Image Classification Of Cats and Dogs using CNN

Presented by:

Utkarsh Saxena

B.Tech(CSE)

MIT, Moradabad

Dataset & Preprocessing

- **Dataset:**
  Kaggle's "Dogs vs Cats" dataset (25,000 labeled images).
- **Download Method:**
  Used opendatasets to directly fetch dataset from Kaggle.
- **Directory Structure:**

1. /train: contains cat and dog images
2. /test: used for validation

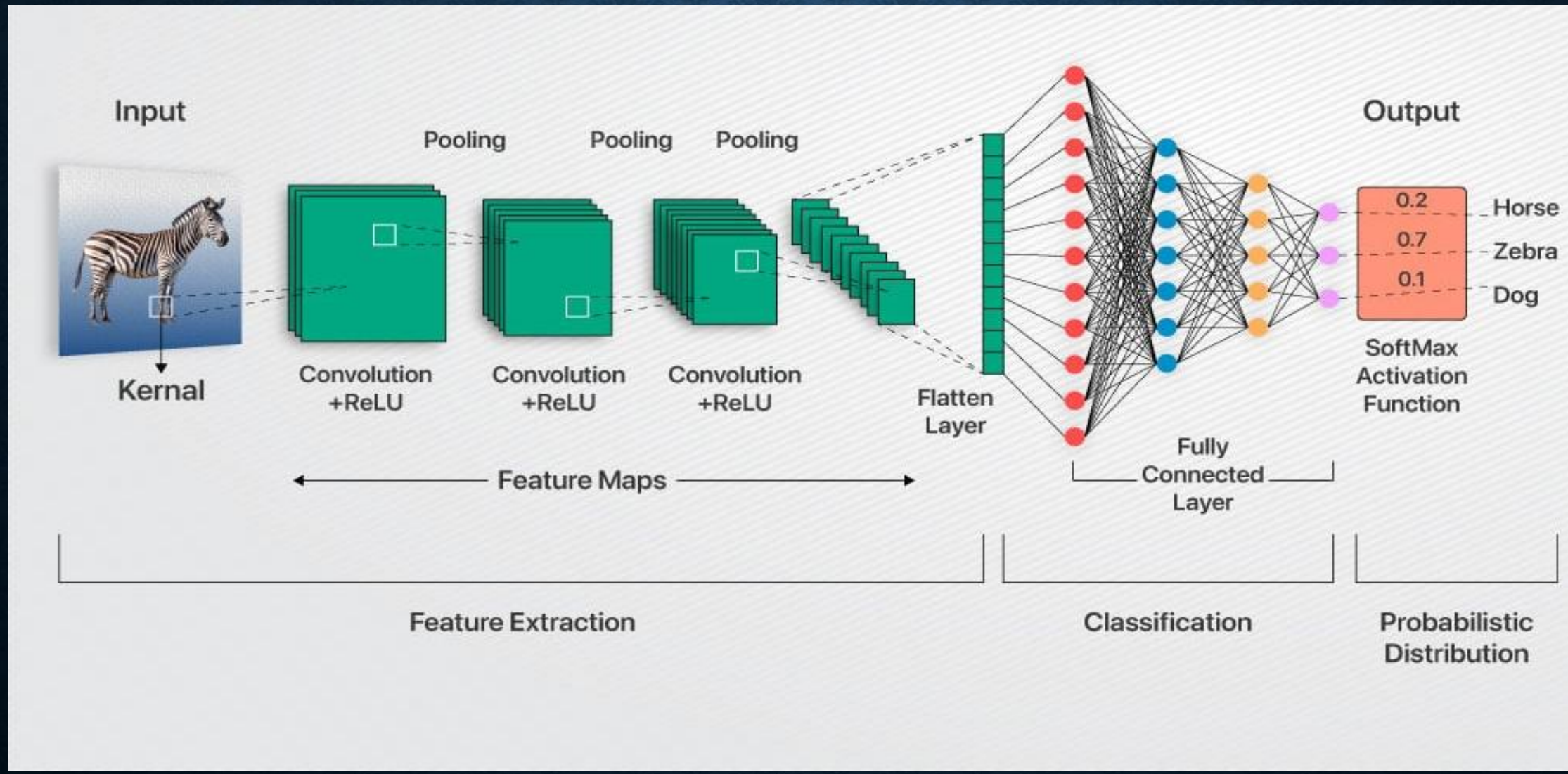
- **Image Loading:**
  Used image_dataset_from_directory() from Keras with:

1. image_size=(256, 256)
2. batch_size=32
3. label_mode='int'

- **Preprocessing:**

1. Normalized pixel values to range [0, 1] using tf.cast()
2. Applied .map() function to transform training and validation datasets

CNN Model Architecture



Layer Type	Details	Purpose
Input Layer	Input shape = (256, 256, 3)	Accepts RGB image
Conv2D	32 filters, kernel size (3x3), ReLU activation	Feature extraction
BatchNormalization	After Conv2D	Normalizes and speeds up training
MaxPooling2D	Pool size (2x2)	Reduces feature map size
Conv2D	64 filters, kernel size (3x3), ReLU activation	Deeper feature extraction
BatchNormalization	After Conv2D	Stabilizes learning
MaxPooling2D	Pool size (2x2)	Downsamples features
Conv2D	128 filters, kernel size (3x3), ReLU activation	High-level feature extraction
BatchNormalization	After Conv2D	Enhances stability
MaxPooling2D	Pool size (2x2)	Reduces dimensions
Flatten	Converts 2D to 1D	Prepares for dense layers
Dense	128 neurons, ReLU activation	Fully connected layer
Dropout	Dropout rate 0.1	Prevents overfitting
Dense	64 neurons, ReLU activation	Second dense layer
Dropout	Dropout rate 0.1	Adds regularization
Output (Dense)	1 neuron, Sigmoid activation	Binary output: Dog (1), Cat (0)

Model Compilation & Training

- **Training Details**

1. **Function Used:** model.fit()
2. **Epochs:** 3
3. **Batch Size:** 32
4. **Validation Data:** Used a separate test set for validation

- **Goal:** Maximize classification accuracy (Cat vs Dog)

Why These Choices?

1. **Adam Optimizer:** Fast and adaptive learning
2. **Binary Crossentropy:** Best suited for binary classification
3. **Validation Set:** Helps monitor overfitting

Accuracy and Loss Visualization

- **Training Progress Evaluation**

Plot Type	Description
Accuracy	Shows how well the model predicts over time
Loss	Measures model error during training

- **What We Did?**

1. Used matplotlib.pyplot to visualize:
   - **Training vs Validation Accuracy**
   - **Training vs Validation Loss**
2. Extracted values from history.history dictionary

- **Observations**

1. Accuracy **increased** with each epoch
2. Loss **decreased**, indicating improved learning
3. Validation curves followed training trends → minimal overfitting

Image Prediction Logic

- Steps to Predict an Image:

Step	Action
1. Load	Used cv2.imread() to read image file (e.g., dog.webp)
2. Resize	Resized image to (256x256) using cv2.resize()
3. Reshape	Converted to (1, 256, 256, 3) for model input
4. Normalize	Scaled pixel values to [0, 1]
5. Predict	Used model.predict() to get output value
6. Interpret	If output ≥ 0.5 → **Dog**, else → **Cat**

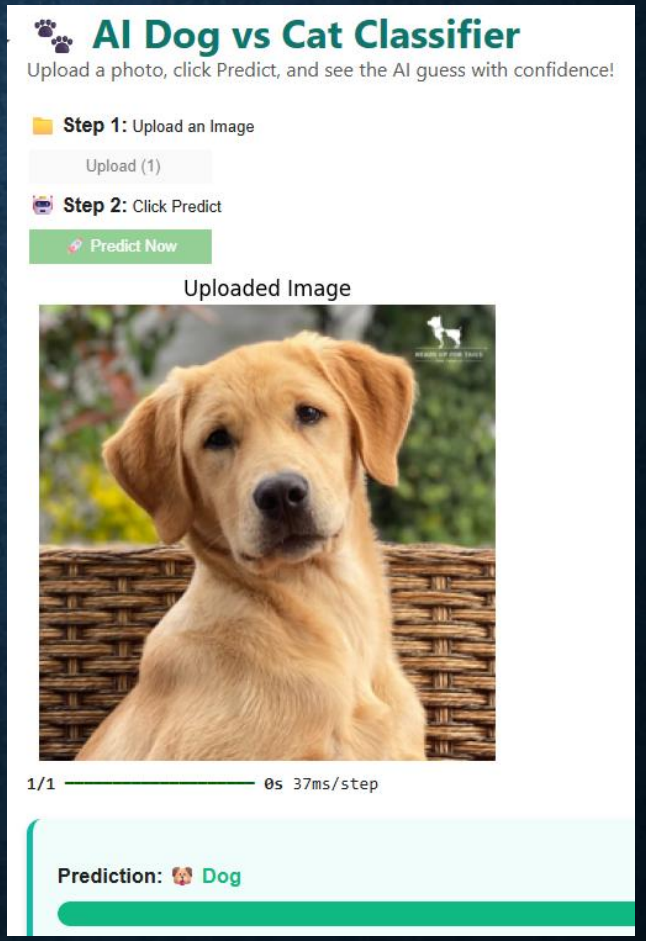
Interactive UI in Colab

- **Interface Design**

1. Used **ipywidgets** for buttons and layout
2. Styled using **HTML + CSS** for a clean look
3. Output displayed in a **"card-style box"** with confidence bar

- **User Experience**

I. Simple 2-step process:
   1.Upload Image
   2.Click “🚀 Predict Now”
II. Real-time prediction with visual feedback



Conclusion

In this project, we successfully developed a Convolutional Neural Network (CNN) model to classify images of cats and dogs with high accuracy. The model was trained using the Kaggle “Dogs vs Cats” dataset and achieved reliable performance in just 3 epochs. To enhance usability, an interactive user interface was built using ipywidgets in Google Colab, allowing users to upload an image and receive real-time predictions along with a confidence percentage. The trained model was saved in both .h5 and .keras formats, making it ready for future use. Overall, this project provides a solid foundation for further improvements and potential deployment as a web or mobile application.