Name-Bachu Ganesh

Roll Number-CH.EN.U4AIE20003

In [1]:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

In [2]:

```
# Load the MNIST Fashion dataset
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

# Reshape the data to have a single channel
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1)
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1)

# Convert the data to floats and normalize it to the range 0-1
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255
```

In [3]:

```
# Define the image augmentation parameters
datagen = ImageDataGenerator(
    rotation_range=10,  # rotate images randomly by up to 10 degrees
    zoom_range=0.1,  # zoom in and out of images randomly by up to 10%
    width_shift_range=0.1,  # shift images horizontally by up to 10%
    height_shift_range=0.1,  # shift images vertically by up to 10%
)
```

In [4]:

```
# Define the model architecture
model = keras.Sequential([
    keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    keras.layers.MaxPooling2D((2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])
```

In [5]:

```
# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accurac
y'])
```

In [6]:

```
# Train the model without data augmentation
history1 = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test), verb
ose=0)
```

In [7]:

```
# Train the model using the augmented data
datagen.fit(x_train)
history2 = model.fit(datagen.flow(x_train, y_train, batch_size=32),
          steps_per_epoch=len(x_train) / 32, epochs=10, validation_data=(x_test, y_test)
```

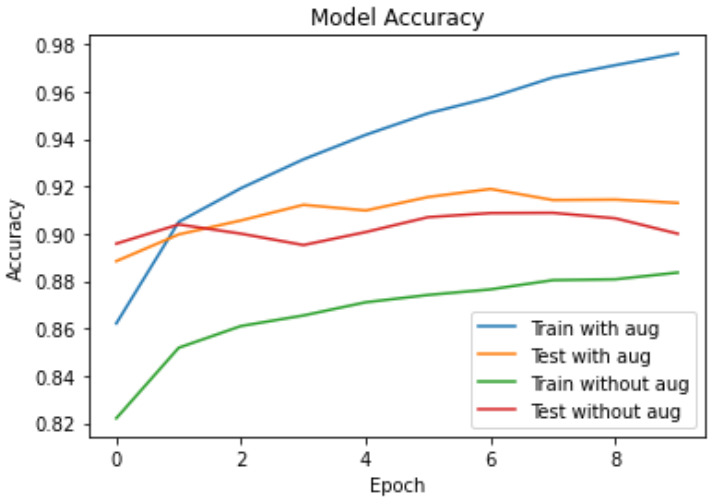
```
, verbose=0)
```

In [9]:

```
# Evaluate the model on the test data
test_loss1, test_acc1 = model.evaluate(x_test, y_test, verbose=2)
test_loss2, test_acc2 = model.evaluate(x_test, y_test, verbose=2)
print('Test accuracy without data augmentation:', test_acc1)
print('Test accuracy with data augmentation:', test_acc2)

# Plot the training and validation accuracy for both cases
plt.plot(history1.history['accuracy'])
plt.plot(history1.history['val_accuracy'])
plt.plot(history2.history['accuracy'])
plt.plot(history2.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train with aug', 'Test with aug', 'Train without aug', 'Test without aug'],
loc='lower right')
plt.show()
```

```
313/313 - 1s - loss: 0.2930 - accuracy: 0.9000 - 1s/epoch - 5ms/step
313/313 - 2s - loss: 0.2930 - accuracy: 0.9000 - 2s/epoch - 5ms/step
Test accuracy without data augmentation: 0.8999999761581421
Test accuracy with data augmentation: 0.8999999761581421
```



In [ ]:

In [ ]: