1.Activation Functions & Optimization Algorithms

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.optimizers import SGD, Adam, RMSprop

# Load and preprocess the Iris dataset

iris = load\_iris()

X = iris.data

y = iris.target.reshape(-1, 1)

# One-hot encode the target labels

encoder = OneHotEncoder(sparse\_output=False)

y = encoder.fit\_transform(y)

# Split into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Function to build and train the model

def build\_and\_train(activation, optimizer):

    model = Sequential([

 Dense(10, input\_shape=(4,), activation=activation),

        Dense(3, activation='softmax')  # Output layer

    ])

    model.compile(optimizer=optimizer, loss='categorical\_crossentropy', metrics=['accuracy'])

    print(f"\nTraining with activation='{activation}' and optimizer='{optimizer.\_\_class\_\_.\_\_name\_\_}'")

    model.fit(X\_train, y\_train, epochs=20, verbose=0)

    loss, acc = model.evaluate(X\_test, y\_test, verbose=0)

    print(f"Test Accuracy: {acc:.4f}")

# Run the experiment

activations = ['tanh']

optimizers = [SGD(learning\_rate=0.01), Adam(learning\_rate=0.01), RMSprop(learning\_rate=0.01)]

for act in activations:

    for opt in optimizers:

        build\_and\_train(act, opt)

2. Weight Initialization Techniques

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Input

from tensorflow.keras.initializers import HeNormal, GlorotUniform, RandomNormal

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, OneHotEncoder

# Load & preprocess the Iris dataset

X, y = load\_iris(return\_X\_y=True)

y = OneHotEncoder(sparse\_output=False).fit\_transform(y.reshape(-1, 1))

X = StandardScaler().fit\_transform(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Function to build & train model with given initializer

def train\_with\_initializer(init):

    model = Sequential([

        Input(shape=(4,)),  # Use Input layer

        Dense(10, activation='relu', kernel\_initializer=init),

        Dense(3, activation='softmax')

    ])

    model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

    print(f"\nUsing initializer: {init.\_\_class\_\_.\_\_name\_\_}")

    model.fit(X\_train, y\_train, epochs=20, verbose=0)

    \_, acc = model.evaluate(X\_test, y\_test, verbose=0)

    print(f"Test Accuracy: {acc:.4f}")

# Try different initializers

for initializer in [HeNormal(), GlorotUniform(), RandomNormal(mean=0.0, stddev=0.05)]:

    train\_with\_initializer(initializer)

3