Shape Recognition AI - Cell-by-Cell Documentation

Cell 1 (Code):*

Purpose: Install kaggle API for dataset

Code Preview:
```
#Install kaggle API for dataset
!pip install -q kaggle
```

Cell 2 (Code):*

Purpose: Upload kaggle API credentials

Code Preview:
```
#Upload kaggle API credentials
from google.colab import files
files.upload()
```

Cell 3 (Code):*

Purpose: Configure kaggle environment

Code Preview:
```
#Configure kaggle environment
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

Cell 4 (Code):*

Purpose: Download and extract dataset Set dataset directory

Code Preview:
```
# Download and extract dataset
!kaggle datasets download -d
singhnavjot2062001/geometric-shapes-circle-square-triangle
!unzip geometric-shapes-circle-square-triangle.zip -d shapes_dataset

# Set dataset directory
```

Cell 5 (Code):*

Purpose: Detect classes from folder names in raw_path Create directories Copy files into splits ========== MAIN EXECUTION (Colab local paths, no Drive) ========== Change this if your KaggleHub download path is different If raw dataset doesn't exist, create a sample one for testing Analyze original dataset Organize into train/test/val Verify the split

Code Preview:
```
import os
import shutil
import random
from PIL import Image, ImageDraw
```

Cell 6 (Code):*

Purpose: Paths Ensure validation split exists Run split creation if needed Class names (consistent with earlier fix) Dataset settings Load datasets

Code Preview:
```
import os
import random
import shutil
import tensorflow as tf
```

Cell 7 (Code):*

Purpose: This cell contains code execution without explicit comments.

Code Preview:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
for i in range(9):
```

Cell 8 (Code):*

Purpose: This cell contains code execution without explicit comments.

Code Preview:
```
from tensorflow.keras import layers

data_augmentation = tf.keras.Sequential([
layers.RandomFlip("horizontal"),
layers.RandomRotation(0.1),
```

Cell 9 (Code):*

Purpose: This cell contains code execution without explicit comments.

Code Preview:
```
from tensorflow.keras import models

custom_cnn = models.Sequential([
layers.Rescaling(1./255, input_shape=(128, 128, 3)),
layers.Conv2D(32, (3, 3), activation='relu'),
```

Cell 10 (Code):*

Purpose: This cell contains code execution without explicit comments.

Code Preview:
```
epochs = 10
history_cnn = custom_cnn.fit(
train_ds,
validation_data=val_ds,
epochs=epochs,
```

Cell 11 (Code):*

Purpose: Confusion Matrix

Code Preview:
```
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns

test_loss, test_acc = custom_cnn.evaluate(test_ds)
print(f"\nTest Accuracy: {test_acc:.4f}")
```

Cell 12 (Code):*

Purpose: This cell contains code execution without explicit comments.

Code Preview:
```
from tensorflow.keras.applications import MobileNetV2

def preprocess_mobilenet(image, label):
image = tf.image.resize(image, (224, 224))
return image, label
```

Cell 13 (Code):*

Purpose: Evaluate MobileNetV2 Confusion matrix Classification report Training history

Code Preview:
```
# Evaluate MobileNetV2
test_loss_mobilenet, test_acc_mobilenet =
mobilenet_model.evaluate(mobilenet_test)
print(f"\nMobileNetV2 Test Accuracy: {test_acc_mobilenet:.4f}")

# Confusion matrix
```

Cell 14 (Code):*

Purpose: Performance Comparison First evaluate CNN model if not already done ROC Curve Function Generate ROC Curves Model Comparison Add value labels on top of bars

Code Preview:
```
# Performance Comparison
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
import numpy as np
```

Cell 15 (Code):*

Purpose: This cell contains code execution without explicit comments.

Code Preview:
```
def plot_predictions(model, dataset, model_name, size=(224, 224)):
plt.figure(figsize=(15, 12))
for images, labels in dataset.take(1):
if images.shape[1:3] != size:
display_images = tf.image.resize(images, size)
```

Cell 16 (Code):*

Purpose: 1. Generate requirements.txt 2. Find the notebook file 3. Convert notebook to python script 4. List all possible output files Core files Model files Visualization files 5. Create zip with only existing files 6. Verify and download Execute the packaging

Code Preview:

```
import os
import zipfile
from google.colab import files

def package_results():
```