Title:
Vision AI in 5 Days: From Beginner to Image Recognition Expert

Subtitle:

Image Classification using CNN in Python

Presented by:

Gayatri Sharma

Dataset: CIFAR-10 / Cats vs Dogs

- Total images:(e.g., 50,000 train, 10,000 test)
- Processing Steps :
- Image resizing
- Normalization
- Train-Test split
- Data augmentation (rotation, flip, zoom)

Dataset Preview

```
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import numpy as np


(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0  # Normalize pixel values


class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']


model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),
```

```
    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

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

    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10)
])


model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])


history = model.fit(x_train, y_train, epochs=10,
                    validation_data=(x_test, y_test), batch_size=64)


test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"\n✅ Test Accuracy: {test_acc * 100:.2f}%")
print(f"❌ Test Loss: {test_loss:.4f}")
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a laye
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 62s 77ms/step - accuracy: 0.3299 - loss: 1.8096 - val_accuracy: 0.5238 - val_loss: 1.3540
Epoch 2/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 82s 78ms/step - accuracy: 0.5534 - loss: 1.2538 - val_accuracy: 0.6114 - val_loss: 1.0989
Epoch 3/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 80s 75ms/step - accuracy: 0.6138 - loss: 1.0870 - val_accuracy: 0.6368 - val_loss: 1.0201
Epoch 4/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 59s 75ms/step - accuracy: 0.6599 - loss: 0.9610 - val_accuracy: 0.6518 - val_loss: 0.9905
Epoch 5/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 59s 75ms/step - accuracy: 0.6861 - loss: 0.8968 - val_accuracy: 0.6723 - val_loss: 0.9581
Epoch 6/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 81s 73ms/step - accuracy: 0.7118 - loss: 0.8233 - val_accuracy: 0.6787 - val_loss: 0.9379
Epoch 7/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 60s 77ms/step - accuracy: 0.7250 - loss: 0.7764 - val_accuracy: 0.6921 - val_loss: 0.8960
Epoch 8/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 61s 78ms/step - accuracy: 0.7493 - loss: 0.7187 - val_accuracy: 0.6976 - val_loss: 0.8721
Epoch 9/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 58s 74ms/step - accuracy: 0.7611 - loss: 0.6819 - val_accuracy: 0.6922 - val_loss: 0.8925
Epoch 10/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 60s 77ms/step - accuracy: 0.7741 - loss: 0.6457 - val_accuracy: 0.7126 - val_loss: 0.8589
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - accuracy: 0.7187 - loss: 0.8484

✅ Test Accuracy: 71.26%
❌ Test Loss: 0.8589
```

Model Architecture:

- **Model Type**: Convolutional Neural Network (CNN)
- **Layers**: Conv → Pool → Conv → Pool → Dense → Output
- **Activation Functions**: ReLU, Softmax
- **Optimizer**: Adam
- **Loss Function**: Categorical Crossentropy

Results & Evaluation

- **Training Accuracy**: ~77%
- **Validation Accuracy**: ~71%
- **Test Accuracy**: 71.26%
- **Test Loss**: 0.8589
- **Visuals**:
  - Accuracy & Loss curves (Matplotlib se plot)
  - Confusion matrix (agar banayi ho)
- Optional: Precision, Recall, F1-score table.

- Conclusion & Future Scope

Conclusion:

- CNN model successfully trained for image classification.
- Achieved 71% accuracy on test dataset.
- **Future Improvements**:
- Use Transfer Learning (MobileNetV2, ResNet)
- More data augmentation
- Hyperparameter tuning

Demo Video :

- https://drive.google.com/file/d/1SyWFwV90ObJnLJc9OjXRXUzF151w24y4/view?usp=sharing