Custom Dataset Modeling via Convolutional Neural Networks

Hosted on Jupyter Notebooks

Collin Sieffert
Electrical and Computer Engineering Department
University of Florida
Gainesville, United States
csieffert@ufl.edu

***Abstract*—This document contains the dataset preprocessing, analysis, and workup of models designed to a; classify flower types from input images, and b; detect cars within images and predict bounding boxes. The models used employ convolutional neural network technologies for extracting features from images.**

***Index Terms*—Regression, Classification, Deep Learning, Convolutional Neural Network, Transfer Learning, TensorFlow, Keras, Image Feature Extraction**

I. Introduction

This document will cover data exploration and preprocessing choices, hyper parameter tuning and model evaluation. Two separate datasets are used in this paper, and as such two separate models are created with different purposes.

II. Data Exploration

A. Dataset Descriptions

Here we'll begin with a quick breakdown of the dataset being used for classification.

Essentially a fractional subset of the ImageNet dataset, the flower dataset contains numerous RGB images different flowers of falling into 10 categories. However, unlike MNIST, none of these are numbers, instead they are: Roses, Magnolias, Lilies, Sunflowers, Orchids, Marigold, Hibiscus, Firebush, Pentas, and Bougainvillea. All of these flowers were captured via a camera and stored as RGB .jpg images and then resized to be 300 by 300 pixel images. The training set contains 1658 total examples of the floewrs, with just under 160 examples of each character (after the testing sample was removed). The test set contains another 215 examples that won't be see during training. See an example from each class.

Fig. 1. Class 0, Rose

Fig. 2. Class 1, Magnolias

Fig. 3. Class 2, Lilies

Fig. 4. Class 3, Sunflowers

Fig. 5. Class 4, Orchids

Fig. 6. Class 5, Marigold

B. Preprocessing Selections

As seen in the flower dataset examples, all ove the RGB images containg flower examples with varying visual features. The only hindrance I ran into when training this model was the size of the images as the training data, my compute units kept running out of memory and crashing. In order to rectify this situation I downsampled the images to be half of their original width and height, from 300x300 to 150x150. This maintained enough distinctive detail to decide between classes whilst not overloading the system RAM being used. For the car dataset, I actually didn't have enough training samples to flood the available RAM on the hiperGator. Due to this I was

Funded by Midnight Oil Co.

Fig. 7. Class 6, Hibiscus

Fig. 8. Class 7, Firebush

Fig. 9. Class 8, Pentas

Fig. 10. Class 9, Bougainvillea

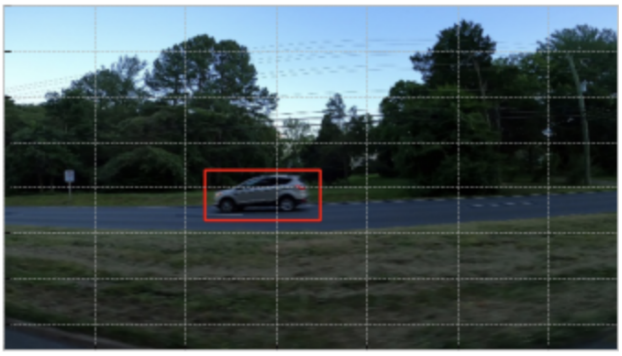

Fig. 11. Object Detection, Car Bounding Box

actually able to keep the images at their original 380x676 size. In addition, for both of the datasets I needed to normalize the pixel values to be on the range of 0 to 1. This required dividing the input data by 255 so that each pixel value scaled down properly. This scaling was crucial in improving the accuracy of the models being trained. In order to make the model for object detection more flexible any transformations applied to the input images is also applied to the training bounding boxes such that the boxes still accurately bound the car or cars in the given image.

III. Convolutional Neural Networks

In order to capture data local to the pixels we should analyze each pixel in conjunction with its neighbors. This brings about the biggest challenges with using linear/fully connected layers for computer vision as the lose all spatial data and flattening an RGB input image creates a massive input vector, resulting in too great a complexity of the model in terms of number of weights. Replacing the fully connected affine linear transform with another linear operation, convolution, solves all of these problems. The most effective tool, so far, we have for extracting features from images is the convolution layers used in convolutional neural networks. This brings about a number of advantages. First off, convolutions are very common, and in some cases, very simple operations that can be applied to images. One perticular example is the sobel edge detectors that can, well, detect edges by convolving with edge masks. Any photoshop sharpening/smoothing and other such operations function by applying convolutions to the images. As such, various convolutions have the ability to extract features and other information from images. This brings about the idea of the convolutional neural network, to learn the features of an image necessary for classifying said image, or in the case of dataset two, identifying the location of an object within an image. This order of operations differs from the conventional methods of finding features of importance, extracting said features, and then training a classifier on these features from the label data. This brings about another couple of features in which convolutional neural networks excel. One is being translation-invariant, i.e. we can detect a car no matter what region of the image it appears in, and also independent of the pixels far away from it, a bird in the sky or a pedestrian in the crosswalk shouldn't impact our detection of the car. Now that we've established, in very brief terms, the functionality of convolutional layers, we can discuss the other elements of the network that make it tick. Another very important component is the pooling method, whether it be max-pooling, global-average etc... pooling the image reduces the size, and then allows deeper layers to extract larger features from a reduced image. After said pooling and convolving, one now just needs to add a couple of dense/fully connected layers to combine all the information and give a probabilistic output to the specific class to which the image belongs. The object detection will instead output trained coordinates for the bounding box.

IV. Transfer Learning

Building on top of the base knowledge of convolution neural networks, due to the size of datasets required to train convolutional models of massive numbers of classes (imagenet for example), ways of sharing pre-trained knowledge greatly reduce the time needed to develop high-performance convolutional. This method is known as transfer learning. I employed this via Keras to bring in the deep trained layers of the Xception network, adding my own input reshaping and resizing layers, as well as my own custom output layers to properly classify the flowers/output the bounding box coordinates. This alone improved my model performances by about 20-30%.

V. Classification vs Object Detection

Just a simple note to define the distinction. Image classification determines if a picture belongs to any of a specific set of classes. I.e. this is a picture of a flower, specifically this species, and can assign a probability. Object detection on the other hand gives the location (or bounding box in simple networks) of a specific object (car in this example) within an image containing that object and potentially many other things.

VI. Object Detection Validation

As there wasn't a built in validation metric for the object detection, I developed my own metric to implement. The basic premise is that we are identifying a region of interest, specified by the labeled bounding box. The question we are looking to answer is that of how well the predicted bounding box detects the object in question. For that we calculate the shared area of the two boxes, and then take the union of both bounding boxes divided by the shared area, in the ideal case where the predicted and labeled boxes are the same, this metric will output a 1, however if the shared area is less than the ideal, then we get a value less than 1.

VII. Conclusion

All in all this has been an enjoyable dive into convolutional neural networks in machine learning. As we all know image classification and object detection are extremely important in today's age, just look at self-driving cars. As the demand for image porcessing/classification/detection/etc increases, the ability to share knowledge across networks will be vital. In temrs of this particular project, I would say that I struggled a bit on this project with computational resources and balancing the down-sampling from the original data. That and the mislabeled/confounding samples were hard to fight against, and as such the accuracy of my models suffered. Please view the accompanying notebooks for quantitative model evaluation metrics.