**PART A**

**EXPERIMENT NO. 7**

**A.1 AIM: -** "Exploring Convolutional Neural Networks: Analyzing Layer Outputs in a CNN Architecture with Two Convolutional Layers, Pooling, and Fully Connected Layers"

PART B

|  |  |
| --- | --- |
| Roll No. C009 | Name: Samarth Borade |
| Class : BTI SEM 10 | Batch : EB1 |
| Date of Experiment: 08/03/24 | Date of Submission |
| Grade : |  |

**B.1 Documentation written by student:**

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

# Define the CNN architecture

def cnn\_model(*input\_shape*):

model = tf.keras.models.Sequential([

# Convolutional Layer 1

tf.keras.layers.Conv2D(*filters*=8, *kernel\_size*=(5, 5), *strides*=(1, 1), *padding*='valid', *activation*='relu', *input\_shape*=*input\_shape*),

# Pooling Layer 1

tf.keras.layers.MaxPooling2D(*pool\_size*=(2, 2), *strides*=(2, 2), *padding*='valid'),

# Convolutional Layer 2

tf.keras.layers.Conv2D(*filters*=16, *kernel\_size*=(5, 5), *strides*=(1, 1), *padding*='valid', *activation*='relu'),

# Pooling Layer 2

tf.keras.layers.MaxPooling2D(*pool\_size*=(2, 2), *strides*=(2, 2), *padding*='valid'),

# Flatten layer

tf.keras.layers.Flatten(),

# Fully Connected Layer 1

tf.keras.layers.Dense(120, *activation*='relu'),

# Fully Connected Layer 2

tf.keras.layers.Dense(84, *activation*='relu'),

# Output Layer

tf.keras.layers.Dense(10, *activation*='softmax')

])

return model

# Function to display the output of each layer

def visualize\_layers(*model*, *input\_image*):

# Reshape input image for compatibility with the model

*input\_image* = np.expand\_dims(*input\_image*, *axis*=0)

layer\_outputs = [layer.output for layer in *model*.layers] # Extracts the outputs of the top 10 layers

activation\_model = tf.keras.models.Model(*inputs*=*model*.input, *outputs*=layer\_outputs) # Creates a model that will return these outputs, given the model input

# Get the activations for each layer

activations = activation\_model.predict(*input\_image*)

# Visualize each layer's output

for i, activation in enumerate(activations):

plt.figure()

if activation.ndim == 4:

plt.matshow(activation[0, :, :, 0], *cmap*='viridis')

elif activation.ndim == 2:

plt.matshow(activation.reshape(1, -1), *cmap*='viridis')

plt.title("Layer {} Output".format(i+1))

plt.show()

# Read input image

input\_image = plt.imread('lion.png')

# Define input shape for the model

input\_shape = input\_image.shape

# Create the model

model = cnn\_model(input\_shape)

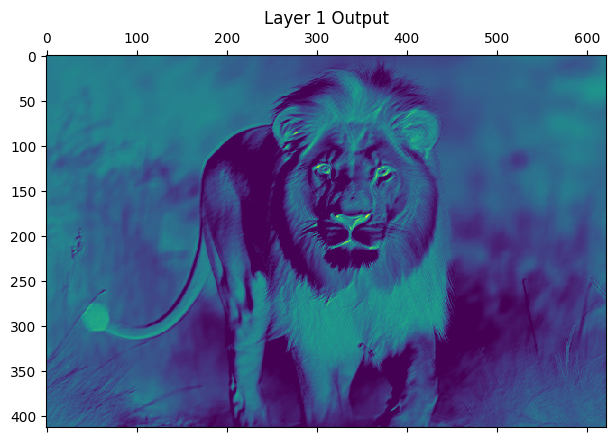
# Display the output of each layer

visualize\_layers(model, input\_image)

**Input image:**

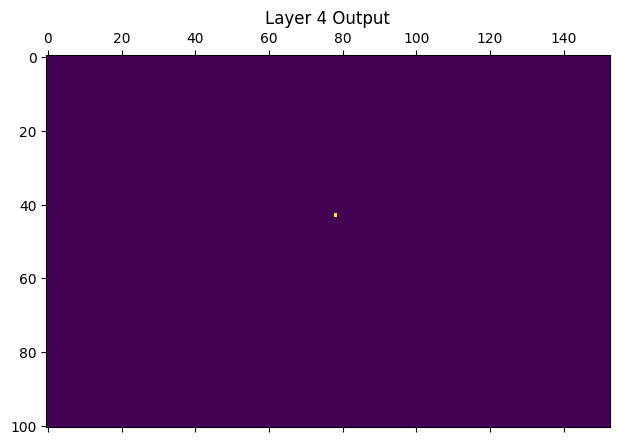
****

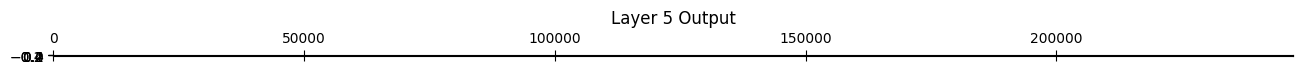
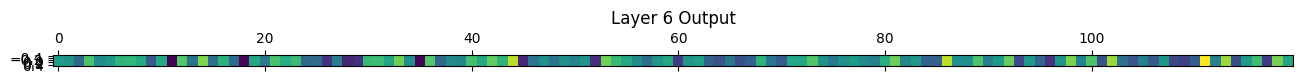
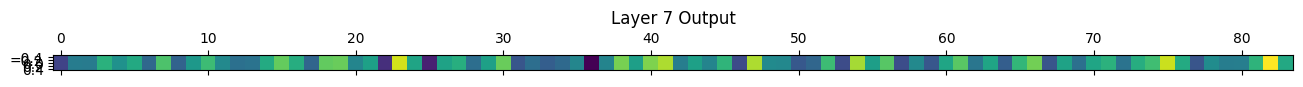
**Output Layers:**

**** **A lion walking in the grass

Description automatically generated**

A purple rectangle with white text

Description automatically generated 

   A yellow and purple squares

Description automatically generated

**B.2 Observations and learning:**

1. Input Image: The initial image is what I used as input for my Convolutional Neural Network (CNN) experiment.
2. Convolution Layer 1: After passing through the first convolutional layer with 8 filters of size 5x5 and no padding, the image underwent feature extraction. This layer highlighted various patterns and edges present in the image.
3. Pooling Layer 1: The output from the first convolutional layer was then downsampled using max pooling with a 2x2 mask. This reduced the dimensionality of the features while retaining their essential information.
4. Convolution Layer 2: Following the first pooling layer, the image passed through the second convolutional layer with 16 filters of size 5x5. This layer further extracted more complex features from the image.
5. Pooling Layer 2: Similar to the first pooling layer, the output from the second convolutional layer was downsampled using max pooling with a 2x2 mask, reducing the dimensionality of the features.
6. Fully Connected Layer 1: The output from the second pooling layer was then flattened and fed into a fully connected layer consisting of 120 neurons. This layer helped in learning higher-level features by connecting all the extracted features.
7. Fully Connected Layer 2: Following the first fully connected layer, the output was passed through another fully connected layer with 84 neurons. This further refined the learned features.
8. Softmax Layer: Finally, the output from the second fully connected layer was passed through a softmax layer with 10 neurons, which produced probabilities for each cl**ass.**

**B.3 Conclusion:**

Through this experiment, I observed the step-by-step transformation of an input image as it traversed through the layers of a CNN. The convolutional layers acted as feature extractors, capturing different patterns and structures within the image. The pooling layers helped in reducing the dimensionality of the features, making the computation more efficient while retaining essential information. The fully connected layers further processed the features, learning complex patterns and relationships within the data. Finally, the softmax layer provided the probability distribution over the output classes. Overall, this experiment provided insights into how CNNs process images and extract meaningful information for classification tasks.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*