In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random

%matplotlib inline
sns.set_style("whitegrid")
```

In [2]:
```
fashion_train_df = pd.read_csv('fashion-mnist_train.csv',sep=',')
fashion_test_df = pd.read_csv('fashion-mnist_test.csv', sep = ',')
```

In [3]:
```
fashion_train_df.head()
```

Out[3]:

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	p
0	2	0	0	0	0	0	0	0	0	0	...	0	0	0	
1	9	0	0	0	0	0	0	0	0	0	...	0	0	0	
2	6	0	0	0	0	0	0	0	5	0	...	0	0	0	
3	0	0	0	0	1	2	0	0	0	0	...	3	0	0	
4	3	0	0	0	0	0	0	0	0	0	...	0	0	0	

5 rows × 785 columns

In [4]:
```
fashion_train_df.tail()
```

Out[4]:

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel7
59995	9	0	0	0	0	0	0	0	0	0	...	0	0	
59996	1	0	0	0	0	0	0	0	0	0	...	73	0	
59997	8	0	0	0	0	0	0	0	0	0	...	160	162	1
59998	8	0	0	0	0	0	0	0	0	0	...	0	0	
59999	7	0	0	0	0	0	0	0	0	0	...	0	0	

5 rows × 785 columns

In [5]:
```
fashion_test_df.head()
```

Out[5]:

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	p
0	0	0	0	0	0	0	0	0	9	8	...	103	87	56	
1	1	0	0	0	0	0	0	0	0	0	...	34	0	0	
2	2	0	0	0	0	0	0	14	53	99	...	0	0	0	
3	2	0	0	0	0	0	0	0	0	0	...	137	126	140	
4	3	0	0	0	0	0	0	0	0	0	...	0	0	0	

5 rows × 785 columns

In [6]:
```
fashion_test_df.tail()
```

Out[6]:

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel77
9995	0	0	0	0	0	0	0	0	0	0	...	32	23	1
9996	6	0	0	0	0	0	0	0	0	0	...	0	0	
9997	8	0	0	0	0	0	0	0	0	0	...	175	172	17
9998	8	0	1	3	0	0	0	0	0	0	...	0	0	
9999	1	0	0	0	0	0	0	0	140	119	...	111	95	7

5 rows × 785 columns

In [7]:
```
fashion_train_df.shape
```

Out[7]:
```
(60000, 785)
```

In [8]:
```
train = np.array(fashion_train_df, dtype='float32')
test = np.array(fashion_test_df, dtype='float32')
```

In [9]:
```
train.shape
```

Out[9]:
```
(60000, 785)
```

In [10]:
```
train
```

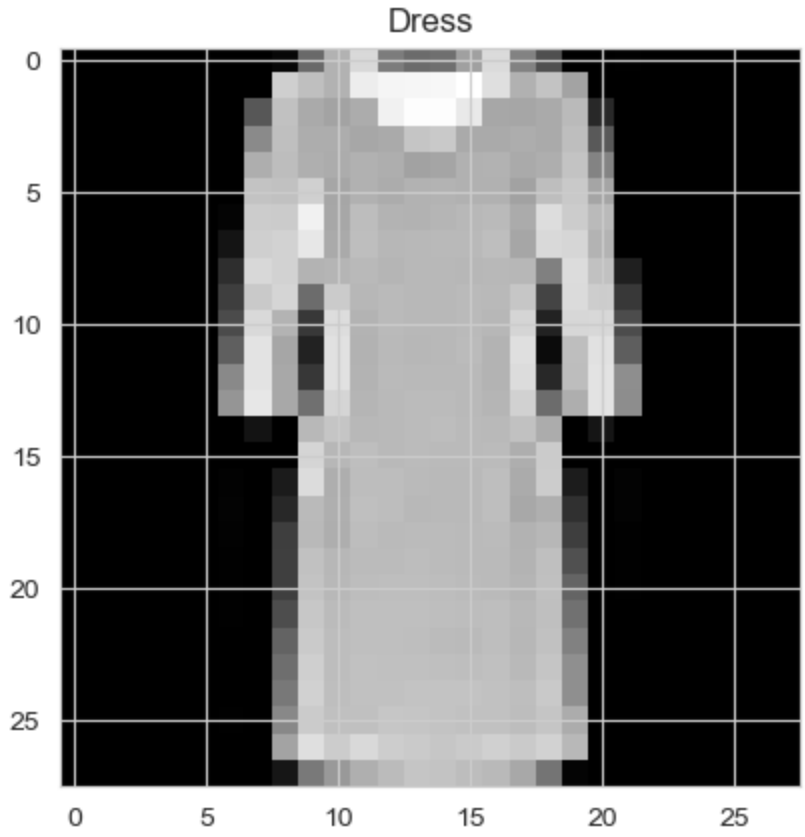
Out[10]:
```
array([[2., 0., 0., ..., 0., 0., 0.],
       [9., 0., 0., ..., 0., 0., 0.],
       [6., 0., 0., ..., 0., 0., 0.],
       ...,
       [8., 0., 0., ..., 0., 0., 0.],
       [8., 0., 0., ..., 0., 0., 0.],
       [7., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

In [11]:
```
test
```

Out[11]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [2., 0., 0., ..., 0., 0., 0.],
       ...,
       [8., 0., 0., ..., 0., 1., 0.],
       [8., 0., 1., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

In [12]:
```
class_names = ['T_shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']


i = random.randint(1,60000)
plt.imshow(train[i,1:].reshape((28,28)))

plt.imshow(train[i,1:].reshape((28,28)) , cmap = 'gray')
label_index = fashion_train_df["label"][i]
plt.title(f"{class_names[label_index]}")
```

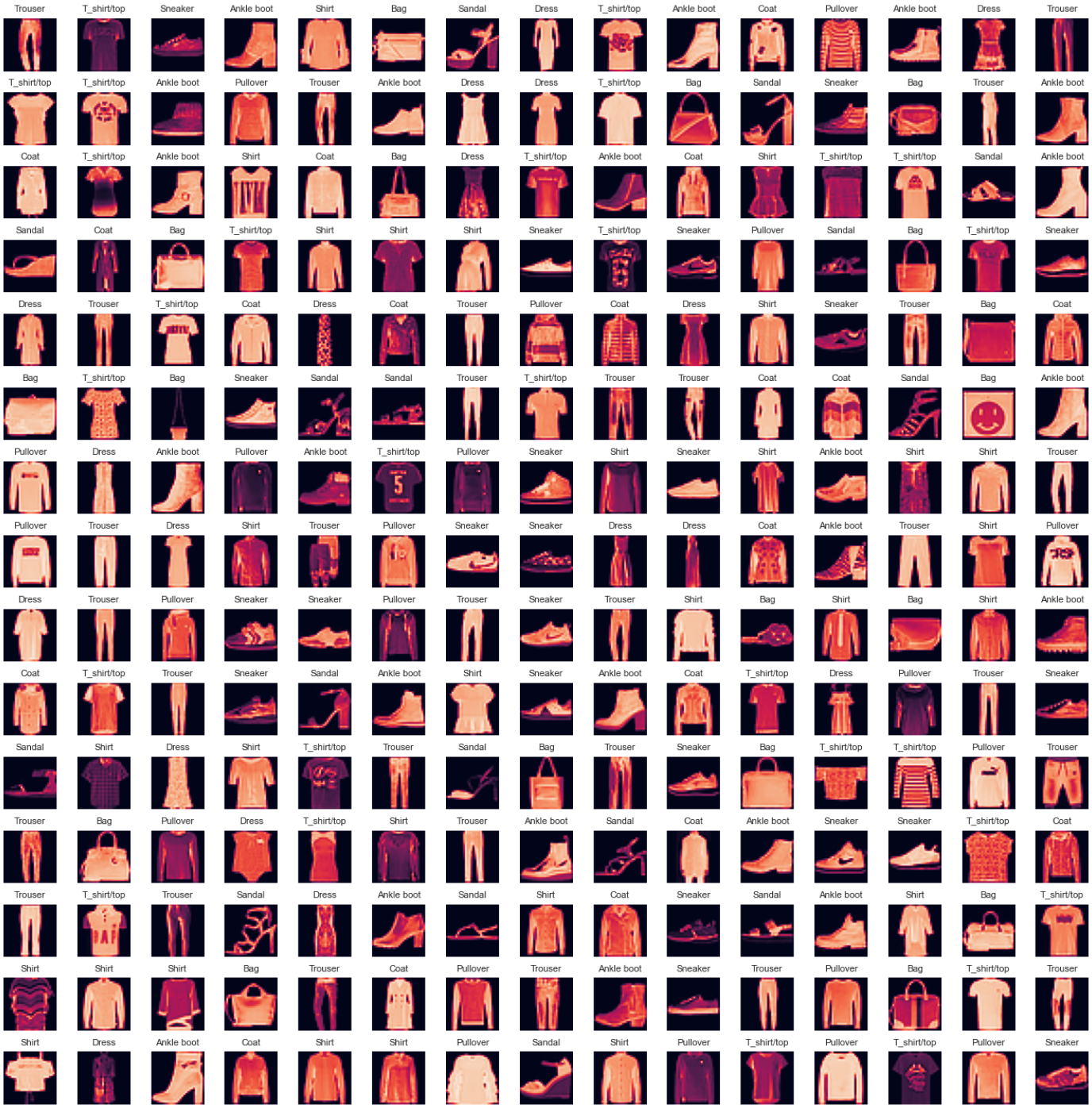
Out[12]:
```
Text(0.5, 1.0, 'Dress')
```



In [13]:
```
label = train[i,0]
label
```

Out[13]:
```
3.0
```

In [14]:
```
W_grid = 15
L_grid = 15



fig, axes = plt.subplots(L_grid, W_grid, figsize=(17,17))

axes = axes.ravel()

n_train = len(train)

for i in np.arange(0, W_grid * L_grid):

    index = np.random.randint(0, n_train)
    axes[i].imshow( train[index,1:].reshape((28,28)) )
    label_index = int(train[index,0])
    axes[i].set_title(class_names[label_index], fontsize=8)
    axes[i].axis('off')

plt.subplots_adjust(hspace=0.4)
```



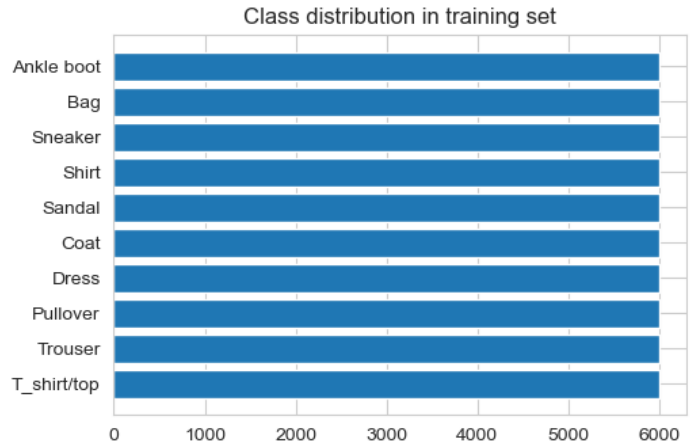
In [15]:
```
X_train = train[:, 1:] / 255
y_train = train[:, 0]

X_test = test[:, 1:] / 255
y_test = test[:,0]
```

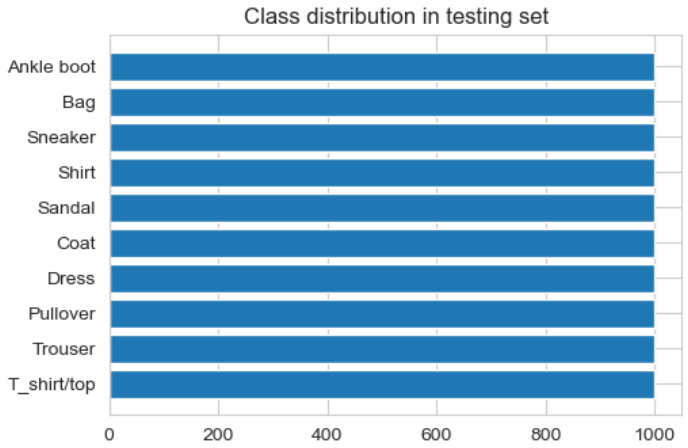
In [16]:
```
plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(X_train[i].reshape((28,28)), cmap=plt.cm.binary)
    label_index = int(y_train[i])
    plt.title(class_names[label_index])
plt.show()
plt.tight_layout()
```


```
<Figure size 640x480 with 0 Axes>
```

In [17]:
```
plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
classes, counts = np.unique(y_train, return_counts=True)
plt.barh(class_names, counts)
plt.title('Class distribution in training set')

plt.subplot(2, 2, 2)
classes, counts = np.unique(y_test, return_counts=True)
plt.barh(class_names, counts)
plt.title('Class distribution in testing set')
```

Out[17]:
```
Text(0.5, 1.0, 'Class distribution in testing set')
```





In [18]:
```
from sklearn.model_selection import train_test_split

X_train, X_validate, y_train, y_validate = train_test_split(X_train, y_train, test_size=
```

In [19]:
```
print(X_train.shape)
print(y_train.shape)
```

```
(48000, 784)
(48000,)
```

In [20]:
```
X_train = X_train.reshape(X_train.shape[0], * (28, 28, 1))
X_test = X_test.reshape(X_test.shape[0], * (28, 28, 1))
X_validate = X_validate.reshape(X_validate.shape[0], * (28, 28, 1))
```

In [21]:
```
print(X_train.shape)
print(y_train.shape)
print(X_validate.shape)
print(y_validate.shape)
```

```
(48000, 28, 28, 1)
(48000,)
(12000, 28, 28, 1)
(12000,)
```

In [22]:
```
import keras
import tensorflow as tf
```

In [23]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout, Batch
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import TensorBoard
```

In [24]:
```
cnn_model = Sequential()

cnn_model.add(Conv2D(filters=32, kernel_size=(3, 3), input_shape=(28,28,1), activation='
cnn_model.add(BatchNormalization())
cnn_model.add(Conv2D(filters=32, kernel_size=(3, 3), input_shape=(28,28,1), activation='
cnn_model.add(BatchNormalization())
cnn_model.add(MaxPooling2D(pool_size=(2, 2)))
cnn_model.add(Dropout(0.2))

cnn_model.add(Conv2D(filters=64, kernel_size=(3, 3), input_shape=(28,28,1), activation='
cnn_model.add(BatchNormalization())
cnn_model.add(Conv2D(filters=64, kernel_size=(3, 3), input_shape=(28,28,1), activation='
cnn_model.add(BatchNormalization())
cnn_model.add(MaxPooling2D(pool_size=(2, 2)))
cnn_model.add(Dropout(0.2))

cnn_model.add(Flatten())
```

```
cnn_model.add(Dense(units=128, activation='relu'))
cnn_model.add(Dropout(0.2))
cnn_model.add(Dense(units=10, activation='softmax'))
```

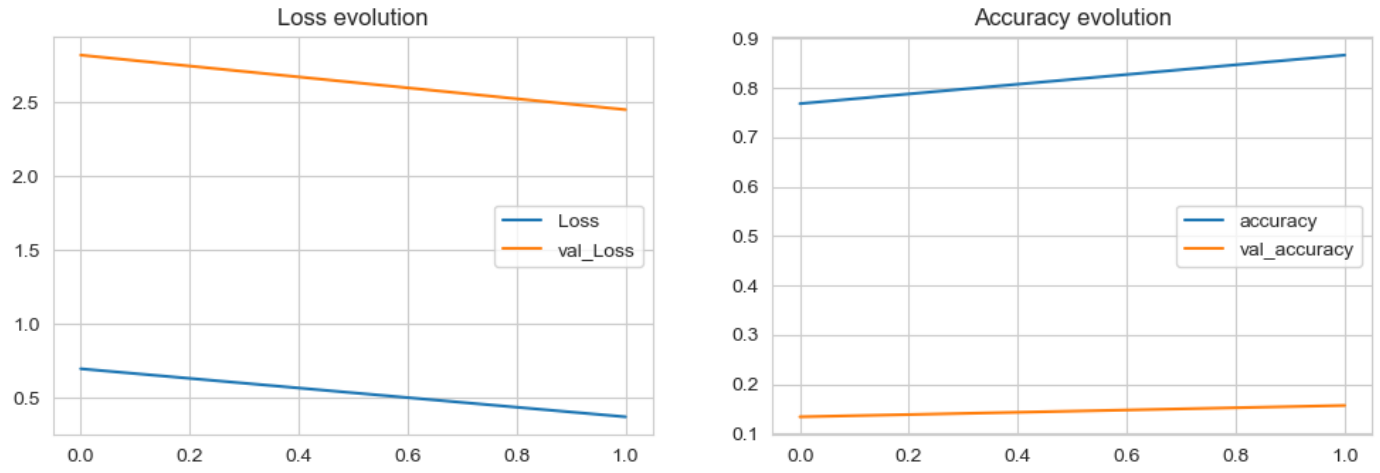
In [25]:
```
METRICS = [
    'accuracy',
    tf.keras.metrics.Precision(name='precision'),
    tf.keras.metrics.Recall(name='recall')
]

cnn_model.compile(loss ='sparse_categorical_crossentropy', optimizer='adam' ,metrics=['a
```

In [26]:
```
epochs = 2
batch_size = 512

history = cnn_model.fit(
    X_train, y_train,
    batch_size=batch_size,
    epochs=epochs,
    verbose=1,
    validation_data=(X_validate, y_validate)
)
```

```
Epoch 1/2
94/94 [==============================] - 66s 672ms/step - loss: 0.6949 - accuracy: 0.768
0 - val_loss: 2.8204 - val_accuracy: 0.1338
Epoch 2/2
94/94 [==============================] - 63s 666ms/step - loss: 0.3691 - accuracy: 0.866
3 - val_loss: 2.4509 - val_accuracy: 0.1568
```

In [27]:
```
plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
plt.plot(history.history['loss'], label='Loss')
plt.plot(history.history['val_loss'], label='val_Loss')
plt.legend()
plt.title('Loss evolution')

plt.subplot(2, 2, 2)
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.legend()
plt.title('Accuracy evolution')
```

Out[27]:
```
Text(0.5, 1.0, 'Accuracy evolution')
```



In [28]:
```
evaluation = cnn_model.evaluate(X_test, y_test)
print(f'Test Accuracy : {evaluation[1]:.3f}')
```

```
313/313 [==============================] - 5s 16ms/step - loss: 2.4612 - accuracy: 0.152
6
Test Accuracy : 0.153
```

In [29]:
```
predicted_classes = cnn_model.predict(X_test)
predicted_classes = np.argmax(predicted_classes, axis=1)
```

```
313/313 [==============================] - 5s 16ms/step
```

In [30]:
```
test_img = X_test[0]
prediction = cnn_model.predict(X_test)
prediction[0]
```

Out[30]:
```
313/313 [==============================] - 5s 15ms/step
array([8.1267291e-01, 8.3292136e-03, 4.1785348e-02, 1.6250428e-02,
       2.5383543e-04, 2.1408116e-03, 8.8493310e-02, 1.1616654e-02,
       1.3512810e-02, 4.9446803e-03], dtype=float32)
```

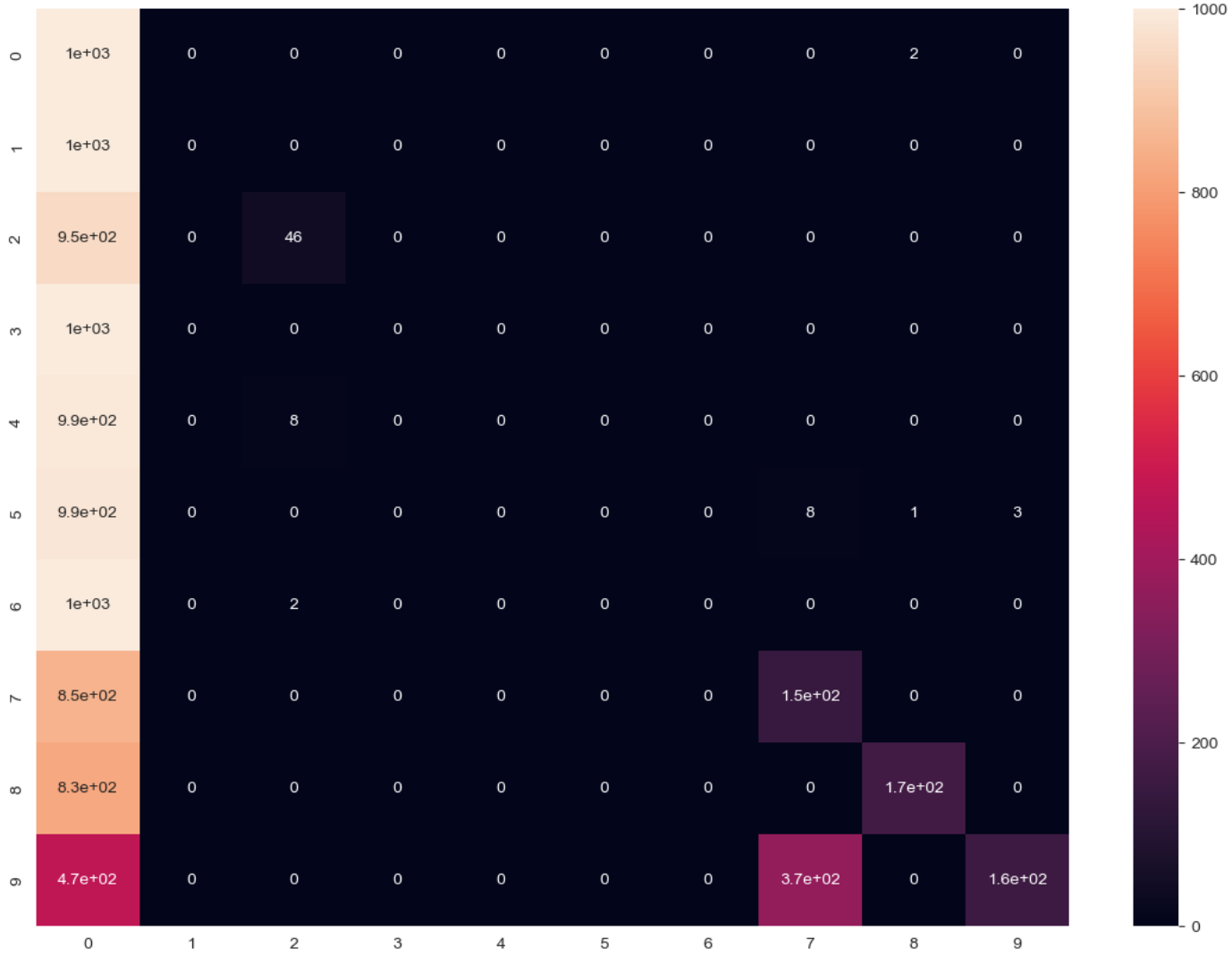
In [31]:
```
np.argmax(prediction[0])
```

Out[31]:
```
0
```

In [32]:
```
L = 5
W = 5
fig, axes = plt.subplots(L, W, figsize = (12,12))
axes = axes.ravel()

for i in np.arange(0, L * W):
    axes[i].imshow(X_test[i].reshape(28,28))
    axes[i].set_title(f"Prediction Class = {(predicted_classes[i]):0.1f}\n True Class =
    axes[i].axis('off')

plt.subplots_adjust(wspace=0.5)
```

In [33]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, predicted_classes)
plt.figure(figsize = (14,10))
sns.heatmap(cm, annot=True)
```

Out[33]:
```
<AxesSubplot:>
```



In [34]:
```
from sklearn.metrics import classification_report

num_classes = 10
target_names = [f"Class {i}" for i in range(num_classes)]

print(classification_report(y_test, predicted_classes, target_names = target_names))
```

```
              precision    recall  f1-score   support

     Class 0       0.11      1.00      0.20      1000
     Class 1       0.00      0.00      0.00      1000
     Class 2       0.82      0.05      0.09      1000
     Class 3       0.00      0.00      0.00      1000
     Class 4       0.00      0.00      0.00      1000
     Class 5       0.00      0.00      0.00      1000
     Class 6       0.00      0.00      0.00      1000
     Class 7       0.28      0.15      0.19      1000
     Class 8       0.98      0.17      0.30      1000
     Class 9       0.98      0.16      0.28      1000

    accuracy                           0.15     10000
   macro avg       0.32      0.15      0.11     10000
weighted avg       0.32      0.15      0.11     10000
```

```
C:\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricW
arning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pre
dicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricW
arning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pre
```

```
dicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricW
arning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pre
dicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```