CS671 : Deep Learning and Applications
Assignment 05 - Report

Convolution Neural Network

Ankit Mehra (S22020)
Urvashi Goswami(S22024)
Ashish Rana (S22019)

Introduction

A Convolutional Neural Network, also known as CNN or ConvNet, is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image. A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion that contains pixel values to denote how bright and what color each pixel should be.

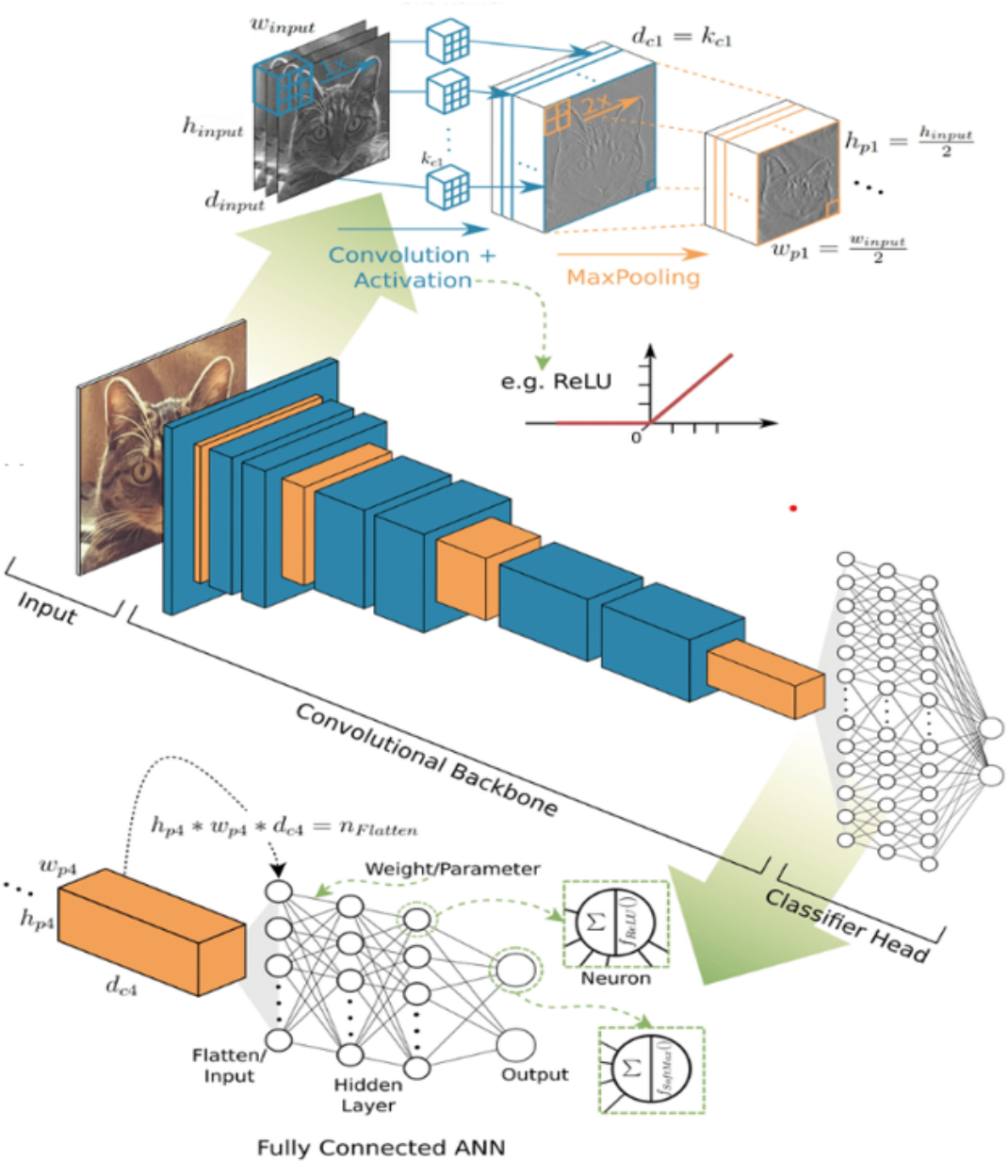


Fig 1 - (Overview)CNN architecture for image recognition
Source- ResearchGate

The human brain processes a huge amount of information the second we see an image. Each neuron works in its own receptive field and is connected to other neurons in a way that they cover the entire visual field. Just as each neuron responds to stimuli only in the restricted region of the visual field called the receptive field in the biological vision system, each neuron in a CNN processes data only in its receptive field as well. The layers are arranged in such a way so that they detect simpler patterns first (lines, curves, etc.) and more complex patterns (faces, objects, etc.) further along. By using a CNN, one can enable sight to computers.

A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

Convolution Layer

The convolution layer is the core building block of the CNN. It carries the main portion of the network's computational load.

This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but is more in-depth. This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels.

During the forward pass, the kernel slides across the height and width of the image-producing the image representation of that receptive region. This produces a two-dimensional representation of the image known as an activation map that gives the response of the kernel at each spatial position of the image. The sliding size of the kernel is called a stride.

If we have an input of size W x H x D and Dout number of kernels with a spatial size of F with stride S and amount of padding P, then the size of output volume can be determined by the following formula:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

Applying the similar formula for height ,will yield an output volume of size *Wout* x *Hou*t x *Dout*

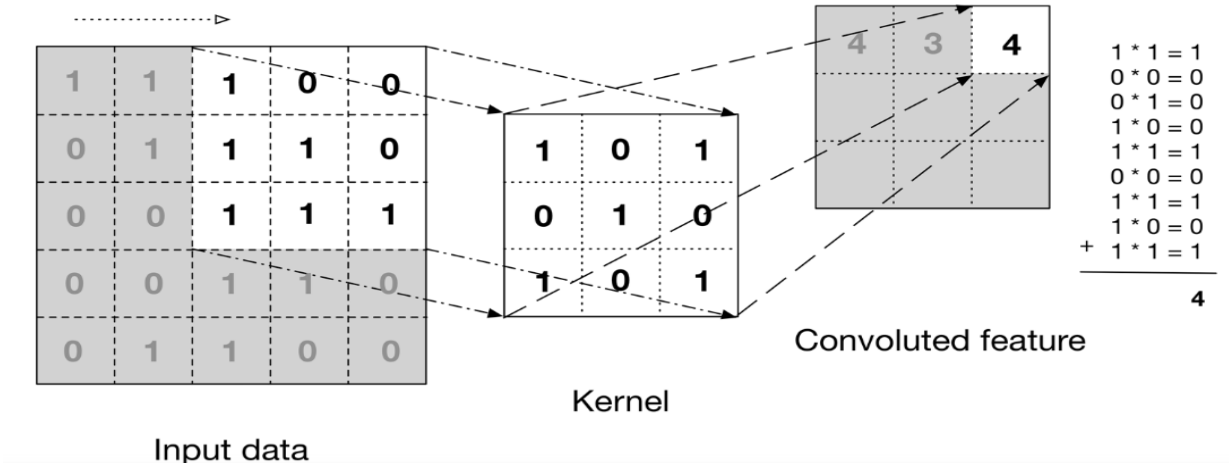


Fig 2- Convolution operation
Source- Vidya Analytics

Pooling Layer

The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually.

There are several pooling functions such as the average of the rectangular neighborhood, L2 norm of the rectangular neighborhood, and a weighted average based on the distance from the central pixel. However, the most popular process is max pooling, which reports the maximum output from the neighborhood.

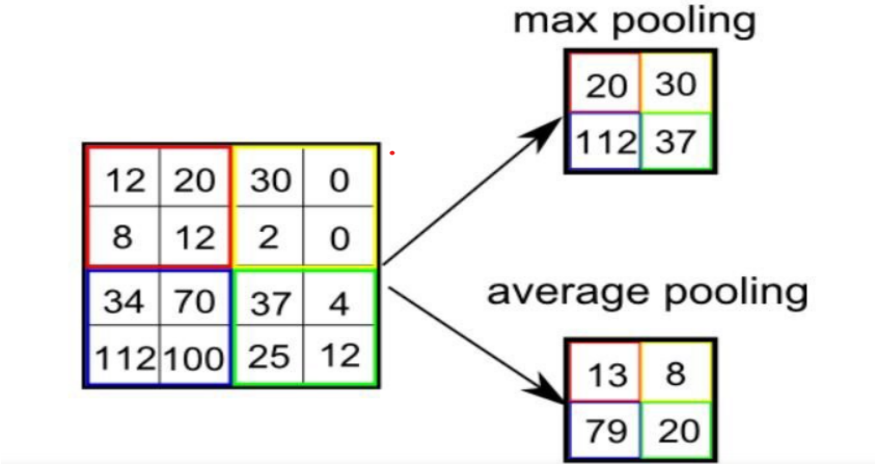


Fig 3- Pooling operations
Source- Vidya Analytics

VGG19

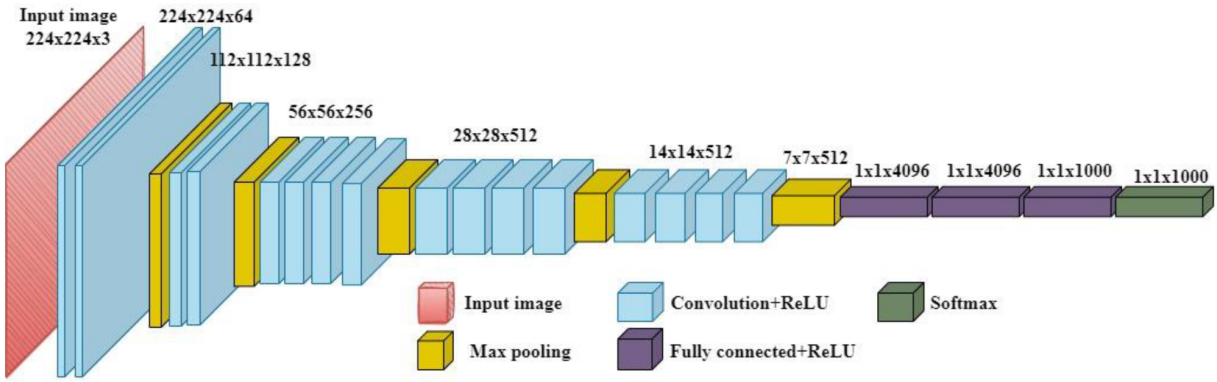


Fig 4- VGG19 architecture

Source-A VGG-19 Model with Transfer Learning and Image Segmentation for Classification of Tomato Leaf Disease

The input to VGG based convNet is a 224*224 RGB image. Preprocessing layer takes the RGB image with pixel values in the range of 0–255 and subtracts the mean image values which is calculated over the entire ImageNet training set.

The input images after preprocessing are passed through these weight layers. The training images are passed through a stack of convolution layers. VGG19 has weight layers consisting of 16 convolutional layers with 3 fully connected layers and the same 5 pooling layers. In both variations of VGGNet there consists of two Fully Connected layers with 4096 channels each which is followed by another fully connected layer with 1000 channels to predict 1000 labels. Last fully connected layer uses the softmax layer for classification purposes.

VGGNet has 19 weight layers consisting of 16 convolutional layers with 3 fully connected layers and the same 5 pooling layers. In both variations of VGGNet there consists of two Fully Connected layers with 4096 channels each which is followed by another fully connected layer with 1000 channels to predict 1000 labels. Last fully connected layer uses the softmax layer for classification purposes.

Grad-CAM

Grad-CAM is a popular technique for visualizing where a convolutional neural network model is looking. Grad-CAM is class-specific, meaning it can produce a separate visualization for every class present in the image:

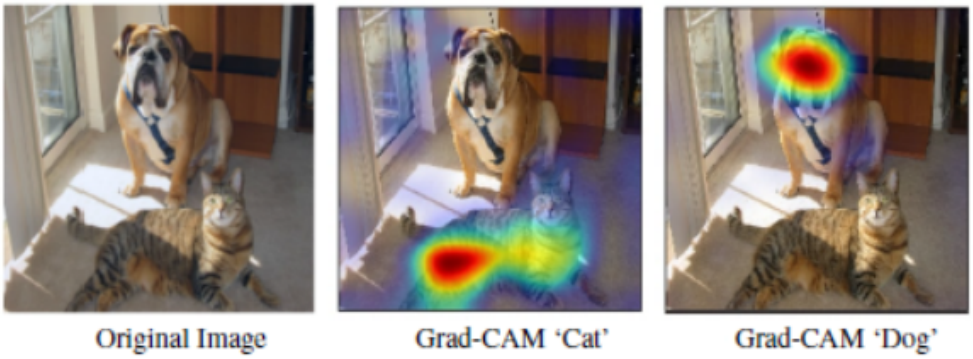


Fig 5- Cat and dog Grad-CAM visualization
Source- Grad-CAM research paper (2019)

Grad-CAM can be used for weakly-supervised localization, i.e. determining the location of particular objects using a model that was trained only on whole-image labels rather than explicit location annotations. Grad-CAM can also be used for weakly-supervised segmentation, in which the model predicts all of the pixels that belong to particular objects, without requiring pixel-level labels for training.Finally, Grad-CAM can be used to gain better understanding of a model, for example by providing insight into model failure modes.

Guided Backpropagation

Guided backpropagation is a technique for visualizing the regions of an input image that are most relevant for a neural network's output. It is a modification of the backpropagation algorithm that allows us to compute gradients with respect to the input pixels, which can then be used to highlight the regions of the image that had the most influence on the network's decision, guided backpropagation sets the gradients of all negative activations to zero during the backpropagation step. This allows us to focus on the positive activations in the network, which are more likely to correspond to relevant image features.

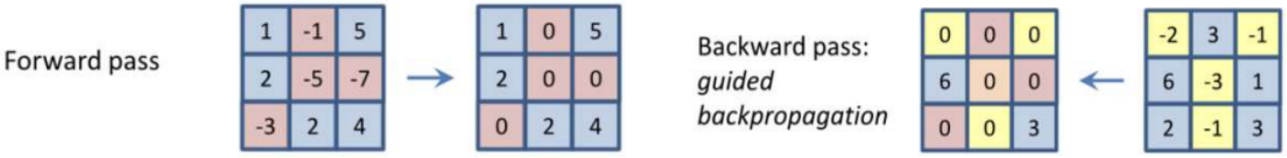


Fig 6- Guided Back Propagation
Source- Class slides Dr. Dileep AD (CS669)

Number of datapoints used to perform the task

Input	Training Data	Validation Data	Testing Data
Car Side	50	10	20
Hawksbill	50	10	20
Kangaroo	50	10	20
Scorpion	50	10	20
Starfish	50	10	20
Total	**250**	**50**	**100**

Table 1- Data Points used in the assignment.

We have the images from the above classes, we have resized the input images to (224×224)

Convolution Neural Network for classification.

Values of parameters taken over all architectures.

Parameters	Values
Learning Rate	0.001
Activation (Hidden)	ReLu
Activation (Output)	Softmax
Batch Size	32
Loss Function	Cross Entropy
Stopping Criterion	Early Stopping

Table 2- Parameters used in the architectures.

Description of various architectures

Architecture 1 (4 layers)

Input Layers	Operation	Kernel details				Calculations	Feature map dimension
		Size	No. of filters	Stride	Padding		
Input Layer (224 x 224 x 3)	Convolution	(11 X 11)	8	4	0	$\frac{(224-11+0)}{4} + 1 =$**54**	(54 x 54 x 8)
Layer 1 (54 x 54 x 8)	Max Pooling	(3 X 3)	-	2	0	$\frac{(54-3+0)}{2} + 1 =$**26**	(26 x 26 x 8)
Layer 2 (26 x 26 x 8)	Convolution	(5 X 5)	16	1	0	$\frac{(26-5+0)}{1} + 1 =$**22**	(22 x 22 x 16)
Layer 3 (22 x 22 x 16)	Max Pooling	(3 X 3)	-	2	0	$\frac{(22-3+0)}{2} + 1 =$**10**	(10 x 10 x 16)

Table 3- Detailed description of Architecture 1

Architecture 2 (6 layers)

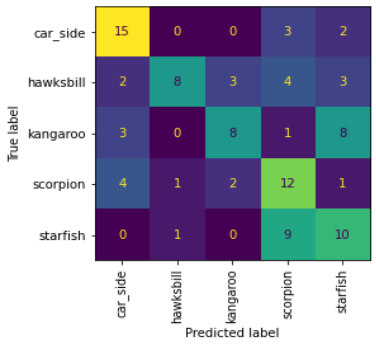
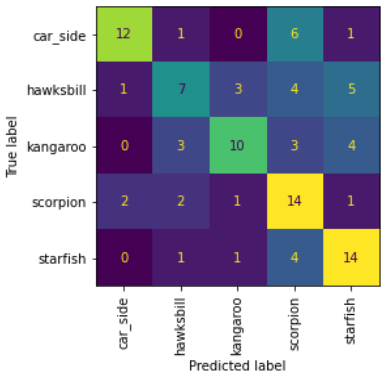
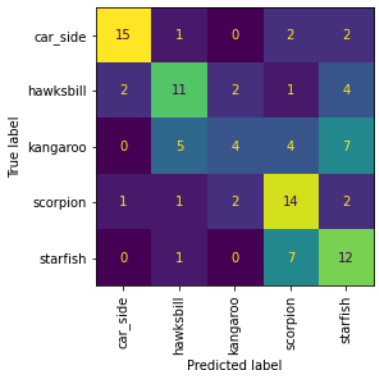
Input Layers	Operation	Kernel details				Calculations	Feature map dimension
		Size	No. of filters	Stride	Padding		
Input Layer (224 x 224 x 3)	Convolution	(11 X 11)	8	4	0	$\frac{(224-11+0)}{4} + 1 =$**54**	(54 x 54 x 8)
Layer 1 (54 x 54 x 8)	Max Pooling	(3 X 3)	-	2	0	$\frac{(54-3+0)}{2} + 1 =$**26**	(26 x 26 x 8)
Layer 2 (26 x 26 x 8)	Convolution	(5 X 5)	16	1	0	$\frac{(26-5+0)}{1} + 1 =$**22**	(22 x 22 x 16)
Layer 3 (22 x 22 x 16)	Max Pooling	(3 X 3)	-	2	0	$\frac{(22-3+0)}{2} + 1 =$**10**	(10 x 10 x 16)
Layer 4 (10 x 10 x 16)	Convolution	(3 X 3)	32	1	0	$\frac{(16-3+0)}{1} + 1 =$**8**	(8 x 8 x 32)
Layer 5 (8 x 8 x 32)	Max Pooling	(3 X 3)	-	2	0	$\frac{(8-3+0)}{2} + 1 =$**3**	(3 x 3 x 32)

Architecture 3 (7 layers)

Input Layers	Operation	Kernel details				Calculations	Feature map dimension
		Size	No. of filters	Stride	Padding		
Input Layer (224 x 224 x 3)	Convolution	(11 X 11)	8	4	0	$\frac{(224-11+0)}{4} + 1 =$**54**	(54 x 54 x 8)
Layer 1 (54 x 54 x 8)	Max Pooling	(3 X 3)	-	2	0	$\frac{(54-3+0)}{2} + 1 =$**26**	(26 x 26 x 8)
Layer 2 (26 x 26 x 8)	Convolution	(5 X 5)	16	1	0	$\frac{(26-5+0)}{1} + 1 =$**22**	(22 x 22 x 16)
Layer 3 (22 x 22 x 16)	Max Pooling	(3 X 3)	-	2	0	$\frac{(22-3+0)}{2} + 1 =$**10**	(10 x 10 x 16)
Layer 4 (10 x 10 x 16)	Convolution	(3 X 3)	32	1	0	$\frac{(16-3+0)}{1} + 1 =$**8**	(8 x 8 x 32)
Layer 5 (8 x 8 x 32)	Convolution	(3 X 3)	64	1	0	$\frac{(8-3+0)}{1} + 1 =$**6**	(6 x 6 x 64)
Layer 6 (6 x 6 x 64)	Max Pooling	(3 X 3)	-	2	0	$\frac{(6-3+0)}{2} + 1 =$**2**	(2 x 2 x 64)

Table 5- Detailed description of Architecture 3

Comparison of different architectures based on validation accuracy.

Architecture	Accuracy		Confusion Matrix
	Training	Validation	
Architecture 1	80.08%	62.00%	
Architecture 2 (Without Transfer Learning)	87.20%	60.00%	
Architecture 2 (With Transfer Learning)	86.00%	68.00%	

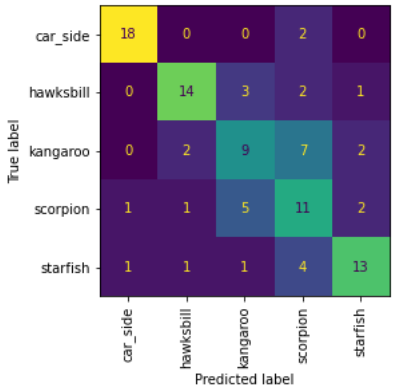
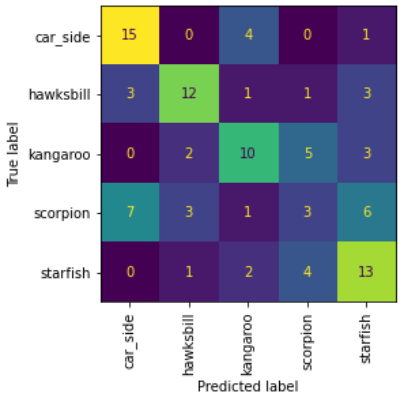
Architecture 3 (Without Transfer Learning)	86.80%	62.00%	
Architecture 3 (With Transfer Learning)	98.00%	58.00%	

Table 6- Comparison of Architectures.

Best Architecture based on validation accuracy

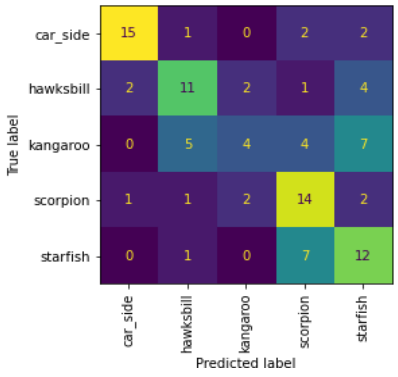
Architecture	Confusion Matrix	Val Accuracy	Test Accuracy
Architecture 2		**68.00 %**	**56.00%**

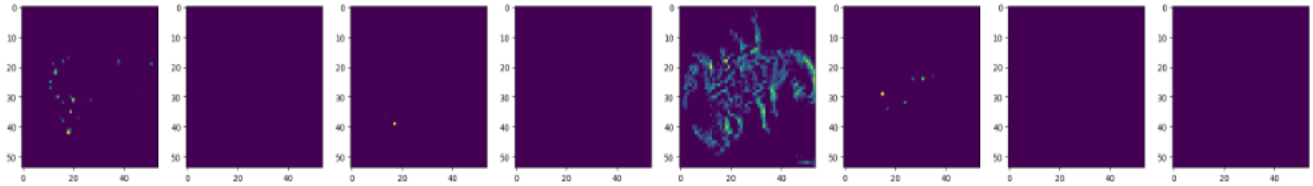
Table 7- Best Architecture.

Feature Maps from different convolution layer for the best architecture

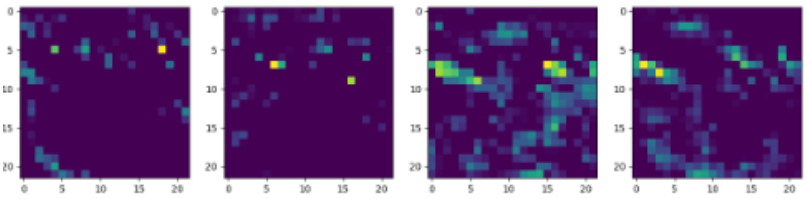
Feature map is the output of one filter applied to the previous layer. It is called a feature map because it is a mapping of where a certain kind of feature is found in the image.

Image chosen from class **"Scorpion"**
Architecture 2 (with transfer learning) is the best architecture.

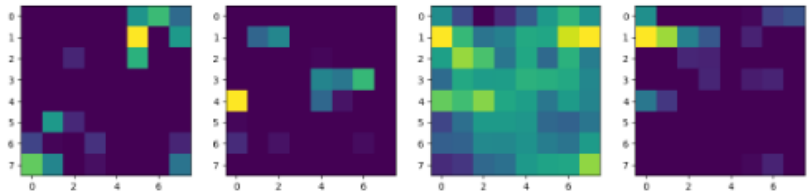
Feature Maps for first convolution layer.



Feature Maps for second convolution layer.



Feature Maps for third convolution layer.



From the above results we can infer that

- Each filter detects a specific pattern in the image such as edges and corners in the scorpion image. Feature map highlights where these features are present.
- In the second and third convolution layer we see that convolution and pooling operations learn increasingly abstract and complex features of the input image.
- The blank filter simply indicates that the filter is not detecting any meaningful features in the areas of those input images.

Visualizing Patches which Maximally Activate a Neuron

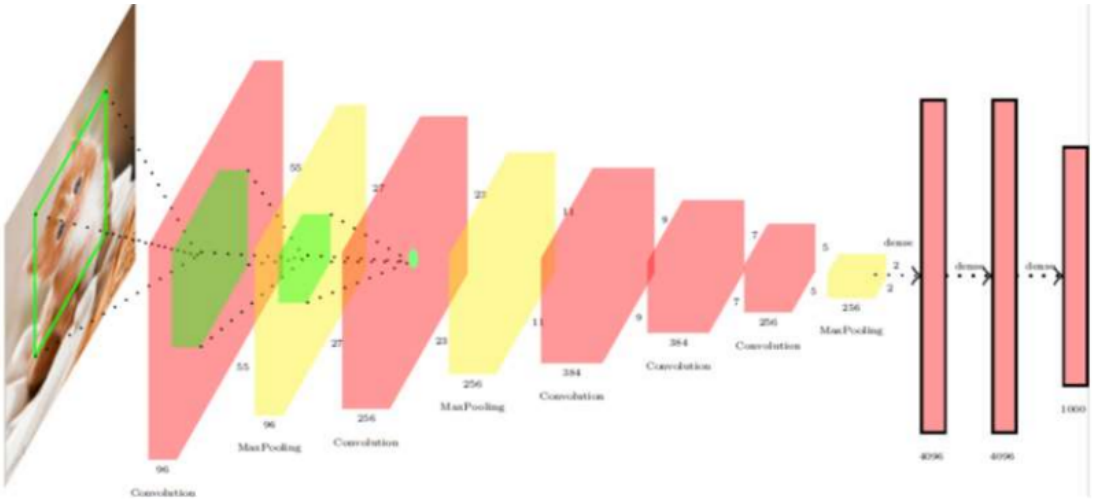


Table 8- Patch visualization.
Source :- Class Slides Dr. Dileep AD(CS699)

Method used

Step 1 :- Using numpy argmax() function we computed maximally activated neuron in the last convolution layer.
Step2 :- We calculated which neuron in the previous layer (second last convolution layer) contributed to that maximally activated neuron.

- The neurons in the previous layer will be obtained in a square with top left index (i x s , j x s) and bottom right index (i x s +k-1, j x s + k-1) where i and j are the coordinates of the maximally activated neuron .

Step 3 :-Do the same for the top left index and bottom right index to get the top left index and bottom right index of the square in the 3rd last convolution layer.
Step 4 :- Do step 3 iteratively till you reach the input layer and get the final patch .

Neuron	Input Image	Patch learnt	
43			

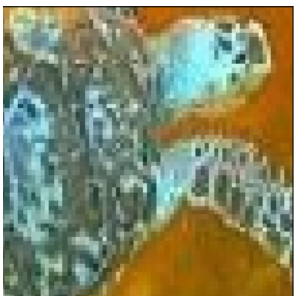
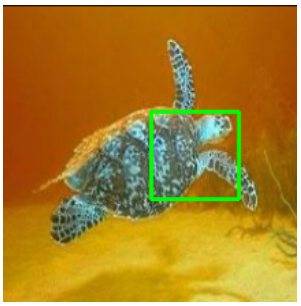
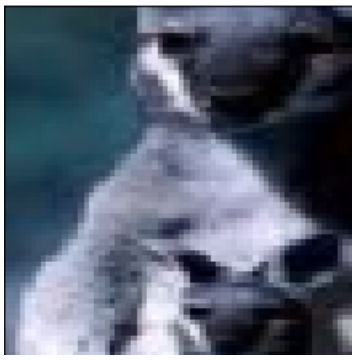
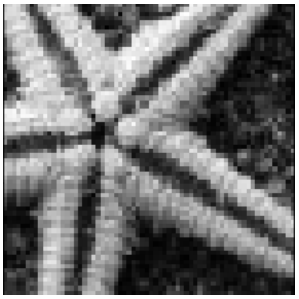
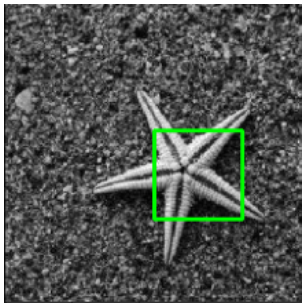
61			
42			
48			
62			

Table 9- Patch Visualization.

From the above results we can infer that

- We see what different neurons in the last layer of CNN are capturing, that is what they are firing for.
- These neurons are trying to learn specific characters of the input , which can have discriminative properties which will help in classification.
- It can be used in debugging tools where we can see what output is coming for what input and try to debug our code accordingly.

Comparison between the best architecture from the previous architectures and VGG-19 pre trained on ImageNet with added classification layer

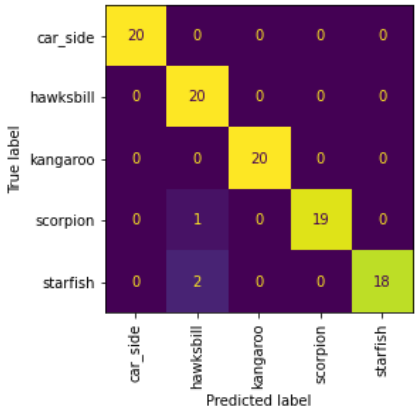
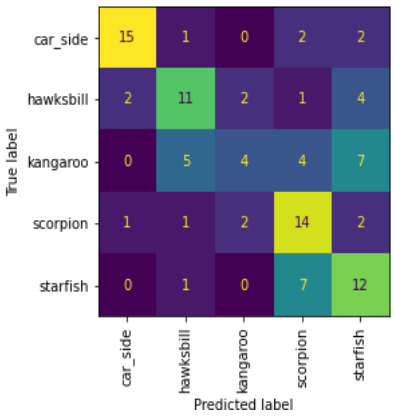
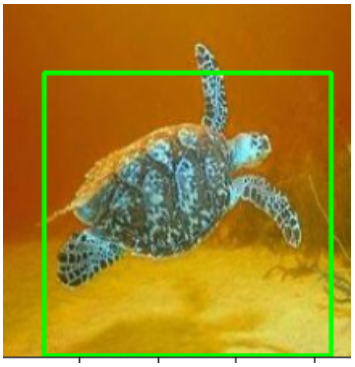
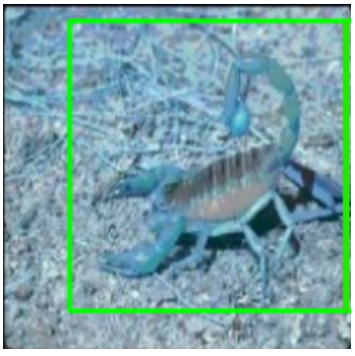
Parameters		Architecture with VGG-19 pre trained layers	Architecture 2(With Transfer learning)
Confusion Matrix			
Accuracy	Train	100.00%	86.00%
	Val	100.00%	68.00%
	Test	99.00%	56.00%

Table 10- Comparison between VGG-19 and Architecture 2.

Visualize the patches that maximally activate the neuron (VGG-19).

Neuron	Input Image	Patch learnt	
43			
122			
147			
131			

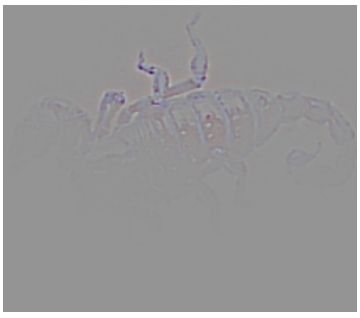
80			

Table 11- Patch Visualization with VGG-19 trained model.

Guided-backpropagation algorithm to find the influence.

We feed an input image to the CNN and do a forward pass. Then we consider one neuron in some feature map at last layer, we want to find the influence of the input on this neuron .We retain this neuron and set all other neurons in the layer to zero then we backpropagate all the way to the inputs

Image is taken from the "**Scorpion**" class.

Feature Map	Resultant Gradient Image
2nd Feature Map	
4th Feature Map	

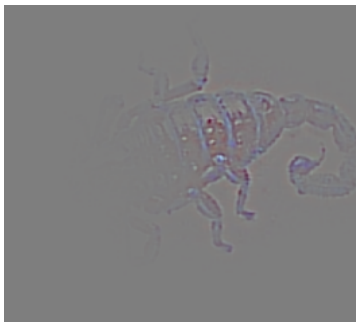
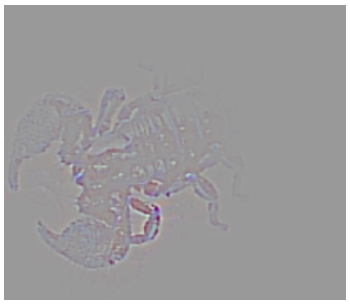
5th Feature Map	
6th Feature Map	
8th Feature Map	

Table 12 :- Guided Backpropagation Results.

From the above results we can infer that : -

- Guided Backpropagation of selected 5 neurons from the last convolution layer highlights the input features that are most responsible for producing certain output from a convolution neural network.
- Guided backpropagation of neurons in the last layer helps us identify which class specific neuron are most active for particular input image , this provides us insights into how the neural network is recognising and distinguishing between the different classes of the object.

Visualizing localization map (heat map) using the GradCAM

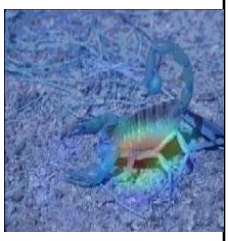
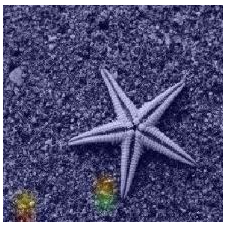
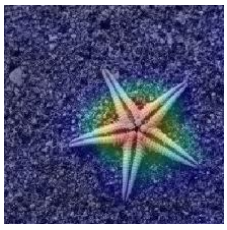
	Carside	Hawksbill	Kangaroo	Scorpion	Starfish
Carside					
Hawksbill					
Kangaroo					
Scorpion					
Starfish					

Table 13 :- GradCam Heatmaps.

From the above results we can infer that :-

- We use Gradcam to visualize the most important regions of different input images that are most responsible for a particular decision made by a CNN .
- Heatmaps generated by Gradcam for different class inputs are different , with some inputs showing higher heatmaps and others showing lower heatmap; it means that the CNN is assigning different levels of importance to the different regions of the input image for each class.
- For example our CNN is trained to recognize starfish, scorpio,carside,kangaroo,hawksbill so it will assign more importance to the head of the kangaroo for one class and fangs of scorpio for the other.