CSC 3730

Machine Learning

Mingxuan Sun

Benjamin Alterman, Adam Kardorff, Jorie Noll, Uluc Ozdenvar

Handwritten Digit Classification Using Tensorflow

Our group approached this project with a general curiosity concerning neural networks and how they applied to the real world. We had little knowledge of neural networks beyond the understanding that they are an interesting application of deep learning that imitate the structure of the human brain. We researched many applications of neural networks, and decided to explore how neural networks are utilized to recognize visual patterns from a variety of images using convolutional neural networks. CNN visual recognition can be used for facial recognition software, document analysis, advertising, and much more. We decided to train a convolutional neural network that can recognize handwritten digits. Handwritten digit analysis can be utilized to create software that converts handwritten documents into digital formats. Document conversion software can be used to transform written notes into formatted text documents that archive important information digitally; this could be useful for students and professionals alike.

To train our convolutional neural network, we had to find a suitable dataset. The MNIST database of handwritten digits was a great choice to move forward with our project. The handwritten digit database has a training set of 60,000 examples, and a test set of 10,000 examples. Each example in the MNIST database is a handwritten digit that has been centered in a fixed-size image. This data is converted into a 28x28 matrix with values ranging from 0 - 255 based on density in order to achieve optimal utility.

Our project goal was to train a convolutional neural network that can identify handwritten digits from a variety of images. We aimed to do this by deploying a sequential model.

By definition, a convolutional neural network is a class of deep neural networks used most commonly to analyze visual imagery. Convolutional neural networks utilize multilayer perceptrons, which are generally fully connected networks in which each neuron in one layer is connected to all neurons in the next layer. A major advantage of convolutional neural networks in comparison to other image classification algorithms is the little amount of pre-processing required to learn filters, making CNNs relatively independent from prior knowledge and human effort. A sequential model is a deep learning approach which allows one to build a model layer by layer. In a sequential model, each layer has weights that correspond to the layer that follows it. These layers interact with each other at every level, so the current output is dependent on the previous input layer. A sequential model is appropriate for a plain stack of layers, where each layer has exactly one input tensor and one output tensor.

For our network, we used three layers for our prediction method. This layered network resulted in 99% accuracy with 7 epochs. The library used for this project was TensorFlow with the Keras API. MNSIT dataset contains images to be used in our data sets. The dataset was already divided into training and testing. We trained the neural network using our training data. To understand how our image looks, we first need to convert the input into a matrix form on a 2x2 matrix. On the matrix, a value between 0 to 255 is used to show the pixel density of each point in the matrix. The current layer must take a two-dimensional shape and flatten it into a one-dimensional array.

The rectified linear activation function (ReLU) is a piecewise linear function that will output the input directly if it is positive, otherwise it will output zero. It is advantageous to use the ReLU function over other activation functions because ReLU does not activate all of the neurons at the same time. For the hidden layer, 128 neurons were selected. The softmax function is a multiclass

logistic regression function that turns a vector of K real values into a vector of K real values that sum to 1. In the softmax function, the input values can be positive, negative, zero, or greater than one, but the softmax transforms them into values between 0 and 1, so that they can be interpreted as probabilities. 10 neurons in the output layer were used for 10 classes from 0-9. Normalizing the training data helps us fit our approach, so that instead of having values from 0 to 255, we have values from 0 to 1 so the model can be trained at a much faster and more efficient rate. The process of applying the sequential model to the flattened layer, followed by utilizing ReLU activation, and finally using the softmax function makes up the Tensorflow Keras model. Then, we must fit the model to the training data using TheAdamOptimizer. AdamOptimizer is a version of the gradient descent algorithm. Seven iterations later, we gather the accuracy for our training data and our loss.

A confusion matrix was used to display where the model misclassified the digits. The model correctly classified 970 (98.9%) 0s and misclassified 10. 1111 (97.8%) correct 1s and 24 misclassified. 1011 (97.9%) 2s correct and 21 misclassified. 987 (97.8%) 3s correct and 23 misclassified. 952 (96.9%) 4s correct and 30 misclassified. 865 (96.9%) 5s correct and 27 misclassified. 930 (97.1%) 6s correct and 28 misclassified. 1002 (97.4%) 7s correct and 26 misclassified. 957 (98.3%) 8s correct and 17 misclassified. 987 (97.8%) 9s correct and 22 misclassified. These accuracy numbers will slightly vary for each run through the model.

In conclusion, we achieved a 99% accuracy on the training data set, and a 97% accuracy on the testing data set. We now have a greater understanding of how deep learning can be achieved using large data sets. We deeper explored how neural networks have real world applications, such as in imagery recognition using a convolutional neural network. Our knowledge of CNNs and sequential modeling increased. We learned how to use Tensorflow and

the Keras API, which will be a valuable skill takeaway from this project as there are many industry applications.