Kaisei Sato

304740455

Psych 186B

Winter 2019

Classifying Handwritten Digits

Presented by Kimmie Hua and Kaisei Sato

Literature Review:

Initially we considered the study conducted by Hosseini and Xiao, their research on the “Limitation of Convolutional Neural Networks in Recognizing Negative Images.” Their study was the first to examine the limitations of neural networks on negative images. While humans can easily recognize the difference between regular and negative images as they have the same structure and semantics, the paper discussed how transformed inputs that semantically represent the same object, also known as Semantic Adversial Examples (such as negative images), and often the model does not recognize them correctly. The results of this study portrayed how as the model gets trained with more negative images, it becomes more accurate with recognizing negative images. However, the graph trained with regular images (fine tuned with negative), show that the regular image recognition decreases with increased negative image training.

Problem:

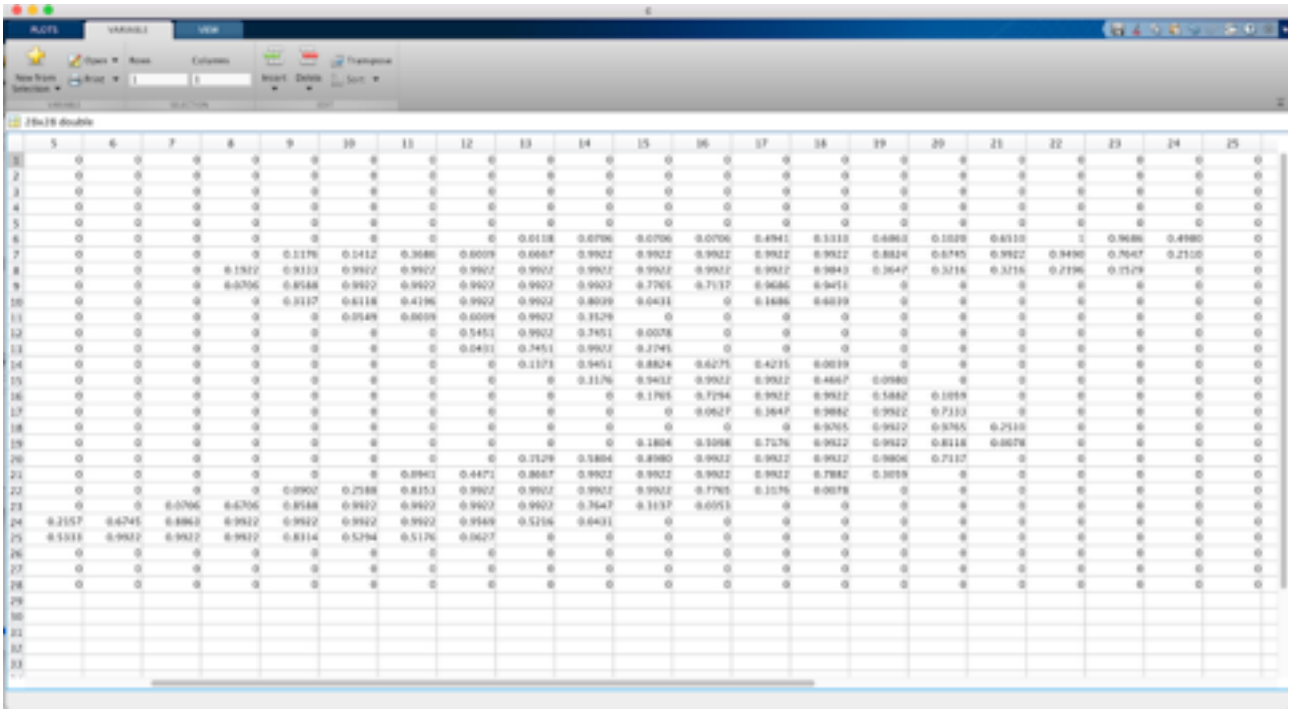
We recognised the fact that current training methods do not allow models to generalize certain concepts, as a regular digit 1 and negative image digit 1 sill represent the conceptual idea of “1”. This means that the model simply memorizes the pixel location for each specific digit, and is unable to “learn” to recognize same digit under different image conditions.

Therefore, my partner and I decided to observe the gradual color intensity change between white and black handwritten digits on the neural network, and how that relates to the performance of our network.

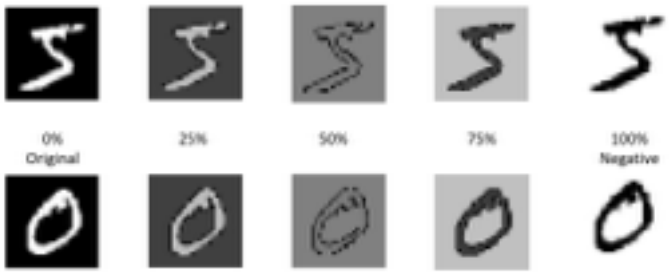
There were 5 different training and testing conditions, with each condition defined by the intensity in which the value is closest to black; 0% being the original image and 100% being the complete negative of the image. The handwritten image of each digit is represented as a 2-d matrix filled with numbers between 1's and 0's; the 0's represent black and 1's represent white.

Each individual conditions were then placed as the training data, in order to see its performance on every testing conditions.

- Condition A- 0% Negative (positive image)
- Condition B- 25% Negative / 75% positive
- Condition C- 50% Negative / 50% positive
- Condition D- 75% Negative / 25% positive
- Condition E- 100% Negative (negative image)



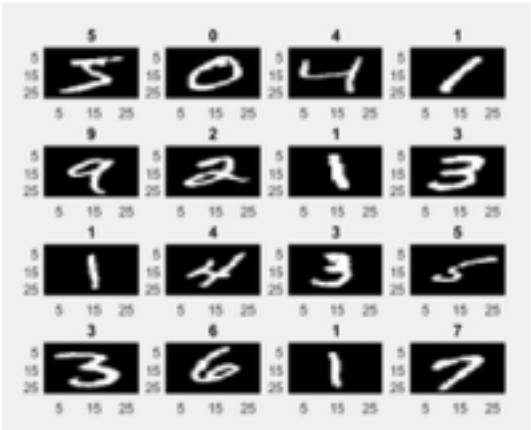
This is the 2D matrix of the digit 5. Every image is composed of a matrices of pixels indicating their "intensity" between 0 and 1, and their location on the image.



These are the 5 different training and testing conditions. Each digit is written in white, with the background black. As the conditions gradually change, the white pixels of each digit slowly turn to black, while black pixels of the background slowly shift to white.

Neural Network Architecture:

We had to include data preprocessing as we needed to parse the dataset. This way, we can work with it in Matlab and modifying our different training and testing conditions. We used the MNIST handwritten dataset, which included a training set of 60,000 examples and a testing set of 10,000 examples.



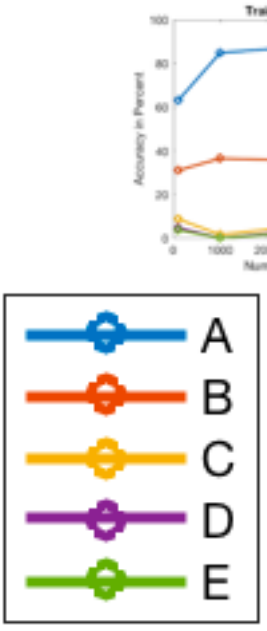
This is the original data. Every image has a corresponding label indicating what digit the image represents, so that the system can classify between digits.

The model had an input layer, which had 784 nodes. This was because our data was images centered in a 28 x 28 pixel image, meaning it had 784 pixels in total.
We used a feedforward network, used to apply the data to find the output of the hidden layer and the output layer. The activation function was used here on the output of the hidden layer and final output layer.

Then the feed forward network was used to apply the data to find the output of the hidden layer and the output layer. The hidden layer consisted of 100 nodes, meaning there were 78,400 weights associated. The sigmoid activation function was used here on the output of the hidden layer and final output layer, which included 10 possible digit outputs, ranging from 0 to 9.

Then back propagation was used to correct the neural network and allow the system to recognize the digits more accurately. Through this, the output is compared with the desired output that we already know, and the error is “propagated” back to the previous layer. This error is noted and the weights are “adjusted” accordingly. This process is repeated until the output error is below a predetermined threshold.

Results:[



The graphs display the results with each form of training. The y axis is labeled ‘accuracy in percent’, while the x axis is level number of trials, which is the number of learning trials. This was done as 100, 1000, and 5000 learning trials.

In general, the tests observed the best accuracy when the model was trained with the same class of modified data. In other words, A type tests performed best when the model was trained with A type training data and the same goes with any class. This trend was what was expected, as the model is able to recognise specific pixel values if the training data showed the same range of pixel values.

One important point to note is the D and E type conditions performed very similar, so much that on the graphs they seem to overlap. In these graphs, the accuracy of both conditions D and E overlap, at around 80-90%, while conditions A and B are nearly at 0%.

Another point, is that training under condition A and B resulted in higher accuracies for these same conditions, while training condition C caused a significant drop in accuracy within condition A and B, but a much higher accuracy in condition C (as expected), and D and E.

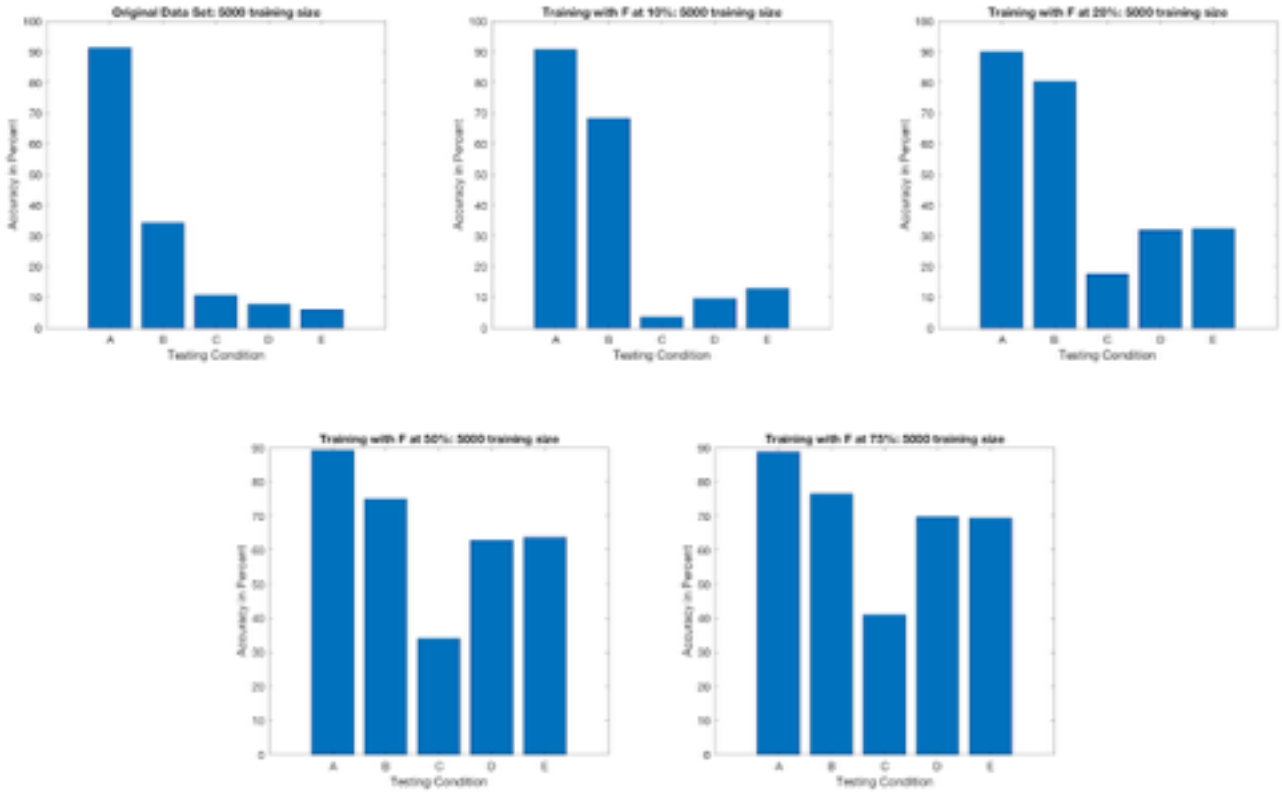
Discussion:

Derived from the results, we questioned why the model fails to recognise augmented data in the other conditions. The neural network is only sufficient in memorising the entire data set (its pixel values and its location), while the network is fragile to modified inputs where the model must generalise from the input data given. Therefore, since the network simply memories the pixel location and pixel intensity of each image, an augmented data input with different intensity in each pixel location will make it harder for the model to accurately define digit the data represents.

Since the results demonstrate that data augmentation causes the model to “semantically” overfit to the distribution of the training data, it leads to poor accuracy on negative images. As stated by the initial paper by Hosseini and Xiao, simple data transformations do not semantically augment the training data. Therefore, simply evaluating the machine learning models based on the test

accuracy can be misleading, as it does not take into account the how models can perform generalisation of concepts.

However, theoretically, if we train the model with all types of modified input, the model will have high accuracy; this is a fundamental limit of the network. Therefore, we decided to train the network with all five conditions of the training datasets to see its performance on each condition.



The graph above shows the final tests conducted where all training conditions are incorporated. We ran 10%, 20%, 50%, and 75% modified training conditions, and was only done with 5000 training trials. 10% indicates that 10% of the original data was modified at random such that 2.5% of the data falls under the condition B category, 2.5% falls under the condition C category, 2.5% under condition D category, and 2.5% under the E category. For 20%, this indicates that 20% of the original data was modified at random such that 5% of the data falls under the condition B

category, 5% falls under the condition C category, 5% under condition D category, and 5% under the E category—and so on for 50% and 75%. (the values for each category B,C,D,E distribution is just modified percentage value divided by 4).

As the graphs show, the original data set as training showed quite a low capability for the model to recognising augmented data, except under the original condition with an accuracy above 90%. However, with only 10% of the training data being modified to include data of other conditions, the accuracy of condition B significantly increases as well, rising to around 68% accuracy. At 50% of all training data being modified to include the other conditions, the network was able to recognise conditions A, B, D, and E with an accuracy above 60%. When the model was given all conditions with 75% modification of data as training, every condition except C increased to at least 70% accuracy.

The interesting finding in our data is the inaccuracy of condition C. One hypothesis behind this occurrence is that since condition C includes a pixelated image of a digit which has the least contrast between positive and negative image (black vs white). The condition therefore makes it much more difficult for the model to recognise the border pixels between the digit and background. Therefore, the average accuracy across different training data has condition C being extremely inaccurate, even with 75% of data across the other conditions has its accuracy at around 40%

An interesting proposal for further research is to conduct studies around a model testing the limits of other possible factors than negativity. Possibly they may look into transformation of images, resizing of images, rotation of images, etc. With more experimentation of the neural networks performance on augmented data, we are able to further test and understand their capabilities of conceptual generalisation.