EMNIST Image Classification Project

Andres Aranguren
andres.aranguren@studenti.unipd.it

1. Introduction

The number of images publicly available has skyrocketed during the 21st century. This has been caused by a number factors, including the internet boom as well as the rise of social media. This has been accompanied by parallel developments in image searching and analytics. The analytical side including but not limited to: object detection, labelling and scene identification. The wide availability of images has prompted the identification of the most powerful and accurate algorithms for the task. We would like to understand which algorithm out of a selected group(RandomForest, Neural Network(NN) and Convolutional Neural Network(CNN)) is the most effective in predicting items from our input data-set. For this task the extended MNIST image database (EMNIST) is adequate. Once the best model is discovered we also want to understand what effect noisy/altered images have on the model's ability to predict correctly. This step is important as in providing the model imperfect examples of images we are hopefully improving its ability to generalize and learn unseen data. In real world applications we will often be processing images from sub-par/different data-set's hence it is important for a model performing image classification to cope with these imperfections. Ultimately our goal is to deremine the most robust model that is able to not only predict accurately but be flexible in image qualities that it encounters. Our initial hypothesis was that the CNN model would fit this description. After concluding our testing it turned out that we were correct. The CNN outperformed all linear models significantly. The closest to it was Random Forest and giving some serious tuning could potentially end up matching a CNN.

2. Dataset

The dataset used during this project consists on a variant of the NIST datdaset, called extended MNIST (EMNIST). IT consists on set of datasets that consitutes a more challenging classification task involving letters an ddigits, sharing the same image structure and parameters as the original MNIST task, hance its compatibility with the already existing classifiers and systems. The dataset contains handwritten digits and characters collected from 500 writers. The database intended to provide a more complex optical character recognition, therefore presents the data in five separate organizations. For this project By Merge group was used since it offers the most useful organization for classification tasks since it contains the segmented digits and characters arrange by classes, it also address a problem which is the similiarity between certain uppercase and lowercase letters. The dataset is divided in 10 classes of digits, 26 uppercase letters, and 11 lowercase letters with classification similarity, for a total of 47 classes[2].

The first step, to address this data-set is to develop a proper baseline, which involves developing the infrastructure for the test harness, so that any proposed model can be evaluated on the data-set. During this project test harness was divided in five key elements, loading of the data-set, preprocessing, definition of the model, evaluation of the model and presentation of results.

[1].

Following, sample images of the dataset group of labels:

Figure 1. Sample of digits images

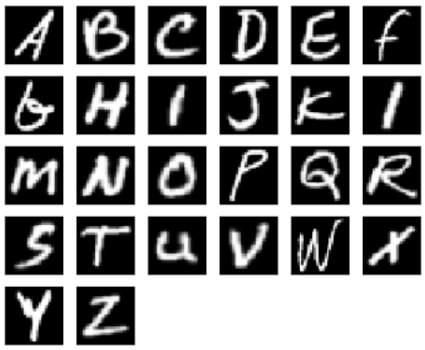

Figure 2. Sample of upper case letters images

Figure 3. Sample of lower case letters images

2.1. Preprocesssing

After loading the dataset, the first aspect to notice is that all the images are pre-segmented, meaning that every image contains a single piece of clothing, all have the same square size of 28*28 pixels.

Pixel value is a single value between 0 and 255 due to the images being gray scale ranging [0,1], which ensures that each pixel has a similar data distribution. Data normalization was carried out by subtracting the mean from each pixel and then dividing the result by the standard deviation. Data augmentation was also implemented which is done by altering images in the dataset.

3. Methods

3.1. Linear Model approach

First of all, a random forest model was implemented with almost default parameters in order to identify its baseline accuracy and perfromance.

1. Random Forest classifier was running with gini impurity. Between the gini impurity and information gain entropy the accuracy was almost identical. Hence decreased the computational load. The max depth for our model was attained at a depth of 80.

3.2. Neural networks approach

The first NN model that we implemented was a linear one layer model. It was used as a benchmark for the development of more complex models that would yield an enhanced performance. Using this model we evaluated the effectiveness of different activation functions such as relu, tanh and sigmoid functions,demonstrating that relu was the strongest one, this can be explained since Relu is a linear function that allow to prevent the vanishing gradient problem where weight update is delayed during backpropagation because of the asymptotic limits of both sigmoid and tanh functions. The best model was comprised of a single relu activated 128 node dense layer forwarding the predictions directly to the output neurons.

After these initial linear models, We proceeded with a single layer CNN's that would in theory outperform the baseline neural network modle. Initially we left the parameters for our simple CNN almost identical to our linear one. A single layer convolutional layer followed by a max pooling 2d. This first result gave a good starting point as the CNN already rivalled the strongest linear model (Random Forest). The next question was to determine what could be improved from this basic model and enhance its metrics. After several testing of four different CNN models we obtained our final architecture which can be seen in Figure (4).

In order to create the model we analysed several parameters. Most critically we increased the number of convolutional and pooling layers within the model. In total for the final mode three Conv2D and MaxPool2D layers were used with a final Conv2D layer and Flatten before output classification, performed using a softmax function to calculate the most likely belonging class of an input. This was the optimal model which we used as our skeleton.
From this point onwards the main goal was reducing model over-fitting as well as including data augmentation, through the addition of gaussian noise. AS final step batch normalization was included between Conv2D and MaxPool2D layers.

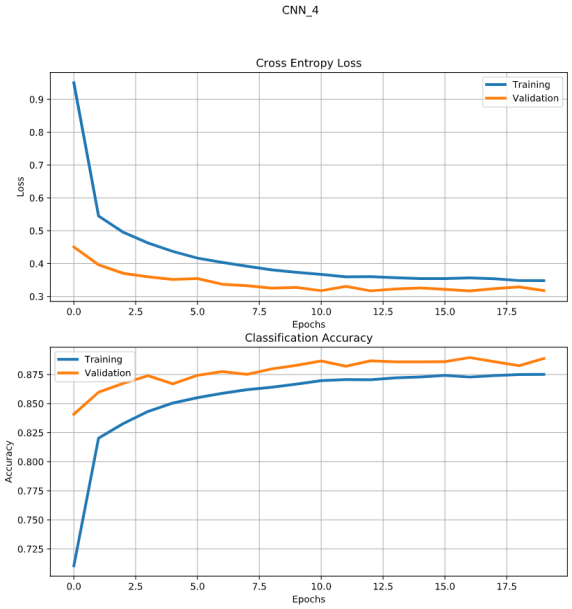

Figure 5. Final CNN Model train-val loss and accuracy over epochs

Correct predictions	Incorrect predictions	Prediction accuracy
20051	2509	0.88

Figure 6. Predictions using final CNN model

4. Experiments

4.1. Neural Networks results

The first parameters we studied were the activation functions. We compiled three models and applied the recti-

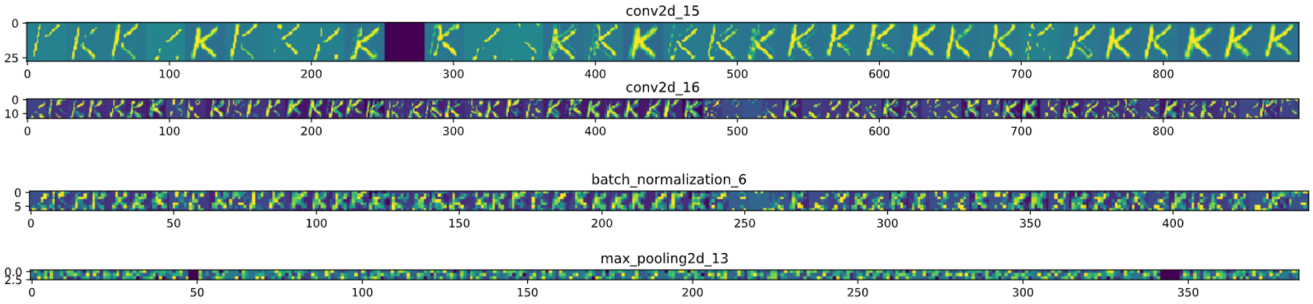

Figure 4. Intermediate steps representations over network

fied activation function (ReLu), hyperbolic tangent function (Tanh), and sigmoid function. These models were then tested and validated using the training and validation data-set. Models' performance were evaluated based on the cross-entropy loss and classification accuracy over 20 iterations as seen in figure (6).

Relu activation function demonstrated higher accuracy compared to tanh and sigmoid functions. This can be explained by a general problem when non linear functions are applied to deep neural networks. It is their saturation which means that during back propagation steps large inputs snaps to 1.0 and small values to -1 or 0 for tanh and sigmoid respectively. The neural networks losses a lot of useful information to update the weights, a problem called vanishing gradient. Since ReLu acts like a linear function this is no longer a problem. The gradient will not reach a local minimum nor a non representative value that will block the algorithm's learning rate [3].

As expected the simple NN exhibits metrics similar to the linear method. This is caused by the loss of relevant image information when converting to 1D arrays and the number of neurons in the single layer.

The chosen hyper parameters with which we found our best model were, alpha = 0.001, batch size 32 and number of epochs=20.

The final model architecture is shown in figure (4). Composed of four convolutional layers, three followed by max pooling operations tasked with extracting the relevant features of the input images.[6] Lastly one fully connected layer composed of 128 neurons leading to an output layer of 10 neurons. This layer was softmax activated in order to predict the final belonging class of the input image.

Two regularization approaches were applied as well as four dropout operations which defined the fraction of neurons activated during forward propagation. We also used batch normalization between the convolutional layers and max pooling. This operation have been heavily demonstrated as capable of increasing model performance [4]. It is however, important to consider their proper placing in the networks' architecture in order to prevent optimization instabilities during weights update. A final precaution to avoid over-fitting and consequently decreased performance in the test data-set was the addition of Gaussian noise. This is added to the input variables, with a hyper-parameter setting the standard deviation to 0.1 [5]. Figure (6) describes the model performance during the fitting of the training and validation data sets. The figure shows accuracy peaking across 20 epochs at 90 %. Figure (**??**) shows the accuracy across the 46 labels within the validation dataset.

The last phase for model testing was applying k-fold cross validation which allowed us to perform re-sampling of the training and validation dataset. A k fold value of 5 was specified, computing the final precision as the average between the five data resamples, results are demonstrated in figure (7).

In order to illustrate the process of an image or input when passing through the model, we evaluated the intermediate steps representations after the convolutional layers and max pooling layers, which allow to retrieve the most relevant information of the input, while mantaining the most of the original size with operations such as valid padding and kernel size configuration.Figure illustrates the intermediate steps over the network. As a final analysis we calculated the confusion matrix which allows to visualize the accuracy based on the predicted values against the true values described by the labels of test data set. See Figure (8)

4.2. Performance Comparison for all Models

Model	Regularization	Precision	Recall	F1-score
Random Forest max_depth = 80		0.81	0.81	0.81
CNN (single layer)		0.83	0.81	0.81
CNN (two layer)		0.83	0.81	0.81
CNN (three layers)	Dropout (0.3) Gaussian noise (0.1) Batch normalization	0.90	0.88	0.88
CNN (four layers)	Dropout (0.3) Gaussian noise (0.1) Batch normalization	0.90	0.89	0.89

Figure 7. Precision/Recall chart for all Models

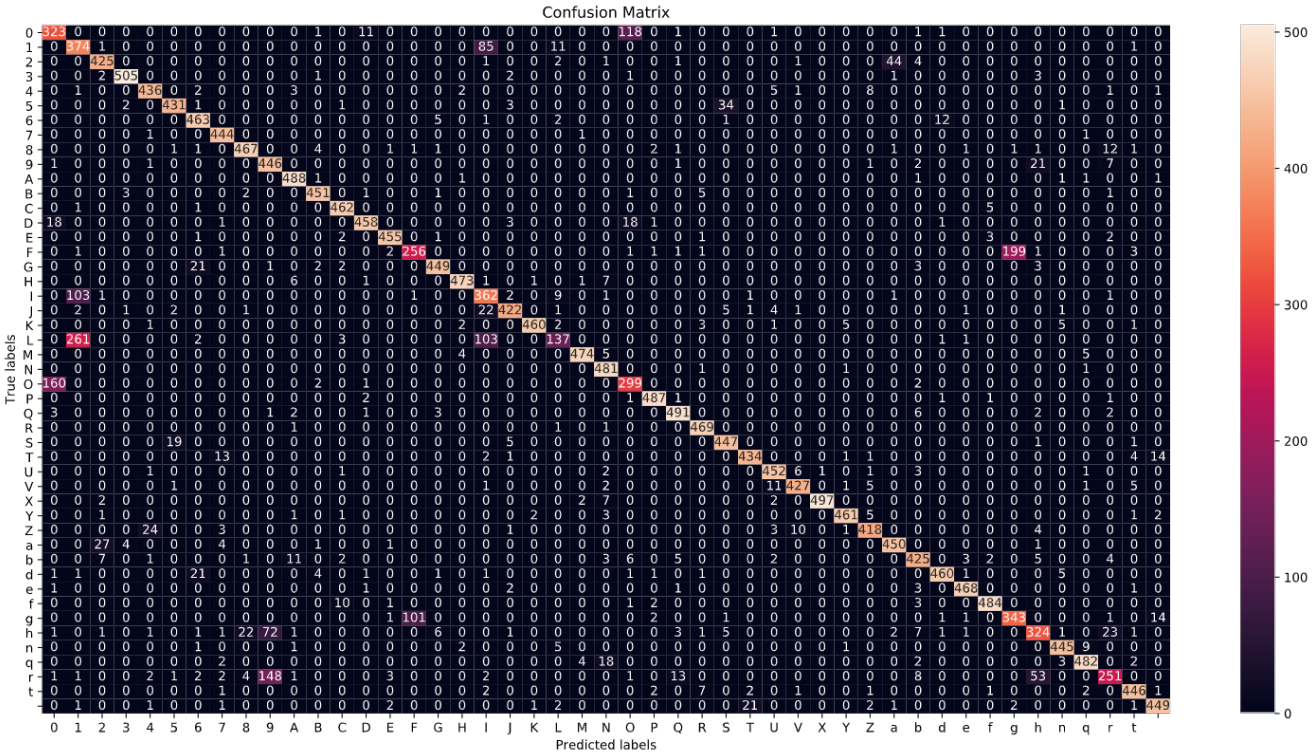

Figure 8. Confusion Matrix using final CNN architecture

4.3. Intermediate steps

References

[1] Ahmed Anter. Classification of garments from fashion mnist dataset using cnn lenet-5 architecture, 02 2020.

[2] Tapson Ryan Cohen, Afshar. Emnist an extension of mnist to handwritten letters, 2001.

[3] Yujun Shi Chang-Yu Hsieh Benben Liao Shengyu Zhang Guangyong Chen, Pengfei Chen. Deep learning using rectified linear units (relu), 02 2019.

[4] Yujun Shi Chang-Yu Hsieh Benben Liao Shengyu Zhang Guangyong Chen, Pengfei Chen. Rethinking the usage of batch normalization and dropout in the training of deep neural networks, 05 2019.

[5] Agnieszka Mikołajczyk and Michał Grochowski. Data augmentation for improving deep learning in image classification problem, 05 2018.

[6] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 11 2015.