Plant Pathology 2021

Innovative Assignment - Deep Learning (2CSDE61)

Submitted to: **Dr Priyank Thakkar**

Team Members:
18BCE114 Mansi G Palsana
18BCE115 Manushri Jain
18BCE128 Dhruvit Nagar
18BCE130 Nandit Pujara
18BCE164 Meet Patel

Introduction to Problem Statement

- For Plant Pathology 2021-FGVC8, the number of foliar disease images is increased and additional disease categories are added.
- The dataset contains approximately 23,000 high-quality RGB images of apple foliar diseases, including a large expert-annotated disease dataset.
- Although computer vision-based models have shown promise for plant disease identification, there are some limitations that need to be addressed.

Objective

- Develop models based on machine and deep learning for multilabel classification.
- To accurately classify a given leaf image from the test dataset to a particular disease category.
- To identify an individual disease from multiple disease symptoms on a single leaf image.

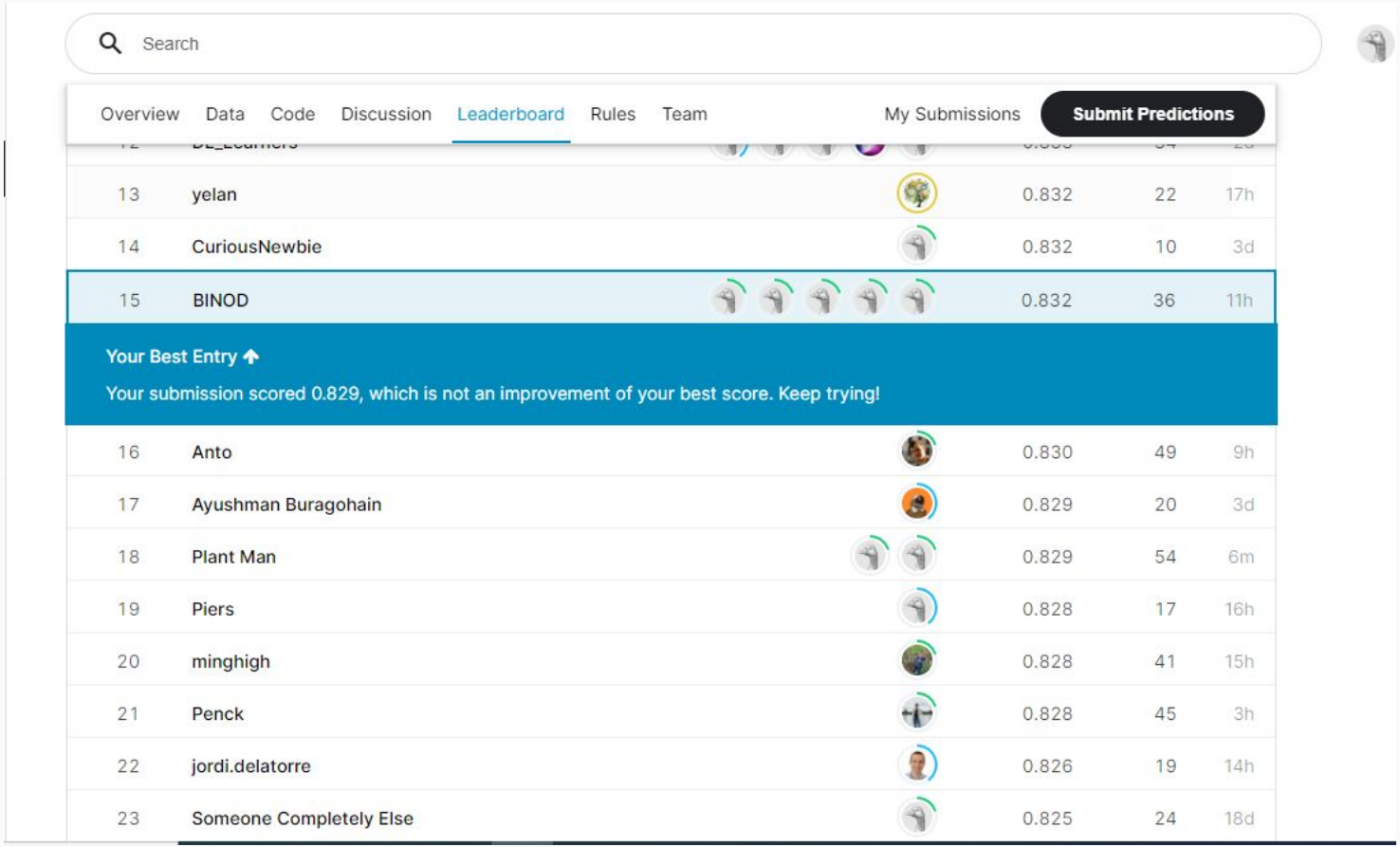
Proportions of Different Classes

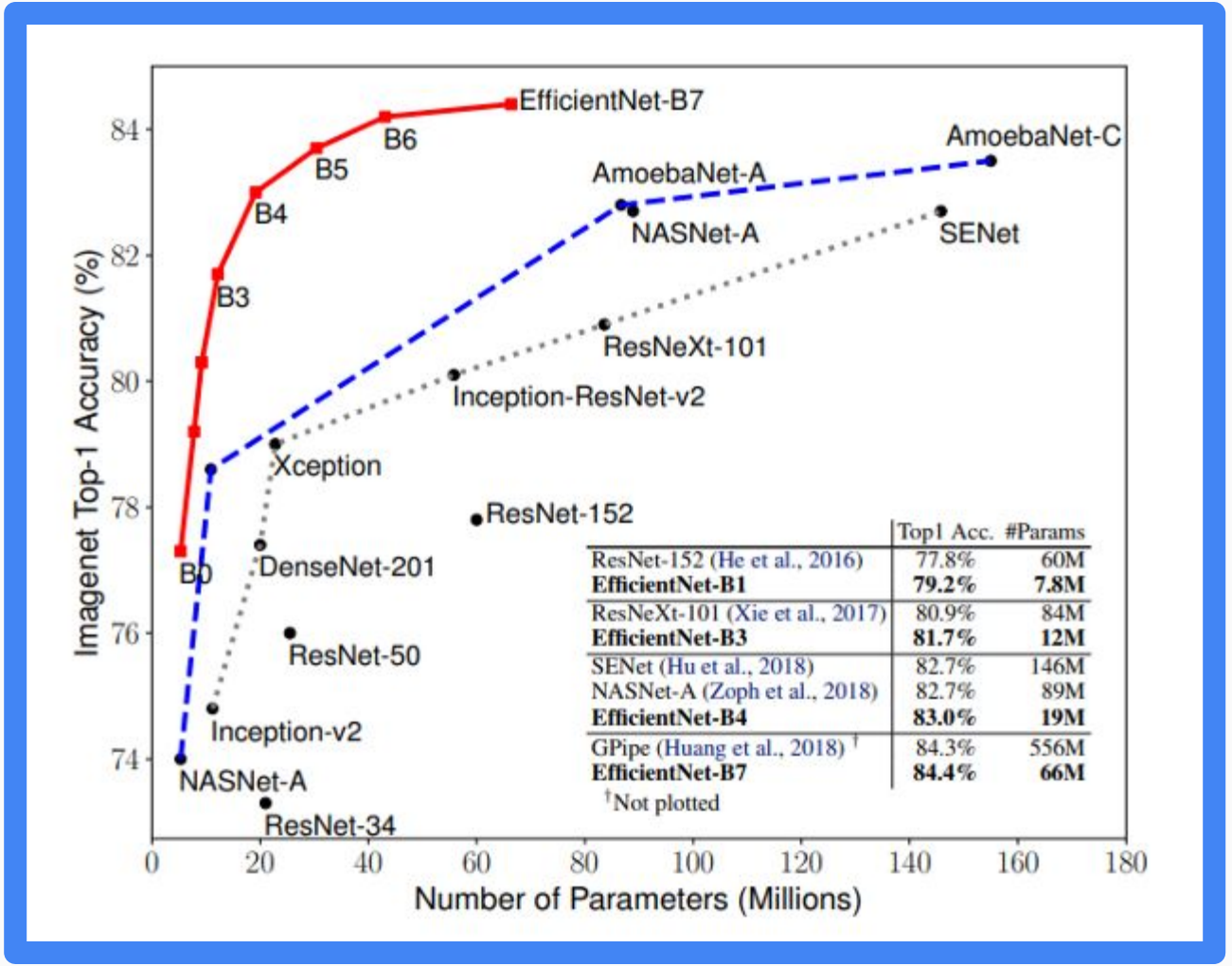
Scab	28.3%
Healthy	22.9%
Complex	10.7%
Rust	10.3%
frog_eye_leaf_spot	21.6%
Powdery_mildew	6.3%

Different Models used for Prediction

Model	Public Score
VGG	-
MobileNet	.444
InceptionV3	.562
ResNet-50	.496
DenseNet121	.806
ResNet-152	.781
Efficient NetB7	.832

Scoreboad


Search
Overview
Data
Code
Discussion
Leaderboard
Rules
Team
My Submissions
Submit Predictions
13 yelan 0.832 22 17h
14 CuriousNewbie 0.832 10 3d
15 BINOD 0.832 36 11h
Your Best Entry
Your submission scored 0.829, which is not an improvement of your best score. Keep trying!
16 Anto 0.830 49 9h
17 Ayushman Buragohain 0.829 20 3d
18 Plant Man 0.829 54 6m
19 Piers 0.828 17 16h
20 minghigh 0.828 41 15h
21 Penck 0.828 45 3h
22 jordi.delatorre 0.826 19 14h
23 Someone Completely Else 0.825 24 18d



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.2%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.7%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**83.0%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

EfficientNet

- One of the key issues in designing CNNs is model scaling i.e deciding how to increase the model size so as to provide better accuracy.
- This is a tedious process, requiring manual hit and trial until a sufficiently accurate model is produced.
- In order to pursue better accuracy, it is important to balance all dimensions of network width, depth, and resolution during ConvNet scaling.
- EfficientNet provides an effective compound scaling method for increasing the model size to achieve maximum accuracy gains.

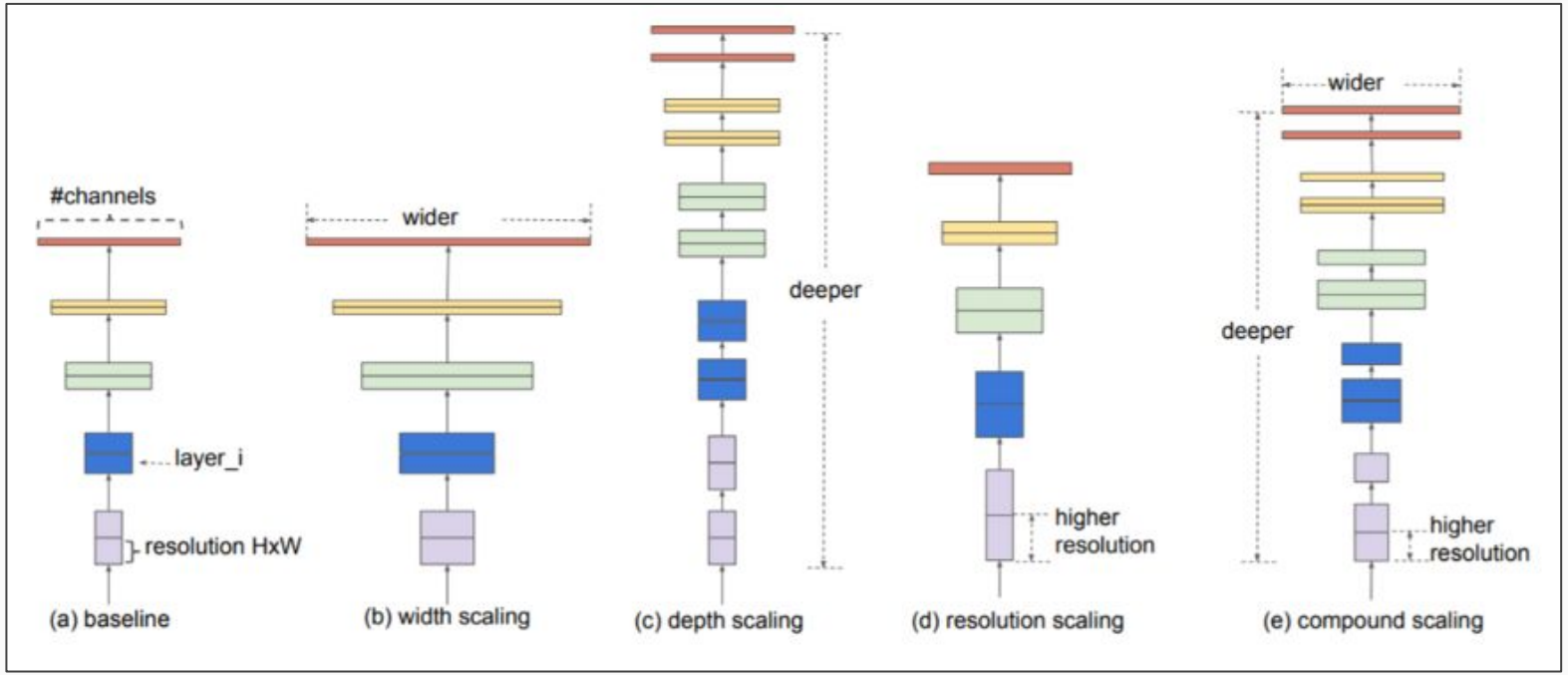
EfficientNetB7

- EfficientNet-B7 achieves 84.3% top-1 accuracy on ImageNet, while being 8.4x smaller and 6.1x faster on inference than the best existing ConvNet.
- Introduced by two engineers from Google brain team named Mingxing Tan and Quoc V. Le in a paper called “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”.
- EfficientNetB7 has 66M parameters and 37B FLOPS(Floating point Operations per second).
- The core idea was about strategically scaling deep neural networks.

Model Scaling

- Scaling the existing model in terms of model depth, model width and input image resolution.
- To improve the performance of the model.
- Depth wise scaling is the most popular among all.
- Compound scaling: It suggests that instead of scaling only one model attribute out of depth, width, and resolution, strategically scaling all three of them together delivers better results.

Compound Scaling



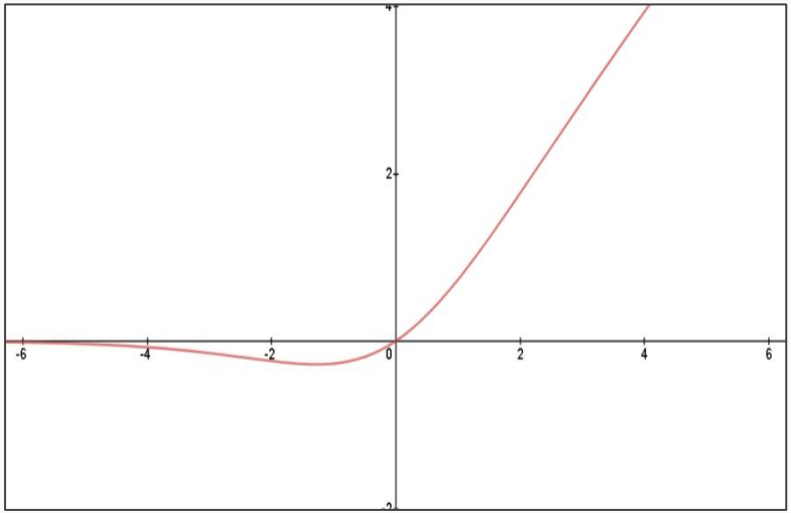
Baseline Architecture of EfficientNet

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

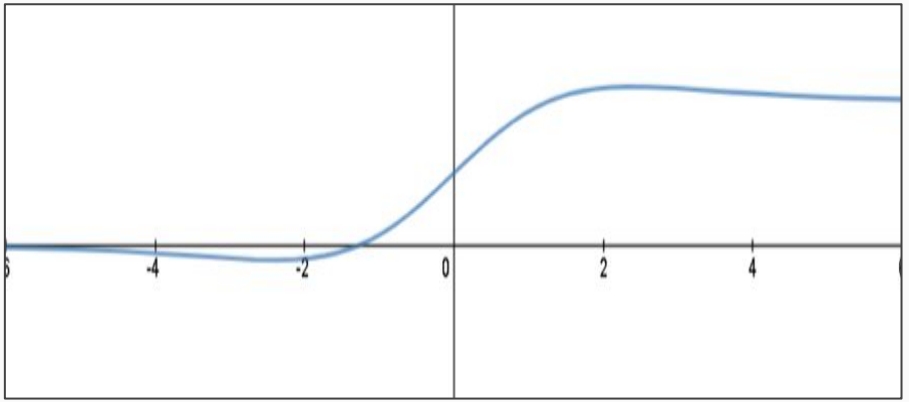
Building Blocks of EfficientNet

1. Swish Activation
2. Inverted Residual Block
3. Squeeze and Excitation Block

1. Swish Activation



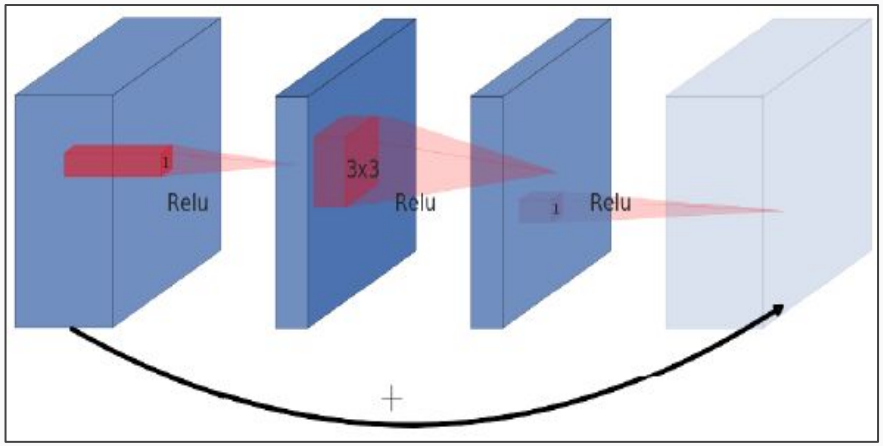
[Swish Activation]



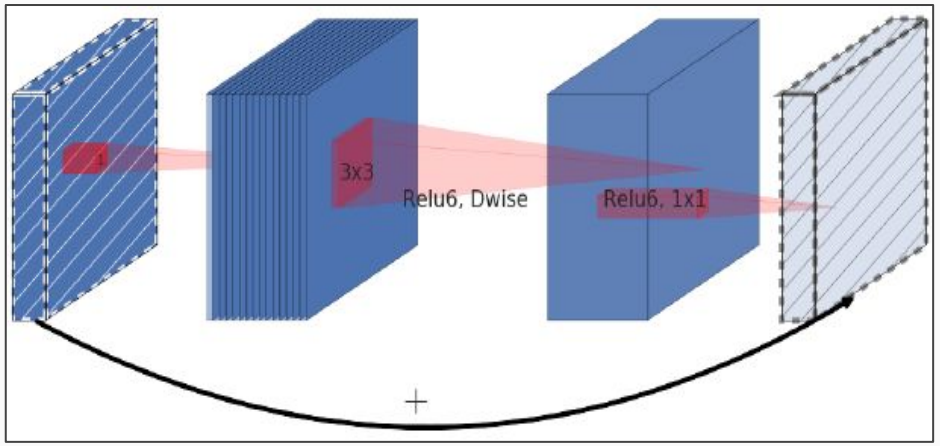
[Derivative of Swish Activation]

Swish(x) = x * Sigmoid(x)

2. Inverted Residual Block



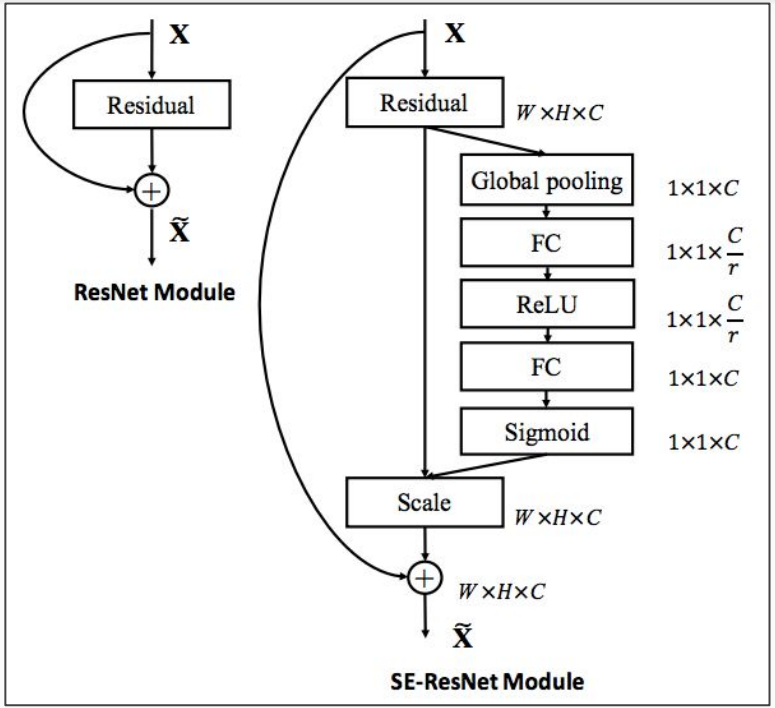
[Residual Block]



[Inverted Residual Block]

3. Squeeze and Excitation Block

- When CNN creates output feature map from a convolutional layer, it gives equal weightage to each of channels.
- Squeeze and excitation (SE) block is a method to give weightage to each channel instead of treating them all equally.
- SE block gives the output of shape (1 x 1 x channels) which specifies the weightage for each channel.



EfficientNet's MBConv Block

- MBConv block takes two inputs, first is data and the other is block arguments.
- The data is output from the last layer.
- A block argument is a collection of attributes to be used inside an MBConv block like input filters, output filters, expansion ratio, squeeze ratio etc.

- **Workflow of MBConv Block**

1. Expansion Phase
2. Depthwise Convolution Phase
3. Squeeze and Excitation Phase
4. Output Phase

Softmax Vs Sigmoid

- If our model's output classes are NOT mutually exclusive and we can choose many of them at the same time, hence in our model we used a sigmoid function on the network's raw outputs.

$$\sigma(z_j) = \frac{e^{z_j}}{1+e^{z_j}}$$

- If our model's output classes are mutually exclusive and we can only choose one, then we would have used a softmax function on the network's raw outputs.

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for j = 1,...,K}$$

Categorical Cross-Entropy Vs Binary Cross-Entropy

Categorical Cross-Entropy

- Categorical Cross-Entropy loss or Softmax Loss is a Softmax activation plus a Cross-Entropy loss.
- If we use this loss, we will train a CNN to output a probability over the C classes for each image.
- It is used for multi-class classification.

Binary Cross-Entropy

- Binary Cross-Entropy is a Sigmoid activation plus a Cross-Entropy loss.
- Unlike Softmax loss it is independent for each vector component (class).
- It is used for multi-label classification

One Cycle Learning Rate

- This scheduling function gradually increases the learning rate from a starting point up to a max value during a period of epochs.
- After that it will decrease the learning rate exponentially and stabilise it to a minimum value.

Augmentation

We applied many different transformations on the training dataset like:

- Flipping the images horizontally (left to right)
- Flipping the images vertically (up to down)
- Scaling the brightness
- Scaling the contrast
- Rotating the image matrix by some degree
- Height and width zoom
- Height and width shift
- Shearing the image

Novelty

- Custom Data-Augmentation functions.
- One-cycle learning rate.
- Used TPU to make training fast
- Transfer Learning using Efficient Net-B7, InceptionV3, DenseNet121, VGG, MobileNet, ResNet-50, 152.

CLAHE
(Contrast Limited Adaptive Histogram Equalization)

[Original Image]

[CLAHE - BGR Image]

[CLAHE - RGB Image]

Conclusion

After trying out different models, we found that EfficientNetB7 gave the best performance for the given problem. We also found that using data augmentation on the training images helped to improve the accuracy of the model.

References

- https://gogul.dev/software/mobile-net-tensorflow-js
- https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec
- https://cv-tricks.com/keras/understand-implement-resnets/
- https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a
- https://cloud.google.com/tpu/docs/inception-v3-advanced
- https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5fd5b736142
- https://towardsdatascience.com/efficientnet-scaling-of-convolutional-neural-networks-done-right-3fde32aef8ff
- https://medium.com/analytics-vidhya/image-classification-with-efficientnet-better-performance-with-computational-efficiency-f480fdb00ac6#:~:text=The%20EfficientNet%2DB0%20architecture%20wasn,accuracy%20and%20floating%2Dpoint%20operations.
- https://www.kaggle.com/cdeotte/rotation-augmentation-gpu-tpu-0-96/notebook
- https://www.tensorflow.org/api_docs/python/tf/data/Dataset
- https://dzlab.github.io/dltips/en/keras/learning-rate-scheduler/#:~:text=Time%2Dbased%20Decay,is%20the%20epoch%2Fiteration%20number