CS-671

**Deep Learning and its Applications**

# Assignment 2
# Task 3
## Network Visualization

*submitted by*

**Team 5**

**Vishal Anand B16040**

**Aman Jain B16044**

**Yash Agrawal B16120**

SCHOOL OF COMPUTING AND ELECTRICAL ENGINEERING

**INDIAN INSTITUTE OF TECHNOLOGY MANDI, MANDI**

**March 2019**

## 1 Objective

The objective of this assignment is to learn a few ways of visualization of the outputs in a neural network. These visualizations will help in understanding the working of these networks and can help in debugging and optimizing the performance.

## 2 Model

The code for the complete model of the above task can be found at :
https://github.com/Vishal1541/DeepLearning/tree/master/Assignment2/Task3_Visualization
Filename : section1MNIST.ipynb, section1LINE.ipynb, multihead.ipynb

## 3 Visualizations

In task1 and task2 we have made architectures : sequential using MNIST and line data, multi-head using line data. Visualization of the network is done for all the three networks described above. Following three network visualization techniques are used on network architectures :

### 3.1 Part 1: Visualizing Intermediate Layer Activations

Intermediate layer activations are the outputs of intermediate layers of the neural network.Plotted the intermediate activations of the layers of the neural networks made in section 1,2 for atleast 6 images.

### 3.2 Part 2: Visualizing Convnet Filters

Visualize the filters in the convolutional neural network. This is done by running Gradient Descent on the value of a convnet so as to maximize the response of a specific filter, starting from a blank input imageand also plotted the filters of the layers of the neural networks made in section 1,2.

### 3.3 Part 3: Visualizing Heatmaps of class activations

Plotted heatmaps of class activations over input images (line dataset created in assignment-1). This will help in visualizing the regions of the input image the convnet is looking at. A class activation heatmap is a 2D grid of scores associated with a particular output class, computed for every location for an input image, indicating how important is each location is with respect to that output class. You are required to plot the heatmaps for atleast 6 images for networks made in section 1,2.

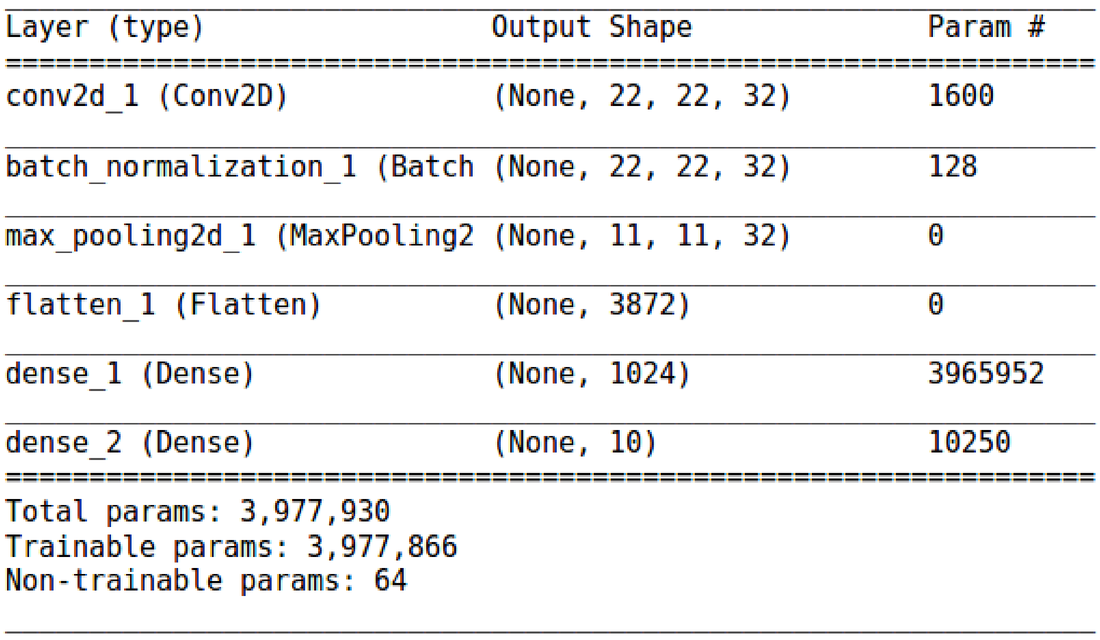
## 4 Network Architecture Used

The visualization techniques described above are used on the following two network architectures :

### 4.1 Sequential Model

The sequential model used is in this visualization is the same network implemented in task 1. Below is the architecture used:

1. 7x7 Convolutional Layer with 32 filters and stride of 1.
2. ReLU Activation Layer.
3. Batch Normalization Layer
4. 2x2 Max Pooling layer with a stride of 2
5. Fully Connected layer with 1024 output units.
6. ReLU Activation Layer.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 22, 22, 32)        1600
_________________________________________________________________
batch_normalization_1 (Batch (None, 22, 22, 32)        128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 11, 11, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3872)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              3965952
_________________________________________________________________
dense_2 (Dense)              (None, 10)                10250
=================================================================
Total params: 3,977,930
Trainable params: 3,977,866
Non-trainable params: 64
_________________________________________________________________
```

### 4.2 Non-Sequential Model (Multihead)

The network architecture is divided in two parts a) Feature network and b) Classification heads. The feature network is extracting the required features from the input and the 4 classification heads attached to it are classifying four types of variations namely length, width, color, angle.

The complete architecture is implemented using Keras functional API because we require multiple outputs from single input.

### 4.2.1 Feature Network

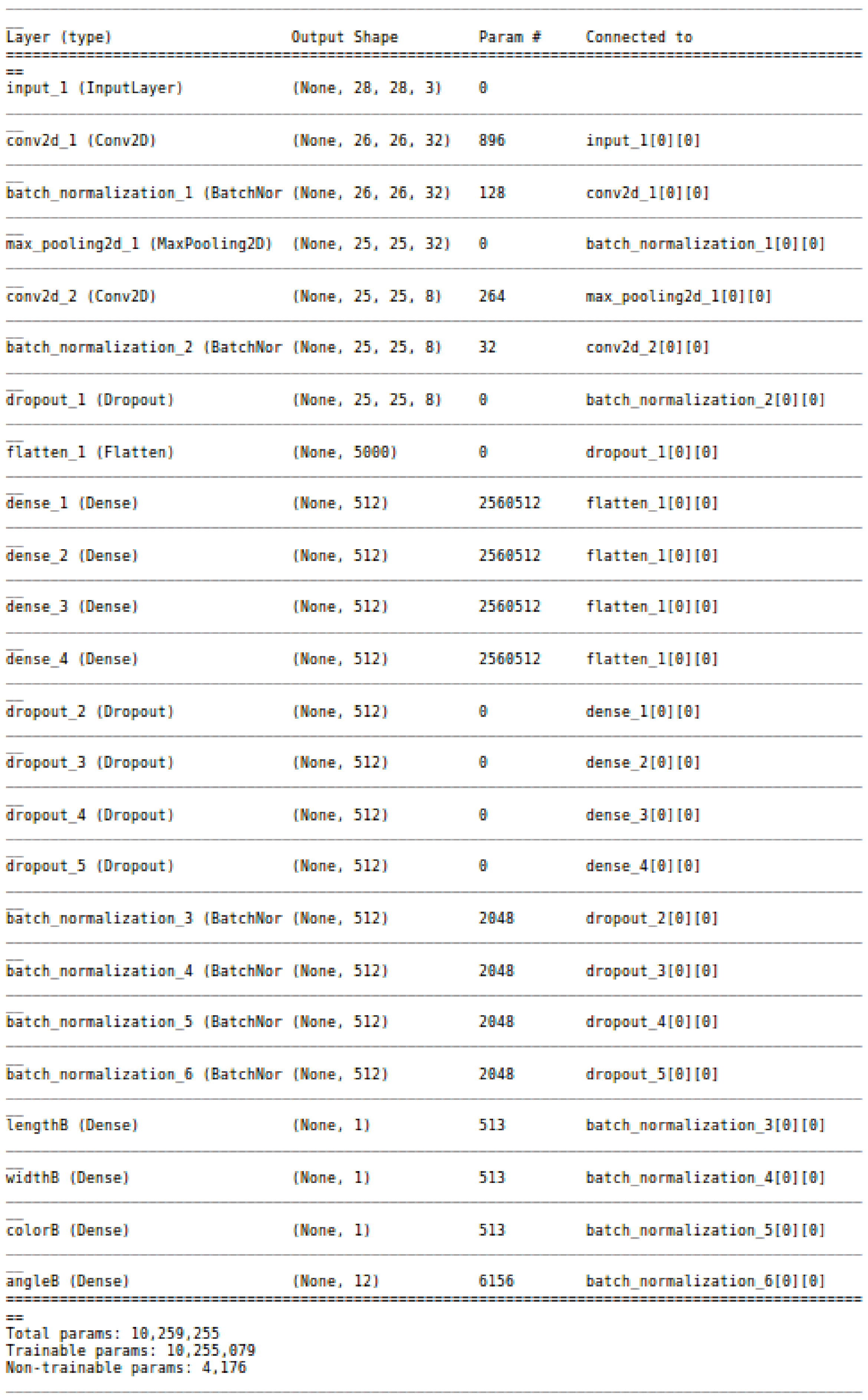
Model with the following architecture was used to extract the features from the input :

1. 3x3 Convolutional Layer with 32 filters and stride of 1.
2. ReLU Activation Layer.
3. Batch Normalization Layer
4. 2x2 Max Pooling layer with a stride of 2
5. 1x1 Convolutional Layer with 8 filters and stride of 1.
6. ReLU Activation Layer.
7. Batch Normalization Layer.
8. Dropout Layer with a rate of 0.4
9. Layer to flatten the features

### 4.2.2 Classification heads

Model with the following architecture was used for classification heads :

1. Fully Connected layer with 512 output units.
2. ReLU Activation Layer.
3. Dropout Layer with a rate of 0.4
4. Batch Normalization Layer
5. Fully Connected layer with 1,1,1,12 output units respectively for length, width, color and angle classifications.
6. Sigmoid, sigmoid, sigmoid, softmax activation functions respectively for length, width, color and angle classifications.

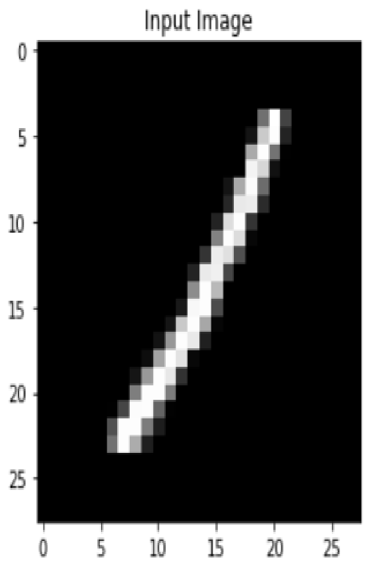


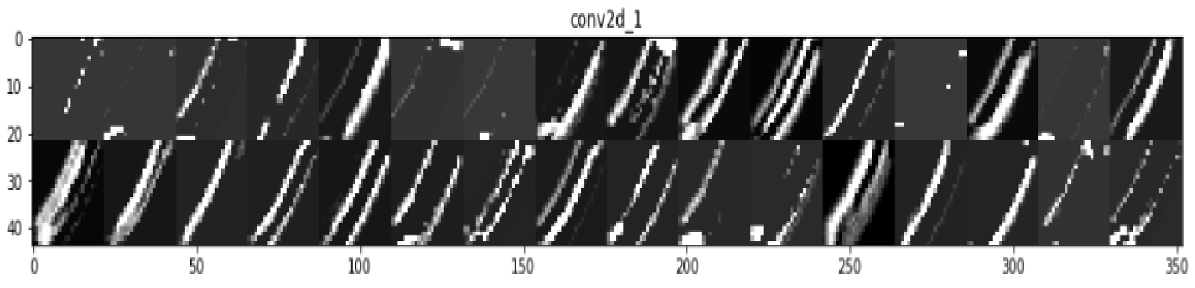
```
________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================================
==
input_1 (InputLayer)            (None, 28, 28, 3)    0
________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 26, 26, 32)   896         input_1[0][0]
________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 26, 26, 32)   128         conv2d_1[0][0]
________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 25, 25, 32)   0           batch_normalization_1[0][0]
________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 25, 25, 8)    264         max_pooling2d_1[0][0]
________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 25, 25, 8)    32          conv2d_2[0][0]
________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 25, 25, 8)    0           batch_normalization_2[0][0]
________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 5000)         0           dropout_1[0][0]
________________________________________________________________________________________________
dense_1 (Dense)                 (None, 512)          2560512     flatten_1[0][0]
________________________________________________________________________________________________
dense_2 (Dense)                 (None, 512)          2560512     flatten_1[0][0]
________________________________________________________________________________________________
dense_3 (Dense)                 (None, 512)          2560512     flatten_1[0][0]
________________________________________________________________________________________________
dense_4 (Dense)                 (None, 512)          2560512     flatten_1[0][0]
________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 512)          0           dense_1[0][0]
________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 512)          0           dense_2[0][0]
________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 512)          0           dense_3[0][0]
________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 512)          0           dense_4[0][0]
________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 512)          2048        dropout_2[0][0]
________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 512)          2048        dropout_3[0][0]
________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 512)          2048        dropout_4[0][0]
________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 512)          2048        dropout_5[0][0]
________________________________________________________________________________________________
lengthB (Dense)                 (None, 1)            513         batch_normalization_3[0][0]
________________________________________________________________________________________________
widthB (Dense)                  (None, 1)            513         batch_normalization_4[0][0]
________________________________________________________________________________________________
colorB (Dense)                  (None, 1)            513         batch_normalization_5[0][0]
________________________________________________________________________________________________
angleB (Dense)                  (None, 12)           6156        batch_normalization_6[0][0]
================================================================================================
==
Total params: 10,259,255
Trainable params: 10,255,079
Non-trainable params: 4,176
________________________________________________________________________________________________
```

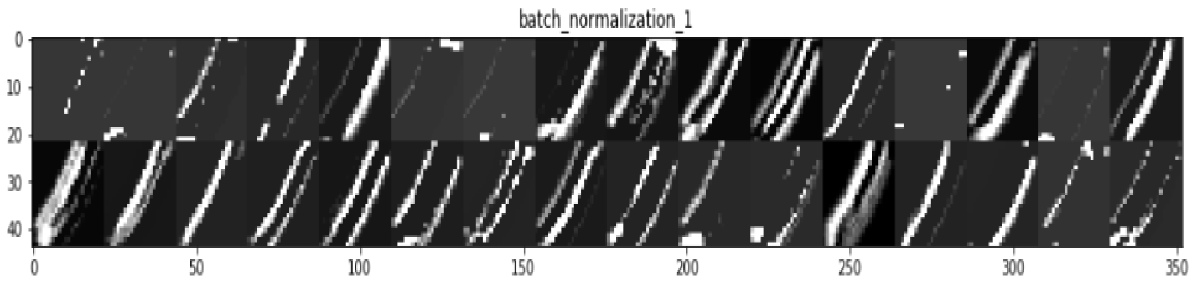
## 5 Results

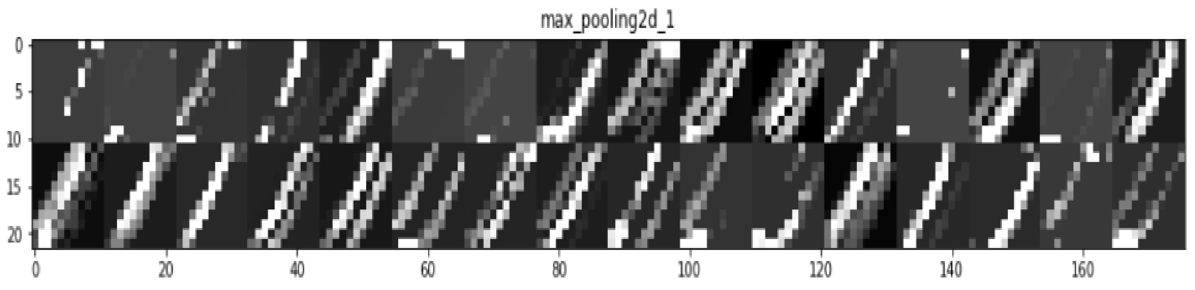
### 5.1 Section-1 : Sequential Model using MNIST Dataset

- Visualizing Intermediate Layer Activations

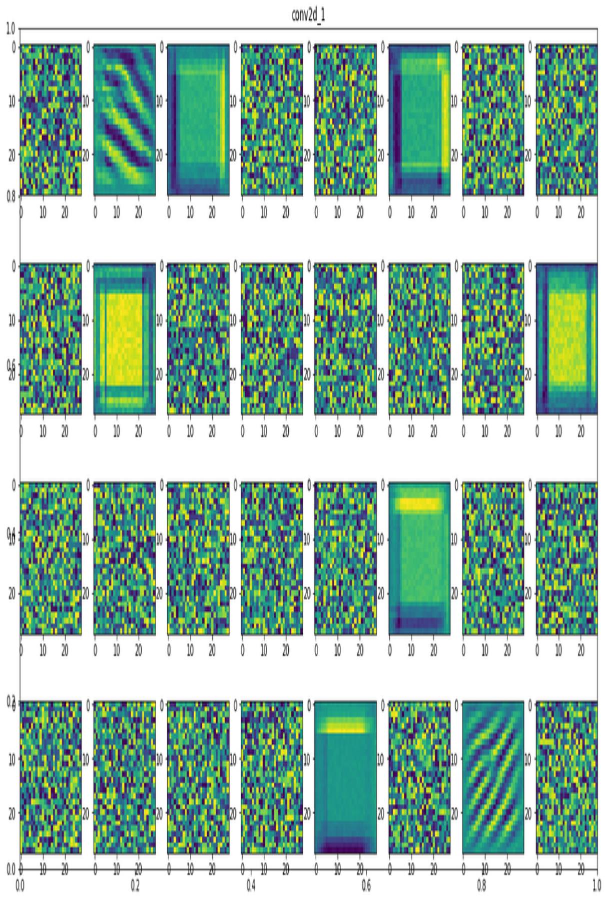


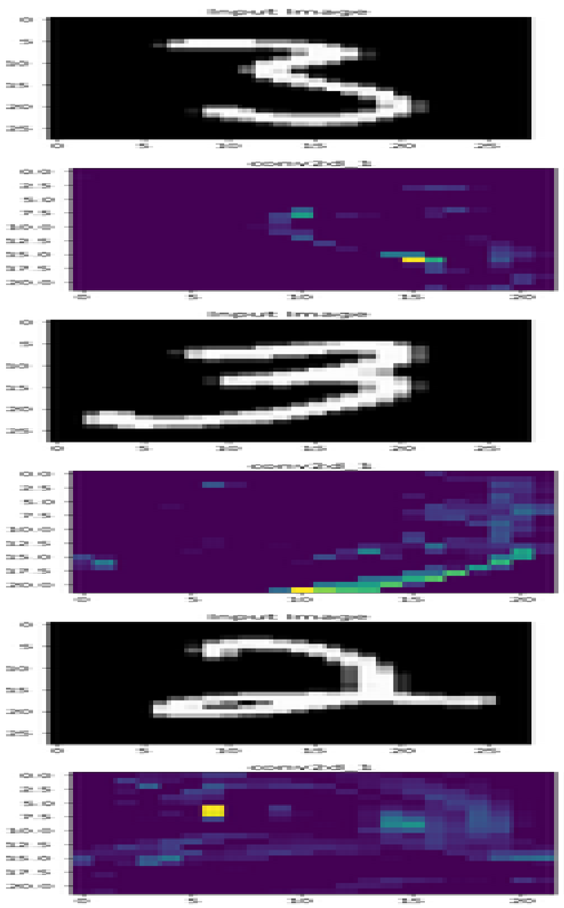






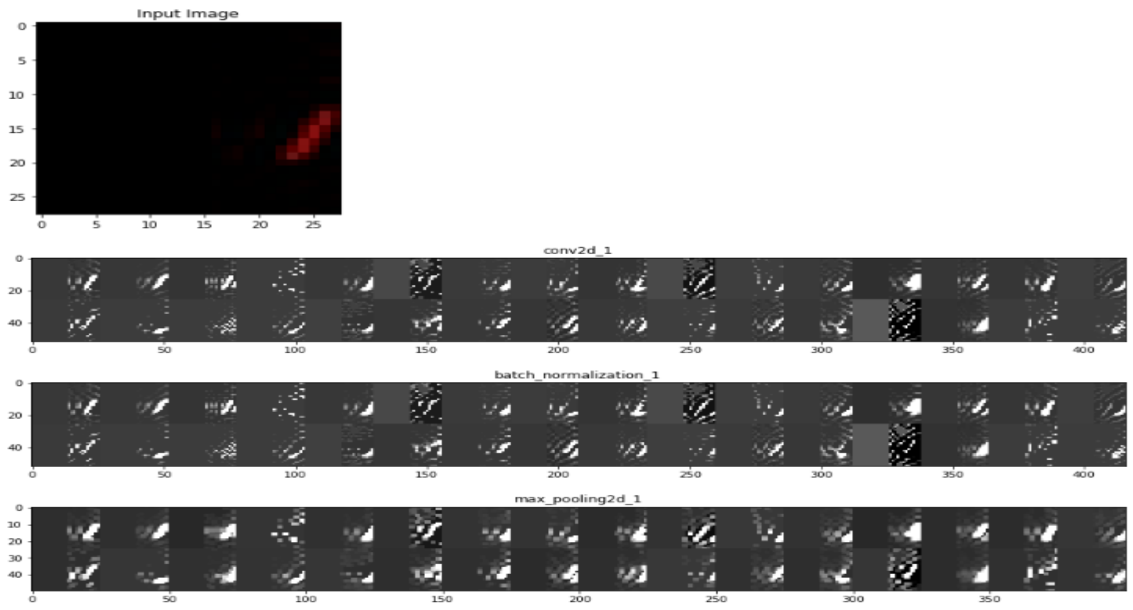
- Visualizing Convnet Filters and Heatmaps of class activations



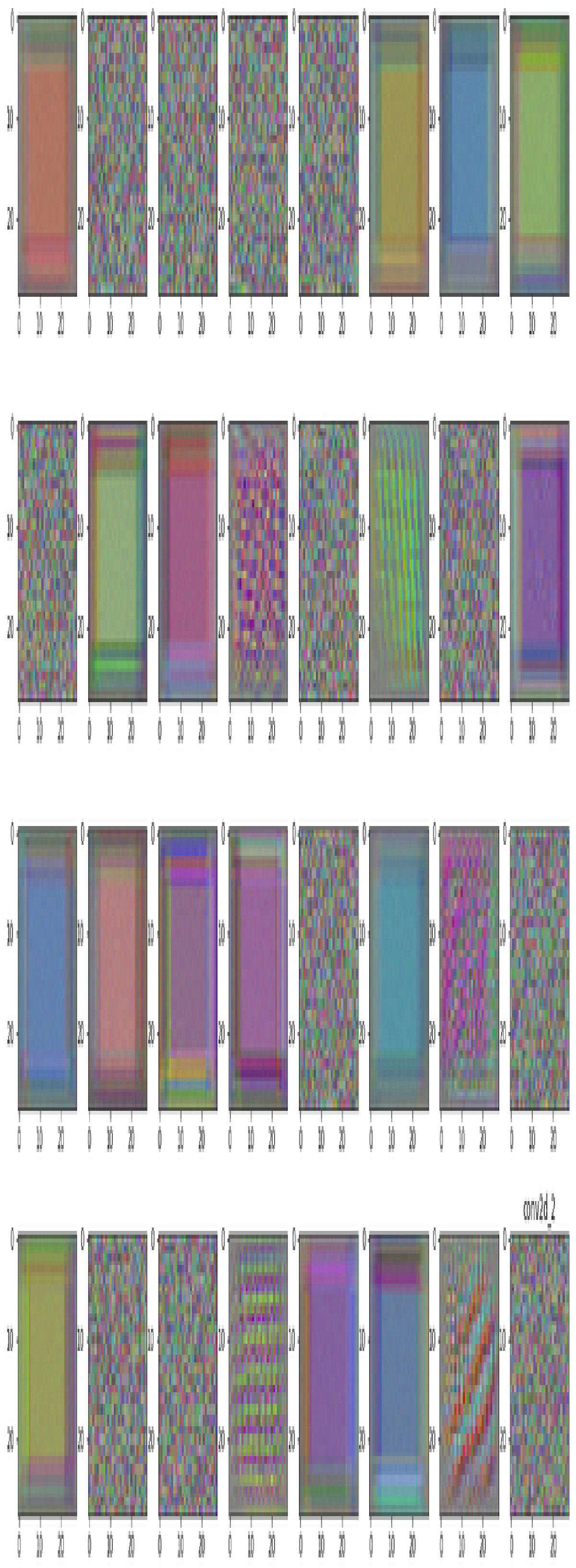


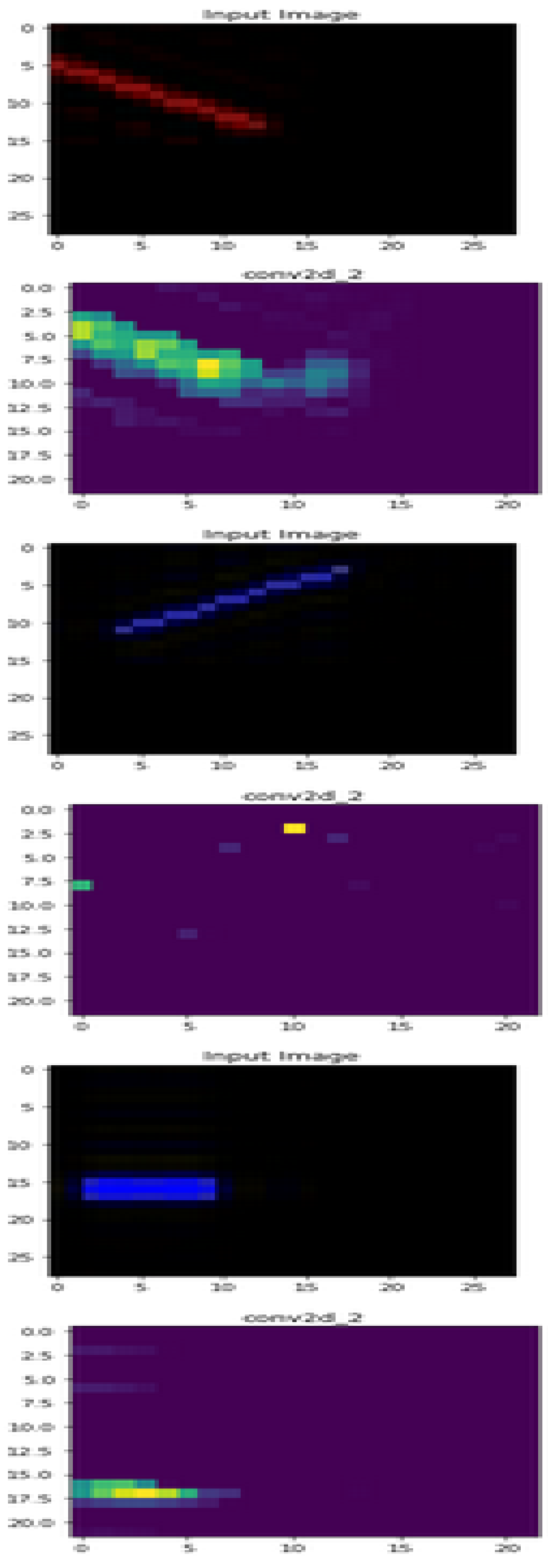
## 5.2 Section-1 : Sequential Model using LINE Dataset

- Visualizing Intermediate Layer Activations



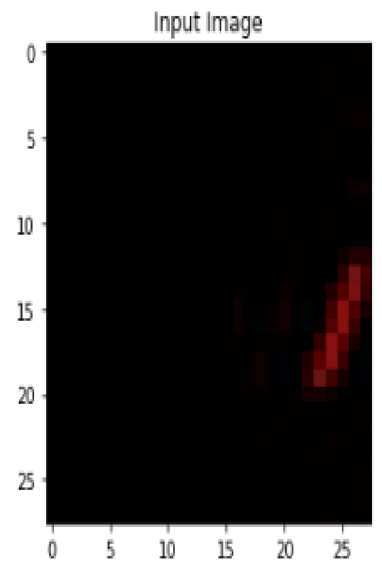
- Visualizing Convnet Filters and Heatmaps of class activations

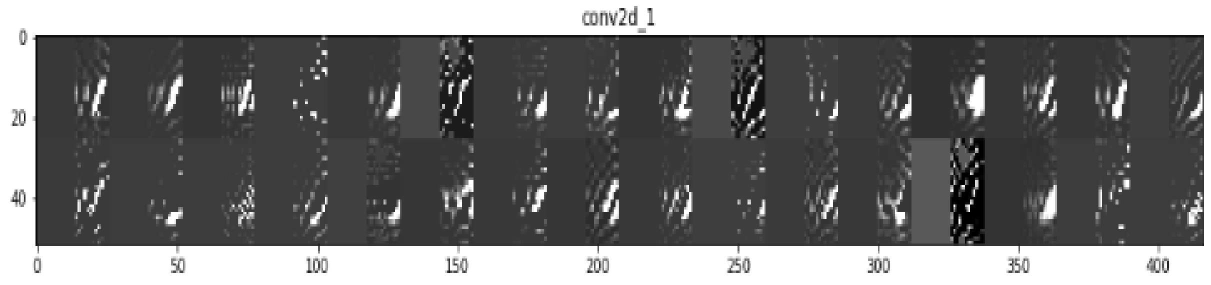


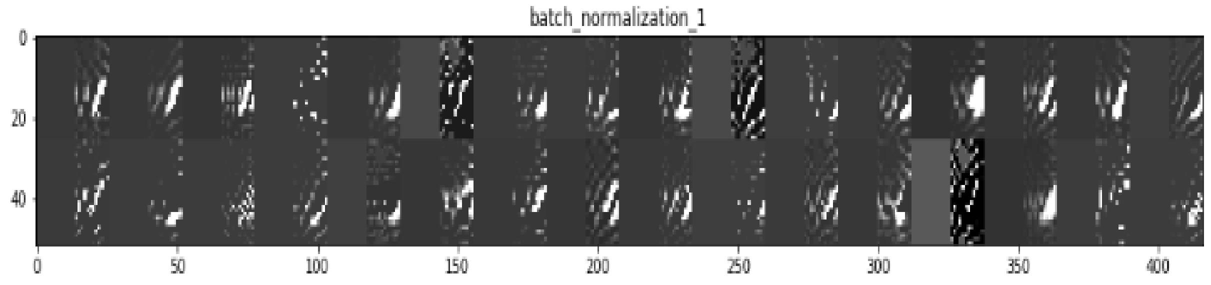


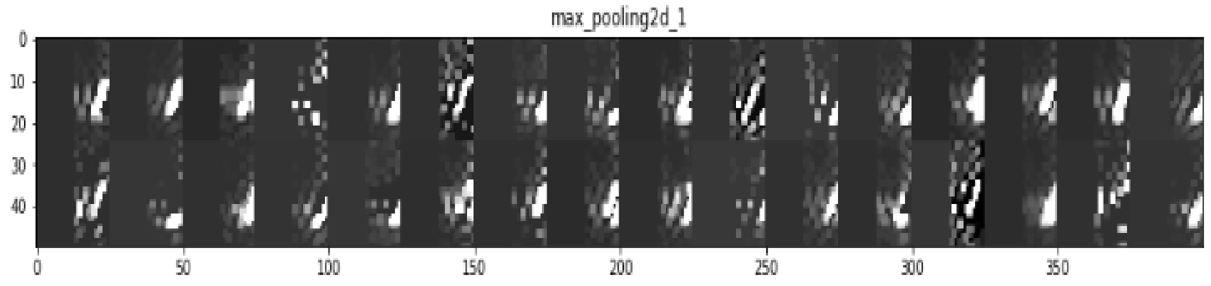
### 5.3 Section-2 : Non-Sequential Model(Multihead) using LINE Dataset

- Visualizing Intermediate Layer Activations

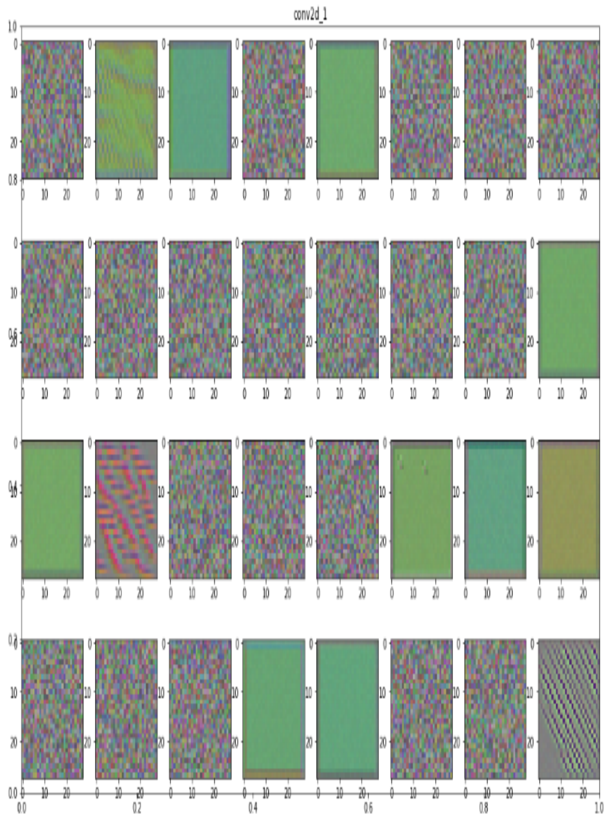


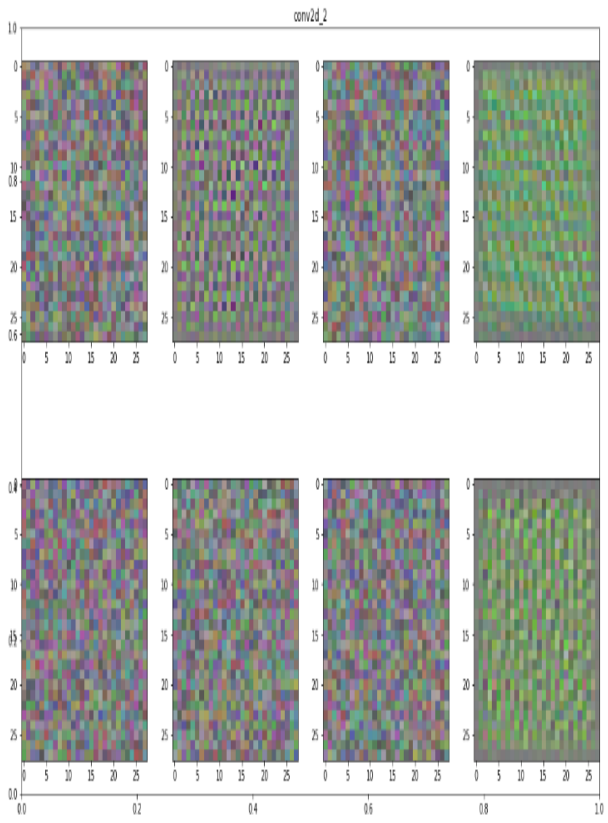


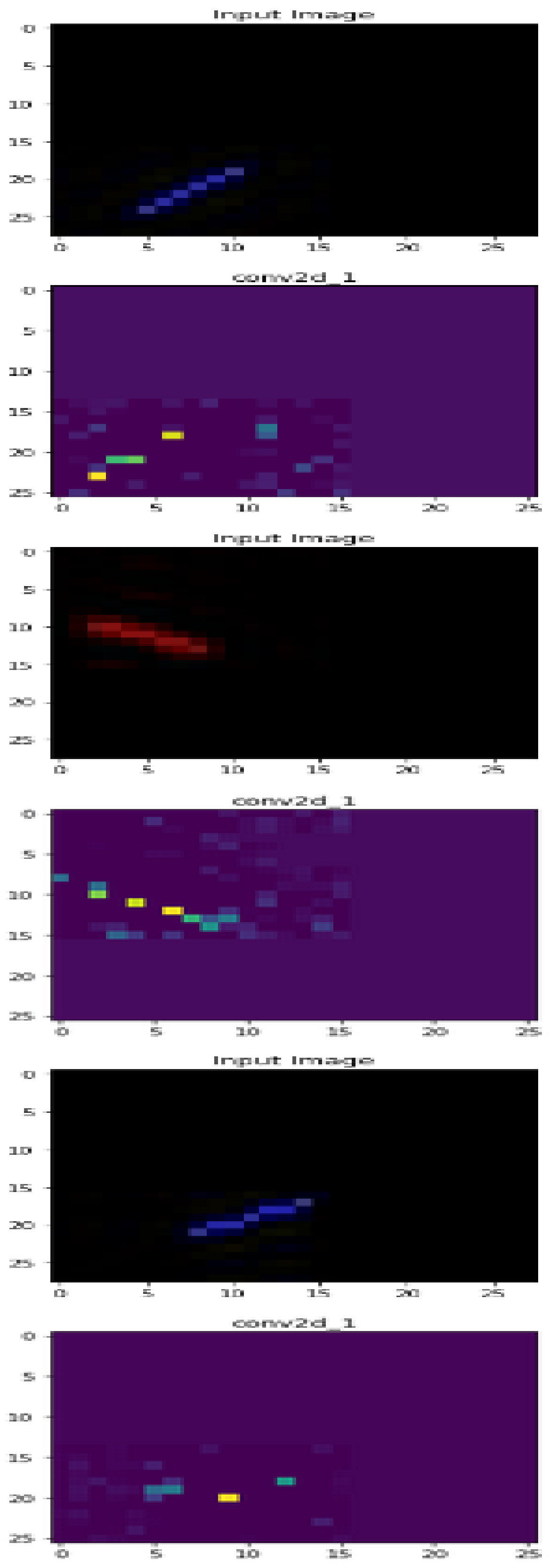




- Visualizing Convnet Filters and Heatmaps of class activations







## 6 Inferences

- The complex models such as multiple inputs, multiple outputs, shared layers, etc can be implemented using Keras functional API.
- The training phase of the model greatly depends on the architecture used for each of the feature network as well as the classification heads.
- visualizing the output of each layer is a good way to understand what exactly the neurons in our network are learning.
- Different types of visualizations are used which shows the progress of the network through output of each layer.
- As seen from the images, each filter in each layer will highlight or activate different parts of the image.
- Some filters are acting as edge detectors, others are detecting a particular region of the flower like its central portion and still others are acting as background detectors.
- From the output of the visualizing intermediate layer activations, we can notice that layers which are deeper in the network will learn more specific features which the layers in the start of the network will tend to visualize general patterns like edges, circles, shapes, colors, boundaries etc.
- Each filter will respond to particular type of pattern, we can show this pattern by displaying the network layer filters.
- To visualize network layer filters, we can use gradient descent in the value of convnet so as to maximize the response of specific filter starting from a n input image.
- The patterns found in starting layers are more generic somewhat similar to lines, edges and so on. From this we can conclude that earlier layers are going to learn more generic features while the deep layers will learn more specific features.
- As we move deeper into the network, the pattern are going to be more complex this is the reason for having blank outputs in deeper layers in previous section as the neuron will not learn that particular filter pattern which is more specific to the output i.e. image does not have the pattern/information which the filter was interested in.
- To be more specific to the visualization heatmaps are used which shows the regions in the input image convnet is looking for. This is technique can be used to check learning of the neurons as sometimes it may happen the network could not classify the image correctly but looking for the class in the correct parts of the image.