Student Information

Name: Tejaswi Narsinga

ID: 700756916

CRN : 23850

Question 1: Tensor Manipulations & Reshaping

```python

import tensorflow as tf

import numpy as np

tf.random.set\_seed(42)

# Create random tensor

random\_tensor = tf.random.normal([4, 6])

print(random\_tensor)

# Find rank and shape

print(tf.rank(random\_tensor).numpy())

print(random\_tensor.shape)

# Reshape and transpose

reshaped\_tensor = tf.reshape(random\_tensor, [2, 3, 4])

print(reshaped\_tensor)

transposed\_tensor = tf.transpose(reshaped\_tensor, perm=[1, 0, 2])

print(transposed\_tensor)

# Broadcasting

small\_tensor = tf.constant([[1.0, 2.0, 3.0, 4.0]])

print(small\_tensor)

broadcasted\_result = small\_tensor + random\_tensor[:, :4]

print(broadcasted\_result)

```

# Expected Output

- Random tensor with shape (4, 6)

- Rank and shape of the tensor

- Reshaped tensor with shape (2, 3, 4)

- Transposed tensor with shape (3, 2, 4)

- Broadcasted result with shape (4, 4)

# Explanation

- Tensor manipulation and reshaping using TensorFlow

- Broadcasting and tensor operations

Question 2: Loss Functions & Hyperparameter Tuning

```python

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from sklearn.datasets import load\_iris

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

# Load iris dataset

iris = load\_iris()

X = iris.data

y = iris.target

# Split dataset into training and testing sets

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

# Define model architecture

model = Sequential([

Dense(64, activation='relu', input\_shape=(4,)),

Dense(32, activation='relu'),

Dense(3, activation='softmax')

])

# Compile model with different loss functions

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

# Train model

model.fit(X\_train, y\_train, epochs=10, batch\_size=32, validation\_data=(X\_test, y\_test))

```

# Expected Output

- Model architecture and compilation

- Training and validation accuracy and loss

# Explanation

- Loss functions and hyperparameter tuning using TensorFlow

- Model architecture and compilation

- Training and validation

Question 3: Training Models with Different Optimizers

```python

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.datasets import mnist

# Load MNIST dataset

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

# Normalize pixel values

X\_train = X\_train.astype('float32') / 255.0

X\_test = X\_test.astype('float32') / 255.0

# Define model architecture

model = Sequential([

Dense(64, activation='relu', input\_shape=(784,)),

Dense(32, activation='relu'),

Dense(10, activation='softmax')

])

# Compile model with different optimizers

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

# Train model

model.fit(X\_train, y\_train, epochs=10, batch\_size=32, validation\_data=(X\_test, y\_test))

```

# Expected Output

- Model architecture and compilation

- Training and validation accuracy and loss

# Explanation

- Training models with different optimizers using TensorFlow

- Model architecture and compilation

- Training and validation

Question 4: TensorBoard Logging

```python

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.callbacks import TensorBoard

# Define model architecture

model = Sequential([

Dense(64, activation='relu', input\_shape=(784,)),

Dense(32, activation='relu'),

Dense(10, activation='softmax')

])

# Compile model

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

# Define TensorBoard callback

tensorboard\_callback = TensorBoard(log\_dir='./logs', histogram\_freq=1)

# Train model with TensorBoard logging

model.fit(X\_train, y\_train, epochs=10, batch\_size=32, validation\_data=(X\_test, y\_test), callbacks=[tensorboard\_callback])

```

# Expected Output

- Model architecture and compilation

- TensorBoard logging setup

- Training and validation accuracy and loss

# Explanation

- TensorBoard logging using TensorFlow

- Model architecture and compilation

- Training and validation with TensorBoard logging