**🖼️ Multiple Image Classification Using Deep Learning**

**📌 Project Content**

This project implements a deep learning-based image classification system capable of identifying multiple classes of images. Using a convolutional neural network (CNN), the system learns patterns from input images and accurately predicts their categories. It is ideal for applications like object recognition, automated sorting, and content tagging.

---

**🧾 Project Code**

```python

# Sample: Load model and predict

from tensorflow.keras.models import load\_model

import numpy as np

from tensorflow.keras.preprocessing import image

model = load\_model('my\_model.h5')

img = image.load\_img('test\_image.jpg', target\_size=(64, 64))

img\_array = image.img\_to\_array(img)

img\_array = np.expand\_dims(img\_array, axis=0)

result = model.predict(img\_array)

For full code, see the Jupyter Notebook or script files in the repo.

**Key Technologies:**

🧠 TensorFlow & Keras – for model building

📦 NumPy & Pandas – data manipulation

🧪 Scikit-learn – evaluation metrics

🖼️ OpenCV – image preprocessing

📊 Matplotlib & Seaborn – data visualization

🗃️ Google Colab / Jupyter – notebook interface

📄 Description

This image classifier accepts a dataset of images categorized into folders (one per class). The model is trained using convolutional layers followed by max pooling and dense layers. It supports multi-class classification using categorical crossentropy and a softmax activation in the final layer.

The typical architecture looks like this:

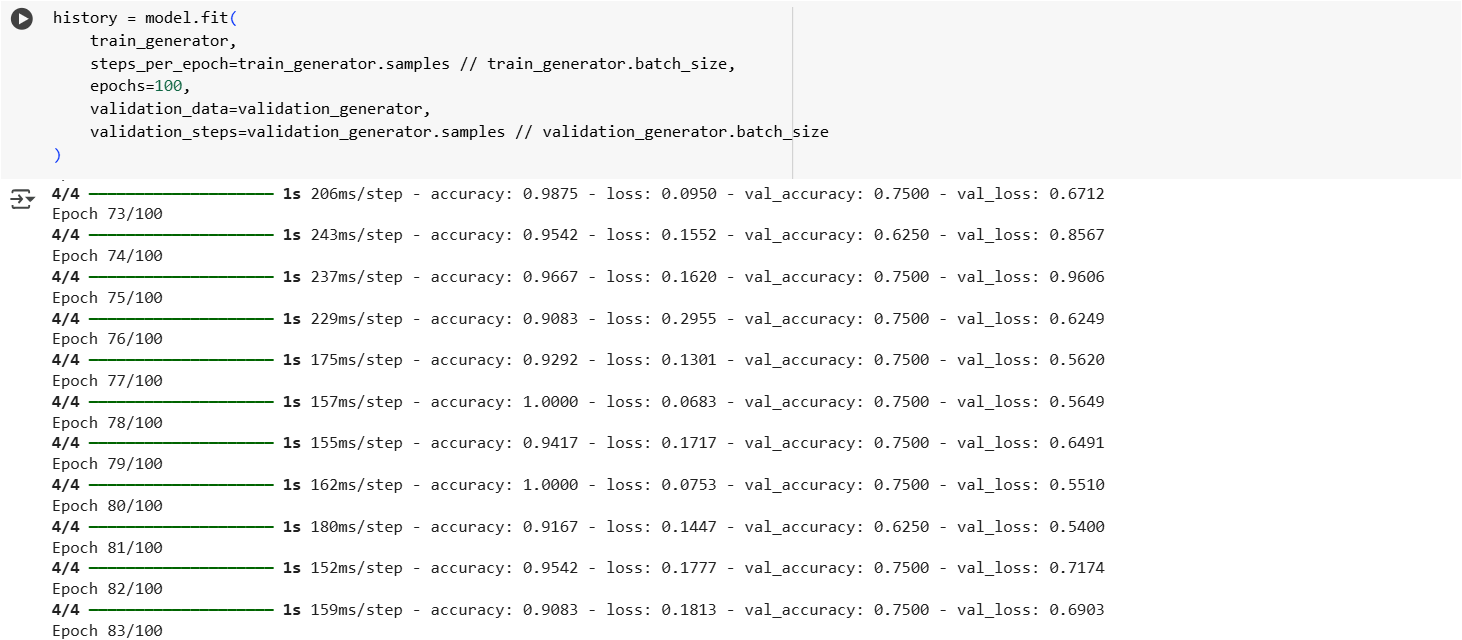
Conv2D → ReLU → MaxPooling

Conv2D → ReLU → MaxPooling

Flatten → Dense → Dropout → Output

Training history (loss/accuracy) is plotted and confusion matrices are generated to evaluate the model's performance.

✅ Dataset expansion with data augmentation

📸 Sample Screenshots



**🖼️ Input Type**

Images are loaded from a structured folder where each subdirectory corresponds to a class label.

dataset/

├── cats/

├── dogs/

├── horses/

└── elephants/

Each folder contains relevant .jpg images. The dataset is split into training and testing using ImageDataGenerator.

**✅ Output**

* 🎯 Predicted class label with probability score
* 📈 Accuracy and loss curves
* 📉 Confusion matrix and classification report
* 💾 Saved model (.h5) and label encodings

**Example Output:**

**Predicted Class: Dog**

**Confidence Score: 97.2%**

**✨ Unique Features**

* **Supports any number of categories by folder naming**
* **Automatically resizes and normalizes images**
* **Visual output: accuracy/loss plots, confusion matrix**
* **Easily extendable with transfer learning models like VGG16 or ResNet**
* **Clean modular notebook design**

**🔮 Further Research**

**This project can be improved or extended with:**

* **✅ Real-time classification from webcam**
* **✅ Transfer learning with MobileNet, ResNet**
* **✅ GUI using Streamlit or Tkinter**
* **✅ Deployment on cloud (Flask/Heroku)**
* **✅ Model quantization for mobile apps**
* **✅ Dataset expansion with data augmentation**

**🏁 How to Run**

**Clone this repository:**

git clone https://github.com/your-username/multi-image-classifier.git

cd multi-image-classifier

**Install dependencies:**

pip install -r requirements.txt

**Run the notebook**:

jupyter notebook main.ipynb

1. Add your dataset inside the dataset/ folder and retrain or test.

**🗂 Folder Structure**

├── dataset/

│ ├── class1/

│ ├── class2/

├── models/

│ └── my\_model.h5

├── images/

│ └── accuracy.png

│ └── confusion\_matrix.png

├── main.ipynb

├── README.md

└── requirements.txt

**🙌 Contributors**

* Mohan Charan (Developer)
* OpenAI GPT (Documentation support)