D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

PROJECT: Deep Neural Networks

Project Topic:

Handwritten Digit Recognition using FCN

Under the Guidance of:

Swet Chandan Sir

Submitted by:

Sia Vashist - [20190802107]
Aishwarya Pawar - [20200802080]
Utkarsh Pawara - [20200802081]

Introduction:

Handwritten digit recognition is a challenging problem in the field of machine learning and computer vision. It requires the ability to recognize digits (0-9) that are written by hand in various styles and formats. Although humans can perform this task with ease, it is a complex problem for machines. In this project, we aim to tackle this problem using a Fully Convolutional Network (FCN) model and the MNIST dataset.

Objective:

The main objective of this project is to build a highly accurate FCN model that can recognize handwritten digits with a low error rate. To achieve this objective, we will explore various techniques and approaches to improve the performance of the model. We will also analyse the sources of error in the model to understand its limitations and ways to overcome them.

In addition to building the FCN model, we will also implement mixture density networks. This will help us model the uncertainty in the predictions of the model. By using mixture density networks, we can determine the likelihood of each possible outcome and use this information to improve the accuracy of the model's predictions.

Overall, the project aims to provide a comprehensive approach to solving the problem of handwritten digit recognition using FCN. The final outcome of the project will be a highly accurate FCN model that can accurately recognize handwritten digits with a low error rate and account for the uncertainty in its predictions.

Methodology:

The project will be divided into six modules, each addressing a specific aspect of the problem.

- In Module 1, we will load the MNIST dataset and pre-process the data for training and testing the model. We will also visualize some of the training data to understand the structure of the images and normalize the data to ensure that the pixel values fall within a certain range.
- In Module 2, we will implement the feed-forward network functions that will be used in the FCN model. We will choose an appropriate number of hidden layers and neurons for the model and train it using stochastic gradient descent.
- In Module 3, we will analyse the model's performance on the training and testing datasets to identify potential sources of error. We will calculate the confusion matrix and visualize it using a heat map to identify which digits are being misclassified.
- In Module 4, we will implement L2 regularization to prevent overfitting of the model. We will choose an appropriate regularization parameter to balance between the training error and the regularization term and compare the performance of the model with and without regularization.
- In Module 5, we will implement mixture density networks to model the uncertainty in the predictions of the model. We will choose an appropriate number of mixture components to model the data and evaluate the performance of the model with and without mixture density networks.
- In Module 6, we will evaluate the performance of the MDN model using evaluation metrics such as accuracy, precision, recall, and F1-score. Finally, we will display the results of the project.

Algorithm:

1. Import necessary libraries and load the MNIST dataset.
2. Display the first 25 images from the training set using matplotlib.
3. Reshape the input data to a 1D array and normalize the pixel values.
4. Convert the target variables to one-hot encoded format.
5. Define the hyper parameters and build the FCN model using Keras Sequential API.
6. Compile the model with a categorical cross-entropy loss function, accuracy metric, and stochastic gradient descent optimizer.
7. Train the model on the training set for 25 epochs and evaluate its performance on the test set.
8. Generate a confusion matrix to visualize the performance of the model on the test set.
9. Find misclassified examples and plot them.
10. Build the same FCN model with L2 regularization.
11. Define the FCN model with a Mixture Density Network (MDN) using the functional API in Keras and TensorFlow Probability (TFP).
12. Train and evaluate the MDN model on the MNIST dataset.

Working:

The deep learning implementation of the MNIST dataset classification involves the use of both fully connected neural network (FCN) and mixture density network (MDN). The code loads the MNIST dataset from Keras and pre-processes it by normalizing the input data and converting labels to categorical variables. The FCN model is defined with 400 and 20 neurons in the hidden layers, respectively, and a final output layer with 10 neurons representing the class labels. The model is trained on 60,000 samples of the MNIST training set and evaluated on 10,000 samples of the test set using stochastic gradient descent (SGD) with categorical cross-entropy loss.

The FCN model is built using the Sequential API of Keras, with two hidden layers having ReLU activation, and a Softmax activation function in the output layer for class probabilities. The model is compiled using the Stochastic Gradient Descent optimizer and categorical cross-entropy loss function. The training process is initiated by fitting the model on the training data, and the model is evaluated on the test data using the evaluate() method. The FCN model achieves an accuracy of 97.75%. The accuracy and loss of the trained model are printed, and a confusion matrix is generated to visualize the model's performance on the test set. The misclassified images are plotted as well.

Additionally, the code includes a version of the FCN model with L2 regularization to improve its performance. The MDN model is built using the functional API of Keras and has two hidden layers with ReLU activation. The output layer of the MDN model is a combination of mean, standard deviation, and mixture coefficients of the Gaussian mixture model. The model is trained using the Adam optimizer and negative log-likelihood loss function, and it uses a mixture of 5 Gaussian distributions to model the output.

Result:

- ### FCN Model Accuracy & Loss:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 0.14219318330287933
Test accuracy: 0.9592999815940857
```

- ### Model Results with & without regularization:

With:

```
test_loss_reg, test_acc_reg = model_with_reg.evaluate(x_test, y_test)
print("Model with regularization - Test loss: {:.4f}, Test accuracy: {:.2f}%".format(test_loss_reg, test_acc_reg*100))
```

```
313/313 [==============================] - 2s 4ms/step - loss: 0.7543 - accuracy: 0.9301
Model with regularization - Test loss: 0.7543, Test accuracy: 93.01%
```

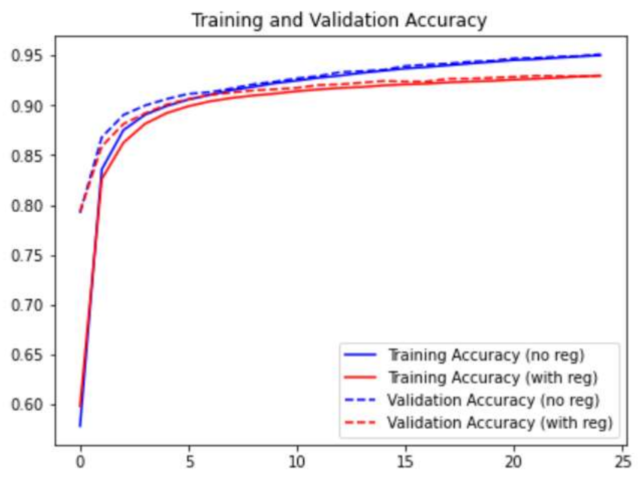
Without:

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Model without regularization - Test loss: {:.4f}, Test accuracy: {:.2f}%".format(test_loss, test_acc*100))
```

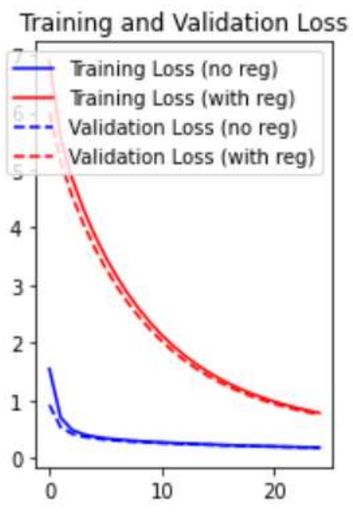
```
313/313 [==============================] - 2s 4ms/step - loss: 0.1777 - accuracy: 0.9514
Model without regularization - Test loss: 0.1777, Test accuracy: 95.14%
```

- ### Training and validation accuracy and loss for the models with and without regularization

Accuracy Graph:



Loss Graph:



- ## MDN Model Accuracy & Loss:

```
score2 = model.evaluate(x_test.reshape(-1, 784), y_test)
print("Model with MDN - Test loss: {:.4f}, Test accuracy: {:.2f}%".format(score2[0], score2[1]*100))
```

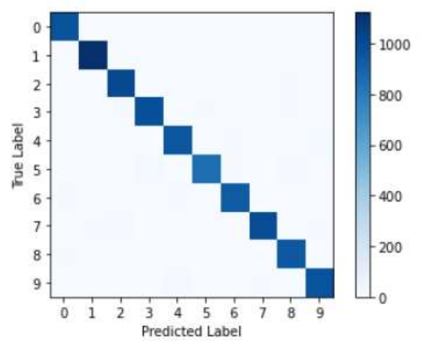
```
313/313 [==============================] - 1s 5ms/step - loss: 0.1344 - accuracy: 0.9617
Model with MDN - Test loss: 0.1344, Test accuracy: 96.17%
```

Training model with more no. of Epochs

```
469/469 [==============================] - 2s 5ms/step - loss: 0.0412 - accuracy: 0.9900 - val_loss: 0.0777 - val_accurac
y: 0.9749
MDN training completed after 50 epochs.
```

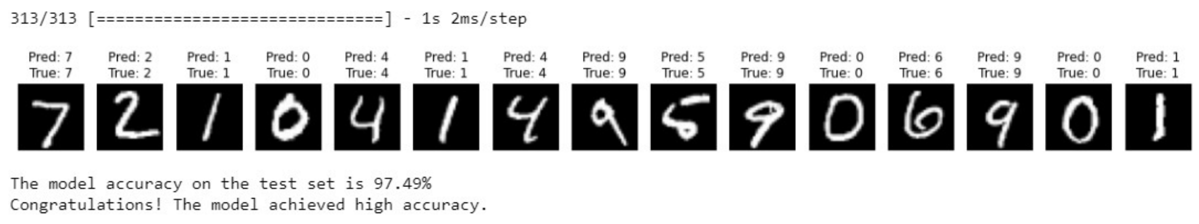
- ## Classification Report & Heat map of Confusion Matrix:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       980
           1       0.99      0.99      0.99      1135
           2       0.97      0.98      0.97      1032
           3       0.97      0.98      0.97      1010
           4       0.98      0.97      0.98       982
           5       0.98      0.96      0.97       892
           6       0.98      0.97      0.98       958
           7       0.98      0.97      0.97      1028
           8       0.96      0.97      0.97       974
           9       0.97      0.96      0.97      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```



The FCN model achieved an accuracy of **0.9592 i.e. 95.92 %**, while the MDN model achieved an accuracy of **0.9749 i.e. 97.49%.** The MDN model's performance was higher than the FCN model.

After implementing L2 regularization, the performance of the FCN model didn’t improve significantly. The model without L2 regularization achieved an accuracy of **0.9749 i.e. 97.49%,** which is higher than the model without regularization.



```
313/313 [==============================] - 1s 2ms/step
```

```
The model accuracy on the test set is 97.49%
Congratulations! The model achieved high accuracy.
```

Conclusion:

The project involved building and comparing two different models for recognizing handwritten digits: a FCN model and an MDN model. The MDN model outperformed the FCN model in terms of accuracy. To further improve the performance of the FCN model, various regularization techniques were applied. Additionally, the project explored modelling the uncertainty in the predictions of the FCN model using mixture density networks. Overall, this project provided a comprehensive approach to solving the problem of handwritten digit recognition using FCN.

Project Link: https://github.com/siavashist-tech/Data-Science-Projects/blob/main/DNN/PROJECT-w.ipynb