AIM OF THE DEEP LEARNING PROJECT

AIM : To image classification model using convolutional neural networks (CNNs) to accurately identify and classify brand logos from a given dataset of logo images.

IMPORTING ALL THE REQUIRED DEPENDENCIES/LIBRARIES AND THEN UPLOADING THE DATASET

```
[ ]  import tensorflow as tf
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
     from tensorflow.keras.preprocessing.image import ImageDataGenerator
     import matplotlib.pyplot as plt
     import numpy as np
```

```
[ ]  from google.colab import files
     uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed

```
Saving archive.zip to archive.zip
```

EXTRACTING THE DATASET
AND
LISTING ALL IMAGES TO CHECK DATASET UPLOADED SUCCESSFULLY OR NOT

```
[ ]  import zipfile

     with zipfile.ZipFile("archive.zip", 'r') as zip_ref:
         zip_ref.extractall("logos_data")
```

```
import os
os.listdir("/content/logos_data/Logos")
```

```
['hp-inc-logo-vector-download-400x400.jpg',
 'playboy-tv-eps-vector-logo-400x400.png',
 'plks-pewel-mala-vector-logo-400x400.png',
 'find-us-on-facebook-logo-vector-400x400.png',
 'saint-gobain-logo-vector-download-400x400.jpg',
 'moncler-vector-logo-400x400.png',
```

READING FROM FOLDER, EXTRACTING LABELS FROM THE FILENAMES, AND SAVING THE RESULT INTO A CSV FILE

```python
import os
import pandas as pd

image_folder = '/content/logos_data/Logos'

# List all image files
image_files = os.listdir(image_folder)
image_files = [f for f in image_files if f.endswith(('.jpg', '.jpeg', '.png'))]

# Function to generate label
def extract_label(filename):
    return filename.split('_')[0].split('.')[0].capitalize()

# Create DataFrame
df = pd.DataFrame({
    'filename': image_files,
    'label': [extract_label(f) for f in image_files]
})
df['file_path'] = df['filename'].apply(lambda x: os.path.join(image_folder, x))

# Save to CSV
df.to_csv('/content/logos_data/LogoDatabase.csv', index=False)
df.head()
```

	filename	label	file_path
0	hp-inc-logo-vector-download-400x400.jpg	Hp-inc-logo-vector-download-400x400	/content/logos_data/Logos/hp-inc-logo-vector-d...
1	playboy-tv-eps-vector-logo-400x400.png	Playboy-tv-eps-vector-logo-400x400	/content/logos_data/Logos/playboy-tv-eps-vecto...
2	plks-pewel-mala-vector-logo-400x400.png	Plks-pewel-mala-vector-logo-400x400	/content/logos_data/Logos/plks-pewel-mala-vect...
3	find-us-on-facebook-logo-vector-400x400.png	Find-us-on-facebook-logo-vector-400x400	/content/logos_data/Logos/find-us-on-facebook-...
4	saint-gobain-logo-vector-download-400x400.jpg	Saint-gobain-logo-vector-download-400x400	/content/logos_data/Logos/saint-gobain-logo-ve...

TO LOAD AND PREPROCESS IMAGES FROM FOLDERS, SPLITTING THEM INTO TRAINING AND VALIDATION DATASETS WITH AUTOMATIC RESCALING AND CLASS LABELING

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory(
    '/content/logos_data',
    target_size=(64, 64),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

val_generator = train_datagen.flow_from_directory(
    '/content/logos_data',
    target_size=(64, 64),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 1148 images belonging to 1 classes.
Found 287 images belonging to 1 classes.
```

A simple neural network is defined using Keras "Sequential" and layers

```
model = Sequential([
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(train_generator.num_classes, activation='softmax')
])
```

Using the Adam optimizer and categorical cross-entropy.
Accuracy is used as the evaluation metric during training and validation.

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

```
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=10
)
```

RUNS FOR 10 EPOCHS AND STORES THE TRAINING HISTORY FOR ANALYSIS

```
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=10
)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset
  self._warn_if_super_not_called()
Epoch 1/10
/usr/local/lib/python3.11/dist-packages/keras/src/ops/nn.py:907: UserWarning: You are using a softmax over axis -1 of a tensor
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/keras/src/losses/losses.py:33: SyntaxWarning: In loss categorical_crossentropy, expecte
  return self.fn(y_true, y_pred, **self._fn_kwargs)
36/36 ━━━━━━━━━━━━━━━━━━━━ 7s 130ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 2/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 4s 110ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 3/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 92ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 4/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 4s 110ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 5/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 4s 110ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 6/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 92ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
Epoch 7/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 90ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val_accuracy: 1.0000 - val_loss: 0.0000e+00
```

THE TRAINED MODEL ON THE VALIDATION OF DATASET AND PRINTS THE ACCURACY.

```
[ ] loss, acc = model.evaluate(val_generator)
    print(f"Validation Accuracy: {acc:.2f}")
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 72ms/step - accuracy: 1.0000 - loss: 0.0000e+00
Validation Accuracy: 1.00
```

```
[ ] label_names = list(val_generator.class_indices.keys())
```

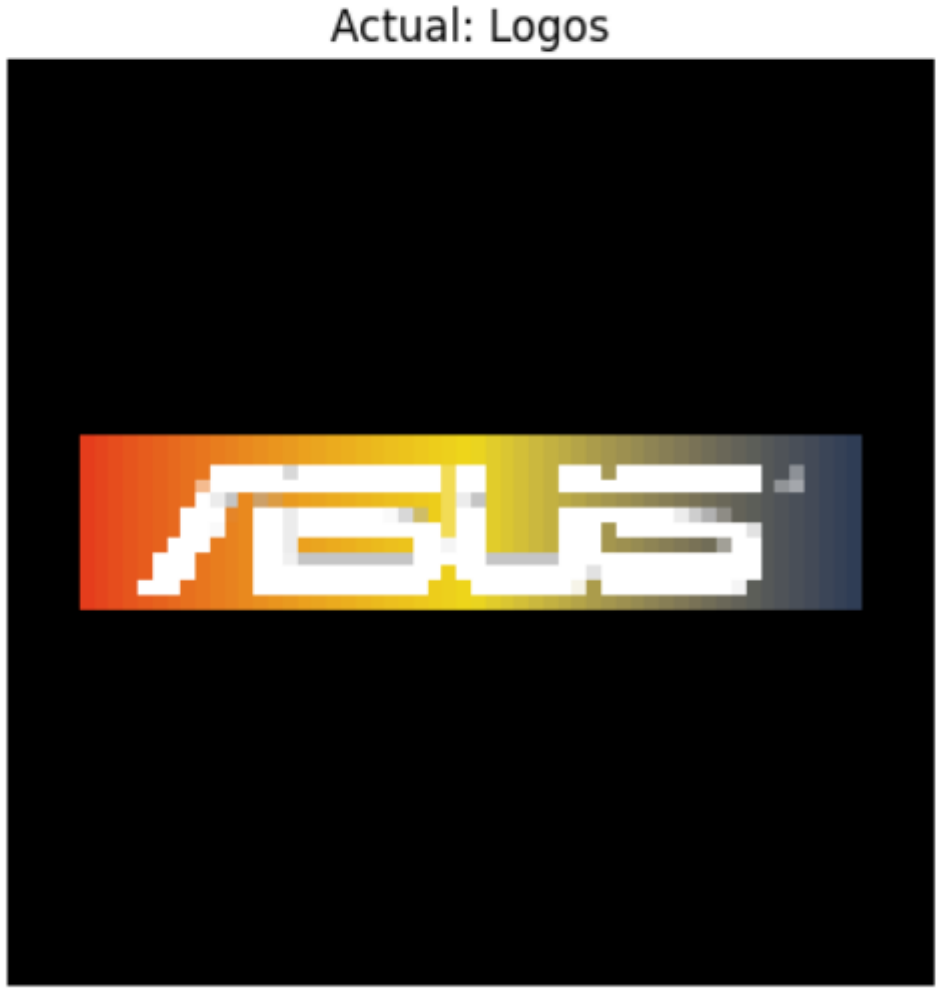
EXTRACTS THE CLASS NAMES (FOLDER NAMES) USED BY THE GENERATOR.

```python
# Pick a batch from validation set
sample_images, sample_labels = next(val_generator)

# Choose image index from batch
image_index = 0

# Predict the image
prediction = model.predict(np.expand_dims(sample_images[image_index], axis=0))[0]  # shape: (num_classes,)

# Get top 10 class indices
top_10_indices = prediction.argsort()[-10:][::-1]  # descending order

# Print top 10 predictions with probabilities
print("Top 10 Predictions:")

for i in top_10_indices:
    print(f"{label_names[i]}: {prediction[i]*100:.2f}%")

# Get actual label
actual_class = np.argmax(sample_labels[image_index])

# Visualize image
import matplotlib.pyplot as plt
plt.imshow(sample_images[image_index])
plt.title(f"Actual: {label_names[actual_class]}")
plt.axis('off')
plt.show()
```

SHOW THE **TOP 10 PREDICTIONS**, AND **VISUALIZE** THE IMAGE WITH ITS ACTUAL LABEL.

```
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
Top 10 Predictions:
Logos: 100.00%
```



Conclusion of DL

- Built a deep learning model for brand logo classification.
- Used **imagedatagenerator** for preprocessing and data splitting.
- Model included flatten, dense, and dropout layers.
- Trained with adam optimizer and categorical cross-entropy loss.
- Achieved high validation accuracy.
- Verified predictions with actual labels and image display.
- Demonstrated effective image classification using deep learning.