Project Pizza

Team Members: Brenden Guillen, Jonathan Thornton

Introduction and Motivation: Our project will be classifying different pizzas based on images of said pizzas. We feel the internet is full of weird people that like pineapple on pizza, and we would like to be able to sort them out from the rest of the population. In all seriousness, a pizza image classifier could have many applications in the food industry. Some of these applications could include quality checking, allergen detection, food grading, etc. This classifier should be able to tell the difference between a handful of different pizza types (cheese, pepperoni, hawaiian, black olive, and taco).

Methods and Algorithms Involved: We used a few different methods to help guide the model. One was K Nearest Neighbors (KNN) from class, in which we will break the image down into its base colors, and based on the number of different colors (ie, if there is a large amount of yellow, it may be a pineapple pizza, or if there is a massive amount of white, it may be a plain cheese pizza) it may be able to predict the type of pizza. This paired with the softmax multiclass classification model may help to classify pizzas into their correct category. We tried to use shape classification with the help of OpenCV to help identify the pizza itself in the image, and the variety of toppings on it (ie, red round shapes may be pepperoni) to help identify the pizza, but we were unable to make this work properly with KNN. We also used Convoluted Neural Networks (CNN) as a second model in this assignment. This was relatively simple in its design up front, but used more resources than a KNN (in both training the model and using it to classify images). Our CNN took in an image into 3 convolutional layers that uses small filters (called kernels) to detect features (like shapes, edges, etc.), the last convolutional

layer then feeds into a pooling layer to reduce the size of the processed data while maintaining important features, and this feeds into a set of fully connected layers to further process the input and then uses a softmax activation function to produce the predicted pizza type. We initially used a model built with the tensorflow python package, which after trying to get it to run on a GPU to expand the size of the network, we were unsuccessful in doing so. We also tried to use a PyTorch model, which we were able to train on the GPU (and expand the model size greatly due to this), but we were unable to work out the bugs in the testing code and did not use this as our main model (but code for it is included under "Not Working Model"). We had to settle with the CPU-bound smaller CNN model for our project.

Data Description and Source: We used 2 different data sets (both a pizza dataset and the MNIST data set). We initially tried to create a script to download 100 pizza images through Bing and give it a general search of "[pizza type] pizza" (ie, "cheese pizza"), but this script would only download about 40 images, with varying amounts in each category (the script is included). This led us to download all the images (using Google instead of Bing) manually using 2 different search terms, both "[pizza type] pizza" and "[pizza type] pizza top-down view" to get a larger and better data set for our models. We kept the size to 100 in each category with a few extra downloaded to test the models out manually. We also tried to use the MNIST data set, but our augmentation would not work properly with the built-in MNIST data set that is available in most neural network packages in Python. Due to this, we also had to download the MNIST data set to be able to run our tests on it. We also limited the MNIST data set to fit the number of categories and size of our pizza data set in testing (so to 5 categories and 100 images for both testing/training the model) to compare the 2 datasets more closely. We also decided to augment the data sets to see if different augmentations would help or hurt the model's results. We were

going to train/test the model with AI-generated pizza images as well, but decided to not use this in the project (but kept the generated images in the dataset included)

Results: When testing our models with various different changes, we obtained the following results. For our KNN model (with k = 20), we achieved a training accuracy of 35% and a testing accuracy of 30% when using our pizza dataset. For our CNN model (with epochs = 20), we achieved a training accuracy of 99% and a testing accuracy of 54%. The high training accuracy and low testing accuracy is most likely a sign of overfitting, as the model is not able to generalize from the training data too well. To overcome this overfitting, we decided to perform data augmentation on our pizza dataset and use this augmented dataset to train our model. Training with the augmented data allowed the model to generalize much more, as the training accuracy reduced to 76% while the testing accuracy increased to 62%. To further prove that our CNN model would work well for other datasets, we tested it with a batch of images from the MNIST dataset, with the batch being the same size as our pizza dataset. The training accuracy was 100%, and the testing accuracy was 94%. Possibly a little bit of overfitting, but good results in general. When testing our KNN with the MNIST dataset, the training accuracy was 92% and the testing accuracy was 93%. Clearly, when using an excellent dataset like the MNIST dataset, the accuracy of the model will be much better. The last thing we tested was augmenting the reduced MNIST dataset and seeing if it would improve the accuracy of the CNN model. The training accuracy was 88% and the testing accuracy was 92%. To our surprise, the accuracies were lower in comparison to the un-augmented dataset. This may be because the transformations being applied to an image may not be realistic, and are not similar to anything in the testing dataset.

Conclusion: From our results, we conclude that CNN models are great for image classification, the quality and size of the dataset makes a huge impact on the accuracy of the model, and data augmentation may benefit the model if the transformations give the model a realistic idea of what it might see in the testing data.

References

https://www.geeksforgeeks.org/how-to-detect-shapes-in-images-in-python-using-opencv/#

https://pyimagesearch.com/2016/02/08/opencv-shape-detection/

https://pyimagesearch.com/2016/09/12/softmax-classifiers-explained/

https://medium.com/@joel_34096/k-means-clustering-for-image-classification-a648f28bdc47

https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20algorithm%2C%20also%20known%20as%20KNN%20or,of%20an%20individual%20data%20point