01/05/2023

Deep Learning Assignment-5

Convolutional Neural Network (CNN)

Presented by:

1. R. Ramakrishna Kashyap – T22101
2. Anuj Kumar Shukla – T22103
3. Nishant Gupta- T22221

Index

1.Introduction

A Convolutional Neural Network (CNN) is a type of deep neural network that is primarily used for image and video processing applications. The fundamental building block of a CNN is the convolutional layer, which performs a mathematical operation called convolution on the input image.

Convolutional neural networks (CNNs) have revolutionized the field of Image Processing and have become the go-to method for image classification, object detection, and other related tasks.

The main idea behind CNNs is to use convolutional layers to extract meaningful features from the input images and then use these features to classify them into different categories.

In this assignment, we will be exploring the topic of CNNs in more depth by working with a subset of the Caltech-101 dataset for image classification. This dataset consists of colour images with varying sizes, and our task is to resize them to a fixed size of 224 x 224 and classify them into five different categories.

Number of images given for training, validation, and testing for all the classes:

Image of	Training Images	Validation Images	Testing Images
Butterfly	50	10	20
Chandelier	50	10	20
Grand piano	50	10	20
Leopards	50	10	20
Revolver	50	10	20

2. Architectures and parameters Used

We have used three architectures for all cases of dimensions.

Architecture-1

Layers	Filters	Stride	Padding	Max. pooling
I (Conv)	8 of 11*11	4	0	3*3
II (Conv)	16 of 5*5	1	0	3*3

Table-1: Details of filers and max pooling for architecture -1

Architecture-2

Layers	Filters	Stride	Padding	Max. pooling
I (Conv)	8 of 11*11	4	0	3*3
II (Conv)	16 of 5*5	1	0	3*3
III(Conv)	32 of 3*3	1	0	3*3

Table-2: Details of filers and max pooling for architecture -2

Architecture-3

Layers	Filters	Stride	Padding	Max. pooling
I (Conv)	8 of 11*11	4	0	3*3
II (Conv)	16 of 5*5	1	0	3*3
III(Conv)	32 of 3*3	1	0	No max pooling
IV(Conv)	64 of 3*3	1	0	3*3

Table-3: Details of filers and max pooling for architecture -3

Output was flattened after the final convolution layer and succeeded with two fully connected layers. 128 hidden nodes with rectified linear activation function in the first hidden layer and 5 neurons with a Softmax activation function in the second layer (output layer) were considered. Also max pooling is used with a stride of 2 at every instances.

3. Structure of a CNN

CNN is implemented as a feedforward neural network.

Characteristics of CNN:

- The connections are much sparser.
- Weights are shared
  - Weight sharing make the job of learning easier (instead of trying to learn the same weights/kernels at different locations again and again)
- Number of nodes in a hidden layer is based on size of the input, size of the kernel (2-D or 3-D), number of kernels and stride.
- Each hidden layer is the resultant of convolution operation on the previous layer.
- Rectified linear function is used as activation function on the output of convolution operation.
  - Each hidden unit (neuron) is considered as ReLU– It has alternate convolution and pooling layers – Last layer of CNN is classification layer (output layer).
  - Every node from the layer before classification layer is connected to every node in classification layer (dense connection).
  - There may be more than 1 dense layer before the classification layer

5. Working of Convolutional Neural Network

The working of a Convolutional Neural Network (CNN) involves a series of interconnected layers that process the input image to generate an output. Here are the main steps involved:

Input Layer: The input to the CNN is an image, represented as a matrix of pixel values. The image is typically pre-processed by normalizing the pixel values and resizing it to a fixed size.

Convolutional Layers: The first set of layers in a CNN are the convolutional layers. Each convolutional layer applies a set of filters to the input image, where each filter extracts a specific feature from the image. The filters are learned during the training process and are adjusted to optimize the performance of the network. The output of each convolutional layer is a set of feature maps, where each map corresponds to a specific filter.

Activation Layers: After each convolutional layer, an activation layer is applied to introduce non-linearity into the network. The most used activation function is the Rectified Linear Unit (ReLU) function, which sets all negative values to zero.

Pooling Layers: The next set of layers are the pooling layers, which reduce the spatial size of the feature maps by taking the maximum or average value over a small region. This helps to reduce the number of parameters in the network and makes it more computationally efficient.

Fully Connected Layers: The final layers of the CNN are the fully connected layers, which take the flattened output of the previous layers and apply a set of learned weights to generate the final output. The output of the fully connected layers is typically a probability distribution over the classes in a classification task.

Loss Function: During the training process, the output of the CNN is compared to the ground truth labels using a loss function, such as cross-entropy. The goal of the training process is to minimize the value of the loss function by adjusting the weights of the network using backpropagation.

Optimization: The optimization process involves adjusting the weights of the network to minimize the loss function. This is typically done using an optimization algorithm, such as stochastic gradient descent.

Prediction: Once the CNN is trained, it can be used to make predictions on new, unseen images. The input image is passed through the layers of the CNN, and the output is a probability distribution over the classes in a classification task.

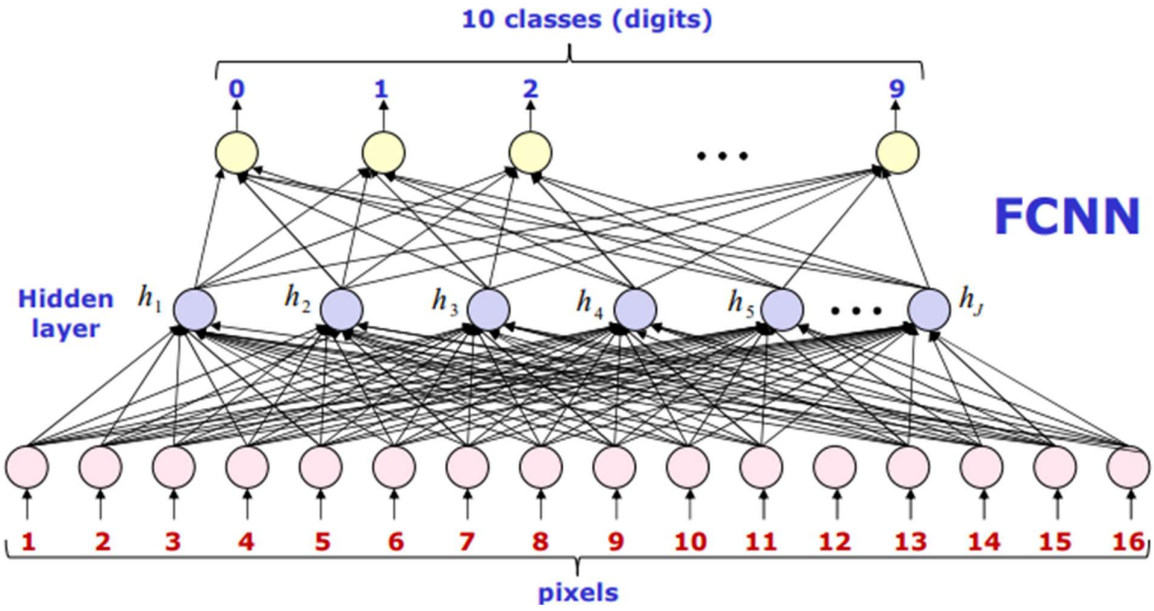


Image source: Lecture slides (CS671: Deep learning and Applications, Dr. Dileep A.D.)

Results and Inferences

Task-1

Architecture	Training Accuracy	Validation Accuracy	Confusion Matrix
Architecture -1	95.2	72	
Architecture - 2	96	76	
Architecture - 3	100	84	

Architecture -1

		Actual Class				
		Class 1	Class2	Class 3	Class 4	Class 5
Predicted Class	Class 1	12	2	0	3	3
	Class 2	3	11	0	4	2
	Class 3	3	3	8	2	4
	Class 4	1	1	0	16	2
	Class 5	1	4	1	1	13

Architecture - 2

		Actual Class				
		Class 1	Class2	Class 3	Class 4	Class 5
Predicted Class	Class 1	12	2	0	3	3
	Class 2	4	11	4	1	0
	Class 3	3	1	13	1	2
	Class 4	2	1	0	17	0
	Class 5	1	2	0	3	14

Architecture - 3

		Actual Class				
		Class 1	Class2	Class 3	Class 4	Class 5
Predicted Class	Class 1	17	0	1	1	1
	Class 2	0	12	3	5	0
	Class 3	1	1	18	0	0
	Class 4	1	0	0	19	0
	Class 5	2	2	0	0	16

For a given input layer and kernel the output feature map dimension is given by [(Input size - kernel_size+2*padding)/stride +1].

Feature Maps for Architecture 1

- Rescaling layer: The input images are rescaled by dividing the pixel values by 255.
- 1st Convolutional Layer: A Convolutional layer with 8 filters, kernel size of 11x11, stride of 4 is applied to the input images. The resulting feature maps have a size [(224-11)/4 +1] = 54.
- Max Pooling Layer: A MaxPooling layer with pool size of 3x3 and stride of 2 is applied to the output of the first Conv2D layer. The resulting feature maps have a size of (54-3)/2 + 1 = 26.
- 2nd Convolutional Layer: A Convolutional layer with 16 filters, kernel size of 5x5 stride of 1 is applied to the output of the MaxPooling layer. The resulting feature map have a size of (26-5+1)/1=22.
- Max Pooling Layer: Another MaxPooling layer with pool size of 3x3 and stride of 2 is applied to the output of the second Conv2D layer. The resulting feature maps have a size of (22-3)/2 + 1 = 10.

- Flatten Layer: The output of the second MaxPooling layer is flattened into a 1D vector.

Feature Maps for Architecture 2

- Rescaling layer: The input images are rescaled by dividing the pixel values by 255.
- 1st Convolutional Layer: A Convolutional layer with 8 filters, kernel size of 11x11, stride of 4 is applied to the input images. The resulting feature maps have a size of (224-11+1)/4 = 54.
- Max Pooling Layer: A MaxPooling layer with pool size of 3x3 and stride of 2 is applied to the output of the first Conv2D layer. The resulting feature maps have a size of (54-3)/2 + 1 = 26.
- 2nd Convolutional Layer: A Convolutional layer with 16 filters, kernel size of 5x5, stride of 1 is applied to the output of the MaxPooling2D layer. The resulting feature maps have a size of (26-5+1)/1 = 22.
- Max Pooling Layer: Another MaxPooling layer with pool size of 3x3 and stride of 2 is applied to the output of the second Conv2D layer. The resulting feature maps have a size of (22-3)/2 + 1 = 10.
- 3rd Convolutional Layer: A Convolutional layer with 32 filters, kernel size of 3x3, stride of 1 is applied to the output of the second MaxPooling2D layer. The resulting feature maps have a size of (10-3+1)/1 = 8.
- Max Pooling Layer: Another MaxPooling layer with pool size of 3x3 and stride of 2 is applied to the output of the third Conv2D layer. The resulting feature maps have a size of (8-3)/2 + 1 = 3.
- Flatten Layer: The output of the second MaxPooling2D layer is flattened into a 1D vector.

Feature Maps for Architecture 3

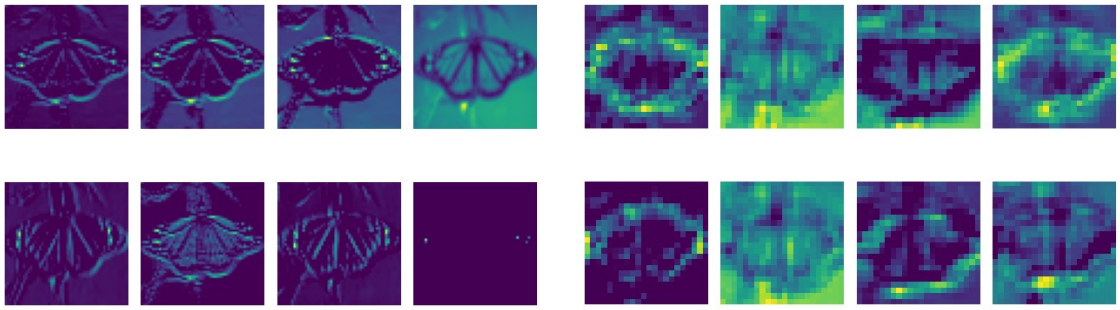
- Rescaling layer: The input images are rescaled by dividing the pixel values by 255.
- 1st Convolutional Layer: A Convolutional layer with 8 filters, kernel size of 11x11, stride of 4 is applied to the input images. The resulting feature maps have a size of (224-11+1)/4 = 54.
- Max Pooling Layer: A MaxPooling layer with pool size of 3x3 and stride of 2 is applied to the output of the first Conv2D layer. The resulting feature maps have a size of (54-3)/2 + 1 = 26.

- 2nd Convolutional Layer: A Convolutional layer with 16 filters, kernel size of 5x5, stride of 1 is applied to the output of the MaxPooling2D layer. The resulting feature maps have a size of (26-5+1)/1 = 22.
- Max Pooling Layer: Another MaxPooling layer with pool size of 3x3 and stride of 2 is applied to the output of the second Conv2D layer. The resulting feature maps have a size of (22-3)/2 + 1 = 10.
- 3rd Convolutional Layer: A Convolutional layer with 32 filters, kernel size of 3x3, stride of 1 is applied to the output of the second MaxPooling2D layer. The resulting feature maps have a size of (10-3+1)/1 = 8.
- Max Pooling Layer: Another MaxPooling layer with pool size of 3x3 and stride of 2 is applied to the output of the third Conv2D layer. The resulting feature maps have a size of (8-3)/2 + 1 = 3.
- 4th Convolutional Layer: A Convolutional layer with 64 filters, kernel size of 3x3, stride of 1 is applied to the output of the second MaxPooling2D layer. The resulting feature maps have a size of (3-3+1)/1 = 8.
- Max Pooling Layer: Another MaxPooling layer with pool size of 3x3 and stride of 2 is applied to the output of the third Conv2D layer. The resulting feature maps have a size of (8-3)/2 + 1 = 3.
- Flatten Layer: The output of the second MaxPooling2D layer is flattened into a 1D vector.

Based on validation accuracy 3rd architecture was found to be the best performing.

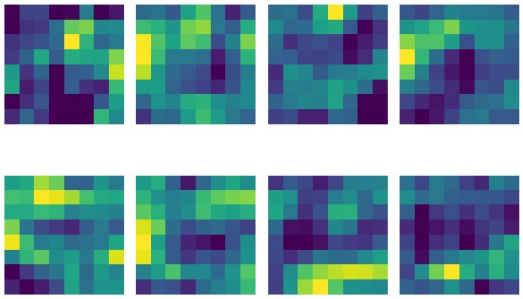
Original Image

Feature Maps of each layer of Best Architecture

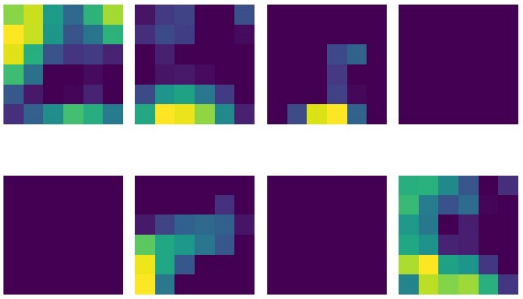


1st Convolutional Layer

2nd Convolutional Layer



3rd Convolutional Layer



4th Convolutional Layer

Based on the observation of the plotted feature maps, it appears that the images in each feature map were observed to be blurry. The blurriness was found to increase as the depth of the network increased.

The observed blurriness in the feature maps may be attributed to the down sampling technique commonly employed in CNNs. Down sampling is a technique utilized to reduce the size of the feature maps, thus extracting more abstract features from the input image. Here Max pooling was used to down sample the output of convolutional layers.

Performance of VGG19

1. Training Accuracy : 99%
2. Validation Accuracy : 96%
3. Testing Accuracy :96%

	Actual Class					
Predicted Class		**Class 1**	**Class2**	**Class 3**	**Class 4**	**Class 5**
	Class 1	17	3	0	0	0
	Class 2	0	20	0	0	0
	Class 3	0	0	0	20	0
	Class 4	0	0	0	20	0
	Class 5	0	1	0	0	19

Confusion Matrix for Test Data

VGG19 performed better than the best architecture in CNN due to its deeper architecture and the use of smaller convolutional filters. VGG19 is a deeper network with 19 layers, compared to the 4 convolutional layers in the CNN network. The increased depth allows VGG19 to learn more complex and abstract features from the input images. Additionally, VGG19 uses smaller convolutional filters (3x3) compared to the larger filters used in your network, which allows it to capture more finer details in the images.

VGG19 has been trained on larger datasets such as ImageNet, which contains millions of labeled images, while the training of CNN have been limited due to the smaller size dataset.