Plant Pathology 2021 - FGVC8

Deep Learning Innovative assignment
2CSDE61

Presented by:
Ishan Tewari - 18BCE080
Jyot Makadiya - 18BCE088
Vatsal Kevadiya - 18BCE092
Labdhi Sheth - 18BCE101
Urvashi Ramdasani - 18BCE247

INDEX

Details about the competition:

The competition revolves around **Foliar (leaf) diseases in apple orchards** which pose a major threat to the overall productivity and quality of apple.

There are computer vision based models which help detect the diseased leaf but there are large variations in the visual symptoms of a single leaf disease across different cultivators. The reasons can be because of the different conditions under which the image is captured, for example: the difference in natural and image capturing environments where the characteristics of the leaf like the leaf color and leaf morphology, age of the infected tissues changes due to different light illumination during imaging.

.

About the dataset

Plant Pathology 2020-FGVC7 challenge competition had a pilot dataset of 3,651 RGB images of foliar disease of apples. For Plant Pathology 2021-FGVC8, we have

- Approximately 23,000 high-quality RGB images of apple foliar diseases
- A large expert-annotated disease dataset.
- Test images has 3 images in it
- Train_csv file has two columns one being images and other labels
- A submission_csv has been provided as a sample

Thus there are 18.6k images and 2 csv files provided to us.

About the dataset

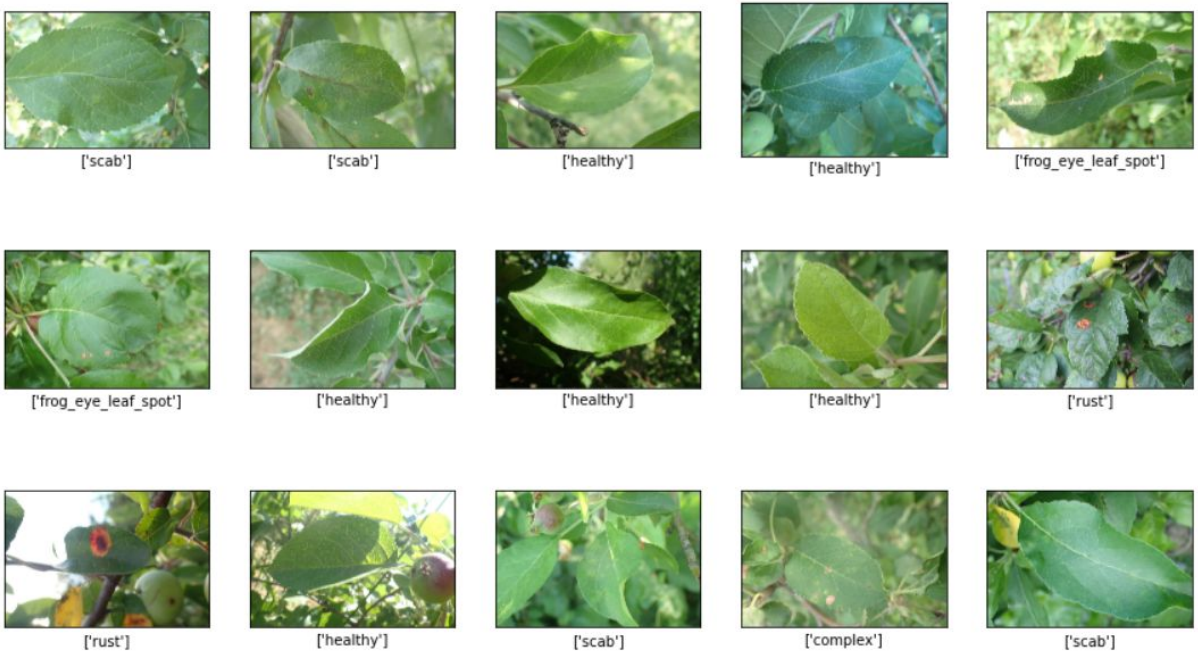
Train data set -

- shape (18632, 2)
- First column is “image” with the image paths and the second column is “label” with unique values as:

```
Counter({'healthy': 4624,
         'scab frog_eye_leaf_spot complex': 200,
         'scab': 4826,
         'complex': 1602,
         'rust': 1860,
         'frog_eye_leaf_spot': 3181,
         'powdery_mildew': 1184,
         'scab frog_eye_leaf_spot': 686,
         'frog_eye_leaf_spot complex': 165,
         'rust frog_eye_leaf_spot': 120,
         'powdery_mildew complex': 87,
         'rust complex': 97})
```

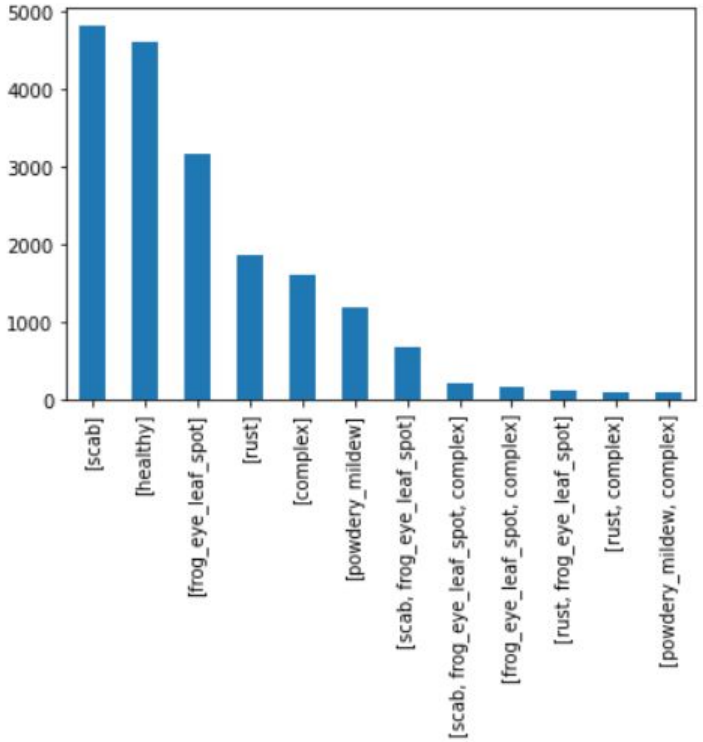
```
Counter({'healthy': 4624,
         'scab': 5712,
         'frog_eye_leaf_spot': 4352,
         'complex': 2151,
         'rust': 2077,
         'powdery_mildew': 1271})
```

Visualization

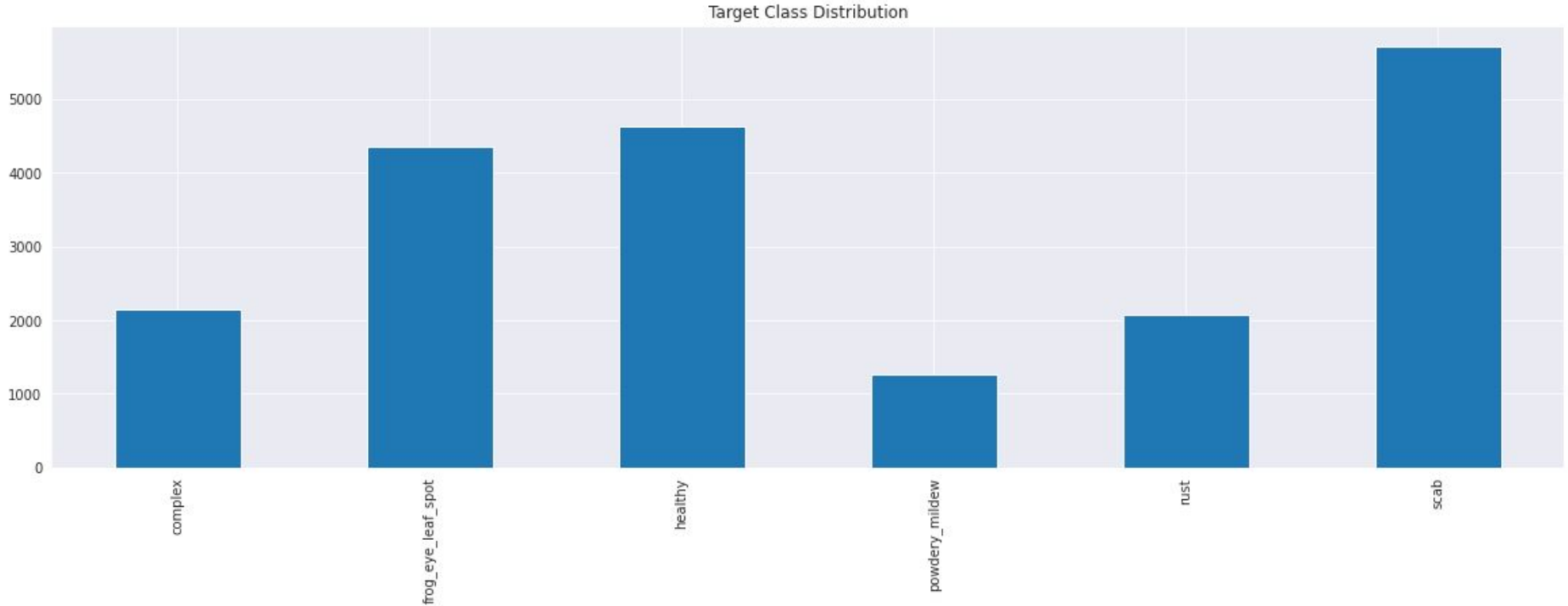


Some sample images from the dataset

```
[scab]                                    4826
[healthy]                                 4624
[frog_eye_leaf_spot]                      3181
[rust]                                    1860
[complex]                                 1602
[powdery_mildew]                          1184
[scab, frog_eye_leaf_spot]                 686
[scab, frog_eye_leaf_spot, complex]        200
[frog_eye_leaf_spot, complex]              165
[rust, frog_eye_leaf_spot]                 120
[rust, complex]                             97
[powdery_mildew, complex]                   87
Name: labels, dtype: int64
```



Distribution of classes in the dataset


Target Class Distribution
5000
4000
3000
2000
1000
0
complex
frog_eye_leaf_spot
healthy
powdery_mildew
rust
scab

Our attempts -1

One of our attempt was done using InceptionV3 model. Its details are as follows:

- The image below shows the model.
- (2672, 4000, 3) -> (512,512,3)
- During training loss: 0.0209 - val_loss: 0.0832 and public score of 79% was achieved by this.

```
inputs = tf.keras.Input(shape=(512,512, 3))
x = tf.keras.applications.InceptionV3(
    include_top=False,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
)(inputs)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
outputs = tf.keras.layers.Dense(6, activation='sigmoid')(x)

model = tf.keras.models.Model(inputs, outputs)
model.compile(loss='binary_crossentropy', optimizer=tf.keras.optimizers.Adam(lr=1e-4))
```

Our attempts - 2

1. Another attempt includes fine-tuning EfficientNet B7 and ResNet152V2. Its details are as follows:
   - The model consists of EffNet B7 followed by GlobalAveragePooling and Dense Output layer.
   - Data augmentation with output size 600 was used to get better results.
   - Accuracy was 90% and F1 score was 88% on validation set for EffNet B7.
   - ResNet152V2 perform slightly better than EffNet B7 with accuracy being 91% and F1 score being 89% for the validation set.
   - Ensembling models did not improve leaderboard score.

Our attempts - 3

Next attempt includes the use pre-trained Xception model with imagenet weights. Its details are as follows:

1. Data was augmented by flipping the images left-right and up-down.
2. The pre-trained model was used by adding Global Average Pooling and Dense layers on the top.
3. Model trained using Adam optimizer and evaluated using f1-score.
4. Achieved accuracy of 88% and Public Score of 60.7%.

Our attempts - 4

- In this attempt, I used pre-trained VGG16 on the imagenet dataset.
- Data augmentation - Image was converted into 224 X 224.
- In training parameters of pre-trained were also updated.
- Public score of 67.7%

```
inputs = tf.keras.Input(shape=(224,224, 3))
model_vgg16 = tf.keras.applications.vgg16.VGG16(include_top=False, weights='imagene
t')

for i in model_vgg16.layers:
    i.trainable=True

model_vgg16 = model_vgg16(inputs)

x = tf.keras.layers.Flatten()(model_vgg16)
outputs = tf.keras.layers.Dense(6, activation='sigmoid')(x)

model = tf.keras.models.Model(inputs, outputs)
```

Our attempts - 5

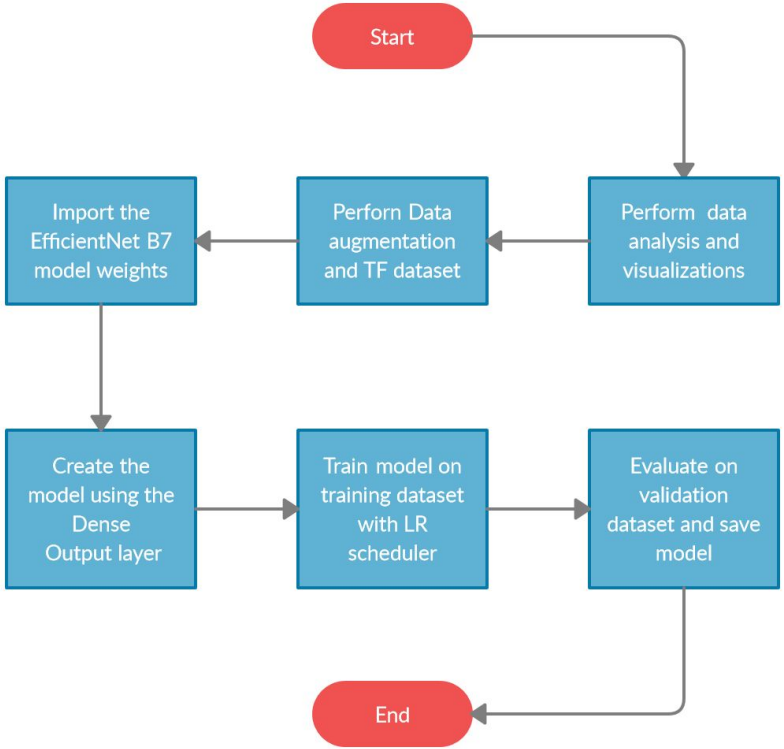
We also tried using a simple model using ResNet152V2 to see the accuracy. Its details are as follows:

1. Image was converted into 224 X 224 patch.
2. It was also normalized by dividing by 255.
3. Only data augmentation was done by horizontally flipping the images.
4. The pre-trained model was used by adding just Dense layer on the top with 6 output neurons.
5. Model trained using Adam optimizer and evaluated using accuracy.
6. Model was trained for 10 epochs and gave a public score of just 44.5%.

Best attempt - Training

- In the leaderboard EfficientNet B7 outperformed other architectures with public leaderboard score of 80.4%.
- Heavy augmentation was used such as random flip, resize + crop, random saturation, hue, brightness and rotation. Final training dataset image size was 600 x 600.
- Validation test size was 0.1 (no stratification only randomly selected)
- Trained model on TPU using tensorflow and keras.
- Used 6 labels: 5 for each disease + 1 for healthy
- Trained using AdamOptimizer with custom learning rate scheduler to save pre-trained weights from breaking.
- Used Binary_CrossEntropy as the problem was multilabel classification.

Best attempt - Training Algorithm

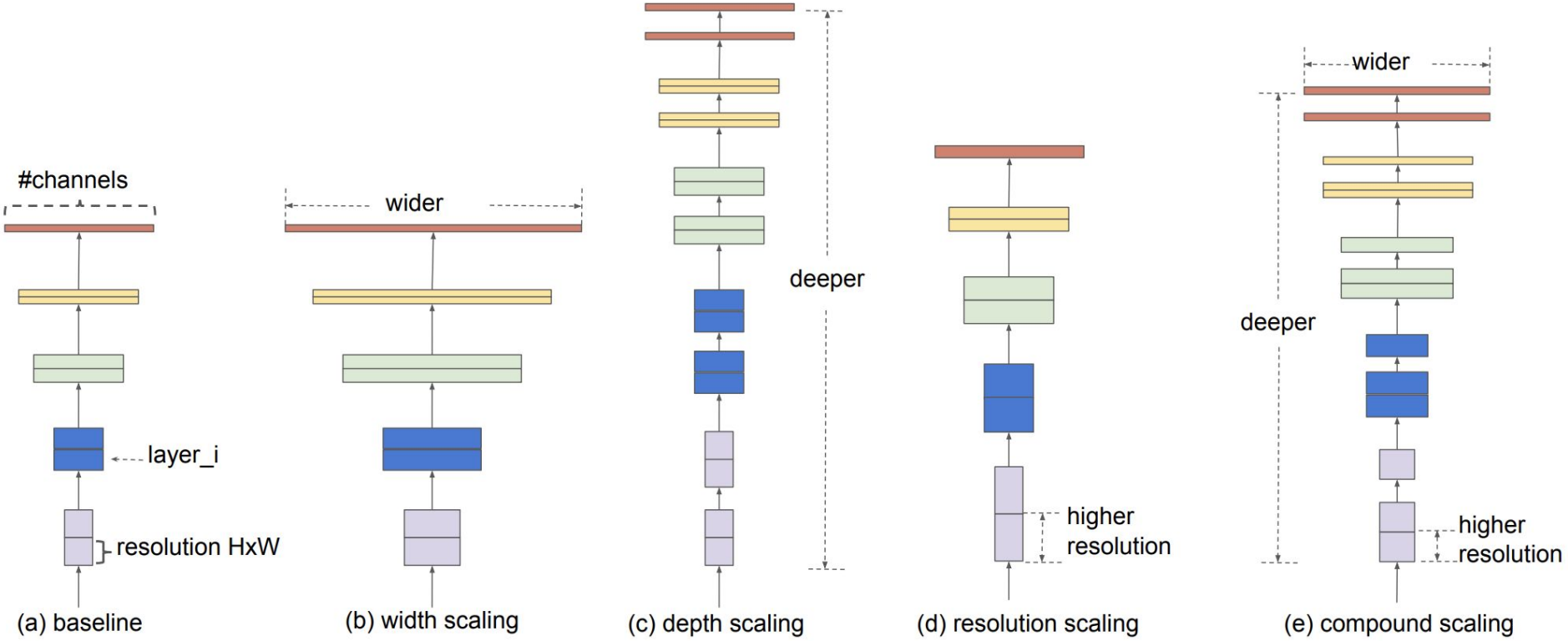


Best attempt - Inference

- Image was converted into 600 X 600.
- As a prediction we got 6 activations for each image. These activations were between 0 and 1 because of sigmoid activation function.
- Class Based Thresholds for each class were used.
- In prediction it may happen that an image be classified as healthy and also having some diseases. In that case image will be classified as healthy only.
- Public score of 80.4%

About EfficientNet

- EfficientNet was a paper by Mingxing Tan and Quoc V. Le, in which they look at the idea of scaling a ConvNet differently.
- It is a family of nets which focuses on scaling of ConvNets.
- The most common ways is to scale up ConvNets:
    1. By their depth
    2. By their width
    3. By Image Resolution (Less Common)
- Previously it was common to scale only one of the three dimensions – depth, width, and image size.
- While, this paper introduced a new method to scale the neural network in a compound manner.


#channels
layer_i
resolution HxW
(a) baseline
wider
(b) width scaling
deeper
(c) depth scaling
higher
resolution
(d) resolution scaling
wider
deeper
higher
resolution
(e) compound scaling

About EfficientNet B7

- They thought of scaling

$$\begin{aligned} \text{depth: } & d = \alpha^\phi \\ \text{width: } & w = \beta^\phi \\ \text{resolution: } & r = \gamma^\phi \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned}$$

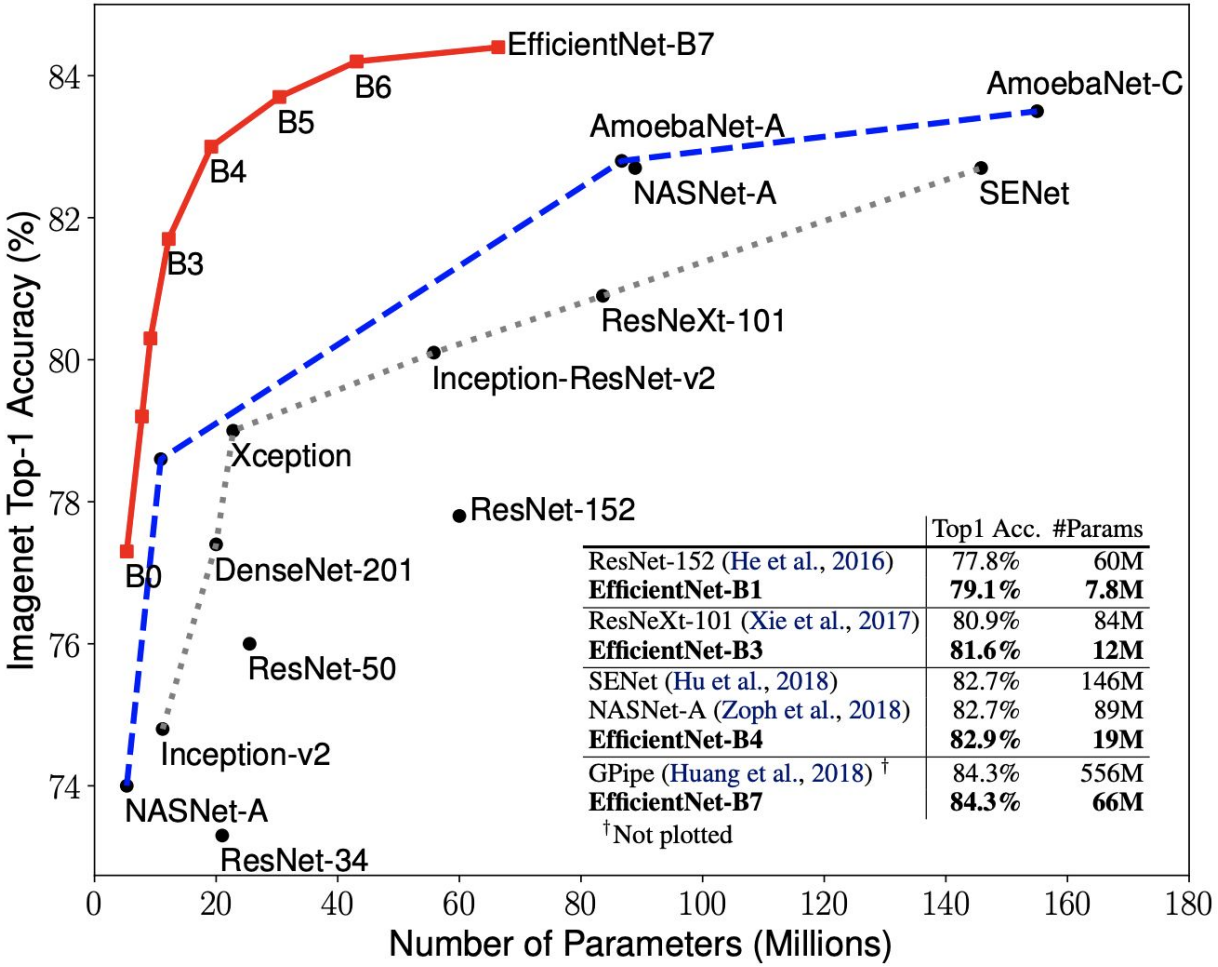
- This increased the total FLOPS by 2^ϕ.

Baseline EfficientNet B0 Architecture

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

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.1%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.1%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.6%**	**95.7%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.9%**	**96.4%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.6%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.3%**	**97.0%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

We omit ensemble and multi-crop models (Hu et al., 2018), or models pretrained on 3.5B Instagram images (Mahajan et al., 2018).



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

Conclusion

- The competition is of classification of foliar (leaf) diseases in apple orchards.
- The dataset consists of train and test leaf images. It also consists of CSV file containing the image name and labels.
- Different approaches use InceptionV3, ResNet152V2, EfficientNetB7, Xception, VGG16
- Best performance observed from EfficientNetB7 with public score 80.4%.

Possible Future Description

- Instead of using EfficientNet B7,

  we can try scaling up different state-of-the-art networks and see the performance:

  1. ResNet,
  2. Xception
- Ensemble different models with different Class based thresholds
- Use stratified folds for dataset normalization

Appendix

Our code can be found here:

- Plant Pathology 2020 solution starter | Kaggle
- Plant Pathology 2021 Submission [Inference] | Kaggle

THANK YOU