94691 Deep Learning – Autumn 2025

Assignment 1 – Part B

Student Name: **Drishya Lal Chuke**

Student Id: 25076922

Dataset and Data Preparation:

The Japanese MNIST dataset composed of 70000 images of handwritten Hiragana characters. The target variable has 10 different classes. The composition of dataset was split into 60,000 for training set and 10,000 for test set. The images were in grayscale and each had 28x28 pixels dimensions.

Initially these images were stored as 3-D arrrays with no explicit channel dimension. To align the standard image processing conventions, the data was reshaped to include a channel dimension and transforming the images for training and testing set. Before moving to model training, the pixel values were normalized by dividing by 255.0, ensuring that all values lie between 0 and 1. This standardization is essential for imporving the network's convergence during training. Alongside the images, the labels were loaded as numerical indices corresponding to the 10 classes. In some parts of the pipeline, these labels were also transformed into one-hot vectors, though they were later converted back to class indices when computing loss with functions like CrossEntropyLoss.

A transformation pipeline using torchvision was applied to ensure that our model was well-prepared for the complexity of real-world data. This pipeline included converting images to tensors and flattening them from a 28×28 matrix into a single 784-dimensional vector, which is required for our fully connected layers.

Overall, the data preparation process was designed to transform raw KMNIST images into a format that is consistent, normalized, and optimally structured for training deep learning models. This meticulous preparation not only ensures better model convergence but also lays the groundwork for reliable performance evaluation during subsequent experiments.

Experiment 1:

1. Architecture

Here's out Fully Connected Layers for experiment 1:

- **Input Layer:** The 28×28 images are flattened into vectors of size 784.
- **Hidden Layer 1:** 512 neurons with ReLU activation, followed by a dropout layer with probability 0.2.
- **Hidden Layer 2:** 256 neurons with ReLU activation, followed by another dropout layer with probability 0.2.
- **Output Layer:** 10 neurons producing raw logits, corresponding to the 10 classes.

The two hidden layers were chosen to provide a moderate network capacity for handwritten character recognition with **Dropout (p=0.2)** helps mitigate overfitting by randomly deactivating 20% of neurons during training. No activation function(like softmax) was applied here because PyTorch's loss functions (e.g., CrossEntropyLoss) expect raw logits.

2. Training Process

Before training the model, the one-hot encoded target features were converted back to integers. Dataloader for train and test set was defined.

Hyperparameters :

- **Optimizer:** Adam with a learning rate of 0.001 (no weight decay).
- **Loss Function:** *nn.CrossEntropyLoss()* (standard choice for multi-class classification).
- **Batch Size**: 128
- **Epochs:** 50

No explicit validation set was used in this experiment. The model was trained to minimize training loss over 50 epochs, and performance was checked on the test set afterward.

3. Results

The test accuracy was 92.54% indicating a reasonably strong model for a baseline fully-connected network. In the learning curve for training loss (fig1) the loss started around 0.45 and dropped sharply in the first 10 epochs, which means it's learning fast. The training loss was close to 0 by epochs 50 sugeesting that the model fit the training data extremely well.

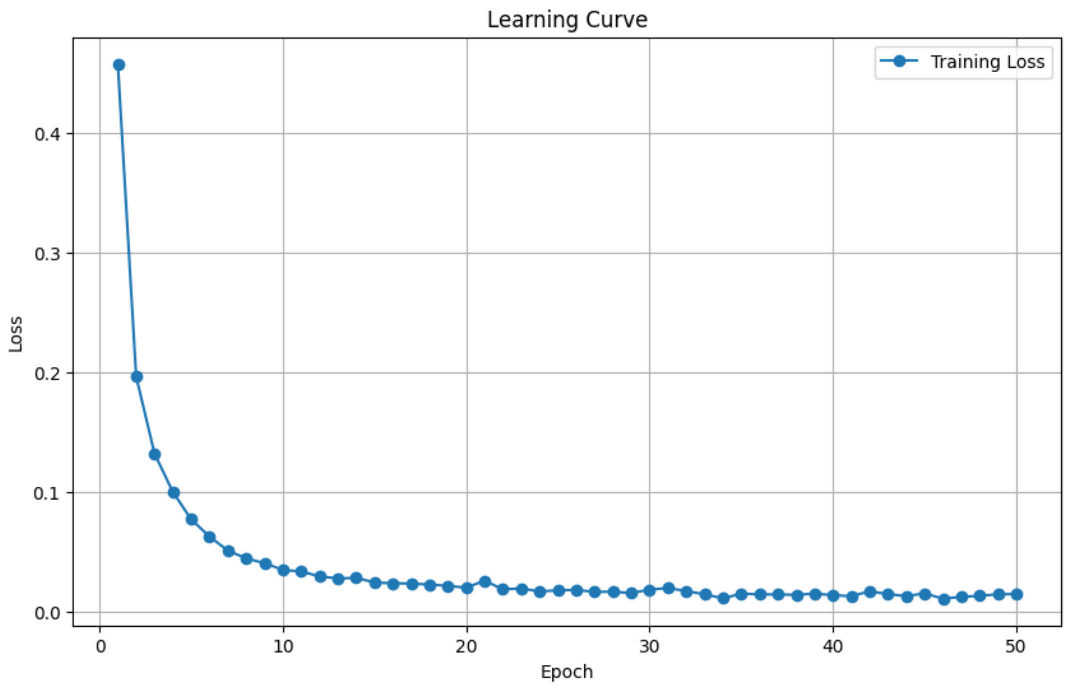
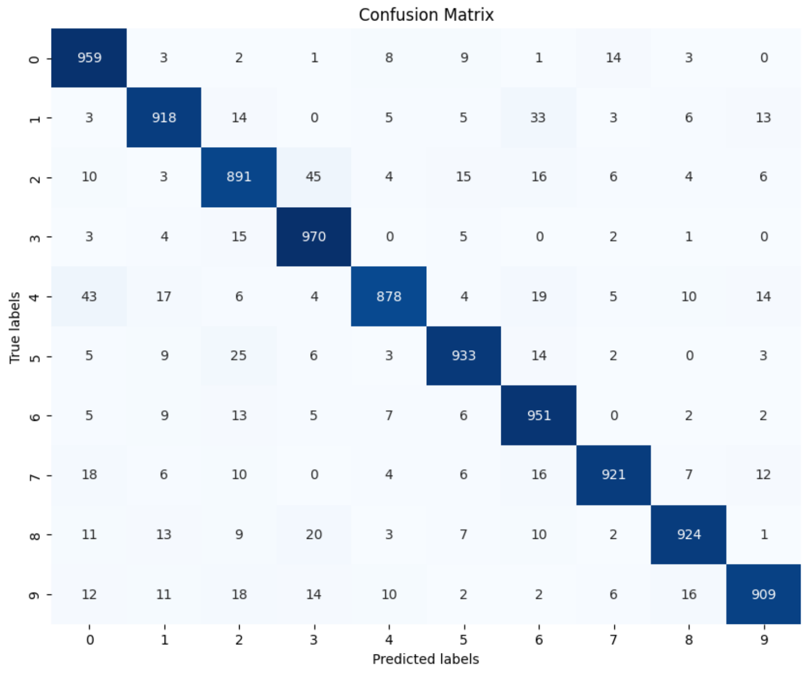


Figure 1: Learning Curve for train loss - Exp 1

Confusion Matrix & Classification Report:

Table 1: Confusion Matrix test-set Exp 1



Confusion Matrix

True labels \ Predicted labels	0	1	2	3	4	5	6	7	8	9
0	959	3	2	1	8	9	1	14	3	0
1	3	918	14	0	5	5	33	3	6	13
2	10	3	891	45	4	15	16	6	4	6
3	3	4	15	970	0	5	0	2	1	0
4	43	17	6	4	878	4	19	5	10	14
5	5	9	25	6	3	933	14	2	0	3
6	5	9	13	5	7	6	951	0	2	2
7	18	6	10	0	4	6	16	921	7	12
8	11	13	9	20	3	7	10	2	924	1
9	12	11	18	14	10	2	2	6	16	909

High Diagonal Values: Most classes show a strong diagonal, meaning the majority of predictions are correct.

Table 2: Classification report test-set Exp 1

Class	Precision	Recall	F1-Score	Support
0	0.8971	0.9590	0.9270	1000
1	0.9245	0.9180	0.9212	1000
2	0.8883	0.8910	0.8897	1000
3	0.9108	0.9700	0.9395	1000
4	0.9523	0.8780	0.9136	1000
5	0.9405	0.9330	0.9367	1000
6	0.8955	0.9510	0.9224	1000
7	0.9584	0.9210	0.9393	1000
8	0.9496	0.9240	0.9366	1000
9	0.9469	0.9090	0.9276	1000

Precision & Recall: Generally above 0.88 for most classes, with some classes (like 3 and 0) exceeding 0.90 in both metrics. Class 4 shows a lower recall (0.878), implying the model sometimes misclassifies actual class-4 samples. Other classes, such as 0 and 3, have very high recall but slightly lower precision, meaning the model occasionally mislabels other classes as these.

4. Observations & Analysis

- **Signs of Overfitting**: The training loss converged near zero, while the test accuracy plateaued around 92.54%. This gap suggests the model memorized training examples effectively but did not generalize perfectly to unseen data.
- **Potential Improvements**
    - **Additional Regularization:** Could include weight decay or higher dropout to further reduce overfitting.
    - **Hyperparameter Tuning:** Adjusting the learning rate, batch size, or dropout rates might push accuracy higher.
    - **Validation Set / Early Stopping:** A separate validation set would help identify overfitting earlier and refine hyperparameters in real time.
- **Baseline Reference**
    - As the first experiment, this model serves as a **baseline** for comparison with subsequent experiments (which introduced weight decay, different dropout rates, or increased network capacity).

5. Summary

Achieved **92.54%** test accuracy, a solid baseline for a fully-connected network on handwritten characters. Fast convergence in training, demonstrating the network can effectively learn the data distribution.

Limitations:

- Overfitting is evident, given the near-zero training loss but only ~92.54% test accuracy.
- Certain classes (e.g., class 4) show room for improvement in recall.

Experiment 2:

1. Architecture

The network architecture remains identical to Experiment 1. The only change from Experiment 1 is the introduction of **weight decay (L2 regularization)** in the optimizer. This was done to penalize large weights and improve the model's ability to generalize.

- **Input Layer:** The 28×28 images are flattened into vectors of size 784.
- **Hidden Layer 1:** 512 neurons with ReLU activation, followed by a dropout layer with probability 0.2.
- **Hidden Layer 2:** 256 neurons with ReLU activation, followed by another dropout layer with probability 0.2.
- **Output Layer:** 10 neurons producing raw logits, corresponding to the 10 classes.

2. Training Process

The 20% training dataset was set as validation set using ramdom_split, and new DataLoader was made.

Hyperparameters:

- **Optimizer:** Adam with a learning rate of 0.001 and weight decay set to 1e-5.
- **Loss Function:** *nn.CrossEntropyLoss()*, with any one-hot targets converted to class indices.
- **Batch Size:** 128
- **Epochs:** 50

During each epoch, training loss was computed and accumulated. At the end of each epoch, the model was evaluated on the validation set to track both validation loss and accuracy. The training and validation losses were recorded to plot learning curves, aiding in detecting overfitting.

3. Results

Validation Accuracy: Achieved a high **98.56%** accuracy on the validation set (12,000 samples).

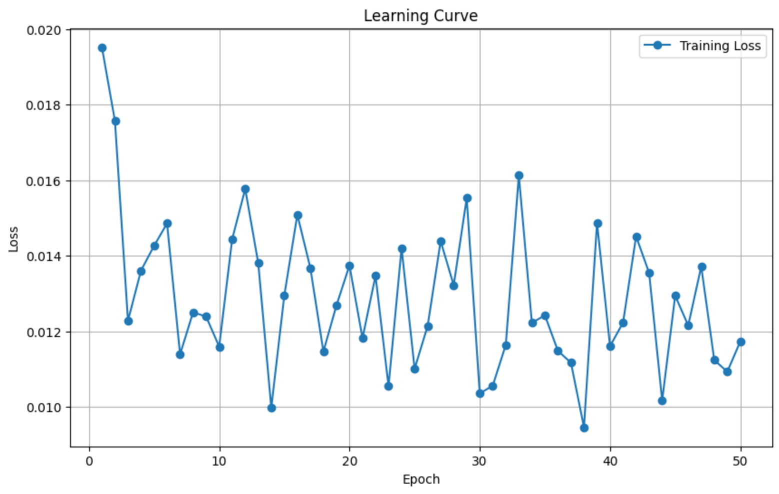


Figure 2: Learning curve train-loss exp-2

The train-loss curve is slightly higher, more flucuating does not mean worse performance. Higher validation accuracy shows that the model is learning more robust features.

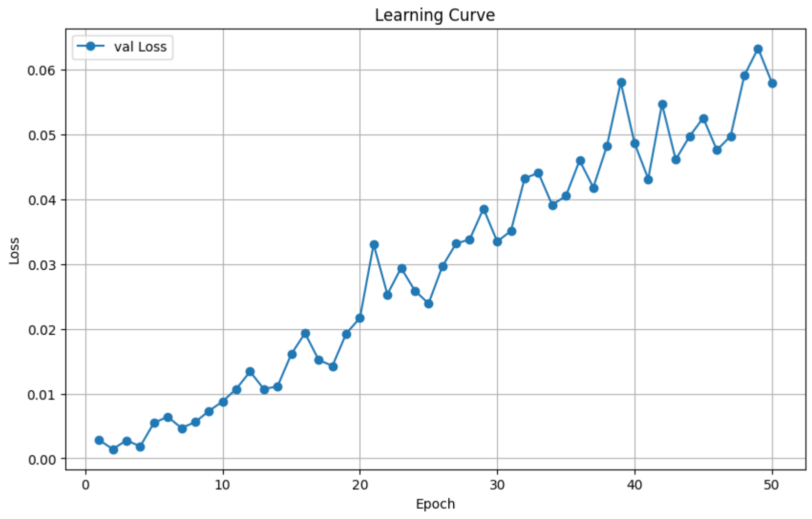


Figure 3: Learning curve val-set exp-2

It's unusal to see validation loss steadily climb if your validation also high, which means the model is becoming extremely confident. It can also point to a possible mismatch in evaluation.

Confusion Matrix & Classification Metrics:

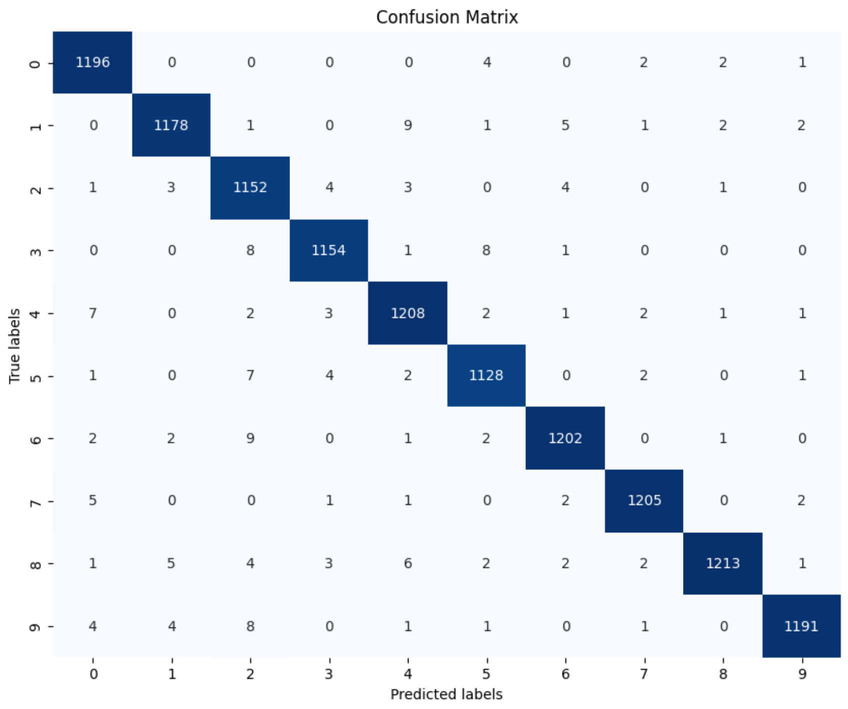


Figure 4: Confusion Matrix val-set exp-2

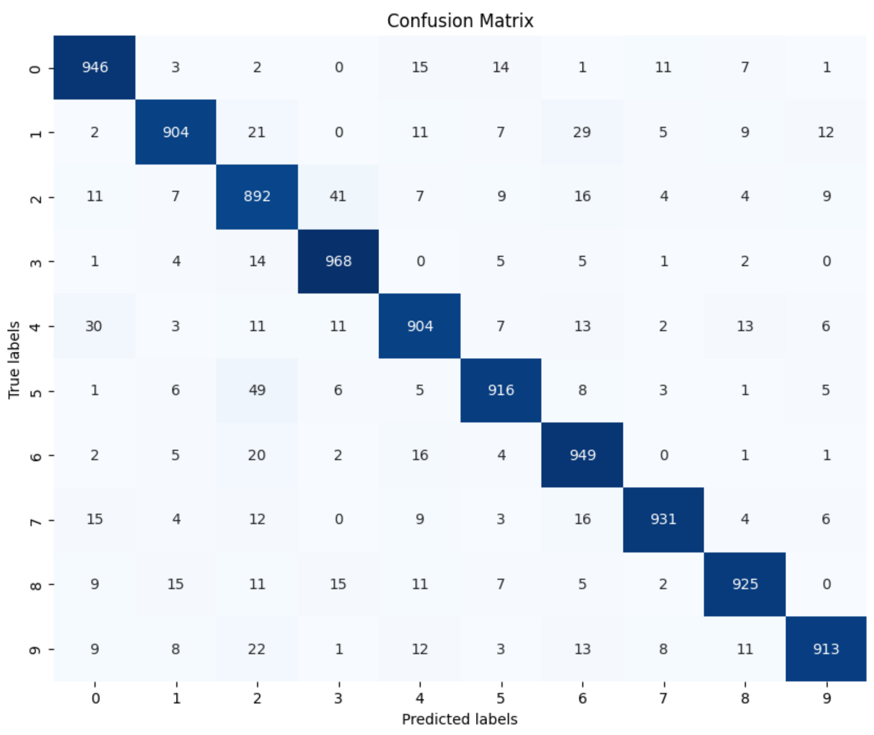
The confusion matrix confirmed that the majority of samples for each class were correctly classified with minimal misclassifications.

Class	Precision	Recall	F1-Score	Support
0	0.9827	0.9925	0.9876	1205
1	0.9883	0.9825	0.9854	1199
2	0.9673	0.9863	0.9767	1168
3	0.9872	0.9846	0.9859	1172
4	0.9805	0.9845	0.9825	1227
5	0.9826	0.9852	0.9839	1145
6	0.9877	0.9861	0.9869	1219
7	0.9918	0.9910	0.9914	1216
8	0.9943	0.9790	0.9866	1239
9	0.9933	0.9843	0.9888	1210

Precision, recall, and F1-scores for all classes were consistently high (generally in the range of 0.96 to 0.99), indicating strong per-class performance.

Test Accuracy: Recorded at **92.48%** on the test set (10,000 samples).

Confusion Matrix and Classification Report:



	0	1	2	3	4	5	6	7	8	9
0	946	3	2	0	15	14	1	11	7	1
1	2	904	21	0	11	7	29	5	9	12
2	11	7	892	41	7	9	16	4	4	9
3	1	4	14	968	0	5	5	1	2	0
4	30	3	11	11	904	7	13	2	13	6
5	1	6	49	6	5	916	8	3	1	5
6	2	5	20	2	16	4	949	0	1	1
7	15	4	12	0	9	3	16	931	4	6
8	9	15	11	15	11	7	5	2	925	0
9	9	8	22	1	12	3	13	8	11	913

Figure 5: Confusion Matrix for test-set exp-2

The test confusion matrix exhibited a strong diagonal, confirming that the majority of predictions were correct across the 10 classes.

Table 3: Classification Report for Test-Set exp-2

Class	Precision	Recall	F1-Score	Support
0	0.9220	0.9460	0.9339	1000
1	0.9426	0.9040	0.9229	1000
2	0.8463	0.8920	0.8685	1000
3	0.9272	0.9680	0.9472	1000
4	0.9131	0.9040	0.9085	1000
5	0.9395	0.9160	0.9276	1000
6	0.8995	0.9490	0.9236	1000
7	0.9628	0.9310	0.9466	1000
8	0.9468	0.9250	0.9358	1000
9	0.9580	0.9130	0.9350	1000

Most classes achieved precision and recall values above 0.90. Some classes (like class 2) had slightly lower precision (around 0.8463) or recall (around 0.8920), but overall performance remained strong.

4. Analysis & Comparison

- **Impact of Weight Decay:** The introduction of weight decay helped regularize the model, reducing overfitting compared to Experiment 1. This is evident from the high validation accuracy (98.56%), suggesting improved generalization on unseen data.
- **Validation vs. Test Performance:** While the validation performance is outstanding 98.56%, the test accuracy of 92.48% is lower. This gap might indicate that the validation set was somewhat easier or more similar to the training set, or that further tuning of regularization parameters could be beneficial.
- **Comparison to Experiment 1:**
  - **Experiment 1** achieved a test accuracy of 92.54% without weight decay, while **Experiment 2** has a comparable test accuracy of 92.48% but a much higher validation accuracy.

  This suggests that although both experiments yield similar test results, the weight decay in Experiment 2 improved the model's ability to generalize on validation data, potentially indicating more robust learning.

The high validation accuracy (98.56%) and strong per-class metrics suggest that weight decay has a positive effect on the model's performance during training.However, the similar test performance between experiments indicates that further fine-tuning (or adjustments to the validation split) might be needed to close the gap between validation and test performance.

5. Summary

Experiment 2 builds upon the baseline by adding weight decay, resulting in:

- A robust validation performance with **98.56%** accuracy and excellent classification metrics across all classes.
- A slight discrepancy between validation (98.56%) and test accuracy (92.48%), indicating possible overfitting or differences in data distributions.
- Overall, while the test accuracy remains in the low 90s (comparable to Experiment 1), the improved validation performance suggests that the model has learned more generalizable features, setting the stage for further improvements in subsequent experiments.

This report highlights how a small change—adding regularization via weight decay—can significantly influence model behavior and provides a strong basis for further experimentation and optimization.

Experiment 3

1. Architecture

The Experiment 3 the number of neurons was increased in the first two hidden layers (1024 in the first layer and 512 in the second), which allows the network to capture more complex patterns in data. A higher dropout rate 0.5 was applied, to provide sttronger regulization to combat overfitting.

- **Input Layer:** Images of size 28×28 are flattened into a 784-dimensional vector.
- **Hidden Layer 1:** 1024 neurons with ReLU activation. Followed by a dropout layer with a dropout rate of 0.5.
- **Hidden Layer 2:** 1024 output is then fed to a second fully connected layer with 512 neurons, again with ReLU activation. Followed by a dropout layer with a dropout rate of 0.5.
- **Output Layer:** 10 neurons producing raw logits corresponding to the 10 classes, suitable for use with CrossEntropyLoss.

2. Training Process

The model was trained on the training set, and performance was monitored on a separate validation set to track generalization.

Hyperparameters:

- **Loss Function:** CrossEntropyLoss (with conversion of one-hot targets to class indices as needed).
- **Optimizer:** Adam with a learning rate of 0.001 and weight decay of 1e-5.
- **Batch Size:** 128
- **Epochs:** 50 (with training loss recorded at each epoch to plot learning curves).

- **Regularization Techniques:**
  - Dropout was set to 0.5 in both hidden layers.
  - Weight decay (L2 regularization) was applied in the optimizer to control weight magnitudes.

Training and validation loss curves were recorded, showing how the model's loss evolved over epochs.

3. Results

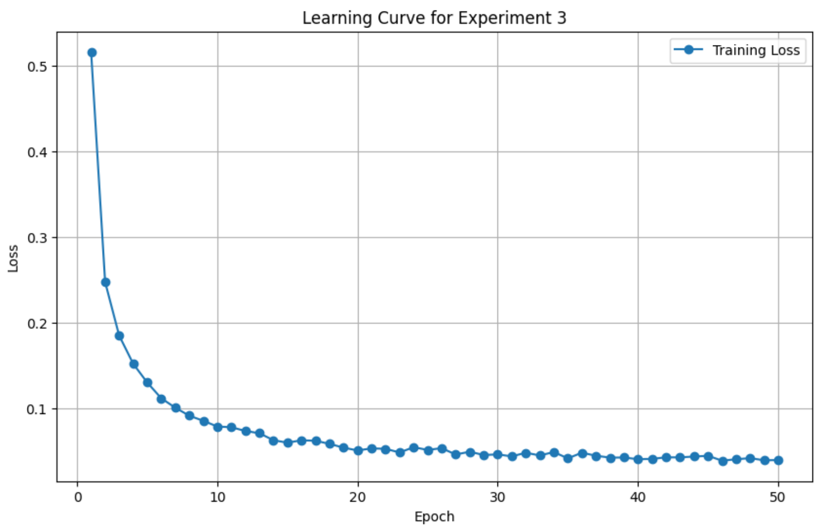


Figure 6: Learning curve of train loss exp-3

The **Training Loss** started around 0.50 and experienced a rapid decline in the first 5–10 epochs, indicating quick initial learning. By the end of 50 epochs, the training loss was very low (approaching 0.03 or below), demonstrating that the model effectively learned the training data.

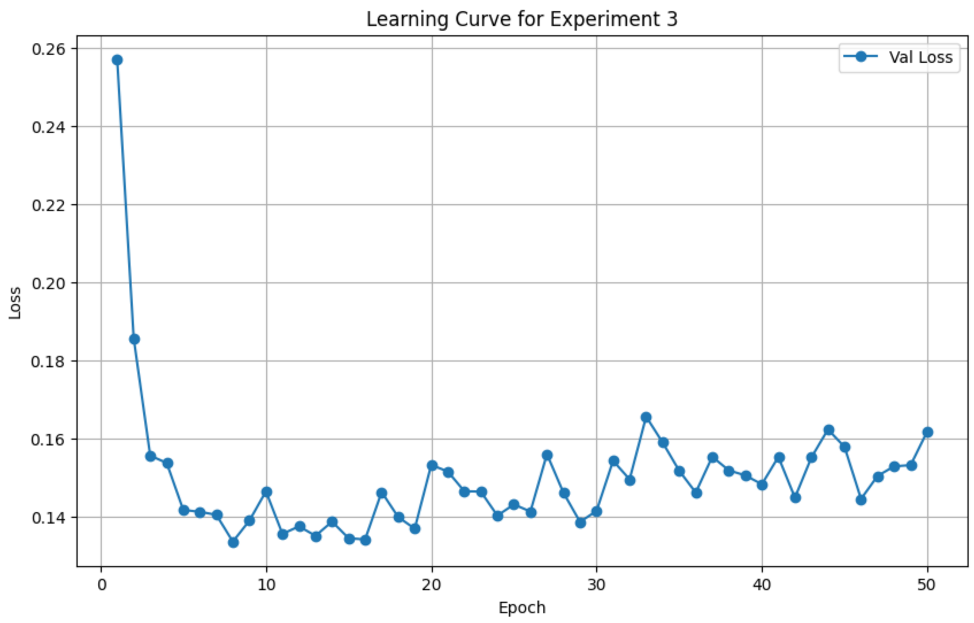


Figure 7: Learning Curve val loss exp-3

The **Validation Loss** initially dropped significantly (from around 0.26 to a low near 0.14), but then oscillated mildly between 0.14 and 0.18 during later epochs. Such oscillations are common when using high dropout rates and a larger network; they indicate that while the model is well-regularized, it is also exploring different weight configurations to balance capacity and generalization. **Validation Accuracy** was 96.78% on 12,000 samples.

Confusion Matrix & Classification Report (Validation):

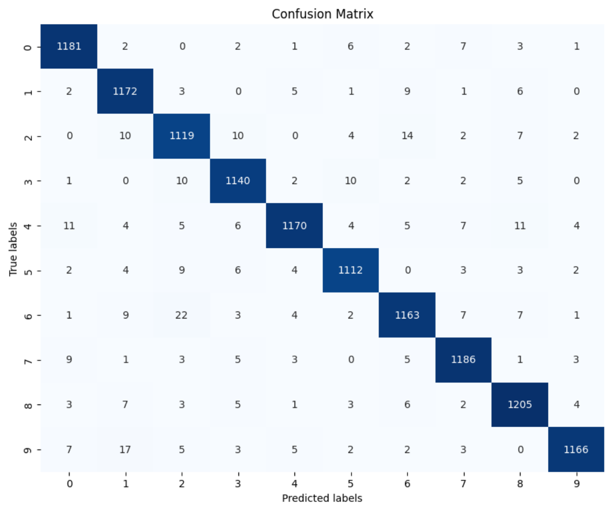


Figure 8: Confusion Matrix val-set exp-3

The confusion matrix showed high diagonal values, confirming that most samples are correctly classified, with only minor confusions among classes.

Class	Precision	Recall	F1-Score	Support
0	0.9704	0.9801	0.9752	1205
1	0.9560	0.9775	0.9666	1199
2	0.9491	0.9580	0.9536	1168
3	0.9661	0.9727	0.9694	1172
4	0.9791	0.9535	0.9661	1227
5	0.9720	0.9712	0.9716	1145
6	0.9627	0.9541	0.9584	1219
7	0.9721	0.9753	0.9737	1216
8	0.9655	0.9726	0.9690	1239
9	0.9856	0.9636	0.9745	1210

Precision, recall, and F1-scores for each class were generally in the range of 0.96–0.98, indicating strong per-class performance with balanced metrics across all classes.

Test Accuracy: 92.26% on 10,000 samples.

Confusion Matrix & Classification Report (Testing):

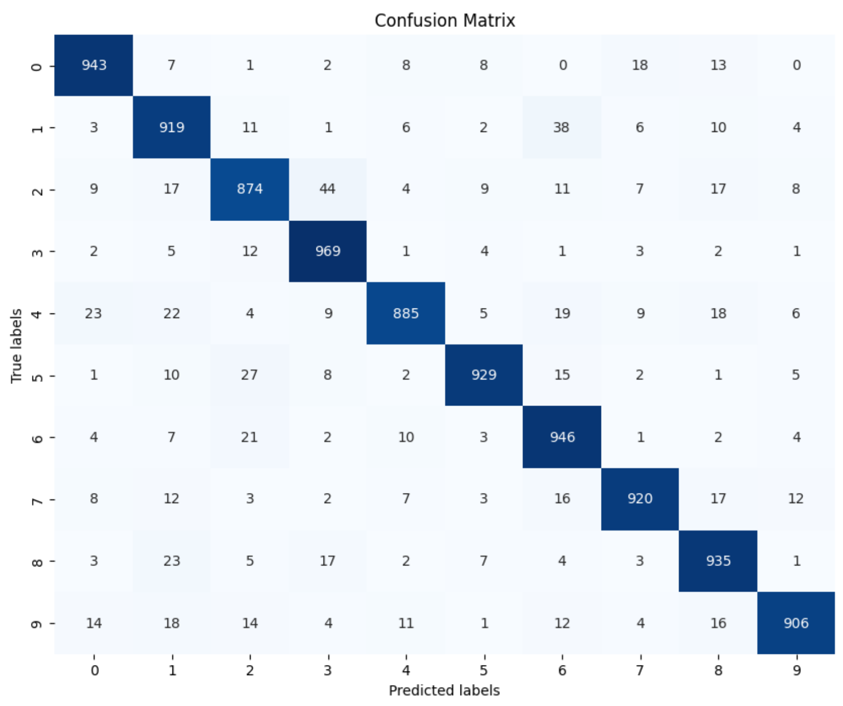


Figure 9: Confusion Matrix test-set exp-3

Similar to the validation results, the test confusion matrix confirmed strong classification performance overall, though the final accuracy is slightly lower compared to Experiment 2.

Table 4: Classification Matrix test-set exp-3

Class	Precision	Recall	F1-Score	Support
0	0.9337	0.9430	0.9383	1000
1	0.8837	0.9190	0.9010	1000
2	0.8992	0.8740	0.8864	1000
3	0.9159	0.9690	0.9417	1000
4	0.9455	0.8850	0.9143	1000
5	0.9567	0.9290	0.9427	1000
6	0.8908	0.9460	0.9176	1000
7	0.9455	0.9200	0.9326	1000
8	0.9069	0.9350	0.9207	1000
9	0.9567	0.9060	0.9307	1000

Overall metrics (precision, recall, F1-score) hovered around 0.92–0.94, with a few classes (e.g., class 1 and class 6) showing slightly lower precision or recall.

4. Analysis & Comparison

- **Strengths of Experiment 3:**
  - **Higher Model Capacity:** With 1024 and 512 neurons, the network had the potential to learn more complex patterns.

    - **Effective Learning:** Training loss reached very low values, showing that the model learned the training data well.
    - **Robust Per-Class Metrics:** The validation classification report demonstrated strong performance across all classes with high precision and recall.
- **Challenges and Trade-offs:**
    - **Over-regularization:** The aggressive dropout rate of 0.5 in two large layers might have limited the model's ability to fully capitalize on its increased capacity.
    - **Validation-Test Gap:** While validation accuracy was reasonably high at 96.78%, the test accuracy dropped to 92.26%. This discrepancy suggests either some degree of over-regularization or a potential mismatch in data distribution between the validation and test sets.
- **Comparison to Experiment 2:**
    - Experiment 2 (with a simpler architecture and lower dropout of 0.2) achieved slightly better validation performance (98.56%) and a comparable test accuracy (92.48%).
    - The increased network size in Experiment 3 did not translate into higher test performance, possibly due to over-regularization.

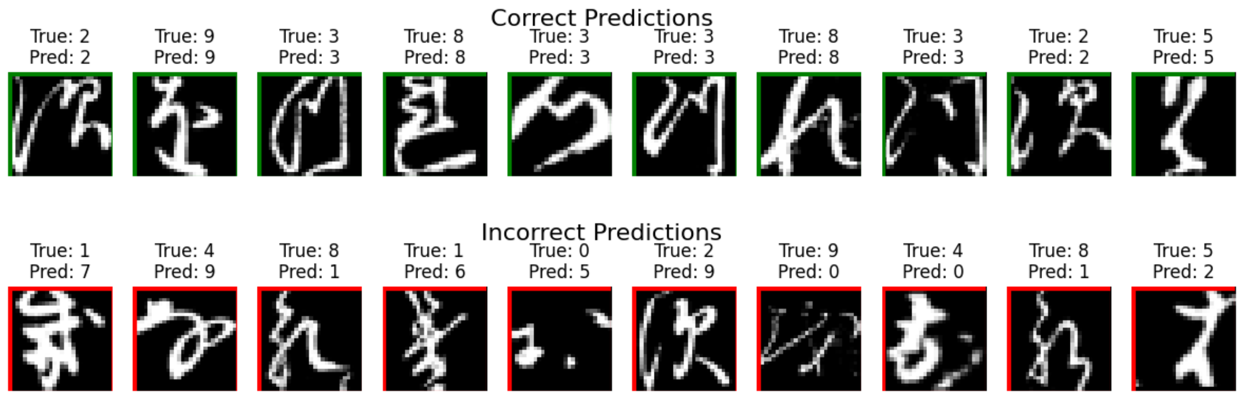
- **Potential Improvements:**
    - **Tuning Dropout Rate:** Experimenting with a slightly lower dropout rate (e.g., 0.3 or 0.4) might allow the network to leverage its extra capacity better while still preventing overfitting.
    - **Hyperparameter Optimization:** Further tuning of learning rate, weight decay, or even the number of epochs could help narrow the gap between validation and test performance.
    - **Data Augmentation:** Additional data augmentation techniques could also improve generalization, especially given the higher capacity of the network.

5. Summary

Experiment 3 introduced a larger network with 1024 and 512 neurons and increased dropout of 0.5, along with weight decay for regularization. The key findings are:

- **Training:** The network learned quickly and achieved very low training loss.
- **Validation:** The validation accuracy reached 96.78% with strong per-class metrics, though the loss oscillated mildly after initial convergence.
- **Testing:** Test accuracy was 92.26%, slightly lower than Experiment 2, suggesting that the aggressive regularization may have limited the model's full potential.
- **Overall:** While the expanded architecture showed promise, its performance was comparable to previous experiments, highlighting the delicate balance between model capacity and regularization.

Visualization



These images provide a brief overview of the performance of the Experiment 3 model on specific samples. The model's confident and precise classifications are displayed in the green-bordered images, which demonstrate how effectively it has learned to differentiate a wide variety of characters. The red-bordered pictures, meanwhile, draw attention to places where the model makes mistakes, frequently on figures that have minor differences or are visually similar. This mixture of accurate and inaccurate predictions serves as a reminder that even very effective models may still have trouble with specific handwriting quirks and offers suggestions for more data augmentation or improvement.

Conclusion

To sum up the KMNIST project gave helpful information about building useful neural networks. These systems recognize written Japanese letters. The project started with a simple model in Experiment 1. It was a firm base because the structure was straightforward and the test gave roughly 92.54 % correct answers.

With Experiment 2, scientists put weight decay into effect and the education centered on validation. This caused the validation set to get much better, with a 98.56 % best answer. This shows why adjusting plus checking models carefully are important.

But Experiment 3 tried to do even better. It grew the network and used a stronger dropout rate. Even though the large design learned the education data well, its test answers were near the simpler models. This shows the over-adjusting reduced what it could do - this set of tests stressed the need to find the ideal balance between model capacity and regularization.