# 1 Introduction

TensorFlow is an open-source library for numerical computation and machine learning, especially deep learning. It allows for the creation and training of neural networks with flexible architecture.

## 1.1 Installation

```
pip install tensorflow
```

## 1.2 Importing TensorFlow

```
import tensorflow as tf
```

# 2 Tensors

## 2.1 Creating Tensors

```
# Scalar
scalar = tf.constant(7)
# Vector
vector = tf.constant([1, 2, 3])
# Matrix
matrix = tf.constant([[1, 2], [3, 4]])
# Random Tensor
tensor = tf.random.normal((3, 3))
```

## 2.2 Tensor Attributes

```
tensor.shape    # Shape of tensor
tensor.dtype    # Data type
```

# 3 Tensor Operations

## 3.1 Mathematical Operations

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

add = a + b
sub = a - b
mult = a * b
div = a / b

# Matrix Multiplication
mat_mult = tf.matmul(a[tf.newaxis, :], b[:, tf.
    newaxis])
```

## 3.2 Tensor Manipulation

```
# Reshaping
reshaped = tf.reshape(a, (3, 1))
# Slicing
slice_a = a[0:2]
# Concatenation
concatenated = tf.concat([a, b], axis=0)
```

# 4 Building Models

## 4.1 Sequential Model

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

model = Sequential([
    Dense(64, activation='relu', input_shape=(32,)
    ),
    Dense(10, activation='softmax')
])
```

## 4.2 Compiling the Model

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

# 5 Training Models

## 5.1 Data Preparation

```
# Assuming X_train and y_train are prepared
```

## 5.2 Model Fitting

```
model.fit(X_train, y_train, epochs=10,
    validation_split=0.2)
```

# 6 Evaluating Models

## 6.1 Model Evaluation

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Loss: {loss}, Accuracy: {accuracy}')
```

## 6.2 Making Predictions

```
predictions = model.predict(X_new)
```

# 7 Advanced Topics

## 7.1 Custom Layers

```
from tensorflow.keras.layers import Layer

class CustomLayer(Layer):
    def __init__(self, units=32):
        super(CustomLayer, self).__init__()
        self.units = units

    def build(self, input_shape):
        self.w = self.add_weight(shape=(
    input_shape[-1], self.units),
                                 initializer='
    random_normal')
        self.b = self.add_weight(shape=(self.units
    ,),
                                 initializer='
    zeros')

    def call(self, inputs):
        return tf.matmul(inputs, self.w) + self.b
```

## 7.2 Callbacks

```
from tensorflow.keras.callbacks import
    EarlyStopping

early_stopping = EarlyStopping(monitor='val_loss',
     patience=3)

model.fit(X_train, y_train, epochs=10,
          validation_split=0.2, callbacks=[
    early_stopping])
```