```
# 1 program of Basic operations on TensorFlow.
import tensorflow as tf

a = tf.constant([2, 4, 6])
b = tf.constant([1, 3, 5])

add = tf.add(a, b)
sub = tf.subtract(a, b)
mul = tf.multiply(a, b)
div = tf.divide(a, b)

print("Tensor a:", a.numpy())
print("Tensor b:", b.numpy())
print("Addition (a + b):", add.numpy())
print("Subtraction (a - b):", sub.numpy())
print("Multiplication (a * b):", mul.numpy())
print("Division (a / b):", div.numpy())

matrix1 = tf.constant([[1, 2], [3, 4]])
matrix2 = tf.constant([[5, 6], [7, 8]])
matmul = tf.matmul(matrix1, matrix2)

print("\nMatrix1:\n", matrix1.numpy())
print("Matrix2:\n", matrix2.numpy())
print("Matrix Multiplication:\n", matmul.numpy())



################################################################################
######################

# 2 Design a neural network forclassifying movie reviews(BinaryClassification) using IMDB
dataset.
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences

max_features = 10000
max_len = 500

(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)

x_train = pad_sequences(x_train, maxlen=max_len)
x_test = pad_sequences(x_test, maxlen=max_len)

model = models.Sequential()
```

```
model.add(layers.Embedding(input_dim=max_features, output_dim=128,
input_length=max_len))
model.add(layers.LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=64, epochs=5, validation_data=(x_test, y_test))

score, accuracy = model.evaluate(x_test, y_test, batch_size=64)
print(f'Test loss: {score}')
print(f'Test accuracy: {accuracy}')

################################################################################
####################

# 3 Design a neural network for predicting house prices using Boston Housing Price
dataset.

import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import boston_housing
from sklearn.preprocessing import StandardScaler

(x_train, y_train), (x_test, y_test) = boston_housing.load_data()

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_dim=x_train.shape[1]))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mae'])

model.fit(x_train, y_train, epochs=30, batch_size=32, validation_data=(x_test, y_test))

loss, mae = model.evaluate(x_test, y_test)
print(f'Test Loss (MSE): {loss}')
print(f'Test MAE (Mean Absolute Error): {mae}')

################################################################################
####################

# 4 Implement word embeddings for IMDB dataset.
```

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences

max_features = 5000
max_len = 500

(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)

x_train = pad_sequences(x_train, maxlen=max_len)
x_test = pad_sequences(x_test, maxlen=max_len)

model = models.Sequential()
model.add(layers.Embedding(input_dim=max_features, output_dim=128,
input_length=max_len))
model.add(layers.LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=64, epochs=3, validation_data=(x_test, y_test))

score, accuracy = model.evaluate(x_test, y_test, batch_size=64)
print(f'Test loss: {score}')
print(f'Test accuracy: {accuracy}')

################################################################################
####################

# 5 Implement a Recurrent Neural Network for IMDB movie review classification problem

import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences

max_features = 5000
max_len = 500

(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)

x_train = pad_sequences(x_train, maxlen=max_len)
x_test = pad_sequences(x_test, maxlen=max_len)

model = models.Sequential()
```

```
model.add(layers.Embedding(input_dim=max_features, output_dim=128,
input_length=max_len))
model.add(layers.LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(layers.Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=64, epochs=3, validation_data=(x_test, y_test))

score, accuracy = model.evaluate(x_test, y_test, batch_size=64)
print(f'Test loss: {score}')
print(f'Test accuracy: {accuracy}')
```

1. Basic Operations on TensorFlow

- **TensorFlow** is a powerful framework for building and deploying machine learning models.
- **Basic operations** include arithmetic (addition, subtraction, multiplication, division) on tensors.
- **Matrix operations** like matrix multiplication are essential for linear algebra and are widely used in neural network computations.
- **Broadcasting** allows operations between tensors of different shapes by automatically expanding dimensions.
- TensorFlow can handle both **sparse tensors** (which save memory by only storing non-zero values) and **dense tensors**.
- **Eager execution** allows immediate execution of operations and easy debugging, while **graph execution** builds a computational graph for optimized performance.
- **TensorFlow** supports running models on both **CPUs and GPUs**, accelerating computational tasks for large datasets.
- TensorFlow is highly efficient in handling **automatic differentiation** for backpropagation during training.

---

2. Neural Network for Classifying Movie Reviews (Binary Classification) using IMDB Dataset

- The **IMDB dataset** is commonly used for sentiment analysis, where each review is labeled as positive or negative.
- **Embedding layers** convert words into dense vectors, which help models understand the meaning of words by capturing semantic relationships.
- **LSTM (Long Short-Term Memory)** layers are used to handle long-term dependencies in text sequences.
- **Binary classification** involves using a **sigmoid activation function** in the output layer to classify reviews into two categories.
- **Text preprocessing** like padding and tokenization is critical for ensuring that all reviews are of the same length.
- **Dropout layers** are added to prevent overfitting by randomly deactivating neurons during training.

- The model uses **binary crossentropy loss**, which is specifically designed for binary classification problems.
- **Model evaluation** is done using accuracy, which measures the proportion of correctly classified reviews.

---

3. Neural Network for Predicting House Prices Using Boston Housing Dataset

- The **Boston Housing dataset** consists of features like the number of rooms, property tax rates, and proximity to schools, and is used for regression tasks.
- **Normalization** of data ensures that all features have the same scale, improving the convergence speed of the model.
- This problem is a **regression problem**, meaning the output is a continuous value (house price).
- **Mean Squared Error (MSE)** is the standard loss function used for regression tasks, as it minimizes the squared differences between predicted and actual values.
- **Dense layers** are used in the model, where each node in one layer connects to every node in the next layer.
- **Activation functions** like ReLU are used to introduce non-linearity into the model, allowing it to learn complex patterns.
- The model can be evaluated using **Mean Absolute Error (MAE)**, which provides a clearer understanding of the average prediction error in terms of actual values.
- The **train-test split** ensures that the model is tested on unseen data to evaluate generalization.

---

4. Word Embeddings for IMDB Dataset

- **Word embeddings** represent words as dense vectors of real numbers, which capture the semantic meaning of the words.
- **Embedding layers** in neural networks learn to map words to vectors that capture relationships between similar words.
- **Pre-trained embeddings** like Word2Vec, GloVe, or FastText can be used to initialize embeddings, providing a strong semantic foundation.
- **Padding sequences** to a fixed length ensures that the input data has a consistent shape and size.

- **Word embeddings** help reduce the dimensionality of input data, making it easier to train models on text.
- **Transfer learning** with pre-trained word embeddings can improve model performance when training on smaller datasets.
- The **embedding layer** typically learns better representations for domain-specific terms in specialized text data.
- **Fine-tuning** embeddings during training can improve the quality of the word vectors for the specific task at hand.

---

5. Recurrent Neural Network for IMDB Movie Review Classification

- **RNNs (Recurrent Neural Networks)** are ideal for tasks involving sequential data, like text or time series.
- **LSTM (Long Short-Term Memory)** cells help preserve long-term dependencies, making RNNs effective for language tasks like sentiment analysis.
- **Dropout layers** are used during training to prevent overfitting by randomly dropping some neurons.
- **Padding sequences** ensures that all input sequences have the same length for feeding into the model.
- The **sigmoid activation function** is used in the output layer for binary classification tasks, outputting values between 0 and 1.
- **Gradient clipping** can be used to prevent the vanishing or exploding gradient problem during backpropagation in RNNs.
- **Backpropagation through time (BPTT)** is used to train RNNs, updating weights based on the error at each time step.
- The model's **evaluation** typically uses metrics like accuracy and loss to assess performance, with validation on the test set.