DermNet (Skin disease classification)

Best performance for training a new network (a VGG flavor) was 72% validation accuracy (experiment # 3)

Best performance of transfer learning (VGG19) after fine-tuning was 69% validation accuracy (experiment # 11)

Note: The validation accuracies are calculated in such a way that examples of all classes were given equal weight. It means that even if the accuracy of the class having maximum number of examples is high it cannot dominate the overall reported result. This fact was specifically verified by using the checkpoint from experiment # 11 and doing two different accuracy calculations on the validation data (**train_application_tune_predict.py**) See **Output-1** at the end of report.

1. Calculations with equal weights (all classes have same number of validation samples produced using duplication. The results was **69% validation accuracy** as reported in experiment # 11
2. Calculation with imbalance in validation samples not removed. The result was **80% validation accuracy**.

This point must be taken into consideration while evaluating validation accuracies for any imbalanced data.

Training Data:

Class imbalance:

There was a great degree of imbalance in the data examples provided for 7 different types of skin diseases. One class had as many as 1879 examples whereas other 6 classes had very few examples, not more than 116 in each case. The smallest number of examples were for class 5 with 56 examples.

Image sizes:

The image sizes varies from 100 to 5184 along width and from 100 to 3456 along height

Pre-processing:

Following procedure was adopted to tackle class imbalance problem:

1. Split each class in 80%/20% for training/validation sets [**setup_data_generic.py**]
2. For each class make the number of training and validation examples equal to the number of examples in the class with largest number of images. This will ensure that network will be trained with image of each class equal number of times. This will further ensure that validation accuracy will reflect the accuracy of each class with equal weightage. [**setup_data_balanced.py**]
3. Used data augmentation while presenting images to the network for training
4. Images were resized to same size for training and validation

Following are the results of various experiments

Part1: Training from scratch [**train_pyimagesearch.py, graph-1**]

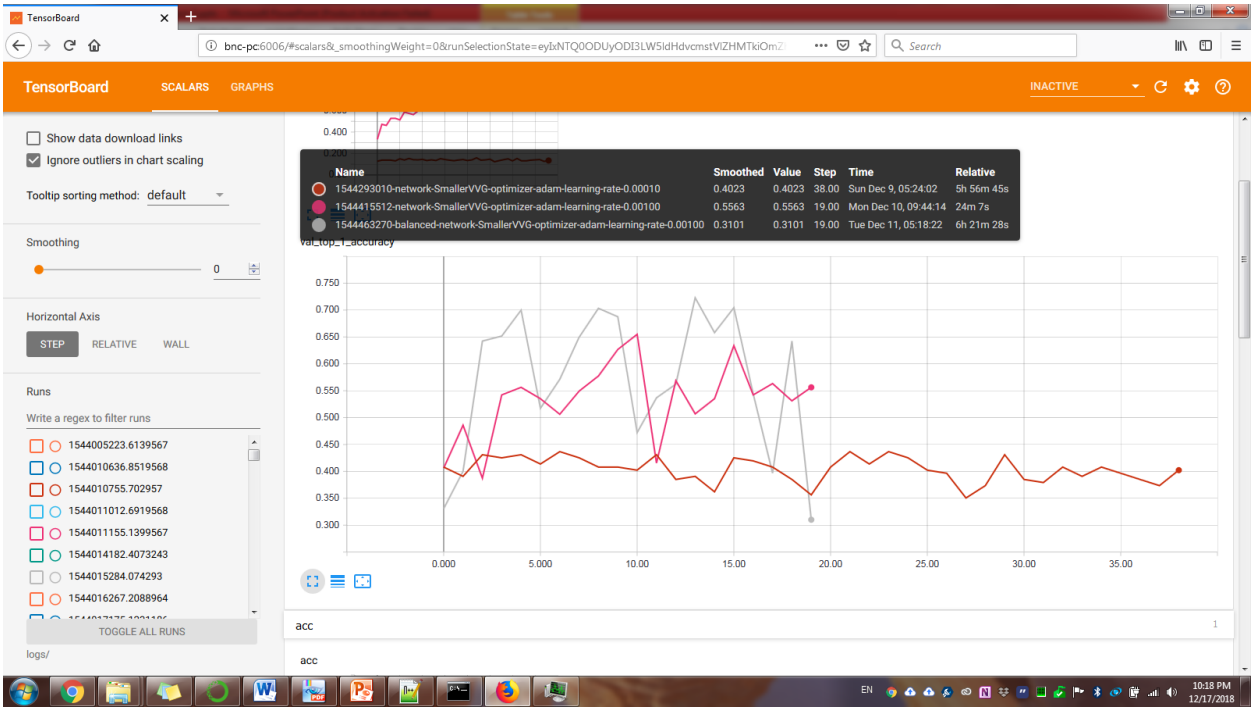
experiment #	**Model & parameters**	**Epochs**	**top_1 accuracy**	**val_top_1 accuracy**	**Observations**
1	smaller vvg with sigmoid final layer, down sample (116 images in each class), Input_shape(224, 224), Batch_size = 1, learning_rate = .001	39	13%	40%	
2	With: Input_shape(96, 96), Batch_size = 32	20	74%	55%	
3	**With: up sample (1879 images in each class)**	**14** **20**	**98%** **98%**	**72%** **31%**	**Up-sampling improved accuracy**
4	With: learning_rate = .0001	11		70%	
5	Added augmentation parameters shift, flip did				Did not improve
6	Added dropout to the input layer	12	82%	28%	Accuracy decreased
7	L2 regularization to dense layer				Did not improve

Part2: Transfer Learning

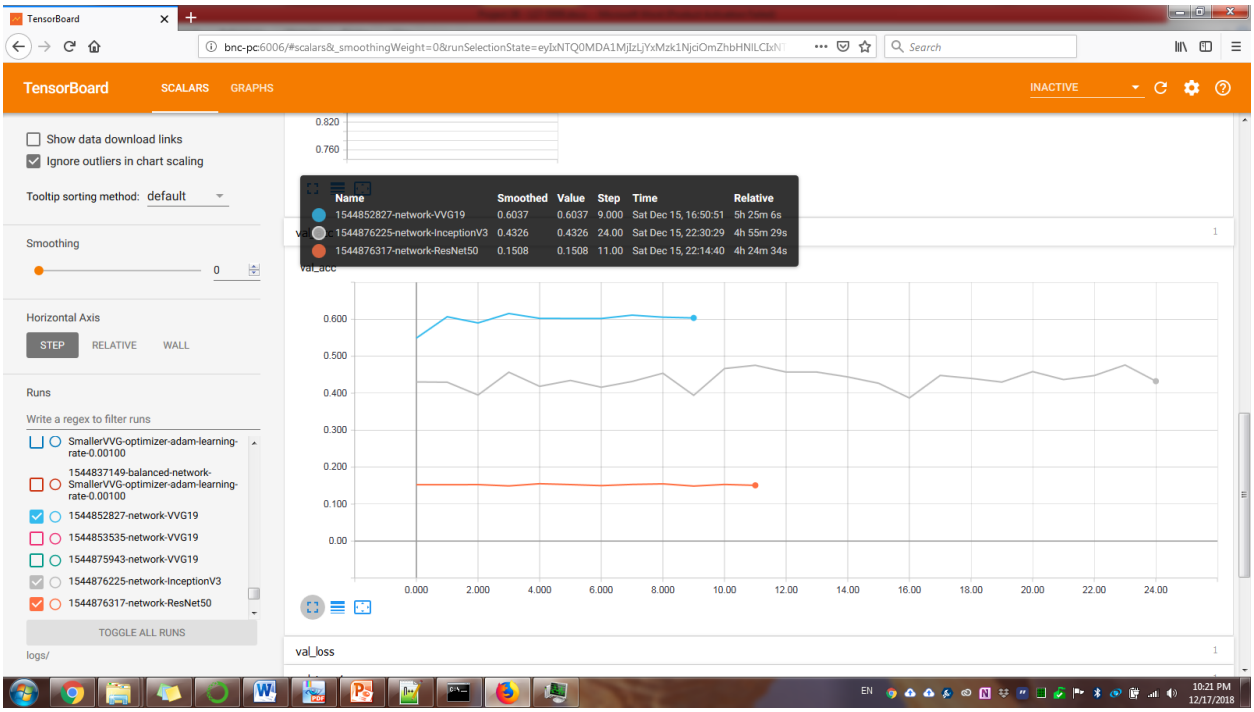
experiment #	**Model & parameters**	**Epochs**	**top_1 accuracy**	**val_top_1 accuracy**	**Observations**
Using pre-trained weights of "imageNet" and removing top layers of the mode. Add a pooling layer, one dense layer and one last "softmax" layer with random weights. Freeze all layers except new layers and train the network. **train_application.py** **graph-2**					
8	VGG19, input_shape= (64, 64)	10	84%	60%	Validation accuracy stays constant.
9	InceptionV3, input_shape= (96, 96)	25	61%	43%	
10	ResNet50, , input_shape= (96, 96)	12	96%	15%	
Fine-tune the above networks after un-freezing last block of each network and training with low learning rate and SGD optimizer **train_application_tune.py** **graph-3**					
11	VGG19	19	95%	69	Validation accuracy stays constant.
12	InceptionV3	20	72%	47	
13	ResNet50	17	95%	14	

Validation Accuracy graphs

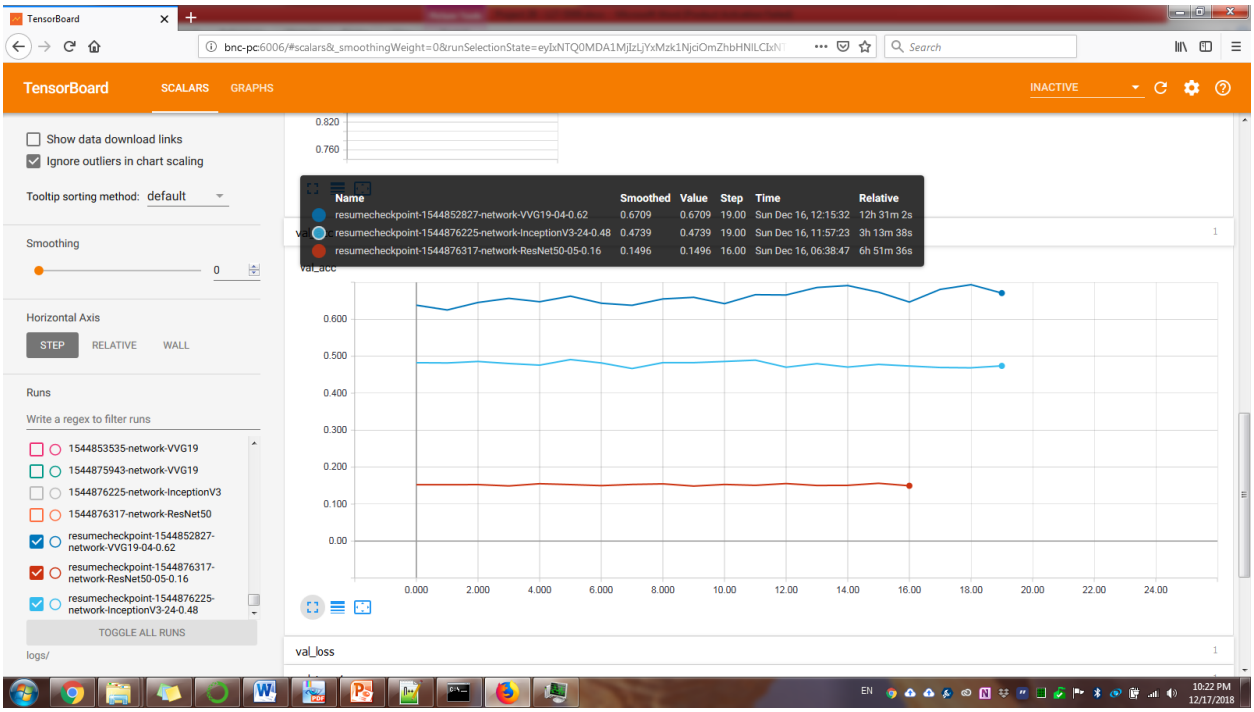
Graph-1: Part1 (train a network from scratch)



Graph-2: Part2-transfer learning (reuse “imageNet” weights” and add top layer and train)



Graph-3: Part2-transfer learning – fine tune (reuse “imageNet” weights” and fine tune last block of network)



Output-1 Validation accuracy with balanced and imbalanced data samples

1. **With balanced data samples:**

block5_conv1 (Conv2D) (None, 4, 4, 512) 2359808

_____

block5_conv2 (Conv2D) (None, 4, 4, 512) 2359808

_____

block5_conv3 (Conv2D) (None, 4, 4, 512) 2359808

_____

block5_conv4 (Conv2D) (None, 4, 4, 512) 2359808

_____

block5_pool (MaxPooling2D) (None, 2, 2, 512) 0

_____

global_average_pooling2d_1 ((None, 512) 0

_____

dense_11 (Dense) (None, 128) 65664

_____

dropout_3 (Dropout) (None, 128) 0

_____

dense_12 (Dense) (None, 7) 903

==========

Total params: 20,090,951

Trainable params: 9,505,799

Non-trainable params: 10,585,152

_____

Found 2521 images belonging to 7 classes.

acc: 0.6948412698412698

2. **With imbalanced data samples:**

```
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv4 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 2, 512)         0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 512)               0
_________________________________________________________________
dense_11 (Dense)             (None, 128)               65664
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_12 (Dense)             (None, 7)                 903
=================================================================
Total params: 20,090,951
Trainable params: 9,505,799
Non-trainable params: 10,585,152
_________________________________________________________________
Found 482 images belonging to 7 classes.
acc: 0.8020833333333334
```