**Handwritten Character Recognition with Convolutional Neural Networks**

**John Nguyen1 Saif Billah1**

**Abstract**

The MNIST (Modified National Institute of Standards and Technology) dataset of handwritten digits is commonly used for training a variety of training and testing purposes in the field of machine learning. It contains black and white images of digits from 0-9 in a 28x28 pixel size in grayscale. This paper explores building a classifier on a variant of the MNIST dataset, the EMNIST (Extended MNIST) dataset. The EMNIST dataset is a set of handwritten characters converted into the same format as the MNIST dataset (28x28 pixels in grayscale). This paper goes on to explore creating a Convolutional Neural Network to classify letters in this dataset, and presents results via learning curves, a confusion matrix, and commonly misclassified letters.

**1. Introduction**

Written character classification is an area of research that has widespread applications in a multitude of industries. Recognition of written characters would allow handwritten forms to be automatically converted into editable documents, letters to be sent to the correct address automatically, digitize handwritten notes to be edited and reviewed on other media, and more. Some of the benefits to using a written character classifier are scalability, near real-time feedback, and consistency. A machine learning classifier would be able to perform more classifications at a faster rate than a human counterpart would be able to, and would not be

prone to human fallacies such as distractions. The benefits to building a classifier for written

characters were noticed, and thus inspired the classifier discussed in this paper.

**2. Data**

Our approach to this problem had multiple initial steps. The first of which was to determine the dimensionality, size, and the format of our dataset, to choose a strong approach for our model. The split of the dataset that was chosen to use was the ‘emnist-train.csv’ split, with 50,000 characters and 26 balanced classes (i.e. labels for each character in the English alphabet). Each character was provided in the format of a 28x28 pixel image in grayscale. This meant that the input to the classifier would essentially be 28x28x1 because distinct RGB values did not have to be considered. [3] The data set size of the training, validation, and test data sets were the following: training data set size to 35,000 characters, validation data set size to 10,050 characters, and test data set size to 4,950 characters. To illustrate what our data looks like, a random sample set of 20 images from our training dataset is shown below:

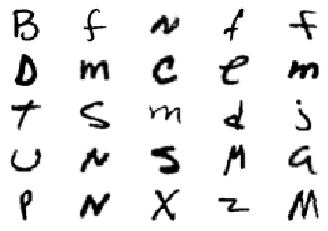


Fig. 1: 20 images from the training dataset.

After initial research into which kind of model to use, it was noted that images in particular have little relation between unique pixels unless those pixels are close to each other. Research on Neural Networks and related models was performed. However, to reduce the computational complexity of training such a classifier on these images, convolutional layers and pooling layers were further looked into, in order to extract features from the data to preserve the relationship between pixels, and reduce the parameter counts. Thus, the solution of implementing a Convolutional Neural Network (CNN) was chosen.

**3. Computer Vision**

Convolution is a popular computer vision pre processing technique used to extract key features from an image such as edges or corners. Essentially, a kernel or small window is slid across an input image to output a filtered subset of the original input image. This kernel or template can be tuned by kernel size, type (average, Gaussian, etc) and by edge case padding. The result of a convolved image is a “feature” map where these features are often interesting artifacts in images.



Eq 1: Convolution window computation

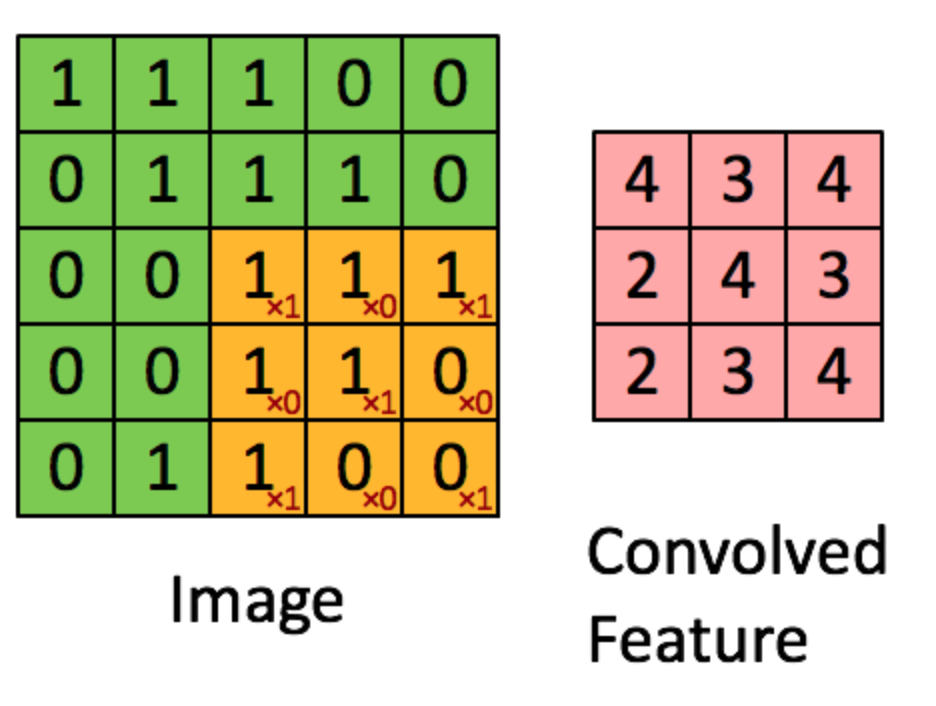


Fig. 2: Convolution of a 5x5 pixel image into a 3x3 convolved matrix.

After obtaining a feature map, non max suppression is used as a smoothing technique. Another form of this is max pooling. Pooling layers were chosen to be inserted after each convolution layer. The intention of doing so was to reduce the spatial size of the image representation, which would reduce the parameter count and then reduce the computational complexity of the problem. The pooling technique used was MaxPooling, which essentially breaks up the input matrix into a specified number of quadrants, and chooses the maximum value from that quadrant to place into a new, smaller matrix. This was also done to try and prevent overfitting the model to the data and to reduce computation time. [2]

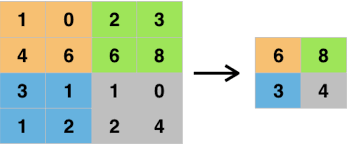


Figure 3: Max Pooling 2x2

**4. Convolutional Neural Network**

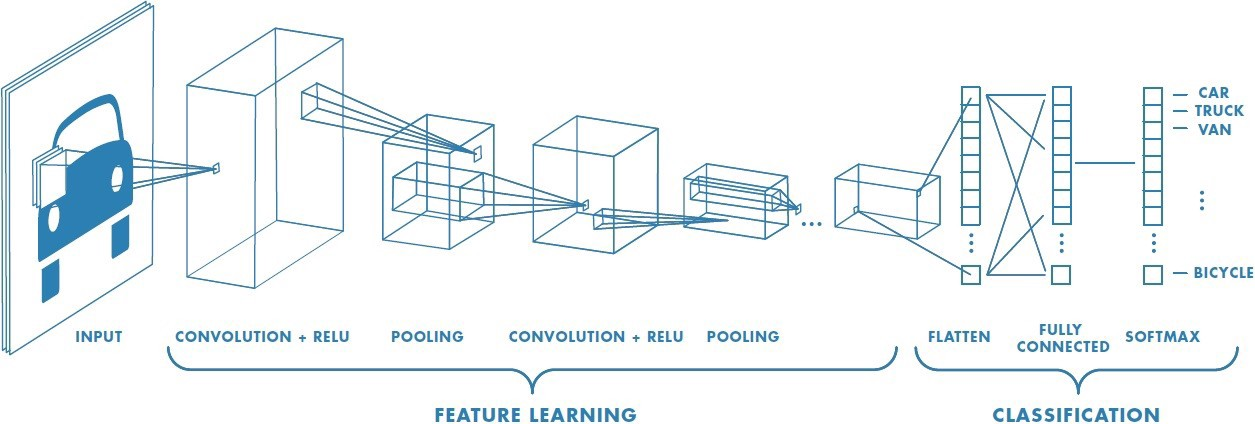


Fig. 4: Sample CNN architecture to classify vehicle images by vehicle type.

A typical CNN architecture is shown above for the task of image classification. A class was created to define the CNN architecture which takes in the data or images to predict class probabilities and choose the highest probability. To begin, the class has one function, forward which computes a forward pass. During the first forward pass, the first activation function is a convolution on the input and then a nonlinear ReLu which has a derivative function and allows for backpropagation. More specifically, a Leaky ReLu was implemented to avoid the dying Relu problem which occurs from a large negative bias resulting in the gradient becoming zero and being unable to learn again. Next off this process is repeated and then the resulting feature map is reduced using max pooling. These two layers are again repeated and both times, edges are accounted for by using zero padding and this makes up the feature learning portion of the CNN. Setting up the CNN involved specifying the input and output counts of each layer and continuously reducing the input features. Next off, the image is then flatten to be passed into the fully connected layer which is a linear transformation of the form:

y = xA^T + b

Eq. 2: Linear Transformation Equation

The second fully connected layer then reduces the output of the first layer to a size of 26 which are the possible classifications of the dataset. [2]

With the CNN defined, the next step is to train the CNN using cross entropy as the loss function of choice. Cross-entropy loss is an appropriate choice as it measures the performance of a classification model or in other words, the difference between probability distributions for a given random variable. Another name for this is log loss, since the logarithmic scale takes into account the uncertainty of the prediction based on its variance from the true label as opposed to accuracy which is either a “yes or no” label. Using 10 epochs with a batch size of 1024 images, a training loop is created to implement a forward pass and adjust the weights iteratively. Using the EMNIST dataset provided, and due to lack of compute performance from available computers, the EMINIST letter dataset is split into 70 percent training, 20 percent validation, and 10 percent testing. From preliminary research, the Adam optimizer achieved high results for digit recognition so the loss and optimizer are defined to be cross entropy and the Adam optimizer which has an advantage over stochastic gradient descent due to its adaptive learning rate. For comparison, stochastic gradient descent was also experimented with which had a much longer training time. SGD is the rate of loss function w.r.t the model parameters where these weights are updated after looping through each batch size. A common issue with SGD occurs since neural networks are non convex functions with many local minima which can cause the gradient to get stuck or explode. [5]

The Adam optimization’s adaptive learning rates is based on using squared gradients to scale the learning rate and also using momentum and the moving average of the gradient. [6] Utilizing previous gradients help to prevent the oscillations and overshooting in these non-convex optimization problems.

For each loop of training, the cross entropy score is noted in order to visualize the learning rate curves. Using pytorch and the tensor libraries, the parameter gradients are initialized and the CNN is created on the testing partition and the validation partition. From there, forward and backward pass are computed per the determined optimized step size and the best model weights are updated. With the best model weights, the predicted labels are generated on the last partition, the test set and metrics such as precision, recall and f1-score can be used to analyze the metrics as will be explored in the results.

**5. Results**

After downloading and storing the EMNIST dataset, the figure below shows the counts of each class label in the training data set with 26 classes for each letter and it appears to be an approximately uniform distribution.

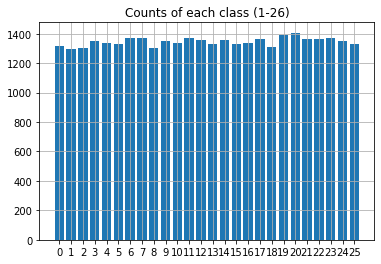


Fig. 5: Counts of each class label from EMNIST.

The following figure is the learning curve where the cross entropy loss for the training set and the validation set are shown to converge around 6 epochs out of the 10 epochs. Partitioning and feeding both these datasets are important to obtaining the best model which prevents overfitting. The CNN performs remarkably well achieving approximate convergence in about 8 epochs.

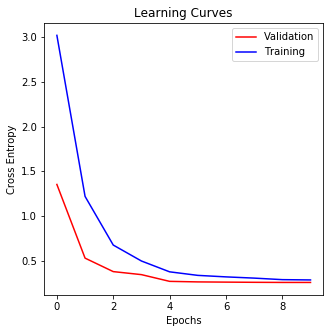


Fig. 6. Learning Curves with Adam

For comparison’s sake, the following learning curve was generated with stochastic gradient descent optimization method with 68% accuracy without optimal convergence.

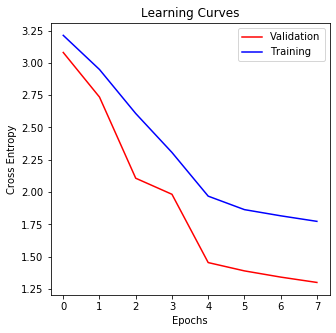
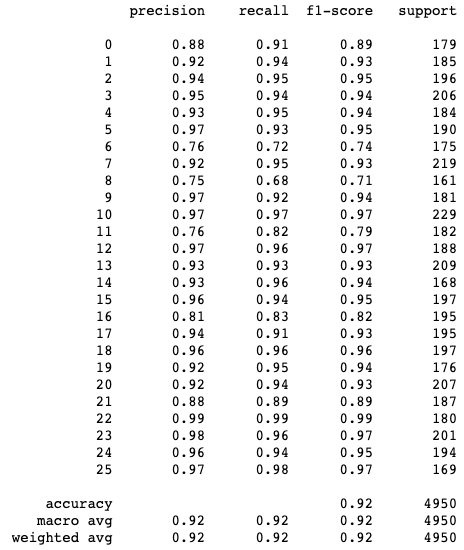


Fig. 7. Learning Curves with SDG

The following classification report from sklearn was generated and the results are shown in the chart below for each character.

Fig. 8. Accuracy of the created CNN on the inputs from the test dataset for each character.

The chart shows that some classes predictions perform better than others. Precision is the ratio of the correctly predicted positive images to the total number of images which is an indicator of few false positives. In comparison, recall is the ratio of correctly predicted images to all observed images in its respective class. Finally, the F1 score is the weighted average of precision and recall and this is similar to accuracy except for uneven class distribution. Thus accuracy is a valuable indicator of performance which is determined to be 92% accurate.

To visualize the results, a confusion matrix is created to help visualize which characters are most commonly misclassified, and what they are most often misclassified as.

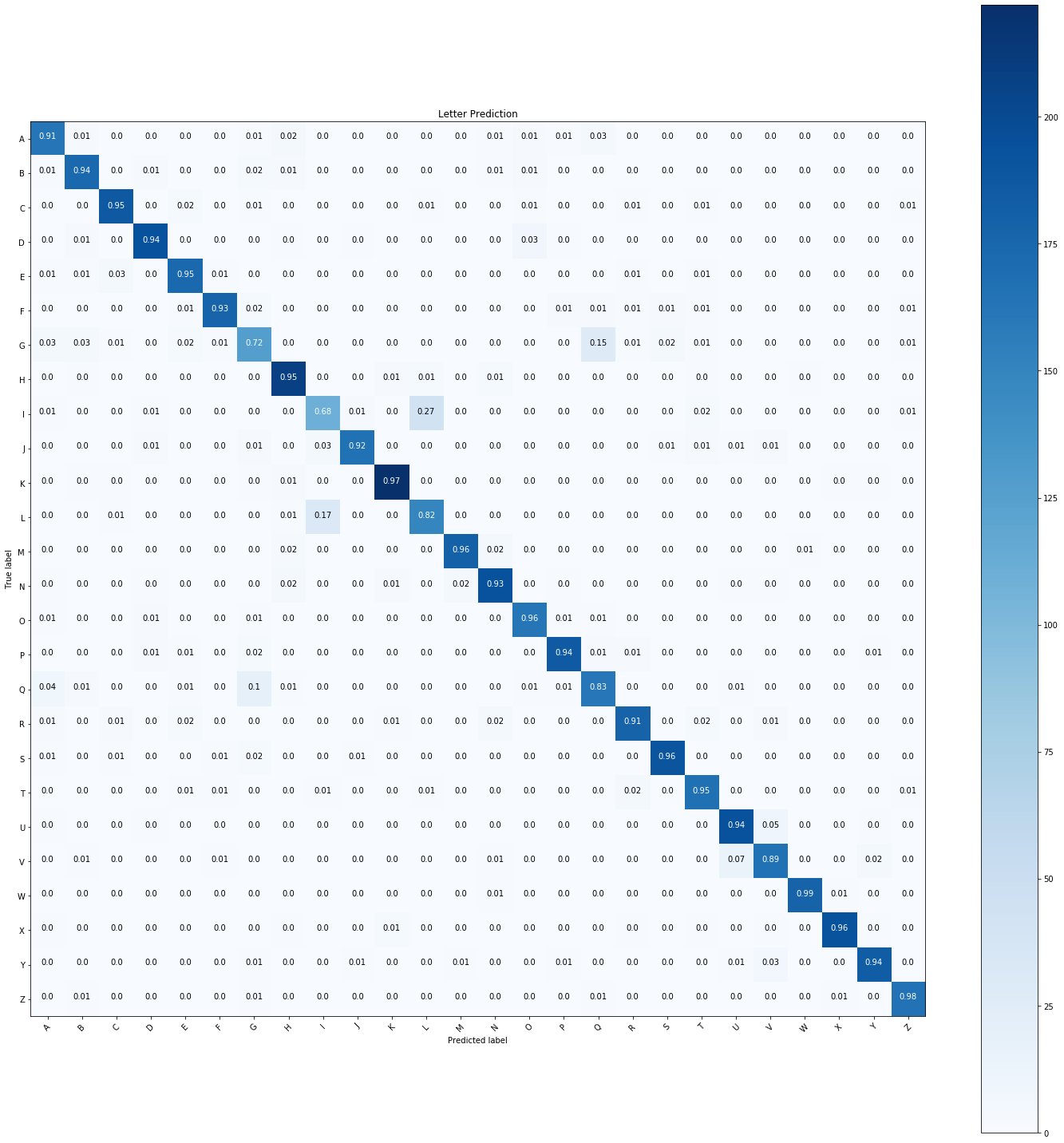


Fig. 9. The confusion matrix for the CNN’s classification results on the test data.

From the confusion matrix in figure above (Fig. 9), it was empirically found that the letter “I” was often confused with “L” (and vice versa), and that the letters “Q” and “G” were often misclassified as each other as well. An image is shown below to illustrate occurrences of misclassified letters to further explain why this took place.

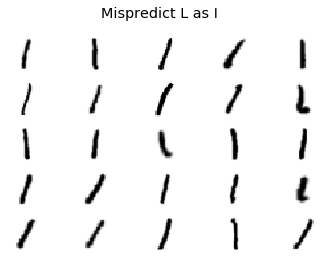
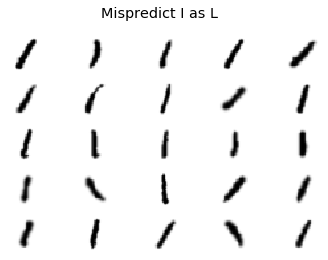
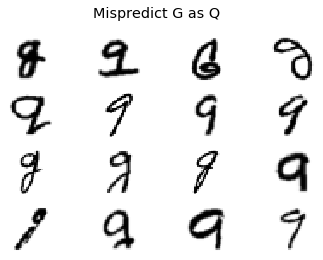
 

Fig. 10. Misclassified “L” and “I”.



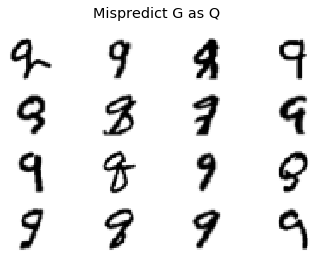


Fig. 11. Misclassified looking character for the characters “Q” and “G”.

For the set of characters that were misclassified as “I” and “L” from the figure above, this is expected as the characters are even difficult to classify with the human eye because they look so similar to each other. The same goes for the characters that are misclassified as “G” and “Q”, but to a lesser extent. These characters are also not easy to classify with the human eye, but are somewhat easier to distinguish between than the previous comparison.

Overall the CNN achieved high results and it is expected that with more training and a larger dataset, the prediction accuracy can improve. However, achieving near perfect results is virtually not possible due to the certain degree of human error. When writing a character, even people have arguments over poor handwriting thus a machine learning error cannot compensate for human bias. A possible improvement would be to make a model that predicts characters based on the characters surrounding the word. People often make inferences on a hard to distinguish character but looking at the word deducing the word with knowledge of the dictionary. This is an interesting problem that can be further explored in future experiments.

**References**

[1] Yal, Orhan Gazi. “Image Classification in 10 Minutes with MNIST Dataset.” *Medium*, Towards Data Science, 5 Sept. 2019, https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d.

[2] Algorithmia. “Convolutional Neural Nets in PyTorch.” *Algorithmia Blog*, 5 Dec. 2018, https://algorithmia.com/blog/convolutional-neural-nets-in-pytorch.

[3] Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. Retrieved from <http://arxiv.org/abs/1702.05373>

[4] Brownlee, Jason. “Gentle Introduction to the Adam Optimization Algorithm for Deep Learning.” *Machine Learning Mastery*, 13 Nov. 2019,

[5] Brownlee, Jason. “A Gentle Introduction to Exploding Gradients in Neural Networks.” *Machine Learning Mastery*, 14 Aug. 2019, https://machinelearningmastery.com/exploding-gradients-in-neural-networks/.

[6] Bushaev, Vitaly. “Adam - Latest Trends in Deep Learning Optimization.” *Medium*, Towards Data Science, 24 Oct. 2018, https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c.