Classifier Training and Findings

December 2019

1 Introduction

After extracting the road sign images as bounding boxes we are required to classify them as one of 43 classes that are available so far.
For this purpose, making use of a **Convolutional Neural Network** architecture seems optimal.

2 Methodology

2.1 Template Architecture (1)

The template architecture consisted of a stack of *Conv2D* and *MaxPooling2D* layers.
As input, a CNN takes tensors of shape (image_height, image_width, color_channels), with the default batch size of 32.

2.2 Improvement and Final Architecture

Because the overall accuracy was not too good, and the results while testing was not satisfactory on our data set, we decide to modify the template architecture and For this architecture, the MaxPooling layers have been omitted from the convolutional base and the number of filters being used per layer increases by a factor of 2 as we go from layer to layer.
The dense layer is made so that the number of outputs decrease to the number of classes we have. In this case, the final dense layer has a softmax activation layer with 43 outputs.
The model summary is as follows:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 16)        448
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 32)        4640
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 26, 26, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 24, 24, 128)       73856
=================================================================
Total params: 97,440
Trainable params: 97,440
Non-trainable params: 0
_________________________________________________________________
```

Figure 1: Convolutional Base

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 16)	448
conv2d_1 (Conv2D)	(None, 28, 28, 32)	4640
conv2d_2 (Conv2D)	(None, 26, 26, 64)	18496
conv2d_3 (Conv2D)	(None, 24, 24, 128)	73856
flatten (Flatten)	(None, 73728)	0
dense (Dense)	(None, 1024)	75498496
dense_1 (Dense)	(None, 256)	262400
dense_2 (Dense)	(None, 43)	11051

Total params: 75,869,387
Trainable params: 75,869,387
Non-trainable params: 0

Figure 2: With Dense Layers

3 Results

3.1 Classifier Training

When evaluated, the model gives and **accuracy** of **0.9825** and:
loss of **0.1513**.

The accuracy plot is as shown below:

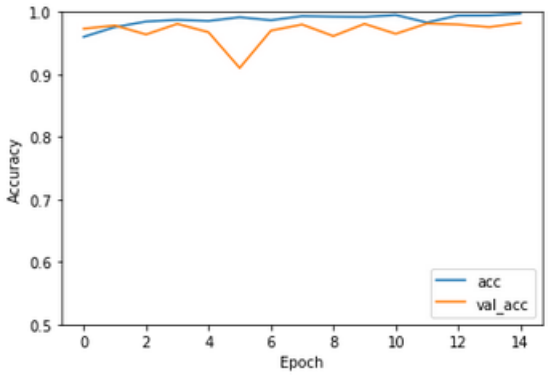

Figure 3: Plot of Accuracy vs Epoch

The implementation on our detected images is as shown in the next section of this document.

3.2 Testing on collected Pictures from DashCam using the *trained model*

Given the high accuracy of the model, we implement the model on a set of images obtained from the DashCam that have been detected using another neural network implementation. The result obtained by implementing the trained model is as shown below: (please see attached files for the code and model, 'classifier_model_new12000.h5')

Figure 4: Testing Results on the trained classifier model

By inspection, we obtain an accuracy of around 6/21 which is around 28.57%.

3.3 Testing on collected Pictures from DashCam using a *pretrained model*

Given the accuracy obtained above is not desirable, we also implemented and ran a pretrained classifier model obtained from (2) and the following results were obtained:

Figure 5: Testing Results on the pretrained classifier model (model.h5 and model.json)

By inspection, we obtain an accuracy of around 9/21 which is around 42.85%. We observe that this outcome than that given by the convolutional neural network described above.

4 Discussion and Conclusion

1. We observe the trained model is giving probability of 1.0 for the classified images. The reason is still unknown, but initial guess and debugging suggest the issue might lie in our dense layers (3).
2. Comparing the outcomes we observe when running our detected images on both the trained and pretrained models, we observe a slightly better performance from the pretrained model. However, an accuracy of 42.85% is still not sufficient and this hints at using probably a different network or getting better images at detection, and also using a larger set of training and validation images.
3. Also, we are testing the models on only 21 detected images out of 116. This is because the 43 classes we have access to are outdated and hence, a more up to date set of classes would certainly allow us to test on a larger set. This would improve the confidence we have on the model outcomes.

5 Appendix

```
Train on 12000 samples, validate on 1939 samples
Epoch 1/15
12000/12000 [==============================] - 1303s 109ms/sample - loss: 0.1429 - acc: 0.9601 - val_loss: 0.1198
- val_acc: 0.9732
Epoch 2/15
12000/12000 [==============================] - 1376s 115ms/sample - loss: 0.0901 - acc: 0.9758 - val_loss: 0.1001
- val_acc: 0.9783
Epoch 3/15
12000/12000 [==============================] - 1329s 111ms/sample - loss: 0.0581 - acc: 0.9846 - val_loss: 0.1882
- val_acc: 0.9639
Epoch 4/15
12000/12000 [==============================] - 1272s 106ms/sample - loss: 0.0523 - acc: 0.9872 - val_loss: 0.0825
- val_acc: 0.9809
Epoch 5/15
12000/12000 [==============================] - 1261s 105ms/sample - loss: 0.0614 - acc: 0.9855 - val_loss: 0.1693
- val_acc: 0.9675
Epoch 6/15
12000/12000 [==============================] - 1271s 106ms/sample - loss: 0.0362 - acc: 0.9915 - val_loss: 0.4864
- val_acc: 0.9103
Epoch 7/15
12000/12000 [==============================] - 1273s 106ms/sample - loss: 0.0541 - acc: 0.9867 - val_loss: 0.1905
- val_acc: 0.9701
Epoch 8/15
12000/12000 [==============================] - 1251s 104ms/sample - loss: 0.0330 - acc: 0.9933 - val_loss: 0.1258
- val_acc: 0.9799
Epoch 9/15
12000/12000 [==============================] - 1244s 104ms/sample - loss: 0.0380 - acc: 0.9924 - val_loss: 0.2474
- val_acc: 0.9613
Epoch 10/15
12000/12000 [==============================] - 1239s 103ms/sample - loss: 0.0374 - acc: 0.9920 - val_loss: 0.1224
- val_acc: 0.9809
Epoch 11/15
12000/12000 [==============================] - 1238s 103ms/sample - loss: 0.0389 - acc: 0.9952 - val_loss: 0.2946
- val_acc: 0.9649
Epoch 12/15
12000/12000 [==============================] - 1240s 103ms/sample - loss: 0.1285 - acc: 0.9830 - val_loss: 0.1668
- val_acc: 0.9814
Epoch 13/15
12000/12000 [==============================] - 1242s 103ms/sample - loss: 0.0400 - acc: 0.9943 - val_loss: 0.1508
- val_acc: 0.9799
Epoch 14/15
12000/12000 [==============================] - 1242s 103ms/sample - loss: 0.0353 - acc: 0.9945 - val_loss: 0.1528
- val_acc: 0.9758
Epoch 15/15
12000/12000 [==============================] - 1241s 103ms/sample - loss: 0.0132 - acc: 0.9972 - val_loss: 0.1513
- val_acc: 0.9825
```

Figure 6: Training summary

References

[1] "Convolutional neural network (cnn)." [Online]. Available: https://www.tensorflow.org/tutorials/images/cnn

[2] "Traffic sign classifier." [Online]. Available: https://github.com/ulmefors/Traffic-Sign-Classifier

[3] "Stackoverflow." [Online]. Available: https://stackoverflow.com/questions/50179709/tensorflow-softmax-returning-only-0-and-1