A Comparison of CNNs for Image Classification

Holt Skinner *University of Missouri – Computer Science 8780*

Introduction

Neural Networks and Deep Learning are the biggest buzzwords in Computer Science today. One of the particular fields that shows a great deal of promise are Convolutional Neural Networks or CNNs these special neural networks use a process known as convolution to learn features in conjunction with traditional Back Propogation. CNNs are particularly useful for processing images because convolution itself treats the data as a 2D matrix. Many different pretrained models exist to process image recognition and they each provide different levels of accuracy and different specialties. There are also a number of APIs featuring pretrained networks to allow more developers to utilize image recognition technology. For this project, these different models and APIs will be compared and contrasted to determine the best use cases.

Diagram of a Basic Convolutional Neural Network

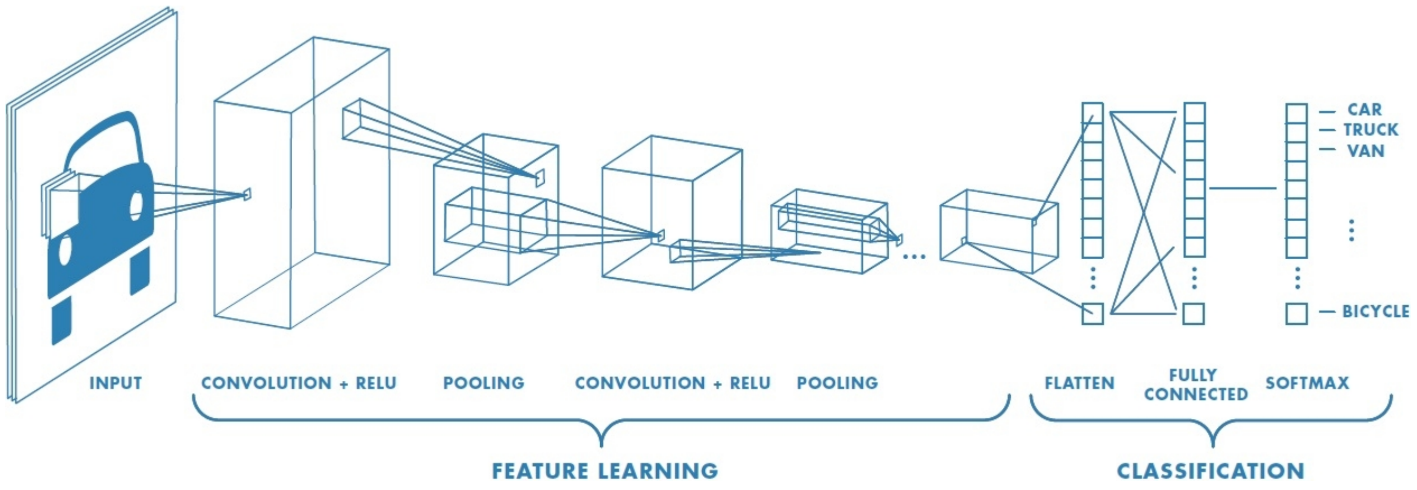


Image Source: Mathworks

Dataset

For simplicity purposes, all classifiers will be trained with a simple dataset that consists of cats and dogs. The classifiers will be analyzed based on their ability to differentiate between the two different creatures. The training set imcludes 8000 images, 4000 cats and 4000 dogs. The test set contains 2000 images, 1000 cats and 1000 dogs. While this dataset is simple, the purpose is to learn the process of the CNN and the basic construction, as well as comparing the models.

Technologies Used

- Python
- TensorFlow

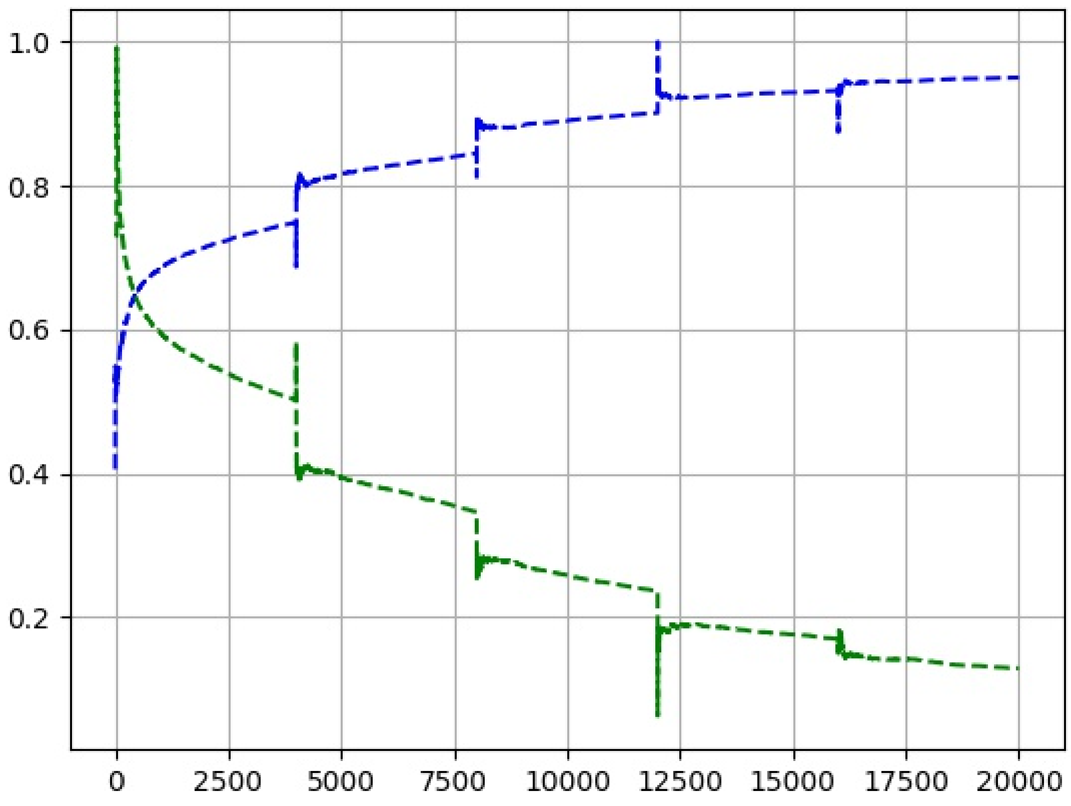
- Keras
- Numpy
- Google Cloud
    - All computation was performed on a Google Cloud Compute Instance running Ubuntu 18 to speed up computation.

Homemade Image Classification with Keras

For the first attempt at image classification, I used the Keras Library on top of Tensor Flow to create a basic image classifier. This Convolutional Nueral Net contained 5 Layers: Convolution (with 32 filters 3*3 each), Max Pooling, Flattening, Fully Connected and Output (Also Fully Connected)

Training this classifier took quite a while even on a high-equipment machine. The classifier was to be trained for 25 epochs on the dataset. Each Epoch took approximately 40 minutes to complete. I first had an issue of my laptop going into sleep mode and killing the process when the ssh connection went down. To avoid this, I found and utilized the Unix command `nohup` which prevents the program from quitting when the shell is exited. However, I then had another issue of accidentally killing the process myself when working on another experiment, to save time, I reran the homemade algorithm with half the dataset and 5 epochs to get a proof of concept.

Below are the graphs depicting the loss (green) and the accuracy (blue) when training and testing. These values were outputted by keras after each datapoint was plugged into the network. Notice how there is a spike at each epoch (every 4000 datapoints) this makes sense because the data is shuffled at each epoch to allow for more variety in training and to prevent an overfit. It is also important to note that the CNN got up to 95% accuracy according to Keras.

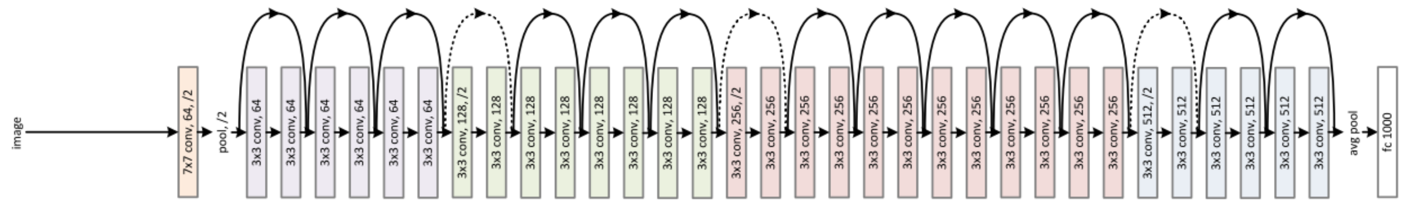


As great as these results seemed, it was not meant to be. When I attempted to run my own cat images through the network, it would very rarely classify them correctly. Upon further investigation, the way I attempted to limit the training data size in Keras actually only trained the network on dog images. I had to completely start over and retrain the network on the full set of images. The process went overnight and took more than 12 hours to fully train.

Resnet50

Resnet50 is a Deep Residual CNN which won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2015. In the competition, it achevied an error rate of 3.57% on the ILSVRC Dataset, which was actually better than human-level control. The Resnet was pioneered by Kaiming He et al and presented in the following paper https://arxiv.org/abs/1512.03385 As the name suggests, this network has 50 layers with a combination of convolution, relu and pooling along with fully connected layers. The Resnet's claim to fame is the fact that data passing through the network can skip through some layers in the network to avoid the vanishing gradient problem. (e.g. Layer 2 can directly connect to layer 6) Residual Networks have been becoming popular due to this phenomeon.

Diagram of ResNet50



For the experiments of classifying dogs and cats, I used a pretrained network on the ImageNet dataset. The advantage of using a pretrained model is the amount of time saved from not having to train. Especially since the deep nets (including ResNet50) can take days or weeks to fully train. However, this dataset provided an interesting challenge in testing because all of the classes in ImageNet are highly specific. For example, there is no "dog" class, but rather "Siberian Husky." To make a more apples to apples comparison, each prediction had to be analyzed for the different types of classes that could be categorized as a dog or a cat. With this further analysis, the accuracy in the Dog/Cat dataset achieved an accuracy of 80% for both cats and dogs. An Unexpected issue occured because Resnet50 in Keras expects images of 224 * 224 pixels, and the Python Error messages weren't entirely clear.

MobileNet

MobileNet is a Neural Network model that was designed specifically to be small, low-latency, and low power for use on mobile devices while still being close to accurate as larger networks. The MobileNet first appeared in a paper in April 2017 by a group of reasearhers at Google. https://arxiv.org/abs/1704.04861 The interesting differences between this network and standard CNNs are in the convolution steps. MobileNet does one standard convolution as the first layer, but then it does a "depthwise separable" convolution for every other convolution. The first step involves performing convolution in each channel (RGB) separately, rather than all at once.

Depthwise Convolution

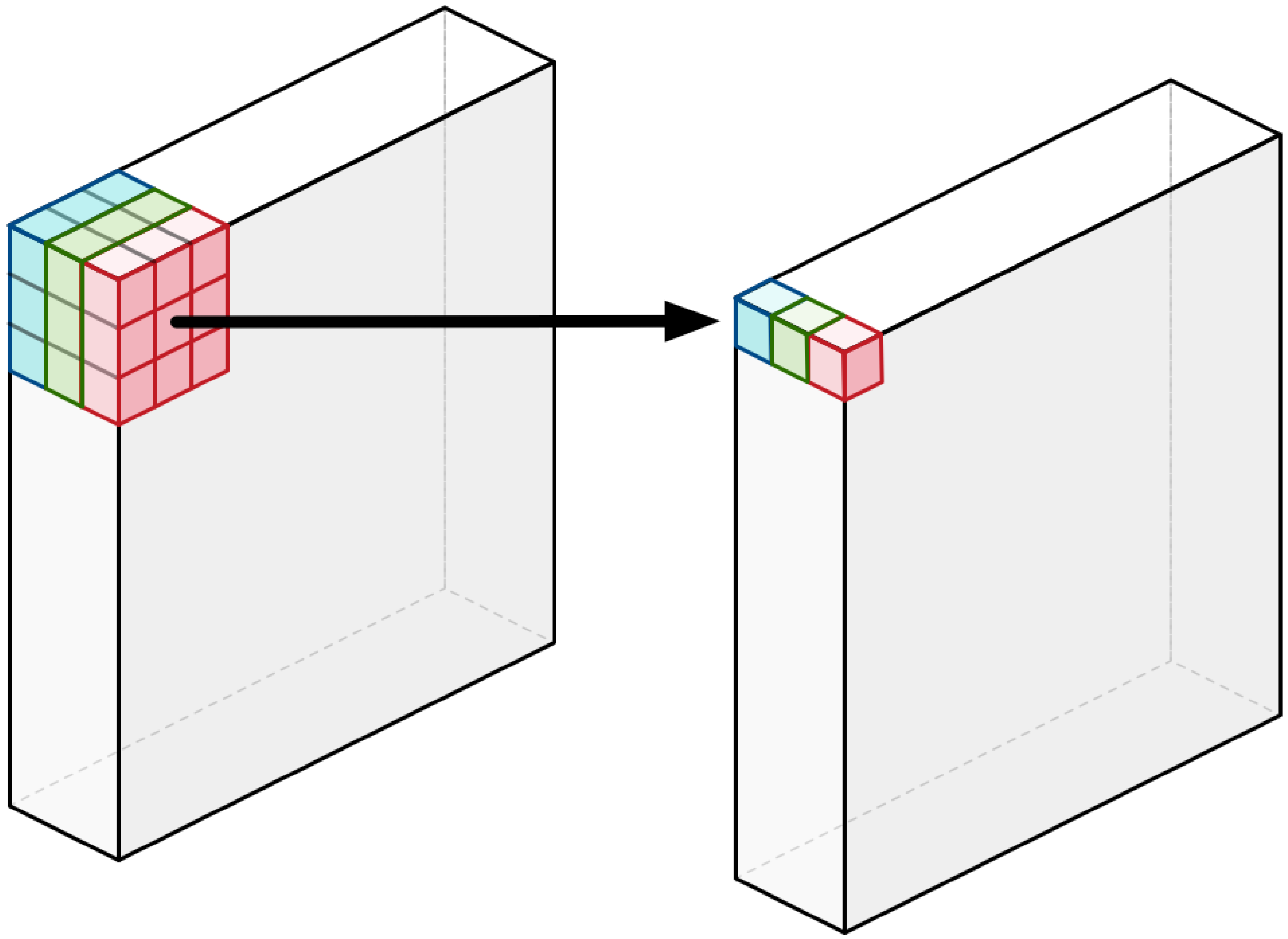
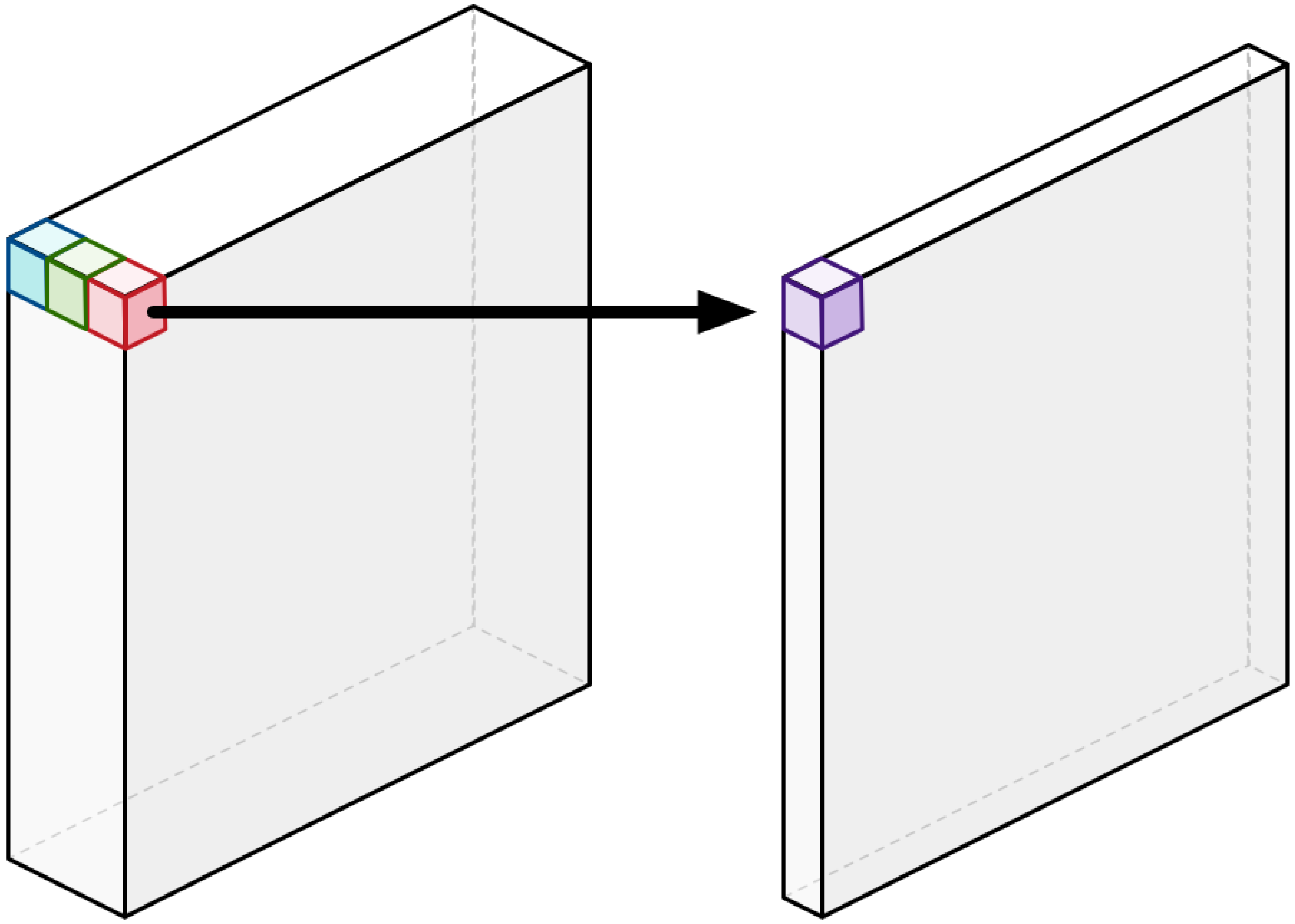


Image Source: Machinethink.net

This operation is followed by a pointwise convolution which adds up all the channels as a weighted sum.



Google Cloud Vision API

To make image recognition even simpler for developers who may not have a machine learning background, companies such as Google and Amazon have released APIs that provide an easy interface for predicting objects in images. Images can be sent either via a url or encoded in base64, which python provides natively. The API requires registering for an API key, but provides the service free for the first 1000 images every month. This limit did prevent testing on the entire dataset. On my Google Cloud Instance, the results came back almost instantaneously for an image, even though the image needed to be encoded before sending. The only bottleneck found when testing was that the service can only classify 16 images per request, so this required a queueing system be implemented to allow more images be tested at once. The Google API correctly classified 100% of the training set cats and dogs submitted (100 of each). Interestingly enough, all of the cats were classified with above 97% confidence, whereas the dogs classification confidence ranged from 90 - 99%.

Summary

Sources