

23/9/25

WEEK-7

Aim: To develop and implement a convolutional Neural Net-work model that classifies images of cats and dogs from the CIFAR-10 dataset.

Objectives:-

- 1) To preprocess the dataset and filter only cat and dog images.
- 2) To design a CNN architecture for binary classification.
- 3) To train the CNN and validate its performance using accuracy and loss metrics.
- 4) To visualize training and validation accuracy/loss.
- 5) To evaluate the model on unseen test data and analyze prediction outcomes.

Algorithm:-

- Import necessary libraries (tensorflow, numpy, matplotlib)
- Load the cats-vs-dogs dataset.
- Extract only classes labeled as cat and dog.
- Normalize image pixel values to range $[0,1]$.
- Define the CNN model architecture:
 - convolution → ReLU → max pooling layers
 - flatten → Dense hidden layers
 - Dropout → output layer
- Compile the model with Adam optimizer and binary crossentropy loss.
- Train the model with training data.
- Evaluate the model on test dataset and calculate accuracy.

Observation :-

- * The CNN model was trained on 50% of the cats vs dogs dataset and validated on 10% of the data.
- * Images were resized to 128×128 pixels and normalized before training.
- * After 5 epochs of training, the model achieved an improving accuracy trend on both the training and validation datasets.
- * The model's binary classification output (dog vs cat) was tested on individual images, showing correct predictions most of the time.
- * Visualization of sample test images confirmed the model could distinguish b/w cats and dogs with reasonable confidence.

Conclusion :-

- A relatively simple CNN architecture with two convolutional layers is capable of learning useful features to classify images with limited training.
- preprocessing steps like resizing and normalization are essential to prepare images for training.
- This experiment highlights the foundational steps of image classification pipelines: data preparation, model building, training, evaluation and visualization of the predictions.

Output :-

Epoch 1/5

364/364

accuracy :- 0.5127 - loss : 0.7406

Epoch 2/5

364/364

accuracy :- 0.6111 - loss : 0.6670

Epoch 3/5

364/364

accuracy :- 0.7371 - loss : 0.5310

Epoch 4/5

364/364

accuracy :- 0.7931 - loss : 0.4893

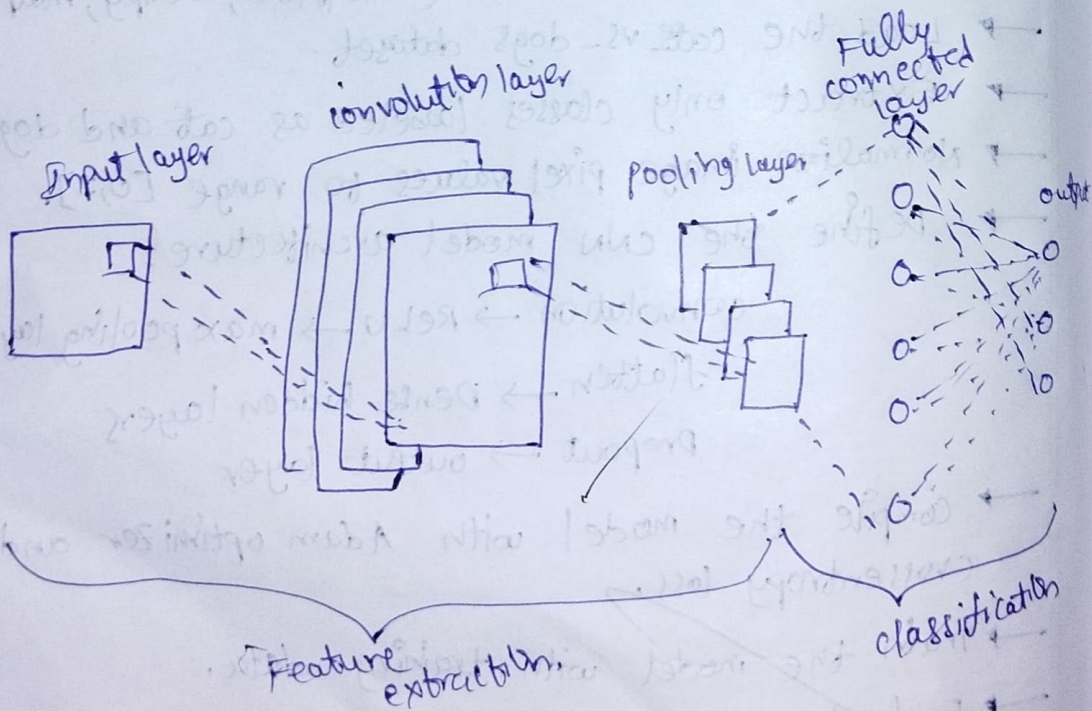
Epoch 5/5

364/364

accuracy :- 0.8365 - loss : 0.3742

Predicted : cat

True : cat





```
[ ]
import tensorflow as tf
import tensorflow_datasets as tfds
import matplotlib.pyplot as plt
splits = ['train[:50%]', 'train[50:60%]']
datasets, info = tfds.load('cats_vs_dogs',
                           split=splits,
                           as_supervised=True,
                           with_info=True)
train, test = datasets[0], datasets[1]

def process(image, label):
    image = tf.image.resize(image, (128, 128))
    image = image / 255.0
    return image, label
train = train.map(process).batch(32).shuffle(1000)
test = test.map(process).batch(32)
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, 3, activation='relu', input_shape=(128, 128, 3)),
    tf.keras.layers.MaxPooling2D(),

    tf.keras.layers.Conv2D(64, 3, activation='relu'),
    tf.keras.layers.MaxPooling2D(),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.fit(train, epochs=5, validation_data=test)
for images, labels in test.take(1):
    img = images[0].numpy()
    true_label = labels[0].numpy()
    img_expanded = tf.expand_dims(img, 0)
    prediction = model.predict(img_expanded)[0][0]
    pred_label = "Dog" if prediction > 0.5 else "Cat"
    true_label_name = "Dog" if true_label == 1 else "Cat"
    plt.imshow(img)
    plt.title(f"Predicted: {pred_label}\nTrue: {true_label_name}")
    plt.axis('off')
    plt.show()
```



Commands + Code + Text ▶ Run all ▼

[]



```
plt.axis('off')  
plt.show()
```



Epoch 1/5

364/364 ————— 277s 718ms/step - accuracy: 0.5127 - loss: 0.7406 - val_accuracy: 0.5632 - val_loss: 0.6816

Epoch 2/5

364/364 ————— 277s 723ms/step - accuracy: 0.6111 - loss: 0.6670 - val_accuracy: 0.7180 - val_loss: 0.5816

Epoch 3/5

364/364 ————— 273s 709ms/step - accuracy: 0.7371 - loss: 0.5310 - val_accuracy: 0.7558 - val_loss: 0.4878

Epoch 4/5

364/364 ————— 332s 738ms/step - accuracy: 0.7931 - loss: 0.4393 - val_accuracy: 0.7842 - val_loss: 0.4546

Epoch 5/5

364/364 ————— 320s 730ms/step - accuracy: 0.8365 - loss: 0.3742 - val_accuracy: 0.7700 - val_loss: 0.4932

1/1 ————— 0s 108ms/step

Predicted: Cat

True: Cat

