Statistical Methods for Machine Learning

Prashant Bahuguna

January 2023

1 Introduction

The project uses pretrained Deep Learning neural networks in TensorFlow2 for the binary classification of cats and dogs. Matrices (accuracy) are obtained as an average of 5 fold cross validation. Models used:

- InceptionV3
- MobileNetV3Small
- EfficientNetV2B2
- VGG16

2 Neural Networks

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the brain. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning largely involves adjustments to the synaptic connections that exist between the neurons.

2.1 Architecture for Neural Networks

- Input Layer- As the name suggests, it accepts inputs in several different formats provided by the programmer.
- Hidden Layer- The hidden layer presents in-between input and output layers. It performs all the calculations
- Output Layer
  The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer. The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\sum_{i=1}^{n} Wi * Xi + b$$

It determines weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.

3 Models used

1. InceptionV3
Inception v3 is a convolutional neural network architecture from the Inception family that makes several improvements including using Label Smoothing, Factorized 7 x 7 convolutions, and the use of an auxiliary classifer to propagate label information lower down the network (along with the use of batch normalization for layers in the sidehead). The key building block is an Inception Module.

- Parameters - 24 Million

2. MobileNetV3Small
MobileNetV3 is a convolutional neural network that is designed for mobile phone CPUs. The network design includes the use of a hard swish activation and squeeze-and-excitation modules in the MBConv blocks.

- Parameters - 3 Million

3. EfficientNetV2B2
EfficientNetV2 is a type convolutional neural network that has faster training speed and better parameter efficiency than previous models. To develop these models, the authors use a combination of training-aware neural architecture search and scaling, to jointly optimize training speed. The models were searched from the search space enriched with new ops such as Fused-MBConv.

- Parameters - 10.2 Million

4. VGG16
VGG16 is a convolution neural net (CNN) which is considered to be one of the excellent vision model architecture till date. Most unique thing about VGG16 is that instead of having a large number of hyper-parameter they focused on having convolution layers of 3x3 filter with a stride 1 and always used same padding and maxpool layer of 2x2 filter of stride 2.The 16 in VGG16 refers to it has 16 layers that have weights.

- Parameters - 138 Million

4 CLI instructions

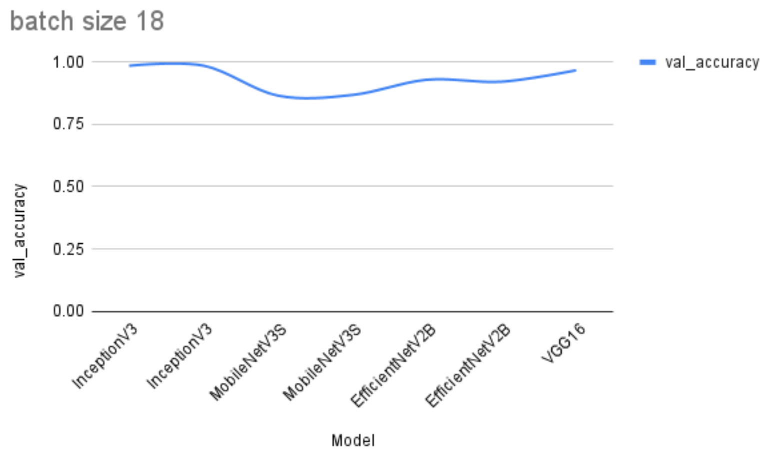
With the given training script, we can train list of models along with list of learning rates and batch sizes. The tensorboard logs and checkpoints (monitoring val loss) are stored automatically.

Batch sizes, Model architectures learning rates should be a list of int, string and float respectively

```
# Examplary CLI instructions
python train.py --imdir "path/to/the/imagedir" --lr [0.001, 0.0001]
--m ["MobileNetV3Small", "EfficientNetV2B2"] --bs [18, 24] --epoch 5

where

imdir = PATH TO IMAGEDIR. The folder contains images of both Cats and Dogs as separate directories.
However, there are 2 erroneous files(11702.jpb in Dogs and  666.jpg in Cats) and should be filtered

lr = Learning rate
m = Model architecture
bs = Batch size
epoch = No. of epochs to train
```

5 Metrics

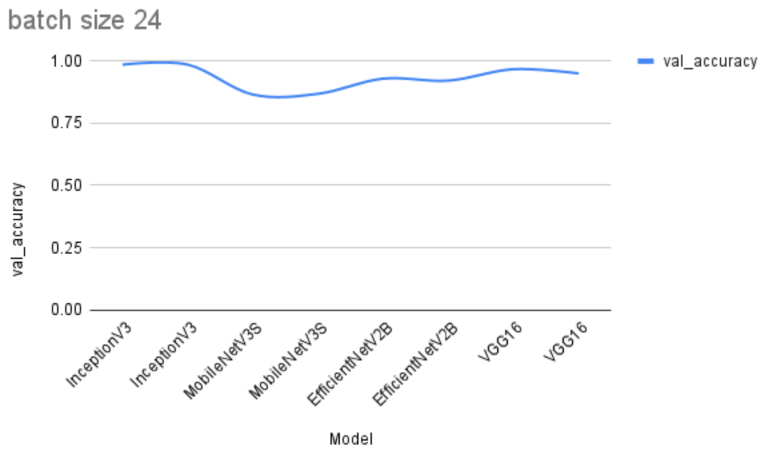
To save training time from all possible combinations, all the results below were computed for the learning rate of 0.0001. Also, they were trained just for 4 epochs with fixed image sizes of (256 x 256). As a rule of thumb, it is always preferable to use shallower networks unless absolutely necessary since bigger neural networks are harder to train.



Architecture	Batch$size$	Accuracy
InceptionV3	18	0.9859
MobileNetV3Small	18	0.8658
EfficientNetV2B2	18	0.9298
VGG16	18	0.96778

Along with them, with a batch size of 24, the metrics are -

Architecture	Batch$_{size}$	Accuracy
InceptionV3	24	0.9862
MobileNetV3Small	24	0.8687
EfficientNetV2B2	24	0.9219
VGG16	24	0.9508



The criterion for saving the checkpoints has been validation loss

5.1 Augmentations

I used the following augmentations:

- Random horizontal flip
- Re-scaling
- Random rotation up-to 10 degrees

While optical augmentations could also be used, I decided to limit to just the ones above.

6 Conclusion and Further developments

- Since the inter class variance is quite high(due to distance features of the binary classes), shallow models were able to train themselves with high accuracy.
- Current images were just (256, 256) to accommodate them in my GPU, however, higher image sizes (512, 512) etc will assuredly lead to better results.
- The model can be further improved with higher batch sizes, high number of epochs along with the inclusion of more robust augmentations and finally the addition of scheduler. Since, I trained it for just 4 epochs, scheduler was a bit unwanted, however, I would incorporate if I train the model for more epochs.