In [2]:

```
### 1. Neural Network Model 4 - Final Exam ###
### Fashion-MNIST as a CNN

# Import the libraries that will be needed.

print('Importing libraries...')
import os
import time
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'  # Set level of TF screen logging suppression {'0', '1', '2'}
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix, ConfusionMatrixDisplay, confusion_matrix
print('TensorFlow version', tf.__version__)
print('Done.')
```

```
Importing libraries...
TensorFlow version 2.11.0
Done.
```

In [3]:

```
# 2. Load training data

print('Loading training data...')
train = pd.read_csv('fashion-mnist_train.csv')
print('Done.')
```

```
Loading training data...
Done.
```

In [4]:

```
# 3. Here is where your class images, the test data, are being loaded
print('Loading test data...')
test = pd.read_csv('classTest.csv.')
print('Done.')
```

```
Loading test data...
Done.
```

In [5]:

```
# 4. Shape data for training and testing

#Store first column as target vector
train_labels = np.array(train.label)
test_labels = np.array(test.label)
```

```
train.drop('label',axis=1, inplace=True)
test.drop('label',axis=1, inplace=True)

#Reshape - adding one more dimension to each image
train_data = np.array(train).reshape(train.shape[0],28,28,1)
test_data = np.array(test).reshape(test.shape[0],28,28,1)

#Print shapes
print('Train tensor shape: ',train_data.shape)
print('Test tensor shape: ',test_data.shape)
print('Single image shape:', train_data[0].shape)
print('Train_labels shape:', train_labels.shape)
print('Test_labels shape:', test_labels.shape)
```

```
Train tensor shape:  (60000, 28, 28, 1)
Test tensor shape:  (138, 28, 28, 1)
Single image shape: (28, 28, 1)
Train_labels shape: (60000,)
Test_labels shape: (138,)
```

In [6]:

```
# 5. Create label names
label_names = ["T-shirt/top","Trouser","Pullover","Dress","Coat","Sandal","Shirt","Sneaker","Bag","Ankle boot"]
```

In [7]:

```
# 6. Show some of the training data images

import random
plt.figure(figsize=(15,7))
for i in range(12):
    ax=plt.subplot(3,4,i+1)
    rand_index=random.choice(range(len(train_data)))
    plt.imshow(train_data[rand_index], cmap=plt.cm.binary)
    plt.axis(False)
    plt.title(label_names[train_labels[rand_index]])
```

In [8]:

```
# 7. Normalize the data

train_norm = train_data / 255.0
test_norm = test_data / 255.0

X_train = train_norm[10000:]
y_train = train_labels[10000:]

X_valid = train_norm[:10000]
y_valid = train_labels[:10000]
```

In [9]:

```
# 8. Build the CNN model

#Set random seed
```

```
import os
import time
import tensorflow as tf


tf.random.set_seed(42)

#Create model
model = tf.keras.Sequential([

    tf.keras.layers.Conv2D(filters=5, kernel_size=2, strides=1, padding="same", activation="relu",
                           input_shape=(28,28,1)),
    tf.keras.layers.Conv2D(10,3, padding="valid", activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=2),
    tf.keras.layers.Conv2D(15,3, padding="valid", activation='relu'),
    tf.keras.layers.Conv2D(20,3, padding="valid", activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=2),
    tf.keras.layers.Conv2D(25,3, padding="valid", activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(10, activation="softmax") #Output layer

])
```

In [10]:

```
# 9. Compile the model
#
# You set the learning rate in this code block.
# Original learning rate = 0.001

model.compile(loss="sparse_categorical_crossentropy",
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              metrics=["accuracy"])

#Show summary
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 5)         25        
                                                                 
 conv2d_1 (Conv2D)           (None, 26, 26, 10)        460       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 13, 13, 10)       0         
 )                                                               
                                                                 
 conv2d_2 (Conv2D)           (None, 11, 11, 15)        1365      
                                                                 
 conv2d_3 (Conv2D)           (None, 9, 9, 20)          2720      
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 4, 4, 20)         0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 2, 2, 25)          4525      
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 1, 1, 25)         0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 25)                0         
                                                                 
 dense (Dense)               (None, 10)                260       
                                                                 
=================================================================
Total params: 9,355
Trainable params: 9,355
Non-trainable params: 0
_________________________________________________________________
```

In [28]:

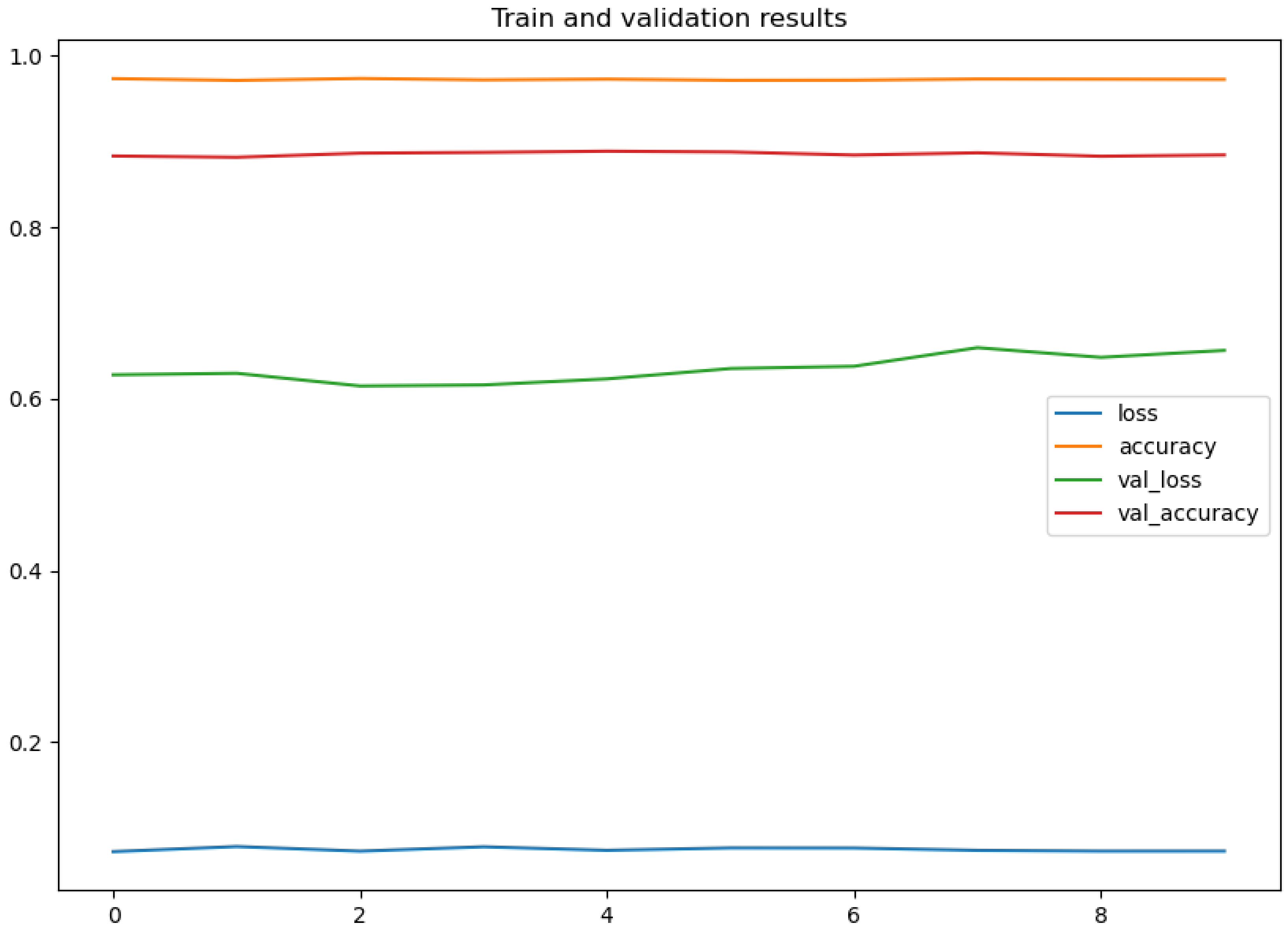
```
# 10. Train the CNN model
#
# You set the number of epochs and batch size in this code block

starting_time = time.time()
print("Training started...")

# Original: epochs = 5, batch_size = 32
history = model.fit(X_train, y_train,
                    epochs=10,batch_size=64,
                    validation_data=(X_valid,y_valid), verbose=1
                    )
```

```
print("Training ended.")
ending_time = time.time()
print('Time to train model: {:.4f}'.format(ending_time - starting_time),'seconds.')
```

```
Training started...
Epoch 1/10
782/782 [==============================] - 22s 28ms/step - loss: 0.0724 - accuracy: 0.9731 - val_loss: 0.6278 - val_acc
uracy: 0.8829
Epoch 2/10
782/782 [==============================] - 21s 27ms/step - loss: 0.0783 - accuracy: 0.9710 - val_loss: 0.6295 - val_acc
uracy: 0.8814
Epoch 3/10
782/782 [==============================] - 21s 27ms/step - loss: 0.0729 - accuracy: 0.9733 - val_loss: 0.6148 - val_acc
uracy: 0.8862
Epoch 4/10
782/782 [==============================] - 23s 29ms/step - loss: 0.0780 - accuracy: 0.9716 - val_loss: 0.6160 - val_acc
uracy: 0.8871
Epoch 5/10
782/782 [==============================] - 23s 29ms/step - loss: 0.0739 - accuracy: 0.9725 - val_loss: 0.6230 - val_acc
uracy: 0.8885
Epoch 6/10
782/782 [==============================] - 23s 30ms/step - loss: 0.0767 - accuracy: 0.9711 - val_loss: 0.6351 - val_acc
uracy: 0.8876
Epoch 7/10
782/782 [==============================] - 23s 30ms/step - loss: 0.0766 - accuracy: 0.9713 - val_loss: 0.6377 - val_acc
uracy: 0.8840
Epoch 8/10
782/782 [==============================] - 23s 30ms/step - loss: 0.0739 - accuracy: 0.9726 - val_loss: 0.6594 - val_acc
uracy: 0.8866
Epoch 9/10
782/782 [==============================] - 23s 29ms/step - loss: 0.0730 - accuracy: 0.9725 - val_loss: 0.6481 - val_acc
uracy: 0.8826
Epoch 10/10
782/782 [==============================] - 21s 27ms/step - loss: 0.0730 - accuracy: 0.9722 - val_loss: 0.6563 - val_acc
uracy: 0.8841
Training ended.
Time to train model: 224.5344 seconds.
```

In [29]:

```
# 11. Plot the training and loss progress

pd.DataFrame(history.history).plot(title="Train and validation results",figsize=(10,7));
```



In [30]:

```
# 12. Get the accuracy of the class images test data

loss, accuracy = model.evaluate(test_norm, test_labels)
print('Test accuracy:' , accuracy * 100,'%')
```

```
5/5 [==============================] - 0s 6ms/step - loss: 0.2277 - accuracy: 0.9565
Test accuracy: 95.652174949646 %
```

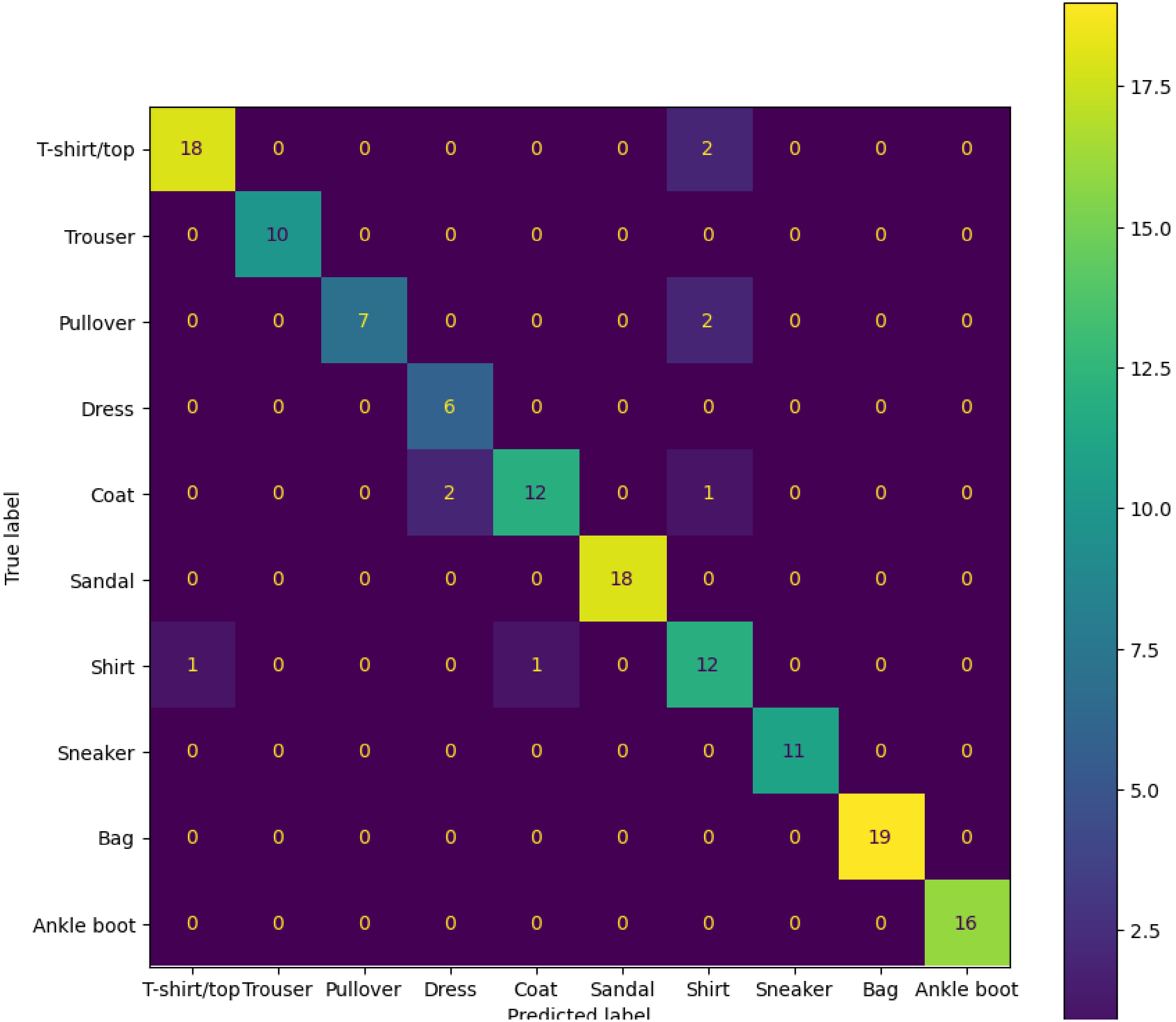
In [31]:

```
# 13. Make predictions for the confusion matrix

y_probs = model.predict(test_norm)

#Convert prediction probabilities into integers
y_preds = y_probs.argmax(axis=1)
```

```
5/5 [==============================] - 0s 4ms/step
```

In [30]:

```
# 14. Display the Confusion Matrix
cm=confusion_matrix(y_preds,test_labels)
#Plot
disp=ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=label_names)
fig, ax = plt.subplots(figsize=(10,10))
disp.plot(ax=ax);
```


T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot
True label
18 0 0 0 0 0 2 0 0 0
0 10 0 0 0 0 0 0 0 0
0 0 7 0 0 0 2 0 0 0
0 0 0 6 0 0 0 0 0 0
0 0 0 2 12 0 1 0 0 0
0 0 0 0 0 18 0 0 0 0
1 0 0 0 1 0 12 0 0 0
0 0 0 0 0 0 0 11 0 0
0 0 0 0 0 0 0 0 19 0
0 0 0 0 0 0 0 0 0 16
T-shirt/top
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle boot
Predicted label
17.5
15.0
12.5
10.0
7.5
5.0
2.5

In [ ]: