Project 1 Report

Course Code: COMP 8720
Course Title: Deep Learning for Vision & NLP
Course Instructor: Dr. Robin Gras

Submitted by:
Siyam Sajnan Chowdhury (110124636)
Kimia Tahayori (110124141)

Dataset

The dataset that we have used are the MNIST dataset. It contains two separate dataset – the digits dataset and the fashion dataset.

The MNIST digits dataset contains 70,000 handwritten digits from 0 through 9. Each of the digits have been written in thousands of different ways by many different people. It is one of the most popular datasets for training ML models especially in the domain of computer vision for digit and document recognition tasks. Each of the images are of resolution 28x28 and are grayscale.

The next dataset is the Fashion-MNIST dataset contains 10 different types of attire. There are also 70,000 different images of these 10 classes of dresses and is widely used for image classification with regards to fashion and clothing. These images are also of 28x28 resolution and are grayscale.

Model

The model that we used for this project is the Convolutional Neural Network or CNN model. It is a type of deep neural network that is created to process image data in a way that is similar to how the visual architecture works in humans and this is very helpful in image classification and recognition tasks.

The core unit of CNN are its convolutional layers which implements filters, or kernels of specific preset sizes that is learnable in order to recognize features such as edges or textures. These are moved around all throughout the image data to help figure out the features. The next layer is the pooling layer which reduces the spatial dimension by downsampling the data.

Then comes the fully connected, or dense layers towards the last stage of the CNN which helps to aggregate all the abstract features and use them to predict or classify from sample data. Non-linear activation or transfer functions are used up to this stage.

The training and adjusting of the parameters are done via backpropagation until a minimum of the loss function is reached.

Explanation of the code implementation for CNN on the MNIST dataset

We have implemented CNN on both the MNIST fashion dataset and the MNIST digits dataset.

We started off by importing the required libraries such as numpy, tensorflow, sklearn and seaborn. We then loaded the datasets from tensorflow's keras.datasets. We then created a function to plot the sample data named "display_sample" which takes in 3 arguments – images, labels and class_name. We set subplots as the number of unique labels and printed the labels and the images using matplotlib. The sample data printed are as follows:

We then preprocessed the data to fit it to our model. We reshaped the 28x28 2d images to 3d tensors of size 28x28x1 with the 1 indicating that the images are grayscale. We also divide it by 255 to normalize it bring the values between 0 and 1. We did this for both the test and train images for the digits and fashion dataset.

We then defined the CNN architecture that we will be using for our MNIST datasets digits and fashion. We created a function defining the model named create_cnn_model where we created a variable model and used the keras.Sequential function of the tensorflow library to define the individual layers in the CNN architecture.

For the first layer, we used 32 filters each being of size 3x3 and used the ReLU activation function. We set the input shape of our preprocessed data – 28x28x1. We also utilized a MaxPooling layer of size 2x2 in order to downsample the large amount of input data by also retaining the most crucial information.

For the second layer, we used 64 filters each of size 3x3 as before and we used ReLU as the activation function or the transfer function. We created another MaxPooling later of the same size as before – 2x2 to further downsample the data from this layer.

We then used a flattening layer in order to transform the 3d output from the second layer into a flat one-dimensional vector. We used this as we are transitioning from a convolutional layer from before to a fully connected or dense layer in the next step.

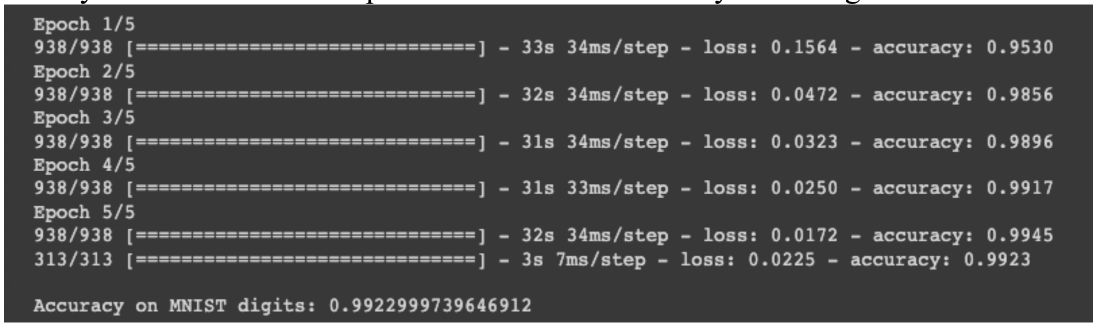
Followed by the flattening layer, we used a dense layer to aggregate the information from the convolutional and pooling layers in order to combine them and preparing them for input to the final layer to give the output. We used 128 neurons with the ReLU activation function. It is a fully connected layer.

Finally, we created a final dense layer to feed from the previous dense layer and give the output. We used 10 units and used the softmax activation function to give the prediction.

We then compiled the model and we used Adam optimizer which is a gradient-based algorithm, we used sparse_categorical_crossentropy as our loss function as we are working with a multi-class classification problem and we set the training monitoring metric as accuracy.

We then created two models using the create_cnn_model function – one for the digits dataset and one for the fashion dataset. We fit the two datasets the same way: we used the training images and training labels as our input and ran it over 5 epochs and a batch size of 64. After running the model, we used the evaluate function to evaluate the accuracy of our model by feeding it the test_images and test_labels data. We then printed the test accuracy of our CNN model on the two datasets.

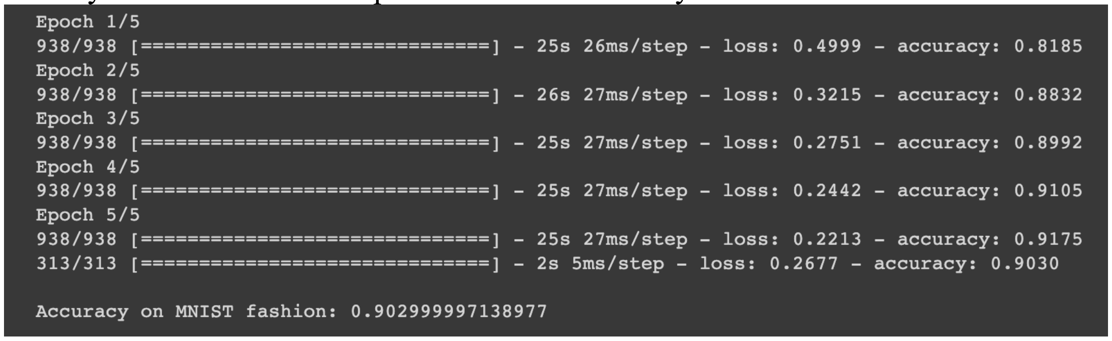
The accuracy and loss over the 5 epochs and the final accuracy of the digits dataset is as follows:



```
Epoch 1/5
938/938 [==============================] - 33s 34ms/step - loss: 0.1564 - accuracy: 0.9530
Epoch 2/5
938/938 [==============================] - 32s 34ms/step - loss: 0.0472 - accuracy: 0.9856
Epoch 3/5
938/938 [==============================] - 31s 34ms/step - loss: 0.0323 - accuracy: 0.9896
Epoch 4/5
938/938 [==============================] - 31s 33ms/step - loss: 0.0250 - accuracy: 0.9917
Epoch 5/5
938/938 [==============================] - 32s 34ms/step - loss: 0.0172 - accuracy: 0.9945
313/313 [==============================] - 3s 7ms/step - loss: 0.0225 - accuracy: 0.9923

Accuracy on MNIST digits: 0.9922999739646912
```

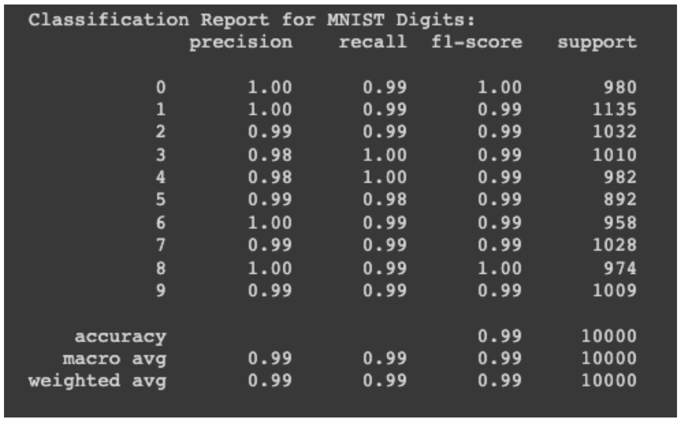
The accuracy and loss over the 5 epochs and final accuracy of the fashion dataset is as follows:



```
Epoch 1/5
938/938 [==============================] - 25s 26ms/step - loss: 0.4999 - accuracy: 0.8185
Epoch 2/5
938/938 [==============================] - 26s 27ms/step - loss: 0.3215 - accuracy: 0.8832
Epoch 3/5
938/938 [==============================] - 25s 27ms/step - loss: 0.2751 - accuracy: 0.8992
Epoch 4/5
938/938 [==============================] - 25s 27ms/step - loss: 0.2442 - accuracy: 0.9105
Epoch 5/5
938/938 [==============================] - 25s 27ms/step - loss: 0.2213 - accuracy: 0.9175
313/313 [==============================] - 2s 5ms/step - loss: 0.2677 - accuracy: 0.9030

Accuracy on MNIST fashion: 0.902999997138977
```

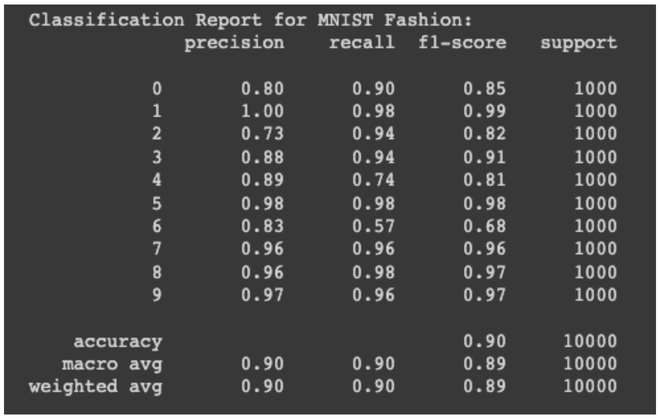
We managed to obtain an accuracy of 99.23% for our digits dataset and an accuracy of 90.30% for our fashion dataset. The classification report for our datasets are as follows:



```
Classification Report for MNIST Digits:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       980
           1       1.00      0.99      0.99      1135
           2       0.99      0.99      0.99      1032
           3       0.98      1.00      0.99      1010
           4       0.98      1.00      0.99       982
           5       0.99      0.98      0.99       892
           6       1.00      0.99      0.99       958
           7       0.99      0.99      0.99      1028
           8       1.00      0.99      1.00       974
           9       0.99      0.99      0.99      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

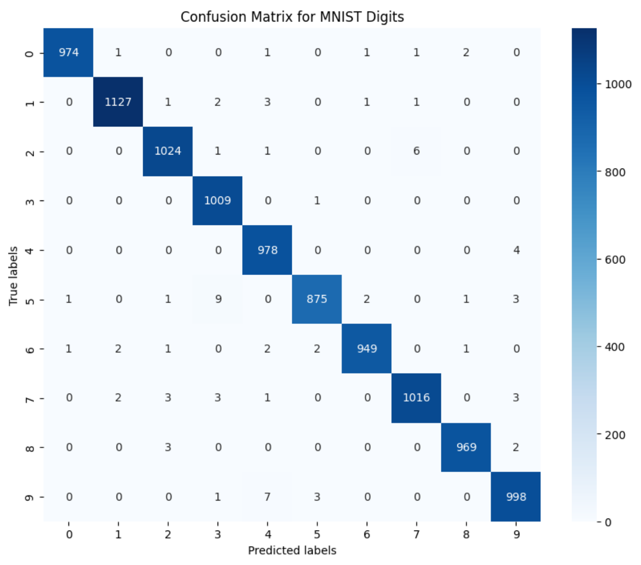


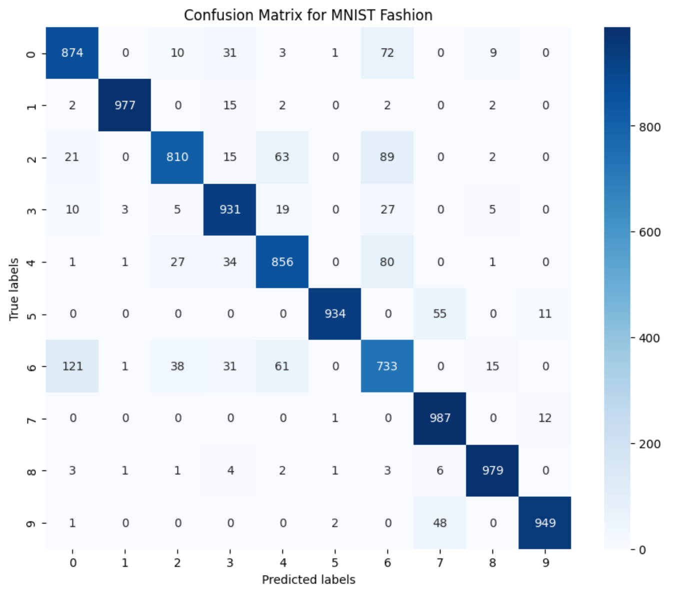
```
Classification Report for MNIST Fashion:
              precision    recall  f1-score   support

           0       0.80      0.90      0.85      1000
           1       1.00      0.98      0.99      1000
           2       0.73      0.94      0.82      1000
           3       0.88      0.94      0.91      1000
           4       0.89      0.74      0.81      1000
           5       0.98      0.98      0.98      1000
           6       0.83      0.57      0.68      1000
           7       0.96      0.96      0.96      1000
           8       0.96      0.98      0.97      1000
           9       0.97      0.96      0.97      1000

    accuracy                           0.90     10000
   macro avg       0.90      0.90      0.89     10000
weighted avg       0.90      0.90      0.89     10000
```

We then plotted the confusion matrix for the two datasets separately using the sklearn's confusion matrix function. The two confusion matrices are as follows:





It can be seen from the confusion matrix that the principal diagonal has a very deep color compared to the rest showing that the predictions were highly accurate. There were negligible misclassifications for the MNIST digits dataset.

For the fashion dataset, it can be seen that class 6 (shirts) was the one that was misclassified the most. It was mostly misclassified as class 0 (T-shirts), class 2 (pullover), class 3 (dress) and class 4 (coat). It is quite obvious why as these clothing types have quite a lot of similarity with shirts which has somewhat obscured the detection capabilities of the model.

<u>Performance comparison with existing papers</u>

The first paper that we would like to compare our paper to is "Convolutional Neural Network (CNN) for Image Detection and Recognition" by Chauhan et al [1]. They had achieved an accuracy of 99.6%. This is very similar to our project. They used a very similar architecture as we did. They used 5 convolutional layer as opposed to 2 convolutional layer in our project. They used dropout layers as well in order to reduce overfitting and also used 10 epochs. During model compilation, they used RMS prop optimizer as opposed to Adam optimizer that we used.

The next paper we compared to was "Gradient-Based Learning Applied to Document Recognition" by LeCun et al. [2]. They used LeNet-1, which is a relatively small convolutional network model which downsampled the images to the size 16x16, and had 4 convolutional layer and instead of MaxPooling, they used Average pooling. They achieved an accuracy of 98.3% on the MNIST dataset.

LeCun et al. also used LeNet-4 and LeNet-5 where they got accuracies of 98.9% and 99.3% respectively.

The last paper we compared to was "An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition" by An et. Al. [3] They achieved a 99.87% accuracy by using an ensemble of 3 CNN models with varying kernel sizes of 3x3, 5x5 and 7x7 and used a voting classifier. They also achieved a 99.91% accuracy using a multi-layer ensemble of the 3 heterogeneous ensembles.

Conclusion

Based on our experiments, our simple CNN architecture performed very well with a relatively shallow neural network with just 2 convolutional layers. The fashion dataset managed an accuracy of 90% which is quite good. The digits dataset managed to predict with over 99% accuracy which is quite astounding given the number of layers used. If the model was made deeper with more layers and connections, or ensemble was used, it could perform even better as seen from the experiments in [3].

Therefore, the potential for CNN in image recognition can be said to be immense.

References

[1] Chauhan, R., Ghanshala, K.K., Joshi, R.C., "Convolutional Neural Network (CNN) for Image Detection and Recognition" *First International Conference on Secure Cyber Computing and Communication (ICSCCC), Kolkata, India*, pp. 278–282, 2018.

[2] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 pp. 2278-2324, 1998.

[3] An, Sanghyeon, et al. "An ensemble of simple convolutional neural network models for MNIST digit recognition." *arXiv preprint arXiv:2008.10400* (2020).

Contributions

Each member contributed equally to both the code and the project report.