Orbuculum - Assignment

Submitted by - Vaibhav Sahu

March 13, 2021

1 Task - 1

1.1 Loading Image Data

We split the dataset class-wise, in the ratio 60:20:20. To avoid loading all the samples at once, we add the file path to the images and their labels to a pandas dataframe. Then we load the images using the Keras' ImageDataGenerator method - flow_from_dataframe

To have consistent image dimensions - we resize all images to 30x30x3 using Keras preprocessing.

1.2 Making CNN

The following is the model summary -

Model: "CNN_model"

Layer (type) – Output Shape – Param
input_5 (InputLayer) – [(None, 30, 30, 3)] – 0
conv2d_8 (Conv2D) – (None, 30, 30, 32) – 2432
dropout_8 (Dropout) – (None, 30, 30, 32) – 0
batch_normalization_8 (Batch – (None, 30, 30, 32) – 128
activation_8 (Activation) – (None, 30, 30, 32) – 0
max_pooling2d_8 (MaxPooling2 – (None, 14, 14, 32) – 0
conv2d_9 (Conv2D) – (None, 14, 14, 64) – 18496
dropout_9 (Dropout) – (None, 14, 14, 64) – 0

batch_normalization_9 (Batch – (None, 14, 14, 64) – 256

activation_9 (Activation) – (None, 14, 14, 64) – 0

max_pooling2d_9 (MaxPooling2 – (None, 6, 6, 64) – 0

flatten_4 (Flatten) – (None, 2304) – 0

dense_8 (Dense) – (None, 128) – 295040

dense_9 (Dense) – (None, 43) – 5547

Total params: 321,899
Trainable params: 321,707
Non-trainable params: 192

This is a fairly simple CNN with convolutional layers being followed by Dropout for regularisation and Batch Normalisation for robustness.

1.3 Results

The CNN was first trained for 20 epochs. The loss and accuracy curves are given below :-

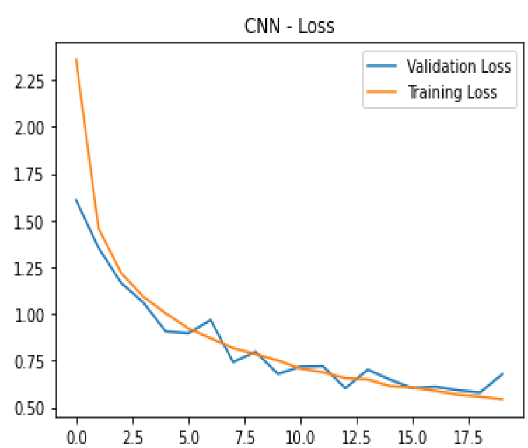

Figure 1: Loss during training for 20 epochs

The model was also trained for 50 epochs with an early-stopping callback with patience parameter set to 5 epochs. This model reached an accuracy of 0.8593 on the test set.

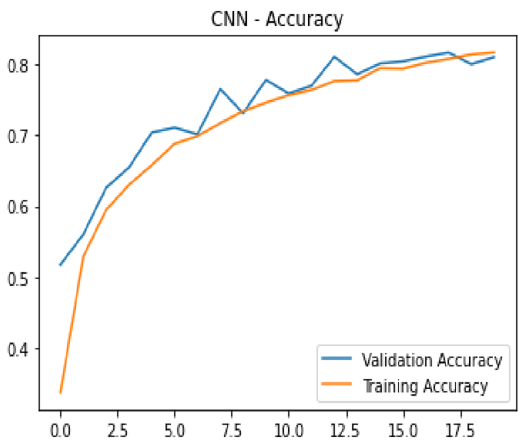

Figure 2: Accuracy during training for 20 epochs

2 Task - 2

2.1 Squeeze and Excitation Networks

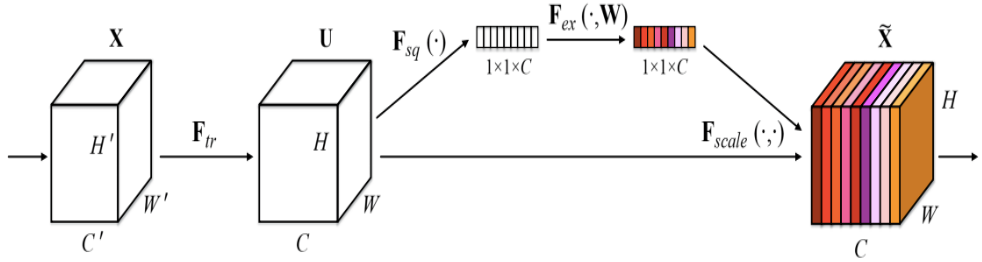

Figure 3: A Squeeze and Excitation Layer

The new model implements multiple squeeze and excitation layers. Here is the summary:-

Model: "SE_Net"

Layer (type) Output Shape Param Connected to

---

input_3 (InputLayer) [(None, 30, 30, 3)] 0

---

conv2d_4 (Conv2D) (None, 30, 30, 32) 2432 input_3[0][0]

---

dropout_6 (Dropout)	(None, 30, 30, 32)	0	conv2d_4[0][0]
batch_normalization_4	(None, 30, 30, 32)	128	dropout_6[0][0]
activation_4 (Activation)	(None, 30, 30, 32)	0	batch_normalization_4[0][0]
global_average_pooling2d_2 (Glo	(None, 32)	0	activation_4[0][0]
dense_8 (Dense)	(None, 2)	66	global_average_pooling2d_2[0][0]
dense_9 (Dense)	(None, 32)	96	dense_8[0][0]
multiply_2 (Multiply)	(None, 30, 30, 32)	0	activation_4[0][0], dense_9[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 14, 14, 32)	0	multiply_2[0][0]
conv2d_5 (Conv2D)	(None, 14, 14, 64)	18496	max_pooling2d_4[0][0]
dropout_7 (Dropout)	(None, 14, 14, 64)	0	conv2d_5[0][0]
batch_normalization_5 (BatchNor	(None, 14, 14, 64)	256	dropout_7[0][0]
activation_5 (Activation)	(None, 14, 14, 64)	0	batch_normalization_5[0][0]
global_average_pooling2d_3 (Glo	(None, 64)	0	activation_5[0][0]
dense_10 (Dense)	(None, 2)	130	global_average_pooling2d_3[0][0]
dense_11 (Dense)	(None, 64)	192	dense_10[0][0]
multiply_3 (Multiply)	(None, 14, 14, 64)	0	activation_5[0][0], dense_11[0][0]
max_pooling2d_5 (MaxPooling2D)	(None, 6, 6, 64)	0	multiply_3[0][0]
flatten_2 (Flatten)	(None, 2304)	0	max_pooling2d_5[0][0]
dense_12 (Dense)	(None, 128)	295040	flatten_2[0][0]
dropout_8 (Dropout)	(None, 128)	0	dense_12[0][0]
dense_13 (Dense)	(None, 43)	5547	dropout_8[0][0]

Total params: 322,383
Trainable params: 322,191
Non-trainable params: 192

As we can see two Squeeze and Excitation blocks were added right before the two maxpooling layers. We will soon see that this boosts the performance of the model.

2.1.1 Results

Training the model for epochs as we did for the CNN earlier, we get slightly superior results. The loss and accuracy for training are as give -

Figure 4: Loss during training for 20 epochs

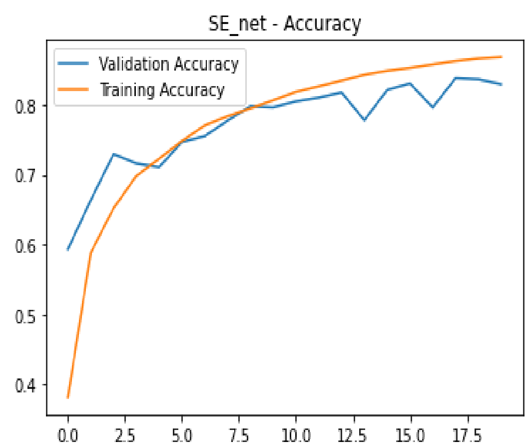

Figure 5: Accuracy during training for 20 epochs

For these 20 epochs of training on the same dataset the following graph shows how the SE layers boost the performance.

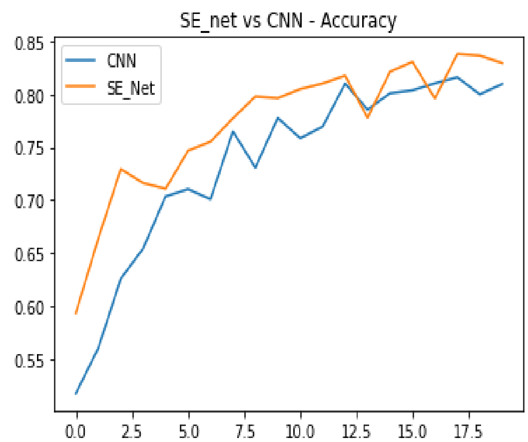

Figure 6: Comparison of accuracy with and without SE layers

Finally, we train this model for 50 epochs with an early stopping callback with patience of 5 epochs and it manages an accuracy of 0.8666 on the test set.

2.2 Analysing other metrics - Precision and Recall

The global metrics are not compatible with batches as is done in keras. Therefore, to better analyse the test set we create a single batch of test samples - "test_unibatch". We analyse this test set with our model hacing Squeeze and Excitation Layers. The results as provided by scikit-learn's classification report are as given -

What we can clearly see is that the model lacks in performance mostly in classes with less support. Hence, a more balanced dataset i.e. more data for classes which had lesser data, would surely result in improvement of accuracy.

	precision	recall	f1-score	support
0	0.90	0.21	0.35	42
1	0.85	0.92	0.89	444
2	0.87	0.70	0.77	450
3	0.60	0.88	0.71	282
4	0.92	0.99	0.95	396
5	0.62	0.59	0.60	372
6	0.76	0.98	0.85	84
7	0.93	0.95	0.94	288
8	0.95	0.74	0.83	282
9	0.95	0.98	0.96	294
10	0.99	0.99	0.99	402
11	0.98	0.90	0.94	264
12	1.00	0.95	0.97	420
13	0.98	0.99	0.99	432
14	1.00	0.90	0.95	156
15	1.00	0.94	0.97	126
16	1.00	0.99	0.99	84
17	1.00	1.00	1.00	222
18	0.97	0.98	0.97	240
19	0.40	0.50	0.44	42
20	0.55	0.44	0.49	72
21	0.76	0.58	0.66	66
22	0.97	1.00	0.99	78
23	0.74	0.49	0.59	102
24	0.98	0.96	0.97	54
25	0.88	1.00	0.94	300
26	0.96	0.98	0.97	120
27	1.00	0.94	0.97	48
28	0.76	0.84	0.80	108
29	0.85	0.63	0.72	54
30	0.72	0.86	0.78	90
31	0.87	0.96	0.91	156
32	0.79	1.00	0.88	48
33	0.66	0.76	0.70	138
34	0.47	0.32	0.38	84
35	0.98	1.00	0.99	240
36	0.73	0.77	0.75	78
37	0.45	0.36	0.40	42
38	0.86	0.88	0.87	414
39	0.40	0.10	0.16	60
40	0.63	1.00	0.77	72
41	0.94	0.67	0.78	48
42	0.72	0.75	0.73	48
accuracy			0.87	7842
macro avg	0.82	0.80	0.80	7842
weighted avg	0.87	0.87	0.86	7842

Figure 7: Classification Report

3 Task - 3: Visualizing the Features

To visualize the layers we plot 16 channels of the layer outputs for each of the two convolutional layers in our models with and without the Squeeze and Excitation Layers.

The features extracted in the first few layers are easier to interpret. Both the models tend to focus inside the circular sign, in some of which we actually see more emphasis on the relevant text.

The layer outputs for the convolutional layers deeper in the network are tougher to interpret. We do expect less noise in the output of the network with SE layers but this cannot be interpreted from the images.

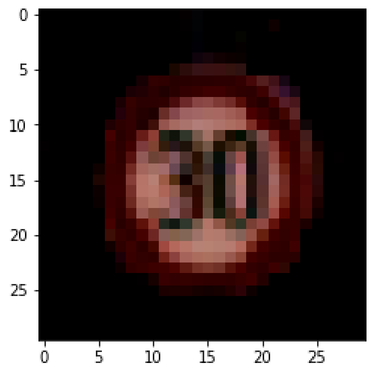

Figure 8: The Sample Image

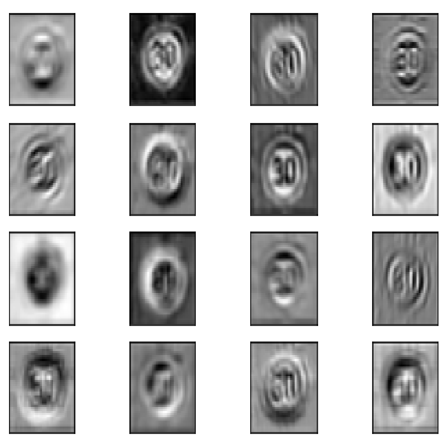

Figure 9: Output of 1st conv layer: CNN without SE

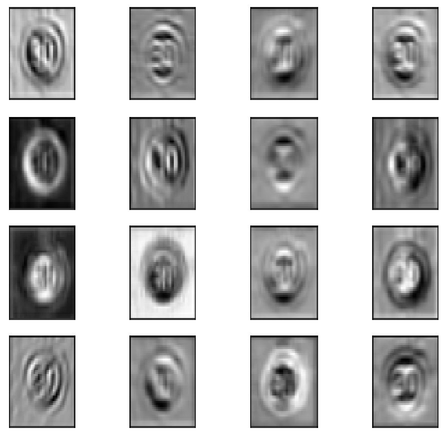

Figure 10: Output of 1st conv layer: CNN with SE

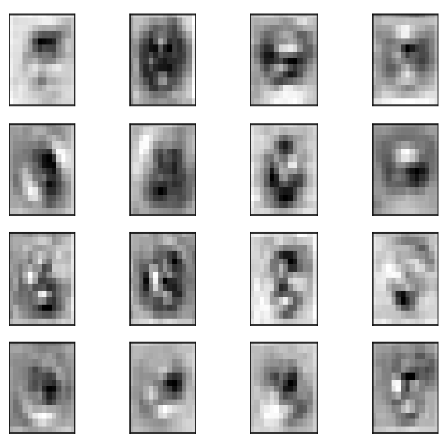

Figure 11: Output of 2nd conv layer: CNN without SE

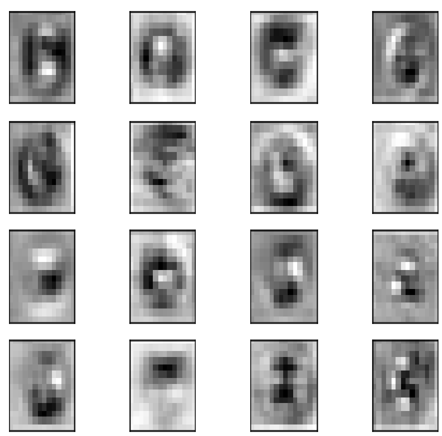

Figure 12: Output of 2nd conv layer: CNN with SE