Train fashion_mnist dataset with LSTM model.

```
import matplotlib.pyplot as plt
from tensorflow.keras import datasets, layers, models

(x_train, y_train), (x_test, y_test) = datasets.fashion_mnist.load_data()
x_train, x_test = x_train / 255.0, x_test /255.0

model = models.Sequential([
    layers.LSTM(64, input_shape=(28,28), activation='relu', return_sequences = True),
    layers.LSTM(64, activation = 'relu'),
    layers.Dense(32, activation = 'relu'),
    layers.Dense(10, activation='softmax')
])

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

Result of trained model.

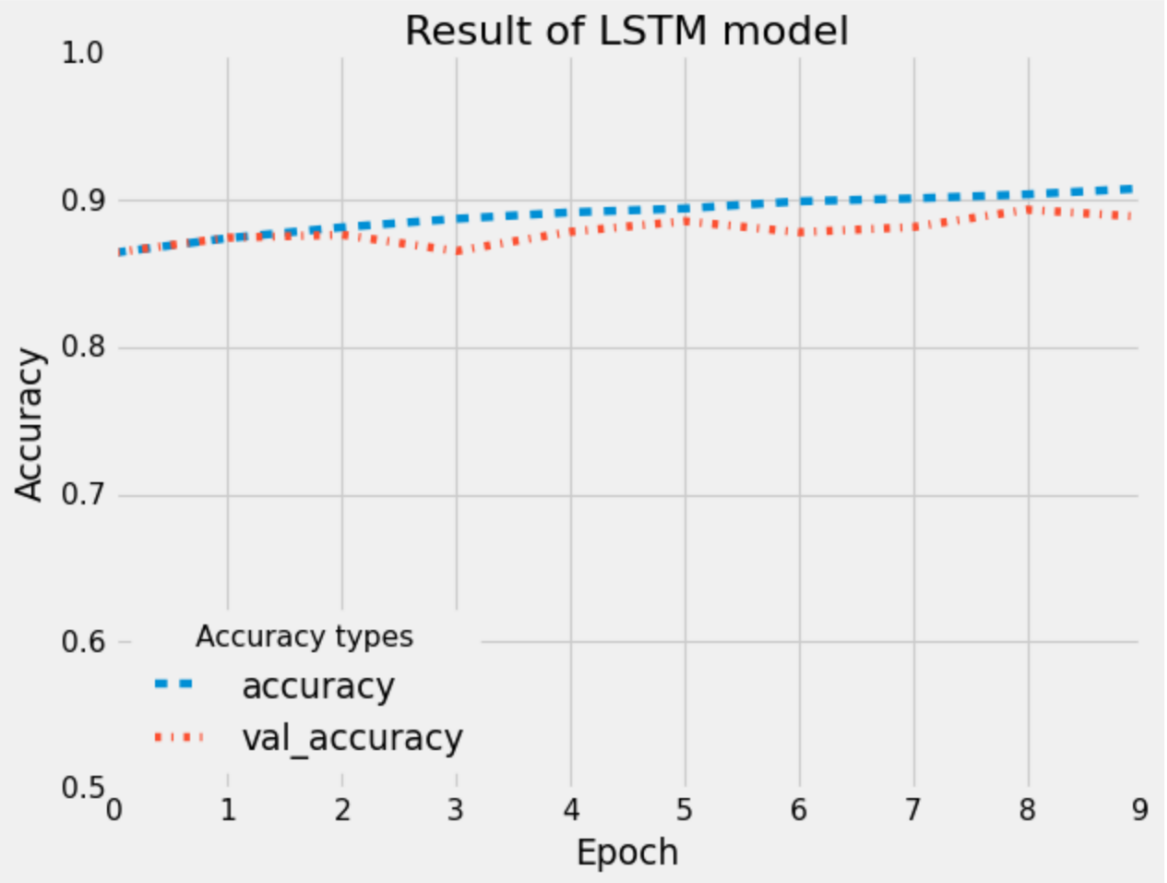
```
history = model.fit(x_train, y_train, epochs = 10, validation_data = (x_test, y_test), batch_size = 64)
test_loss, test_acc = model.evaluate(x_train, y_train, batch_size = 64)

plt.style.use("fivethirtyeight")
plt.plot(history.history['accuracy'], '--', label = 'accuracy')
plt.plot(history.history['val_accuracy'], '-.', label = 'val_accuracy')

plt.title("Result of LSTM model")
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc = 'best', title='Accuracy types')
plt.show()
```

```
Epoch 1/10
938/938 [==============================] - 45s 48ms/step - loss: 0.3609 - accuracy: 0.8642 - val_loss: 0.3682 - val_accuracy: 0.8643
Epoch 2/10
938/938 [==============================] - 45s 48ms/step - loss: 0.3355 - accuracy: 0.8742 - val_loss: 0.3517 - val_accuracy: 0.8745
Epoch 3/10
938/938 [==============================] - 44s 47ms/step - loss: 0.3164 - accuracy: 0.8816 - val_loss: 0.3358 - val_accuracy: 0.8765
Epoch 4/10
938/938 [==============================] - 44s 47ms/step - loss: 0.3021 - accuracy: 0.8873 - val_loss: 0.3590 - val_accuracy: 0.8654
Epoch 5/10
938/938 [==============================] - 44s 47ms/step - loss: 0.2898 - accuracy: 0.8917 - val_loss: 0.3318 - val_accuracy: 0.8785
Epoch 6/10
938/938 [==============================] - 45s 48ms/step - loss: 0.2794 - accuracy: 0.8943 - val_loss: 0.3095 - val_accuracy: 0.8857
Epoch 7/10
938/938 [==============================] - 45s 47ms/step - loss: 0.2690 - accuracy: 0.8992 - val_loss: 0.3328 - val_accuracy: 0.8782
Epoch 8/10
938/938 [==============================] - 44s 47ms/step - loss: 0.2615 - accuracy: 0.9012 - val_loss: 0.3270 - val_accuracy: 0.8817
Epoch 9/10
938/938 [==============================] - 45s 48ms/step - loss: 0.2524 - accuracy: 0.9039 - val_loss: 0.3004 - val_accuracy: 0.8935
Epoch 10/10
938/938 [==============================] - 44s 47ms/step - loss: 0.2446 - accuracy: 0.9080 - val_loss: 0.3032 - val_accuracy: 0.8887
938/938 [==============================] - 16s 17ms/step - loss: 0.2267 - accuracy: 0.9133
```

Visualize the accuracy.



Replace LSTM model with NN model.

```
model2 = models.Sequential([
    layers.Flatten(input_shape = (28,28)),
    layers.Dense(256, activation = 'relu'),
    layers.Dense(64, activation = 'relu'),
    layers.Dense(10, activation='softmax')
])

model2.summary()
model2.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

history2 = model2.fit(x_train, y_train, epochs = 10, validation_data = (x_test, y_test), batch_size = 64)
test_loss2, test_acc2 = model2.evaluate(x_train, y_train, batch_size = 64)
```

Result of trained NN model.

```
Epoch 1/10
938/938 [==============================] - 6s 6ms/step - loss: 0.4967 - accuracy: 0.8253 - val_loss: 0.4265 - val_accuracy: 0.8383
Epoch 2/10
938/938 [==============================] - 6s 6ms/step - loss: 0.3656 - accuracy: 0.8650 - val_loss: 0.3938 - val_accuracy: 0.8534
Epoch 3/10
938/938 [==============================] - 6s 6ms/step - loss: 0.3326 - accuracy: 0.8781 - val_loss: 0.3579 - val_accuracy: 0.8684
Epoch 4/10
938/938 [==============================] - 5s 6ms/step - loss: 0.3041 - accuracy: 0.8872 - val_loss: 0.3636 - val_accuracy: 0.8654
Epoch 5/10
938/938 [==============================] - 6s 6ms/step - loss: 0.2891 - accuracy: 0.8918 - val_loss: 0.3454 - val_accuracy: 0.8734
Epoch 6/10
938/938 [==============================] - 5s 6ms/step - loss: 0.2725 - accuracy: 0.8986 - val_loss: 0.3335 - val_accuracy: 0.8783
Epoch 7/10
938/938 [==============================] - 5s 5ms/step - loss: 0.2602 - accuracy: 0.9023 - val_loss: 0.3330 - val_accuracy: 0.8836
Epoch 8/10
938/938 [==============================] - 5s 6ms/step - loss: 0.2483 - accuracy: 0.9059 - val_loss: 0.3364 - val_accuracy: 0.8820
Epoch 9/10
938/938 [==============================] - 5s 6ms/step - loss: 0.2376 - accuracy: 0.9107 - val_loss: 0.3346 - val_accuracy: 0.8858
Epoch 10/10
938/938 [==============================] - 7s 7ms/step - loss: 0.2289 - accuracy: 0.9133 - val_loss: 0.3349 - val_accuracy: 0.8857
938/938 [==============================] - 3s 3ms/step - loss: 0.2146 - accuracy: 0.9206
```

Visualize the result's accuracy of two models.

```
plt.style.use("ggplot")

plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], 'o--', label = 'accuracy')
plt.plot(history.history['val_accuracy'], 'D-.', label = 'val_accuracy')
plt.title("Result of LSTM model")
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.8, 1])

plt.legend(loc = 'best', title='Accuracy types')

plt.subplot(1,2,2)
plt.plot(history2.history['accuracy'], 'o--', label = 'accuracy')
plt.plot(history2.history['val_accuracy'], 'D-.', label = 'val_accuracy')
plt.title("Result of NN model")
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.8, 1])

plt.suptitle("LSTM model vs NN model")
plt.legend(loc = 'best', title='Accuracy types')
plt.show()
```

The LSTM model helps to get more stable and higher accuracy in this case.