Day 01

What is Image Recognition?

What is Deep Learning?

What is a Dataset?

Why Preprocessing?

What is EDA (Exploratory Data Analysis)?

```
!pip install -q kaggle
```

```
from google.colab import files
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
```

```
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

```
!chmod 600 ~/.kaggle/kaggle.json
```

```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("hojjatk/mnist-dataset")

print("Path to dataset files:", path)
```

```
Path to dataset files: /kaggle/input/mnist-dataset
```

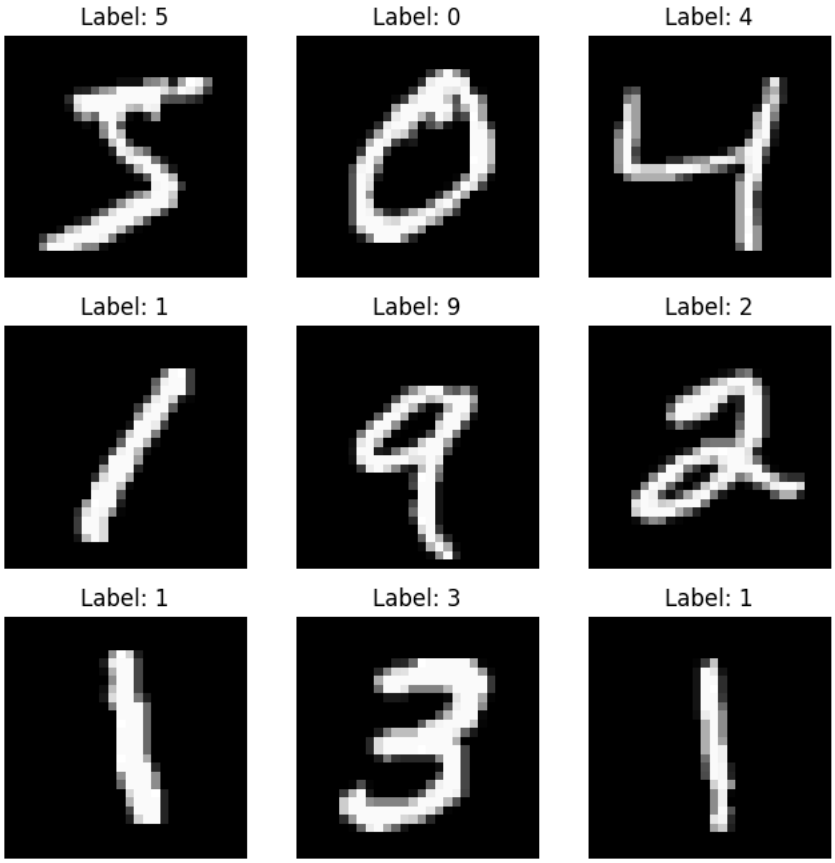
```
import tensorflow as tf
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()
print(f"Train shape: {X_train.shape}, Test shape: {X_test.shape}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
Train shape: (60000, 28, 28), Test shape: (10000, 28, 28)
```

```
X_train = X_train / 255.0
X_test = X_test / 255.0
X_train = X_train.reshape(-1, 28, 28, 1)
X_test = X_test.reshape(-1, 28, 28, 1)
print(f"Train reshaped: {X_train.shape}")
```

```
Train reshaped: (60000, 28, 28, 1)
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,8))
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(X_train[i].reshape(28,28), cmap='gray')
    plt.title(f"Label: {y_train[i]}")
    plt.axis('off')
plt.show()
```




```
(X_train_c10, y_train_c10), (X_test_c10, y_test_c10) = tf.keras.datasets.cifar10.load_data()
X_train_c10 = X_train_c10 / 255.0
X_test_c10 = X_test_c10 / 255.0
print(f"CIFAR-10 train shape: {X_train_c10.shape}")
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 12s 0us/step
CIFAR-10 train shape: (50000, 32, 32, 3)
```

DAY **02**

What is a Convolutional Neural Network (CNN)?

How does model training work?

Why do we use Dropout?

How to evaluate a model?

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

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

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_shape`/`
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential"

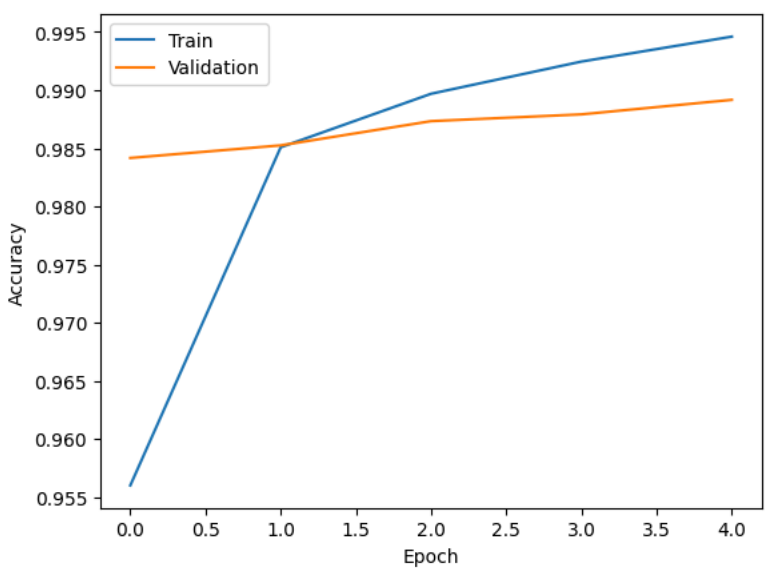
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten (Flatten)	(None, 1600)	0
dense (Dense)	(None, 128)	204,928
dense_1 (Dense)	(None, 10)	1,290

Total params: 225,034 (879.04 KB)
Trainable params: 225,034 (879.04 KB)
Non-trainable params: 0 (0.00 B)

```
history = model.fit(X_train, y_train, epochs=5, batch_size=32, validation_split=0.2)
```

```
Epoch 1/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 51s 33ms/step - accuracy: 0.9012 - loss: 0.3338 - val_accuracy: 0.9842 - val_loss: 0.0540
Epoch 2/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 81s 32ms/step - accuracy: 0.9845 - loss: 0.0486 - val_accuracy: 0.9852 - val_loss: 0.0477
Epoch 3/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 82s 32ms/step - accuracy: 0.9896 - loss: 0.0324 - val_accuracy: 0.9873 - val_loss: 0.0428
Epoch 4/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 45s 30ms/step - accuracy: 0.9931 - loss: 0.0205 - val_accuracy: 0.9879 - val_loss: 0.0404
Epoch 5/5
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 86s 33ms/step - accuracy: 0.9953 - loss: 0.0150 - val_accuracy: 0.9892 - val_loss: 0.0381
```

```
plt.plot(history.history['accuracy'],label='Train')
plt.plot(history.history['val_accuracy'],
         label='Validation')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```




```
test_loss,test_acc=model.evaluate(X_test,y_test)
print("Test Accuracy:{test_acc:.4f}")
y_pred=model.predict(X_test).argmax(axis=1)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - accuracy: 0.9854 - loss: 0.0466
Test Accuracy:{test_acc:.4f}
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step
```

```
from tensorflow.keras.layers import Dropout

model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    MaxPooling2D(2,2),
    Dropout(0.25),
```

```
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Day **03**

Why deeper CNNs?

What is Data Augmentation?

What is Advanced Evaluation Metrics?

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen= ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
)
datagen.fit(X_train_c10)
```

```
from tensorflow.keras.layers import BatchNormalization
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(32,32,3)),
    BatchNormalization(),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
model.fit(datagen.flow(X_train_c10,y_train_c10,batch_size=32),epochs=10,validation_data=(X_test_c10,y_test_c10))
```

```
Epoch 1/10
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` cl
  self._warn_if_super_not_called()
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 127s 80ms/step - accuracy: 0.3484 - loss: 1.7747 - val_accuracy: 0.5228 - val_loss: 1.3857
Epoch 2/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 124s 80ms/step - accuracy: 0.5349 - loss: 1.3222 - val_accuracy: 0.6173 - val_loss: 1.0876
Epoch 3/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 123s 79ms/step - accuracy: 0.5883 - loss: 1.1730 - val_accuracy: 0.6443 - val_loss: 1.0223
Epoch 4/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 122s 78ms/step - accuracy: 0.6151 - loss: 1.1000 - val_accuracy: 0.6462 - val_loss: 1.0270
Epoch 5/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 126s 80ms/step - accuracy: 0.6336 - loss: 1.0565 - val_accuracy: 0.6510 - val_loss: 0.9948
Epoch 6/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 124s 79ms/step - accuracy: 0.6472 - loss: 1.0176 - val_accuracy: 0.6584 - val_loss: 0.9818
Epoch 7/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 127s 82ms/step - accuracy: 0.6598 - loss: 0.9810 - val_accuracy: 0.6858 - val_loss: 0.9118
Epoch 8/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 123s 79ms/step - accuracy: 0.6651 - loss: 0.9626 - val_accuracy: 0.7092 - val_loss: 0.8480
Epoch 9/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 129s 82ms/step - accuracy: 0.6743 - loss: 0.9376 - val_accuracy: 0.6655 - val_loss: 0.9706
Epoch 10/10
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 125s 80ms/step - accuracy: 0.6847 - loss: 0.9156 - val_accuracy: 0.6950 - val_loss: 0.9005
<keras.src.callbacks.history.History at 0x794724b1fd50>
```

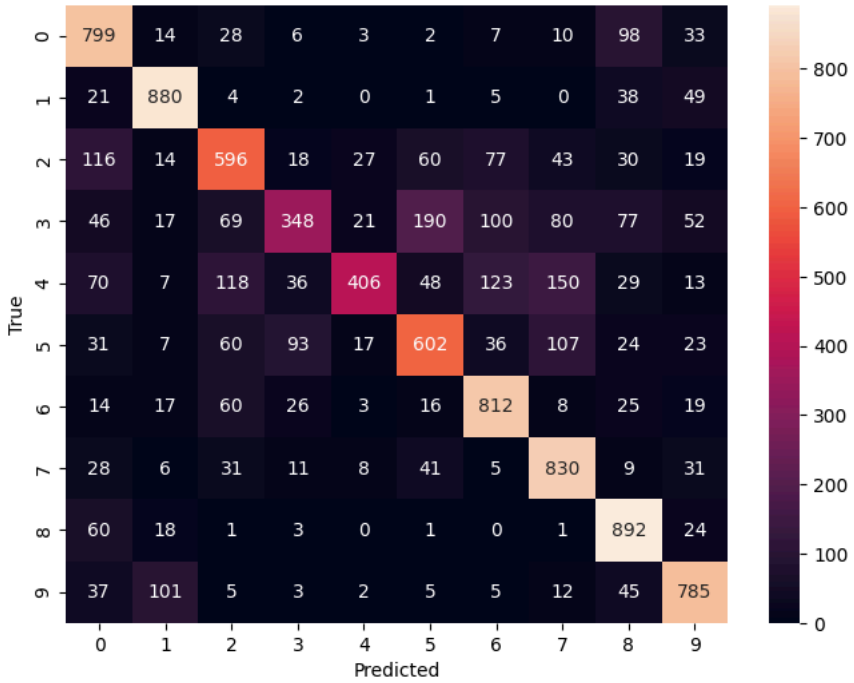
```
from sklearn.metrics import confusion_matrix,classification_report
import numpy as np

y_pred_c10=model.predict(X_test_c10).argmax(axis=1)
cm=confusion_matrix(y_test_c10,y_pred_c10)
print(classification_report(y_test_c10,y_pred_c10))

import seaborn as sns
plt.figure(figsize=(8,6))
sns.heatmap(cm,annot=True,fmt='d')
```

```
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 7s 22ms/step
              precision    recall  f1-score   support

           0       0.65      0.80      0.72      1000
           1       0.81      0.88      0.85      1000
           2       0.61      0.60      0.60      1000
           3       0.64      0.35      0.45      1000
           4       0.83      0.41      0.55      1000
           5       0.62      0.60      0.61      1000
           6       0.69      0.81      0.75      1000
           7       0.67      0.83      0.74      1000
           8       0.70      0.89      0.79      1000
           9       0.75      0.79      0.77      1000

    accuracy                           0.69     10000
   macro avg       0.70      0.70      0.68     10000
weighted avg       0.70      0.69      0.68     10000
```

```
matplotlib.pyplot.show
def show(*args, **kwargs) -> None
```

```
Display all open figures.

Parameters
----------
block : bool, optional
    Whether to wait for all figures to be closed before returning.
```



DAY **04**

What are pre-trained model? MobileNetV2 , ResNet ,VGG ,ImageNet

What is Transfer Learning?

How to Fine-Tune and Optimize?

How to Deploy a Model?

```
!kaggle datasets download -d tongpython/cat-and-dog
```

```
Dataset URL: https://www.kaggle.com/datasets/tongpython/cat-and-dog
License(s): CC0-1.0
Downloading cat-and-dog.zip to /content
 71% 154M/218M [00:00<00:00, 700MB/s]
100% 218M/218M [00:00<00:00, 351MB/s]
```

```
import zipfile
with zipfile.ZipFile('/content/cat-and-dog.zip','r')as zip_ref:
  zip_ref.extractall('/content/dogs-vs-cats')
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
train_generator = datagen.flow_from_directory(
    '/content/dogs-vs-cats',
    target_size=(224,224),
    batch_size=32,
    class_mode='binary',
    subset='training'
)
val_generator = datagen.flow_from_directory(
    '/content/dogs-vs-cats',
    target_size=(224,224),
    batch_size=32,
    class_mode='binary',
    subset='validation'
)
```

```
Found 8023 images belonging to 2 classes.
Found 2005 images belonging to 2 classes.
```

```
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense

base_model=MobileNetV2(weights='imagenet',include_top=False,input_shape=(224,224,3))
base_model.trainable=False

model=Sequential([base_model,GlobalAveragePooling2D(),Dense(128,activation='relu'),Dense(1,activation='sigmoid')])
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
model.fit(train_generator,epochs=5,validation_data=val_generator)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` cl
  self._warn_if_super_not_called()
Epoch 1/5
251/251 ━━━━━━━━━━━━━━━━━━━━ 432s 2s/step - accuracy: 0.7787 - loss: 0.5487 - val_accuracy: 0.7985 - val_loss: 0.5114
Epoch 2/5
251/251 ━━━━━━━━━━━━━━━━━━━━ 444s 2s/step - accuracy: 0.8049 - loss: 0.4871 - val_accuracy: 0.7985 - val_loss: 0.5220
Epoch 3/5
251/251 ━━━━━━━━━━━━━━━━━━━━ 493s 2s/step - accuracy: 0.7882 - loss: 0.4964 - val_accuracy: 0.7985 - val_loss: 0.5193
Epoch 4/5
251/251 ━━━━━━━━━━━━━━━━━━━━ 442s 2s/step - accuracy: 0.8023 - loss: 0.4669 - val_accuracy: 0.7985 - val_loss: 0.5210
Epoch 5/5
251/251 ━━━━━━━━━━━━━━━━━━━━ 436s 2s/step - accuracy: 0.8009 - loss: 0.4551 - val_accuracy: 0.7970 - val_loss: 0.5407
<keras.src.callbacks.history.History at 0x794702d82110>
```

```
base_model.trainable = True
model.compile(optimizer=tf.keras.optimizers.Adam(1e-5), loss='binary_crossentropy', metrics=['accuracy'])
model.fit(train_generator, epochs=3, validation_data=val_generator)
```

```
Epoch 1/3
251/251 ━━━━━━━━━━━━━━━━━━━━ 1849s 7s/step - accuracy: 0.8145 - loss: 0.3935 - val_accuracy: 0.7885 - val_loss: 0.5456
Epoch 2/3
251/251 ━━━━━━━━━━━━━━━━━━━━ 1813s 7s/step - accuracy: 0.8281 - loss: 0.3597 - val_accuracy: 0.7845 - val_loss: 0.5510
Epoch 3/3
251/251 ━━━━━━━━━━━━━━━━━━━━ 1820s 7s/step - accuracy: 0.8414 - loss: 0.3305 - val_accuracy: 0.7731 - val_loss: 0.5611
<keras.src.callbacks.history.History at 0x79468f8c3d90>
```

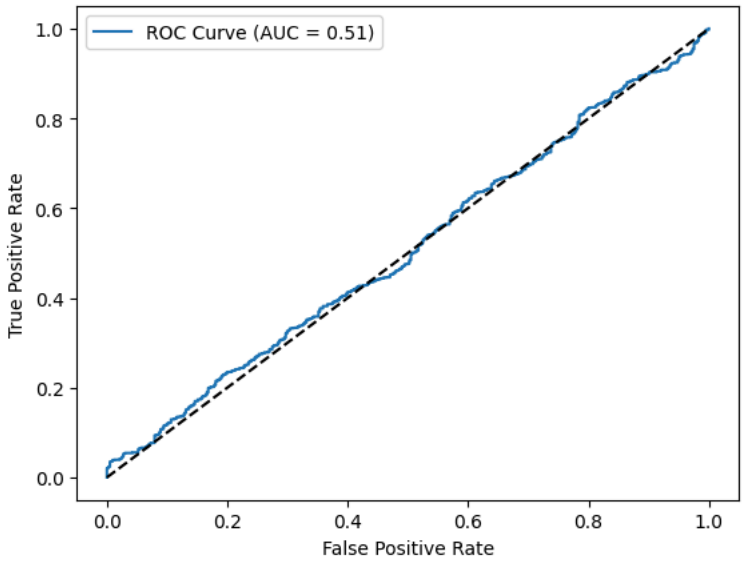
```
model.save('mobilenet_cats_dogs.h5')
#To load
import tensorflow as tf
loaded_model=tf.keras.models.load_model('mobilenet_cats_dogs.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is c
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until y
```

```
from sklearn.metrics import roc_curve, auc

y_pred_proba = loaded_model.predict(val_generator)
fpr, tpr, _ = roc_curve(val_generator.classes, y_pred_proba)
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
```

```
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 84s 1s/step
```



Day 05

How to Predict on New Data?

How to Create Portfolio Assets?

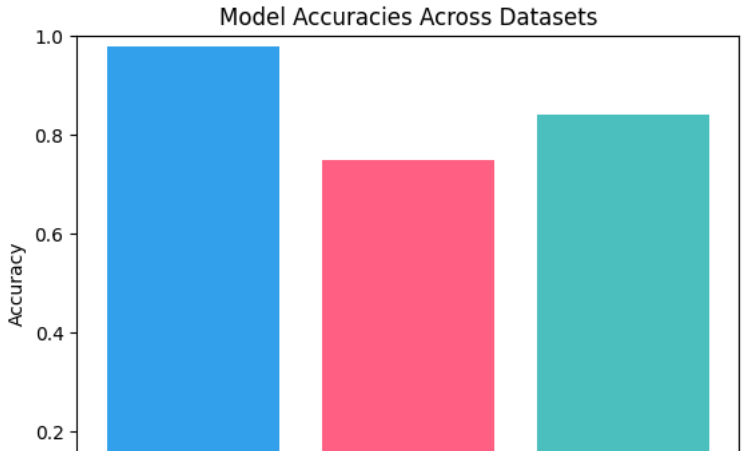
How to Present Results?

```
from google.colab import files
uploaded = files.upload()  # Upload e.g. 'mycat.jpg'

from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
import numpy as np

img_path = 'download.jpg'  # Replace with your filename
img = image.load_img(img_path, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)

prediction = loaded_model.predict(img_array)
print("Predicted class:", "Dog" if prediction[0][0] > 0.5 else "Cat")
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving download1.jpg to download1.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 255ms/step
```

```
import matplotlib.pyplot as plt
datasets = ['MNIST', 'CIFAR-10', 'Cats vs. Dogs']
accuracies = [0.98, 0.75, 0.84]
plt.bar(datasets, accuracies, color=['#36A2EB', '#FF6384', '#4BC0C0'])
plt.title('Model Accuracies Across Datasets')
plt.xlabel('Dataset')
plt.ylabel('Accuracy')
plt.ylim(0,1)
plt.show()
```


Model Accuracies Across Datasets
Accuracy
1.0
0.8
0.6
0.4
0.2