Comparative Analysis of CNN and MLP Classifiers on Japanese Character Recognition: Architecture, Training Techniques, and Performance Evaluation

Tess O. Christensen
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL USA
tolivia.christen@ufl.edu
Performed the CNN experiment and aided in the production of this paper

Catalina B. Murray
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL USA
catalinamurray@ufl.edu
Performed the MLP experiment and aided in the production of this paper

***Abstract*—This study comprehensively analyzes two prominent neural network classifiers, the Convolutional Neural Network (CNN) and the Multi-Layer Perceptron (MLP), in their application to Japanese character recognition. We rigorously trained both classifiers, experimenting with a range of hyperparameters and stopping criteria.**

Our results, showcased through confusion matrices, graphs, and accuracy tables, offer a comparative view of the classifiers, underscoring their advantages and drawbacks. The CNN achieved an impressive accuracy of 98%, while the MLP and the Radial Basis Function (RBF) network, which was assessed as an alternative to the MLP, attained accuracies of 89% and 86%, respectively.

Additionally, we demonstrate our rationale behind hyperparameter selection through extensive numerical evidence, ensuring optimal generalization and performance. Beyond the quantitative results, this study emphasizes the methodology and reasoning that informed our choices, particularly in hyperparameter selection, ensuring enhanced generalization and performance.

I. Introduction

If you did a Google search for Kuzushiji, you would be hard-pressed to find any historical information about it, and most search results would provide research regarding Machine Learning to help decipher it. Digging further into these bits of research, we find that it is described as a traditional form of Japanese calligraphy (i.e., Japanese cursive) that was used over a thousand years ago to record ancient historical documents [5]. It was eliminated from the regular school curriculum in 1900, leaving many of today's scholars needing help deciphering it [7]. With this, there is an impending risk of losing access to a significant part of Japan's cultural legacy.

In recent years, the surge in machine learning and deep learning applications offers a promising avenue for automating the recognition of such intricate scripts. Of these applications for classifying unique character languages, the most effective and accurate have been techniques (i.e., Japanese, Hindi) such as the *Modular Neural Networks* [4], *Convolutional Neural Networks* with and without data augmentation [5], [7], *Online Character Recognition* [6], *Multilayer Perceptrons* [8], *Radial Basis Functions* [8], and *Optical Character Recognition* [2].

Convolutional Neural Networks (CNNs), in particular, have revolutionized the domain of image recognition due to their capacity to learn hierarchical features from raw pixel data [3]. On the other hand, Multi-Layer Perceptrons (MLPs), a class of feedforward artificial neural networks [1], have demonstrated their efficacy in tasks that require understanding patterns in the data. While MLPs might appear simplistic compared to CNNs, their layered architecture and capacity for nonlinear transformations can be surprisingly effective for specific classification tasks.

This study aims to compare both CNNs and MLPs to classify Kuzushiji characters. By constructing and training models separately based on each approach, we aim to dissect their strengths and limitations. The culmination of our experiment lies in a comprehensive comparison of their performances, shedding light on the most effective strategy for Kuzushiji character recognition.

Our comparative analysis discovered that the CNN was the optimal model for training this dataset. Though it was computationally more expensive and had longer runtimes, the higher accuracy of 98% compared to the MLP's 89% made it the better design.

This paper is organized as follows: Section II details the precise methodology used in our experimentation process. Section III describes the results of the experiment. Section IV provides a qualitative and quantitative analysis of the results, listing limitations and future work. Finally, Section V concludes this paper.

II. Methodology

A systematic and comparative approach was adopted to develop an efficient and robust system for classifying the Japanese characters in the Kuzushiji MNIST dataset. This study performed two distinct experiments, leveraging the capabilities of different neural network architectures. The primary

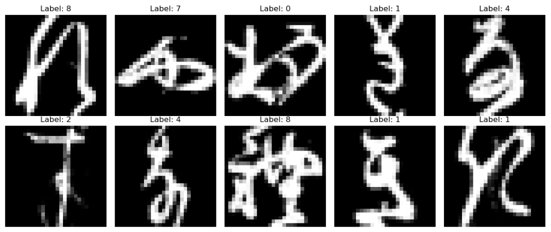


Fig. 1. Example images from the dataset

objective was to train these networks for accurate character classification and compare their performances, providing a comprehensive understanding of their strengths and potential shortcomings.

One experiment utilized a Convolutional Neural Network (CNN), and the other experiment was built upon a Multi-Layer Perceptron (MLP) Both experiments were executed by separate coders, ensuring the process was independent and unbiased. Upon completion of the training phase, the core metric of interest was the accuracy of the classifiers. This criterion was the foundation for the subsequent comparison between the CNN and MLP models.

We start with the same dataset for both models, comprising 70,000 grayscale images of Japanese characters, each size 28×28. These characters span ten distinct categories, with an equal distribution of 7,000 images per category. A sample of the characters is found in Fig 1. The dataset was divided into a training set of 60,000 images and a test set containing 10,000 images to facilitate training and evaluation.

As we delve deeper into the methodology, we will elaborate on the data preparation, network design, training process, and evaluation metrics adopted in each experiment.

A. CNN Approach

This section describes the CNN approach performed by one individual coder.

a) Data Preparation and Set Division: For the training phase, two-thirds of the dataset was used, ensuring a balanced representation of each character class. The remaining one-third was allocated for validation purposes. This split was executed randomly, but care was taken to maintain an equivalent distribution of exemplars across training and validation sets for each class.

b) Convolutional Neural Network (CNN) Architecture: The organization of the layers of our CNN were composed as follows:

1) **Convolutional Layer**: Employs 32 filters of size 3×3 with ReLU activation.
2) **Max Pooling Layer**: Uses a pool size of 2×2.
3) **Convolutional Layer**: Comprises 64 filters of size 3×3 with ReLU activation.
4) **Max Pooling Layer**: Uses a pool size of 2×2.
5) **Flatten Layer**: Converts the 2D feature maps into a 1D feature vector.
6) **Dense Layer**: Comprises 128 neurons with ReLU activation.
7) **Dropout Layer**: Layer with a variable dropout rate in experimentation.
8) **Output Layer**: A dense layer with 10 neurons.

A visual representation of this design can be seen in Fig 2 below.

We included one lower-level *convolutional layer* to capture low-level features like edges and gradients and a second in the third layer to recognize more abstract features like shapes, patterns, and combining edges. We decided on the RELU activation function for each convolutional layer because it is computationally more efficient for this amount and type of data than that of the sigmoid and Tanh functions.

Additionally, following both convolutional layers were the *pooling layers* to reduce the spatial dimensions while retaining significant features. This method also effectively reduces overfitting and the *dropout layer*. Next, we have the *flatten layer* that re-shapes the multidimensional input into a one-dimensional feature vector to be fed into the fully connected layers.

After the tensor has been flattened, we have two *dense layers* (i.e., fully connected layers) that serve as a bridge between the previous layers and the final classification output. The first applies a RELU activation function to ensure non-linearity, allowing the network to learn and make decisions based on the combinations of features. The second dense layer (i.e., the output layer) has 10 units corresponding to the 10 categories of the Japanese characters from the data. Here, we used the softmax activation function to ensure that the outputs of the layer sum to 1, which allows them to be interpreted as probabilities.

Finally, we have the *dropout layer* following the initial dense layers, which acts as a network regularizer to reduce overfitting by generating a more generalized and robust model.

The Adam optimizer, a widely acknowledged optimization algorithm in deep learning, was employed due to its adaptive learning rate capabilities. The loss function chosen was the sparse categorical cross-entropy, suitable for multi-class classification tasks.

c) Hyperparameter Tuning: A manual search over specific hyperparameters was performed to ascertain the optimal performance of the CNN. The hyperparameters under consideration were:

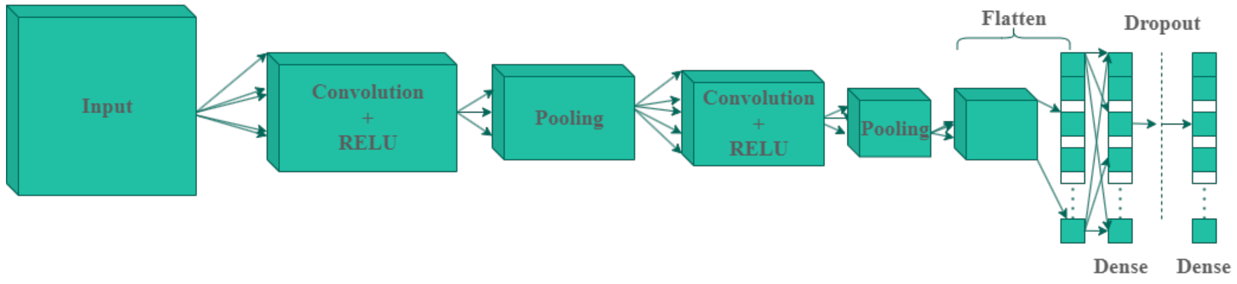


Fig. 2. Diagram of the implemented convolutional network

- **Learning Rate (LR)**: Explored values were [0.001, 0.01, 0.1].
- **Dropout Rate (DR)**: Values considered included [0.2, 0.5, 0.7].
- **Batch Size (BS)**: The batch sizes investigated were [32, 64, 128].

B. MLP Approach

This section describes the MLP approach performed by the second individual.

a) Data Preparation and Set Division: One-third of the training data was set aside for validation during the training phase. A random stratified split was used, ensuring each of the ten classes had an equal distribution across the training and validation sets.

b) Multi-Layer Perceptron (MLP) Architecture: The organization of the MLP architecture is as follows:

1) **Input Layer**: Responsible for flattening the data from a 2D vector into a 1D format.
2) **Hidden Layer 1**: Employs a variable number of units (based on hyper-parameter experimentation) with ReLu activation function.
3) **Hidden Layers 2, 3, 4**: Variable additional layers, each with fifty percent of the units in Hidden Layer 1 and the ReLu activation function. Whether or not to include additional hidden layers was evaluated during hyper-parameter tuning.
4) **Dropout Layers**: A dropout layer follows each additional hidden layer. The dropout rate was evaluated for best performance. Deactivates a fraction of the neurons during training to prevent over-fitting.
5) **Output Layer**: A dense layer with ten units and the SoftMax activation function.

Once the various hyper-parameters were chosen, a visual of the final MLP model can be seen in figure 3.

The basic structure of the MLP included one flattened layer, which helps to create a one-dimensional vector from the 2-dimensional input shape. This is required to complete the computation by the following layers. This was followed by at least one hidden layer, a dense layer with several units determined by the iterative search. An iterative search also determined the number of hidden layers to include. If more hidden layers were included, they were followed by dropout layers to help generalize the final model. Each hidden layer used the ReLu activation function, a common choice for the Multi-Layer Perceptron. It was chosen based on its computational efficiency compared to the sigmoid and tanh functions. Additionally, it helps to avoid the vanishing gradient problem. The final layer was the output layer, composed of 10 neurons for each of the 10 classes. It utilized the SoftMax activation function, a good choice for multi-class classification problems.

The Adam optimizer, which utilizes an adaptive learning rate to update weights in training efficiently, was chosen for its stability and robustness. Adam is usually a good choice and provides many advantages, including its ability to converge faster than traditional gradient descent methods. The categorical cross-entropy function was chosen as the loss function because of its ability to deal with multi-class classification problems.

An early stopping criterion was used to ensure learning was stopped at the best point to optimize generalization and prevent over-fitting. The model was fit with 15 epochs at first to ensure the model was able to stop training at the best time.

c) Hyperparameter Tuning: The architecture decisions are as follows:

- **Number of Layers**: The number of hidden layers and units were found through an exhaustive search. [1,2,3,4] hidden layers were evaluated. In an MLP, the first hidden layer is responsible for the basic structure, while the deeper layers are usually attributed to the more complex features. After visualizing the data and its moderately complex nature, it was noted that more than one layer would probably be needed, especially since the data is in the form of images but not more than 4.
- **Number of Units**: The number of neurons follows a narrowing network where the number of neurons is reduced in deeper layers. This is a common choice because the deeper layers, which capture more high-level and abstract features, often require fewer parameters. Additionally, this structure helps with the computational complexity. The number of units tested in the first hidden layer was [64, 128, and 256]. The Units were then halved in each of the following hidden layers, if there were any.
- **Global Learning Rate**: The global learning rate directly impacts how quickly the optimization algorithm will con-

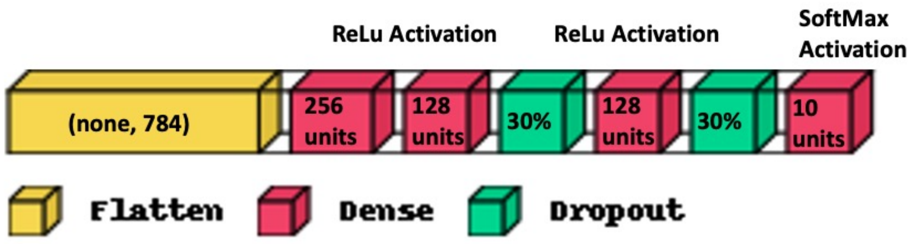


Fig. 3. The MLP architecture

verge. Since an adaptive learning rate was used, Adam, the global learning rate determines the initial conditions of how to scale the gradients during back-propagation. The evaluated learning rates were [0.001, 0.01, and 0.1].

- **Dropout Rate**: Dropout layers were added to reduce the chance of over-fitting and improve generalization. A dropout layer was added after the second, third, and fourth hidden layers. Dropout rates of [0.3, 0.4, and 0.5] were evaluated with the manual search.
- **Early Stopping Criteria**: An early stopping criterion was introduced to prevent over-fitting. To do this, the validation loss was monitored, and training was stopped if the validation loss did not improve for several consecutive epochs. This number is denoted as "patience," two values were evaluated: a patience of [3,5]. Once the training is stopped, the model restores the best weights found during training.
- **Batch Size**: The batch size can often affect model convergence. While a smaller batch size is more computationally complex, it can sometimes reach a better solution. Batch sizes of [12,32] were evaluated.
- **Other Structures: RBF** Alternative structures, specifically the Radial Basis Function Neural Network (RBFNN), were also considered. The RBFNN architecture resembles an MLP but includes a Radial Basis Function (RBF) Layer. The RBF layer transforms the input data into a new space and uses a Gaussian function to calculate the radial distance to update the weights. The architecture for the RBF network included two hidden layers, the first with 256 units and the second with 128 units. The third layer was the RBF layer. One of the essential hyperparameters for an RBF network is the number of RBF units dependent on the data's complexity. This hyperparameter was determined through a manual search. In the case of an RBF network, a dropout layer does not need to be included because the RBF layer inherently creates some regularization.

C. Evaluation

Post-training, the model's performance was evaluated based on its accuracy on the validation set. This metric provided insights into the model's ability to generalize on unseen data and guided the decision-making process in the hyperparameter selection phase.

III. Results

In our pursuit to classify Kuzushiji characters, two separate experiments were conducted employing distinct neural network architectures. An individual coder managed each experiment to ensure independent and unbiased results. Below are the results for each experiment, which we will compare and analyze in the following section.

A. CNN Experiment

a) Hyperparameter Evaluation: Our experimentation with CNNs started with a manual exploration of a combination of hyperparameters.

- **Learning Rate**: A learning rate that's too high can cause the model to converge too quickly to a sub-optimal solution, whereas a learning rate that's too low can cause the process to get stuck or take an excessive amount of time to converge. To this end, our search found a learning rate of 0.001 to be the optimal rate.
- **Dropout Rate**: A dropout rate determines the number of neurons in the dropout layer that are dropped out or turned off during training. It can provide strong regularization effects, forcing the network to learn more robust and generalized features while not being so aggressive as to weaken its capacity to learn from the data. Our search found the best value for the dropout rate for this model was 0.5.
- **Batch Size**: Batch sizes small enough to fit in most GPU memories generally balance optimization benefits. Additionally, mini-batches have been found to converge faster and generalize better in many tasks. Therefore, our manual search found that a batch size of 64 was used in the final model.

As previously stated, our manual search achieved the best performance with a learning rate of 0.001, a dropout rate of 0.5, and a batch size of 64. With these values, the model yielded an accuracy of 98.5% and a loss of 0.069. To visually assess our model's performance, we analyzed three plots displayed in Figs 6, 4, and 5

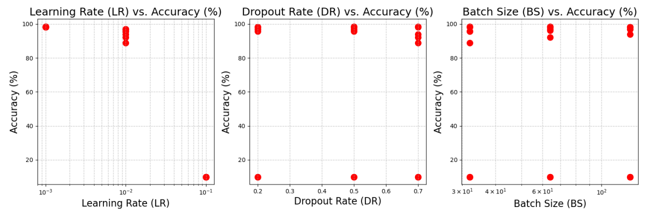


Fig. 4. Scatterplot representing the accuracies per each hyperparameter

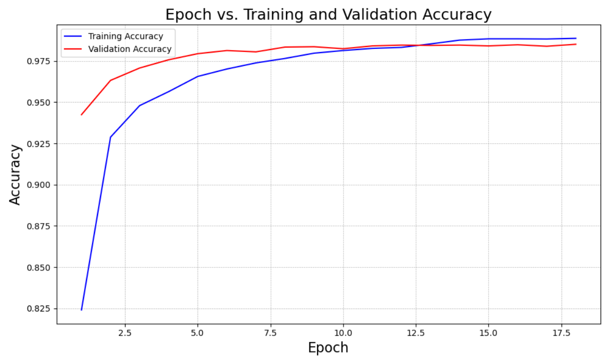


Fig. 5. Accuracies by epoch for training and validation.

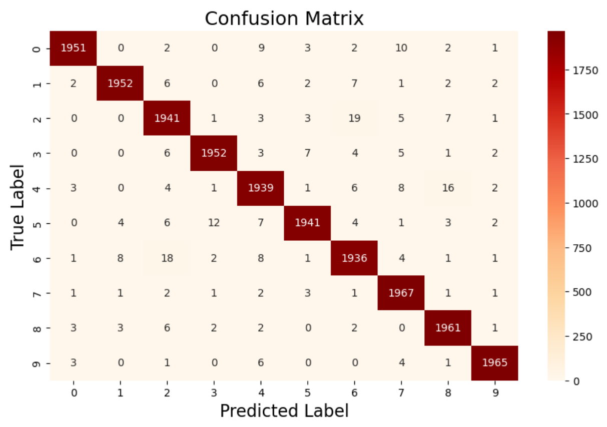


Fig. 6. Confusion matrix regarding the CNN experiment.

The confusion matrix (Fig 6) provides an understanding of the true positives, false positives, true negatives, and false negatives our model produced. The diagonal elements representing our true positives are strong, with very slight values in the other areas. The scatterplot (Fig 4) depicts how varying hyperparameters influence the accuracy of the model, which shows that the accuracy value had the most apparent effect as any training with a Learning Rate above 0.01 performed poorly. Finally, the Accuracy vs. Epochs Plot (Fig 5 shows the convergence of our model over epochs for both the training and validation sets. The epoch value was initialized to 50; however, the stop criteria forced the training to stop after epoch 18.

B. MLP Results

a) Hyperparameter Evaluation: Each of the hyperparameters listed above was evaluated with a manual search. The results of the search can be found in figure 7.

- **Number of Layers**: Figure 7 shows that as the number of layers increases, variance in validation accuracy also increases. Four layers prove to over-fit the data since the accuracy in the validation set is much more variable and often lower than the training accuracy. In the final model, three layers were chosen because they provided the best balance between high training accuracy and reasonably good validation accuracy.
- **Number of Units**: As the number of units increased, the validation accuracy also increased. In the final model, 256 units were used in the first hidden layer. 128 units were used in the second and third hidden layers.
- **Learning Rate**: The validation accuracy increased as the learning rate decreased. The final model utilized a learning rate of 0.01.
- **Batch Size**: The evaluated batch sizes did not significantly affect the model's outcome in validation. A batch size of 32 was used in the final model since it would increase computational efficiency.
- **Dropout Rate**: A dropout rate 0.3 was used in the final mode. It consistently had higher validation scores. While a higher dropout rate would increase the amount of regularization, this model is relatively simple. Therefore, a dropout rate smaller than 0.5 proved to be effective.
- **Patience**: The patience of the early stopping criteria did not significantly affect the model. A value of 3 was chosen for its increased computational efficiency. Additionally, at this stage, it was shown that this method utilizing the early stopping criteria ensured learning was stopped at the most appropriate time to improve generalization. During the training phase, each run ran between 7 and 10 epochs, and this was when validation accuracy started to decrease, even though training accuracy continued to increase. This is shown in figure 8.
- **Other Structures: RBF Network** For the alternative structure the main hyper-parameter that was evaluated was the number of RBF units to include. After a manual search 30 units were found to produce optimal results. The RBF structure itself was also assessed and compared to the MLP.

b) Evaluation in Test: : The test set containing 10 percent of the original data, which was left out during the training stage, was used to evaluate the model's generalization ability. The accuracy during the test using the MLP method was 89 percent. The confusion matrix is shown in figure 9.

The following section will analyze and discuss the results of each experiment and make a final comparison regarding their performance.

IV. Discussion

A. CNN Analysis

The Convolutional Neural Network (CNN) model deployed in this study to classify Kuzushiji characters demonstrated promising results, achieving an accuracy of 98.5%. The model

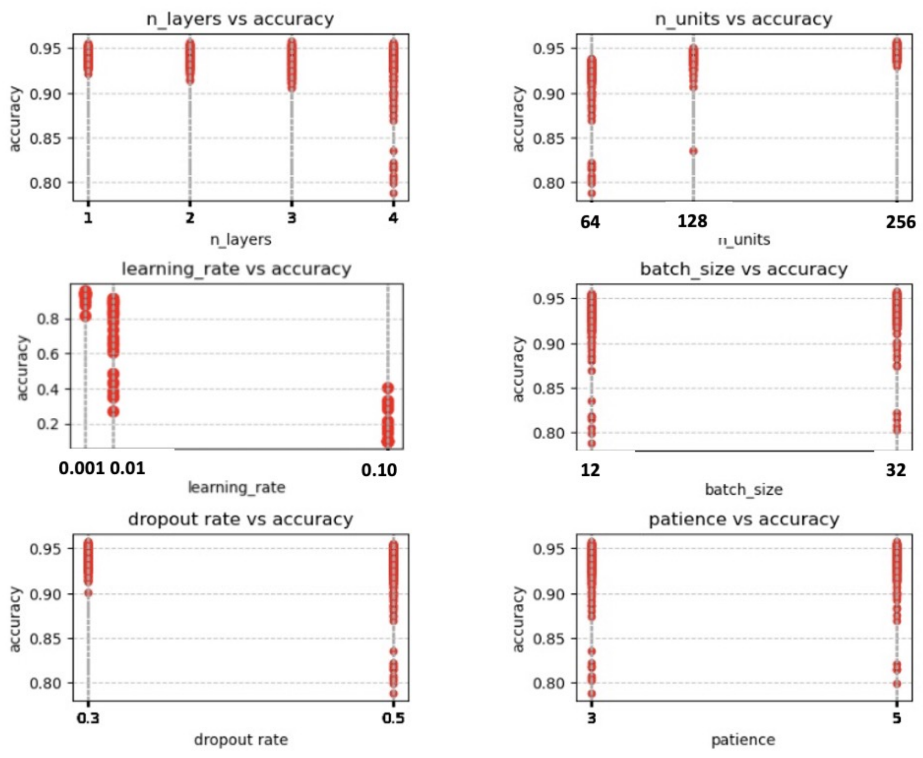


Fig. 7. Tables of the MLP hyperparameters vs validation accuracy

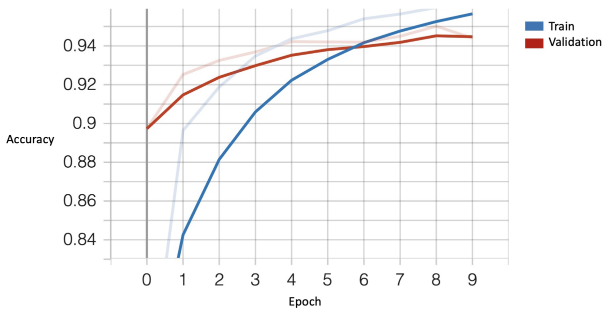
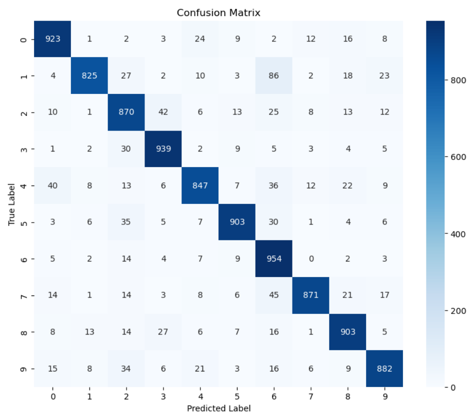


Fig. 8. For 762 runs, the number of Epochs versus accuracy was averaged to show learning was stopped at the appropriate time.



	0	1	2	3	4	5	6	7	8	9
0	923	1	2	3	24	9	2	12	16	8
1	4	825	27	2	10	3	86	2	18	23
2	10	1	870	42	6	13	25	8	13	12
3	1	2	30	939	2	9	5	3	4	5
4	40	8	13	6	847	7	36	12	22	9
5	3	6	35	5	7	903	30	1	4	6
6	5	2	14	4	7	9	954	0	2	3
7	14	1	14	3	8	6	45	871	21	17
8	8	13	14	27	6	7	16	1	903	5
9	15	8	34	6	21	3	16	6	9	882

Fig. 9. The confusion matrix using MLP method

was designed with two convolutional layers followed by max-pooling layers, a flatten layer, and two dense layers, with dropout applied before the final dense layer.

a) Model Structure and Design Choices: Using two convolutional layers is standard for simpler image classification tasks. Each convolutional layer was paired with a max-pooling layer, which reduces the spatial dimensions of the output, thereby decreasing the number of parameters and computational cost. It also helps in making the model more translation-invariant.

The dense layers following the convolutional layers help in the classification task. A dropout layer was introduced before the final dense layer to prevent overfitting. This is particularly beneficial when the model starts memorizing the training data rather than generalizing it.

b) Improvements for Future Work: Though our accuracy was high, some changes to the model could increase the accuracy of the model. The literature shows that *data augmentation* could prove valuable for this dataset [7]. We could achieve this by applying rotations, translations, and zooming to the training data. This can also help prevent overfitting. Additionally, we could increase the *depth of the Network* by adding more convolutional layers to aid in capturing more intricate patterns in the data. This may risk overfitting, however. Finally, it may be beneficial to perform a more in-depth and automated hyperparameter search, using *Grid Search*, *Random Search*, or even *Bayesian Optimization*. Due to the computationally expensive nature of these searches, time we limited our ability to perform an optimal hyperparameter search.

While the CNN model produced satisfactory results, deep learning models always have room for improvement. Future work could focus on implementing the suggested improvements, leading to even higher accuracies and a more robust model.

B. MLP Analysis

The final MLP model utilized three hidden layers with a narrowing network and two dropout layers. This model achieved relatively good results with 89% accuracy.

a) Model Structure and Design Choices: The model structure that was ultimately chosen struck a balance between increasing training accuracy and improving generalization. Multiple hidden layers were added to enable the model to characterize both the simpler features and some of the more complicated features. Dropout layers were utilized to improve generalization.

b) MLP compared to Radial Basis Function (RBF) Network: This network is advantageous because it has a simple architecture and learning speed. However, it achieved an 86 percent accuracy, lower than the original MLP. RBFs like MLPs are not a common choice for image classification tasks because they don't have a way to capture spatial features effectively.

c) Improvements for Future Work: Admittedly 89% accuracy is less than desirable. However, increasing the complexity of the MLP architecture would not improve the accuracy, which would only lead to overfitting. Alternatively, a different option to increase accuracy would be to employ preprocessing functions such as data augmentation or feature extraction to make input data more applicable to an MLP. Another method would be ensemble learning. By combining multiple MLPs, the model's performance could be improved. Finally, transfer learning could also be used by leveraging other models trained on similar tasks, and their features could help to fine-tune this model's performance.

C. Comparison

For this task, the CNN demonstrated the optimal performance. CNNs are widely recognized as a good choice for image classification tasks because they can effectively capture spatial features through convolutional layers. A detailed analysis of the results is listed in table I, revealing that the CNN consistently outperformed the MLP and RBF network in accuracy and F1 scores. The F1 score explains the balance between precision and recall and is a valuable metric for classification tasks. MLPs are advantageous because they are simple to implement but prone to overfitting and need more spatial awareness. CNNs are more complex and, therefore, require more significant amounts of data but usually have superior performance for image classification tasks. For this task, the best choice is the CNN because the data set contains a large amount of labeled image data.

Method	Acc (%)	F1 Score (min, max)	Runtime (s)
CNN	98	0.98, 0.99	15
MLP	89	0.85, 0.93	4
RBF	86	0.84, 0.92	4

TABLE I
ACCURACY, MIN AND MAX F1 SCORE, AND RUNTIME FOR EACH METHOD.

V. CONCLUSION

In our investigation of neural network classifiers for Japanese character recognition, the Convolutional Neural Network (CNN) emerged as the most proficient model, boasting an accuracy rate of 98%. This significantly outperformed both the Multi-Layer Perceptron (MLP) and the Radial Basis Function (RBF) network, which achieved 89% and 86% accuracy, respectively.

After thoroughly evaluating both methods, including extensive hyperparameter tuning, the CNN method proved the most successful. While the MLP architecture was a simple and efficient choice with a smaller runtime than the CNN, it could have achieved more accurate results. These two methods often show a trade-off between model efficiency and accuracy.

In summary, while each classifier has its merits, the CNN stands out as our study's most promising tool for Japanese character recognition. Its superior accuracy, combined with the insights gained from our methodology, sets a benchmark for future endeavors in this domain.

REFERENCES

[1] Ahmet Cevahir Cinar. Training feed-forward multi-layer perceptron artificial neural networks with a tree-seed algorithm. *Arabian Journal for Science and Engineering*, 45(12):10915–10938, 2020.

[2] Soumendu Das and Sreeparna Banerjee. An algorithm for japanese character recognition. *International Journal of Image, Graphics and Signal Processing*, 7(1):9, 2014.

[3] Fan Feng, Shuangting Wang, Chunyang Wang, and Jin Zhang. Learning deep hierarchical spatial–spectral features for hyperspectral image classification based on residual 3d-2d cnn. *Sensors*, 19(23):5276, 2019.

[4] Tadashi Horiuchi and Satoru Kato. A study on japanese historical character recognition using modular neural networks. In *2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*, pages 1507–1510. IEEE, 2009.

[5] Christoph Lippert, Md Hasan Shahriar, Md Shafayet Hossen Chowdhury, Alexander Junger, Md Golam Rasul, and Mohammad Yakub. Kuzushiji classification.

[6] Cheng-Lin Liu and Xiang-Dong Zhou. Online japanese character recognition using trajectory-based normalization and direction feature extraction. In *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft, 2006.

[7] Kazuya Ueki and Tomoka Kojima. Survey on deep learning-based kuzushiji recognition. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10-15, 2021, Proceedings, Part VII*, pages 97–111. Springer, 2021.

[8] Brijesh K Verma. Handwritten hindi character recognition using multilayer perceptron and radial basis function neural networks. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 2111–2115. IEEE, 1995.