Overview of CNN

Introduction

The question at hand here is it possible to train a computer to recognize hand written text? If a computer can be trained to do that, what other kinds of tasks can be performed with the knowledge gained? If those all can be done, how one use these features within context and apply them to pipelines or into business applications? It is with those questions in mind, the research for this paper began. We will be covering the history of people trying to solve this problem. One of the other more interesting aspects of the research for this paper was also participating in a Kaggle competition, and we'll be covering the lessons learned from participating as well.

Computer Vision is the art and practice of creating a computational model which mimics the human eye. Computer vision takes aim at an engineering point of view, and seeks to create autonomous systems which are able to replicate the tasks that the human visual systems performs. Computer vision is a field which is known as being notoriously difficult. There are almost no problems within the field that have completely and satisfactory solved. A good example of this is facial recognition. The human brain is able to break down facial recognition problems despite variations such as lighting, view points, facial expressions, and many other factors (Huang, 1996).

Kaggle competition is a competition platform for data science. They host data sets and a wide variety of competition types to the data science community in order to challenge individuals regardless of where they are within their data science careers. Kaggle hosts several competition types, including featured which are what they are best known for, which feature a cash prize. The getting started competition type is for some of the less complicated and more approachable types of problems trying to be solved with data science. They are meant to attract new comers and offer a host of tutorials associated with them ("How to use Kaggle", n.d).

The Digit Recognizer competition falls into the getting started category. The competition uses the MNIST ("Modified National Institute of Standards and Technology")  data set. It is known as the de facto "hello world" data set for computer vision. It was originally released in 1999. The data features 2 files, a test and train CSV. The data in the MNIST data set is a set of images of hand written numbers 0-9. Each image is a 28 by 28 pixel image, for a total of 784 pixels. Each pixel is a value between 0-255 based on the shading of the image. The goal of the competition is to use computer vision in order to classify the images of the MNIST dataset. The competition is scored by the number of correctly labeled images (KAGGLE).

**Background**

Neural Networks

Before digging into the subcategory of convolutional neural networks typically used for digit classification, it would be helpful to review neural networks as a whole. The neural network we will be examining is the perceptron. Perceptrons were first conceived in the 1950's by Cornell psychologist Frank Rosenblatt. A perceptron works by outputting a singly binary result from several binary inputs. The work that Rosenblatt brought to perceptrons was the use of weights. Weights are real numbers that are used to determine the importance of the input to the output (Nielsen 3, 2015).

When perceptrons are bundled together, they can be used for a surprising variety of tasks. While extremely powerful, one of the biggest drawbacks is that a slight change to the weight can change the entirety of the output. For example, within the context of digit recognition, say for example the number 4 is classified correctly, but not the number 9. A slight change in the weights of the model now allow for the number 9 to be correctly classified, but now the perceptrons no longer classify the number 4 correctly. This problem was solved by creating a new type of neuron similar to the perceptron called the sigmoid neuron. The benefit of using a sigmoid neuron is that the weights are able to be more fine tuned. This allows for a change in weight and bias to result in a small change to the output. By limiting the impact to the output, the network is able to be fined tuned as it learns. This takes the form of the input not longer being binary, but a decimal point value somewhere between 0 and 1 (Nielsen, 8, 2015).

When we bundle either perceptrons or sigmoid neurons, in essence we get what is generally referred to as a neural network. A neural network is typically comprised of 3 layers, an input layer, a so called hidden layer, and an output layer. The first layer is the input layer, and the neurons contained within this layer are called input neurons. The middle layer, known as the hidden layer can actually be comprised of multiple layers. The term 'hidden' is used because the inputs and outputs of these layers are not revealed. Last layer is the output layer, which are the ultimate results you are trying to achieve. Looking down to figure 4, we have 4 layers. The input layer feeds to 2 hidden layers, and a single neuron is the output layer. This type of network is often also referred to as a multilayered perceptron (Nielsen, 11, 2015).

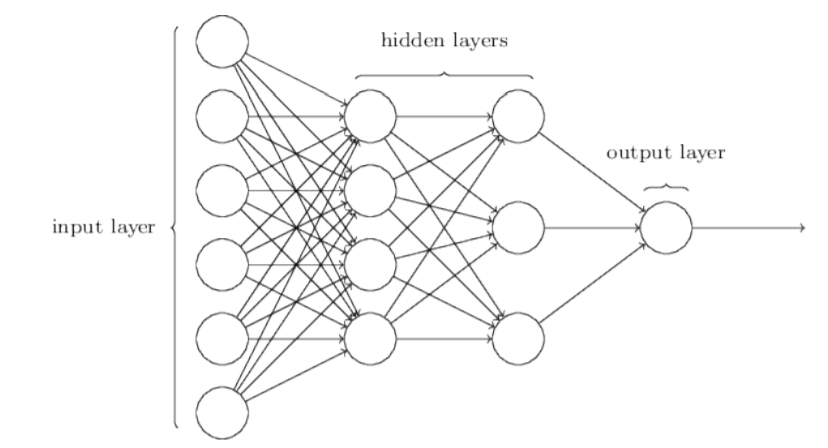


Figure 1: Nielsen, 2015

The techniques and image above help describe the original network types, and they're a sub class called feedforward neural networks. In this context, feedforward means that the data only goes in one direction. Data is fed to the input layer, and it only goes in one direction (Nielsen, 11).

When training a model, there needs to be a way in which feedback is passed in order for the model to adjust. Models are typically scored with what is known as the Mean Squared Error (MSE). The MSE is a measure of how closely a fitted line matches to its data points. For every value, you take the distance vertically, and then square the value. Those values are then added up and averaged. The smaller this metric is, typically the more accurate your model is (What are Mean Squared Error and Root Mean Squared Error?, N.D).

As you train your model, in order to reduce the MSE, an algorithm known as gradient descent is typically used.

BACKPROP

The technique of back propagation is the short hand term for backward propagation of errors. It is a technique which uses the gradient of descent for weighting the different variables. The process of back propagation is essentially a learning algorithm. Within a neural network, the process repeatedly adjusts the weights that the network uses to make predictions. The goal is to minimize the difference between the actual output and the desired output. This technique was first described in David Rumelhart's landmark paper in the 1988 (Rumelhart, 1988).

References

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541-551.

Huang, T. (1996). Computer vision: Evolution and promise.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. Cognitive modeling, 5(3), 1.

What are Mean Squared Error and Root Mean Squared Error? (n.d.). Retrieved from https://www.vernier.com/til/1014/