1. Problem Statement :

Classification of Cat and Dog images using a convolution neural network.

2. Proposed Solution :

Design a Convolution neural network using keras to train and test models using kaggle Cat and Dog images to successfully classify them. We will be using VGG16 pretrained model for convolution to improve the performance. We employ transfer leatning to accomplish it. In the last step we perfrom Data Augmentation to help the model generalize better.

3. Implementation details :

A. Download Data, Create Directories and copy images, Splitting of data

- The data is split into 1600 images of Cat and Dog, 400 images of Cat and Dog for validation and 400 images for testing*

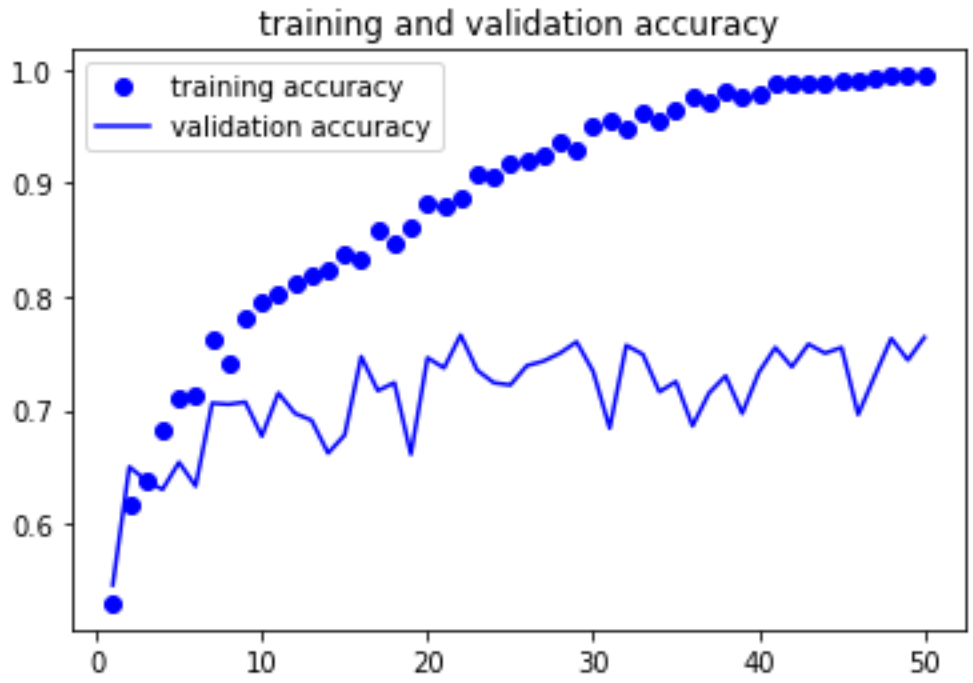
C. Create Covnet Model

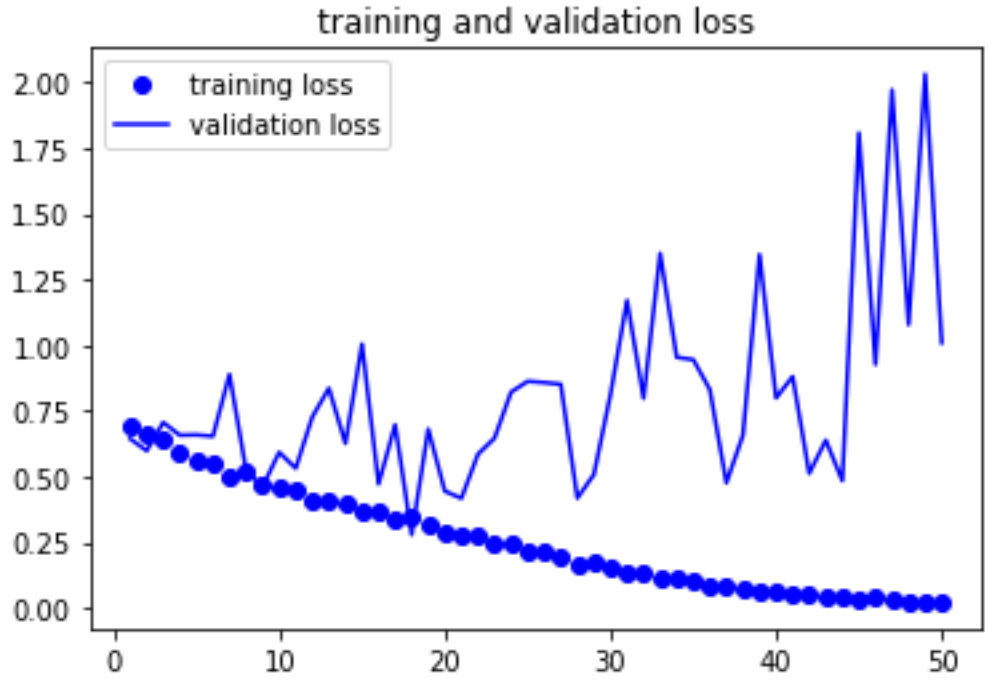
Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 74, 74, 32)	0
conv2d_2 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 36, 36, 64)	0
conv2d_3 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_3 (MaxPooling2	(None, 17, 17, 128)	0
conv2d_4 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_4 (MaxPooling2	(None, 7, 7, 128)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_1 (Dense)	(None, 512)	3211776
dense_2 (Dense)	(None, 1)	513

Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0

D. Perform Hyperparameter tuning





As you can after 18 epoch validation set starts to overfit so we choose epoch value as 18 and retrain the model with new hyperparamter (epoch = 18)

On Test set :
 Accuracy : 0.75

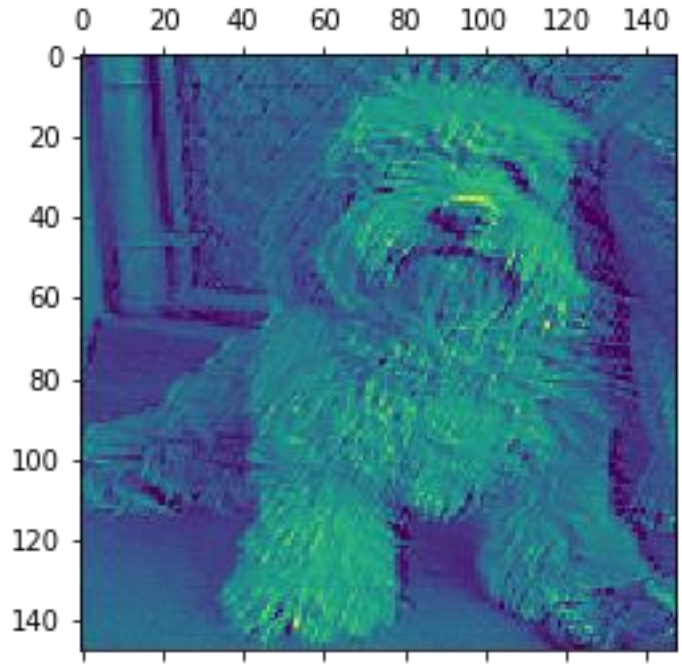
Loss : 0.4849088191986084

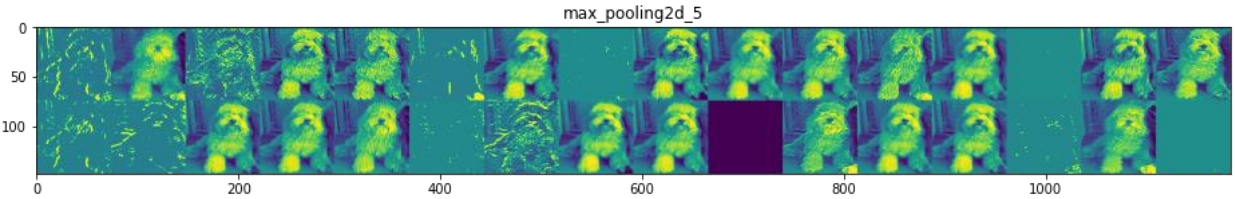
Visualization :

E. Activation Function Visulization :

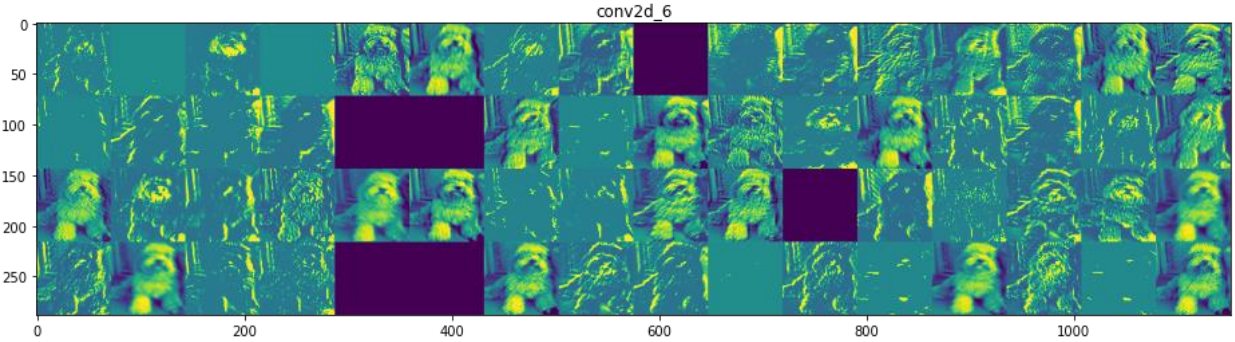
Images from the activations of convolution layers :

We are visualizing image of a dog when it is fed into different layers:

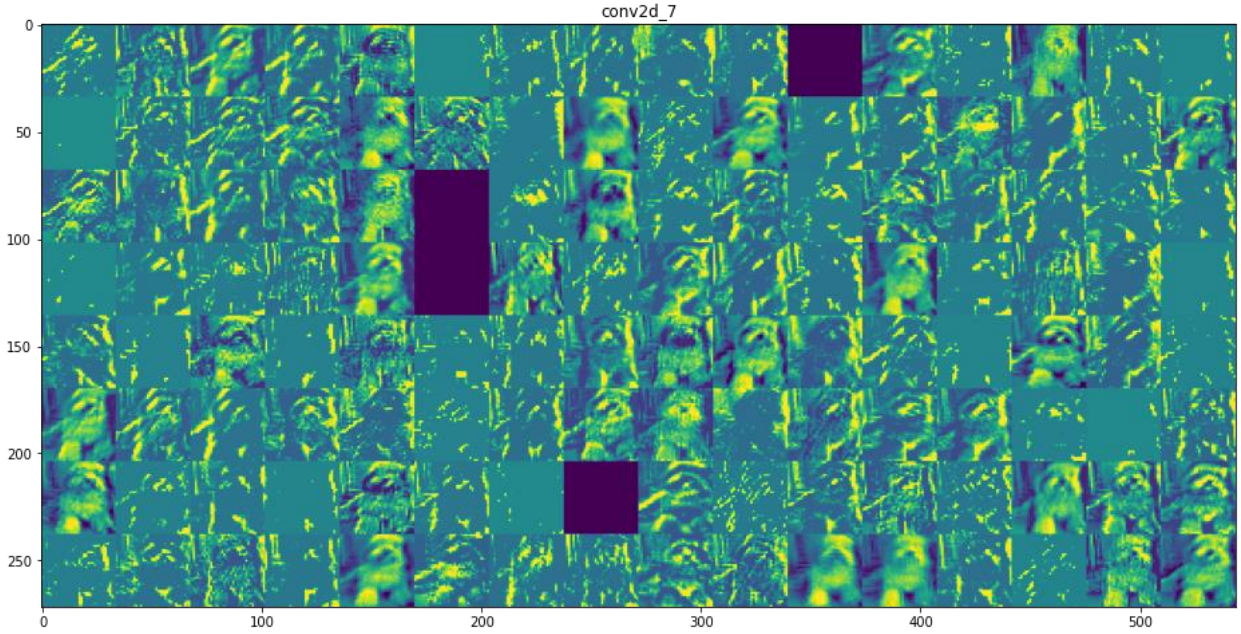




Below mentioned layer tries to detect vertical, horizontal edges.



Below mentioned layer tries to detect specialized features like eyes, ear

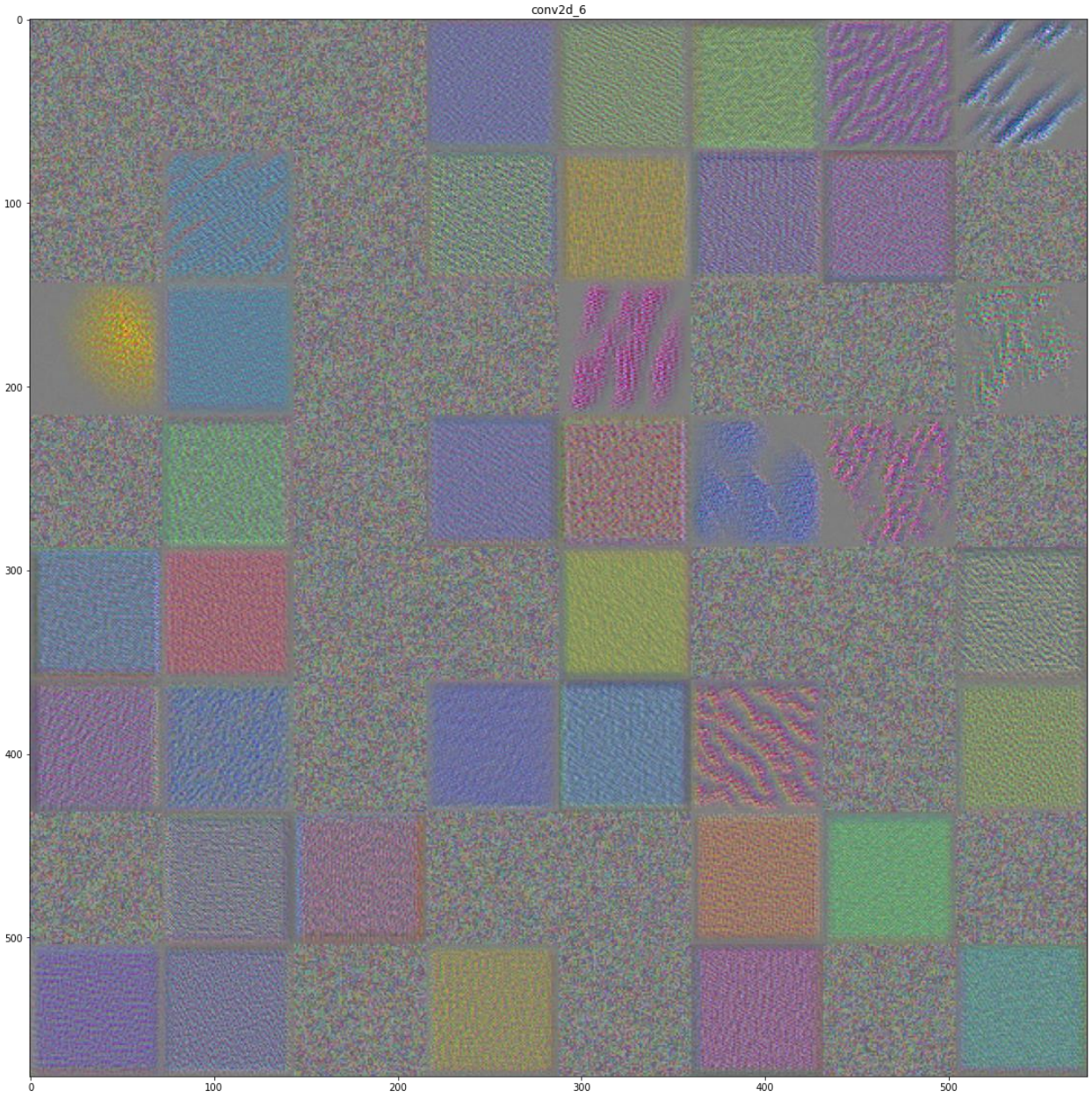


- Visualization of activation function makes us realise that initial layers are useful for basic feature extraction like edges and deeper layers are responsible for detecting specific features like eyes, ears.

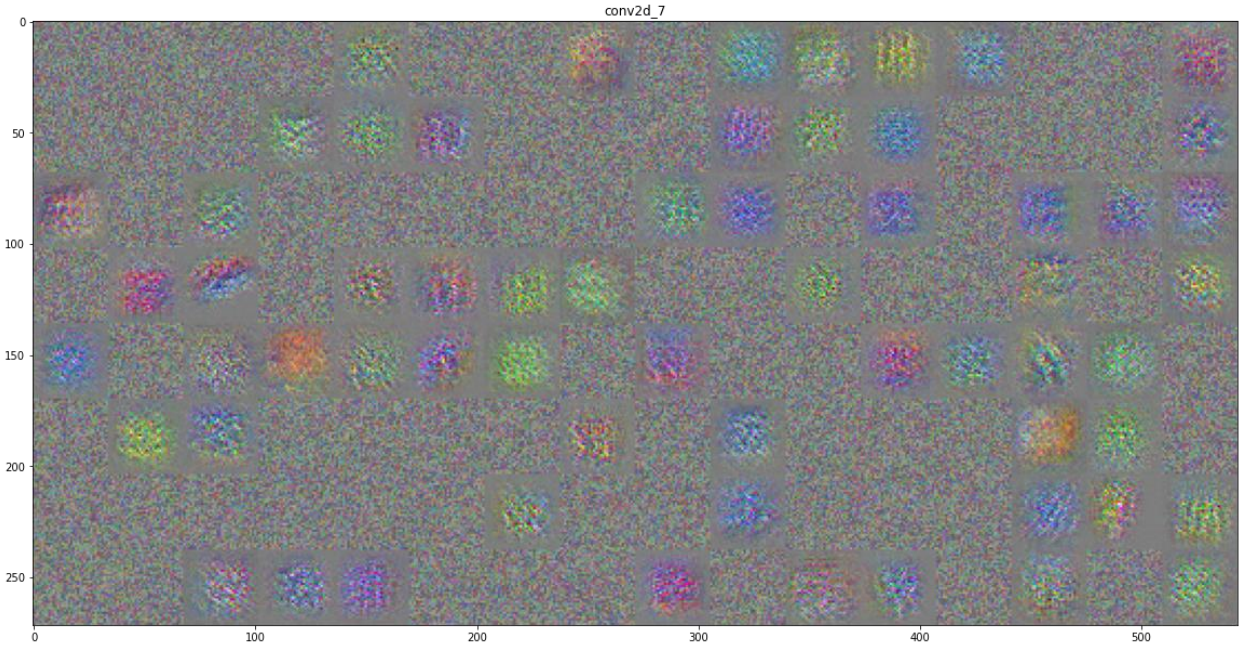
F. Filter Visulization :

Images from the Two layers of convolution :

- Below mentioned figure shows that this filter tries to detect patters like slant lines, dots.

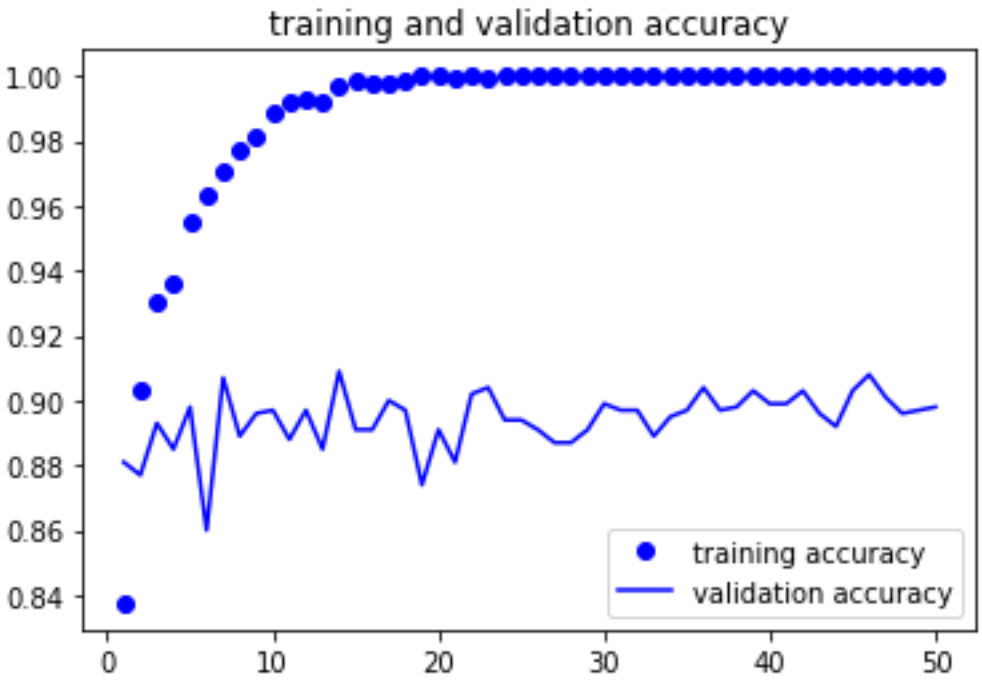


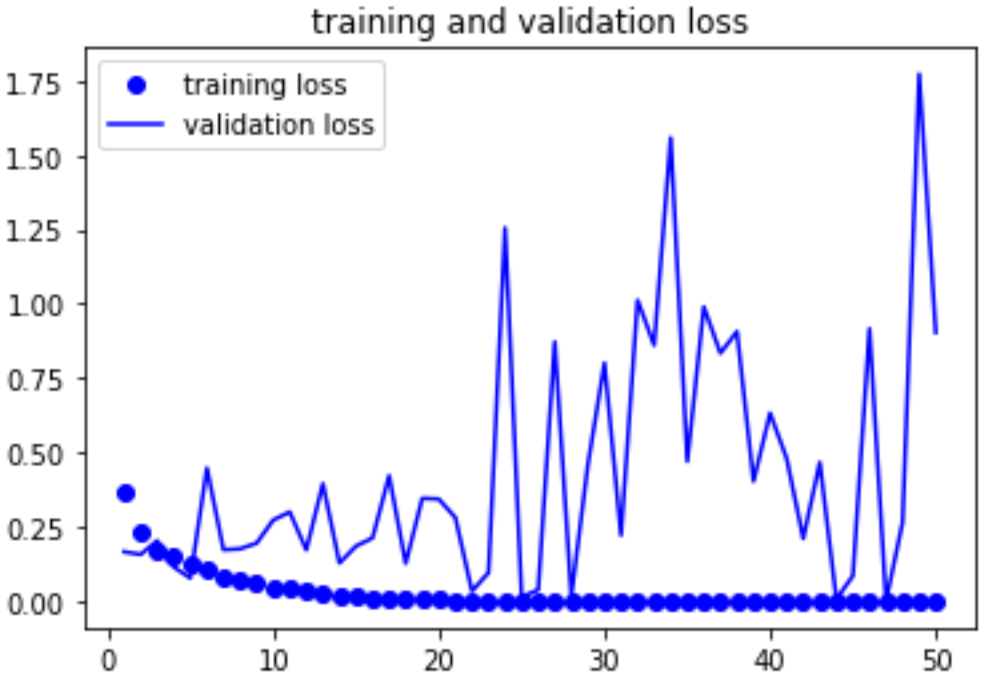
- In the below mentioned image, we see more complex combination of patterns from previous step. We can observe verical, horizontal, diagonal, dot patterns with different colors.



G. Use VGG16 Covnet and freeze and train

- We first freeze the VGG16 convolution base and train only the Fully connected network so the network gets trained in the direction of VGG16 first and to make sure that the existing weights of the VGG16 model is not lost.
- Model Architecture :
- Model: "sequential_1"
- _____
- Layer (type) Output Shape Param #
- ===
- vgg16 (Model) (None, 4, 4, 512) 14714688
- _____
- flatten_1 (Flatten) (None, 8192) 0
- _____
- dense_1 (Dense) (None, 256) 2097408
- _____
- dense_2 (Dense) (None, 1) 257
- ===
- Total params: 16,812,353
- Trainable params: 16,812,353
- Non-trainable params: 0





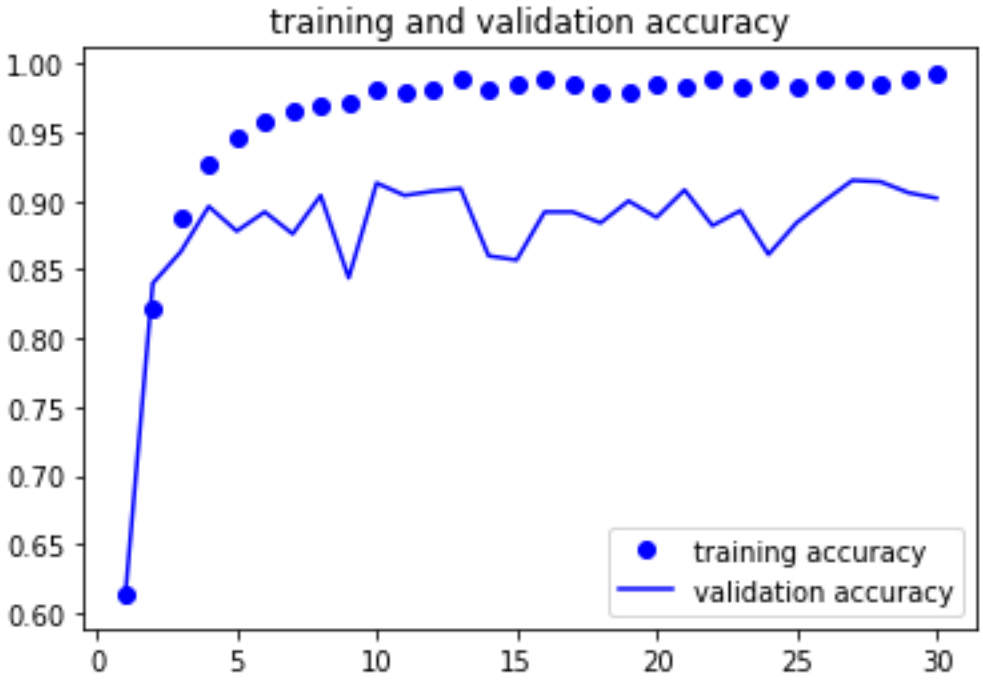
As you can after 5 epoch validation set starts to overfit so we choose epoch value as 5 and retrain the model with new hyperparamter (epoch = 5)

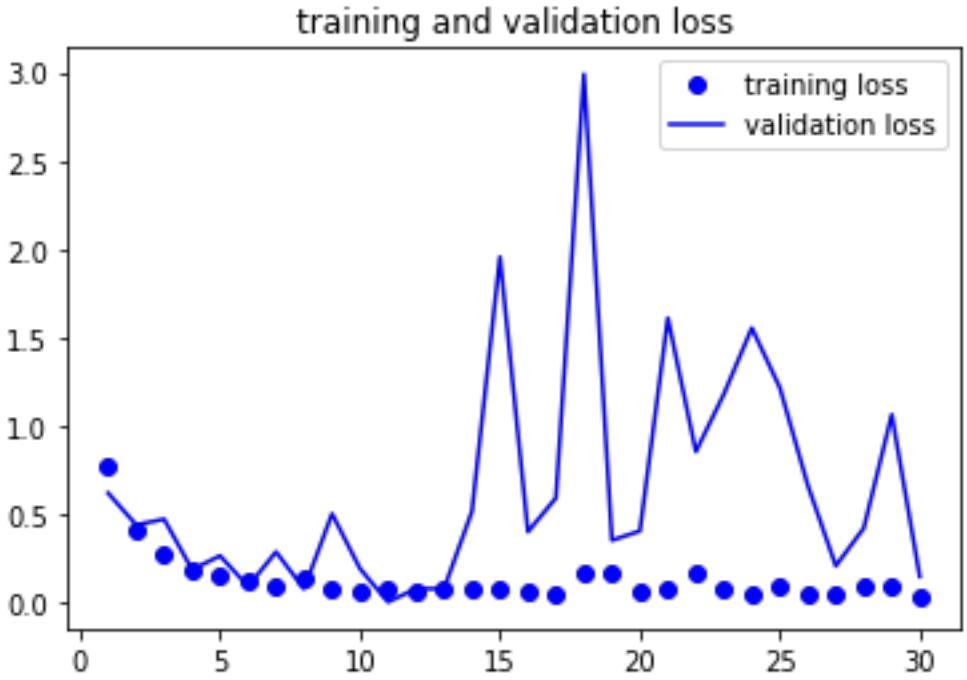
```
On Test set :
Accuracy : 0.9049999713897705
Loss : 0.3842967450618744
```

- We observe that training with VGG16 frozen performs better than our custom network.

H. Unfreeze VGG16 Covnet and continue to train

- We set the trainable field to True so that the whole model is updated now to move in the direction of the output





As you can after 12 epoch validation set starts to overfit so we choose epoch value as 12 and retrain the model with new hyperparamter (epoch = 12)

- Test Set Performance:
  Accuracy : 0.9225000143051147
  Loss : 0.5954384207725525

As you can see network with unfrozen VGG16 improved performance from frozen VGG16 (from 90 to 92)

8. Perform Data Augmentation with VGG16 frozen and train with augmented data

Generalization of Model

I. Data Augmentation

We set the trainable field back to False. We then retrain the model with Augmented data so that the model can learn to generalize better as in the previous step, now we want the model to generalize better, So we perform Data Augmentation by modifiying different attributes of the training image so that it resembles real world chaos.

Train data augmentation is performed with following attributes :

```
rescale= 1./255,
  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'
```

We see that there is an increase in the validation accuracy.

- Test performance:
  Accuracy :0.9275000095367432
  Loss : 0.17063912749290466

As compared to network with Frozen layer (VGG16, giving 90% accuracy), our network which used augmented data and frozen VGG16 layer improved the performance (from 90% to 92%)