Indian Institute of Technology Mandi

CS-671 Deep Learning and Applications

Assignment 5

Convolutional Neural Networks

Submitted by:
Group 13
Prachi Sharma (V21078)
Ayush Tiwari (T22053)

Instructor:
Prof. A D Dileep

May 1, 2023

Table of Contents

Convolutional Neural Networks

1. The convolution operation

The convolutional neural networks, abbreviated as CNNs, are a special type of neural network used to perform learning on data that has a grid-like structure. A CNN uses a *convolution* operation in at least one of its layers. A convolution operation on a 1-D input is defined as:

$$s_t = \sum_{a=0}^{\infty} x_{t-a} w_a = (\mathbf{x} * \mathbf{w})_t$$

Here, $\mathbf{x}$ is the **input**, $\mathbf{w}$ is the **kernel** or filter and t is the time step. The output **s** is called the feature map.

2. Structure of a CNN

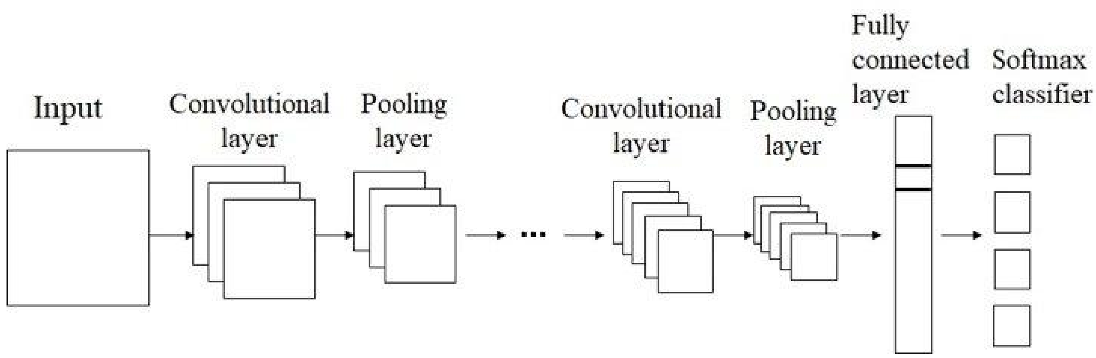


Fig 1: Structure of a basic CNN

A typical CNN has a structure as shown in Fig 1. The input is sent to a **convolutional layer**, which has a K number of filters of size F. A convolution operation is performed by all the filters on the input and the output is then sent to a pooling layer. A **pooling layer** modifies the outputs of the previous layer by operating it with a pooling function. Two such functions are *maxpool* and *average pool.* A max pool function reports the maximum output within a rectangular output while an average pool takes the average of all values in a rectangular output. After several consecutive convolution and pooling layers, the network is connected to a dense layer of neuron units, called **dense layers**, and finally a fully-connected **output layer**.

3. Backpropagation in CNNs

The connections in a convolution neural network are much sparser and weights are shared to make learning easier. CNNs can be trained just like a feed-forward neural network by backpropagating the error. Only the active parameter weights are updated at each instance of backpropagation.

Experiments, Results and Analysis

In this section, we will present our results and observations on image classification using different CNN architectures. In the first part, we train three different CNNs to classify a subset of the Caltech-101 dataset, which consists of color images. Different performance measures have been observed and compared of all three networks to conclude the best performing network. In the second part, we make use of the pretrained VGG19 model available in Tensorflow and perform transfer learning to classify the given image dataset.

1. Classification using different CNNs

We have been given 5 classes from 101 classes of the Caltech-101 image dataset. Each class has 80 images belonging to classes: **brain, buddha, ketch, Leopards, scorpion**. The train, validation and test split among all classes is 5:1:2. To ensure same input dimension, the images have been resized to 224×224.

Following architectures have been considered for the assigned task.

Model	Input layer	Convolution layer 1	Pooling layer 1	Convolution layer 2	Pooling layer 2	Convolution layer 3	Pooling layer 3	Convolution layer 4	Pooling layer 4	Flatten layer	Dense layer	Output layer
1	$28 \times 28 \times 3$	$F = 11$ $K = 8$ $S = 4$ $P = 0$	$F = 3$ $K = 8$ $S = 2$ $P = 0$	$F = 5$ $K = 16$ $S = 1$ $P = 0$	$F = 3$ $K = 16$ $S = 2$ $P = 0$	-	-	-	-		128	5
2	$28 \times 28 \times 3$	$F = 11$ $K = 8$ $S = 4$ $P = 0$	$F = 3$ $K = 8$ $S = 2$ $P = 0$	$F = 5$ $K = 16$ $S = 1$ $P = 0$	$F = 3$ $K = 16$ $S = 2$ $P = 0$	$F = 3$ $K = 32$ $S = 1$ $P = 0$	$F = 3$ $K = 32$ $S = 2$ $P = 0$	-	-		128	5
3	$28 \times 28 \times 3$	$F = 11$ $K = 8$ $S = 4$ $P = 0$	$F = 3$ $K = 8$ $S = 2$ $P = 0$	$F = 5$ $K = 16$ $S = 1$ $P = 0$	$F = 3$ $K = 16$ $S = 2$ $P = 0$	$F = 3$ $K = 32$ $S = 1$ $P = 0$	-	$F = 3$ $K = 64$ $S = 1$ $P = 0$	$F = 3$ $K = 64$ $S = 2$ $P = 0$		128	5

The output layer has 5 hidden units representing five classes of given dataset. The data is normalized using MinMax normalization to scale all input features in the range of [0,1].

Parameters for all models:

Parameters	Values
Hidden layers	Relu activation
Output layer	Softmax activation
Loss function	Cross Entropy
Batch size	32
Learning rate (η)	0.001
Optimiser	Adam
Beta 1 (β_1)	0.9
Beta 2 (β_2)	0.999
Epsilon (ϵ)	10^{-8}
Convergence criterion	Difference between successive epochs is less than or equal to 10^{-4}.

a. Architecture-1 (4 layer architecture)

Layer Name	Output dimensions
Input layer	(224,224,3)
Convolution layer 1	(54,54,8)
Max Pooling layer 1	(26,26,8)
Convolution layer 2	(22,22,16)
Max Pooling layer 2	(10,10,16)
Flatten	1600
Dense layer 1	128
Output layer	5

<table>
<tr><td rowspan="6">Confusion matrix
on
validation data</td><td colspan="6">A C T U A L</td></tr>
<tr><td rowspan="5">P
R
E
D
I
C
T
E
D</td><td>9</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>8</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>2</td><td>8</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>10</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>10</td></tr>
<tr><td rowspan="2">Classification accuracy</td><td colspan="3">Train</td><td colspan="3">Validation</td></tr>
<tr><td colspan="3">100%</td><td colspan="3">90%</td></tr>
</table>

b. Architecture-2 (5 layer architecture)

Layer Name	Output dimensions
Input layer	(224,224,3)
Convolution layer 1	(54,54,8)
Max Pooling layer 1	(26,26,8)
Convolution layer 2	(22,22,16)
Max Pooling layer 2	(10,10,16)
Convolution layer 3	(8,8,32)
Max Pooling layer 3	(3,3,32)
Flatten	288
Dense layer 1	128
Output layer	5

<table>
<tr><td rowspan="6">Confusion matrix
on
validation data</td><td colspan="6">A C T U A L</td></tr>
<tr><td rowspan="5">P
R
E
D
I
C
T
E
D</td><td>10</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>8</td><td>0</td><td>2</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>7</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>10</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>8</td></tr>
<tr><td rowspan="2">Classification accuracy</td><td colspan="3">Train</td><td colspan="3">Validation</td></tr>
<tr><td colspan="3">100%</td><td colspan="3">86%</td></tr>
</table>

c. Architecture-3 (6 layer architecture)

Layer Name	**Output dimensions**
Input layer	(224,224,3)
Convolution layer 1	(54,54,8)
Max Pooling layer 1	(26,26,8)
Convolution layer 2	(22,22,16)
Max Pooling layer 2	(10,10,16)
Convolution layer 3	(8,8,32)
Convolution layer 4	(6,6,64)
Max Pooling layer 4	(2,2,64)
Flatten	256
Dense layer 1	128
Output layer	5

<table>
<tr><td rowspan="6">Confusion matrix
on
validation data</td><td colspan="6">ACTUAL</td></tr>
<tr><td rowspan="5">PREDICTED</td><td>8</td><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>7</td><td>0</td><td>2</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>8</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>9</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>9</td></tr>
<tr><td rowspan="2">Classification accuracy</td><td colspan="3">Train</td><td colspan="3">Validation</td></tr>
<tr><td colspan="3">100%</td><td colspan="3">82%</td></tr>
</table>

d. Results of best architecture

After comparing the classification accuracy on validation data, we can observe that architecture 1 with 4 hidden layers works best. This section provides results on best architecture and feature map visualization for the same.

- **Results on test data**

<table>
<tr><td rowspan="6">Confusion matrix</td><td colspan="6">ACTUAL</td></tr>
<tr><td rowspan="5">PREDICTED</td><td>18</td><td>2</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>17</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>3</td><td>16</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>19</td><td>0</td></tr>
<tr><td>2</td><td>0</td><td>1</td><td>1</td><td>16</td></tr>
<tr><td>Classification accuracy</td><td colspan="6">86%</td></tr>
</table>

- **Feature maps for first convolution layer**

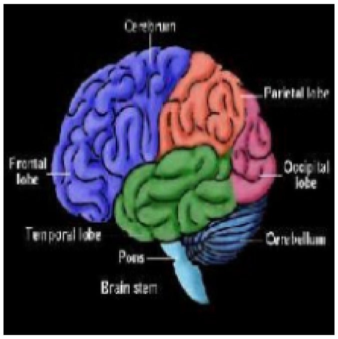


Fig 2: Image of a Scorpion from Caltech-101 dataset

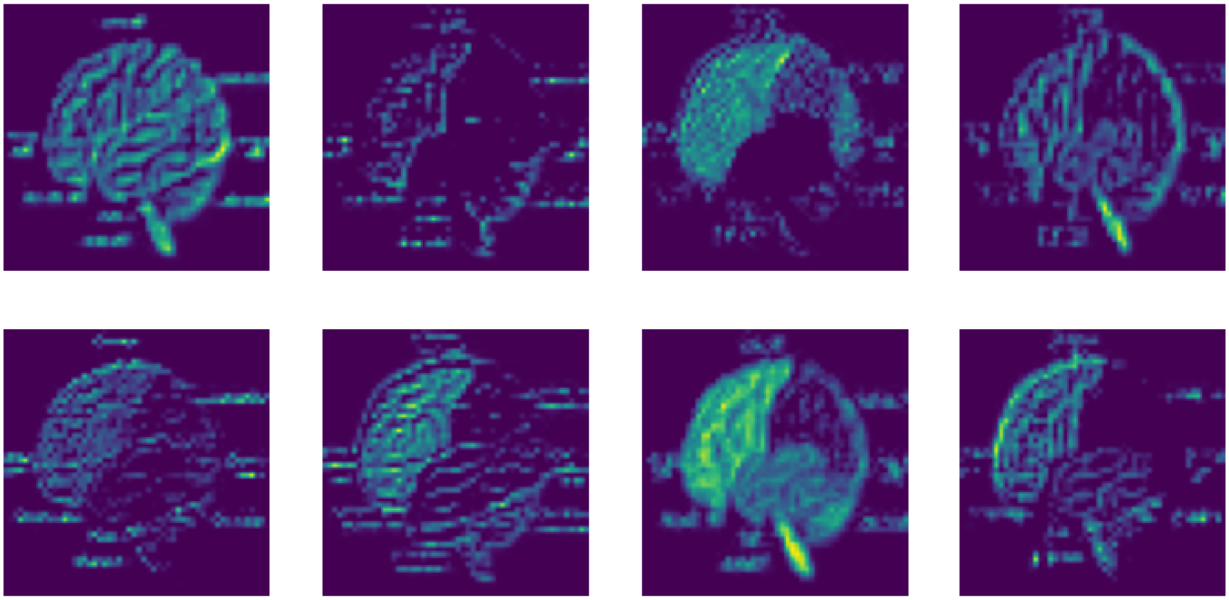


Fig 3: Feature maps for 8 filters of size 11 × 11 applied on the above image of scorpion.

● Feature maps for second convolution layer

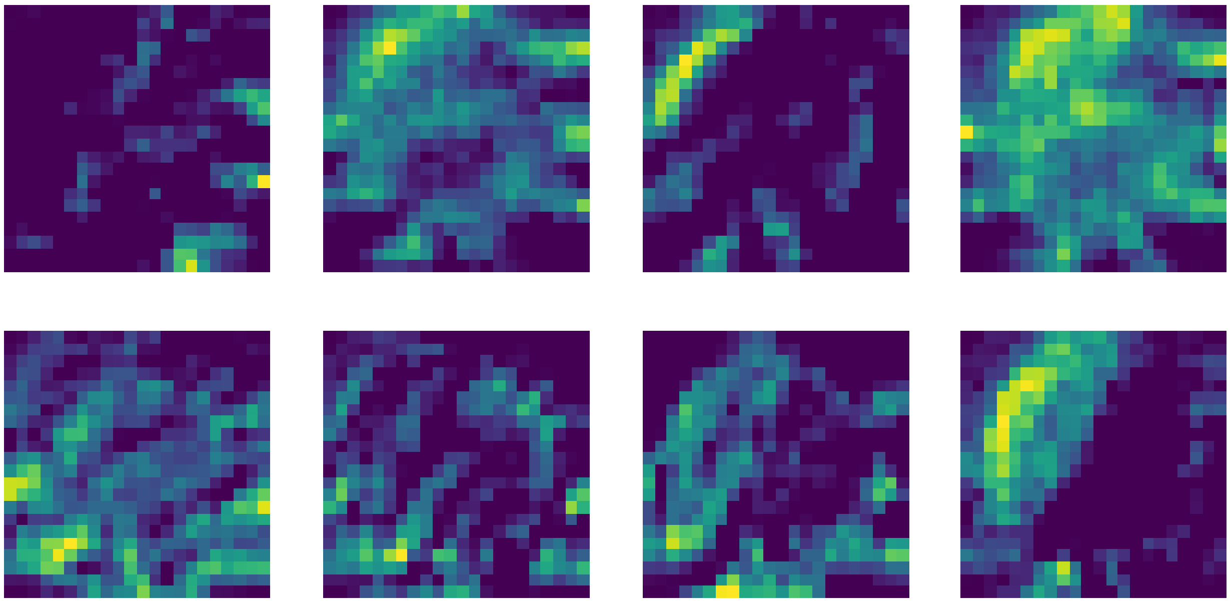
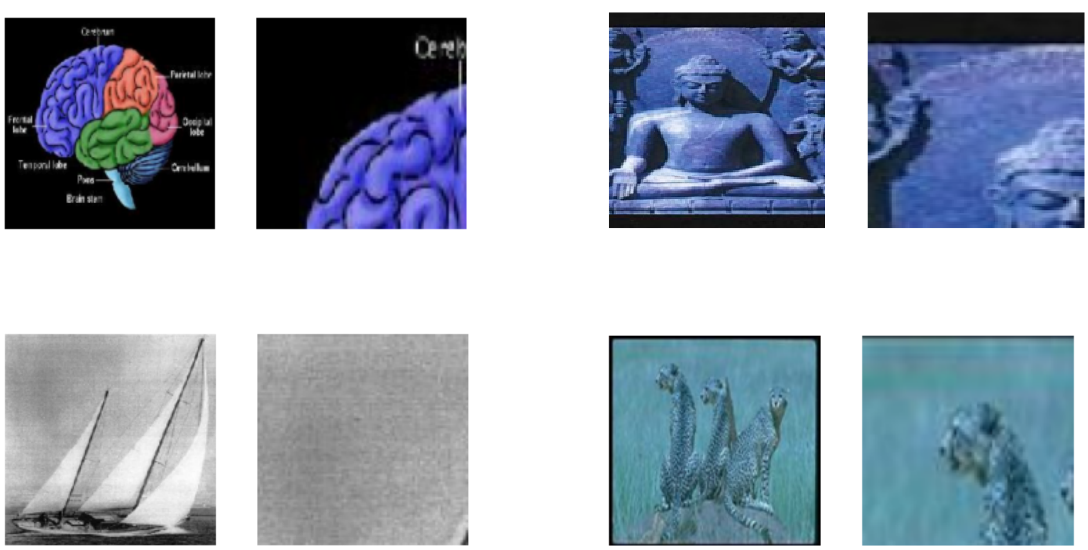


Fig 3: Selected 8 feature maps out of 16 filters of size 5 × 5 applied on the above feature maps.

● Visualizing input image patch for maximally activated neuron

To observe which part of the input image is responsible for maximal activation of our last convolution layer, we backtrace from maximally activated neuron through all previous layers.



Observations:

1. In all the architectures, the classification accuracy on unseen data is not very high ($\leq$ 90%) while the accuracy on training data for is 100% which indicates that the models are not able to generalize well. Lack of training data could be one reason behind it.

2. The feature maps plot of both convolution layers shows how the model is trying to learn different features at different layers. First layer learns simpler features like edges and gradients. Some parts of the feature maps are more activated for a particular color (7^{th} feature map is most activated for blue color). Second layer learns more abstract features, mostly curves and parts of the image.

2. Classification using VGG

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0
 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168
 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080
 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
 block3_conv4 (Conv2D)       (None, 56, 56, 256)       590080
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_conv4 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv4 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
 flatten_1 (Flatten)         (None, 25088)             0
 dense_1 (Dense)             (None, 5)                 12544
=================================================================
Total params: 20,149,829
Trainable params: 125,445
Non-trainable params: 20,024,384
_________________________________________________________________
```

- **Results on train data**

	ACTUAL

Confusion matrix	PREDICTED	50	0	0	0	0
		0	50	0	0	0
		0	0	50	0	0
		0	0	0	50	0
		0	0	0	0	50
Classification accuracy	100%					

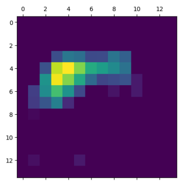
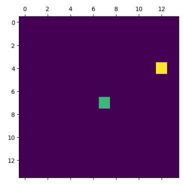
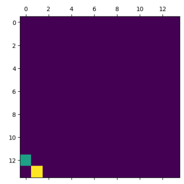
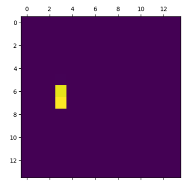
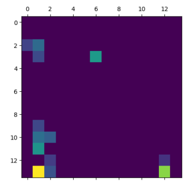
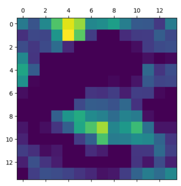
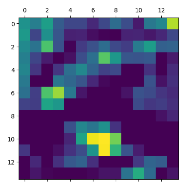
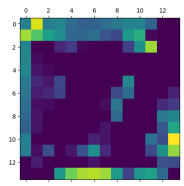
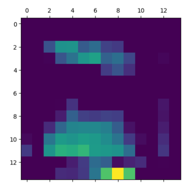
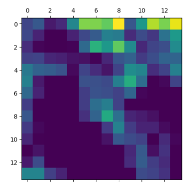
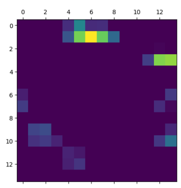
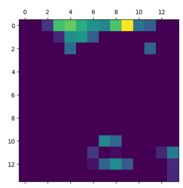
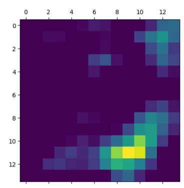
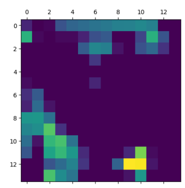
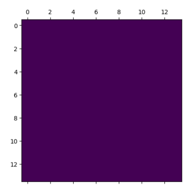
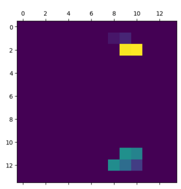
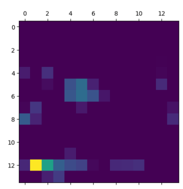
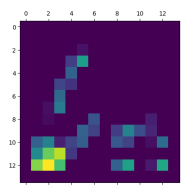
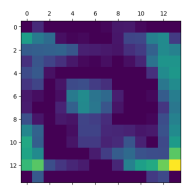
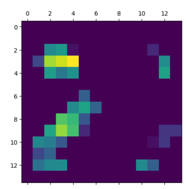
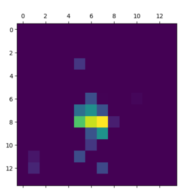
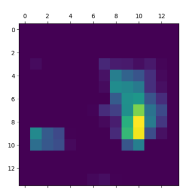
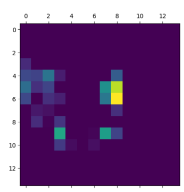
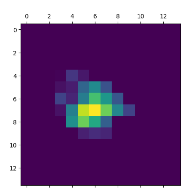
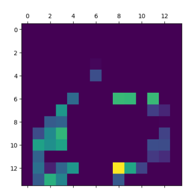
- **Results on validation data**

Confusion matrix	ACTUAL					
	PREDICTED	10	0	0	0	0
		0	10	0	0	0
		0	0	10	0	0
		0	0	0	10	0
		0	0	0	0	10
Classification accuracy	100%					

- **Results on test data**

Confusion matrix	ACTUAL					
	PREDICTED	20	0	0	0	0
		0	20	0	0	0
		0	0	20	0	0
		0	0	0	20	0
		0	0	0	0	20
Classification accuracy	100%					

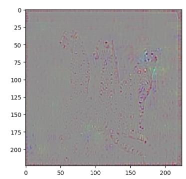
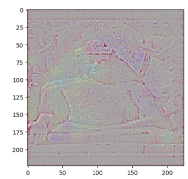
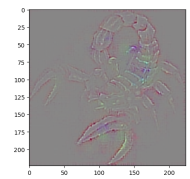
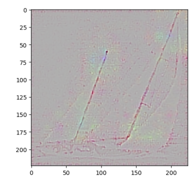
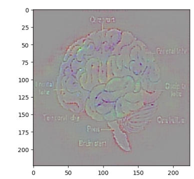
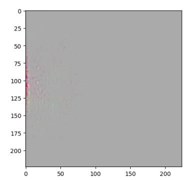
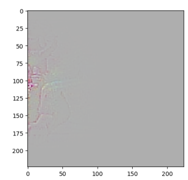
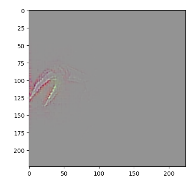
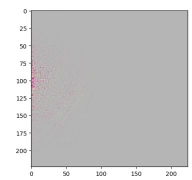
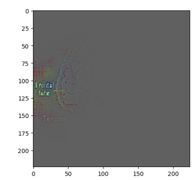
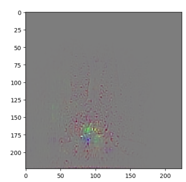
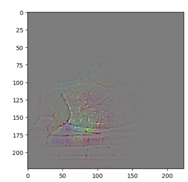
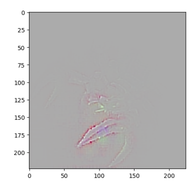
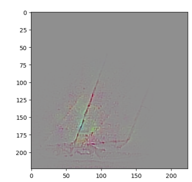
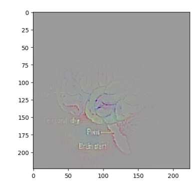
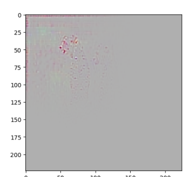
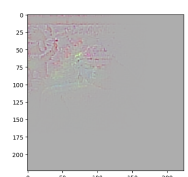
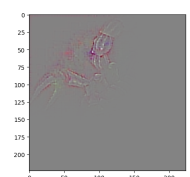
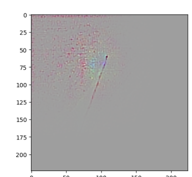
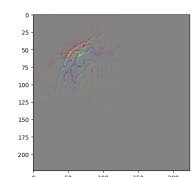
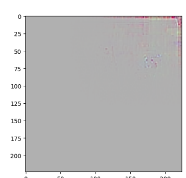
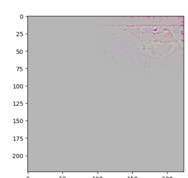
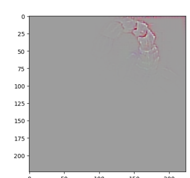
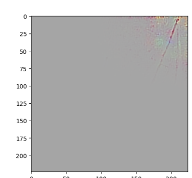
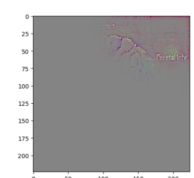
GradCam Results:

	Brain	Buddha	Ketch	Leopard	Scorpion
Brain					
Buddha					
Ketch					
Leopard					
Scorpion					

Observation:

If we consider the diagonal heatmaps of the above classes, then it somewhat represents the original shape which the model is trying to learn. The grid (1,1) clearly represents the shape of the brain. The other non-diagonal value is what the heatmap corresponds to the features which the model learns to distinguish other classes from that particular class.

Using Guided BackProp:

Neuron number	Leopard	Buddha	Scorpion	Ketch	Brain
93619					
43517					
67482					
23476					
5987					

Observations:

The first neuron (93619) is the maximally activated neuron among all other neurons. On tracing back to the filter, it can be clearly seen that this particular neuron (and its associated feature maps in the previous layers) is responsible for learning a better overall representation of each of the classes. The other neurons are randomly selected and it is observed that each neuron tries to learn some of the patch/portion of the input images.