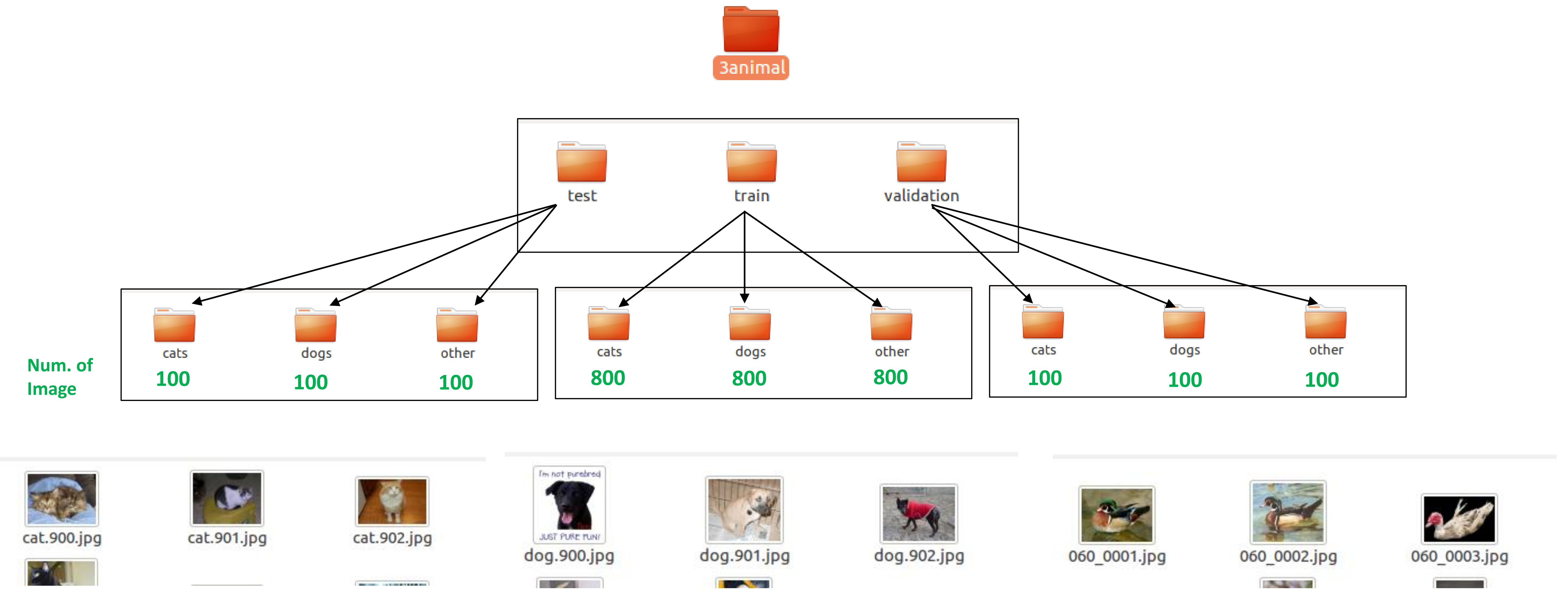
DOE for Weed Detection Simulation

Sayem

3/1/2018

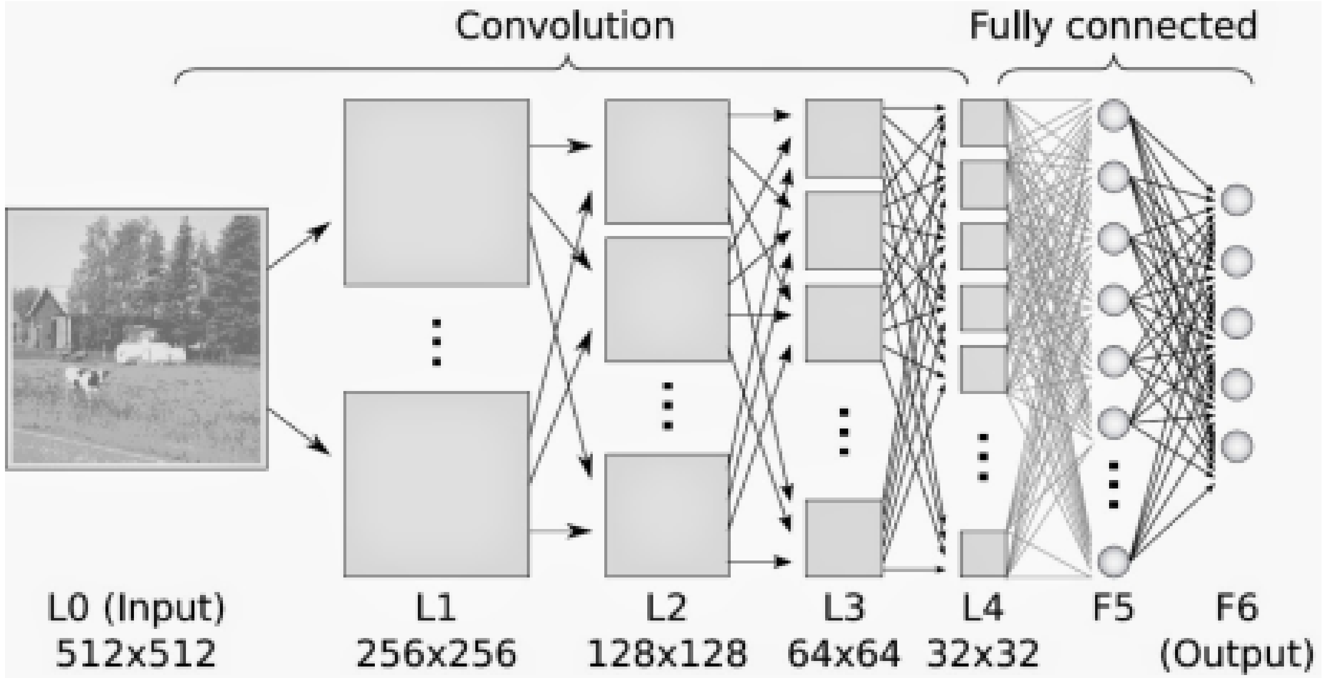
DATA


3animal
test
train
validation
cats
dogs
other
cats
dogs
other
cats
dogs
other
Num. of Image
100
100
100
800
800
800
100
100
100
cat.900.jpg
cat.901.jpg
cat.902.jpg
dog.900.jpg
dog.901.jpg
dog.902.jpg
060_0001.jpg
060_0002.jpg
060_0003.jpg

Steps

1. Build a small CNN (4 layers)
2. Small CNN + data Augmentation
3. Use a Pre-trained CNN
4. Use Pre-trained CNN + data augmentation
5. Use Pre-trained CNN + data augmentation + train last few layers of Pre-trained CNN
6. Hyperparameter tuning

1. Build a small CNN (4 layers)

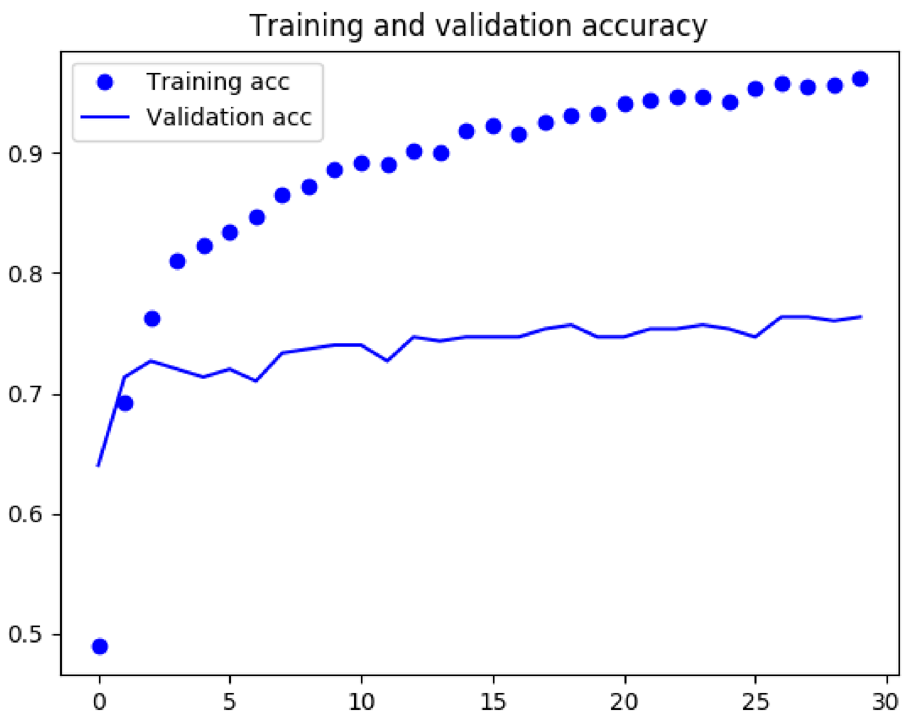


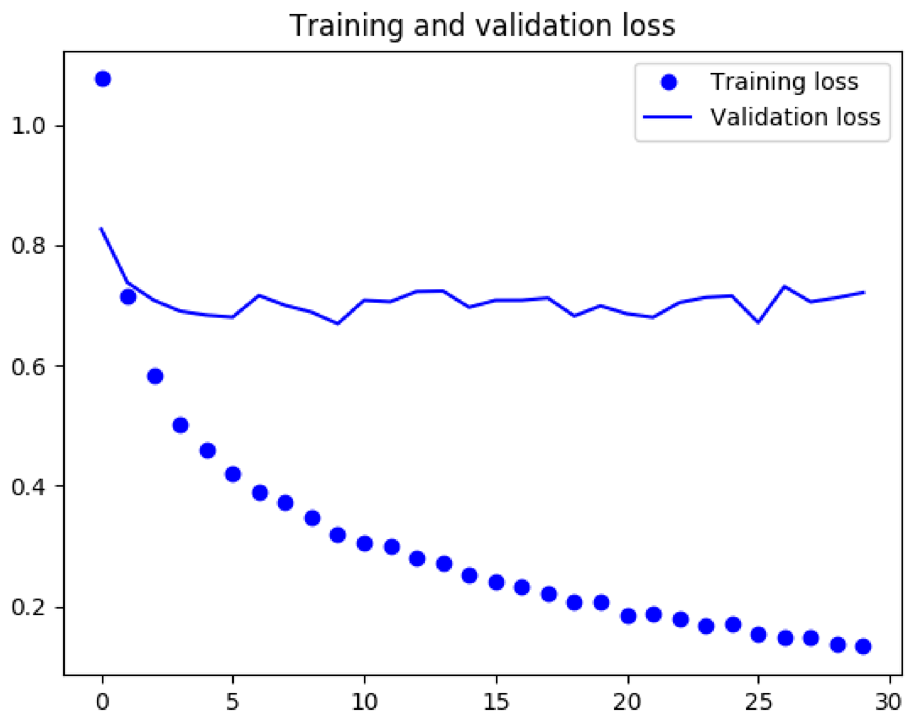

```
_________________________________________________________________
Layer (type)                 Output Shape
=================================================================
conv2d_1 (Conv2D)            (None, 148, 148, 32)
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 32)
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 72, 72, 64)
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 36, 36, 64)
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 34, 128)
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 17, 17, 128)
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 15, 15, 128)
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 128)
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)
_________________________________________________________________
dense_1 (Dense)              (None, 512)
_________________________________________________________________
dense_2 (Dense)              (None, 1)
=================================================================
```

1. Build a small CNN (4 layers)

Run Time (GPU): 46 secs

Overfitting : Increase training data to compensate

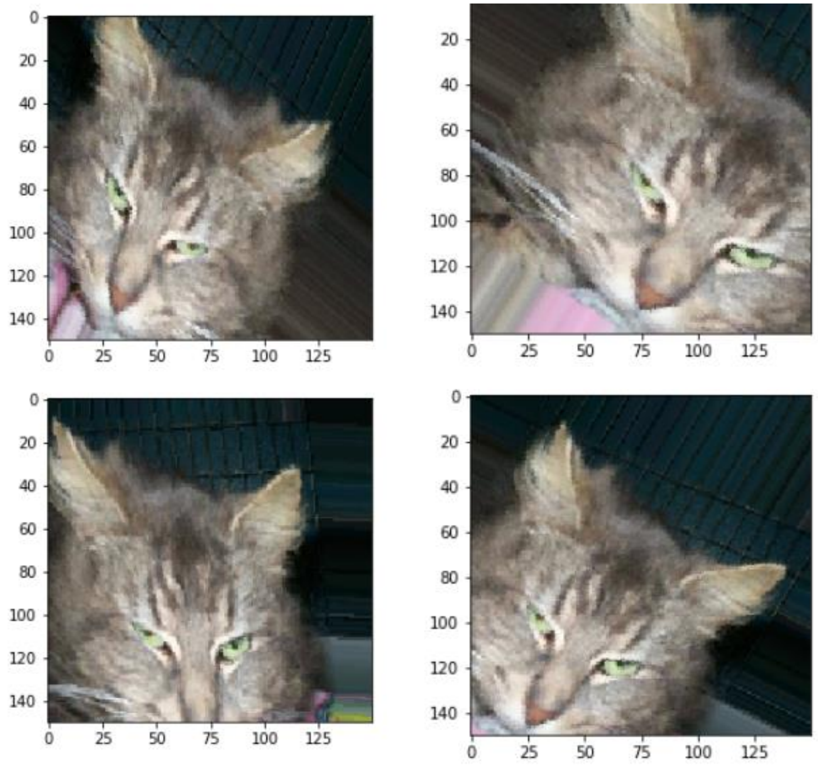


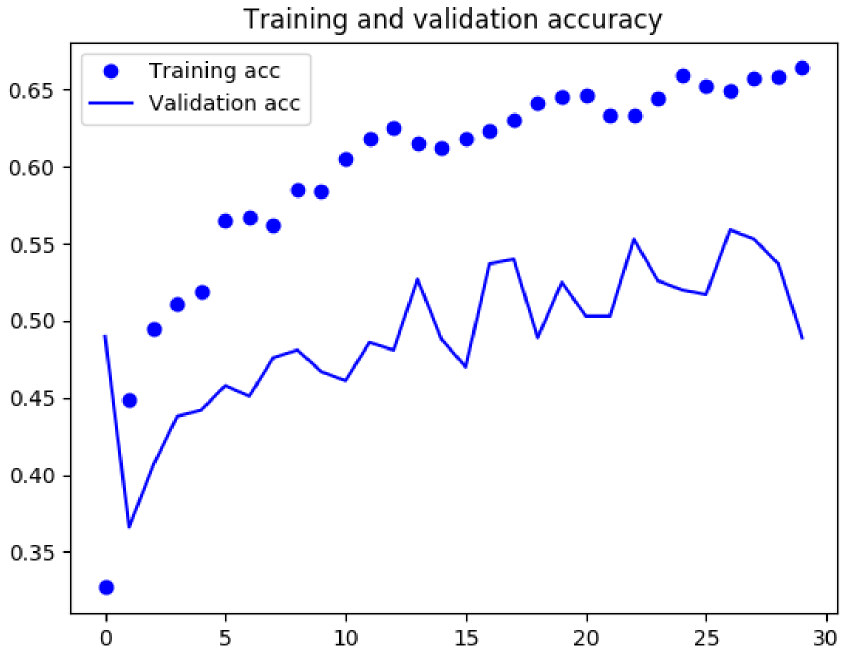


2. Small CNN + data Augmentation

Data augmentation takes the approach of generating more training data from existing training samples, by "augmenting" the samples via a number of random transformations that yield believable-looking images.

```
datagen = ImageDataGenerator(
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')
```

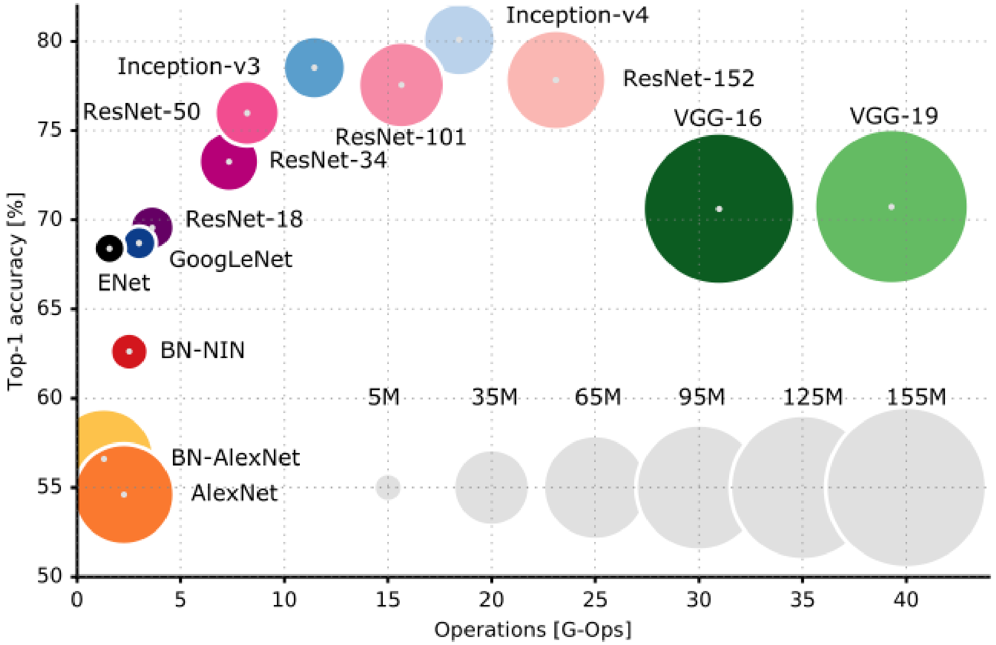




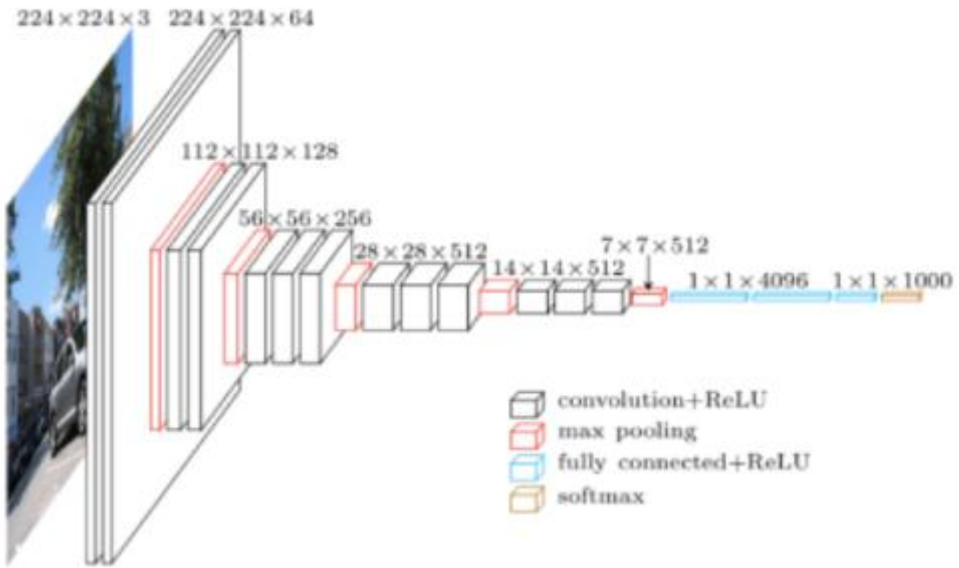
ImageNet

- The **ImageNet** project is a large visual database designed for use in visual object recognition software research
- Over 14 million URLs of images have been hand-annotated by ImageNet to indicate what objects are pictured
- ImageNet contains over 20 thousand ambiguous categories
- Now, in the case of **top-1** score, you check if the top class (the one having the highest probability) is the same as the target label.
- In the case of **top-5** score, you check if the target label is one of your top 5 predictions (the 5 ones with the highest probabilities).

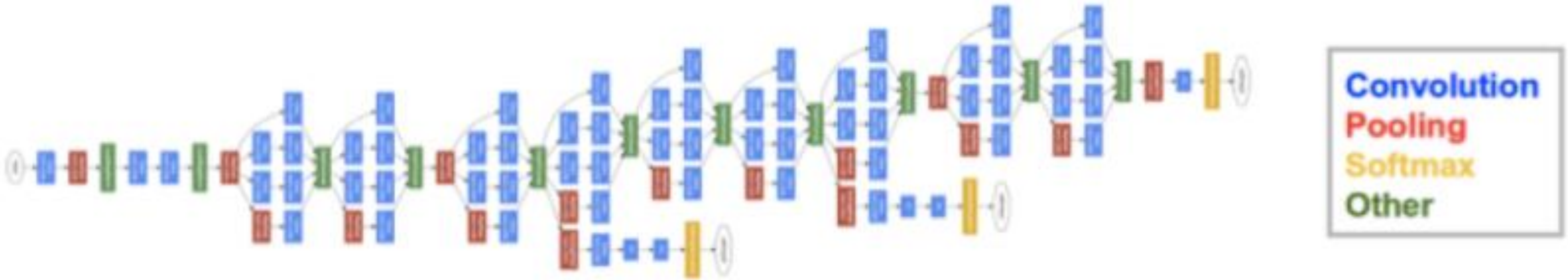
3. Use a Pre-trained CNN



Schematic Diagram of VGG16 Model:



Schematic Diagram of the 27-layer Inception-V1 Model (Idea similar to that of V3):



3. Use a Pre-trained CNN

1. ***Chollet, François. "Xception: Deep learning with depthwise separable convolutions." arXiv preprint (2016).***

2. ***Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).***

3. ***Szegedy, Christian, et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." AAAI. Vol. 4. 2017.***

4. ***Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. Vol. 1. No. 2. 2017.***

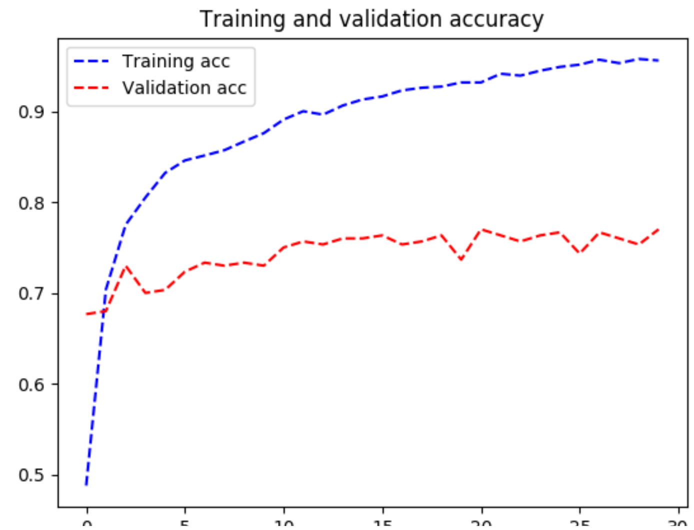
Available Pre-trained CNN models

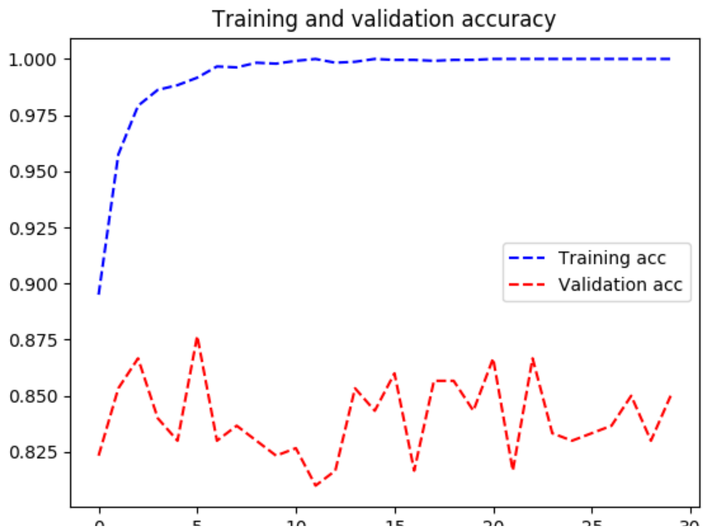
Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.715	0.901	138,357,544	23
VGG19	549 MB	0.727	0.910	143,667,240	26
ResNet50	99 MB	0.759	0.929	25,636,712	168
InceptionV3	92 MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215 MB	0.804	0.953	55,873,736	572
MobileNet	17 MB	0.665	0.871	4,253,864	88
DenseNet121	33 MB	0.745	0.918	8,062,504	121
DenseNet169	57 MB	0.759	0.928	14,307,880	169
DenseNet201	80 MB	0.770	0.933	20,242,984	201

Xception input size for this model is 299x299 or lower
VGG16 input size for this model is 224x224 or lower
InceptionResNetV2 input size for this model is 299x299 or lower
DenseNet201 input shape has to be (224, 224, 3)

3. Use a Pre-trained CNN

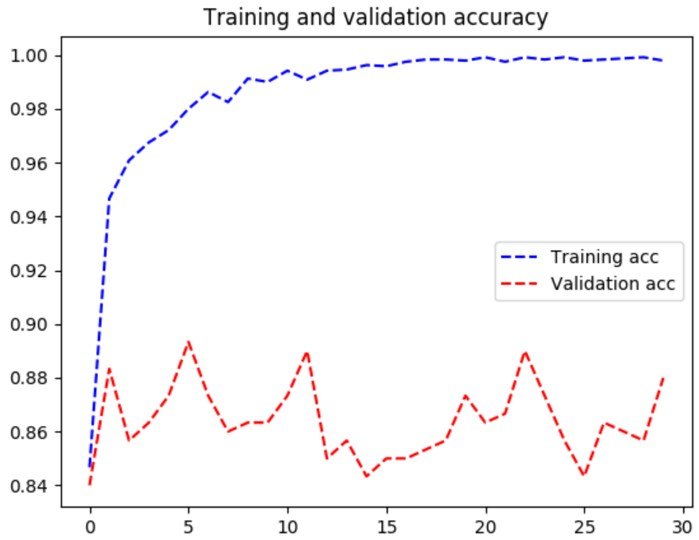
VGG16
i5 cpu: 13 min 58 sec
GPU : 44 sec

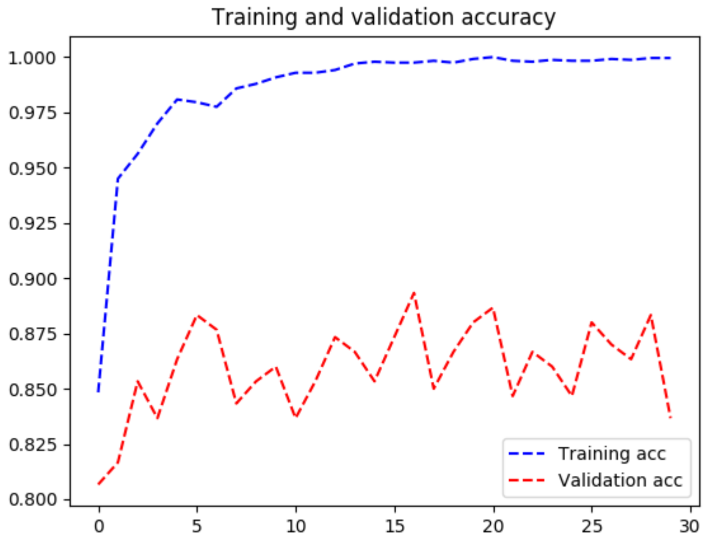




Xception
i5 cpu: 13 min 44 sec
GPU : 1 min 40 sec

InceptionResNetV2
i5 cpu: 9 min 44 sec
GPU :1 min 22 sec

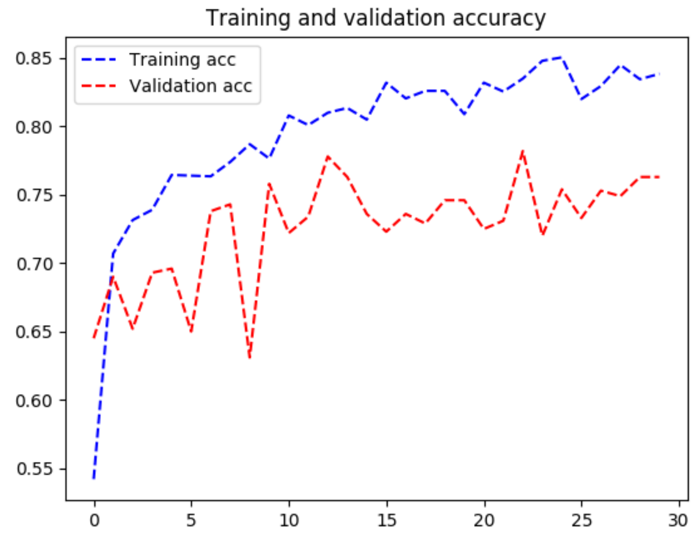


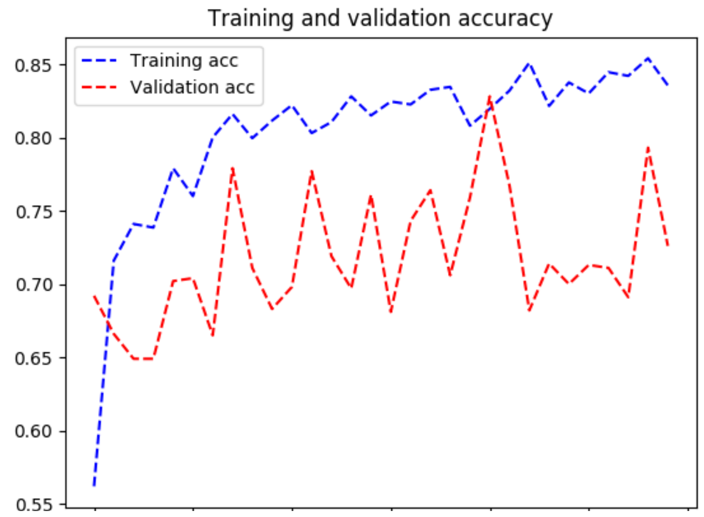


DenseNet201
i5 cpu: 8 min 59 sec
GPU : 1 min 24 sec

4. Use Pre-trained CNN + data augmentation

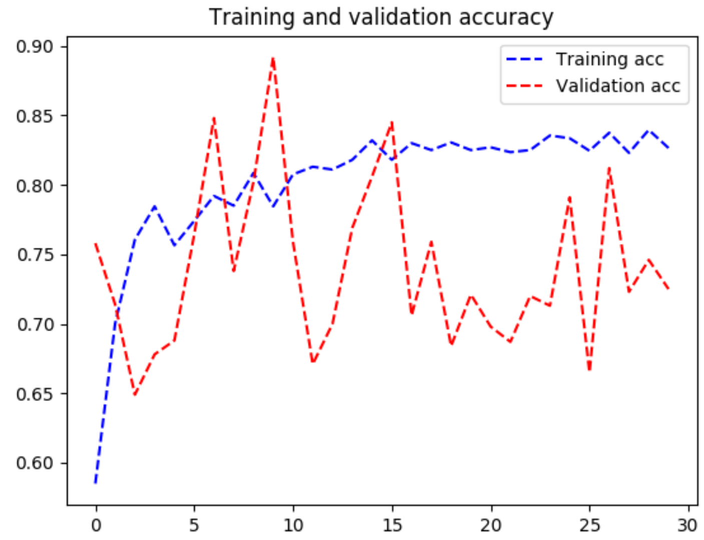
VGG16
GPU : 8 min 46 sec

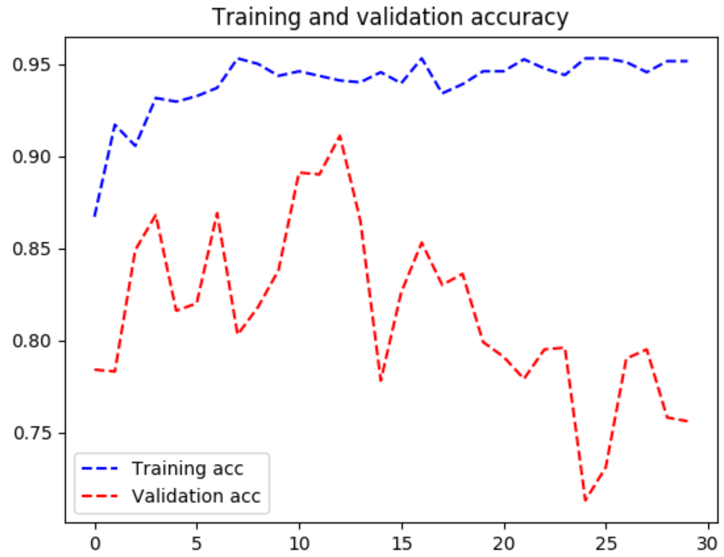




Xception
GPU : 8 min 40 sec

InceptionResNetV2
GPU :9 min 55 sec

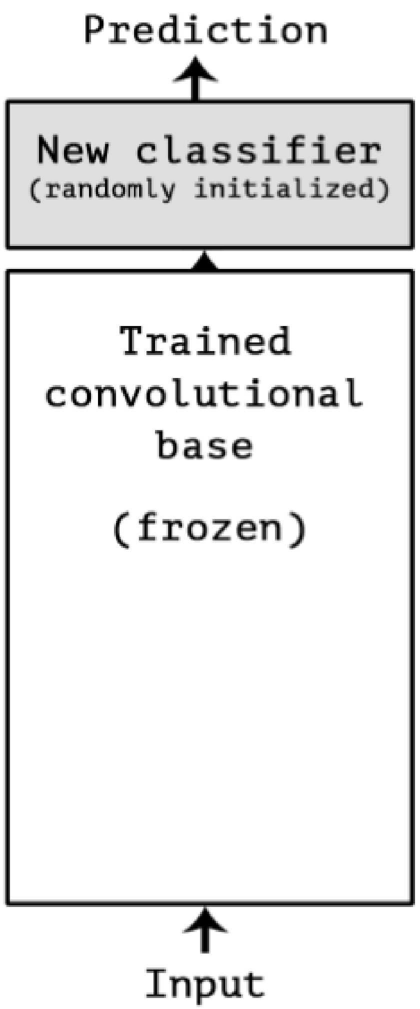




DenseNet201
GPU : 19 min 56 sec

5. Use Pre-trained CNN + data augmentation + train last few layers of Pre-trained CNN

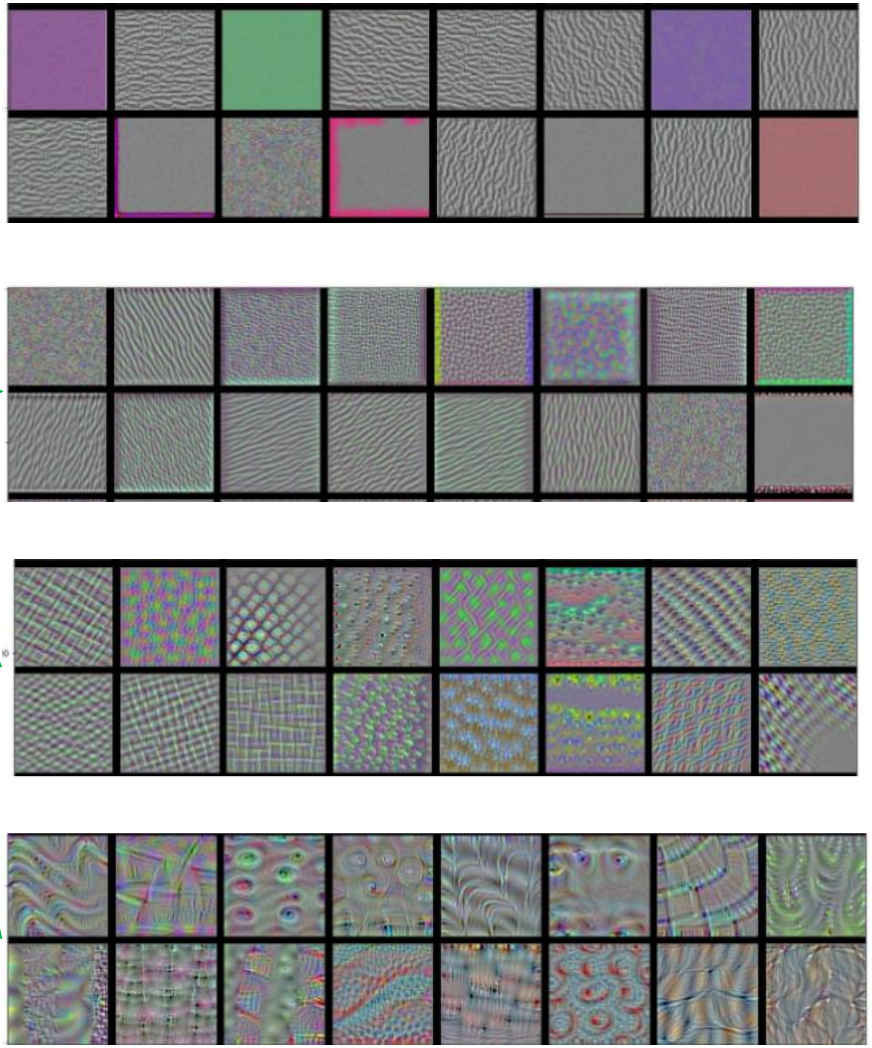
Until Now



VGG16

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 150, 150, 3)	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1180160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2359808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2359808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2359808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0

Filter



5. Use Pre-trained CNN + data augmentation + train last few layers of Pre-trained CNN

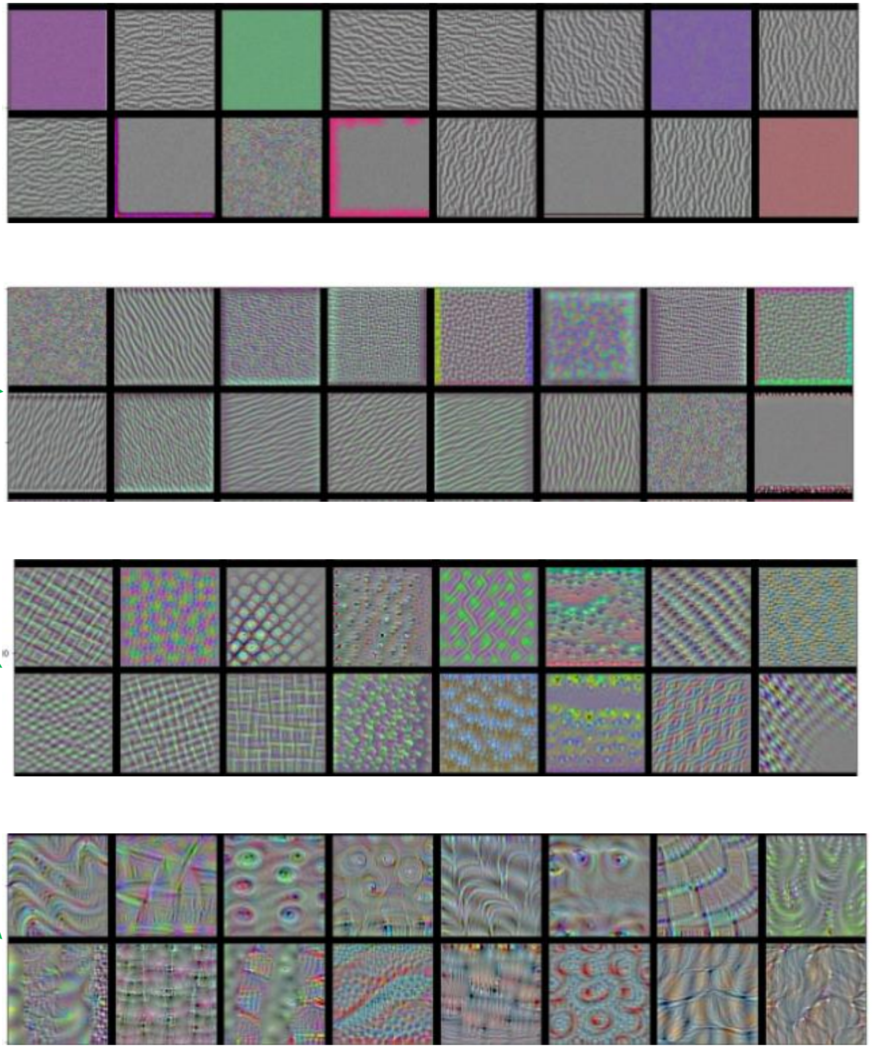
These filter visualizations tell us a lot about how convnet layers see the world: each layer in a convnet simply learns a collection of filters such that their inputs can be expressed as a combination of the filters. The filters in these convnet filter banks get increasingly complex and refined as we go higher-up in the model:

- The filters from the first layer in the model (block1_conv1) encode simple directional edges and colors (or colored edges in some cases).
- The filters from block2_conv1 encode simple textures made from combinations of edges and colors.
- The filters in higher-up layers start resembling textures found in natural images: feathers, eyes, leaves, etc

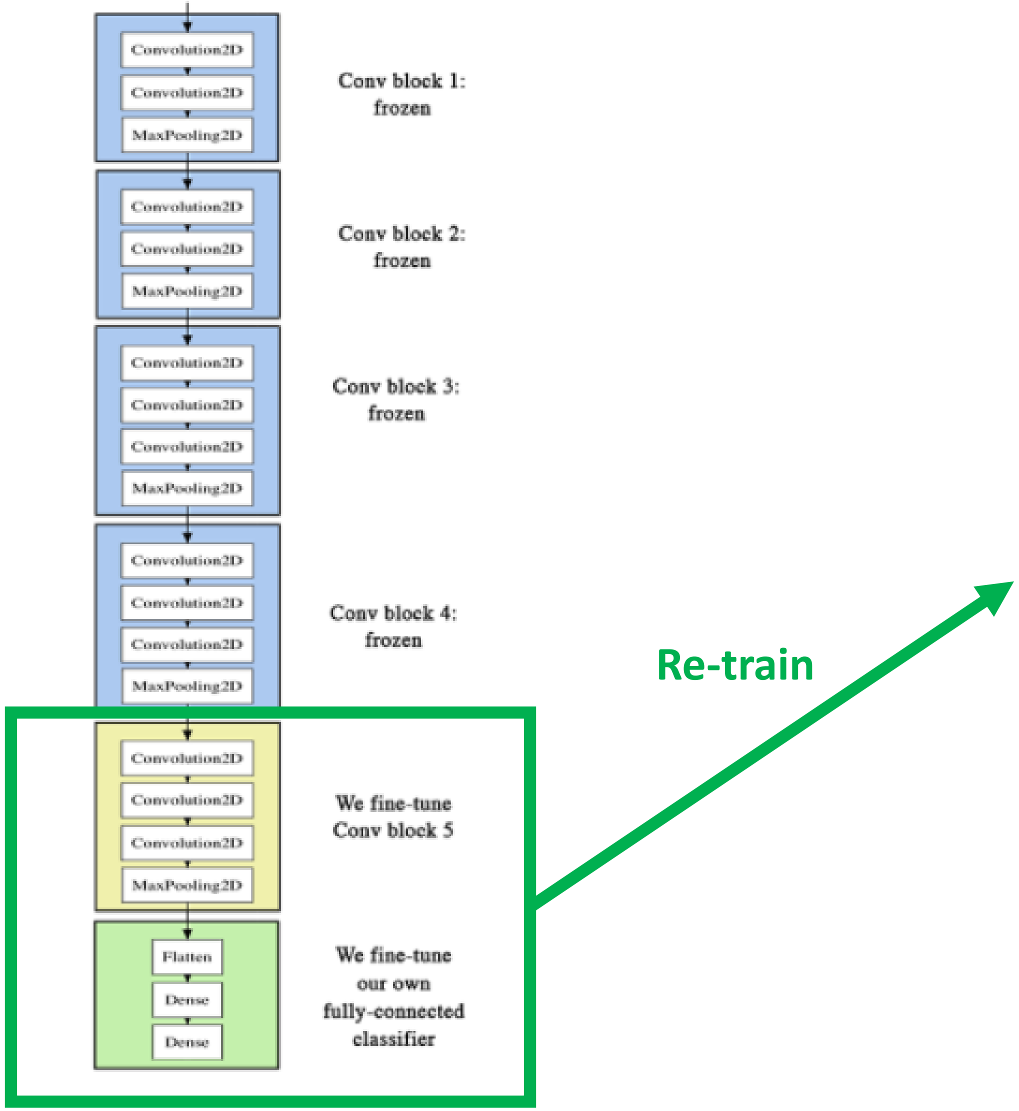
VGG16

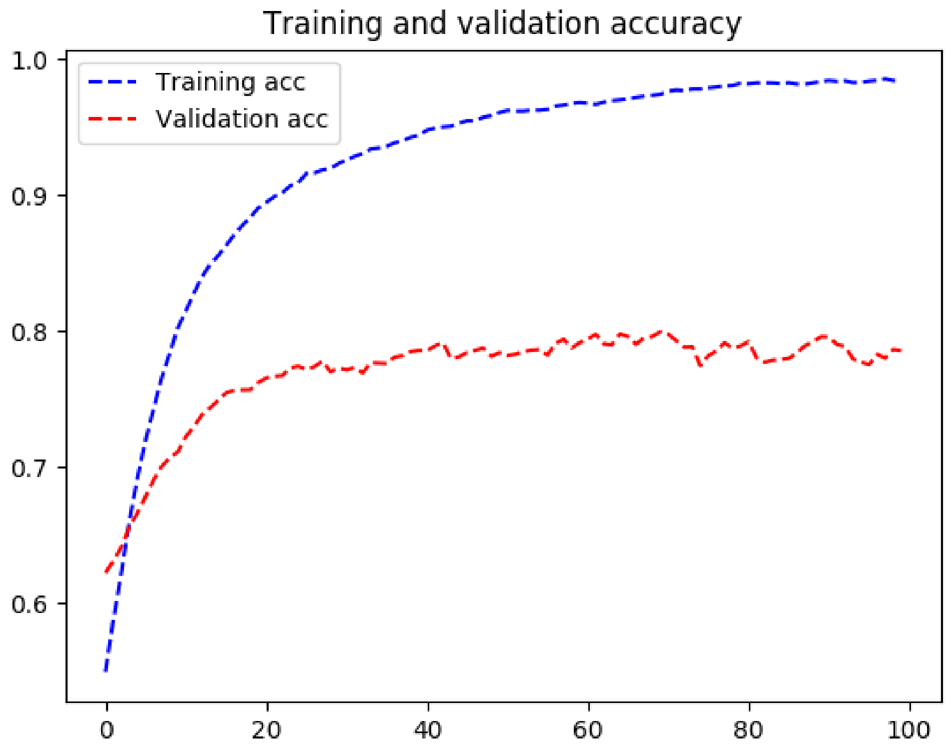
```
____________________________
Layer (type)
============================
input_1 (InputLayer)
____________________________
block1_conv1 (Conv2D)
____________________________
block1_conv2 (Conv2D)
____________________________
block1_pool (MaxPooling2D)
____________________________
block2_conv1 (Conv2D)
____________________________
block2_conv2 (Conv2D)
____________________________
block2_pool (MaxPooling2D)
____________________________
block3_conv1 (Conv2D)
____________________________
block3_conv2 (Conv2D)
____________________________
block3_conv3 (Conv2D)
____________________________
block3_pool (MaxPooling2D)
____________________________
block4_conv1 (Conv2D)
____________________________
block4_conv2 (Conv2D)
____________________________
block4_conv3 (Conv2D)
____________________________
block4_pool (MaxPooling2D)
____________________________
block5_conv1 (Conv2D)
____________________________
block5_conv2 (Conv2D)
____________________________
block5_conv3 (Conv2D)
____________________________
block5_pool (MaxPooling2D)
____________________________
```

Filter



5. Use Pre-trained CNN + data augmentation + train last few layers of Pre-trained CNN





Run time; 29 min 50 sec

6. Hyperparameter tuning

- How many **layers** should you stack? How many **units** or **filters** should go in each layer? Should you use ‘relu’ as **activation**, or a different function? Should you use **BatchNormalization** after a given layer? How much **dropout** should you use?....etc

- Mostly applicable for NN which are built from scratch.
- Have to look into tools available.

Things to do:

- Use our custom “trained classifier” to detect object(weed) from video feed.
- Check maximum FPS.

- Create weed dataset + Interlink video feed with ROS (Khunsa & Harsha).