Breanna Shinn

Prof. Xinchao Wang

CS 559A

04 May 2020

Image Classification by K-NN and CNN Models

Overview

Image classification has become an important machine learning task in recent years, with applications in automated image organization, optimizing search results, and facial recognition software. Another important use of image classification is checking whether current CAPTCHA and HIP security tests are still viable. In the case of the Kaggle Dogs vs. Cats data set I used in my tests, experiments in image classification showed that the HIP test, which asked users to select the dogs/cats in photos, was no longer a secure way to check if the user was not a bot. Since it used to be near-impossible with previous knowledge of image classification to categorize these photos with above a 60% accuracy, the HIP used to be effective in determining whether the person was a bot, but now with current knowledge of machine learning, a moderately simple model can perform with over 80% accuracy. It is also important to determine which algorithms are the most effective, in order to create effective software, but also to know which models to test on digital security systems. My solution was to examine the differences in accuracy in a simple classifier (k-Nearest Neighbor) and a more complex classifier (convolutional neural network) to determine how each performs in sorting images from my chosen data set.

Problem and Solution

In the experiment, I tested the accuracy of using a k-Nearest Neighbor classifier to classify images versus a simple convolutional neural network. With this experiment, I can determine how much better an advanced classifier like CNN can perform as opposed to a simple classifier like k-NN, and draw conclusions about why the algorithm is more accurate. I tailored these algorithms to the Kaggle Dogs vs. Cats data set and used subsets of this data set to train and test the models. The data set is a collection of pictures of dogs and cats, with a group labeled for training purposes, and a testing set without labels. It was simple, since it involved only two classes, dogs and cats, which made it easier to analyze the results of the tests. The data set was also large enough to perform many independent trials on the model to see how it performed on average. For each model, I used a subset of 6000 photos(3000 cat, and 3000 dog) and split it in half to form my training and testing sets. I also ran each model 10 times to average success rate over different testing sets.

Descriptions of Models and Implementation

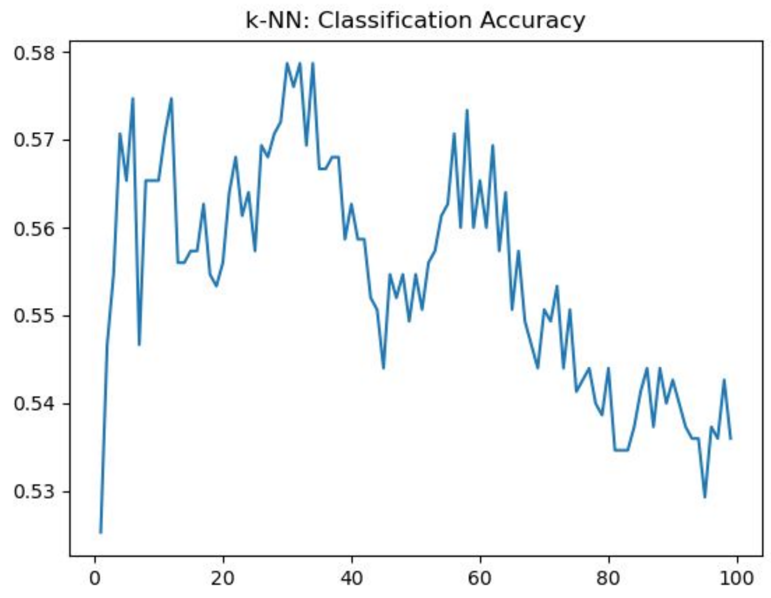
The k-Nearest Neighbor classifier is an early classifier, first introduced in 1951 as the first non-parametric method for pattern classification. I chose it for my test since it is a simple, non-parametric method that contrasts the CNN method well. The k-NN model works by sorting the training data, then using the training set to compare to each element in the testing set in order to find its “nearest neighbors” or the elements most similar to that element.

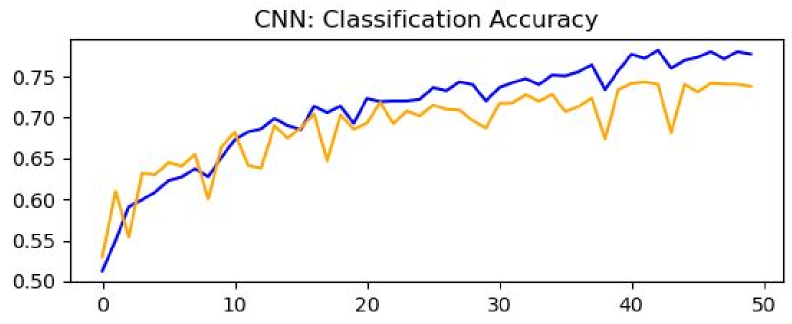
I started by reading the images and transferring them to grayscale, then assigning them to their appropriate training and validation categories. In order to find my model's maximum accuracy, I created a function which finds the best k value for the data, then applied that k value to my classifier. This step ensured that I was getting the maximum accuracy according to my training and validation stages. Once I assigned the k-value, I reshaped my data and created my kNN class, which contains functions for training and predicting, along with some helper functions. I then have a classifier which utilizes this class to make predictions about specific images. I also calculate and graph the accuracy of the test in order to compare its performance to the CNN model.

In order to contrast the k-NN model, I chose the convolutional neural network model. First implemented in the 1990s, CNN models were found to be incredibly useful for image recognition and classification. CNN models work by passing the images through different convolution layers with filters known as kernels. While simple CNN models require certain kernels, this also allows the model to increase in complexity by adding additional kernels to aid in edge detection and blur reduction for the set of photos. I decided to build and test a simple form of the model in order to compare the initial improvement of using a CNN model to the previous attempt at using a k-NN model.

In order to form the CNN model, I used keras to create a sequential model and the layers needed to form my model, which included two max pooling layers, two convolution layers, a flatten layer, and two dense layers. In order to easily create my training and testing data for this test, I used image data generators to grayscale and standardize the size of the photos pulled from my training and testing files. After defining my model, I created an accuracy function that reports on the accuracy of my tests, and graphs the results so I can compare it to the k-NN model graphically and numerically. I also defined a function that would train, test, and output the accuracy of the given training and testing data sets.

Results





The graphs show the accuracy of the k-NN and CNN models during a validation stage. After running 10 trials, the average accuracy of the k-NN model was 57.9 percent. After looking into how other k-NN models have performed in similar image classifications, it appears that my results were roughly average. The CNN model performed much better, with an average accuracy of 73.8 percent. This score is also roughly the average accuracy of simple CNN models in similar image classification tasks. While the k-NN classifier did not perform much better than random-guessing, the CNN model performed well-over random guessing, always scoring with an accuracy over 70 percent. As can be seen by the graph, which represents one trial, the CNN model appears to be improving over the course of the training and validation stage, with its accuracy, on the y-axis, increasing as it runs through the data sets. The k-NN model, however, does not show much improvement, but rather converges at a success rate a little above 50. These graphs, while only representing one trial, have approximately the same curve in each trial and show that the k-NN model does not seem to respond nearly as well in the training stage even when given more data.

Interpretation

The results of the experiment show that the CNN classifier is more effective at classifying the Kaggle Dogs vs. Cats data set. Since the CNN model specializes in image classification, it starts with an advantage on the k-NN classifier, which is meant to take a broader range of data. The CNN model has many tools for specifically classifying images. Since images have excess noise, the k-NN classifier may have trouble discerning what elements of the image make it similar to other images, and may not be judging on the subject of the photo. The CNN model also contains more layers which allow for more specification and further editing of the photo to make the photos more readable by the model.

Future Improvements

The main approach I would take to improve the k-NN model would be to train my model on a larger training set in order to increase the number of images with which it can compare new images. To ensure I had sufficient time to test my models, I limited my training data set to a subset of the Kaggle Dogs vs. Cats data set. With more time, I could run it on the full training set and increase my accuracy up to about 60 percent.

The CNN model, due to its optional layering, can be improved in many more ways and up to approximately 93 percent accuracy on the Kaggle Dogs vs. Cats data set. Training my model on more data would increase its accuracy as it would ensure I do not overfit my data to a small data set. My simple model also shows signs of underfitting since its validation accuracy is below its testing accuracy. This inaccuracy can be amended by adding additional convolutional layers and adding dropout layers.

While more complex models can perform at much higher accuracy rates than the simple k-NN and CNN models in this experiment, the CNN model performed surprisingly well in spite of its simplicity. In the future, I would like to look at building CNN models for more complex data sets with more than two classes as well as improving my model's accuracy on the Kaggle Dogs vs. Cats data set. I would also like to look at using k-NN classifiers on other types of data sets to see if it performs better or worse.