A logo with text overlay

AI-generated content may be incorrect.

Dr:Radwa Fatahalla

TA:Mahmoud ElMorshedy

Nardin Eslam 211000791

Code:

import tensorflow as tf

import tensorflow\_datasets as tfds

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from tensorflow.keras import models, layers, optimizers

# ✅ Fix for NotImplementedError (eager mode)

tf.config.run\_functions\_eagerly(True)

# --- Setup and Data Loading ---

IMG\_SIZE = (150, 150)

BATCH\_SIZE = 32

(ds\_train, ds\_val), ds\_info = tfds.load(

'rock\_paper\_scissors',

split=['train[:90%]', 'train[90%:]'],

shuffle\_files=True,

as\_supervised=True,

with\_info=True

)

def preprocess(image, label):

image = tf.image.resize(image, IMG\_SIZE) / 255.0

return image, label

train\_data = ds\_train.map(preprocess).shuffle(1000).batch(BATCH\_SIZE).prefetch(tf.data.AUTOTUNE)

val\_data = ds\_val.map(preprocess).batch(BATCH\_SIZE).prefetch(tf.data.AUTOTUNE)

# --- Model Builders (with Sigmoid activations) ---

def build\_mlp\_model(input\_shape):

return models.Sequential([

tf.keras.Input(shape=input\_shape),

layers.Flatten(),

layers.Dense(128, activation='sigmoid'),

layers.Dense(64, activation='sigmoid'),

layers.Dense(3, activation='softmax')

])

def build\_cnn\_model(input\_shape):

return models.Sequential([

tf.keras.Input(shape=input\_shape),

layers.Conv2D(32, (3, 3), activation='sigmoid'),

layers.MaxPooling2D(2, 2),

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

layers.MaxPooling2D(2, 2),

layers.Flatten(),

layers.Dense(64, activation='sigmoid'),

layers.Dense(3, activation='softmax')

])

# --- Training and Evaluation Loop ---

optimizer\_names = ['SGD', 'SGD\_Momentum', 'Adagrad', 'RMSProp', 'Adam']

results = {

'Model': [],

'Optimizer': [],

'Train Loss': [],

'Train Acc': [],

'Validation Loss': [],

'Validation Acc': []

}

history\_dict = {}

input\_shape = IMG\_SIZE + (3,)

for model\_type, model\_builder in [('MLP', build\_mlp\_model), ('CNN', build\_cnn\_model)]:

for opt\_name in optimizer\_names:

# ✅ Create a fresh optimizer instance every time

if opt\_name == 'SGD':

opt = optimizers.SGD()

elif opt\_name == 'SGD\_Momentum':

opt = optimizers.SGD(momentum=0.9)

elif opt\_name == 'Adagrad':

opt = optimizers.Adagrad()

elif opt\_name == 'RMSProp':

opt = optimizers.RMSprop()

elif opt\_name == 'Adam':

opt = optimizers.Adam()

print(f"\n Training {model\_type} with {opt\_name}...")

# Build and compile model

model = model\_builder(input\_shape=input\_shape)

model.compile(optimizer=opt,

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

# Train the model

history = model.fit(

train\_data,

validation\_data=val\_data,

epochs=10,

verbose=0

)

# Evaluate performance

train\_loss, train\_acc = model.evaluate(train\_data, verbose=0)

val\_loss, val\_acc = model.evaluate(val\_data, verbose=0)

# Store results

results['Model'].append(model\_type)

results['Optimizer'].append(opt\_name)

results['Train Loss'].append(train\_loss)

results['Train Acc'].append(train\_acc)

results['Validation Loss'].append(val\_loss)

results['Validation Acc'].append(val\_acc)

history\_dict[f"{model\_type}\_{opt\_name}"] = history.history

# --- Plotting ---

def plot\_metric(metric, title):

plt.figure(figsize=(10, 5))

for key, history in history\_dict.items():

if metric in history:

plt.plot(history[metric], label=key)

plt.title(title)

plt.xlabel('Epochs')

plt.ylabel(metric)

plt.legend()

plt.grid(True)

plt.show()

plot\_metric('loss', "Train Loss for all Models")

plot\_metric('val\_loss', "Validation Loss for all Models")

plot\_metric('accuracy', "Train Accuracy for all Models")

plot\_metric('val\_accuracy', "Validation Accuracy for all Models")

# --- Final Results Table ---

results\_df = pd.DataFrame(results)

print("\n Final Results Table:\n")

print(results\_df)

