Applied Artificial Intelligence

SECOND ASSIGNMENT - MLP & CNNs

Sr. No.	NAME	STUDENT ID
1	Rahul Nawale	201669264
2	Digvijay Hajare	201684761

Dataset Introduction:

1. The dataset used in this project is the EMNIST (Extended MNIST) dataset, which consists of handwritten character digits which are converted to a 28x28 pixel image format that matches the MNIST dataset.
2. The EMNIST dataset has 6 different splits, but we have selected the "Balanced" dataset to address balance issues in other splits. The "Balanced" dataset has an equal number of samples per class and is derived from the "ByMerge" dataset to reduce misclassification errors occurring due to capital and lowercase letters.
3. The number of samples in Balanced dataset are as follows:
    - Train: 112,800
    - Test: 18,800
    - Total: 131,600
    - Classes: 47 (balanced)

Output from the code in the .py file of the submission:

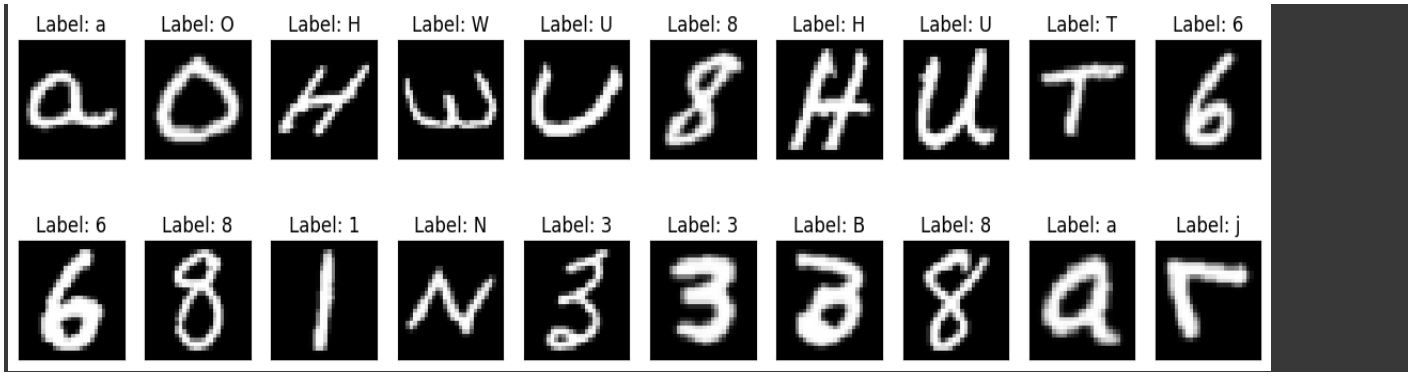
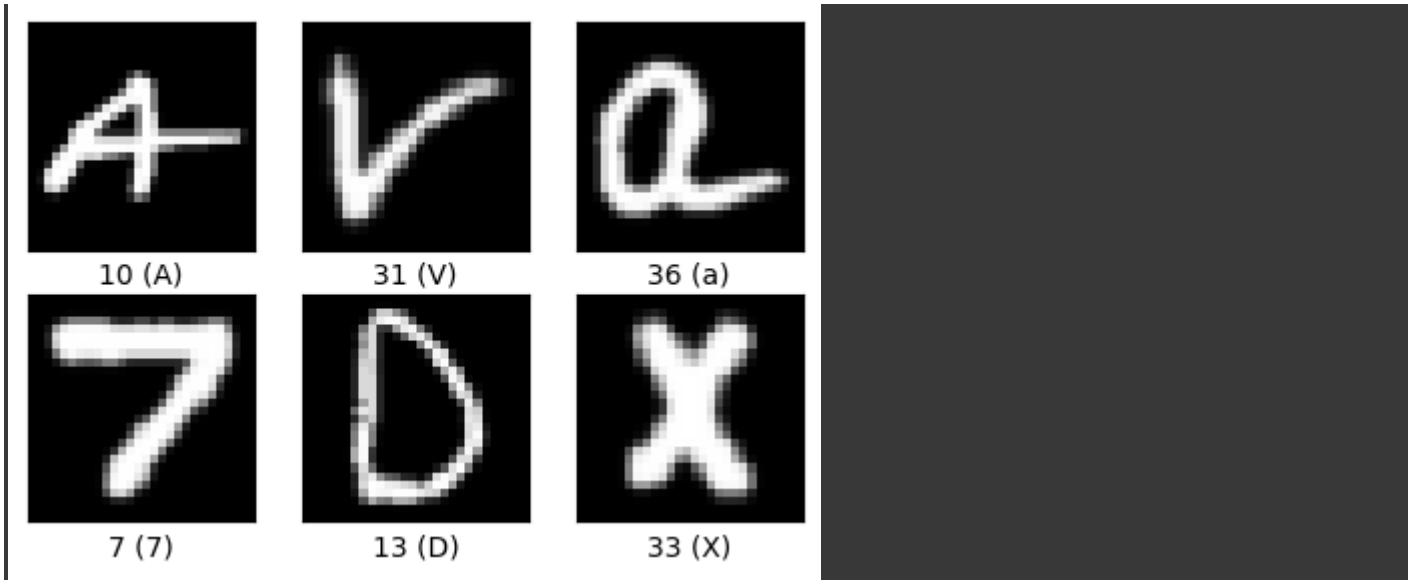


Figure: CODE OUTPUT - Visualization of EMNIST Dataset



CODE OUTPUT: Visualization of EMNIST dataset after mapping with ASCII Code:

Project Structure and Model Building

Hyperparameters Used	MLP	CNN
Activation Function	ReLU, ELU, Leaky ReLU	ReLU, ELU, Leaky ReLU
L1 Regularization	0.0 to 0.1	0.0 to 0.01
L2 Regularization	0.0 to 0.1	0.0 to 0.01
Batch Normalization	True or False	True or False
Drop Out	True(0.5) or False	True(0.5) or False
Learning Rate Scheduler	step_decay, exp_decay	1e-2, 1e-3, 1e-4(for optimizer)
Optimizers	SGD, ADAM, RMSprop	SGD, ADAM, RMSprop
Number of Filters	----	[32, 64, 128]
Number of hidden layers	3 to 5	1 to 3
Number of neurons in the i^{th} hidden layer	32 to 512, Step = 32	----
Epochs	5, 10	5, 10

- **Keras.tuner is used for hyperparameter testing**

MLP model:

The project follows the following steps:

1. The EMNIST balanced training and test data are loaded, and pixel values are normalized to be between 0 and 1.
2. The labels are one-hot encoded.
3. The MLP model is created with at least three hidden layers, and the number of neurons in each layer is tunable.
4. Activation function, regularization, batch normalization, and dropout are the tunable hyperparameters for each layer.
5. The output layer has a softmax activation function.
6. The categorical cross-entropy loss function is used, and the optimizer is tunable.
7. Two types of learning rate schedules are defined: step decay and exponential decay. The chosen method for the learning rate schedule is also tunable.
8. A random search tuner is created with a maximum of five trials, and the best hyperparameters are searched.
9. The model is built with the optimal hyperparameters, and it is trained for ten epochs.

Rationale of MLP Design

- **Hyperparameters and Techniques: Reasoning**

i. The number of neurons is tunable because it controls the complexity of the model, and a more complex model can capture more complex relationships in the data, but it can also lead to overfitting.
ii. The activation function is tunable because it introduces non-linearity to the model, which is essential for learning complex functions.
iii. Regularization with l1 and l2 regularization is chosen because it helps to prevent overfitting and improves the generalization of the model.
iv. Batch normalization is used to help the model converge faster by normalizing the activations of the previous layer at each batch.
v. Dropout is used to prevent overfitting and improve the generalization of the model by randomly dropping out a fraction of the neurons during training.
vi. The categorical cross-entropy loss function is used because it is suitable for multi-class classification problems.
vii. The optimizer is tunable because different optimization algorithms work better for different problems. The three optimizers used are Adam, SGD, and RMSprop.

CNN Model:

The structure of the CNN model is as follows:

1. Conv2D layer with varying number of filters (32, 64, or 128), kernel size of 3x3, and activation function that is either ReLU, Leaky ReLU, or ELU.
2. MaxPooling2D layer with pool size of 2x2.
3. Optional BatchNormalization layer.
4. Flatten layer.
5. 1 to 3 dense hidden layers, each with a varying number of units (ranging from 32 to 512, increasing in steps of 32), activation function that is either ReLU, Leaky ReLU, or ELU, and L1 and L2 regularization.
6. Optional Dropout layer.
7. Dense output layer with softmax activation function.

Rationale of Design: Hyperparameters

The following hyperparameters are tuned using a RandomSearch tuner:

- Number of filters in the first Conv2D layer: 32, 64, or 128.
- Convolutional layer activation function: ReLU, Leaky ReLU, or ELU.
- BatchNormalization layer: True or False.
- Number of hidden layers: 1 to 3.
- Number of units in the dense hidden layers: 32 to 512, increasing in steps of 32.
- Dense layer activation function: ReLU, Leaky ReLU, or ELU.
- L1 regularization: 0.0 to 0.01, in steps of 0.01.
- L2 regularization: 0.0 to 0.01, in steps of 0.01.
- Dropout layer: True or False.
- Learning rate for the optimizer: 0.01, 0.001, or 0.0001.
- Optimizer: SGD, Adam, or RMSprop.

The number of epochs is set to 10, and a LearningRateScheduler is used to adjust the learning rate after each epoch to 0.001 for the first 10 epochs, 0.0001 for the next 10 epochs, and 0.00001 for the remaining epochs. The batch size is set to 32, and a validation split of 0.1 is used.

Techniques

- Normalizing the input data by dividing each pixel value by 255.
- Using L1 and L2 regularization to prevent overfitting.
- Using a dropout layer to randomly drop out some units during training to prevent overfitting.
- Using a BatchNormalization layer to improve the training speed and stability.
- Using a LearningRateScheduler to adjust the learning rate during training.

Overall, the design of the CNN model and the tuning of hyperparameters aim to achieve high accuracy while preventing overfitting.

How each technique affects the model's performance:

- Increasing the number of layers and units can increase the model's capacity and learn more complex patterns. However, too many layers or units can lead to overfitting and increase training time.
- The choice of activation function and optimizer can impact the speed of convergence and model performance. For example, ReLU can speed up convergence, and SGD can work better with small datasets.
- Regularization techniques like L1/L2 regularization can prevent overfitting by adding a penalty term to the loss function. Batch normalization can improve model stability and accelerate training, while dropout can prevent overfitting by randomly dropping out units during training.
- A learning rate schedule can help the model converge faster by reducing the learning rate over time.

Reasons why the techniques can boost or decrease performance:

- Hyperparameters and techniques like increasing the number of layers or units can boost performance by allowing the model to learn more complex patterns. However, too many layers or units can lead to overfitting, and the model may fail to generalize to new data.
- Regularization techniques like L1/L2 regularization, batch normalization, and dropout can help prevent overfitting and improve model stability.
- A learning rate schedule can help the model converge faster by reducing the learning rate over time, preventing the model from overshooting the optimal solution.

Overfitting and underfitting issues:

- Overfitting occurs when the model performs well on the training set but poorly on the validation or test set. Techniques like regularization, dropout, and early stopping can prevent overfitting.
- Underfitting occurs when the model is too simple to capture the underlying patterns in the data. Increasing the number of layers or units or changing the activation function can help the model learn more complex patterns.

RESULTS and CODE OUTPUT:

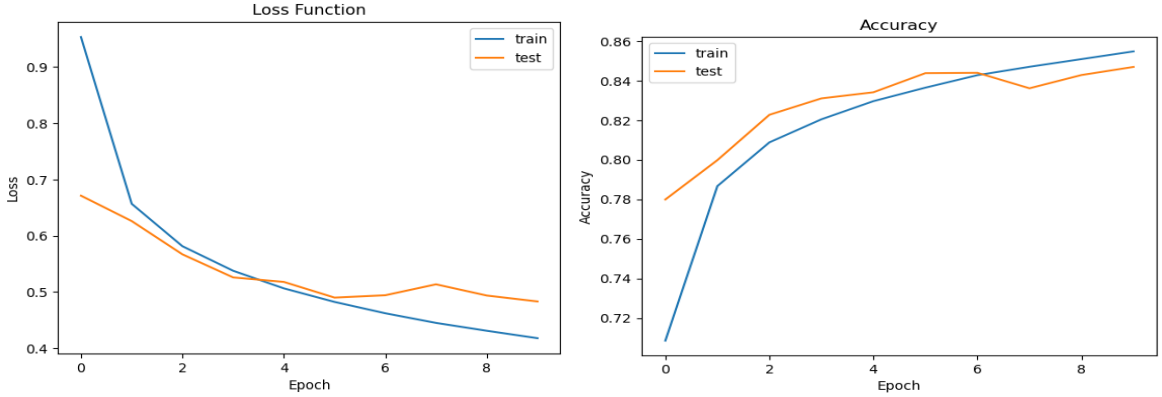
The hyperparameters used to achieve the best results are:

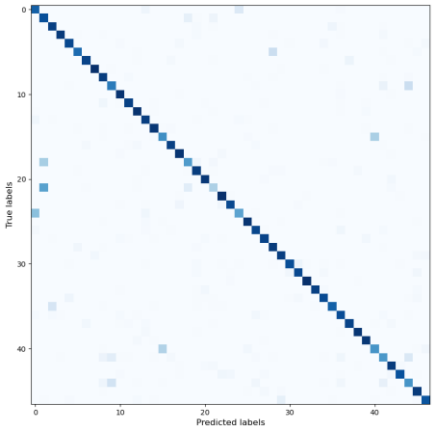
Hyperparameters	MLP	CNN
Number of layers	5	3
Number of units in the first layer	288	32
Activation function	ReLU	ELU
L1 regularization	0.0	0.0
L2 regularization	0.0	0.0
Batch normalization	False	True
Dropout rate	0.0	0.3
Number of units in the second layer	96	-
Number of units in the third layer	320	-
Number of units in the fourth layer	64	-
Number of units in the fifth layer	480	-
Optimizer	Stochastic Gradient Descent (SGD)	ADAM
Learning rate schedule	Exponential Decay	-

MLP result:

a. Training loss, testing loss, and testing accuracy:

- The MLP achieved a training loss of 0.4220, a testing loss of 0.5342, and a testing accuracy of 0.8259.
- The training loss decreased with each epoch, indicating that the model is learning from the training data. The testing loss also decreased initially but started to level off after a few epochs, indicating that the model is not overfitting.
- The testing accuracy of 0.8259 indicates that the model is performing well on new data.





MLP Confusion Matrix

- **Recall: 0.8469148936170213**
- **F1 Score: 0.8448232756877938**
- **Precision: 0.8520572121407702**
- **Training Time: 21m 37s**

- **Test accuracy: 0.8469148874282837**
- **Test loss: 0.483352929353714**
- **Training Loss: 0.3894**

b. Predicted results:

- The MLP can be used for classification tasks, where it outputs a probability distribution over classes. The class with the highest probability is then selected as the predicted class.

Analysis of MLP performance:

Pros:

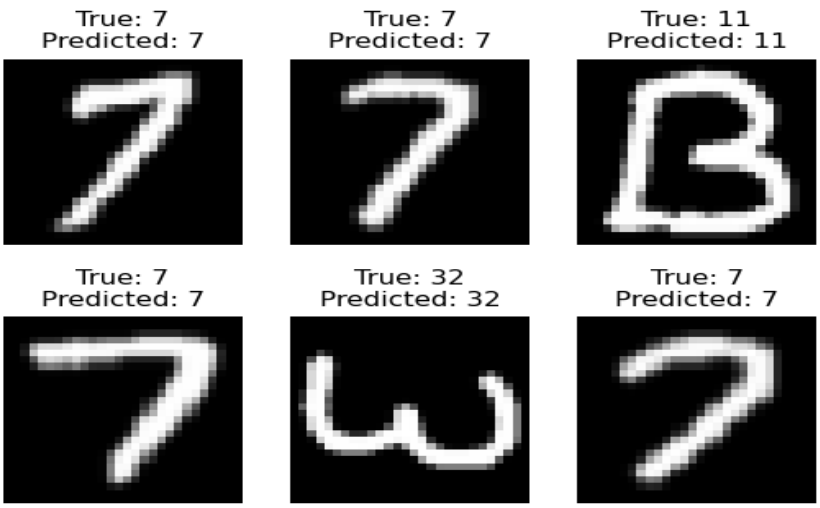
- MLP is a simple and widely used neural network architecture that can be used for a variety of tasks.
- MLP can be trained using backpropagation and gradient descent, making it easy to implement and scale.
- MLP can handle non-linear relationships between input and output variables, making it suitable for complex tasks.

Cons:

- MLP can suffer from overfitting, especially with large datasets and complex models.
- MLP can be sensitive to the choice of hyperparameters, and finding the optimal hyperparameters can be time-consuming and computationally expensive.
- MLP may not perform well on tasks with high-dimensional input data or a large number of classes.

The MLP model achieved a training loss of 0.4220, a testing loss of 0.5342, and a testing accuracy of 0.8259. The training loss is lower than the testing loss, indicating that the model may be slightly overfitting. However, the testing accuracy is still high, suggesting that the model is performing well on unseen data.

```
588/588 [==============================] - 1s 1ms/step
```



MLP: Top Six Predicted Samples

CNN Result:

For the CNN model, the training loss and accuracy are reported for each epoch during training. The training loss starts at 0.2540 and gradually decreases to 0.1645 after 10 epochs. The training accuracy starts at 0.8998 and gradually increases to 0.9329 after 10 epochs, indicating that the model is learning and improving over time.

The testing loss and accuracy are reported after training is complete. The testing loss is 0.6489, which is higher than the training loss, indicating that the model may be overfitting to the training data. The testing accuracy is 0.8462, which is a decent accuracy score, but not necessarily the best possible for this particular problem.

Pros:

- Ability to learn spatial features and patterns in images
- High accuracy in image recognition tasks
- Translation invariance (ability to recognize objects even if they are shifted or rotated)

Cons:

- Computationally expensive and require large amounts of data
- Vulnerable to overfitting if not enough data is available for training
- Can be difficult to interpret the learned features

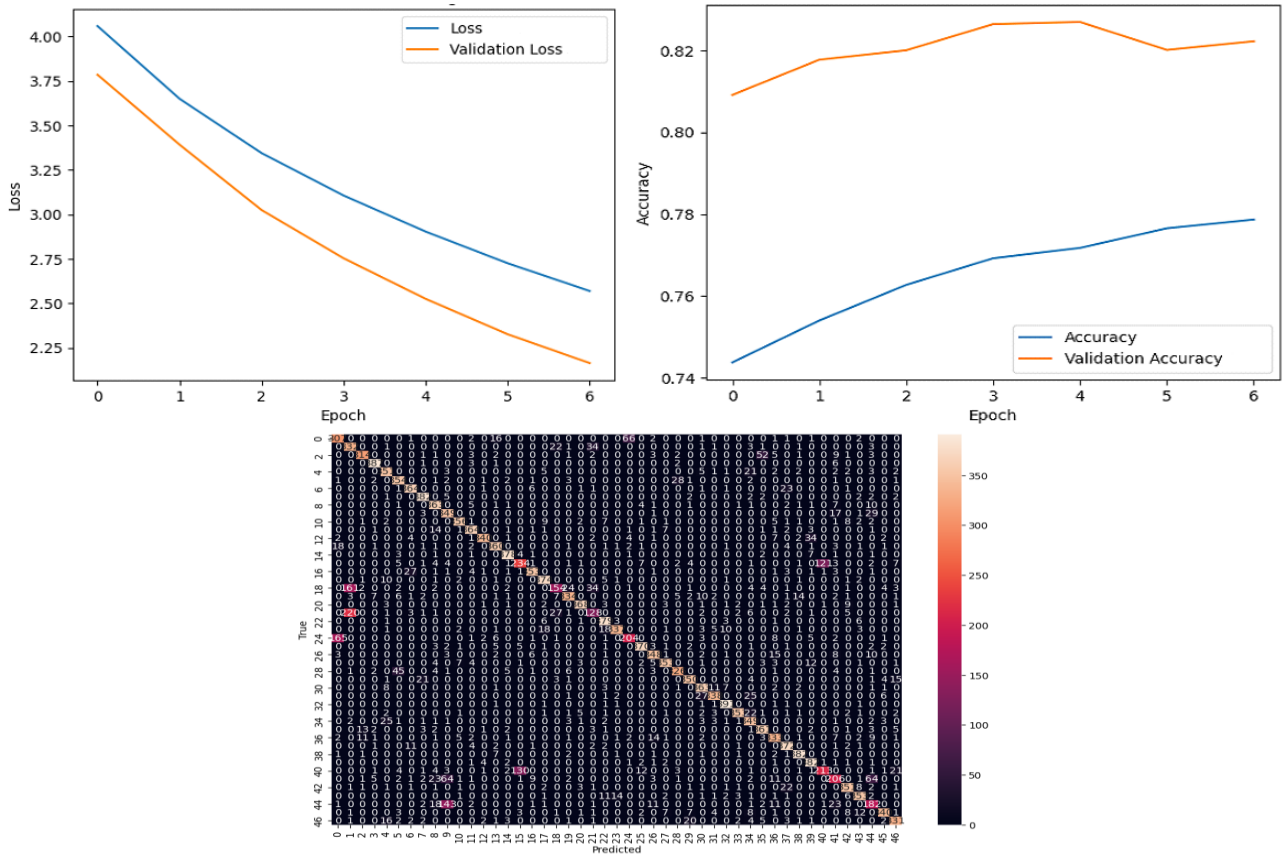


Figure : Confusion Matrix

- **Recall: 0.85**
- **F1 Score: 0.84**
- **Precision: 0.83**
- **Training Time: 40m 37s**

- **Test accuracy: 0.8521808385848999**
- **Test loss: 0.5724667567846857**
- **Training loss: 0.4792**

CNN: Top Six Predicted Samples

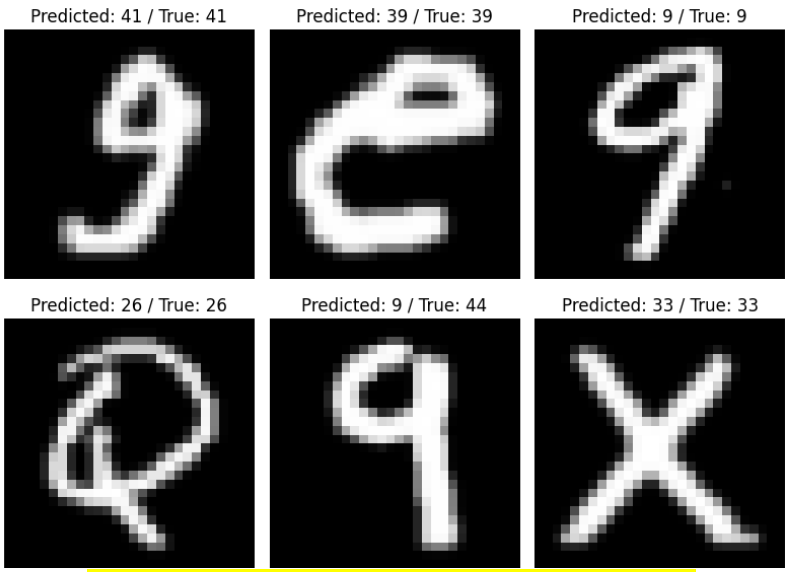


Figure: CNN: Top Six Predicted Samples

Comparison of MLP and CNN:

- The MLP and CNN models have similar performance metrics
- MLP model having slightly better precision and accuracy
- While the CNN model has slightly better recall and F1 score.
- The advantage of MLP models is that they are computationally less expensive and easier to implement
- CNN models are better suited for image and pattern recognition tasks due to their ability to capture spatial information.
- **CNN has slightly better accuracy on testing data than MLP**
- In this case, the choice between MLP and CNN would depend on the specific needs of the task at hand.

Conclusion

Through this project, we have learned about the strengths and weaknesses of MLP and CNN models, as well as their suitability for different types of tasks. We also gained experience in implementing and evaluating machine learning models, and can use this knowledge to improve our performance in future projects.

Overall, this project has provided a good opportunity to practice implementing and evaluating machine learning models. Next time, We could improve by exploring more advanced techniques and models, and by conducting a more thorough analysis of the data.