MobileNetV2 as Sentence Classifier

Name: Chethan Singh Mysore Jagadeesh

Andrew Id: cmysorej

Introduction and Architecture details:

Sentence classifier presented for this assignment is based on below papers:

1. Convolutional Neural Networks for Sentence Classification by Yoon Kim
2. MobileNetV2: Inverted Residuals and Linear Bottleneck by Mark Sandler et. al.

Although MobileNetV2 is for image classification, I wanted to try out the same architecture to classify sentences.

Input Channel	Output Channel	Expansion Factor	Iterations (N)	strides(As mentioned in MobileNetV2 paper)
1	32	1	1	1
32	16	1	1	1
16	24	6	2	2
24	32	6	3	2
32	64	6	4	2
64	96	6	3	1
96	160	6	3	2
160	320	6	1	1
320	1280	1	1	1

Network Architecture: MobileNetV2 architecture uses a concept called **'BottleNeck Residual layer'**. This is special CNN layers followed by each of the CNN layer mentioned in above table. This 'BottleNeck Residual layer' takes the output channel of the CNN layer, expands its output channel further by the **'expansion factor'** mentioned above and again brings it back to the original output channel length before passing it to next layer. For example, in the output channel size of third layer if 24, bottleneck layer expands it by the expansion factor of 6 which is 144 using a CNN layer and again scales it back to 24 with another CNN layer before passing on to next CNN layer. This is inline with Yoon Kim's paper which mentions us to use varying layers of output channels. Along with this, each layer is repeated by **'Iterations(IN)'** times.

Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

Table 1: *Bottleneck residual block* transforming from k to k' channels, with stride s, and expansion factor t.

Along with this trimmed down version (shown below) of the original architecture was used but did not give optimal results. Results shared in Experiments section

Less complex MobilenetV2:
input_channels = [1, 32, 16, 24, 32, 64, 96]
output_channels = [32, 16, 24, 32, 64, 96, 160]
expansion_factor = [1, 1, 3, 3, 3, 3, 1]
iterations = [1, 1, 2, 3, 2, 3, 1]
strides = [1, 1, 2, 2, 2, 1, 1]

Activation: Also, each of the CNN layer(both normal and bottleneck) is followed by batchnorm2d layer. Batch Norm layer was followed by Tanh layers. But experiments were done with relu, leaky relu and tanh. Tanh was finally chosen because it gave optimal result.

Drop out: Final layer is fully connected Linear layer. Experiments were conducted with and without drop out layer between this fully connected linear layer. Finally drop out of 0.5 was chosen. Though it did not have great effect on the final result.

Prediction: For prediction Softmax layer is used followed by argmax to predict the label number.

Training:

Loss Function	Cross Entropy Loss
Optimizer	SGD with RMS prop of 0.9
Weight Initialization	Xavier
Learning Rate (Epoch 0-2)	0.003
Learning Rate (Epoch 3-4)	0.0003
Learning Rate (Epoch 5-10)	0.00003
Data Loader Batch size	64

Word to vectors:

Used Pretrained vector for Google news data from below link. It has ~2.8mn words. I used only top 600,000 words to build the word vectors for each sentence. In the below link google news word vector is in text format.

http://vectors.nlpl.eu/repository/

Experiments:

1. Without Drop out and RelU activation:

Epoch No.	Average Training Loss	Average Validation Loss	Accuracy
0	1.4118	0.9182	0.7061
1	0.9612	0.7286	0.7776
2	0.8767	0.6737	0.7869
3	0.7655	0.7725	0.7823
4	0.7321	0.6352	0.7916
5	0.7017	0.6743	0.7932
6	0.697	0.7275	0.7916

2. With drop out of 0.2 during training and ReLU:

Epoch No.	Average Training Loss	Average Validation Loss	Accuracy
0	1.4362	0.8866	0.7341
1	0.9787	0.7753	0.7745
2	0.8978	0.7467	0.776
3	0.7868	0.7555	0.7807
4	0.7545	0.8065	0.7807
5	0.7252	0.7867	0.7947
6	0.7204	0.7699	0.7823
7	0.7175	0.7286	0.7854

3. With Dropout of 0.5 during training and Leaky ReLU:

Epoch No.	Average Training Loss	Average Validation Loss	Accuracy
0	1.4151	0.8695	0.7434
1	0.9663	0.8345	0.7512
2	0.8873	0.8084	0.7574
3	0.7753	0.7791	0.7869
4	0.7412	0.6741	0.79
5	0.7111	0.6479	0.7916
6	0.7056	0.742	0.7823
7	0.7028	0.8293	0.7792

4. Less complex version of MobilenetV2, with Dropout of 0.5 during training and Tanh activation:

Epoch No.	Average Training Loss	Average Validation Loss	Accuracy
0	1.5661	1.3377	0.6267
1	1.245	1.1935	0.703

2	1.1585	1.1725	0.7201
3	1.0251	1.1335	0.7481
4	1.0003	1.1544	0.7543
5	0.9795	1.1492	0.7527
6	0.9758	1.1488	0.7589
7	0.9745	1.1482	0.7574

5. Full MobilenetV2 with Dropout of 0.5 during training and Tanh activation. (loss data is missing because I got excited and missed to save the data and only had accuracy to report)

Epoch No.	Accuracy
0	0.7642
1	0.7823
2	0.8016
3	0.8072
4	**0.8088**
5	0.8088
6	0.8072
7	0.8088

This result is used as final submission

Conclusions:

- MobileNetV2 architecture can be used for sentence classification with accuracy of 80.88% on validation data. Although it is primarily designed for image classification.
- It was interesting to see how leaky ReLU had no effect compared to ReLU and Tanh performed the best.
- Also, in above experimentations, it noteworthy that drop out layer in final layer of a complex architecture like MobileNetV2 had little to no effect in the outcome of the result.