ROB 535 Group 14: Image Classification

Anthony Liang
University of Michigan

Julius Yeh
University of Michigan

Shih-Heng Yu
University of Michigan

Hui-Ching Chen
University of Michigan

Hsuan-Kai Chang
University of Michigan

Jang Wu
University of Michigan

Hsiao-Jou Lin
University of Michigan

1 Introduction

For this Kaggle challenge, we were given a training dataset of 7000+ images and tasked with creating a model to predict on a set of test images. The dataset contained a mixture of images from 23 different classes which were grouped under 3 distinct labels. In our approach to this challenge, we experimented with various deep learning models such as VGG16 and also YOLOv3. The second challenge was localization in which we needed to identify the pose of the vehicle in space given a training set of point clouds. For this task, we approached it using SSD detector to extract features and use these features to train a simple multilayer network. The following sections will provide more details regarding our techniques and their performance.

2 Methods and Training

Task 1

Our first iteration of a model was a pretrained VGG16 network of which we replace the last fully connected layer. VGG16 was originally pretrained on the ImageNet dataset which is a collection of millions of images coming from 1000 classes. The pretrained model already contains weights for all of its parameters and thus saves us time from having to train the model from scratch. This approach was not very robust because the object being classified in the original image are often times occluded or far in the distance. This presents an inherent issue as the network isnt able to train on objects that it is not in view or is too small relative to the rest of the image. This issue is further propagated by the fact that VGG16 requires the input image to be resized to 224 x 224. Our solution to this was to use the information provided by the 3D bounding boxes to project a 2D bounding box onto the image. Then we used the cropped and resized image as the input to the network. This simple change improved our validation accuracy by almost 10%.

Next, we tried to implement a different state of the art method for object detection, YOLOv3 [1]. Previously, we had to use the bounding box to extract the object for training. The bounding box information is not provided to us for testing. YOLO is appropriate for this task in that it can do both detection and classification. We first experimented with using YOLO preconfigured with weights and starter code. The predictions were based on the COCO dataset so it was only able to identify a few classes that overlapped with our dataset.

Then we experimented with training YOLO from scratch using the dataset given. We let it train for around 1000 iterations until the average loss was 0.8. It performed slightly better but it did not improve by a significant margin. Lastly, from taking a look at the training data, we noticed that it was skewed to class 1 and 2 and there werent many 0s. Thus instead of trying to predict 3 classes, we loosen the constraint and simplified the problem to a binary classifier. We trained an InceptionV3 model which is one of the state of the art image classification architectures. To put our own twist on it, we decided to try an ensemble model between Inception and Resnet. Ideally the ensemble model should help the performance because the two models are learning different feature representations of the data. This ensemble classifier was able to achieve 51% accuracy which is only slightly above the baseline.

Task 2

For the second task, we decided to use a feature based approach for our predictions. We used an open sourced model for SSD found online to perform detection over our input images to identify bounding boxes for the vehicles. The predicted class and bounding box coordinates will be used as features to our custom network to predict the centroid. Our network was a simple multilayer perception that receives the 10 dimensional feature vector (features from training bounding boxes 2D/3D) and outputs the three coordinate values. It has two fully connected layers to learn combinations for the feature vector and uses Xavier uniform for weight initialization. We used the RSME metric as the loss function for local net and trained the network for around 1000 epochs at which loss was 8.8. For prediction, we used the features from SSD predicted on the test images as input to the network. The final accuracy of the network is 17.04291.

3 Results

Method (Task 1)	**Performance (Accuracy)**
VGG16 Pretrained	49.26%
YOLOv3 Detector	49.37%
InceptionV3 + Resnet	51.05%
Method (Task 2)	**Performance (RMSE)**
SSD + LocalNet	17.04291

References

[1] Redmon, Joseph, Farhadi, Ali, "YOLOv3: An Incremental Improvement," *arXiv*, 2018.

Thank you to the EECS498 Staff for holding office hours and helping us with this project through Piazza!