Building Recognition Using Convolutional Neural Networks

Sean McGuire
Machine Learning 1
December 26 2017

***Abstract*—This paper will discuss Convolutional Neural Networks and their application in image recognition. The performance of this techniques was then analyzed to the application of building recognition. The network constructed was able to achieve an accuracy of 89.6% on images never before seen by the network.**

I. Introduction

THIS paper demonstrates the functionality, and implementation of convolutional neural networks (CNN). The CNN is constructed using keras, a wrapper for TensorFlow. Images of seven buildings around the Rowan University campus were used to train and test the network. These images came from angles covering all sides of the buildings. The buildings used for this project are: Rowan Hall, Bunce Hall, Business Hall, Science Hall, James Hall, Chestnut, and Robinson Hall.

II. Background

Neural Networks consist of a series of nodes connected to each other. Each connection holds a weight and these weights are trained to the optimal values which will allow the network to take input data and process it through the network, the final output of the network will result in a classification. Convolutional Neural Networks (CNNs) are a type of deep neural network. Deep neural network means that the layers of nodes are stacked on top of each other resulting in a "Deep" network with hidden layers. [1]

A. Applications

Different applications for convolutional neural networks include, Image recognition, natural language processing, and recommender systems. Convolutional neural networks are currently one of the best methods for image recognition. [2]

B. Layers

Convolutional Neural Networks are deep neural networks so they are constructed by stacking different layers. The layers used in a CNN are convolutional, activation, pooling, and fully connected.

1) Convolutional: The convolutional layer is constructed of many nodes which are then connected to a set of nodes in the next layer. This can be seen in figure 1.

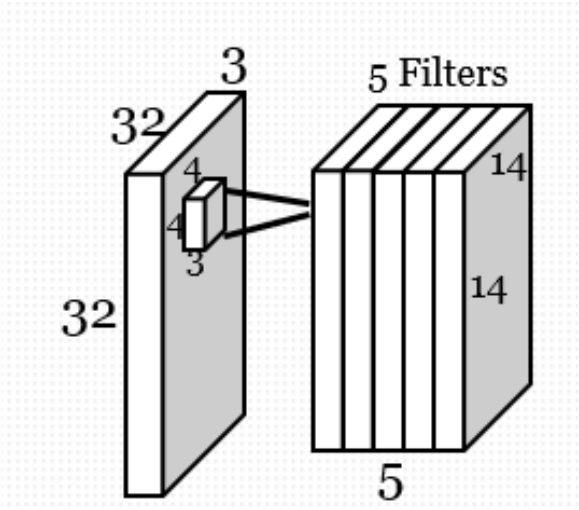

Fig. 1. Example of Convolutional Layer

This figure shows how each set of 4x4 pixels is connected to one node in the next layer. The depth of the next layer is determined by the number of filters, for 5 filters each block of 4x4 would be connected to 5 output nodes. The parameters that each convolutional layer requires is the width, and height of the filter, and the number of filters. Another parameter that was not utilized in this network is the padding. The padding applies zeros around the edge of the layer. The size of the output layer can be determined by using equation 1.

$$Output = \frac{InputSize - FilterSize + 2 * ZeroPadding}{Stride} + 1 \tag{1}$$

2) Activation: Activation functions are used to manipulate the data coming from the previous layer. The output size of the activation layer is equivalent to the input layer size. Two types of activation functions were utilized in this network, Rectified Linear Unit (ReLU) and softmax. ReLU is demonstrated by figure 2. [4]

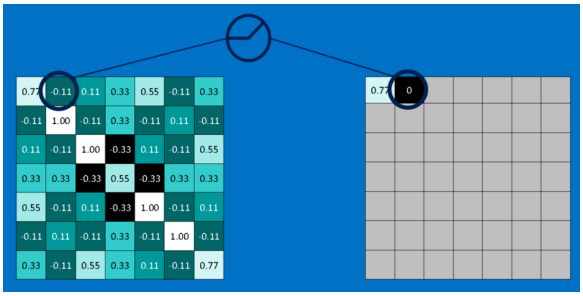

Fig. 2. Max Pooling Example

The ReLU activation function is applied after every convolution layer in the network. ReLU takes the output of the layer and sets any negative values to zero. This activation function can be computed very quickly when compared to the other activation functions such as the softmax activation function. The softmax activation function is utilized in the last fully connected layer. This activation function shows the distribution of all classes. The highest value is selected as it is the most probable class for the image. The softmax function can be computed using equation 2.

$$\sigma(z)_j = \frac{e^{zj}}{\sum_K^{k=1} e^{zk}} \tag{2}$$

3) Pooling: Max pooling was utilized in this network. Max pooling takes in the parameters of the length and width of the pooling square, the max value in this square is then put into the next layer. Pooling was needed in the network constructed as it reduces the dimensionality of the data meaning that there are less weights which need to be calculated, this results in a decrease in runtime and memory needed for the network. 2x2 max pooling can be seen in figure 3. [3]

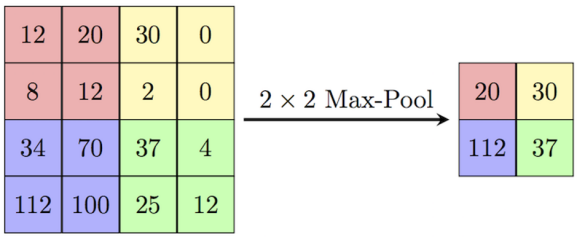

Fig. 3. Max Pooling Example

4) Fully Connected: In the fully connected layer each node is connected to all nodes in the next layer. This type of layer requires many more connections per node than convolutional layers. This is why the fully connected layers are not implemented until the end of the network where the layer dimensions are much smaller. It is important to note that the 2-D layer of nodes must be converted to a vector of nodes before a fully connected layer can be used. An illustration of this layer can be seen in figure 4. [4]

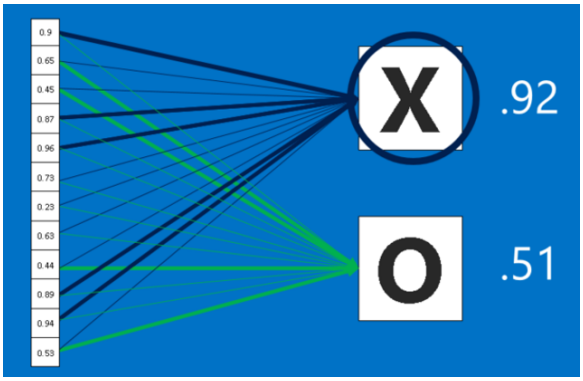

Fig. 4. Fully Connected Layer

III. Project Design

The goal of this project was to take pictures of different buildings around the Rowan University campus and have a CNN classify the images.

A. Constraints

Some constraints were set on this project such as runtime, and processing power. When constructing the network it was found that the size of the network was constrained. After the first couple iterations of design it was found that if the network was made bigger than the final iteration of the design it would be too big to fit on the graphics card which was used to train the network. This is because the number of weights which would have to be trained exceeded the total memory on the Graphical Processing Unit (GPU). The network took 6.5 hours to train on average. This factor limited the number of iterations of the network design. During the testing process around 15 different sets of parameters were utilized which amounts to 4 days of run time.

B. Standards

This project has followed a few standards. One standardized process was taking pictures. All pictures used in this experiment were 2988x5312 RGB resolution taken in landscape. Keras, a python library, was used to interface with TensorFlow. This library made it easier and faster to rapidly build the convolutional neural networks than using TensorFlow directly. Another standard utilized is the training/test data split of 70/30%. In this case 193 training images and 84 testing images were used.

C. Network Architecture

The network is constructed of convolutional, max pooling, Activation, and Fully Connected layers. The CNN architecture can be seen in figure 5.

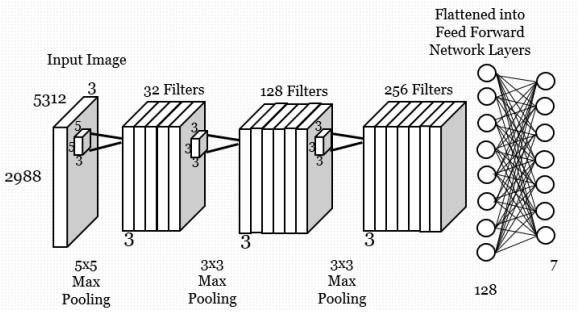

Fig. 5. Image of CNN Architecture

IV. Results

When running this neural network through the test data it was found to have an accuracy of 99.4%. When the testing data is passed through the network it was found to have an accuracy of 89.2%.

A. Misclassification

In the training set there was only one misclassified image. This image was investigated. The image in question can be seen in figure 6, it is clearly an image of the business building from a farther distance.

Fig. 6. Misclassified Image of Business Building

The network classified this image as Robinson Hall. After looking at the training set it makes sense how this image was misclassified. There was only one image of the business building from a farther distance like the one in figure 6. It is also important to note that there were 3 images taken from farther distances of Robinson hall such as figure 7 that were used to train the network.

Fig. 7. Image of Robinson Hall

The features in both pictures show the grass and sky with the building at a farther distance from the camera, and even the colors appear to be similar. This is most likely how the image of the business building was classified as Robinson Hall by the network.

B. Output of First Convolutional Layer

It is interesting to visualize the data moving through the network. To do this the image seen in figure 8 is passed through the trained network.

Fig. 8. Original Image Passed through Network for Visualization

Some of the outputs of the first convolutional layer can be seen in figures 9, 10, 11. These images show the output of 3 of the 32 filters which exist in the first convolutional layer. It can be seen from the images that each one detects different features of the image, for example figure 9 detects many color differences between different areas of the images. Figure 10 contains information about the edges of all of the areas of the image. Figure 11 shows the distinction between the sky and the rest of the building.

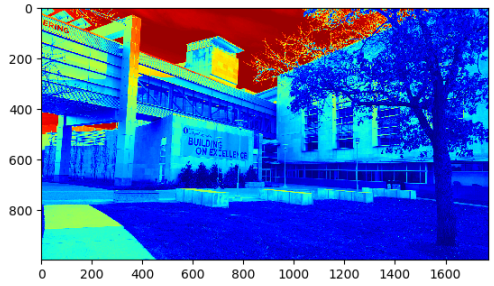

Fig. 9. First filter output of First Convolutional Layer

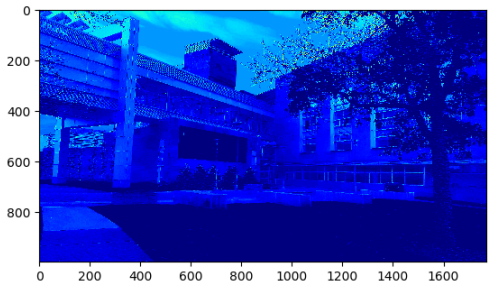

Fig. 10. Second filter output of First Convolutional Layer

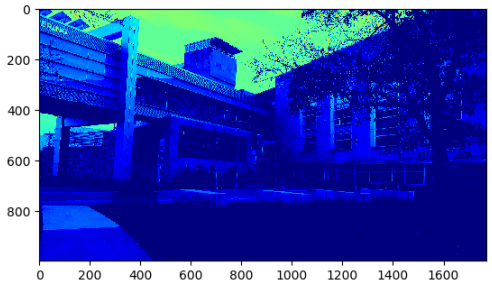

Fig. 11. Third filter output of First Convolutional Layer

C. Performance

The two charts on the left side of figure 12 show how the accuracy and loss of the training data increase and decrease as the number of epochs increase. The next two charts in the figure show the changing accuracy and loss over time for the test data (Not seen before by network).

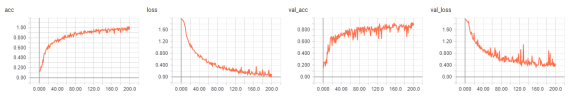

Fig. 12. Tensor Board graphs of training

Figure 12 shows that the accuracy and loss improve very rapidly at the beginning, but then as time elapses they start to level off. It took a GTX 1060 with 3 GB of memory 6.5 hours to train this network. The best model was trained for 198 epochs with the stop condition set on loss, this means that it will not stop training until 15 epochs pass without the loss decreasing. Some different models has been saved as a .json file and are available at the GitHub Repository in the appendix.

V. Conclusion

From this project it was found that CNNs performed very well when used for image classification. The network designed was able to achieve training accuracy of 99.4% and test accuracy of 89.2%. Some downside were also realized throughout this project. One unfortunate limitation is that the resolution is bounded to that of the camera used for the project. This means that the network will not work for pictures taken on a camera from a different phone. Another disadvantage is the training time and memory usage.
If time and computational resources are not an issue, convolutional neural networks can achieve very high accuracy and are extremely effective when used for image classification.

Appendix A
GitHub Project

Trained Models and python code can be found at https://github.com/mcguires5/MachineLearning1 in the Building Classifier CNN folder. The dataset used is very large, please contact mcguires5@students.rowan.edu if you wish to access the images used for training and testing.

References

[1] L. Hardesty, “Explained: Neural networks,” MIT News.
[2] S. Hijazi, R. Kumar, C. Rowen, and I. P. Group, “Using Convolutional Neural Networks for Image Recognition,” Semantic Scholar.
[3] Veličković, P. (2017). Deep learning for complete beginners: convolutional neural networks with keras. Cambridgespark.com. Available at: https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html.
[4] Data Science and Robots Blog. (2017). How do Convolutional Neural Networks work?. [online] Available at: http://brohrer.github.io/how_convolutional_neural_networks_work.html [Accessed 26 Dec. 2017].