Neural Nets and Optical Character Recognition

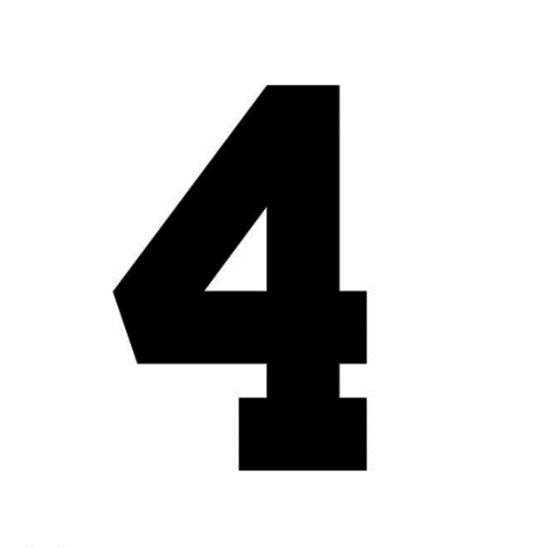
An independent study conducted by Emily Wasylenko, guided by Dr. Dellinger

**Abstract**

This paper explores the role played by neural net modifications and data preprocessing in solving a common problem in computer vision. The neural net is trained first on a standard character dataset, then on two custom modifications of the set. Results are compared to show the effects of different datasets.

**Introduction**

Optical character recognition (OCR) is an integral part of the growing field of computer vision. It refers to a computer’s ability to correctly recognize and interpret symbols, such as Arabic numerals. Though trivial for a human, recognizing characters is a difficult task for a computer. There are virtually limitless ways to write each character. For example, every human will have a slightly different way of writing a “4”.



**Fig 1**: Two different ways to write the same number [1][2]

To recognize that the two images above represent the same thing, a program must find the prevailing pattern across many different examples. A tool that is commonly used for such a task is the neural net.

The neural net in this study made use of the library Deep Learning for Java (DL4J), a resource for building neural nets in the Java programming language. The neural net was trained on data from the Mixed National Institute of Standards and Technology (MNIST) database: a set of handwritten digits and the corresponding matrix of pixels needed to form the digit. [3] Every row in the dataset follows the same pattern: an integer label between zero and nine, followed by a 28-by-28 grid of pixels. Pixel values range from 0 to 255, and these provide a grayscale image of the label.

**Related Work**

Research on OCR has been conducted since the mid-19th Century [4]. The work done in this study, however, is loosely based on more recent papers, primarily those dealing with feature extraction techniques. These techniques highlight important features and patterns in the original dataset. This altered dataset is then given to the neural net, with the intended goal of improving performance.

The first feature extraction technique used in this study is region mapping. This involves counting how many whitespace regions in the pixel matrix are separated by dark strokes. Faisal et al use similar techniques when analyzing text on a page [5]. They transform each individual character into a matrix of standard size: 15 x 15 pixels. Pixel values of 1 represent dark regions, while values of 0 represent lighter regions and whitespace. The number of whitespace regions are of interest to our study, though future work may include location, angle, and prevalence of dark regions.

The second feature extraction technique in this study is based on the work of Hadsell and colleagues [6]. In their research, they perform several advanced variations on dimensionality reduction, the process of transforming a large, detailed dataset to a smaller dataset with less detail but equivalent value. Their research is especially applicable here, since it also makes use of the MNIST dataset.

**Basic Terminology**

The artificial neural net (ANN) is a computer program based loosely on the human brain. It contains representations of “neurons.” As the neurons of a human brain send electrical signals to each other, so the nodes of an ANN send around numerical data. The final output value of this data depends on several things: the original inputs, the weight of each neuron (how the neuron alters the data), and the activation function (how big or small the input must be for any given neuron to alter it and send it forward through the net.

Most ANNs have two or more layers: an input layer of neurons to receive initial data, and an output layer to return final data. Often, ANNs will have hidden layers. These sit between the input and output layers and perform additional work on the data.

ANNs are first trained to recognize data with a *train set*, and then their ability to perform correctly is tested on a *test set.* Each set contains *labels* and corresponding data. Labels are the things the ANN is learning to recognize. For example, in this study, the digits 0 through 9 are the labels, and the pixel matrices are the corresponding data.

Upon completion of training and testing, the ANN prints out a *confusion matrix*. Rows in the confusion matrix represent the ANN’s value predictions, and columns represent actual values. In this study, the presence of higher values along the diagonal is a good sign. This means the ANN predicted a zero where the actual value was a zero, a one where the actual value was a one, etcetera.

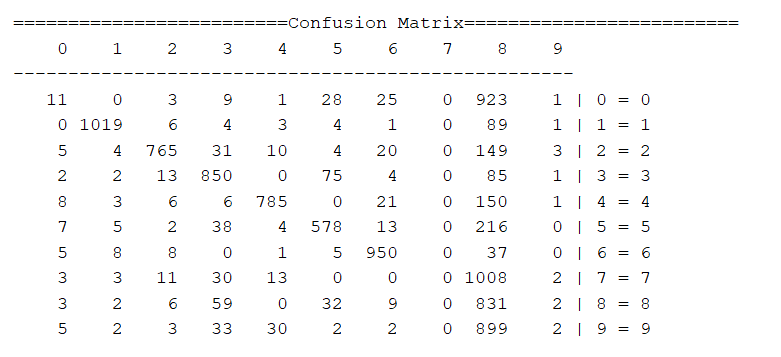
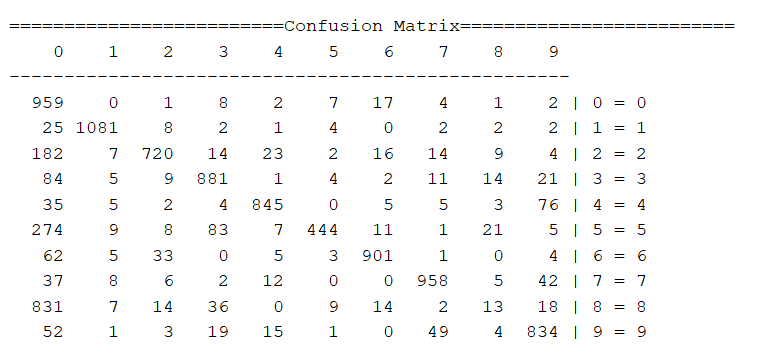
In order to learn properly, the ANN must see the labels and data many times. This training is accomplished with a *dataset* *iterator*, a structure that reads through a large data set and processes the data so that the ANN can understand it. A custom dataset iterator is implemented here. It is built to handle ten labels (one for every digit). The number of *epochs* determines the number of times a dataset iterator shows the training data to the ANN: one epoch is equivalent to a pass to the end, and back to the beginning of the dataset.

The dataset iterator reads the data into a *neural net configuration builder*. This is a vital piece of the ANN. It defines the structure of layers and how data is interpreted within the net with a series of variables. Five such variables that will be referenced later in this paper are *regularization*, *weight*, *learning rate*, *momentum*, and *activation function*. Their functions are defined briefly here.

Regularization is a technique used by ANNs to avoid overfitting data. An ANN suffering from overfitting will look too closely at individual points in the training set, missing the underlying pattern. Weight is the degree to which any two neurons are related. If a high output from one neuron causes a high output from another neuron, the weight is high. Learning rate is the extent to which the model changes based on inputs at each step. Momentum optimizes the algorithm, preventing it from getting stuck in a rut too quickly. Finally, the activation function filters out weaker signals and noise in the data, helping the ANN find meaningful patterns.

**Initial Result**

Several issues arose when trying to train our machine on the MNIST dataset. A recurring problem was the column of confusion: values in one or more columns were substantially and unreasonably higher or lower than values in other columns (Fig. 2).

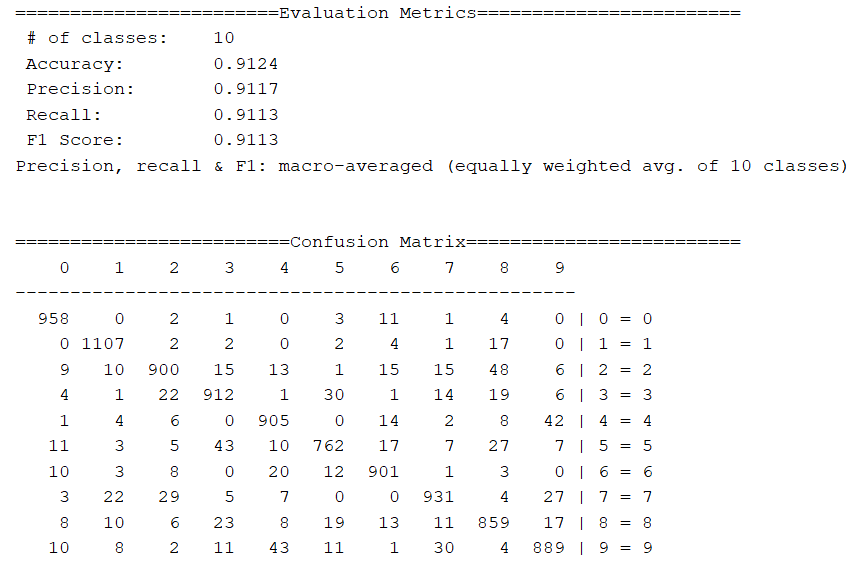


**Fig. 2a:** Confusion in columns 7 and 8 **Fig. 2b:** Confusion in column 0

The column of confusion usually lay in the 8s (Fig. 2a). Adjusting the learning rate and the momentum sometimes changed the column of confusion to a different number (Fig 2b), but the overarching column of confusion was the 8s. Sometimes (Fig. 2a), this caused another column to have values much lower than the other columns.

This error was due to a lack of normalization. The iterator is set up to handle non-label inputs within the range of zero and one. When data values with an upper bound of 255 were read into the iterator, it became confused and produced meaningless output.

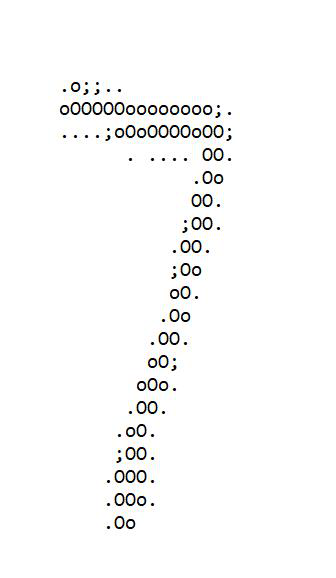
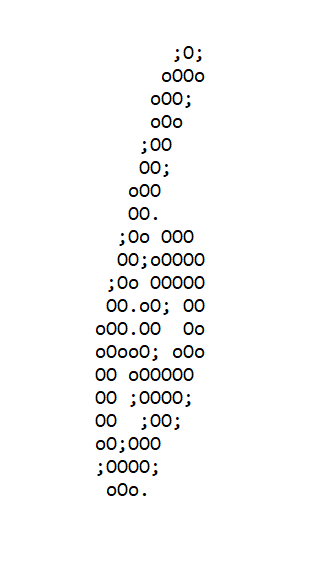
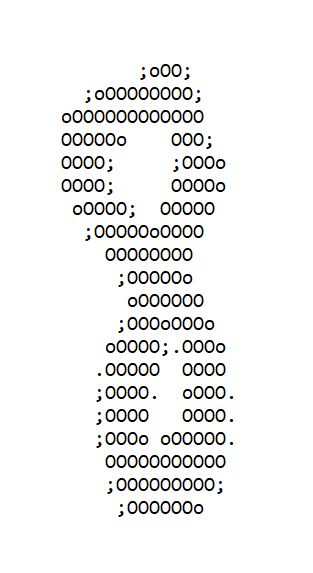
Data was normalized by dividing all non-label values by a common denominator. Pixel values were divided by 255, and region numbers were divided by 8, since this was the largest number of regions present. Once the data was properly normalized, the neural net produced an expected output (Fig. 2).

****

**Fig 3:** Corrected columns with original MNIST dataset; number of epochs = 105

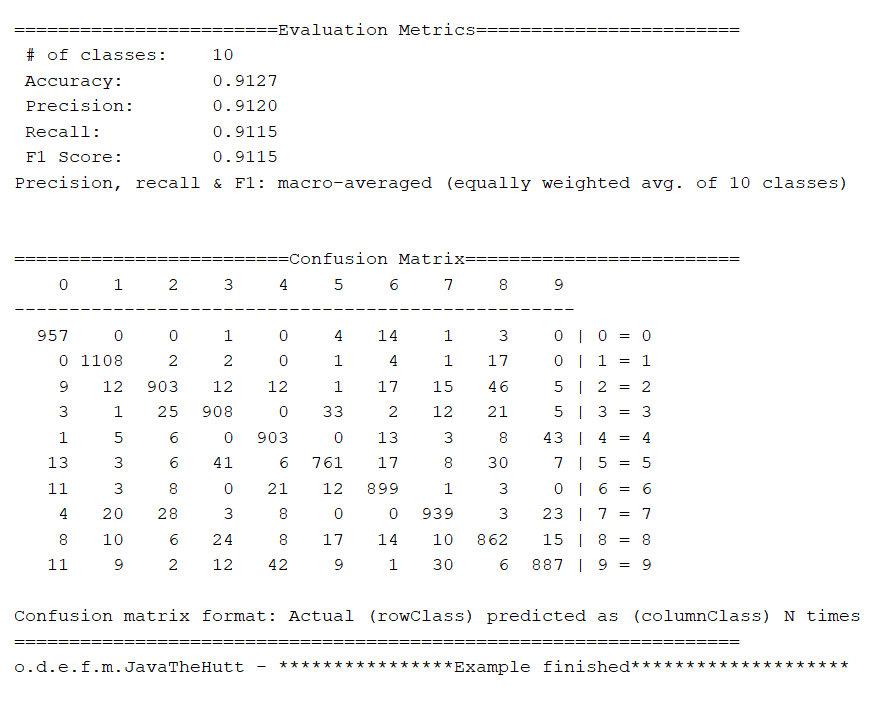
**Region Mapping**

Our first feature extraction technique involved finding the number of regions in each number. Regions consist of contiguous pixels with a value low enough to be considered whitespace. The numbers below contain one, three and five regions, respectively.



**Fig. 4:** Three printed MNIST symbols, recognizable as a seven, an eight, and a six.

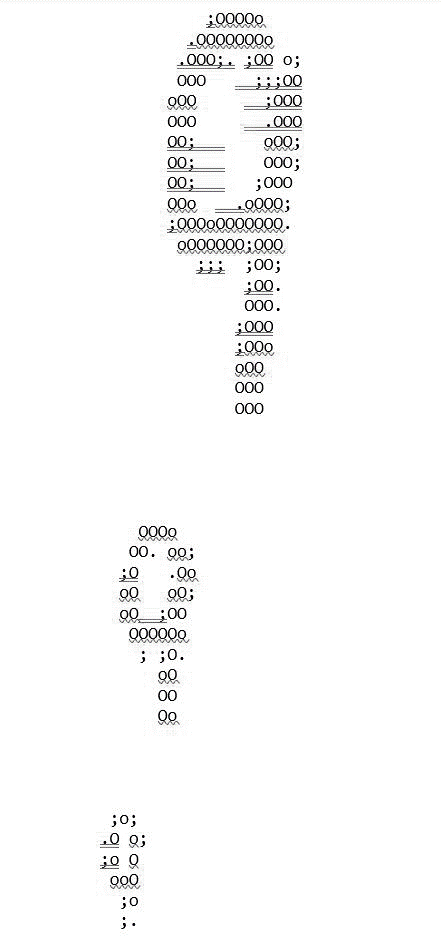
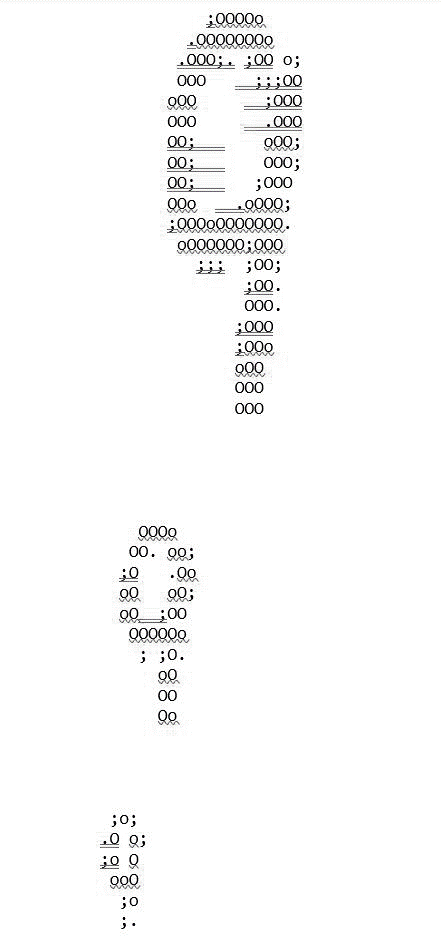
The region mapper collects this information by iterating over each pixel matrix, searching for contiguous blank pixels by index until there are none left. It then skips over filled pixels until it finds another region of blank pixels. Fig. 5 shows the data collected after training region mapped pixels. Note the diagonal line, denoting correct identification of number.

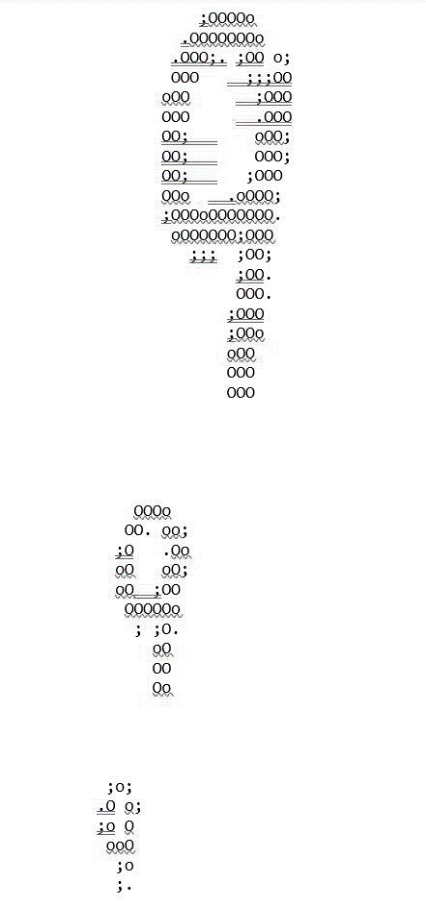


**Fig. 5:** Normalized output with region numbers included; number of epochs = 105

**Dimensionality Reduction**

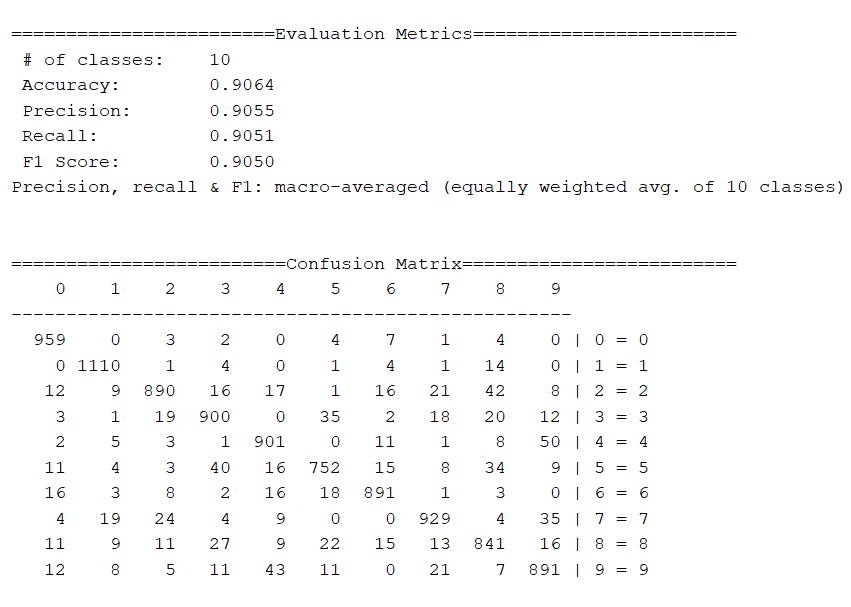
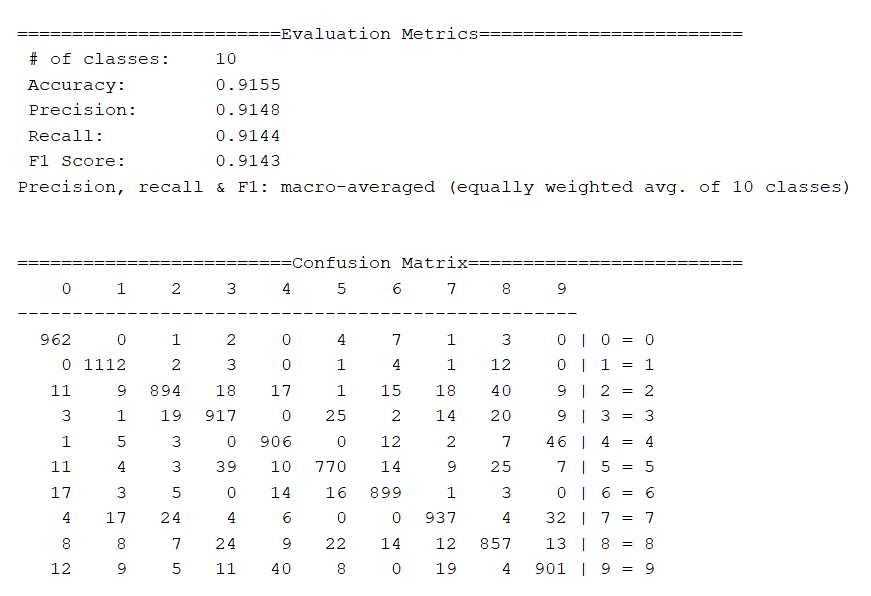
Our second feature extraction technique consists of generalizing the values of each pixelated grid into a smaller grid. Iterating through the original 28 x 28 matrix, the four pixel values in each ordinal direction are averaged and added to a new 14 x 14 matrix. This process is repeated, resulting in a 7 x 7 matrix. The pixels in both matrices are then added to the end of the original matrix, and the neural net is trained on this data.



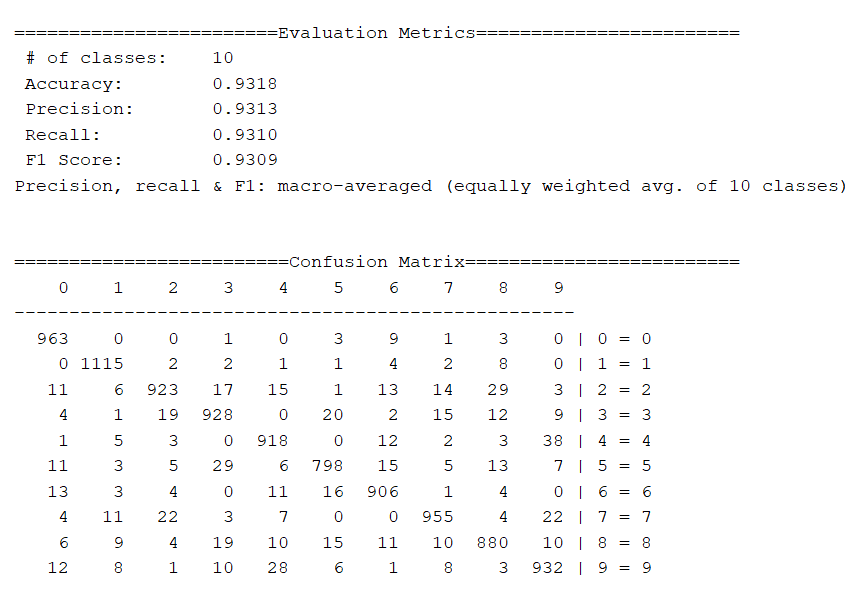
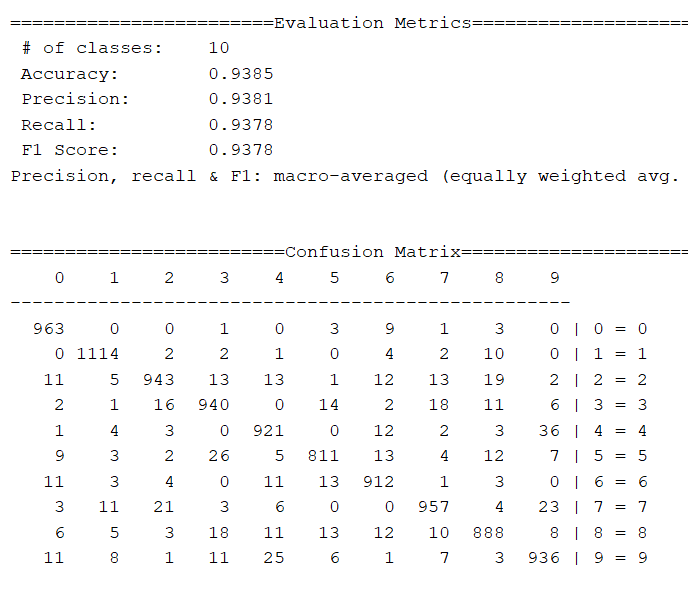


**Fig. 6**: Dimensionality reduction performed on the number nine

The ANN performed well with dimensionality reduction. Fig. 7, below, indicates some examples of output. Although the number of epochs increases over a wide range, the confusion matrices are all fairly similar, as are the accuracy and precision measurements. An especially interesting result is that a high score, above 90%, is achieved with only five epochs (Fig. 7a). The two previous datasets required 105 epochs in order to reach this level of accuracy and precision. Number of epochs is decreased twenty-one times; a fairly significant order of magnitude.

****

**Fig. 7a:** Results with 5 epochs **Fig. 7b:** Results with 10 epochs

****

**Fig 7c:** Results with 50 epochs **Fig 7d:** Results with 105 epochs

**Future Work**

The research performed in this study possesses limited scope; however, it is useful in providing a basic and foundational pattern for building, training, and experimenting with ANNs. Future work on this could take many paths.

One such path involves expanding on OCR techniques to build on some of the more advanced work found while exploring related research. [7][8]

Additionally, it would be interesting to transform musical scores into numerical datasets and train an ANN to recognize patterns within the notes. Perhaps the net could be trained to recognize a composer based on familiar melodies. A more ambitious future project might include teaching an ANN to generate music.

**Conclusion**

We have observed the results of a simple neural net when trained on a standard dataset, and two preprocessed datasets. With only labels and their corresponding pixels, the neural net was able to achieve an accuracy and precision of about 91%. With the region mapped pixel data, accuracy and precision were marginally better. The final preprocessing, however, which included the original pixel matrix and two subsequent dimensionality reductions, only took only five epochs to reach an accuracy/precision level of above 90%. When 105 epochs were used, the accuracy/precision level was nearly 94%. While this is a desirable score, it is only a marginal improvement on the high levels already achieved with five epochs. These results indicate that, when it comes to the information you give your neural net, more truly is more.

WORKS CITED

[1] Simon H., “WMC Fest IV-The-mixtape!”. [Online]. Available: <https://dribbble.com/shots/1156833-WMC-Fest-IV-The-mixtape/attachments/149933?project_id=146153>. [Accessed: 7-May-2019]

[2] [Online]. Available: <http://lettersandnumbers.org/college-style/numbers/college-number-4.jpg>. [Accessed: 7-May-2019]

[P] Redmon, J. C., “MNIST in CSV”. [Online]. Available: https://pjreddie.com/projects/mnist-in-csv/. [Accessed: 2-Feb-2019]

[4] L. Eikvil, “Optical character recognition,” 1993.

[5] M. Faisal, J. Anarase, M. Shingote, and P. Ghanwat, "Optical character recognition implementation using pattern matching", *International Journal of Computer Science and Information Technologies*, pp. 2088-2090, 2014.

[6] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping", *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06),* 2006.

[7] U. Meier, D. C. Ciresan, L. M. Gambardella, and J. Schmidhuber. “Better digit recognition with a committee of simple neural nets,” *2011 International Conference on Document Analysis and Recognition*, pp. 1250-1254, 2011.

[8] K. Chellapilla, M. Shilman, and P. Simard, “Combining multiple classifiers for faster optical character recognition”, *International Workshop on Document Analysis Systems,* pp. 358-367, 2006.

[9] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition”, *ICML deep learning workshop*, 2015.