# COMP9417 Assignment 2: Character Recognition (Topic 1.2: Coda Lab datasets – Image Classification)

Surya Avala - z5096886, Paul Fortuin - z5017149, Ryan Mckewen - z5019386

## Introduction

Written Character Recognition by computers is an important task for computers to master in AI age as it has a wide variety of applications including language translation, document preparation and preservation, as well as archiving old documents, reading written forms quickly as well as other purposes. The MNIST data set provides a set of 70,000 examples of handwritten digits (with labels) that can be used for training and testing programs designed to recognise these digits. This dataset however was digits(0-9 inclusive) only, which is good for the purpose of digit classification but insufficient for the purposes listed above where full use of the alphabet (both lower and upper case) is used.

Luckily a team at Western Sydney University led by Gregory Cohen has compiled the EMNIST (Extended MNIST) which contains 814,000 data points of alphanumeric characters in the same format as the MNIST. This was the primary data set used to train our neural net.

We have implemented two and three layer Convolutional Neural networks using the TensorFlow library to do character recognition on these data sets.

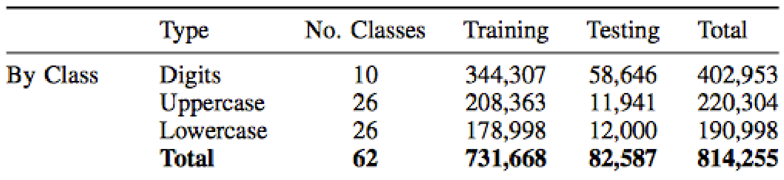
## Domain Specific Information

The MNIST data set is database of digits compiled by Yann LeCun, Corinna Cortes and Christopher J.C. Burges. The digit images within the data set have been pre-processed as explained on the MNIST website:

“The original black and white (bilevel) images from NIST were size normalised to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. the images were centred in a 28x28 image by computing the centre of mass of the pixels, and translating the image so as to position this point at the centre of the 28x28 field.” [1]

The images from the EMNIST have been formatted in the same way so they will be compatible with with learning algorithms designed for the MNIST’s data. This is useful as the machine learning library TensorFlow has built in functions to import MNIST into a training model and can be easily modified to import the EMNIST.

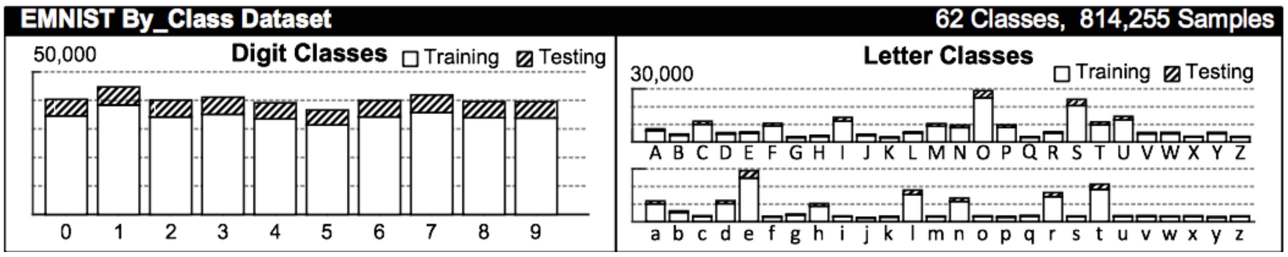
The EMNIST's has a number of different partitions of its data. We have chosen to do classifications by class which partitions into 62 distinct classes (26 uppercase, 26 lower case and 10 digits). This data set is unbalanced, there is not equal amounts of training and testing data for each classification



| | Type | No. Classes | Training | Testing | Total |
|---|---|---|---|---|---|
| By Class | Digits | 10 | 344,307 | 58,646 | 402,953 |
| | Uppercase | 26 | 208,363 | 11,941 | 220,304 |
| | Lowercase | 26 | 178,998 | 12,000 | 190,998 |
| | **Total** | **62** | **731,668** | **82,587** | **814,255** |

class.

The following graph represents the distribution of the training and testing



data:

## Method

The first model that was used was a 2 Layer convolutional neural net (CNN) which was implemented on the 28 x 28 pixel grid input of each data point. A Convolutional layer with size 5x5 and stride 1 is used with Relu activation and a 2 x 2 pooling function is used after this to transform the data set into a feature map of 32 features.

The second conv layer again uses a Relu activation function with a 5x5 filter window and stride 1 to produce a 64 output feature map. 2x2 pooling is used again here.

In the third model as third layer is used to transform the 64 feature map variable into a 128 feature map variable with the same stride filter width and pooling features as the first two layers.

In both methods dropout is then performed at this step before finally a soft-max function is used as the cross entropy. An adam optimiser function is then used to minimise the cross entropy.

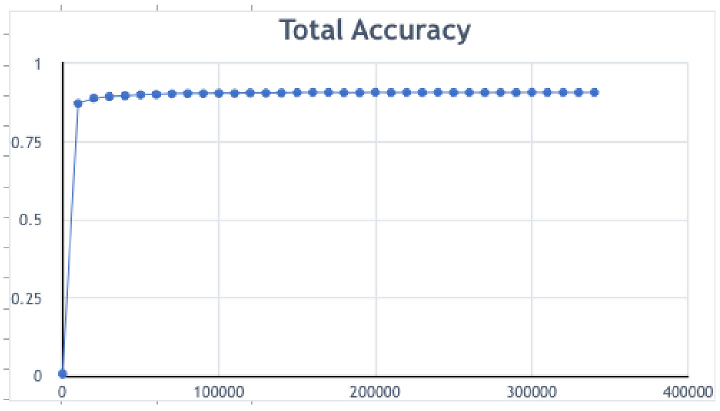
## Results

The full results of the testing data can be in the results files included with this report. Briefly, after training the 2 layer Convolutional model on the MNIST data after 15,000,000 training examples (300,000 batches of 50) the model passes the test set with a 99.6 percent accuracy.

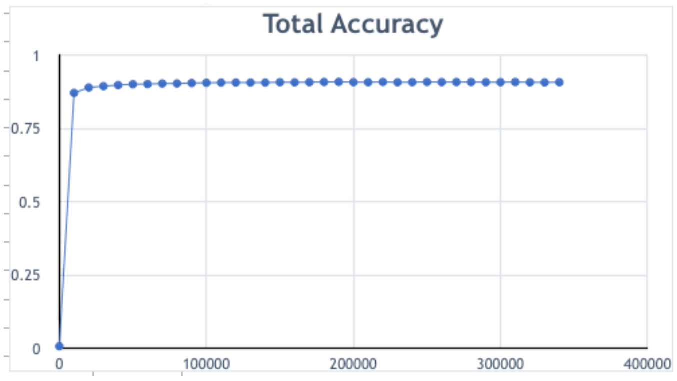
On the EMNIST data set the two layer convolutional network reached a maximum accuracy of 90.82% when run on the testing set after 15,000,000 training examples(300,000 batches of 50). Further training yielded slighty varying accuracies around 90.79%.

On the EMNIST data set the three layer convolutional network reached a maximum accuracy of 90.92% when run on the testing set set after 9,500,000 training examples(190,000 batches of 50). Further training again yielded slightly varying accuracy between 90.70% and 90.90%.

Graph showing the accuracy of the model against the amount of training for the two layer CNN:



Graph showing the accuracy of the model against the amount of training for



the three layer CNN:

## Discussion

The two layer network was not as effective on the EMNIST as the MNIST as it had about 9% less accuracy on the same amount of training data. The three layer network was more effective on the EMNIST data set than the two layer network and reached that peak in about two thirds of the time. We still consider this a success as the compilers of the database could only get a model of accuracy of 69.7% on the same data set. [2]

In either case the accuracy of the two layer model on MNIST could not be replicated on EMNIST by either model and there could be a number of reasons for this. The EMNIST data set was unbalanced(uneven amounts of each data class in the training set) which could create biases in the data for over represented classes and skew the performance especially regarding similar looking letters. Secondly regarding case issues i.e. Uppercase C and lowercase c have the same shape but are distinct classes in the data set, this issue was easily avoided in the digits only set as the digits are all distinct shapes as well as the digit 0 and the letter O.

## Related Work

Classifiers for written characters have many applications and are therefore many attempts to implement our desired scheme as well as extensions. These include attempts to classify various aspects of punctuation including comma’s, period and exclamation points. There is also a need to classify standard mathematical symbols such as (+, -, x , ÷) as well as more advance mathematical notation (for a classier) such as fractions and integral signs. More complicated still are exponentials(which use superscript e.g $e^x$ ), as these have multiple characters embedded with in the character as a whole.

Applications such as MathPix can convert images into written mathematical script into Latex and other written formats. It is our understanding that MathPix uses Long-Short Term Memory neural networks.

## Conclusion

We believe our algorithm performed well in classifying the MNIST data compared to high accuracy programs with an error rate of only 0.4%. As of 2018 the world record for classification on the MNIST database is 0.18% error rate[4]. We also believe the performance of our Neural networks on the EMNIST were also successful but also for some way to go and there are many methods that need to be considered. Some of these including merging similar classes of characters (Such as lowercase and uppercase C), this was done by Cohen's team and allowed them to improve the accuracy of their trains models from 69.7% to 71.5%. [3]

Also using more Convolutional layers as well as different sized pooling layers and different activation functions. Other Machine learning algorithms such as LSTM and SVM as well as hybrids of these with CNN’s should also be considered.

Changing the data set may also help improve overall functionality of the data set. A balanced data set would give the model a better ‘understanding’ of the lesser used characters such a x and z.

## References

[1] http://yann.lecun.com/exdb/mnist/

[2, 3] https://arxiv.org/pdf/1702.05373.pdf

[4] https://arxiv.org/abs/1805.01890

Data Used:

MNIST Database taken from http://yann.lecun.com/exdb/mnist/

EMNIST database taken from https://www.westernsydney.edu.au/bens/home/reproducible_research/emnist

Acknowledgements:

The following tutorial was used as a guide to build our CNN:

https://www.tensorflow.org/tutorials/layers

Other Acknowledgements go to the compilers of these datasets.