Deep Learning for Computer Vision- Final Project

Image classification task

Submitted by: Chen Hazut

Chosen classes for classification:

- Bicycle
- Bus
- Motorcycle
- Pickup truck
- Traffic sign
- Streetcar
- Tractor
- Train
- Car

Datasets in use:

- **ImageNet:** Real world image dataset which contains over million images classified into 1000 classes.
  The use of the ImageNet dataset was done by using the MobileNet model which was originally trained on the ImageNet dataset.
- **CIFAR-100:** Consist of 100 classes, each class contains 600 32x32 colored images (500 for training and 100 more for validating).
  CIFAR-100 dataset was received from `keras.datasets` library as 2 tuples:
  - **x_train, x_test:**
    uint8 array of RGB image data with shape (num_samples, 3, 32, 32)
  - **y_train, y_test:**
    uint8 array of category labels (integers in range 0-9) with shape (num_samples**)**

  Those images were saved as .png, sorted in folders according to the corresponded class and divided to train and validation images. From all of 100 classes of this dataset, 7 relevant classes were chosen: Bicycle, Bus, Motorcycle, Pickup truck, Streetcar, Tractor and Train.
  The Car class was taken from CIFAR-10 by using the same process as the classes from CIFAR-100
- **German Traffic Sign Recognition Benchmark (GTSRB):** multi-class, single-image classification challenge held at the International Joint Conference on Neural Networks (IJCNN) 2011. Contains 43 classes of different traffic signs and more than 50,000 32x32 colored images in total**.**

the dataset was downloaded almost prepared for using. All the images were sorted in folders according to the correct class, and the images were in .png format.
Because this dataset is huge relatively to the number of images in class of the CIFAR dataset, only 10-14 images of each class of the GTSRB were taken as training data and also 100 random images were taken as validation data

Transfer Learning

Because the datasets in use are not that big, and the computational power available is relatively low for training a model from scratch, a transfer learning approach was taken. A MobileNet pre-trained model was chosen mainly because MobileNets are light weight networks which originally designed mobile and embedded based vision applications where there is lack of compute power. By being pre-trained on the ImageNet dataset, the MobileNet has learned a huge number of images and was able to get good filters to identify certain objects.

By removing the last layer of the model which is the classifier of the 1000 classes, and replacing it with the following layers, the filters that the MobileNet received after its pre-training were uses to help on the task of classifying the requested classes.

The new layers that were added to the model instead of its last layer are:

- Average pooling operation for spatial data, with window of shape (2,2) to reduce the data and to prepare the model for the final classification layer.
- Fully Connected layer with 512 neurons and a ReLU activation function.
- Fully Connected layer with 9 (number of classes) neurons and a Softmax activation function. This layer replaces the Fully Connected layer of the MobileNet which has been removed, and acts as the new classifier for the model.

The first 15 layers of the model were set to be non-trainable, so that the weights of the pre-trained model will stay frozen.

Also, an ImageDataGenerator was applied on the training and validation data in order to increase the size of the dataset by altering the images in random ways each time like: rotating the image in a random angle, shifting the image a bit, zooming in and horizontal flipping.

The code was written using Keras API because for me it was more intuitive then other libraries available like TensorFlow and PyTorch.

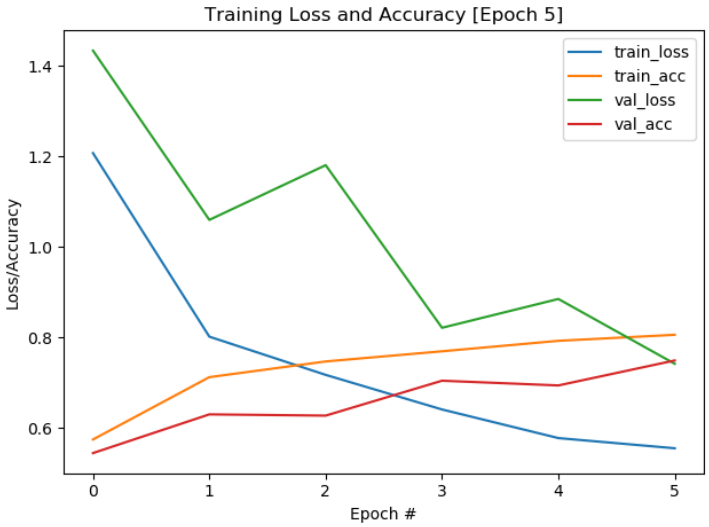
Experimenting

The final model was trained using a 32GB RAM and an intel core i9-9900K CPU for one hour (after 6 epochs).

During the quest to find the parameters for building the best model, many variations of the model were tested. For most of the tries, it was trial and error approach with

educated guess of the parameters that needed to be changed. The parameters that were altered during the process were the optimizer (SGD/Adam), the steps per epoch, the values for the ImageDataGenerator, the number of layers that are non-trainable and the new layers of the model (adding and removing the layers). The following description of the models are a sample of the process:

1. One of the models tested was using:
    - **Layers**:
        - MobileNet layers except for its last FC layer
        - Average pooling layer
        - Fully connected layer with 512 units and ReLU activation function
        - Fully connected layer with 9 units and Softmax activation function
    - **Frozen layers**: the first 15 layers of the MobileNet
    - **Optimizer**: Adam, with learning rate of 0.001
    - **Epochs**: 6. Each took about 10 minutes



It can be seen from the graph above that for the validation accuracy and loss there are ups and downs but the loss is generally heading toward down and the validation accuracy increasing along with the train accuracy.

Maybe if this model ran for longer time it would have given better results.

This model achieved:

Loss: 0.5568 accuracy: 0.8055

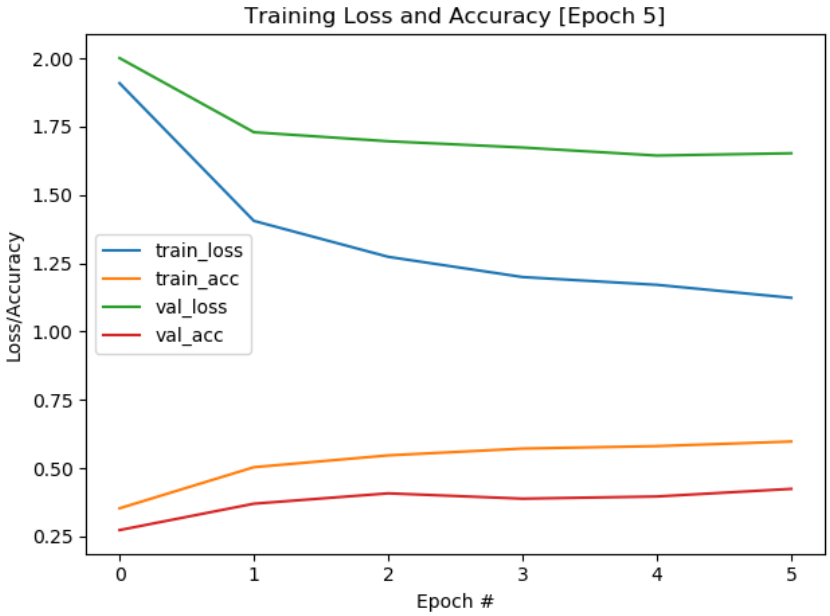
validation loss: 0.7419 validation accuracy: 0.7492

That is the best model I was able to achieve.

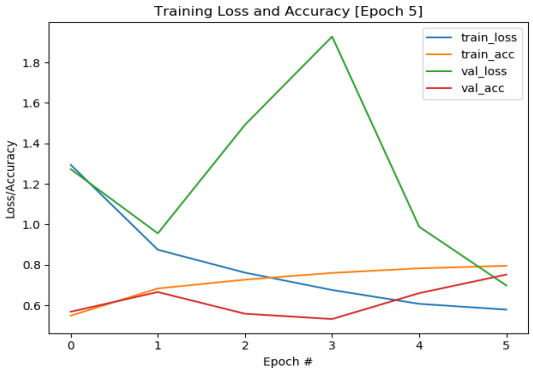
2. Another model training tries:
    - **Layers**:
        - MobileNet layers except for its last FC layer
        - Average pooling layer
        - Dropout of 0.3
        - Fully connected layer with 9 units and Softmax activation function

- **Frozen layers**: only 5 last layers were trained
- **Optimizer**: Adam, with learning rate of 0.001
- **Epochs**: 6. Each took about 5 minutes

In this model the accuracy never got above 60% and the validation accuracy reached its higher point on epoch 2. I guess that with more training this could improve a little bit, but not enough to be a proper model, due to the very low increasing rate it presented so far.



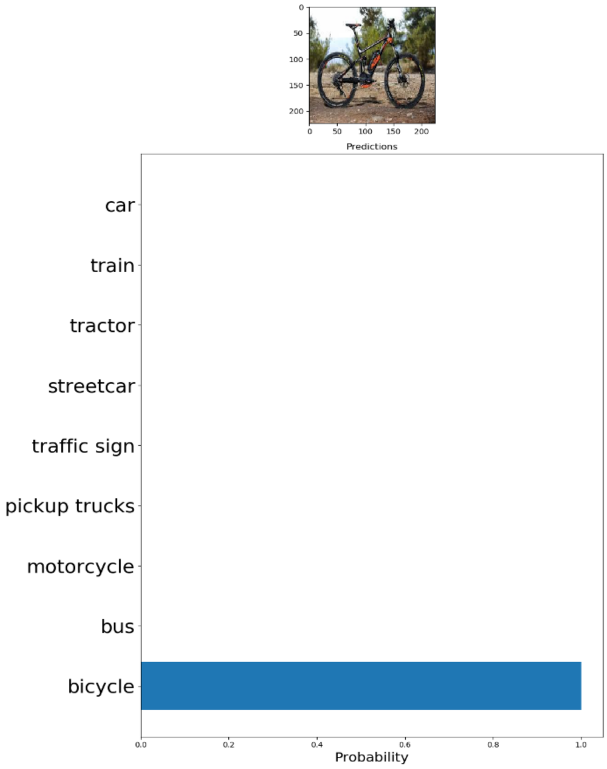
3. Another model is:
   - **Layers**:
     - MobileNet layers except for its last FC layer
     - Average pooling layer
     - Dropout of 0.2
     - Fully connected layer with 512 units and ReLU activation function
     - Dropout of 0.2
     - Fully connected layer with 9 units and Softmax activation function
   - **Frozen layers**: first 30 layers
   - **Optimizer**: Adam, with learning rate of 0.001
   - **Epochs**: 6. Each took about 8 minutes

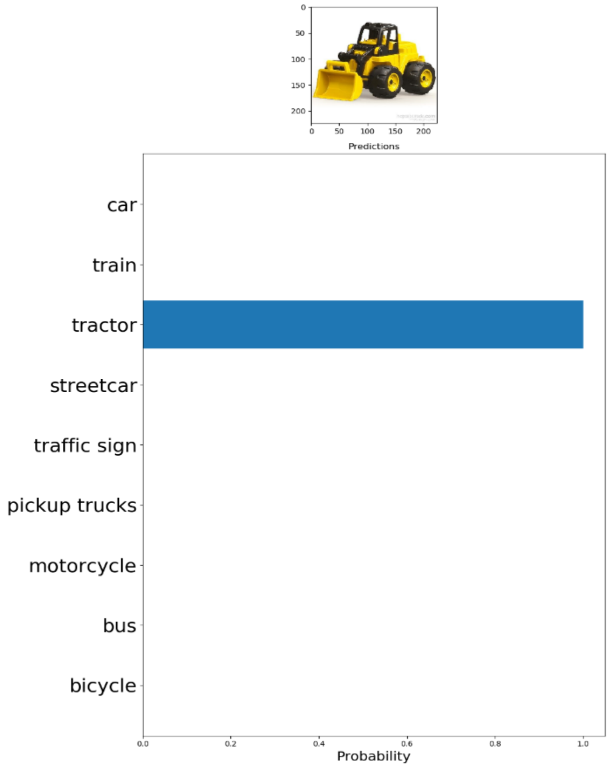


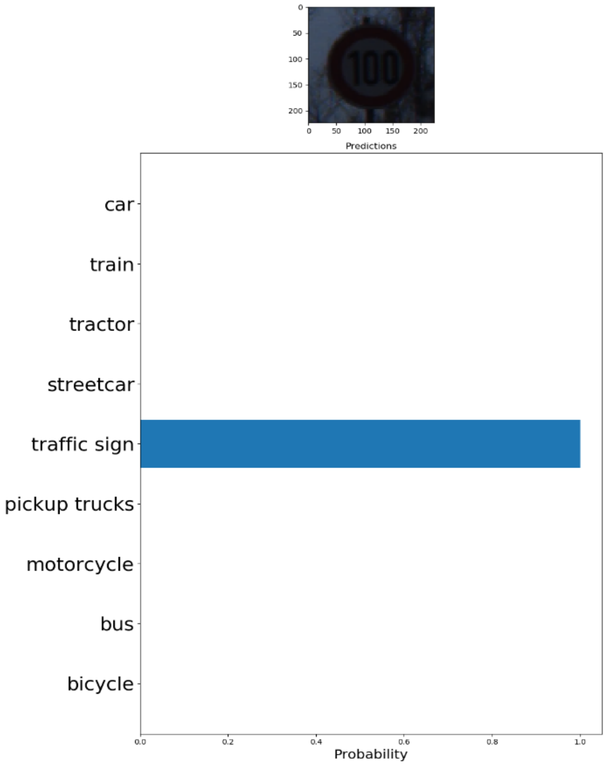
This training ended with 79.4% training accuracy and 75.1% validation accuracy. But even thought the accuracy is relatively high it gave poor prediction accuracy, as will be seen later.

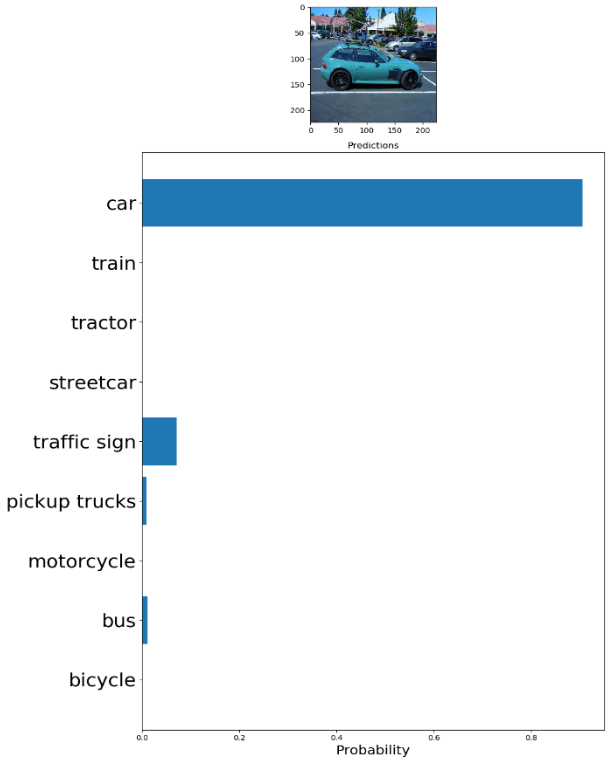
Results

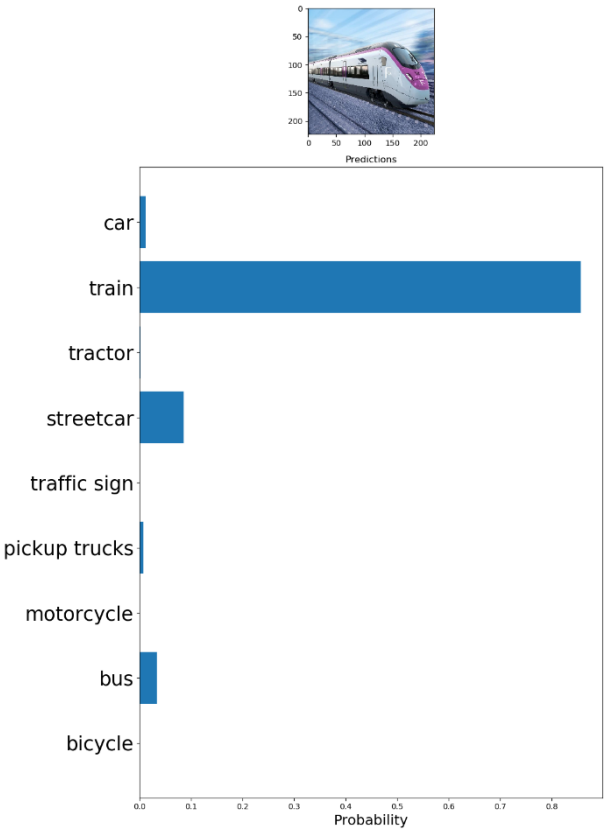
The best model received the following results:

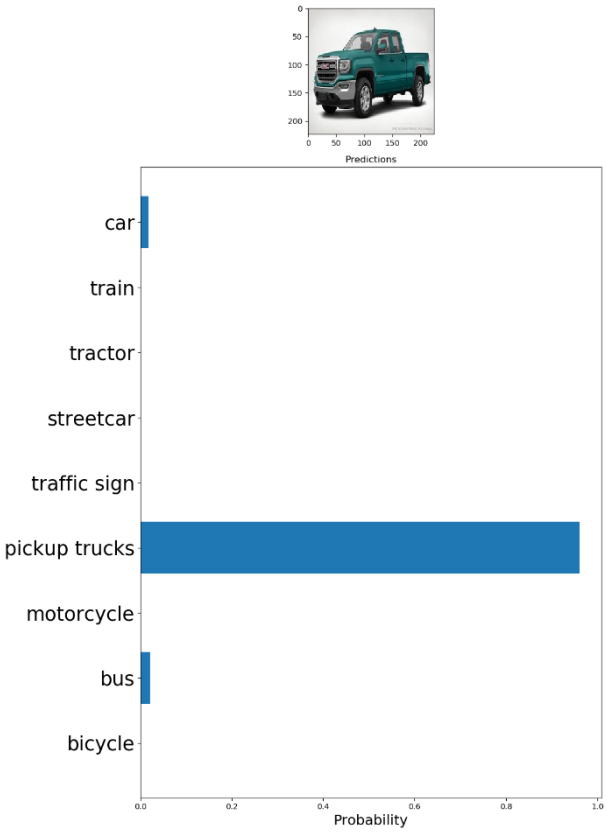


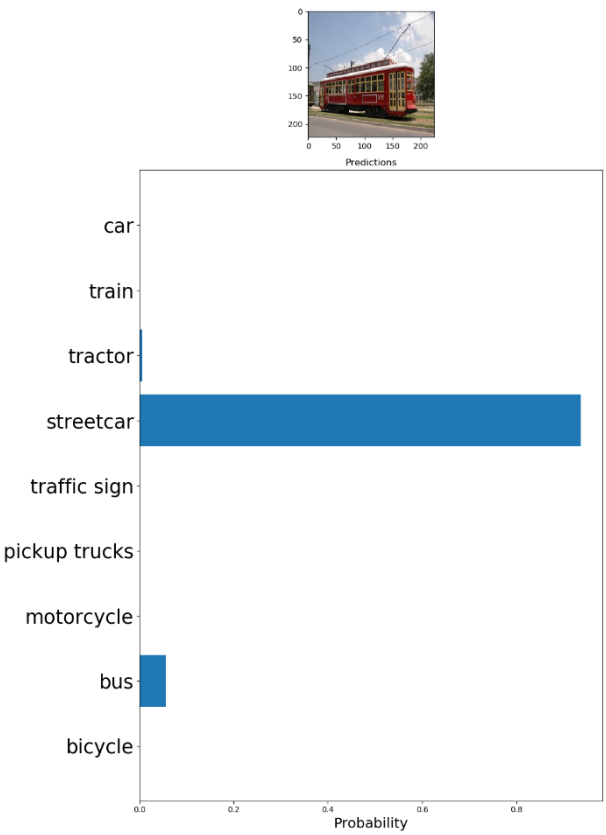


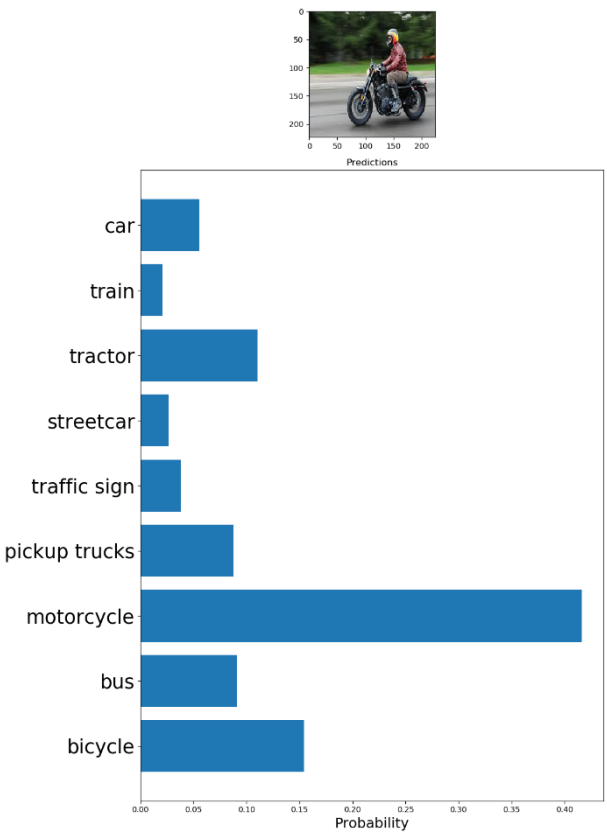





Predictions
car
train
tractor
streetcar
traffic sign
pickup trucks
motorcycle
bus
bicycle
Probability

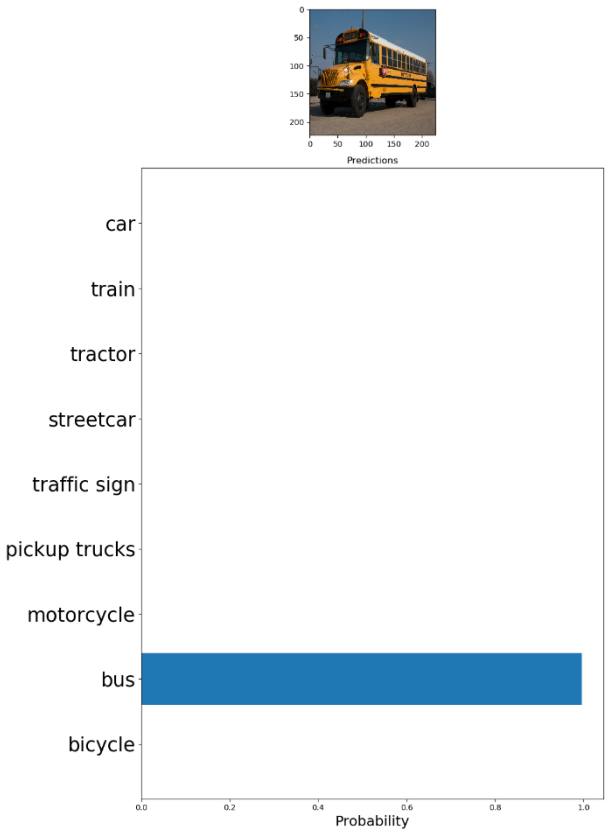

Predictions
car
train
tractor
streetcar
traffic sign
pickup trucks
motorcycle
bus
bicycle
Probability


Predictions
car
train
tractor
streetcar
traffic sign
pickup trucks
motorcycle
bus
bicycle
Probability

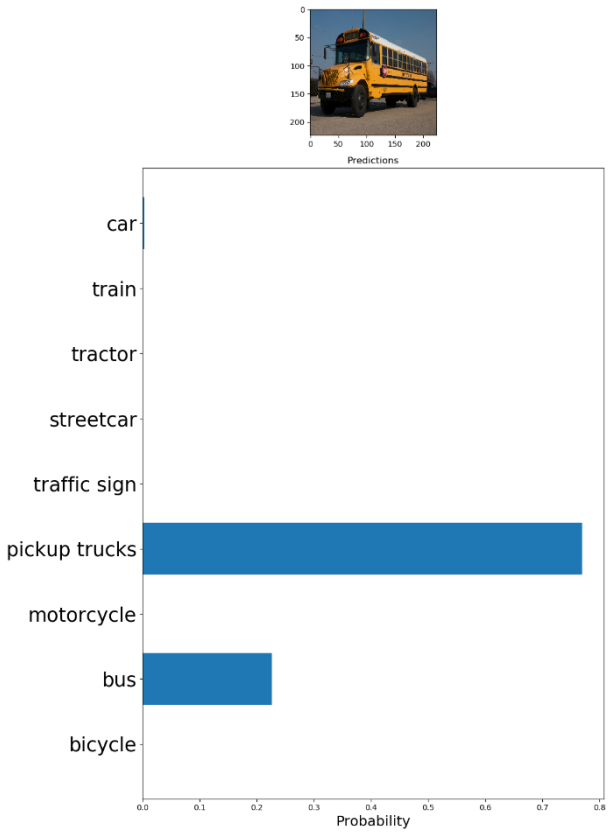

Predictions
car
train
tractor
streetcar
traffic sign
pickup trucks
motorcycle
bus
bicycle
Probability

For comparison, the following results were received by the first (the best model) and last (received 79% accuracy) model in the description above:

First model

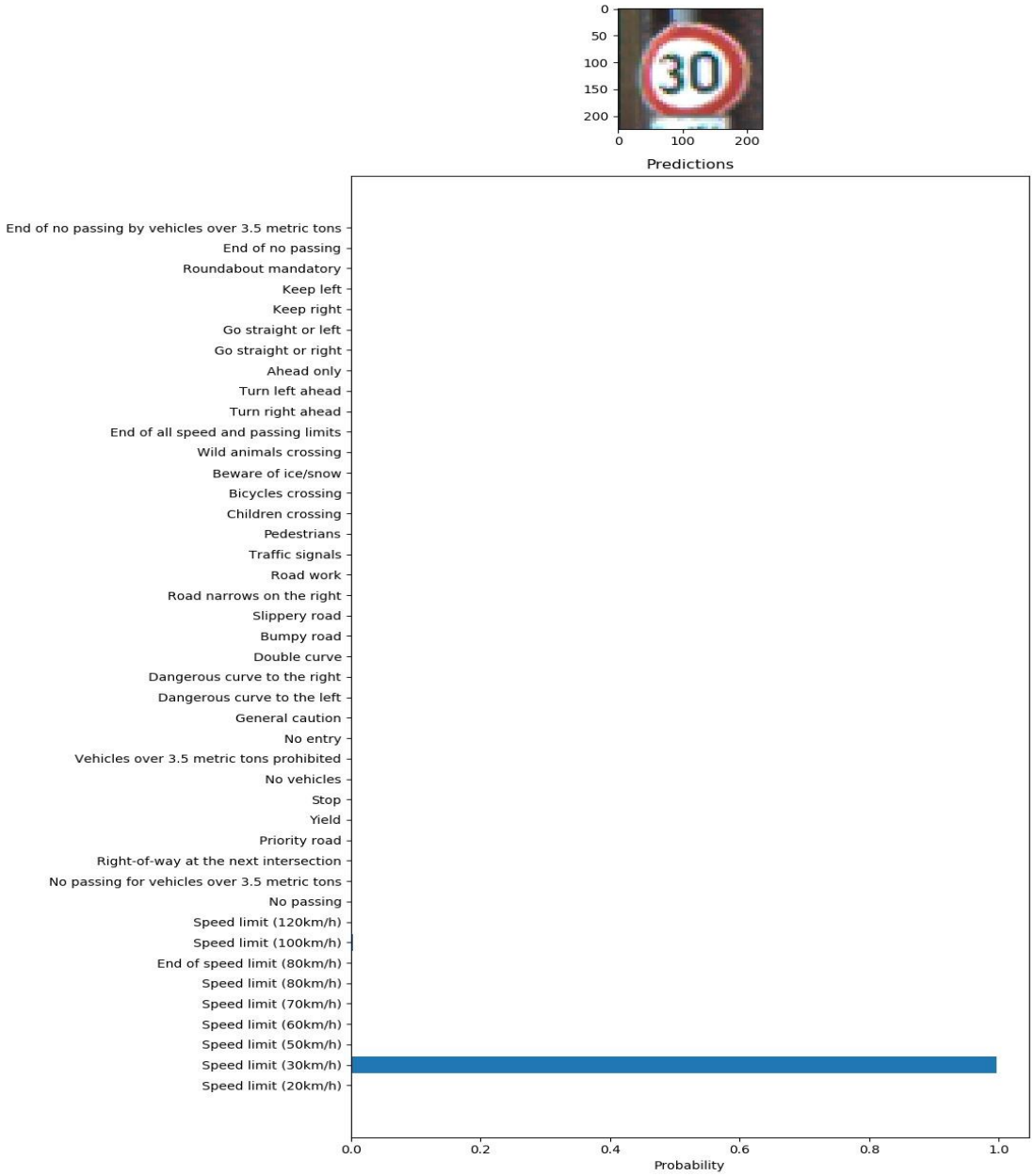


Third model

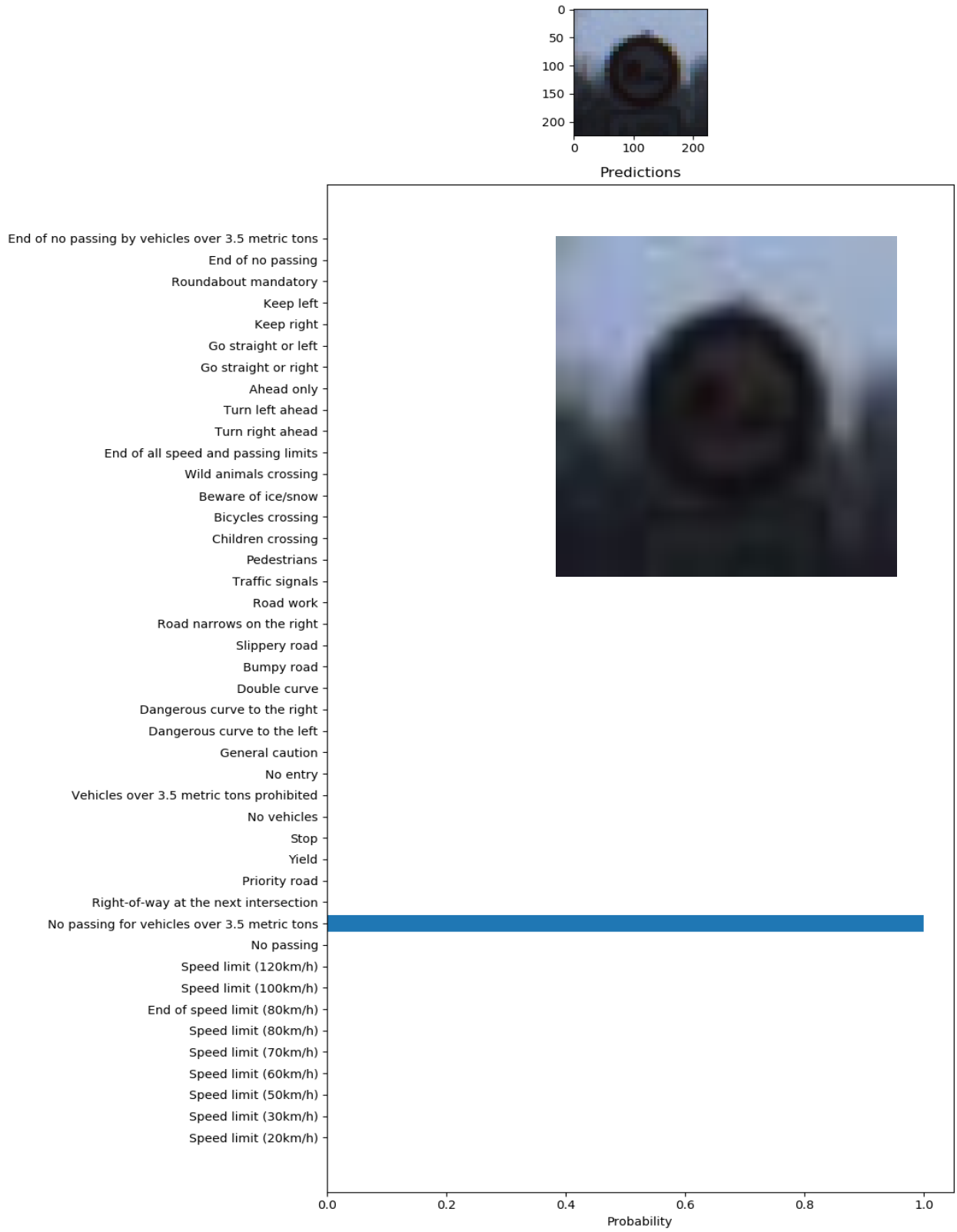


Traffic Signs Classification

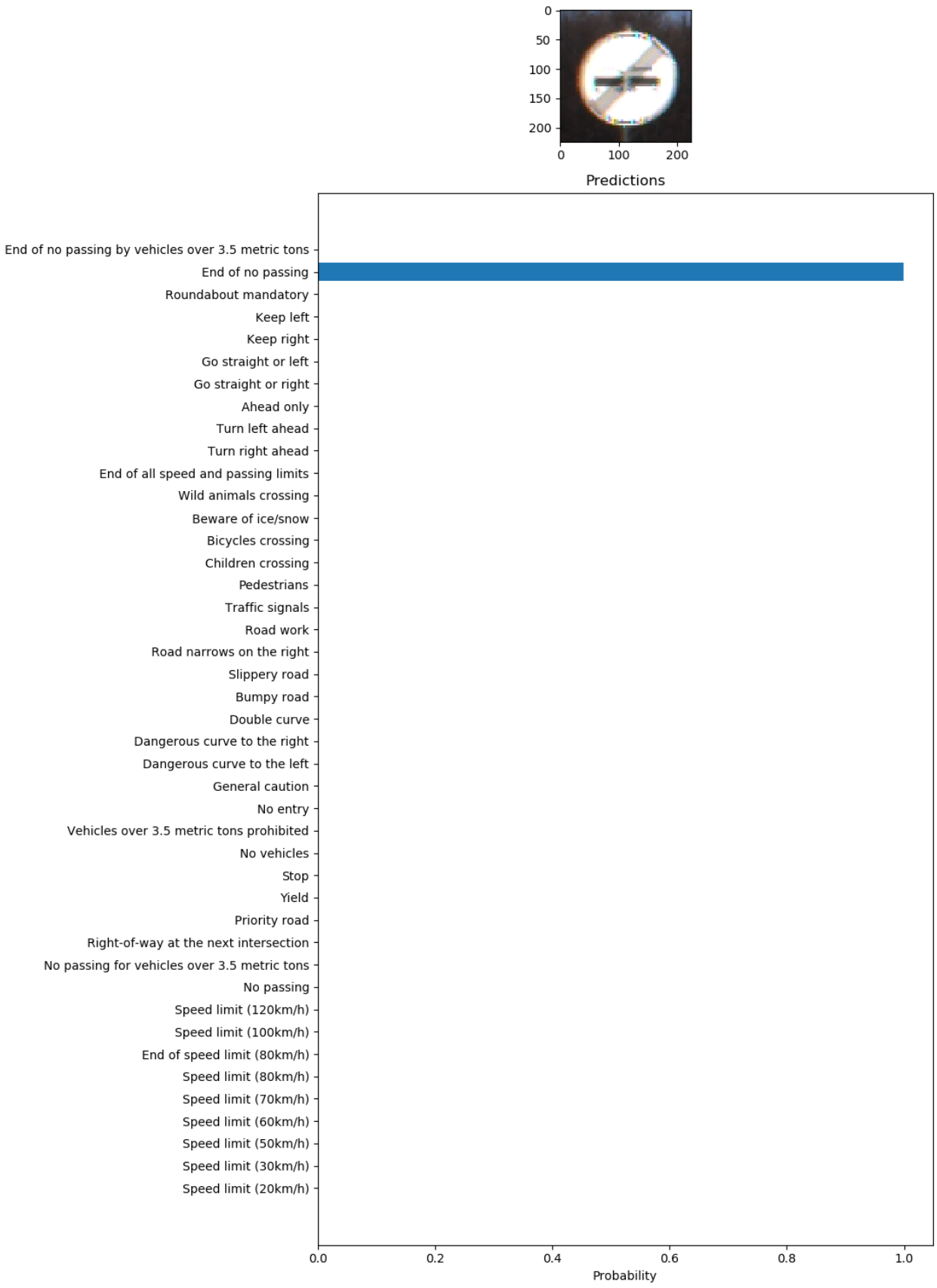
Before I understood the requirements for this assignment, I wanted to do classification of traffic signs and after working on that task, and achieving the tools that needed to be used for doing this work, the task was completed and a classifier for traffic signs was build. Because it was a real effort and because of my excitement about recognizing which sign is in the image, the results of the prediction of this classifier are presented here:



It can be seen from this prediction that the model can identify the sign even when it dark and not clear.



Classification of overly bright images is also done perfectly



The model does have a weak spot. Due to the augmentation done on the dataset, where I set it to perform a horizontal flip, and because of that flip, part of the images in the class of the signs turn left has become tern right and vice versa.