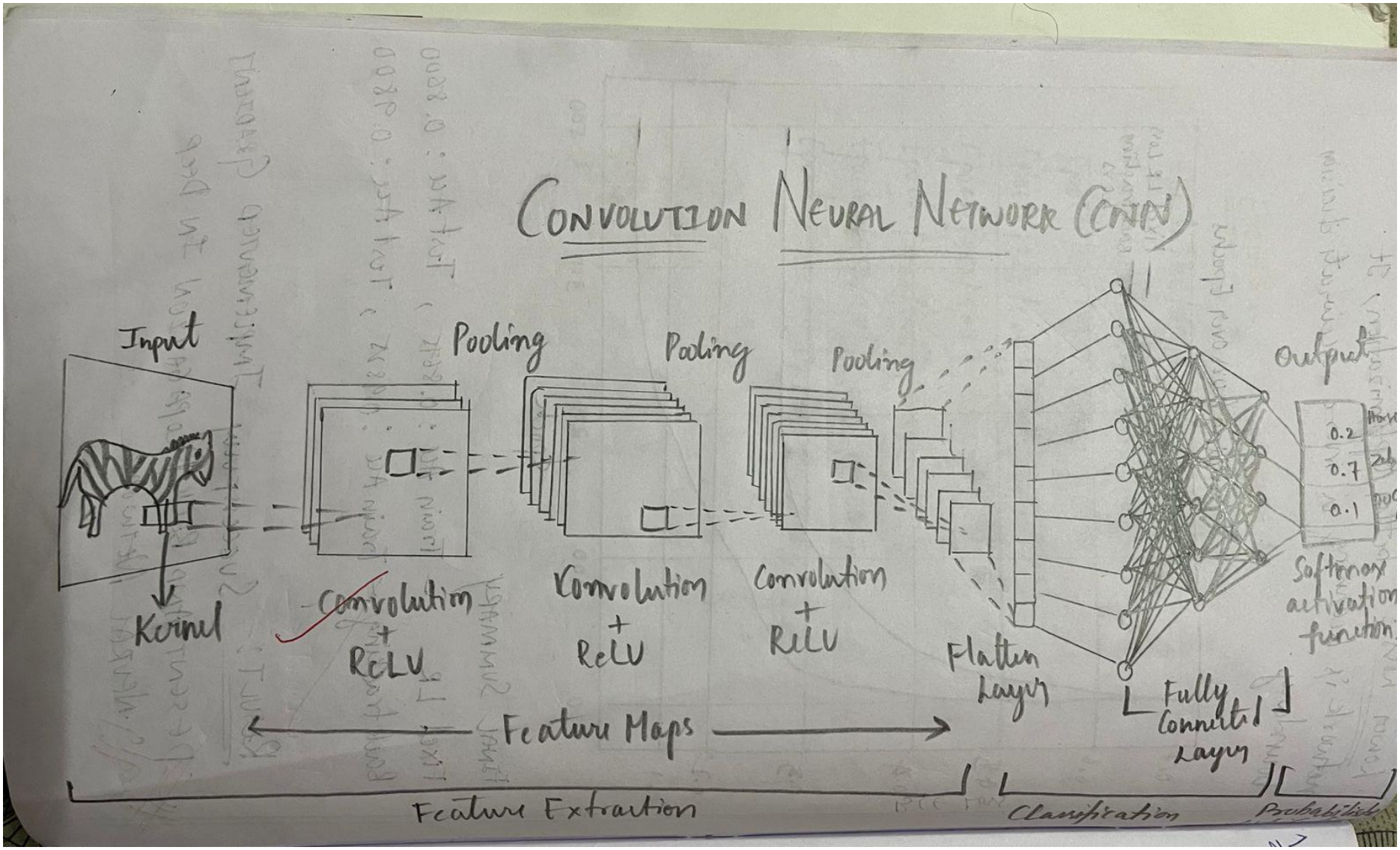
NAME: K. Sumanth B TECH-A-I-"A"

STD: 3rd Year DIV: ROLL NO.: RA2311047010036

SUBJECT: DEEP LEARNING

INDEX

SR. NO.	DATE	TITLE	Marks PAGE NO.	TEACHER'S SIGN
1	24/7/25	Exploring the Deep learning		
2	31/07/25	Implement a classifier using Open Source		7/8/25
3	31/07/25	Study of the classifier with respect to statistical parameters.		14/8/25
4	14/08/25	Build a simple feed forward neural network to recognize handwritten characters. (MNIST DATASET)		
5	22/08/25	STUDY OF ACTIVATION FUNCTIONS AND THEIR ROLE		
6	9/09/25	IMPLEMENT GRADIENT DESCENT AND BACK PROPAGATION IN DEEP NEURAL NETWORK		9/9
7	16/09/25	Build a CNN model to classify Cat and Dog image.		23/9/25


CONVOLUTION NEURAL NETWORK (CNN)
Input
Pooling
Pooling
Pooling
Output
0.2
0.7
0.1
Softmax activation function
Kernel
Convolution + ReLU
Convolution + ReLU
Convolution + ReLU
Flatten Layer
Fully Connected Layer
Feature Maps
Feature Extraction
Classification
Probability

10/9/25 LAB 1

BUILD A CNN MODEL TO CLASSIFY CAT AND DOG IMAGE

AIM:- To build and train a convolutional neural network (CNN) model that classifies images into cat or Dog categories

OBJECTIVE:-

1) To understand the working CNN for image classification
2) To preprocess and normalize image data for efficient learning
3. To design and implement a CNN architecture with convolutional, pooling and fully connected layers.
4. To train the CNN model on the cat and Dog dataset
5. To evaluate the model using accuracy and loss metrics.

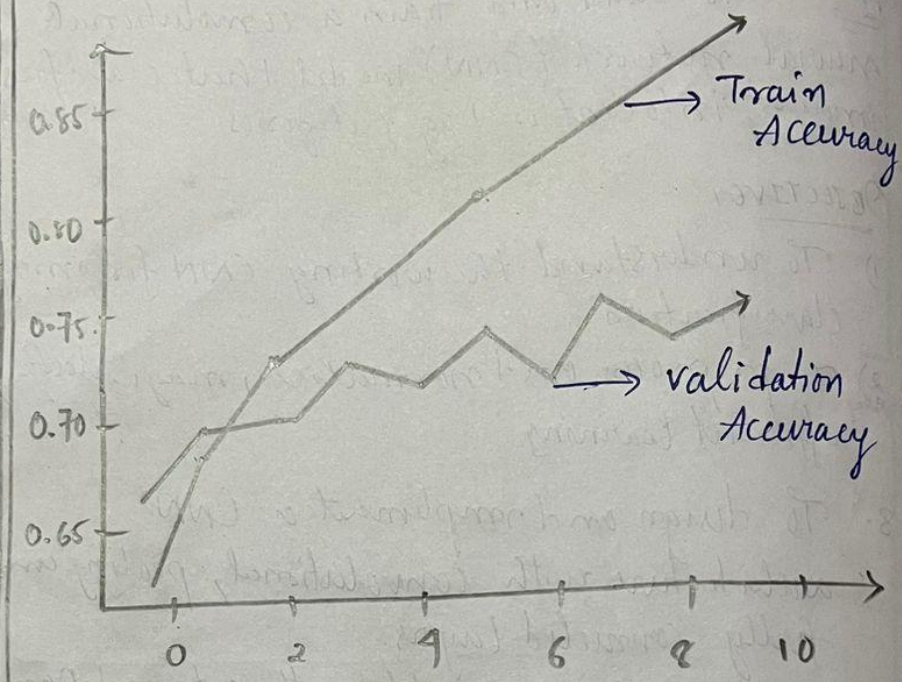
PSEUDOCODE:-

1. Import required libraries (Tensor Flow / Keras / Numpy, matplotlib)
2. Load dataset (cat vs dog).
3. preprocess data:
   - → Normalize pixel values (0-1)
   - → Split into train and test set.

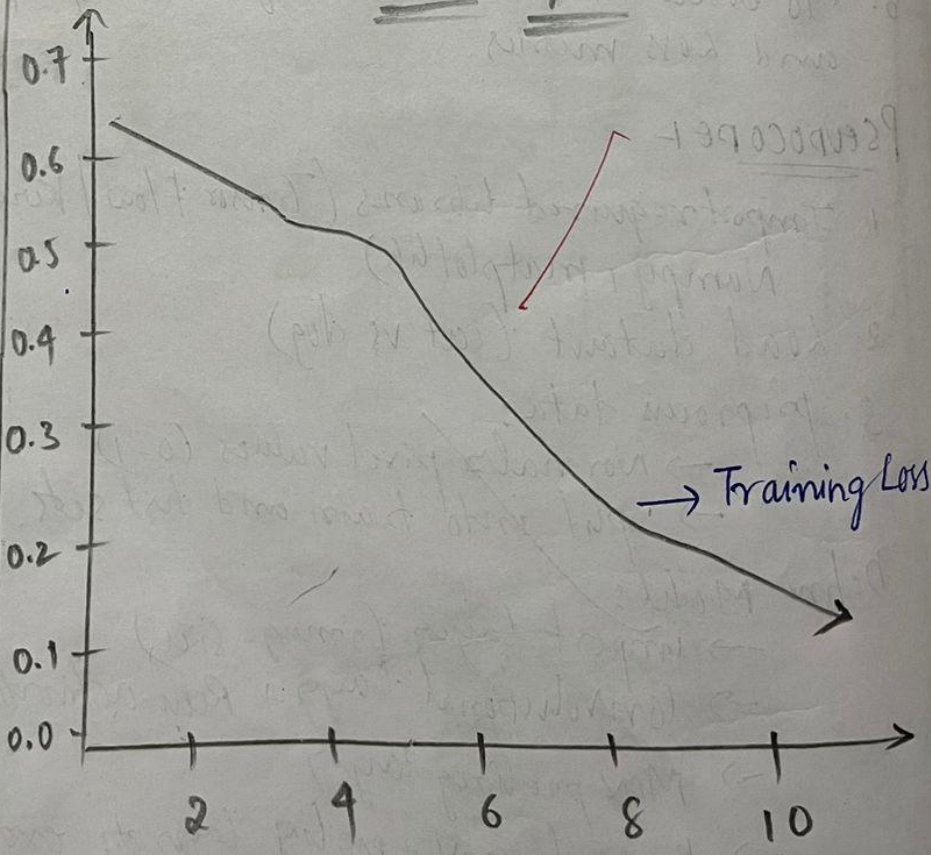
Define Model:

- → import layers (image size)
- → Convolutional Layer + Relu activation
- → Max pooling layer
- → Repeat conv + pooling layers to extract

Accuracy GRAPH



Loss GRAPH



5. Compile model with:
   → Loss function = Binary cross entropy
   → optimizer = adam
   → Metrics = Accuracy
6. Train model on training data
7. validate using test / validation data
8. Evaluate model performance (accuracy, Loss)
9. Save the model for future predictions

END.

OBSERVATION

→ Training Accuracy improved by 96.02%.
→ validation accuracy in 10 epochs
→ Average loss : 0.0620

Train Accuracy : 96.02% / validation Acc = 84.88

Epoch [1/10] → Accuracy training Loss → 0.5626
Training Accuracy - 63.61%
Validation Accuracy - 71.95%

Epoch [2/10] → Accuracy training Loss - 0.5198
Training Accuracy - 72.76%
validation Accuracy - 77.26%

Epoch [3/10] → Accuracy training Loss - 0.4556
Training Accuracy - 77.42%
Validation Accuracy - 80.38%

Epoch [9/10] → Accuracy training Loss - 0.3910
Training Accuracy - 81.21%.
validation Accuray - 83.36%.

Epoch [9/10] → Accuray training Loss - 0.1187
Training Accuray - 94.86%.
validation Accuray - 84.53%.

Epoch [10/10] → Accuray training Loss - 0.0872
Training Accuray - 96.02%.
validation Accuray - 84.65%.

RESULT :-

Successfully implemented cat vs dog CNN.