

16.9.25

## 7. Build a CNN model to classify cat and dog image

task:

To build a Convolutional Neural Network model that classifies images as either cats or dogs.

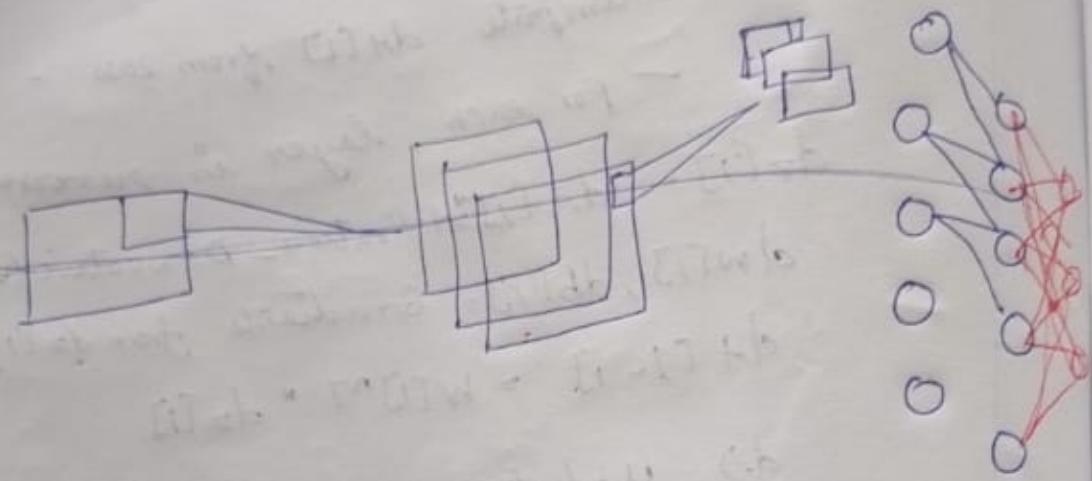
objective:

preprocess and load the dataset of cat and dog images.

Design a CNN architecture to extract features and classify images.

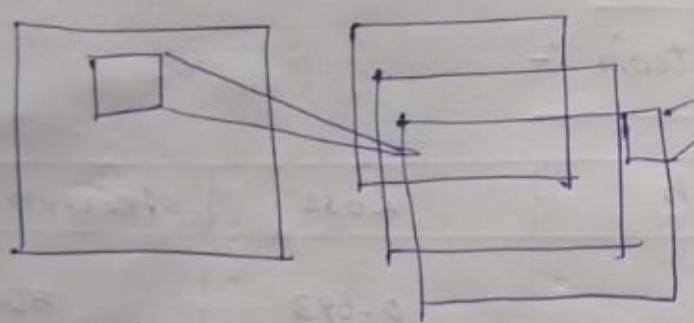
Train a model and evaluate its performance.

Achieve high accuracy on the validation / test dataset.



CNN architecture.

Pooling



T/P

Convolution

Feature Extraction

Classification

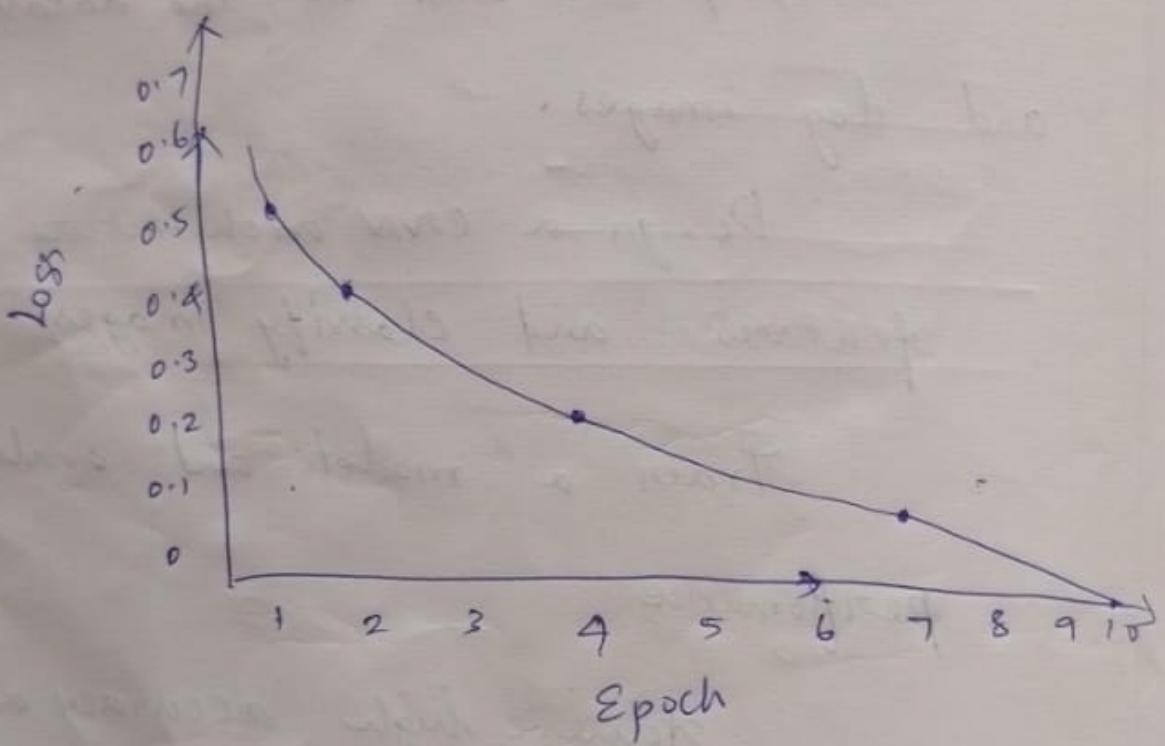
- 1- Model starts learning basic features.
- 2- Accuracy improves; loss decreases.
- 4- Model captures more complex patterns.
- 7- Validation accuracy saturates; slight overfit.
- 10- Training accuracy high validation plateaus.

Overall, model performs well with about 87-88% validation accuracy after 10 epochs.

### Result :-

~~Not successful~~ Successfully built the CNN model to classify cat and dog image.

Loss curve



Untitled3.ipynb

```
# Compile model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Train
history = model.fit(
    train_data,
    validation_data=val_data,
    epochs=5
)

# Save model
model.save("cat_dog_cnn.h5")

# Plot accuracy
plt.plot(history.history['accuracy'], label='train acc')
plt.plot(history.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
```

Found 2000 files belonging to 2 classes.  
Found 1000 files belonging to 2 classes.  
Epoch 1/5  
63/63 72s 1s/step - accuracy: 0.5173 - loss: 0.7893 - val\_accuracy: 0.6100 - val\_loss: 0.6889  
Epoch 2/5  
63/63 70s 1s/step - accuracy: 0.5791 - loss: 0.6857 - val\_accuracy: 0.5730 - val\_loss: 0.6648

← → ⌂ colab.research.google.com/drive/1zvS32HhcV8LeBpuvj6ni-qO8EfOEI- Guest

Untitled3.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all Connect

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing import image_dataset_from_directory
import matplotlib.pyplot as plt
import os, zipfile

# Download dataset
dataset_url = "https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip"
zip_path = tf.keras.utils.get_file("cats_and_dogs_filtered.zip", origin=dataset_url)

# Extract dataset
with zipfile.Zipfile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(os.path.dirname(zip_path))

# Correct base directory
base_dir = os.path.join(os.path.dirname(zip_path), "cats_and_dogs_filtered")
train_dir = os.path.join(base_dir, "train")
val_dir = os.path.join(base_dir, "validation")

# Image size and batch size
IMG_SIZE = (150, 150)
BATCH_SIZE = 32

# Load datasets
train_data = image_dataset_from_directory(
    train_dir,
    image_size=IMG_SIZE,
```

Variables Terminal

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** Untitled3.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help.
- Header Buttons:** Share, Gemini, K.
- Left Sidebar:** Commands, Code, Text, Run all, Connect.
- Code Cell:** Contains Python code for data loading, normalization, and building a CNN model.

```
ImageSize=IMG_SIZE,
batch_size=BATCH_SIZE
)
val_data = image_dataset_from_directory(
    val_dir,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE
)

# Normalize pixel values
normalization_layer = tf.keras.layers.Rescaling(1./255)
train_data = train_data.map(lambda x, y: (normalization_layer(x), y))
val_data = val_data.map(lambda x, y: (normalization_layer(x), y))

# Build CNN model
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid') # Binary classification
])
```

## pseudocode

- Load and preprocess images
- Build CNN:
- Conv + ReLU + Maxpooling layers
- Flatten
- Dense + Sigmoid output
- Compile model with binary cross-entropy loss
- Train model on training data
- Evaluate on validation data
- print accuracy

## Observation :-

epoch	Training Accuracy	Validation accuracy	Training loss	Validation loss
1	0.72	0.70	0.58	0.60
2	0.81	0.78	0.42	0.45
4	0.89	0.85	0.28	0.32
7	0.94	0.88	0.16	0.28
10	0.96	0.87	0.10	0.30