function that allows a small gradient for negative values, preventing neurons from dying during training. The Leaky ReLU is defined as:

$$f(x) = \{\, x\ , if x > 0 \alpha x, if x \leq 0$$

where α is a small positive constant (e.g., 0.01).

These activation functions play a crucial role in shaping the behavior of neural networks and can be chosen based on the specific requirements of the problem at hand and were mentioned in the paper.