Computational Intelligence Lab

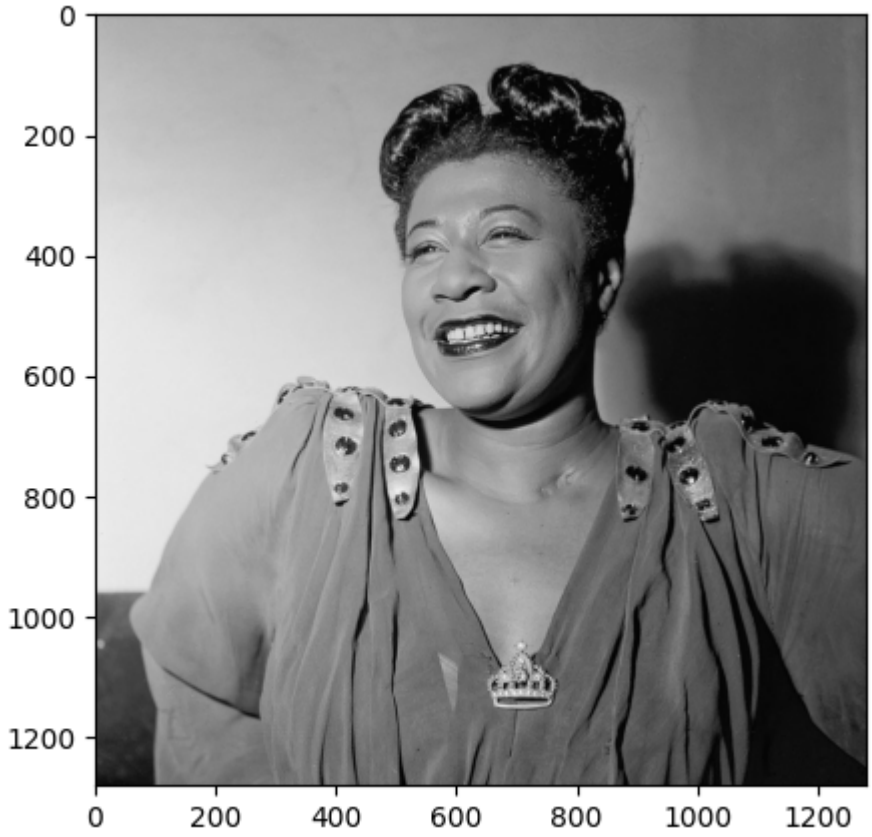
Assignment 3

Karolina Kotlowska IO lab 9:30 czwartek

3.1 Convolutional Neural Networks - Introduction

In [1]:
```
#Load an image
from PIL import Image
import requests
from io import BytesIO
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize'] = (5, 5)


response = requests.get('https://cdn.pixabay.com/photo/2016/05/11/18/57/e
img = Image.open(BytesIO(response.content))


plt.imshow(img)
```

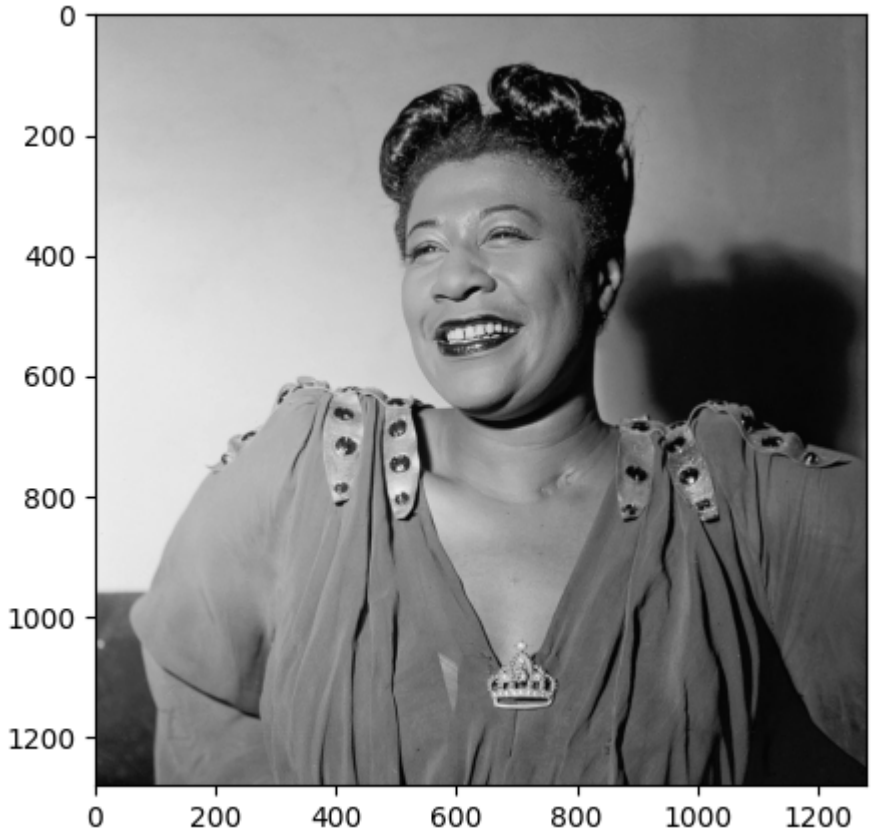
Out[1]:
```
<matplotlib.image.AxesImage at 0x11713dcf0>
```



In [2]:
```
import numpy as np
X_img = np.array(img)
print(f'X_img.shape={X_img.shape}')

X=X_img[:,:,0]
print(f'X.shape={X.shape}')
plt.rcParams['image.cmap'] = 'gray'
plt.imshow(X)
```

```
X_img.shape=(1280, 1280, 3)
X.shape=(1280, 1280)
```

Out[2]:
```
<matplotlib.image.AxesImage at 0x117245810>
```



Convolution 2D

During the convolution a small filter F slides along an image X. Then the filter elements are multiplied by corresponding pixel values and summed. Convolution can be used to blurr an image, but also detect edges or extract gradients of pixel intensity.

Diagram

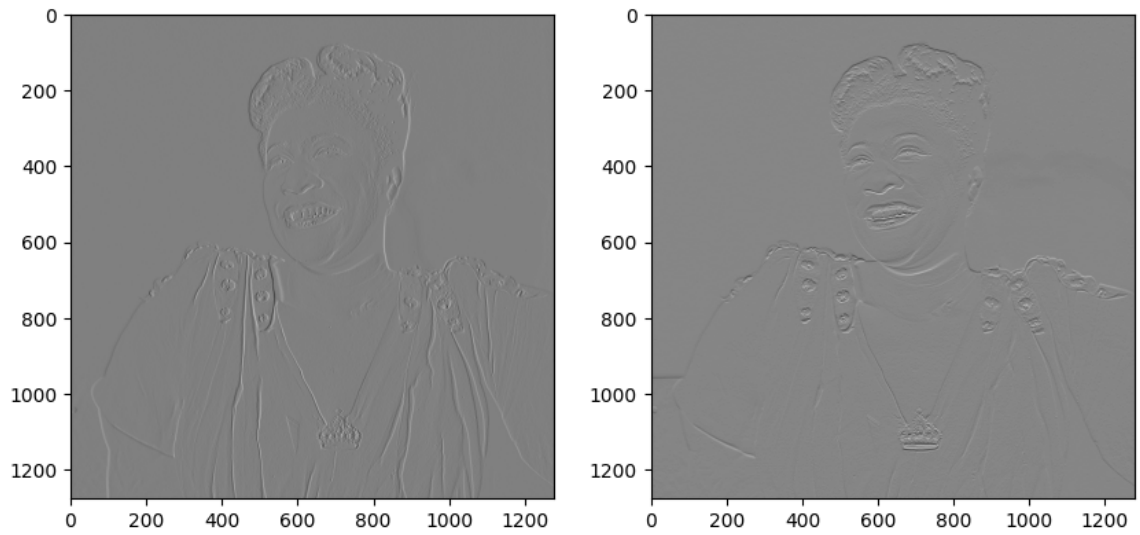
In [3]:
```
def conv2D(X,F,bias=0): #f - filter
  w = F.shape[1]//2
  h = F.shape[0]//2
  Y=[ [ np.sum(X[i-h:i+h+1,j-w:j+w+1]*F) + bias for j in np.arange(w,X.sh
  return np.array(Y)
```

In [4]:
```
def apply_two_convolutions(X):
  F_sobel_y=np.array([[-1,-2,-1],[0,0,0],[1,2,1]])
  F_sobel_x=np.array([[-1,0,1],[-2,0,2],[-1,0,1]])

  XX = conv2D(X,F_sobel_x,1)
  XY = conv2D(X,F_sobel_y,1)
  return XX,XY

AX,AY = apply_two_convolutions(X)
```

In [5]:
```
print(f'X.shape={X.shape} AX.shape={AX.shape} AY.shape={AY.shape}')
```

```
X.shape=(1280, 1280) AX.shape=(1278, 1278) AY.shape=(1278, 1278)
```

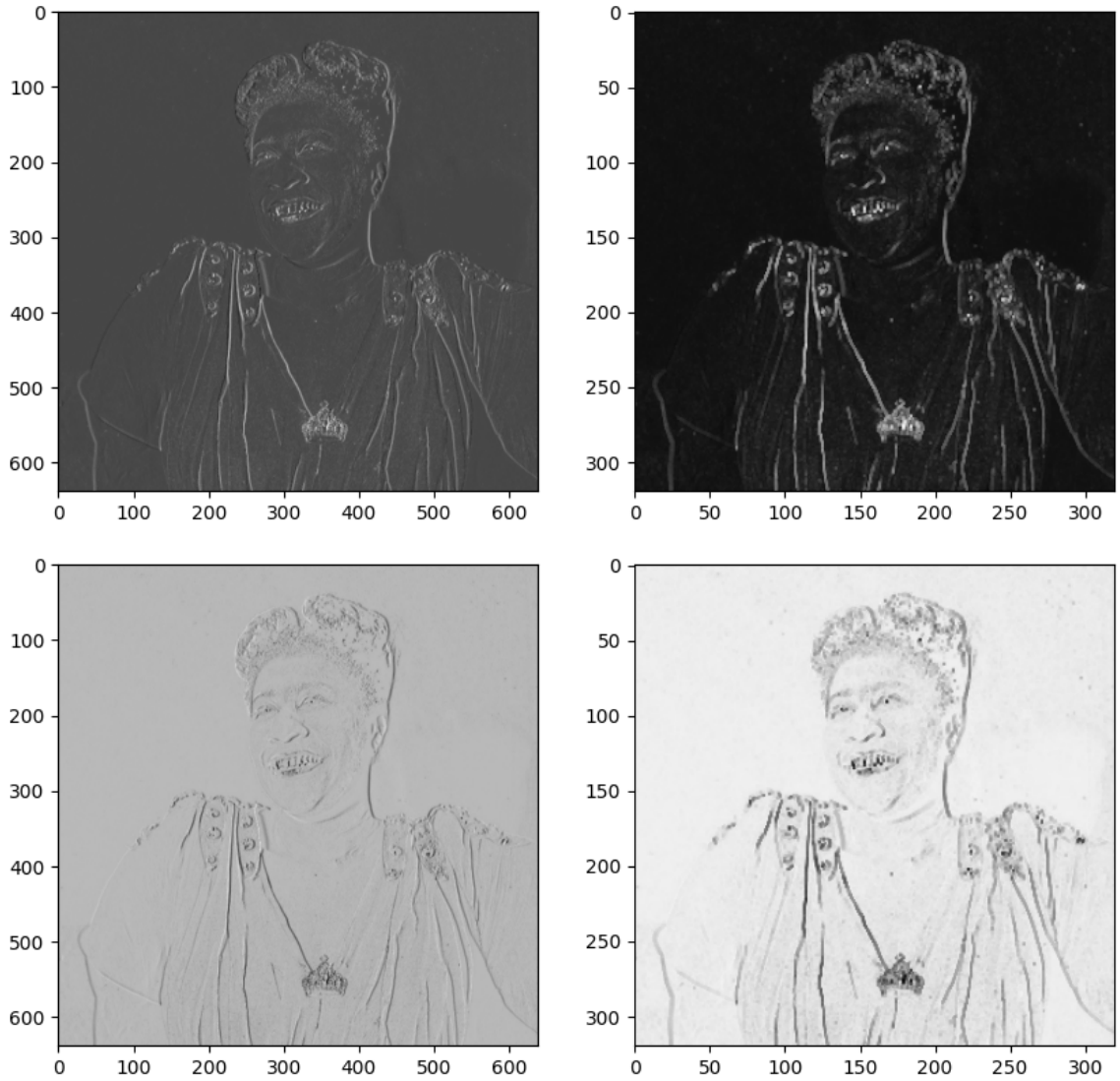
In [6]:
```
def plot(X,Y,cmap='gray'):
  fig = plt.figure(figsize=(10,10))
  plt.set_cmap(cmap)
  fig.add_subplot(1,2,1)
  # plt.axis('off')
  plt.imshow(X)
  fig.add_subplot(1,2,2)
  plt.imshow(Y)

plot(AX,AY)
```



During max pooling a window slides through an image (the strides are equal to the window sizes). At each position the maximal value of the pixel within the window is selected. Max pooling shrinks the image size.

In [7]:
```
def maxPool2D(X,shape):
  w = shape[1]
  h = shape[0]
  # print(f'h={h} w={w}')
  Y=[ [ np.max(X[i:i+h+1,j:j+w+1]) for j in np.arange(0,X.shape[1],shape[
  return np.array(Y)

AX=maxPool2D(AX,(2,2))
AY=maxPool2D(AX,(2,2))
plot(AX,AY)
plot(AX,AY,cmap='gray_r')
print(f'image size:{AX.shape}')
```

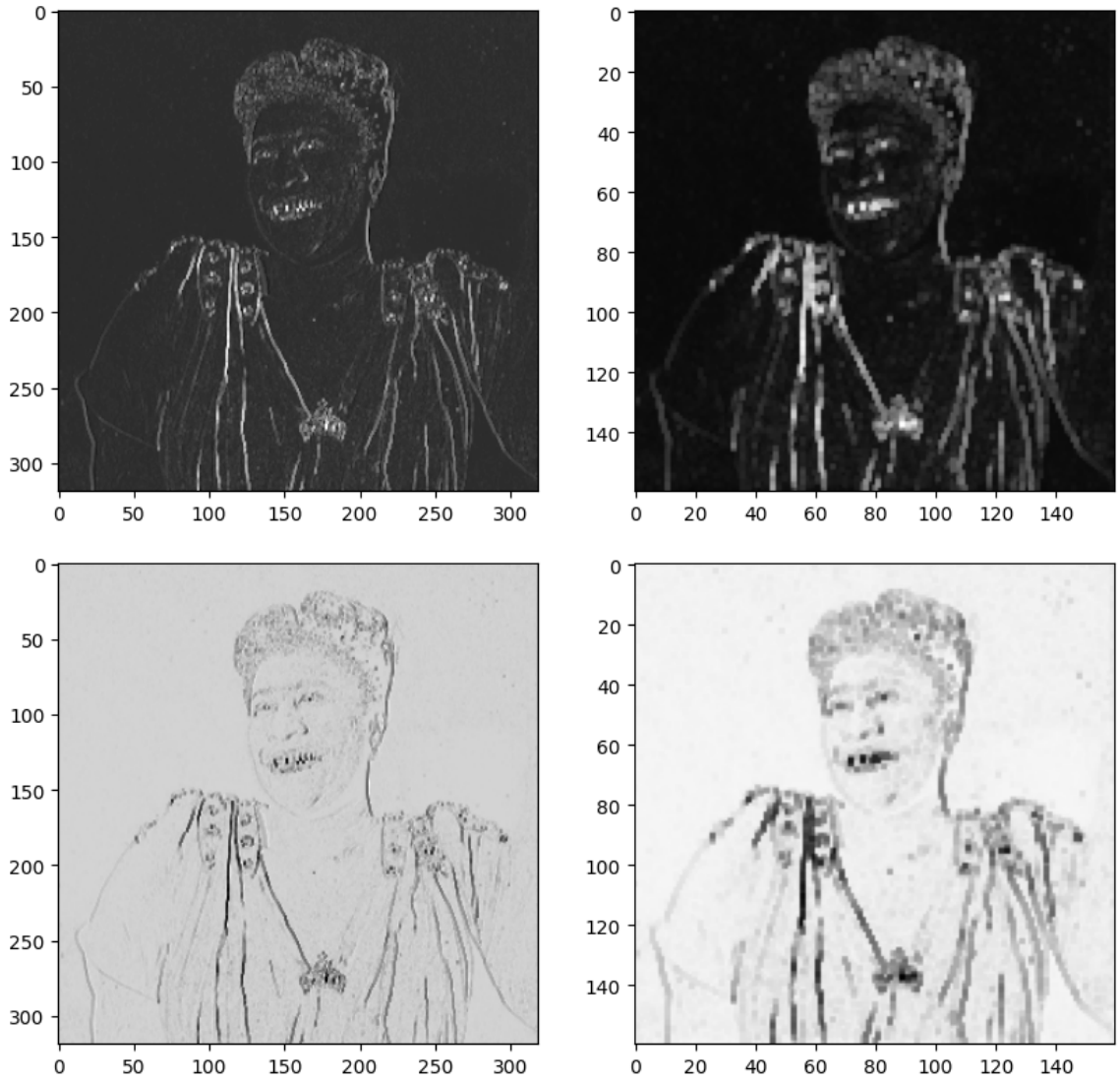
```
image size:(639, 639)
```



We will repeat the convolution and max-pooling steps several times...

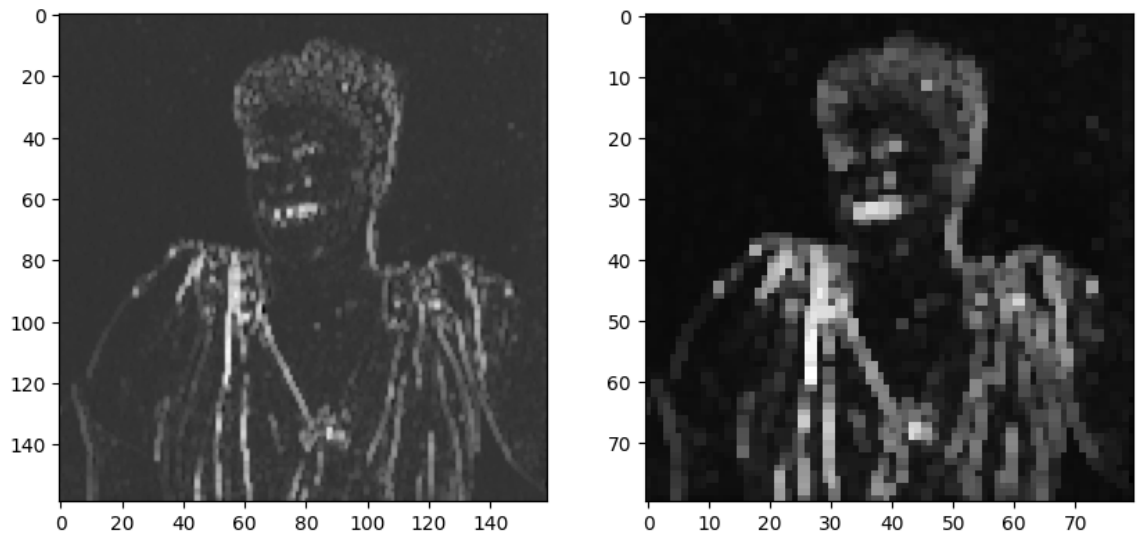
In [8]:
```
AX,AY = apply_two_convolutions(AX)
AX=maxPool2D(AX,(2,2))
AY=maxPool2D(AX,(2,2))
plot(AX,AY)
plot(AX,AY,cmap='gray_r')
print(f'image size:{AX.shape}')
```

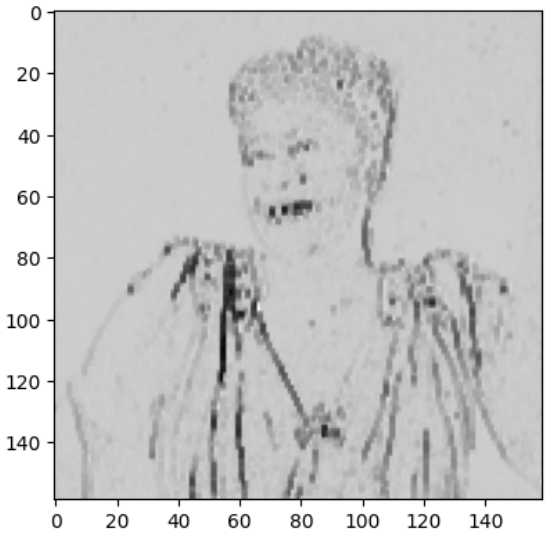
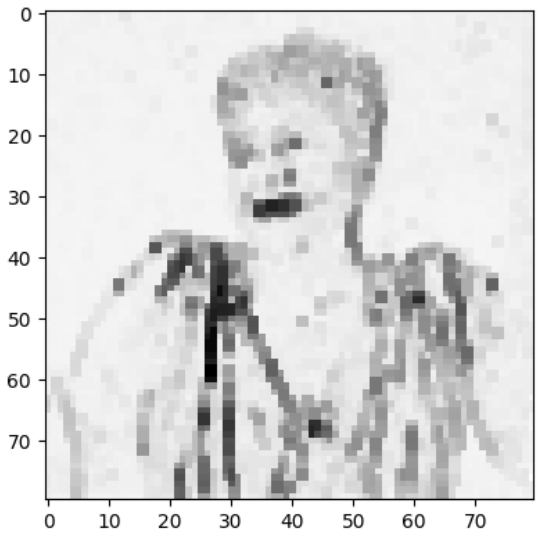
```
image size:(319, 319)
```



In [9]:
```
AX,AY = apply_two_convolutions(AX)
AX=maxPool2D(AX,(2,2))
AY=maxPool2D(AX,(2,2))
plot(AX,AY)
plot(AX,AY,cmap='gray_r')
print(f'image size:{AX.shape}')
```

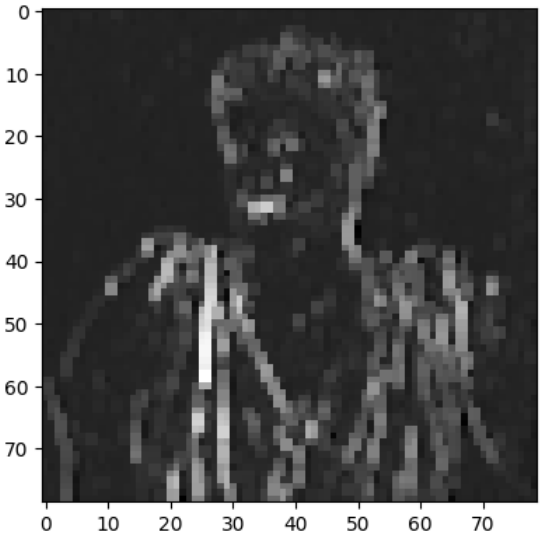
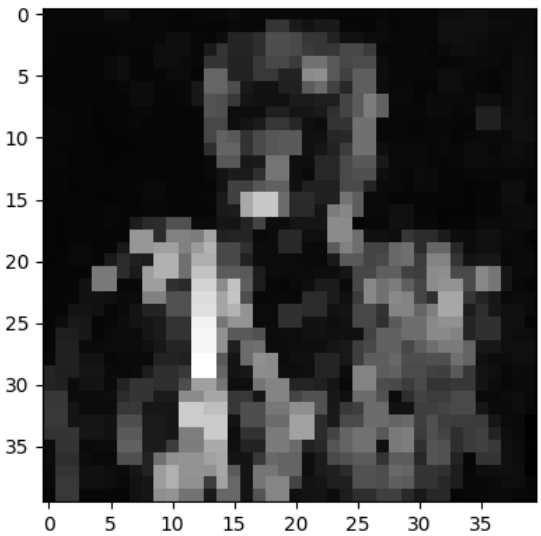
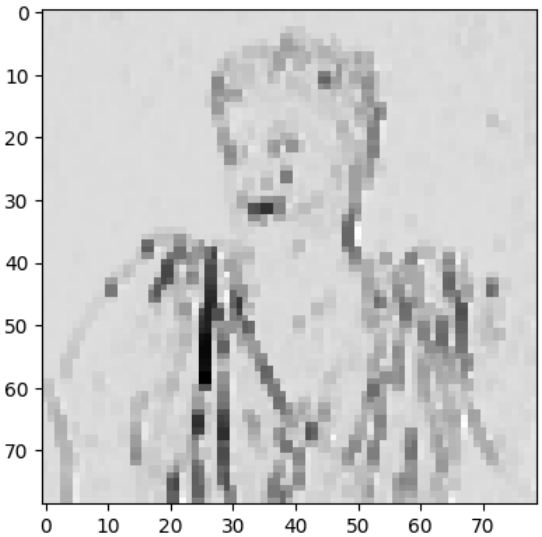
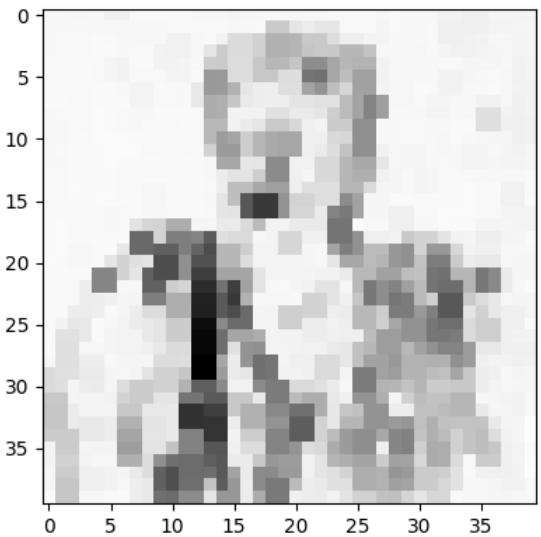
```
image size:(159, 159)
```



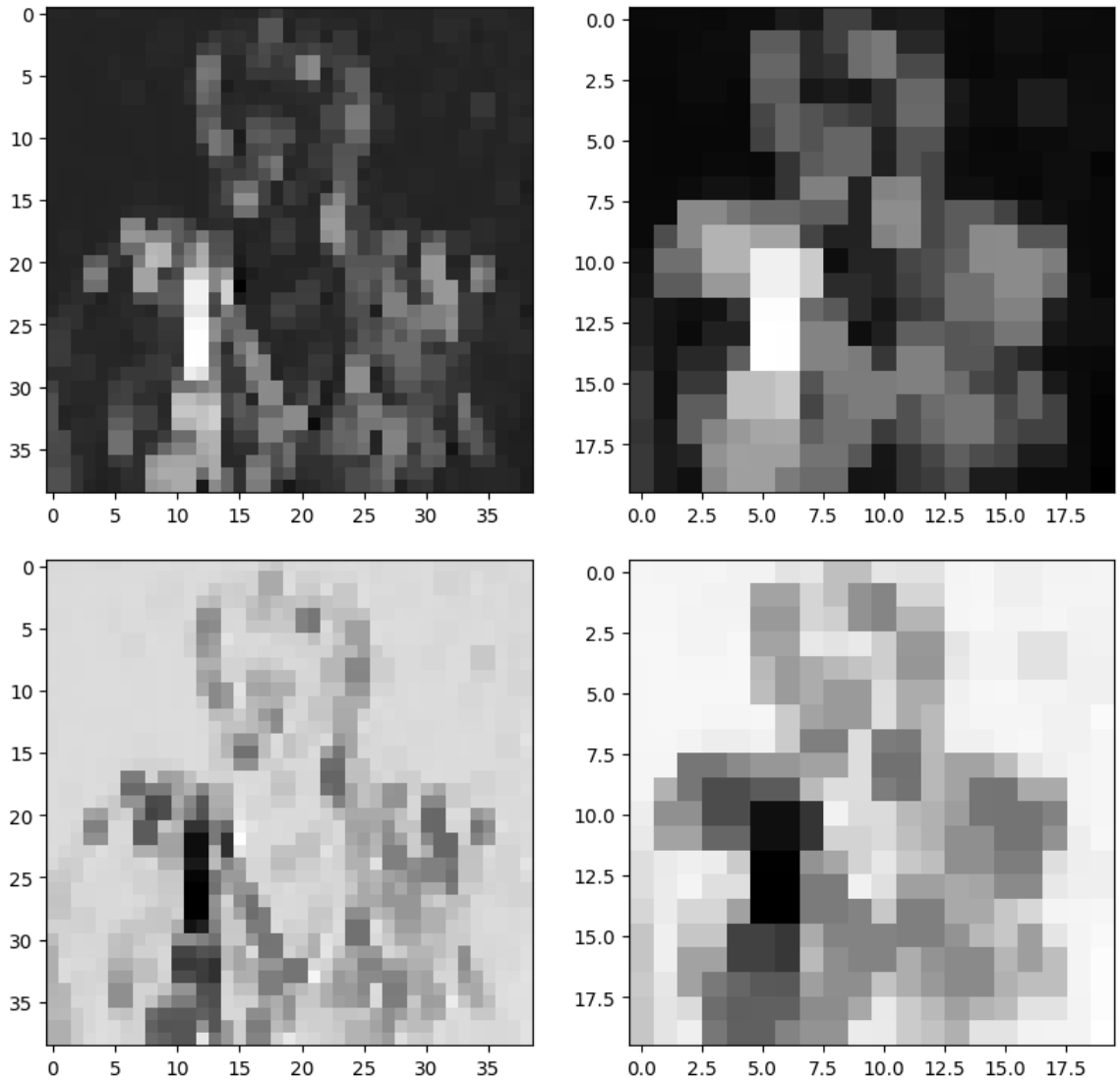
In [10]:
```
AX,AY = apply_two_convolutions(AX)
AX=maxPool2D(AX,(2,2))
AY=maxPool2D(AX,(2,2))
plot(AX,AY)
plot(AX,AY,cmap='gray_r')
print(f'image size:{AX.shape}')
```

```
image size:(79, 79)
```

In [11]:
```
AX,AY = apply_two_convolutions(AX)
AX=maxPool2D(AX,(2,2))
AY=maxPool2D(AX,(2,2))
plot(AX,AY)
plot(AX,AY,cmap='gray_r')
print(f'image size:{AX.shape}')
```

```
image size:(39, 39)
```



Probably the last convolution went too far...

Then the image is flattened (converted to a vector) and submitted as input to next processing steps (e.g. classification).

In [12]:
```
AX_flattended = AX.flatten()
print(f'After flattening: {AX_flattended.shape}')
```

```
After flattening: (1521,)
```

CNN

Typical process in image recogntion (old approach):

Feature extraction

1. Preprocess an image (eg. apply filters, convert to grayscale)
2. Select a convolution filter
3. Apply convolution in order to extract features, use a functions like scipy.signal.convolve2d, probably written in C
4. Optionally: apply max pooling
5. Optionally: repeat steps 2, 3, 4

Training and testing 6. Select a classifier, e.g. SVM 7. Train the classifier using vectors of extracted features as observations 8. Test its performance 8. If not satisfied return to 1

Basic ideas of CNNs:

- Feature extraction is included into the training process: convolutional layers are parts of the architecture
- No savvy filter selection is required: filters are learned (based on the loss function that expresses classification or regression goals)
- At a given steps a number of filters can be applied simultaneously adding the corresponding number of channels at output
- Architecture may contain multiple sequences of convolution and pooling steps. It is believed that this corresponds to extracting more and more abstract features, eg:
  - First sequence: extract contours
  - Second layer: extract such objects as eyes, noses
  - Third layer: extract faces, etc

TODO 3.1.1 Calculate the number of trainable parameters.

Provide the formula that gives the number of trainable parameters based on the values of defined variables.

$$P = weight_number + bias * filter_number$$

Fill in the table below

(img_cols,img_rows)	channels	filters	kernel size	Tensor shape after convolution	Tensor shape after pooling	number of parameters
(10,10)	1	1	(3,3)	(None, 8, 8, 1)	(None, 4, 4, 1)	10+17
(10,10)	1	5	(3,3)	(None, 8, 8, 5)	(None, 4, 4, 5)	50+81
(10,10)	3	1	(5,5)	(None, 6, 6, 1)	(None, 3, 3, 1)	76+10
(10,10)	3	5	(3,3)	(None, 8, 8, 5)	(None, 4, 4, 5)	140+81
(16,16)	3	64	(3,3)	(None, 14, 14, 64)	(None, 7, 7, 64)	1792+3137

Observe the following:

- None in an output shape is the dimension for multiple observations (here: images) in an input batch
- The same filter is applied to all image channels, then the bias term added.
- All filters have separate parameters
- Number of parameters for a dense layer is equal to the number of connections + one for the bias term

In [13]:
```
from keras import layers
from keras import models
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D

#variables
img_rows=10
img_cols=10
channels=1
filters=1
kernel_size=(3,3)

model = models.Sequential()
model.add(layers.Conv2D(filters, kernel_size=kernel_size, activation='rel
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(1))
model.summary()
```

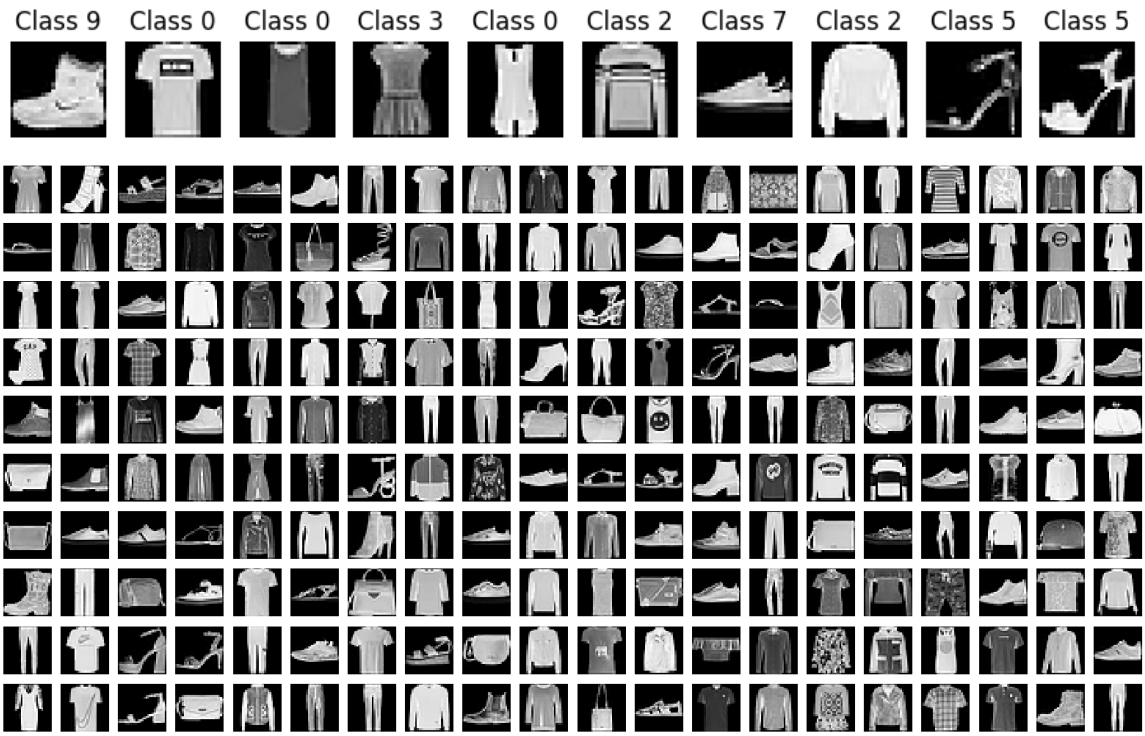
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 8, 8, 1)           10

 max_pooling2d (MaxPooling2D  (None, 4, 4, 1)          0
 )

 flatten (Flatten)           (None, 16)                0

 dense (Dense)               (None, 1)                 17

=================================================================
Total params: 27
Trainable params: 27
Non-trainable params: 0
_________________________________________________________________
```

3.2 Applying CNN to Fashion dataset

In [14]:
```
import numpy as np
from tensorflow.keras.datasets import fashion_mnist

(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
X=np.concatenate((np.array(x_train),np.array(x_test)),axis=0);
```

```
y=np.concatenate((np.array(y_train),np.array(y_test)),axis=0);
X=X/255
```

In [15]:
```
# Show a few sample clothes from the training set
import matplotlib.pyplot as plt
plt.rcParams['image.interpolation'] = 'nearest'
plt.rcParams['image.cmap'] = 'gray'

plt.rcParams['figure.figsize'] = (2.5, 2.5) # set default size of plots
col1 = 10
row1 = 1
fig = plt.figure(figsize=(col1, row1))
for index in range(0, col1*row1):
    fig.add_subplot(row1, col1, index + 1)
    plt.axis('off')
    plt.imshow(x_train[index]) # index of the sample picture
    plt.title("Class " + str(y_train[index]))
plt.show()

# Show a few sample clothes from the training set
plt.rcParams['figure.figsize'] = (1.0, 1.0) # set default size of plots
col2 = 20
row2 = 10
fig = plt.figure(figsize=(col2, row2))
for index in range(col1*row1, col1*row1 + col2*row2):
    fig.add_subplot(row2, col2, index - col1*row1 + 1)
    plt.axis('off')
    plt.imshow(x_train[index]) # index of the sample picture
plt.show()
```



In [16]:
```
print(f'X.shape={X.shape}')
img_rows = X.shape[1]
img_cols = X.shape[2]
num_classes = np.max(y)+1
```

```
X.shape=(70000, 28, 28)
```

In [17]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
```

Create a CNN model

In [18]:
```
from keras import layers
from keras import models
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
import tensorflow as tf

tf.random.set_seed(42)

model1 = models.Sequential()
model1.add(layers.Conv2D(8, (3, 3), activation='relu', input_shape=(X_tra
model1.add(layers.MaxPooling2D((2, 2)))
# Flatten the output to input data to the Dense layer
model1.add(layers.Flatten())
# Dense layers - similar to the previous model, but using the smaller num
model1.add(layers.Dense(32, activation='relu'))
model1.add(layers.Dense(num_classes, activation='softmax'))

model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_1 (Conv2D)           (None, 26, 26, 8)         80

 max_pooling2d_1 (MaxPooling  (None, 13, 13, 8)        0
 2D)

 flatten_1 (Flatten)         (None, 1352)              0

 dense_1 (Dense)             (None, 32)                43296

 dense_2 (Dense)             (None, 10)                330

=================================================================
Total params: 43,706
Trainable params: 43,706
Non-trainable params: 0
_________________________________________________________________
```

In [19]:
```
model1.compile(optimizer=tf.keras.optimizers.RMSprop(0.02),loss='sparse_c
```

In [20]:
```
epochs = 20
batch_size = 2048
hist = model1.fit(X_train, y_train, epochs = epochs, batch_size = batch_s
```

```
Epoch 1/20
2023-03-18 08:20:29.434085: W tensorflow/tsl/platform/profile_utils/cpu_
utils.cc:128] Failed to get CPU frequency: 0 Hz
```

```
23/23 [==============================] - 1s 46ms/step - loss: 1.3607 - a
ccuracy: 0.5545 - val_loss: 0.6524 - val_accuracy: 0.7589
Epoch 2/20
23/23 [==============================] - 1s 42ms/step - loss: 0.5530 - a
ccuracy: 0.7917 - val_loss: 0.5452 - val_accuracy: 0.7924
Epoch 3/20
23/23 [==============================] - 1s 44ms/step - loss: 0.4904 - a
ccuracy: 0.8175 - val_loss: 0.4169 - val_accuracy: 0.8440
Epoch 4/20
23/23 [==============================] - 1s 41ms/step - loss: 0.4326 - a
ccuracy: 0.8399 - val_loss: 0.4286 - val_accuracy: 0.8442
Epoch 5/20
23/23 [==============================] - 1s 41ms/step - loss: 0.3908 - a
ccuracy: 0.8552 - val_loss: 0.3900 - val_accuracy: 0.8585
Epoch 6/20
23/23 [==============================] - 1s 43ms/step - loss: 0.3619 - a
ccuracy: 0.8663 - val_loss: 0.4199 - val_accuracy: 0.8419
Epoch 7/20
23/23 [==============================] - 1s 42ms/step - loss: 0.3564 - a
ccuracy: 0.8660 - val_loss: 0.3261 - val_accuracy: 0.8810
Epoch 8/20
23/23 [==============================] - 1s 42ms/step - loss: 0.3263 - a
ccuracy: 0.8788 - val_loss: 0.3338 - val_accuracy: 0.8783
Epoch 9/20
23/23 [==============================] - 1s 41ms/step - loss: 0.3189 - a
ccuracy: 0.8802 - val_loss: 0.3278 - val_accuracy: 0.8819
Epoch 10/20
23/23 [==============================] - 1s 43ms/step - loss: 0.2961 - a
ccuracy: 0.8884 - val_loss: 0.3121 - val_accuracy: 0.8880
Epoch 11/20
23/23 [==============================] - 1s 45ms/step - loss: 0.2889 - a
ccuracy: 0.8906 - val_loss: 0.3143 - val_accuracy: 0.8895
Epoch 12/20
23/23 [==============================] - 1s 41ms/step - loss: 0.2788 - a
ccuracy: 0.8927 - val_loss: 0.3255 - val_accuracy: 0.8794
Epoch 13/20
23/23 [==============================] - 1s 46ms/step - loss: 0.2724 - a
ccuracy: 0.8966 - val_loss: 0.3077 - val_accuracy: 0.8894
Epoch 14/20
23/23 [==============================] - 1s 42ms/step - loss: 0.2603 - a
ccuracy: 0.8996 - val_loss: 0.3628 - val_accuracy: 0.8728
Epoch 15/20
23/23 [==============================] - 1s 46ms/step - loss: 0.2523 - a
ccuracy: 0.9029 - val_loss: 0.3016 - val_accuracy: 0.8942
Epoch 16/20
23/23 [==============================] - 1s 45ms/step - loss: 0.2480 - a
ccuracy: 0.9054 - val_loss: 0.3953 - val_accuracy: 0.8648
Epoch 17/20
23/23 [==============================] - 1s 41ms/step - loss: 0.2449 - a
ccuracy: 0.9066 - val_loss: 0.3951 - val_accuracy: 0.8674
Epoch 18/20
23/23 [==============================] - 1s 49ms/step - loss: 0.2278 - a
ccuracy: 0.9133 - val_loss: 0.3767 - val_accuracy: 0.8762
Epoch 19/20
23/23 [==============================] - 1s 42ms/step - loss: 0.2344 - a
ccuracy: 0.9098 - val_loss: 0.3363 - val_accuracy: 0.8822
Epoch 20/20
23/23 [==============================] - 1s 40ms/step - loss: 0.2141 - a
ccuracy: 0.9169 - val_loss: 0.3070 - val_accuracy: 0.8984
```

Compute scores on the validation set

In [21]:
```
val_loss1, val_acc1 = model1.evaluate(X_test, y_test)
print('validation accuracy:', val_acc1)
print('validation loss:', val_loss1)
```

```
722/722 [==============================] - 1s 1ms/step - loss: 0.3070 -
accuracy: 0.8984
validation accuracy: 0.8983982801437378
validation loss: 0.30697599053382874
```

Display the confusion matrix

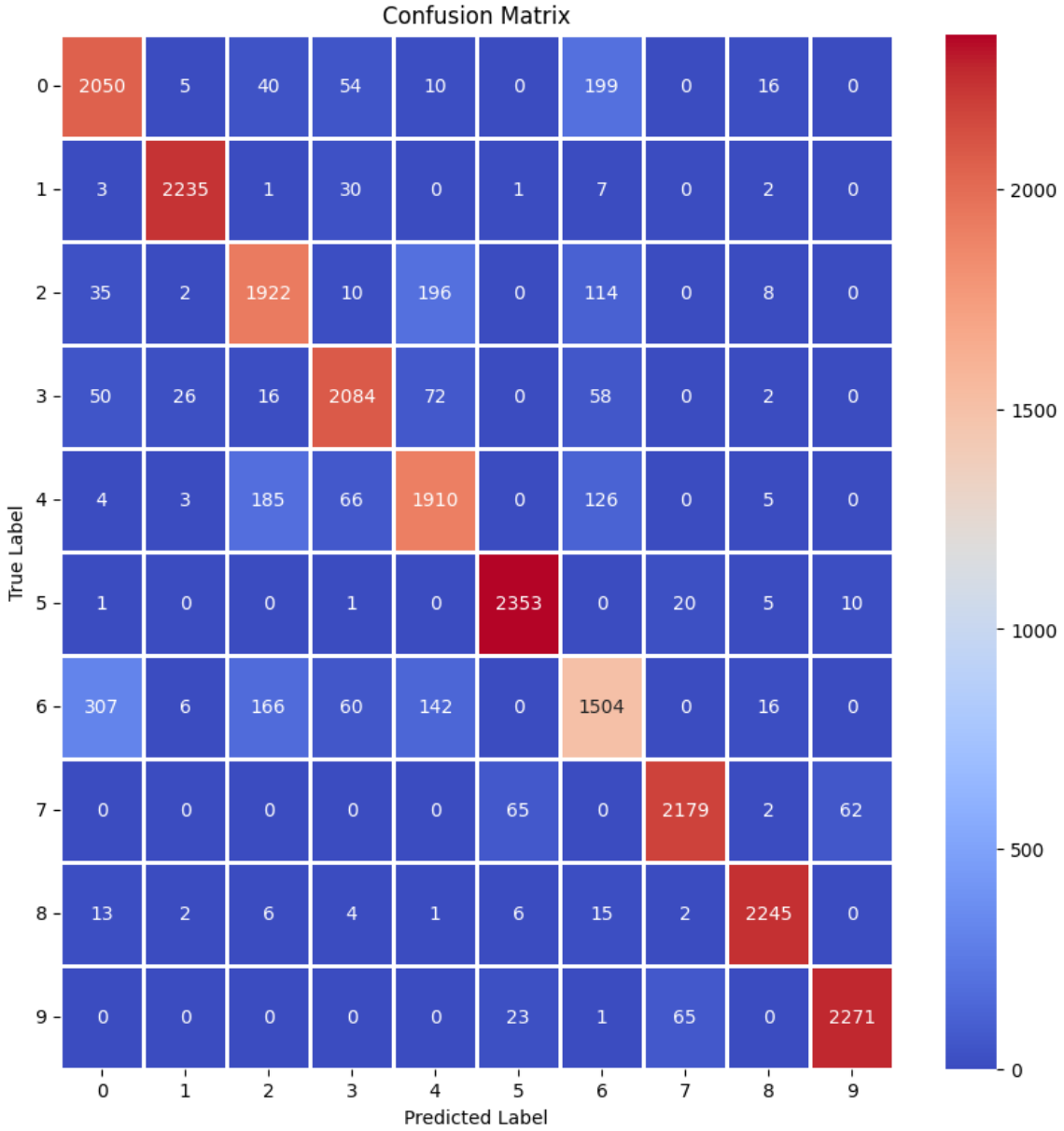
The function from the previous classes was split into two functions...

In [22]:
```
import seaborn as sns
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report
import matplotlib.pyplot as plt


# Define the confusion matrix for the results
def show_confusion_matrix(matrix, labels=None):
    num_classes = matrix.shape[0]
    if labels is None:
      labels = [str(i) for i in range(num_classes)]
    plt.figure(figsize=(num_classes, num_classes))
    hm = sns.heatmap(matrix,
                cmap='coolwarm',
                linecolor='white',
                linewidths=1,
                xticklabels=labels[0:num_classes],
                yticklabels=labels[0:num_classes],
                annot=True,
                fmt='d')
    plt.yticks(rotation = 0)  # Don't rotate (vertically) the y-axis labe
    # hm.set_ylim(0, len(matrix))
    plt.title('Confusion Matrix')
    plt.ylabel('True Label')
    plt.xlabel('Predicted Label')
    plt.show()

def compute_and_show_confusion_matrix(validations, predictions, labels=No
    matrix = metrics.confusion_matrix(validations, predictions)
    show_confusion_matrix(matrix, labels)
```

In [23]:
```
probs = model1.predict(X_test)
y_pred = np.argmax(probs,axis=1)
print(f'y_pred.shape={y_pred.shape}')
```

```
722/722 [==============================] - 1s 950us/step
y_pred.shape=(23100,)
```

In [24]:
```
compute_and_show_confusion_matrix(y_test, y_pred)
```



In [25]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.86      0.85      2374
           1       0.98      0.98      0.98      2279
           2       0.82      0.84      0.83      2287
           3       0.90      0.90      0.90      2308
           4       0.82      0.83      0.83      2299
           5       0.96      0.98      0.97      2390
           6       0.74      0.68      0.71      2201
           7       0.96      0.94      0.95      2308
           8       0.98      0.98      0.98      2294
           9       0.97      0.96      0.97      2360

    accuracy                           0.90     23100
   macro avg       0.90      0.90      0.90     23100
weighted avg       0.90      0.90      0.90     23100
```

3.3 Wrap it as a function

Digression related to Python technicalities:

- it is possible to define a function taking as parameter a dictionary mapping keywords to values
- and call it specifying keywords or passing a dictionary

In [26]:
```
def foo(**kwargs):
  print('foo')
  for k in kwargs:
    print(k,kwargs[k])

foo(ala=0,ma=1,kota=2)
```

```
foo
ala 0
ma 1
kota 2
```

In [27]:
```
def boo(**kwargs):
  ala = kwargs.get('ala',0)
  ma = kwargs.get('ma',111)
  kota = kwargs.get('kota',222)
  print('boo',ala,ma,kota)

boo(ala=23)

boo(**{'ala':123,'ma':12})
```

```
boo 23 111 222
boo 123 12 222
```

It is up to you what to do with unknown keywords

In [28]:
```
def woo(**kwargs):
  keywords = ['ala','ma','kota']
  for k in kwargs:
    if k not in keywords:
      raise ValueError(f'Unknown keyword {k}')
  ala = kwargs.get('ala',0)
  ma = kwargs.get('ma',1)
  kota = kwargs.get('kota',2)
  print('woo',ala,ma,kota)

woo(**{'ala':-1})
try:
  woo(**{'ala':-1,'ola':-2})
except Exception as e:
  print(e)
```

```
woo -1 1 2
Unknown keyword ola
```

Define a function building a CNN model

TODO 3.3.1 define at least two other models and name it using keywords, e.g 'model_1', 'model_2', etc

In [39]:
```
from keras import layers
from keras import models
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
import tensorflow as tf

def build_model(model_name,input_shape,num_classes,random_state=42):
  tf.random.set_seed(random_state)
  if model_name == 'model_0':
    model = models.Sequential()
    model.add(layers.Conv2D(8, (3, 3), activation='relu', input_shape=inp
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(32, activation='relu'))
    model.add(layers.Dense(num_classes, activation='softmax'))
    return model
  elif model_name == 'model_1':
    model = models.Sequential()
    model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=in
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(num_classes, activation='softmax'))
    return model
  elif model_name == 'model_2':
    model = models.Sequential()
    model.add(layers.Conv2D(16, (5, 5), activation='relu', input_shape=in
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(32, (5, 5), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(32, activation='relu'))
    model.add(layers.Dense(num_classes, activation='softmax'))
    return model
  else:
    return None
```

The function that trains a model, tests it and return scores

In [35]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score

def perform_test(X_train,y_train,X_test,y_test,**kwargs):
  model_name = kwargs['model_name']
  epochs = kwargs.get('epochs',20)
  learning_rate = kwargs.get('learning_rate',0.01)
  batch_size = kwargs.get('batch_size',1024)
  random_seed = kwargs.get('random_seed',42)
  returnModel = kwargs.get('returnModel',True)
```

```
  input_shape = list(X_train.shape[1:])
  input_shape.append(1)
  input_shape=tuple(input_shape)

  num_classes = np.max(y_train)+1
  model = build_model(model_name,input_shape=input_shape,num_classes=num_
  model.compile(optimizer=tf.keras.optimizers.RMSprop(learning_rate),
                loss='sparse_categorical_crossentropy',metrics=['accuracy
  model.fit(X_train,y_train,epochs=epochs,batch_size=batch_size)
  preds = model.predict(X_test)
  y_pred = np.argmax(preds,axis=1)
  print(f'y_pred {y_pred.shape} {y_test.shape}')
  results = {}
  results['confusion_matrix'] = confusion_matrix(y_pred,y_test)
  results['accuracy'] = accuracy_score(y_pred,y_test)
  results['precision'] = precision_score(y_pred,y_test,average='macro')
  results['recall'] = recall_score(y_pred,y_test,average='macro')
  results['f1'] = f1_score(y_pred,y_test,average='macro')
  if returnModel:
    results['model'] = model
  return results
```

Configurations

We will define a number of configurations to be tested, Each configuration defines the model name and a number of hyperparameters.

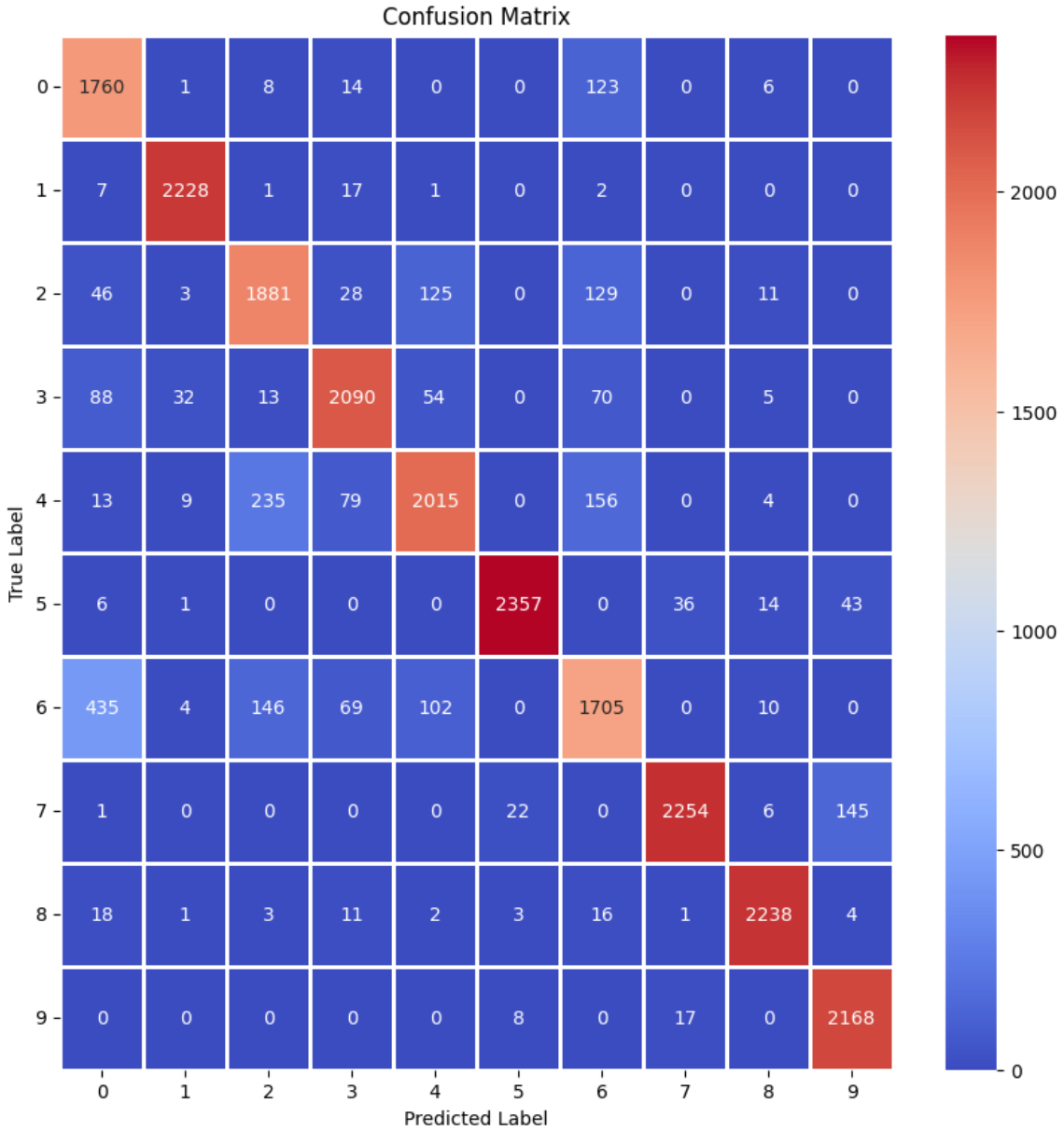
In [41]:
```
configurations={
    'First configuration':{'model_name':'model_0','epochs':20,'learning_r
    'Second configuration':{'model_name':'model_1','epochs':10,'learning_
    'Third configuration':{'model_name':'model_2','epochs':10,'learning_r
    'Fourth configuration':{'model_name':'model_0','epochs':150,'learning
    'Fifth configuration':{'model_name':'model_1','epochs':5,'learning_ra
    'Sixth configuration':{'model_name':'model_2','epochs':20,'learning_r
    'Seventh configuration':{'model_name':'model_0','epochs':30,'learning
}
```

TODO 3.3.2 Call the function perform_test for each configuration, print returned score values and display confusion matrix.

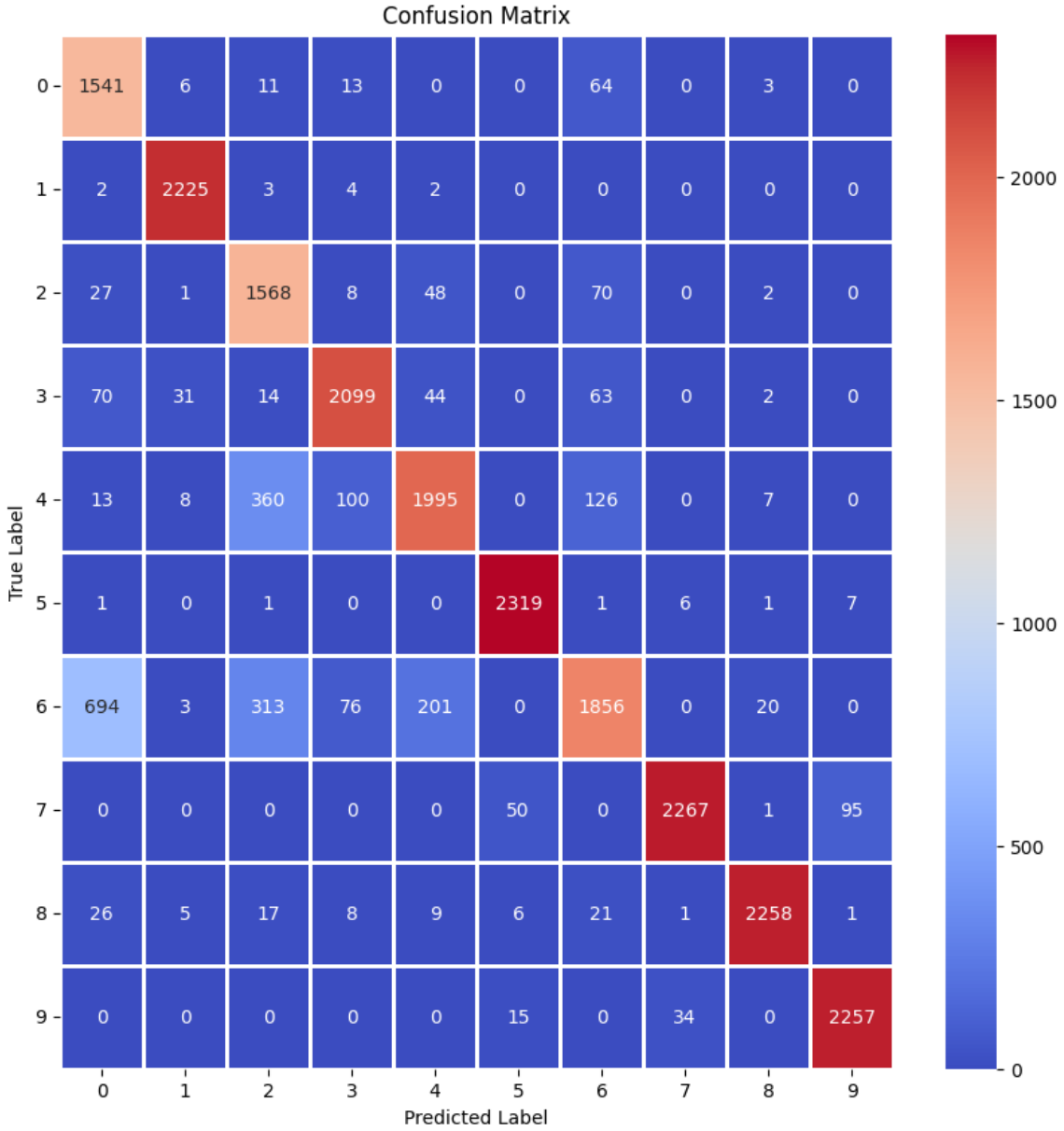
In [40]:
```
for cname in configurations:
  results = perform_test(X_train, y_train, X_test, y_test, **configuratio
  print(f'--- {cname} ---')
  # show results
  show_confusion_matrix(results['confusion_matrix'])
  print('Accuracy:\t %.3f' % results['accuracy'])
  print('Precision:\t %.3f' % results['precision'])
  print('Recall:\t %.3f' % results['recall'])
  print('F1:\t %.3f' % results['f1'])
```

```
Epoch 1/20
46/46 [==============================] - 1s 21ms/step - loss: 0.9383 - a
ccuracy: 0.6786
Epoch 2/20
46/46 [==============================] - 1s 19ms/step - loss: 0.4883 - a
ccuracy: 0.8229
Epoch 3/20
46/46 [==============================] - 1s 20ms/step - loss: 0.4126 - a
ccuracy: 0.8507
Epoch 4/20
46/46 [==============================] - 1s 20ms/step - loss: 0.3675 - a
ccuracy: 0.8677
Epoch 5/20
46/46 [==============================] - 1s 20ms/step - loss: 0.3412 - a
ccuracy: 0.8762
Epoch 6/20
46/46 [==============================] - 1s 20ms/step - loss: 0.3180 - a
ccuracy: 0.8830
Epoch 7/20
46/46 [==============================] - 1s 20ms/step - loss: 0.3026 - a
ccuracy: 0.8880
Epoch 8/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2884 - a
ccuracy: 0.8933
Epoch 9/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2719 - a
ccuracy: 0.8996
Epoch 10/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2622 - a
ccuracy: 0.9025
Epoch 11/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2561 - a
ccuracy: 0.9039
Epoch 12/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2453 - a
ccuracy: 0.9077
Epoch 13/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2360 - a
ccuracy: 0.9123
Epoch 14/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2272 - a
ccuracy: 0.9146
Epoch 15/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2192 - a
ccuracy: 0.9173
Epoch 16/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2096 - a
ccuracy: 0.9202
Epoch 17/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2051 - a
ccuracy: 0.9229
Epoch 18/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2004 - a
ccuracy: 0.9249
Epoch 19/20
46/46 [==============================] - 1s 21ms/step - loss: 0.1917 - a
ccuracy: 0.9274
Epoch 20/20
46/46 [==============================] - 1s 20ms/step - loss: 0.1841 - a
ccuracy: 0.9297
```

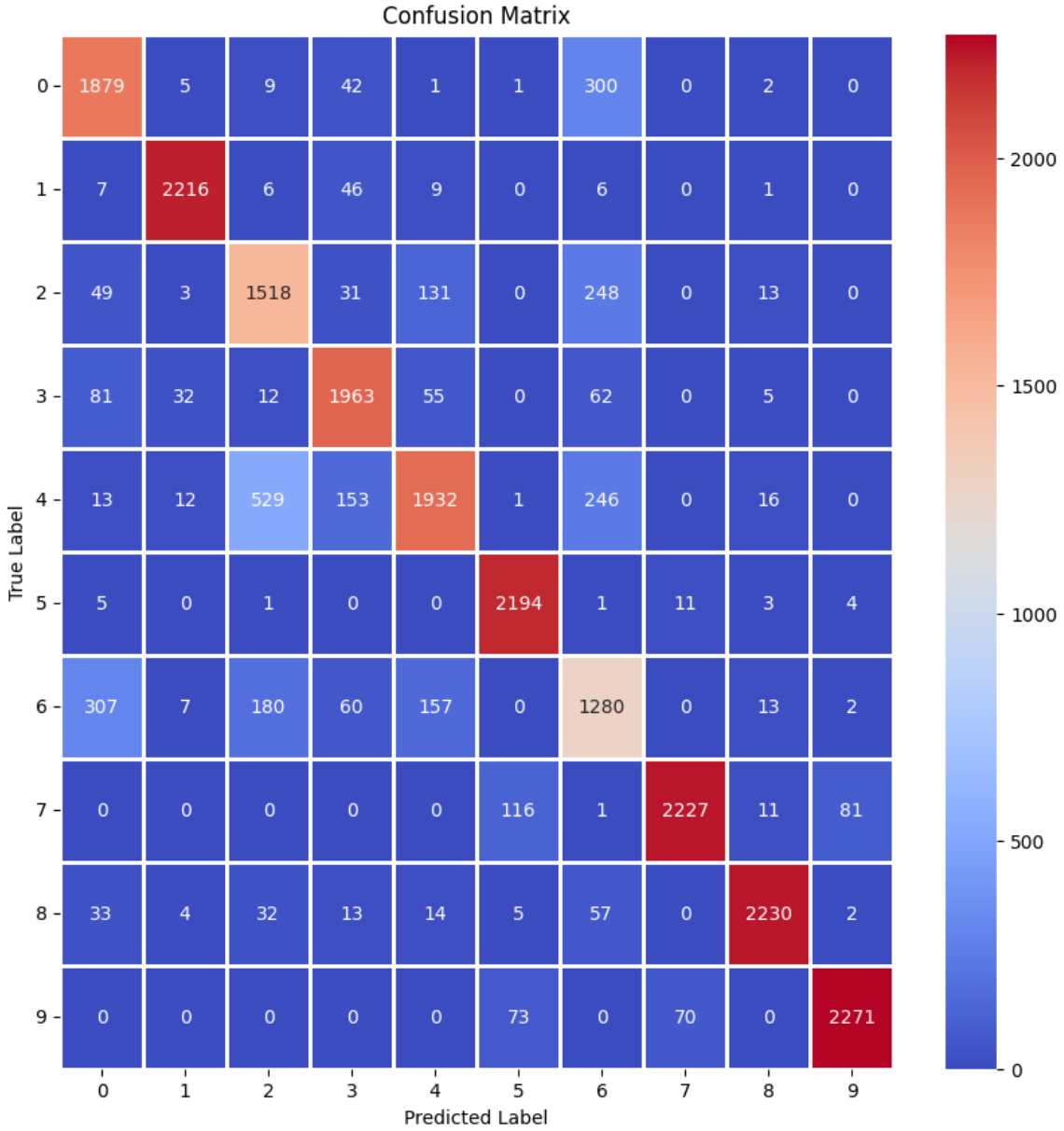
```
722/722 [==============================] - 1s 1ms/step
y_pred (23100,) (23100,)
--- First configuration ---
```



```
Accuracy:        0.896
Precision:       0.896
Recall:  0.899
F1:      0.895
Epoch 1/10
46/46 [==============================] - 4s 82ms/step - loss: 1.1732 - a
ccuracy: 0.5817
Epoch 2/10
46/46 [==============================] - 4s 84ms/step - loss: 0.5384 - a
ccuracy: 0.8003
Epoch 3/10
46/46 [==============================] - 4s 82ms/step - loss: 0.4247 - a
ccuracy: 0.8430
Epoch 4/10
46/46 [==============================] - 4s 83ms/step - loss: 0.3745 - a
ccuracy: 0.8598
Epoch 5/10
46/46 [==============================] - 4s 83ms/step - loss: 0.3379 - a
ccuracy: 0.8722
Epoch 6/10
46/46 [==============================] - 4s 83ms/step - loss: 0.3103 - a
ccuracy: 0.8816
Epoch 7/10
46/46 [==============================] - 4s 82ms/step - loss: 0.2900 - a
ccuracy: 0.8902
Epoch 8/10
46/46 [==============================] - 4s 82ms/step - loss: 0.2797 - a
ccuracy: 0.8931
Epoch 9/10
46/46 [==============================] - 4s 82ms/step - loss: 0.2586 - a
ccuracy: 0.9004
Epoch 10/10
46/46 [==============================] - 4s 83ms/step - loss: 0.2383 - a
ccuracy: 0.9088
722/722 [==============================] - 2s 3ms/step
y_pred (23100,) (23100,)
--- Second configuration ---
```


Confusion Matrix
True Label
Predicted Label
0 1541 6 11 13 0 0 64 0 3 0
1 2 2225 3 4 2 0 0 0 0 0
2 27 1 1568 8 48 0 70 0 2 0
3 70 31 14 2099 44 0 63 0 2 0
4 13 8 360 100 1995 0 126 0 7 0
5 1 0 1 0 0 2319 1 6 1 7
6 694 3 313 76 201 0 1856 0 20 0
7 0 0 0 0 0 50 0 2267 1 95
8 26 5 17 8 9 6 21 1 2258 1
9 0 0 0 0 0 15 0 34 0 2257
0 1 2 3 4 5 6 7 8 9
2000
1500
1000
500
0

```
Accuracy:        0.882
Precision:       0.882
Recall:  0.897
F1:      0.883
Epoch 1/10
46/46 [==============================] - 3s 51ms/step - loss: 3.3559 - a
ccuracy: 0.3277
Epoch 2/10
46/46 [==============================] - 2s 48ms/step - loss: 1.8152 - a
ccuracy: 0.6152
Epoch 3/10
46/46 [==============================] - 2s 49ms/step - loss: 0.6909 - a
ccuracy: 0.7271
Epoch 4/10
46/46 [==============================] - 2s 52ms/step - loss: 0.5974 - a
ccuracy: 0.7729
Epoch 5/10
46/46 [==============================] - 2s 48ms/step - loss: 0.5385 - a
ccuracy: 0.7990
Epoch 6/10
46/46 [==============================] - 2s 48ms/step - loss: 0.5960 - a
ccuracy: 0.8046
Epoch 7/10
46/46 [==============================] - 2s 47ms/step - loss: 0.4246 - a
ccuracy: 0.8409
Epoch 8/10
46/46 [==============================] - 2s 50ms/step - loss: 0.4281 - a
ccuracy: 0.8417
Epoch 9/10
46/46 [==============================] - 2s 47ms/step - loss: 0.4218 - a
ccuracy: 0.8404
Epoch 10/10
46/46 [==============================] - 2s 51ms/step - loss: 0.3909 - a
ccuracy: 0.8532
722/722 [==============================] - 2s 3ms/step
y_pred (23100,) (23100,)
--- Third configuration ---
```


Confusion Matrix
True Label
Predicted Label
0 1879 5 9 42 1 1 300 0 2 0
1 7 2216 6 46 9 0 6 0 1 0
2 49 3 1518 31 131 0 248 0 13 0
3 81 32 12 1963 55 0 62 0 5 0
4 13 12 529 153 1932 1 246 0 16 0
5 5 0 1 0 0 2194 1 11 3 4
6 307 7 180 60 157 0 1280 0 13 2
7 0 0 0 0 0 116 1 2227 11 81
8 33 4 32 13 14 5 57 0 2230 2
9 0 0 0 0 0 73 0 70 0 2271
0 1 2 3 4 5 6 7 8 9
2000
1500
1000
500
0

```
Accuracy:        0.853
Precision:       0.852
Recall:  0.854
F1:      0.851
Epoch 1/150
46/46 [==============================] - 1s 19ms/step - loss: 1.3500 - a
ccuracy: 0.6049
Epoch 2/150
46/46 [==============================] - 1s 19ms/step - loss: 0.5731 - a
ccuracy: 0.7907
Epoch 3/150
46/46 [==============================] - 1s 20ms/step - loss: 0.4477 - a
ccuracy: 0.8350
Epoch 4/150
46/46 [==============================] - 1s 20ms/step - loss: 1.0940 - a
ccuracy: 0.8072
Epoch 5/150
46/46 [==============================] - 1s 20ms/step - loss: 0.3662 - a
ccuracy: 0.8673
Epoch 6/150
46/46 [==============================] - 1s 20ms/step - loss: 0.3815 - a
ccuracy: 0.8576
Epoch 7/150
46/46 [==============================] - 1s 21ms/step - loss: 0.3662 - a
ccuracy: 0.8655
Epoch 8/150
46/46 [==============================] - 1s 21ms/step - loss: 0.3459 - a
ccuracy: 0.8725
Epoch 9/150
46/46 [==============================] - 1s 20ms/step - loss: 0.3337 - a
ccuracy: 0.8748
Epoch 10/150
46/46 [==============================] - 1s 22ms/step - loss: 0.3344 - a
ccuracy: 0.8761
Epoch 11/150
46/46 [==============================] - 1s 21ms/step - loss: 0.3088 - a
ccuracy: 0.8826
Epoch 12/150
46/46 [==============================] - 1s 21ms/step - loss: 0.3060 - a
ccuracy: 0.8851
Epoch 13/150
46/46 [==============================] - 1s 22ms/step - loss: 0.3001 - a
ccuracy: 0.8894
Epoch 14/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2811 - a
ccuracy: 0.8946
Epoch 15/150
46/46 [==============================] - 1s 22ms/step - loss: 0.2823 - a
ccuracy: 0.8929
Epoch 16/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2739 - a
ccuracy: 0.8957
Epoch 17/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2741 - a
ccuracy: 0.8972
Epoch 18/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2612 - a
ccuracy: 0.9007
Epoch 19/150
46/46 [==============================] - 1s 20ms/step - loss: 0.2702 - a
```

```
ccuracy: 0.8991
Epoch 20/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2571 - a
ccuracy: 0.9023
Epoch 21/150
46/46 [==============================] - 1s 20ms/step - loss: 0.2555 - a
ccuracy: 0.9039
Epoch 22/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2590 - a
ccuracy: 0.9021
Epoch 23/150
46/46 [==============================] - 1s 22ms/step - loss: 0.2425 - a
ccuracy: 0.9091
Epoch 24/150
46/46 [==============================] - 1s 22ms/step - loss: 0.2430 - a
ccuracy: 0.9067
Epoch 25/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2361 - a
ccuracy: 0.9096
Epoch 26/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2473 - a
ccuracy: 0.9065
Epoch 27/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2292 - a
ccuracy: 0.9113
Epoch 28/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2340 - a
ccuracy: 0.9114
Epoch 29/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2333 - a
ccuracy: 0.9105
Epoch 30/150
46/46 [==============================] - 1s 20ms/step - loss: 0.2225 - a
ccuracy: 0.9173
Epoch 31/150
46/46 [==============================] - 1s 20ms/step - loss: 0.2303 - a
ccuracy: 0.9114
Epoch 32/150
46/46 [==============================] - 1s 20ms/step - loss: 0.2203 - a
ccuracy: 0.9162
Epoch 33/150
46/46 [==============================] - 1s 20ms/step - loss: 0.2198 - a
ccuracy: 0.9163
Epoch 34/150
46/46 [==============================] - 1s 19ms/step - loss: 0.2248 - a
ccuracy: 0.9147
Epoch 35/150
46/46 [==============================] - 1s 20ms/step - loss: 0.2202 - a
ccuracy: 0.9162
Epoch 36/150
46/46 [==============================] - 1s 20ms/step - loss: 0.2164 - a
ccuracy: 0.9170
Epoch 37/150
46/46 [==============================] - 1s 20ms/step - loss: 0.2106 - a
ccuracy: 0.9191
Epoch 38/150
46/46 [==============================] - 1s 22ms/step - loss: 0.2082 - a
ccuracy: 0.9200
Epoch 39/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2196 - a
```

```
ccuracy: 0.9177
Epoch 40/150
46/46 [==============================] - 1s 20ms/step - loss: 0.2055 - a
ccuracy: 0.9216
Epoch 41/150
46/46 [==============================] - 1s 20ms/step - loss: 0.2045 - a
ccuracy: 0.9209
Epoch 42/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2159 - a
ccuracy: 0.9194
Epoch 43/150
46/46 [==============================] - 1s 22ms/step - loss: 0.2009 - a
ccuracy: 0.9235
Epoch 44/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2118 - a
ccuracy: 0.9211
Epoch 45/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2020 - a
ccuracy: 0.9223
Epoch 46/150
46/46 [==============================] - 1s 21ms/step - loss: 0.1927 - a
ccuracy: 0.9267
Epoch 47/150
46/46 [==============================] - 1s 21ms/step - loss: 0.2228 - a
ccuracy: 0.9178
Epoch 48/150
46/46 [==============================] - 1s 21ms/step - loss: 0.1888 - a
ccuracy: 0.9271
Epoch 49/150
46/46 [==============================] - 1s 22ms/step - loss: 0.1947 - a
ccuracy: 0.9256
Epoch 50/150
46/46 [==============================] - 1s 22ms/step - loss: 0.1965 - a
ccuracy: 0.9248
Epoch 51/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1965 - a
ccuracy: 0.9260
Epoch 52/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1885 - a
ccuracy: 0.9281
Epoch 53/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1987 - a
ccuracy: 0.9251
Epoch 54/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1866 - a
ccuracy: 0.9299
Epoch 55/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1999 - a
ccuracy: 0.9258
Epoch 56/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1863 - a
ccuracy: 0.9282
Epoch 57/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1886 - a
ccuracy: 0.9285
Epoch 58/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1808 - a
ccuracy: 0.9314
Epoch 59/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1888 - a
```

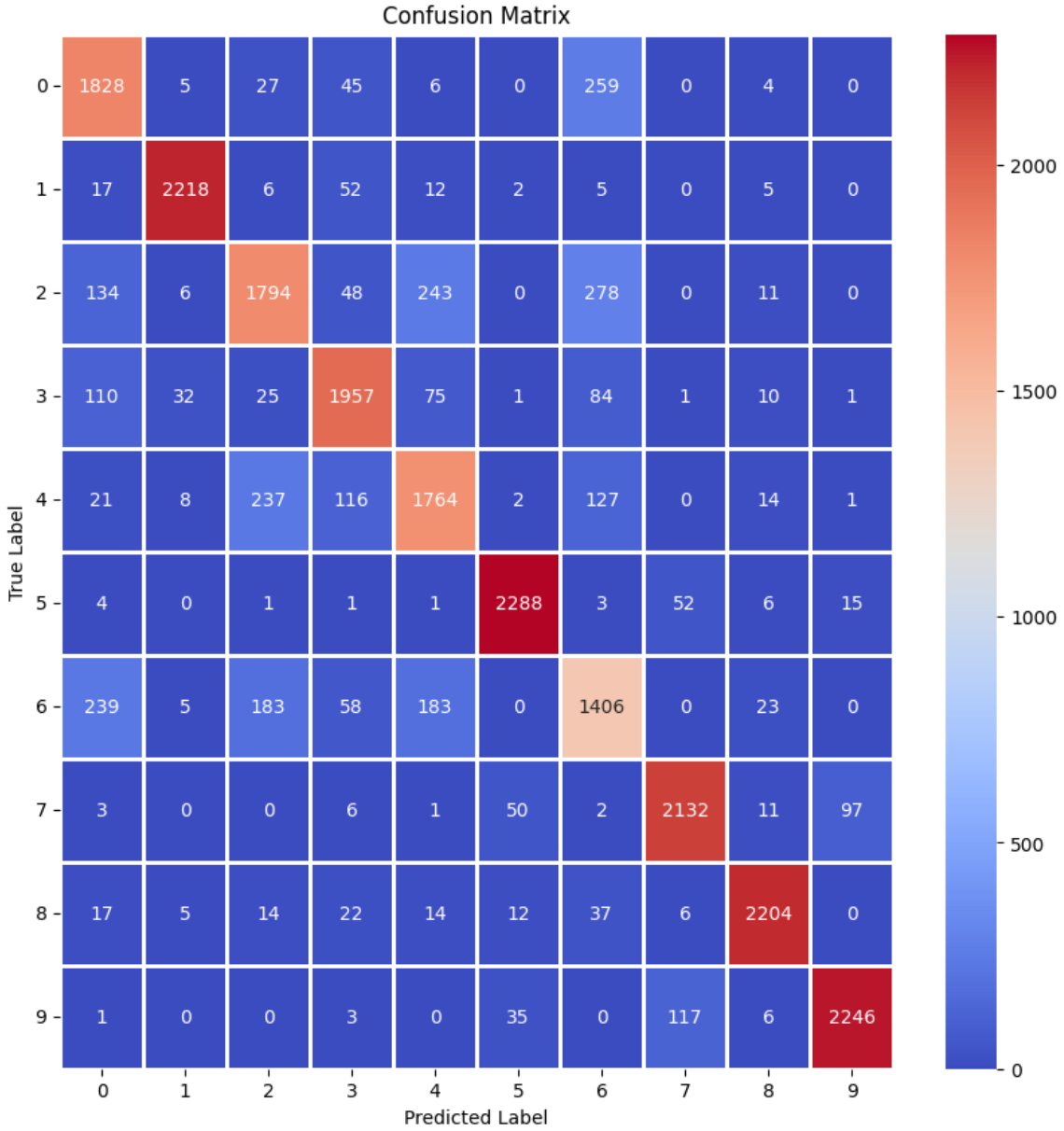
```
ccuracy: 0.9288
Epoch 60/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1794 - a
ccuracy: 0.9314
Epoch 61/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1858 - a
ccuracy: 0.9287
Epoch 62/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1815 - a
ccuracy: 0.9308
Epoch 63/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1840 - a
ccuracy: 0.9311
Epoch 64/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1856 - a
ccuracy: 0.9307
Epoch 65/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1796 - a
ccuracy: 0.9318
Epoch 66/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1865 - a
ccuracy: 0.9306
Epoch 67/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1771 - a
ccuracy: 0.9339
Epoch 68/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1818 - a
ccuracy: 0.9313
Epoch 69/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1733 - a
ccuracy: 0.9340
Epoch 70/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1754 - a
ccuracy: 0.9344
Epoch 71/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1799 - a
ccuracy: 0.9336
Epoch 72/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1684 - a
ccuracy: 0.9362
Epoch 73/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1767 - a
ccuracy: 0.9340
Epoch 74/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1629 - a
ccuracy: 0.9378
Epoch 75/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1784 - a
ccuracy: 0.9334
Epoch 76/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1707 - a
ccuracy: 0.9363
Epoch 77/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1685 - a
ccuracy: 0.9354
Epoch 78/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1681 - a
ccuracy: 0.9355
Epoch 79/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1757 - a
```

```
ccuracy: 0.9339
Epoch 80/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1654 - a
ccuracy: 0.9370
Epoch 81/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1691 - a
ccuracy: 0.9352
Epoch 82/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1714 - a
ccuracy: 0.9354
Epoch 83/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1713 - a
ccuracy: 0.9366
Epoch 84/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1589 - a
ccuracy: 0.9386
Epoch 85/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1646 - a
ccuracy: 0.9377
Epoch 86/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1692 - a
ccuracy: 0.9360
Epoch 87/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1743 - a
ccuracy: 0.9349
Epoch 88/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1676 - a
ccuracy: 0.9363
Epoch 89/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1601 - a
ccuracy: 0.9390
Epoch 90/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1641 - a
ccuracy: 0.9377
Epoch 91/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1665 - a
ccuracy: 0.9372
Epoch 92/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1613 - a
ccuracy: 0.9381
Epoch 93/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1668 - a
ccuracy: 0.9378
Epoch 94/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1546 - a
ccuracy: 0.9409
Epoch 95/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1626 - a
ccuracy: 0.9398
Epoch 96/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1654 - a
ccuracy: 0.9374
Epoch 97/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1607 - a
ccuracy: 0.9398
Epoch 98/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1547 - a
ccuracy: 0.9419
Epoch 99/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1571 - a
```

```
ccuracy: 0.9413
Epoch 100/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1614 - a
ccuracy: 0.9389
Epoch 101/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1614 - a
ccuracy: 0.9380
Epoch 102/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1588 - a
ccuracy: 0.9411
Epoch 103/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1549 - a
ccuracy: 0.9411
Epoch 104/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1618 - a
ccuracy: 0.9410
Epoch 105/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1564 - a
ccuracy: 0.9400
Epoch 106/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1512 - a
ccuracy: 0.9422
Epoch 107/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1612 - a
ccuracy: 0.9402
Epoch 108/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1654 - a
ccuracy: 0.9384
Epoch 109/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1498 - a
ccuracy: 0.9425
Epoch 110/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1491 - a
ccuracy: 0.9435
Epoch 111/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1545 - a
ccuracy: 0.9409
Epoch 112/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1581 - a
ccuracy: 0.9422
Epoch 113/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1585 - a
ccuracy: 0.9416
Epoch 114/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1456 - a
ccuracy: 0.9446
Epoch 115/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1537 - a
ccuracy: 0.9437
Epoch 116/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1525 - a
ccuracy: 0.9422
Epoch 117/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1513 - a
ccuracy: 0.9447
Epoch 118/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1494 - a
ccuracy: 0.9448
Epoch 119/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1509 - a
```

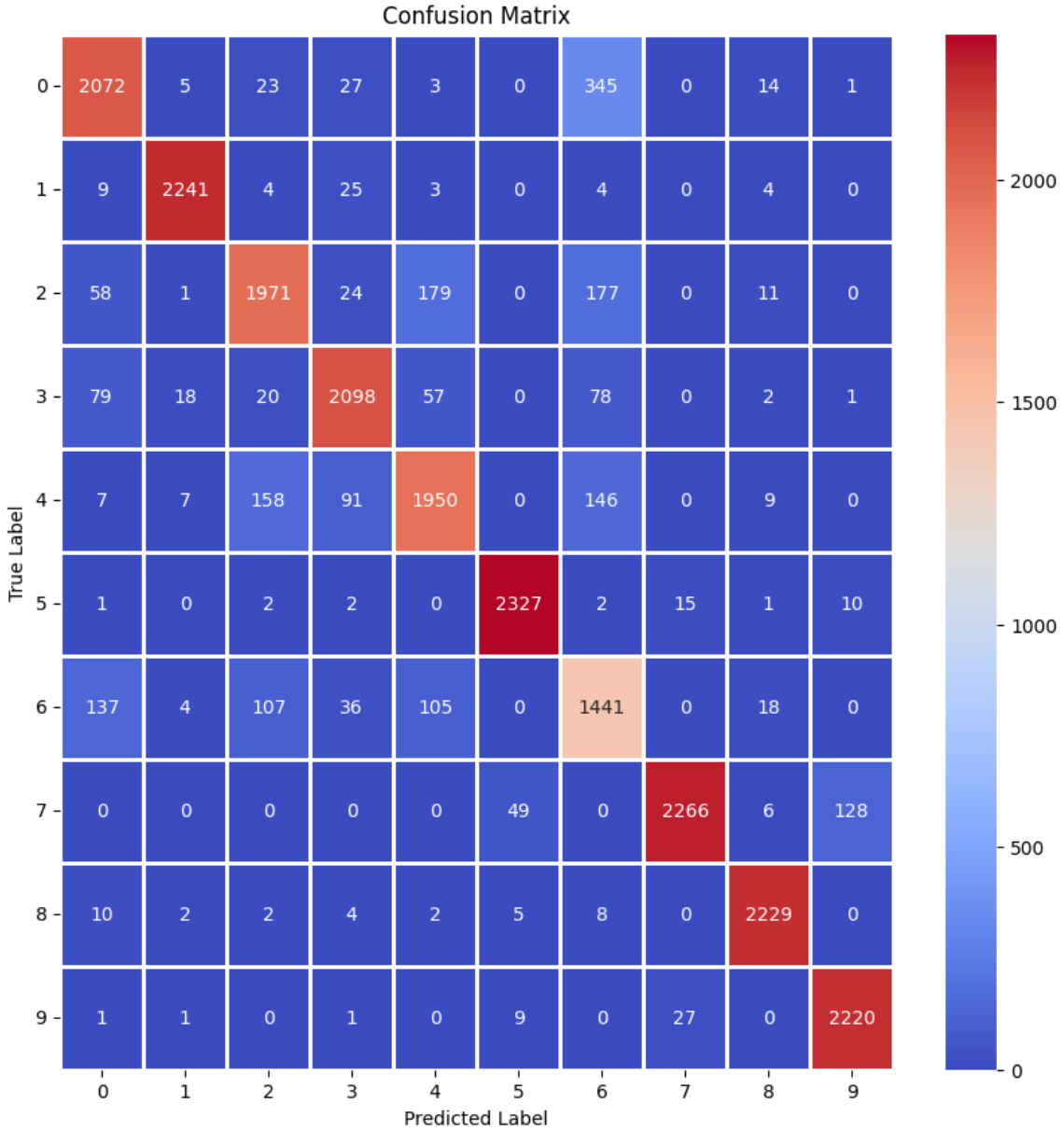
```
ccuracy: 0.9441
Epoch 120/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1574 - a
ccuracy: 0.9429
Epoch 121/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1511 - a
ccuracy: 0.9431
Epoch 122/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1442 - a
ccuracy: 0.9453
Epoch 123/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1427 - a
ccuracy: 0.9466
Epoch 124/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1528 - a
ccuracy: 0.9433
Epoch 125/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1497 - a
ccuracy: 0.9446
Epoch 126/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1493 - a
ccuracy: 0.9444
Epoch 127/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1455 - a
ccuracy: 0.9448
Epoch 128/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1462 - a
ccuracy: 0.9449
Epoch 129/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1490 - a
ccuracy: 0.9444
Epoch 130/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1481 - a
ccuracy: 0.9437
Epoch 131/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1468 - a
ccuracy: 0.9456
Epoch 132/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1497 - a
ccuracy: 0.9450
Epoch 133/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1491 - a
ccuracy: 0.9453
Epoch 134/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1534 - a
ccuracy: 0.9449
Epoch 135/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1517 - a
ccuracy: 0.9440
Epoch 136/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1321 - a
ccuracy: 0.9504
Epoch 137/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1424 - a
ccuracy: 0.9466
Epoch 138/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1465 - a
ccuracy: 0.9445
Epoch 139/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1458 - a
```

```
ccuracy: 0.9456
Epoch 140/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1527 - a
ccuracy: 0.9444
Epoch 141/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1482 - a
ccuracy: 0.9442
Epoch 142/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1441 - a
ccuracy: 0.9482
Epoch 143/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1352 - a
ccuracy: 0.9497
Epoch 144/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1479 - a
ccuracy: 0.9454
Epoch 145/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1388 - a
ccuracy: 0.9478
Epoch 146/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1449 - a
ccuracy: 0.9474
Epoch 147/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1450 - a
ccuracy: 0.9469
Epoch 148/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1368 - a
ccuracy: 0.9487
Epoch 149/150
46/46 [==============================] - 1s 19ms/step - loss: 0.1460 - a
ccuracy: 0.9454
Epoch 150/150
46/46 [==============================] - 1s 20ms/step - loss: 0.1394 - a
ccuracy: 0.9488
722/722 [==============================] - 1s 1ms/step
y_pred (23100,) (23100,)
--- Fourth configuration ---
```


Confusion Matrix
True Label
Predicted Label
0 1828 5 27 45 6 0 259 0 4 0
1 17 2218 6 52 12 2 5 0 5 0
2 134 6 1794 48 243 0 278 0 11 0
3 110 32 25 1957 75 1 84 1 10 1
4 21 8 237 116 1764 2 127 0 14 1
5 4 0 1 1 1 2288 3 52 6 15
6 239 5 183 58 183 0 1406 0 23 0
7 3 0 0 6 1 50 2 2132 11 97
8 17 5 14 22 14 12 37 6 2204 0
9 1 0 0 3 0 35 0 117 6 2246
0 1 2 3 4 5 6 7 8 9
2000
1500
1000
500
0

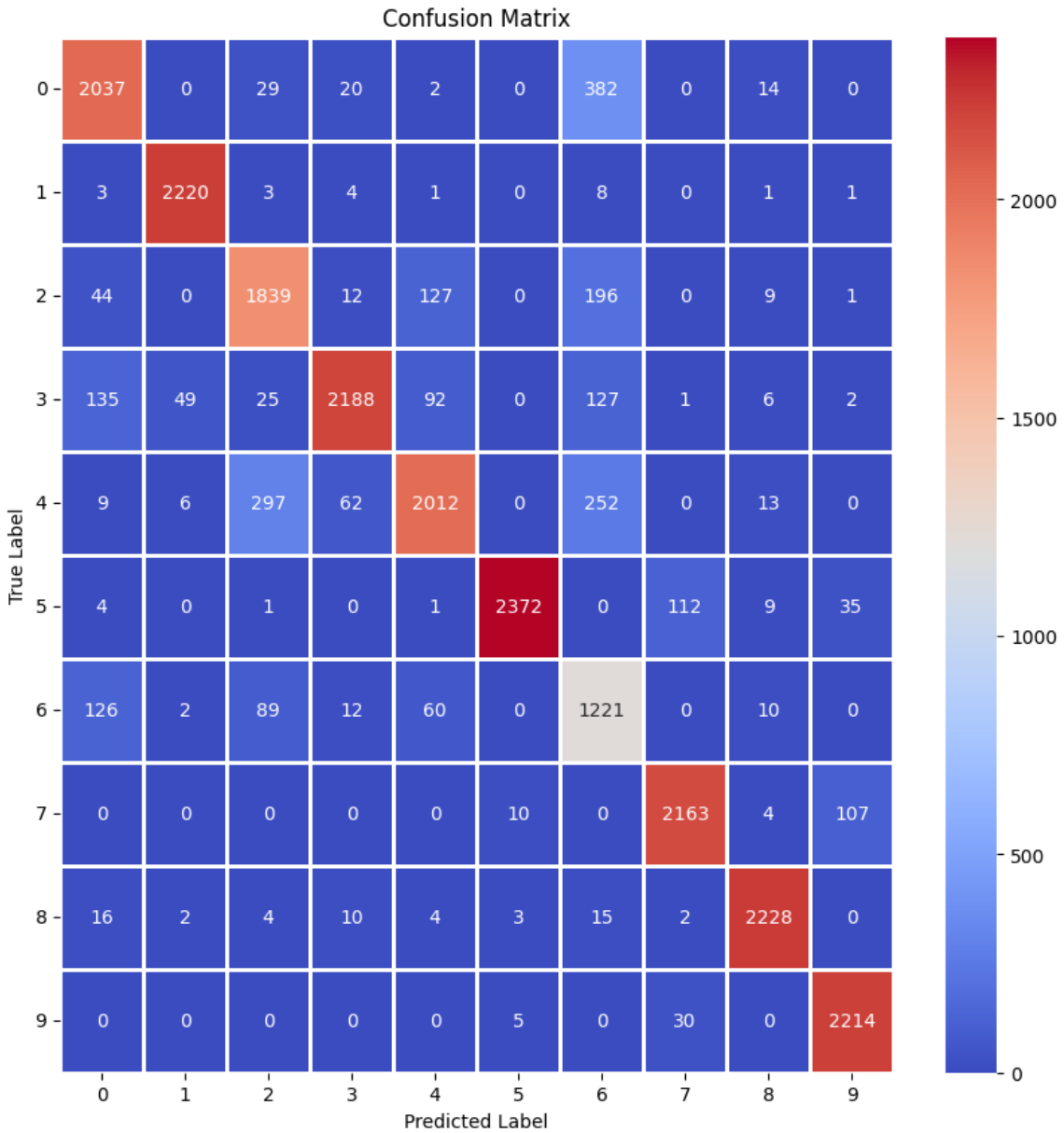
```
Accuracy:        0.859
Precision:       0.858
Recall:  0.857
F1:      0.857
Epoch 1/20
46/46 [==============================] - 4s 83ms/step - loss: 1.2536 - a
ccuracy: 0.5560
Epoch 2/20
46/46 [==============================] - 4s 85ms/step - loss: 0.5754 - a
ccuracy: 0.7853
Epoch 3/20
46/46 [==============================] - 4s 83ms/step - loss: 0.4624 - a
ccuracy: 0.8262
Epoch 4/20
46/46 [==============================] - 4s 84ms/step - loss: 0.4000 - a
ccuracy: 0.8500
Epoch 5/20
46/46 [==============================] - 4s 85ms/step - loss: 0.3528 - a
ccuracy: 0.8675
Epoch 6/20
46/46 [==============================] - 4s 85ms/step - loss: 0.3265 - a
ccuracy: 0.8771
Epoch 7/20
46/46 [==============================] - 4s 85ms/step - loss: 0.3019 - a
ccuracy: 0.8855
Epoch 8/20
46/46 [==============================] - 4s 83ms/step - loss: 0.2875 - a
ccuracy: 0.8911
Epoch 9/20
46/46 [==============================] - 4s 82ms/step - loss: 0.2661 - a
ccuracy: 0.8979
Epoch 10/20
46/46 [==============================] - 4s 83ms/step - loss: 0.2562 - a
ccuracy: 0.9015
Epoch 11/20
46/46 [==============================] - 4s 82ms/step - loss: 0.2413 - a
ccuracy: 0.9070
Epoch 12/20
46/46 [==============================] - 4s 82ms/step - loss: 0.2337 - a
ccuracy: 0.9094
Epoch 13/20
46/46 [==============================] - 4s 83ms/step - loss: 0.2196 - a
ccuracy: 0.9161
Epoch 14/20
46/46 [==============================] - 4s 83ms/step - loss: 0.2109 - a
ccuracy: 0.9185
Epoch 15/20
46/46 [==============================] - 4s 83ms/step - loss: 0.1993 - a
ccuracy: 0.9230
Epoch 16/20
46/46 [==============================] - 4s 83ms/step - loss: 0.1912 - a
ccuracy: 0.9249
Epoch 17/20
46/46 [==============================] - 4s 83ms/step - loss: 0.1823 - a
ccuracy: 0.9298
Epoch 18/20
46/46 [==============================] - 4s 82ms/step - loss: 0.1765 - a
ccuracy: 0.9316
Epoch 19/20
46/46 [==============================] - 4s 83ms/step - loss: 0.1681 - a
```

```
ccuracy: 0.9354
Epoch 20/20
46/46 [==============================] - 4s 82ms/step - loss: 0.1620 - a
ccuracy: 0.9373
722/722 [==============================] - 2s 3ms/step
y_pred (23100,) (23100,)
--- Fifth configuration ---
```



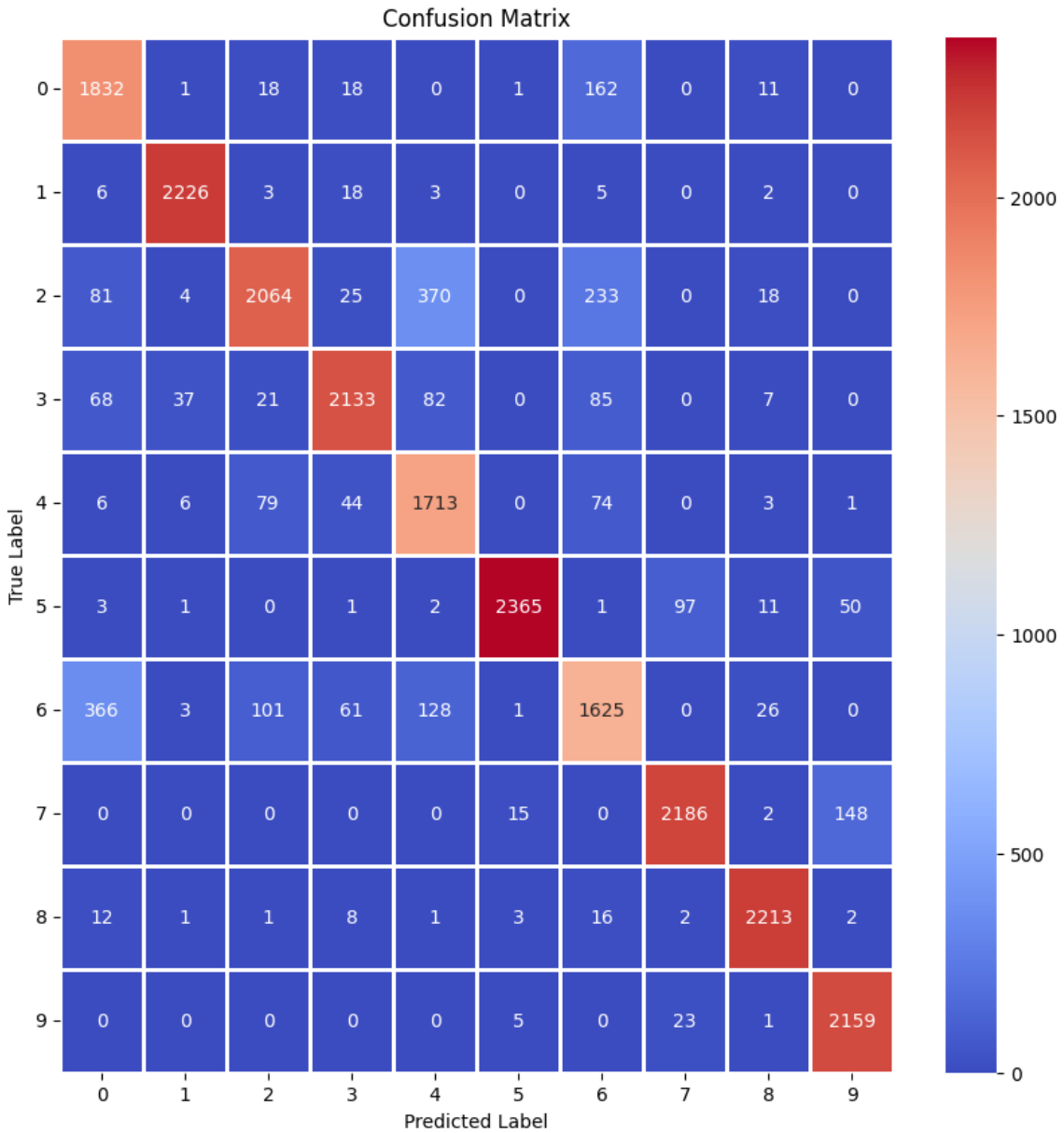
```
Accuracy:        0.901
Precision:       0.900
Recall:  0.900
F1:      0.899
Epoch 1/20
46/46 [==============================] - 2s 47ms/step - loss: 1.4425 - a
ccuracy: 0.4816
Epoch 2/20
46/46 [==============================] - 2s 47ms/step - loss: 0.6999 - a
ccuracy: 0.7374
Epoch 3/20
46/46 [==============================] - 2s 52ms/step - loss: 0.5369 - a
ccuracy: 0.7993
Epoch 4/20
46/46 [==============================] - 2s 52ms/step - loss: 0.4606 - a
ccuracy: 0.8294
Epoch 5/20
46/46 [==============================] - 2s 48ms/step - loss: 0.4086 - a
ccuracy: 0.8478
Epoch 6/20
46/46 [==============================] - 2s 49ms/step - loss: 0.3797 - a
ccuracy: 0.8591
Epoch 7/20
46/46 [==============================] - 2s 48ms/step - loss: 0.3553 - a
ccuracy: 0.8689
Epoch 8/20
46/46 [==============================] - 2s 48ms/step - loss: 0.3394 - a
ccuracy: 0.8736
Epoch 9/20
46/46 [==============================] - 2s 48ms/step - loss: 0.3217 - a
ccuracy: 0.8791
Epoch 10/20
46/46 [==============================] - 2s 49ms/step - loss: 0.3079 - a
ccuracy: 0.8856
Epoch 11/20
46/46 [==============================] - 2s 48ms/step - loss: 0.2984 - a
ccuracy: 0.8871
Epoch 12/20
46/46 [==============================] - 2s 48ms/step - loss: 0.2877 - a
ccuracy: 0.8911
Epoch 13/20
46/46 [==============================] - 2s 48ms/step - loss: 0.2789 - a
ccuracy: 0.8957
Epoch 14/20
46/46 [==============================] - 2s 48ms/step - loss: 0.2707 - a
ccuracy: 0.8962
Epoch 15/20
46/46 [==============================] - 2s 49ms/step - loss: 0.2623 - a
ccuracy: 0.9006
Epoch 16/20
46/46 [==============================] - 2s 49ms/step - loss: 0.2535 - a
ccuracy: 0.9035
Epoch 17/20
46/46 [==============================] - 2s 49ms/step - loss: 0.2484 - a
ccuracy: 0.9039
Epoch 18/20
46/46 [==============================] - 2s 48ms/step - loss: 0.2413 - a
ccuracy: 0.9091
Epoch 19/20
46/46 [==============================] - 2s 48ms/step - loss: 0.2353 - a
```

```
ccuracy: 0.9084
Epoch 20/20
46/46 [==============================] - 2s 48ms/step - loss: 0.2368 - a
ccuracy: 0.9088
722/722 [==============================] - 2s 2ms/step
y_pred (23100,) (23100,)
--- Sixth configuration ---
```



```
Accuracy:        0.887
Precision:       0.885
Recall:  0.887
F1:      0.883
Epoch 1/20
46/46 [==============================] - 1s 19ms/step - loss: 1.0527 - a
ccuracy: 0.6517
Epoch 2/20
46/46 [==============================] - 1s 21ms/step - loss: 0.4988 - a
ccuracy: 0.8174
Epoch 3/20
46/46 [==============================] - 1s 21ms/step - loss: 0.4180 - a
ccuracy: 0.8484
Epoch 4/20
46/46 [==============================] - 1s 20ms/step - loss: 0.3755 - a
ccuracy: 0.8645
Epoch 5/20
46/46 [==============================] - 1s 20ms/step - loss: 0.3479 - a
ccuracy: 0.8730
Epoch 6/20
46/46 [==============================] - 1s 21ms/step - loss: 0.3213 - a
ccuracy: 0.8826
Epoch 7/20
46/46 [==============================] - 1s 20ms/step - loss: 0.3092 - a
ccuracy: 0.8855
Epoch 8/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2931 - a
ccuracy: 0.8934
Epoch 9/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2799 - a
ccuracy: 0.8971
Epoch 10/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2753 - a
ccuracy: 0.8982
Epoch 11/20
46/46 [==============================] - 1s 21ms/step - loss: 0.2630 - a
ccuracy: 0.9013
Epoch 12/20
46/46 [==============================] - 1s 21ms/step - loss: 0.2542 - a
ccuracy: 0.9059
Epoch 13/20
46/46 [==============================] - 1s 21ms/step - loss: 0.2439 - a
ccuracy: 0.9091
Epoch 14/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2402 - a
ccuracy: 0.9092
Epoch 15/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2277 - a
ccuracy: 0.9155
Epoch 16/20
46/46 [==============================] - 1s 20ms/step - loss: 0.2246 - a
ccuracy: 0.9156
Epoch 17/20
46/46 [==============================] - 1s 21ms/step - loss: 0.2177 - a
ccuracy: 0.9183
Epoch 18/20
46/46 [==============================] - 1s 23ms/step - loss: 0.2139 - a
ccuracy: 0.9204
Epoch 19/20
46/46 [==============================] - 1s 22ms/step - loss: 0.2091 - a
```

```
ccuracy: 0.9217
Epoch 20/20
46/46 [==============================] - 1s 21ms/step - loss: 0.2003 - a
ccuracy: 0.9246
722/722 [==============================] - 1s 1ms/step
y_pred (23100,) (23100,)
--- Seventh configuration ---
```




```
Accuracy:       0.888
Precision:      0.887
Recall:  0.892
F1:      0.887
```

TODO 3.3.3 Prepare 7 (**seven**) configurations adding **at least** two new NN models and perform tests. Try to outperform the results obtained with the sample code.

Present the results in the form of a table. A table can be entered as a markdown: see the table generator. As an alternative, you may load data to Pandas DataFrame and display using df.head().

Configuration	Model name	Hiperparameters	Accuracy	Precision	Recall	F1
First Configuration	model_0	epochs=20 learning_rate=0.01 random_seed=42	0.896	0.896	0.899	0.895
Second Configuration	model_1	epochs=10 learning_rate=0.01 random_seed=42	0.882	0.882	0.897	0.883
Third Configuration	model_2	epochs=10 learning_rate=0.03 random_seed=15	0.853	0.852	0.854	0.851
Fourth Configuration	model_0	epochs=150 learning_rate=0.04 random_seed=42	0.859	0.858	0.857	0.857
Fifth Configuration	model_1	epochs=5 learning_rate=0.06 random_seed=42	0.901	0.900	0.900	0.899
Sixth Configuration	model_2	epochs=20 learning_rate=0.04 random_seed=10	0.887	0.885	0.887	0.883
Seventh Configuration	model_0	epochs=30 learning_rate=0.02 random_seed=10	0.888	0.887	0.892	0.887

3.4 Cross validation

In [80]:
```
import numpy as np
from tensorflow.keras.datasets import fashion_mnist

(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()
```

We define a code that uses StratifiedKFold class to perform k-fold cross validation. Stratified - means preserving proportions of strata, i.e. class labels.

The cross validation will be performed only on the training set. Then the model will be tested on the training set

Cross validation

In [81]:
```
from sklearn.model_selection import StratifiedKFold, train_test_split

n_splits = 5
random_state =  123
config = configurations['First configuration']

cv_results = {}

skf = StratifiedKFold(n_splits=n_splits, shuffle=True,random_state=random
for fold_number, (train_index, test_index) in enumerate(skf.split(X_train
  X_train_fold = X_train[train_index]
  y_train_fold = y_train[train_index]
  X_test_fold = X_train[test_index]
```

```
  y_test_fold = y_train[test_index]

  print(f'============= Fold#{fold_number+1} =============')
  results = perform_test(X_train_fold,y_train_fold,X_test_fold,y_test_fol

  for k in results:
    if k=='model':
      continue
    rlist=cv_results.get(k,[])
    rlist.append(results[k])
    cv_results[k]=rlist
```

```
============= Fold#1 =============
Epoch 1/20
47/47 [==============================] - 2s 20ms/step - loss: 41.7819 -
accuracy: 0.5910
Epoch 2/20
47/47 [==============================] - 1s 19ms/step - loss: 0.9978 - a
ccuracy: 0.7235
Epoch 3/20
47/47 [==============================] - 1s 20ms/step - loss: 0.6756 - a
ccuracy: 0.7792
Epoch 4/20
47/47 [==============================] - 1s 20ms/step - loss: 0.5189 - a
ccuracy: 0.8204
Epoch 5/20
47/47 [==============================] - 1s 19ms/step - loss: 0.4094 - a
ccuracy: 0.8458
Epoch 6/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3755 - a
ccuracy: 0.8569
Epoch 7/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3461 - a
ccuracy: 0.8669
Epoch 8/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3241 - a
ccuracy: 0.8740
Epoch 9/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3130 - a
ccuracy: 0.8787
Epoch 10/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3068 - a
ccuracy: 0.8827
Epoch 11/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2948 - a
ccuracy: 0.8862
Epoch 12/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2878 - a
ccuracy: 0.8889
Epoch 13/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2743 - a
ccuracy: 0.8920
Epoch 14/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2799 - a
ccuracy: 0.8917
Epoch 15/20
47/47 [==============================] - 1s 20ms/step - loss: 0.2640 - a
ccuracy: 0.8956
Epoch 16/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2557 - a
ccuracy: 0.8999
Epoch 17/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2507 - a
ccuracy: 0.9029
Epoch 18/20
47/47 [==============================] - 1s 20ms/step - loss: 0.2551 - a
ccuracy: 0.9005
Epoch 19/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2436 - a
ccuracy: 0.9043
Epoch 20/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2432 - a
```

```
ccuracy: 0.9045
375/375 [==============================] - 0s 936us/step
y_pred (12000,) (12000,)
============= Fold#2 =============
Epoch 1/20
47/47 [==============================] - 1s 18ms/step - loss: 32.8539 -
accuracy: 0.5850
Epoch 2/20
47/47 [==============================] - 1s 19ms/step - loss: 1.0817 - a
ccuracy: 0.7584
Epoch 3/20
47/47 [==============================] - 1s 19ms/step - loss: 0.6358 - a
ccuracy: 0.8144
Epoch 4/20
47/47 [==============================] - 1s 20ms/step - loss: 0.4413 - a
ccuracy: 0.8425
Epoch 5/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3915 - a
ccuracy: 0.8556
Epoch 6/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3577 - a
ccuracy: 0.8661
Epoch 7/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3377 - a
ccuracy: 0.8756
Epoch 8/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3296 - a
ccuracy: 0.8805
Epoch 9/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3059 - a
ccuracy: 0.8874
Epoch 10/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2997 - a
ccuracy: 0.8884
Epoch 11/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2887 - a
ccuracy: 0.8928
Epoch 12/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2765 - a
ccuracy: 0.8966
Epoch 13/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2696 - a
ccuracy: 0.9005
Epoch 14/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2676 - a
ccuracy: 0.9018
Epoch 15/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2564 - a
ccuracy: 0.9056
Epoch 16/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2606 - a
ccuracy: 0.9057
Epoch 17/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2493 - a
ccuracy: 0.9061
Epoch 18/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2493 - a
ccuracy: 0.9071
Epoch 19/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2402 - a
```

```
ccuracy: 0.9111
Epoch 20/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2363 - a
ccuracy: 0.9112
375/375 [==============================] - 0s 896us/step
y_pred (12000,) (12000,)
============= Fold#3 =============
Epoch 1/20
47/47 [==============================] - 1s 18ms/step - loss: 40.0515 -
accuracy: 0.5212
Epoch 2/20
47/47 [==============================] - 1s 19ms/step - loss: 0.8800 - a
ccuracy: 0.7696
Epoch 3/20
47/47 [==============================] - 1s 19ms/step - loss: 0.6459 - a
ccuracy: 0.8227
Epoch 4/20
47/47 [==============================] - 1s 20ms/step - loss: 0.4869 - a
ccuracy: 0.8490
Epoch 5/20
47/47 [==============================] - 1s 18ms/step - loss: 0.3886 - a
ccuracy: 0.8593
Epoch 6/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3562 - a
ccuracy: 0.8694
Epoch 7/20
47/47 [==============================] - 1s 18ms/step - loss: 0.3382 - a
ccuracy: 0.8757
Epoch 8/20
47/47 [==============================] - 1s 18ms/step - loss: 0.3270 - a
ccuracy: 0.8795
Epoch 9/20
47/47 [==============================] - 1s 18ms/step - loss: 0.3109 - a
ccuracy: 0.8841
Epoch 10/20
47/47 [==============================] - 1s 18ms/step - loss: 0.2959 - a
ccuracy: 0.8909
Epoch 11/20
47/47 [==============================] - 1s 18ms/step - loss: 0.2982 - a
ccuracy: 0.8904
Epoch 12/20
47/47 [==============================] - 1s 18ms/step - loss: 0.2860 - a
ccuracy: 0.8927
Epoch 13/20
47/47 [==============================] - 1s 18ms/step - loss: 0.2861 - a
ccuracy: 0.8942
Epoch 14/20
47/47 [==============================] - 1s 18ms/step - loss: 0.2703 - a
ccuracy: 0.8995
Epoch 15/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2685 - a
ccuracy: 0.8995
Epoch 16/20
47/47 [==============================] - 1s 20ms/step - loss: 0.2652 - a
ccuracy: 0.9027
Epoch 17/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2595 - a
ccuracy: 0.9048
Epoch 18/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2585 - a
```

```
ccuracy: 0.9055
Epoch 19/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2502 - a
ccuracy: 0.9074
Epoch 20/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2497 - a
ccuracy: 0.9080
375/375 [==============================] - 0s 881us/step
y_pred (12000,) (12000,)
============= Fold#4 =============
Epoch 1/20
47/47 [==============================] - 1s 19ms/step - loss: 5.6216 - a
ccuracy: 0.6860
Epoch 2/20
47/47 [==============================] - 1s 19ms/step - loss: 0.5356 - a
ccuracy: 0.8134
Epoch 3/20
47/47 [==============================] - 1s 19ms/step - loss: 0.4296 - a
ccuracy: 0.8428
Epoch 4/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3815 - a
ccuracy: 0.8575
Epoch 5/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3518 - a
ccuracy: 0.8669
Epoch 6/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3282 - a
ccuracy: 0.8764
Epoch 7/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3071 - a
ccuracy: 0.8848
Epoch 8/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2959 - a
ccuracy: 0.8880
Epoch 9/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2861 - a
ccuracy: 0.8918
Epoch 10/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2804 - a
ccuracy: 0.8938
Epoch 11/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2618 - a
ccuracy: 0.9002
Epoch 12/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2595 - a
ccuracy: 0.9022
Epoch 13/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2469 - a
ccuracy: 0.9078
Epoch 14/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2570 - a
ccuracy: 0.9041
Epoch 15/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2294 - a
ccuracy: 0.9128
Epoch 16/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2283 - a
ccuracy: 0.9138
Epoch 17/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2327 - a
```

```
ccuracy: 0.9136
Epoch 18/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2202 - a
ccuracy: 0.9179
Epoch 19/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2122 - a
ccuracy: 0.9195
Epoch 20/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2195 - a
ccuracy: 0.9187
375/375 [==============================] - 0s 963us/step
y_pred (12000,) (12000,)
============= Fold#5 =============
Epoch 1/20
47/47 [==============================] - 1s 19ms/step - loss: 34.6001 -
accuracy: 0.5328
Epoch 2/20
47/47 [==============================] - 1s 19ms/step - loss: 0.9565 - a
ccuracy: 0.7075
Epoch 3/20
47/47 [==============================] - 1s 20ms/step - loss: 0.7202 - a
ccuracy: 0.7598
Epoch 4/20
47/47 [==============================] - 1s 21ms/step - loss: 0.5260 - a
ccuracy: 0.8142
Epoch 5/20
47/47 [==============================] - 1s 20ms/step - loss: 0.4349 - a
ccuracy: 0.8418
Epoch 6/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3899 - a
ccuracy: 0.8553
Epoch 7/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3634 - a
ccuracy: 0.8665
Epoch 8/20
47/47 [==============================] - 1s 21ms/step - loss: 0.3347 - a
ccuracy: 0.8753
Epoch 9/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3314 - a
ccuracy: 0.8772
Epoch 10/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3129 - a
ccuracy: 0.8800
Epoch 11/20
47/47 [==============================] - 1s 19ms/step - loss: 0.3028 - a
ccuracy: 0.8848
Epoch 12/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2967 - a
ccuracy: 0.8886
Epoch 13/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2838 - a
ccuracy: 0.8920
Epoch 14/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2771 - a
ccuracy: 0.8953
Epoch 15/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2762 - a
ccuracy: 0.8950
Epoch 16/20
47/47 [==============================] - 1s 18ms/step - loss: 0.2752 - a
```

```
ccuracy: 0.8947
Epoch 17/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2584 - a
ccuracy: 0.9016
Epoch 18/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2586 - a
ccuracy: 0.9020
Epoch 19/20
47/47 [==============================] - 1s 18ms/step - loss: 0.2548 - a
ccuracy: 0.9013
Epoch 20/20
47/47 [==============================] - 1s 19ms/step - loss: 0.2516 - a
ccuracy: 0.9055
375/375 [==============================] - 0s 870us/step
y_pred (12000,) (12000,)
```

TODO 3.4.1 Process the data from cv_results to display average scores and aggregated confussion matrix. In the provided code sample pandas DataFrame is used, however, it can be achieved in many other ways, e.g. by converting lists to numpy arrays.

In [82]:
```
# display results
import pandas as pd

df = pd.DataFrame(cv_results)
df.head()
```

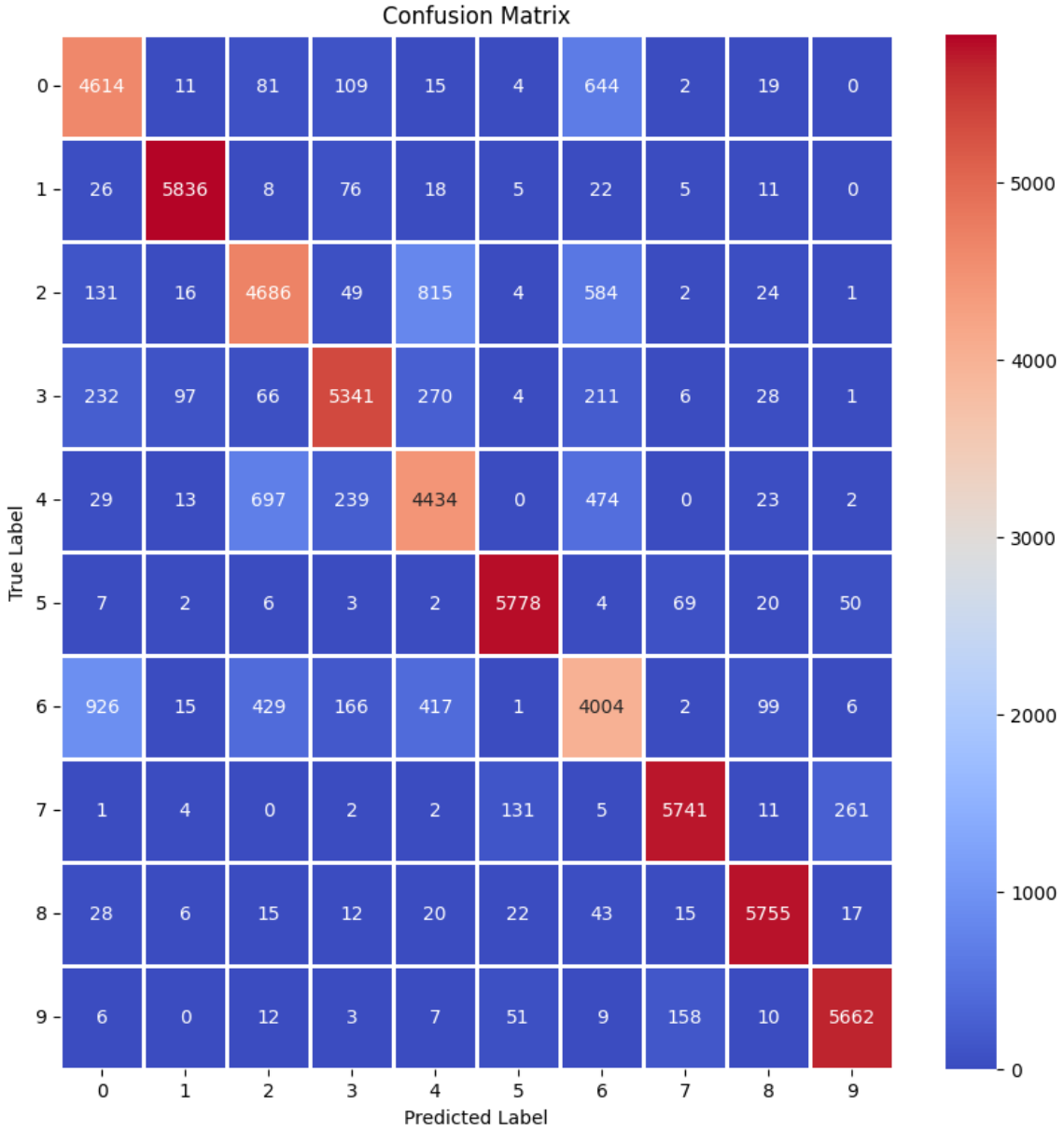
Out[82]:

	confusion_matrix	accuracy	precision	recall	f1
0	[[958, 2, 30, 31, 6, 0, 151, 1, 3, 0], [1, 116...	0.867083	0.867083	0.869252	0.866842
1	[[977, 5, 18, 20, 6, 0, 179, 0, 5, 0], [0, 116...	0.870750	0.870750	0.872000	0.869480
2	[[990, 2, 12, 30, 2, 3, 149, 1, 5, 0], [1, 115...	0.860833	0.860833	0.870435	0.859787
3	[[830, 2, 17, 17, 1, 1, 78, 0, 3, 0], [6, 1180...	0.866167	0.866167	0.873518	0.867095
4	[[859, 0, 4, 11, 0, 0, 87, 0, 3, 0], [18, 1175...	0.856083	0.856083	0.862718	0.855067

In [83]:
```
# Compute avarage scores
print("Average accuracy %.3f" % df['accuracy'].mean())
print("Average precision %.3f" % df['precision'].mean())
print("Average recall %.3f" % df['recall'].mean())
print("Average F1 %.3f" % df['f1'].mean())
```

```
Average accuracy 0.864
Average precision 0.864
Average recall 0.870
Average F1 0.864
```

In [84]:
```
# Aggregate (i.e. sum) confusion matrices for particular folds and displa
cm = np.sum (df['confusion_matrix'])
show_confusion_matrix(cm)
```



TODO 3.4.2 Perform the above experiments using **three** best configurations (based on train-test procedure. Compare results obtained by testing and cross-validation.

In [85]:
```
n_splits = 5
random_state =  123
config = configurations['Fifth configuration']

cv_results = {}

skf = StratifiedKFold(n_splits=n_splits, shuffle=True,random_state=random
for fold_number, (train_index, test_index) in enumerate(skf.split(X_train
  X_train_fold = X_train[train_index]
  y_train_fold = y_train[train_index]
  X_test_fold = X_train[test_index]
  y_test_fold = y_train[test_index]

  print(f'============= Fold#{fold_number+1} =============')
  results = perform_test(X_train_fold,y_train_fold,X_test_fold,y_test_fol

  for k in results:
    if k=='model':
      continue
```

```
    rlist=cv_results.get(k,[])
    rlist.append(results[k])
    cv_results[k]=rlist

# Compute avarage scores
print("Average accuracy %.3f" % df['accuracy'].mean())
print("Average precision %.3f" % df['precision'].mean())
print("Average recall %.3f" % df['recall'].mean())
print("Average F1 %.3f" % df['f1'].mean())
```

```
============= Fold#1 =============
Epoch 1/5
47/47 [==============================] - 4s 81ms/step - loss: 7855.8667
- accuracy: 0.5141
Epoch 2/5
47/47 [==============================] - 4s 80ms/step - loss: 2.3196 - a
ccuracy: 0.5429
Epoch 3/5
47/47 [==============================] - 4s 79ms/step - loss: 6.6517 - a
ccuracy: 0.1285
Epoch 4/5
47/47 [==============================] - 4s 79ms/step - loss: 2.3103 - a
ccuracy: 0.1004
Epoch 5/5
47/47 [==============================] - 4s 79ms/step - loss: 2.3087 - a
ccuracy: 0.0989
375/375 [==============================] - 1s 2ms/step
y_pred (12000,) (12000,)
============= Fold#2 =============
Epoch 1/5
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-p
ackages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Recall is ill-defined and being set to 0.0 in labels with no true sample
s. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
47/47 [==============================] - 4s 79ms/step - loss: 35472.8906
- accuracy: 0.5013
Epoch 2/5
47/47 [==============================] - 4s 79ms/step - loss: 0.9419 - a
ccuracy: 0.7064
Epoch 3/5
47/47 [==============================] - 4s 79ms/step - loss: 12.2383 -
accuracy: 0.2871
Epoch 4/5
47/47 [==============================] - 4s 79ms/step - loss: 2.3144 - a
ccuracy: 0.1003
Epoch 5/5
47/47 [==============================] - 4s 79ms/step - loss: 2.3139 - a
ccuracy: 0.0998
375/375 [==============================] - 1s 2ms/step
y_pred (12000,) (12000,)
============= Fold#3 =============
Epoch 1/5
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-p
ackages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Recall is ill-defined and being set to 0.0 in labels with no true sample
s. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
47/47 [==============================] - 4s 78ms/step - loss: 12882.5957
- accuracy: 0.4044
Epoch 2/5
47/47 [==============================] - 4s 79ms/step - loss: 0.9434 - a
ccuracy: 0.6899
Epoch 3/5
47/47 [==============================] - 4s 79ms/step - loss: 108.4986 -
accuracy: 0.5053
Epoch 4/5
47/47 [==============================] - 4s 79ms/step - loss: 1.3442 - a
ccuracy: 0.5379
Epoch 5/5
47/47 [==============================] - 4s 78ms/step - loss: 75.0953 -
accuracy: 0.4199
375/375 [==============================] - 1s 2ms/step
y_pred (12000,) (12000,)
============= Fold#4 =============
Epoch 1/5
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-p
ackages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:
Recall is ill-defined and being set to 0.0 in labels with no true sample
s. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
47/47 [==============================] - 4s 79ms/step - loss: 12720.5117
- accuracy: 0.5212
Epoch 2/5
47/47 [==============================] - 4s 79ms/step - loss: 0.8995 - a
ccuracy: 0.6989
Epoch 3/5
47/47 [==============================] - 4s 79ms/step - loss: 11.0554 -
accuracy: 0.2531
Epoch 4/5
47/47 [==============================] - 4s 80ms/step - loss: 1.4061 - a
ccuracy: 0.4449
Epoch 5/5
47/47 [==============================] - 4s 81ms/step - loss: 428.1472 -
accuracy: 0.3470
375/375 [==============================] - 1s 2ms/step
y_pred (12000,) (12000,)
============= Fold#5 =============
Epoch 1/5
47/47 [==============================] - 4s 80ms/step - loss: 9076.6064
- accuracy: 0.5186
Epoch 2/5
47/47 [==============================] - 4s 81ms/step - loss: 1.0768 - a
ccuracy: 0.6559
Epoch 3/5
47/47 [==============================] - 4s 78ms/step - loss: 301.8121 -
accuracy: 0.5029
Epoch 4/5
47/47 [==============================] - 4s 78ms/step - loss: 1.7048 - a
ccuracy: 0.4974
Epoch 5/5
47/47 [==============================] - 4s 78ms/step - loss: 1.3231 - a
ccuracy: 0.5264
375/375 [==============================] - 1s 2ms/step
y_pred (12000,) (12000,)
Average accuracy 0.864
Average precision 0.864
Average recall 0.870
Average F1 0.864
```

In [86]:

```
n_splits = 5
random_state =  123
config = configurations['Seventh configuration']

cv_results = {}

skf = StratifiedKFold(n_splits=n_splits, shuffle=True,random_state=random
for fold_number, (train_index, test_index) in enumerate(skf.split(X_train
  X_train_fold = X_train[train_index]
  y_train_fold = y_train[train_index]
  X_test_fold = X_train[test_index]
  y_test_fold = y_train[test_index]

  print(f'============= Fold#{fold_number+1} =============')
  results = perform_test(X_train_fold,y_train_fold,X_test_fold,y_test_fol

  for k in results:
    if k=='model':
      continue
    rlist=cv_results.get(k,[])
    rlist.append(results[k])
```

```
    cv_results[k]=rlist

# Compute avarage scores
print("Average accuracy %.3f" % df['accuracy'].mean())
print("Average precision %.3f" % df['precision'].mean())
print("Average recall %.3f" % df['recall'].mean())
print("Average F1 %.3f" % df['f1'].mean())
```

```
============= Fold#1 =============
Epoch 1/30
47/47 [==============================] - 1s 19ms/step - loss: 28.7803 -
accuracy: 0.6254
Epoch 2/30
47/47 [==============================] - 1s 19ms/step - loss: 0.7717 - a
ccuracy: 0.7661
Epoch 3/30
47/47 [==============================] - 1s 19ms/step - loss: 0.7469 - a
ccuracy: 0.7644
Epoch 4/30
47/47 [==============================] - 1s 20ms/step - loss: 0.5528 - a
ccuracy: 0.8039
Epoch 5/30
47/47 [==============================] - 1s 20ms/step - loss: 0.5652 - a
ccuracy: 0.8080
Epoch 6/30
47/47 [==============================] - 1s 19ms/step - loss: 0.4828 - a
ccuracy: 0.8291
Epoch 7/30
47/47 [==============================] - 1s 21ms/step - loss: 0.4382 - a
ccuracy: 0.8405
Epoch 8/30
47/47 [==============================] - 1s 21ms/step - loss: 0.4636 - a
ccuracy: 0.8393
Epoch 9/30
47/47 [==============================] - 1s 21ms/step - loss: 0.4055 - a
ccuracy: 0.8505
Epoch 10/30
47/47 [==============================] - 1s 19ms/step - loss: 0.4021 - a
ccuracy: 0.8537
Epoch 11/30
47/47 [==============================] - 1s 20ms/step - loss: 0.4210 - a
ccuracy: 0.8525
Epoch 12/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3770 - a
ccuracy: 0.8633
Epoch 13/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3711 - a
ccuracy: 0.8629
Epoch 14/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3657 - a
ccuracy: 0.8676
Epoch 15/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3607 - a
ccuracy: 0.8701
Epoch 16/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3729 - a
ccuracy: 0.8679
Epoch 17/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3501 - a
ccuracy: 0.8744
Epoch 18/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3587 - a
ccuracy: 0.8711
Epoch 19/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3407 - a
ccuracy: 0.8771
Epoch 20/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3403 - a
```

```
ccuracy: 0.8786
Epoch 21/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3340 - a
ccuracy: 0.8815
Epoch 22/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3383 - a
ccuracy: 0.8777
Epoch 23/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3377 - a
ccuracy: 0.8811
Epoch 24/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3223 - a
ccuracy: 0.8844
Epoch 25/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3366 - a
ccuracy: 0.8807
Epoch 26/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3175 - a
ccuracy: 0.8862
Epoch 27/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3264 - a
ccuracy: 0.8832
Epoch 28/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3286 - a
ccuracy: 0.8853
Epoch 29/30
47/47 [==============================] - 1s 22ms/step - loss: 0.3233 - a
ccuracy: 0.8838
Epoch 30/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3111 - a
ccuracy: 0.8890
375/375 [==============================] - 0s 1ms/step
y_pred (12000,) (12000,)
============= Fold#2 =============
Epoch 1/30
47/47 [==============================] - 1s 19ms/step - loss: 69.9137 -
accuracy: 0.6270
Epoch 2/30
47/47 [==============================] - 1s 20ms/step - loss: 0.7700 - a
ccuracy: 0.7759
Epoch 3/30
47/47 [==============================] - 1s 20ms/step - loss: 0.7585 - a
ccuracy: 0.7787
Epoch 4/30
47/47 [==============================] - 1s 20ms/step - loss: 0.5371 - a
ccuracy: 0.8108
Epoch 5/30
47/47 [==============================] - 1s 19ms/step - loss: 0.5111 - a
ccuracy: 0.8179
Epoch 6/30
47/47 [==============================] - 1s 19ms/step - loss: 0.4661 - a
ccuracy: 0.8327
Epoch 7/30
47/47 [==============================] - 1s 19ms/step - loss: 0.4363 - a
ccuracy: 0.8425
Epoch 8/30
47/47 [==============================] - 1s 20ms/step - loss: 0.4692 - a
ccuracy: 0.8379
Epoch 9/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3988 - a
```

```
ccuracy: 0.8534
Epoch 10/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3935 - a
ccuracy: 0.8555
Epoch 11/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3880 - a
ccuracy: 0.8578
Epoch 12/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3940 - a
ccuracy: 0.8577
Epoch 13/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3707 - a
ccuracy: 0.8640
Epoch 14/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3818 - a
ccuracy: 0.8617
Epoch 15/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3549 - a
ccuracy: 0.8697
Epoch 16/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3554 - a
ccuracy: 0.8688
Epoch 17/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3600 - a
ccuracy: 0.8693
Epoch 18/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3491 - a
ccuracy: 0.8741
Epoch 19/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3437 - a
ccuracy: 0.8766
Epoch 20/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3435 - a
ccuracy: 0.8767
Epoch 21/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3328 - a
ccuracy: 0.8808
Epoch 22/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3270 - a
ccuracy: 0.8786
Epoch 23/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3367 - a
ccuracy: 0.8810
Epoch 24/30
47/47 [==============================] - 1s 18ms/step - loss: 0.3190 - a
ccuracy: 0.8834
Epoch 25/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3219 - a
ccuracy: 0.8830
Epoch 26/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3562 - a
ccuracy: 0.8804
Epoch 27/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3070 - a
ccuracy: 0.8863
Epoch 28/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3199 - a
ccuracy: 0.8842
Epoch 29/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3108 - a
```

```
ccuracy: 0.8876
Epoch 30/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3130 - a
ccuracy: 0.8873
375/375 [==============================] - 0s 903us/step
y_pred (12000,) (12000,)
============= Fold#3 =============
Epoch 1/30
47/47 [==============================] - 1s 19ms/step - loss: 56.3827 -
accuracy: 0.5533
Epoch 2/30
47/47 [==============================] - 1s 21ms/step - loss: 0.8077 - a
ccuracy: 0.7459
Epoch 3/30
47/47 [==============================] - 1s 21ms/step - loss: 0.5695 - a
ccuracy: 0.8016
Epoch 4/30
47/47 [==============================] - 1s 20ms/step - loss: 0.5090 - a
ccuracy: 0.8151
Epoch 5/30
47/47 [==============================] - 1s 19ms/step - loss: 0.4639 - a
ccuracy: 0.8305
Epoch 6/30
47/47 [==============================] - 1s 19ms/step - loss: 0.4528 - a
ccuracy: 0.8364
Epoch 7/30
47/47 [==============================] - 1s 20ms/step - loss: 0.4265 - a
ccuracy: 0.8454
Epoch 8/30
47/47 [==============================] - 1s 20ms/step - loss: 0.4100 - a
ccuracy: 0.8496
Epoch 9/30
47/47 [==============================] - 1s 20ms/step - loss: 0.4005 - a
ccuracy: 0.8550
Epoch 10/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3993 - a
ccuracy: 0.8547
Epoch 11/30
47/47 [==============================] - 1s 22ms/step - loss: 0.3957 - a
ccuracy: 0.8583
Epoch 12/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3691 - a
ccuracy: 0.8659
Epoch 13/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3780 - a
ccuracy: 0.8635
Epoch 14/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3628 - a
ccuracy: 0.8667
Epoch 15/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3658 - a
ccuracy: 0.8691
Epoch 16/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3637 - a
ccuracy: 0.8684
Epoch 17/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3505 - a
ccuracy: 0.8744
Epoch 18/30
47/47 [==============================] - 1s 22ms/step - loss: 0.3499 - a
```

```
ccuracy: 0.8728
Epoch 19/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3514 - a
ccuracy: 0.8730
Epoch 20/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3470 - a
ccuracy: 0.8785
Epoch 21/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3472 - a
ccuracy: 0.8770
Epoch 22/30
47/47 [==============================] - 1s 18ms/step - loss: 0.3640 - a
ccuracy: 0.8789
Epoch 23/30
47/47 [==============================] - 1s 18ms/step - loss: 0.3292 - a
ccuracy: 0.8817
Epoch 24/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3496 - a
ccuracy: 0.8820
Epoch 25/30
47/47 [==============================] - 1s 18ms/step - loss: 0.3340 - a
ccuracy: 0.8826
Epoch 26/30
47/47 [==============================] - 1s 18ms/step - loss: 0.3288 - a
ccuracy: 0.8866
Epoch 27/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3227 - a
ccuracy: 0.8855
Epoch 28/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3279 - a
ccuracy: 0.8863
Epoch 29/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3208 - a
ccuracy: 0.8849
Epoch 30/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3216 - a
ccuracy: 0.8888
375/375 [==============================] - 0s 924us/step
y_pred (12000,) (12000,)
============= Fold#4 =============
Epoch 1/30
47/47 [==============================] - 1s 18ms/step - loss: 25.1028 -
accuracy: 0.6396
Epoch 2/30
47/47 [==============================] - 1s 19ms/step - loss: 0.6841 - a
ccuracy: 0.7670
Epoch 3/30
47/47 [==============================] - 1s 19ms/step - loss: 0.5823 - a
ccuracy: 0.7931
Epoch 4/30
47/47 [==============================] - 1s 20ms/step - loss: 0.5335 - a
ccuracy: 0.8135
Epoch 5/30
47/47 [==============================] - 1s 19ms/step - loss: 0.5722 - a
ccuracy: 0.8126
Epoch 6/30
47/47 [==============================] - 1s 21ms/step - loss: 0.4376 - a
ccuracy: 0.8429
Epoch 7/30
47/47 [==============================] - 1s 18ms/step - loss: 0.4594 - a
```

```
ccuracy: 0.8383
Epoch 8/30
47/47 [==============================] - 1s 18ms/step - loss: 0.4331 - a
ccuracy: 0.8463
Epoch 9/30
47/47 [==============================] - 1s 18ms/step - loss: 0.4181 - a
ccuracy: 0.8488
Epoch 10/30
47/47 [==============================] - 1s 20ms/step - loss: 0.4147 - a
ccuracy: 0.8492
Epoch 11/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3953 - a
ccuracy: 0.8546
Epoch 12/30
47/47 [==============================] - 1s 20ms/step - loss: 0.4231 - a
ccuracy: 0.8553
Epoch 13/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3868 - a
ccuracy: 0.8589
Epoch 14/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3852 - a
ccuracy: 0.8577
Epoch 15/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3683 - a
ccuracy: 0.8651
Epoch 16/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3843 - a
ccuracy: 0.8603
Epoch 17/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3537 - a
ccuracy: 0.8700
Epoch 18/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3638 - a
ccuracy: 0.8646
Epoch 19/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3553 - a
ccuracy: 0.8702
Epoch 20/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3511 - a
ccuracy: 0.8704
Epoch 21/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3551 - a
ccuracy: 0.8706
Epoch 22/30
47/47 [==============================] - 1s 22ms/step - loss: 0.3441 - a
ccuracy: 0.8750
Epoch 23/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3429 - a
ccuracy: 0.8740
Epoch 24/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3476 - a
ccuracy: 0.8745
Epoch 25/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3407 - a
ccuracy: 0.8741
Epoch 26/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3385 - a
ccuracy: 0.8793
Epoch 27/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3328 - a
```

```
ccuracy: 0.8788
Epoch 28/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3396 - a
ccuracy: 0.8780
Epoch 29/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3393 - a
ccuracy: 0.8818
Epoch 30/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3301 - a
ccuracy: 0.8835
375/375 [==============================] - 0s 909us/step
y_pred (12000,) (12000,)
============= Fold#5 =============
Epoch 1/30
47/47 [==============================] - 1s 19ms/step - loss: 32.3709 -
accuracy: 0.6147
Epoch 2/30
47/47 [==============================] - 1s 19ms/step - loss: 0.8789 - a
ccuracy: 0.7220
Epoch 3/30
47/47 [==============================] - 1s 19ms/step - loss: 0.6221 - a
ccuracy: 0.7927
Epoch 4/30
47/47 [==============================] - 1s 20ms/step - loss: 0.5373 - a
ccuracy: 0.8114
Epoch 5/30
47/47 [==============================] - 1s 20ms/step - loss: 0.5592 - a
ccuracy: 0.8160
Epoch 6/30
47/47 [==============================] - 1s 19ms/step - loss: 0.4524 - a
ccuracy: 0.8335
Epoch 7/30
47/47 [==============================] - 1s 18ms/step - loss: 0.4325 - a
ccuracy: 0.8412
Epoch 8/30
47/47 [==============================] - 1s 23ms/step - loss: 0.4701 - a
ccuracy: 0.8418
Epoch 9/30
47/47 [==============================] - 1s 21ms/step - loss: 0.4703 - a
ccuracy: 0.8420
Epoch 10/30
47/47 [==============================] - 1s 19ms/step - loss: 0.4022 - a
ccuracy: 0.8529
Epoch 11/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3845 - a
ccuracy: 0.8595
Epoch 12/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3978 - a
ccuracy: 0.8567
Epoch 13/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3742 - a
ccuracy: 0.8634
Epoch 14/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3734 - a
ccuracy: 0.8616
Epoch 15/30
47/47 [==============================] - 1s 18ms/step - loss: 0.3597 - a
ccuracy: 0.8678
Epoch 16/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3577 - a
```

```
ccuracy: 0.8690
Epoch 17/30
47/47 [==============================] - 1s 19ms/step - loss: 0.3542 - a
ccuracy: 0.8673
Epoch 18/30
47/47 [==============================] - 1s 18ms/step - loss: 0.3701 - a
ccuracy: 0.8665
Epoch 19/30
47/47 [==============================] - 1s 18ms/step - loss: 0.3470 - a
ccuracy: 0.8738
Epoch 20/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3565 - a
ccuracy: 0.8700
Epoch 21/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3463 - a
ccuracy: 0.8754
Epoch 22/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3414 - a
ccuracy: 0.8758
Epoch 23/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3361 - a
ccuracy: 0.8765
Epoch 24/30
47/47 [==============================] - 1s 22ms/step - loss: 0.3505 - a
ccuracy: 0.8736
Epoch 25/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3466 - a
ccuracy: 0.8770
Epoch 26/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3299 - a
ccuracy: 0.8789
Epoch 27/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3367 - a
ccuracy: 0.8781
Epoch 28/30
47/47 [==============================] - 1s 21ms/step - loss: 0.3817 - a
ccuracy: 0.8727
Epoch 29/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3388 - a
ccuracy: 0.8814
Epoch 30/30
47/47 [==============================] - 1s 20ms/step - loss: 0.3172 - a
ccuracy: 0.8836
375/375 [==============================] - 0s 977us/step
y_pred (12000,) (12000,)
Average accuracy 0.864
Average precision 0.864
Average recall 0.870
Average F1 0.864
```

In [ ]:

```
n_splits = 5
random_state =  123
config = configurations['Sixth configuration']

cv_results = {}

skf = StratifiedKFold(n_splits=n_splits, shuffle=True,random_state=random
for fold_number, (train_index, test_index) in enumerate(skf.split(X_train
  X_train_fold = X_train[train_index]
  y_train_fold = y_train[train_index]
```

```
  X_test_fold = X_train[test_index]
  y_test_fold = y_train[test_index]

  print(f'============= Fold#{fold_number+1} =============')
  results = perform_test(X_train_fold,y_train_fold,X_test_fold,y_test_fol

  for k in results:
    if k=='model':
      continue
    rlist=cv_results.get(k,[])
    rlist.append(results[k])
    cv_results[k]=rlist

# Compute avarage scores
print("Average accuracy %.3f" % df['accuracy'].mean())
print("Average precision %.3f" % df['precision'].mean())
print("Average recall %.3f" % df['recall'].mean())
print("Average F1 %.3f" % df['f1'].mean())
```

Testing the winning model

TODO 3.4.3 Build a model using the winnining configuration (according to CV) and perform the final test on the delivered test set (X_test,y_test).

In [55]:
```
# test based on training data
config=configurations['First configuration']
results = perform_test(X_train, y_train, X_test, y_test, **config)
```

```
Epoch 1/20
59/59 [==============================] - 2s 19ms/step - loss: 3.5043 - a
ccuracy: 0.7086
Epoch 2/20
59/59 [==============================] - 1s 19ms/step - loss: 0.5155 - a
ccuracy: 0.8244
Epoch 3/20
59/59 [==============================] - 1s 18ms/step - loss: 0.4349 - a
ccuracy: 0.8473
Epoch 4/20
59/59 [==============================] - 1s 18ms/step - loss: 0.3877 - a
ccuracy: 0.8604
Epoch 5/20
59/59 [==============================] - 1s 18ms/step - loss: 0.3575 - a
ccuracy: 0.8698
Epoch 6/20
59/59 [==============================] - 1s 18ms/step - loss: 0.3371 - a
ccuracy: 0.8764
Epoch 7/20
59/59 [==============================] - 1s 19ms/step - loss: 0.3232 - a
ccuracy: 0.8809
Epoch 8/20
59/59 [==============================] - 1s 21ms/step - loss: 0.3110 - a
ccuracy: 0.8853
Epoch 9/20
59/59 [==============================] - 1s 20ms/step - loss: 0.3012 - a
ccuracy: 0.8896
Epoch 10/20
59/59 [==============================] - 1s 21ms/step - loss: 0.2949 - a
ccuracy: 0.8913
Epoch 11/20
59/59 [==============================] - 1s 18ms/step - loss: 0.2886 - a
ccuracy: 0.8942
Epoch 12/20
59/59 [==============================] - 1s 18ms/step - loss: 0.2805 - a
ccuracy: 0.8968
Epoch 13/20
59/59 [==============================] - 1s 18ms/step - loss: 0.2741 - a
ccuracy: 0.8996
Epoch 14/20
59/59 [==============================] - 1s 18ms/step - loss: 0.2727 - a
ccuracy: 0.9000
Epoch 15/20
59/59 [==============================] - 1s 20ms/step - loss: 0.2591 - a
ccuracy: 0.9046
Epoch 16/20
59/59 [==============================] - 1s 18ms/step - loss: 0.2618 - a
ccuracy: 0.9045
Epoch 17/20
59/59 [==============================] - 1s 18ms/step - loss: 0.2557 - a
ccuracy: 0.9044
Epoch 18/20
59/59 [==============================] - 1s 18ms/step - loss: 0.2477 - a
ccuracy: 0.9080
Epoch 19/20
59/59 [==============================] - 1s 19ms/step - loss: 0.2500 - a
ccuracy: 0.9084
Epoch 20/20
59/59 [==============================] - 1s 19ms/step - loss: 0.2411 - a
ccuracy: 0.9114
```

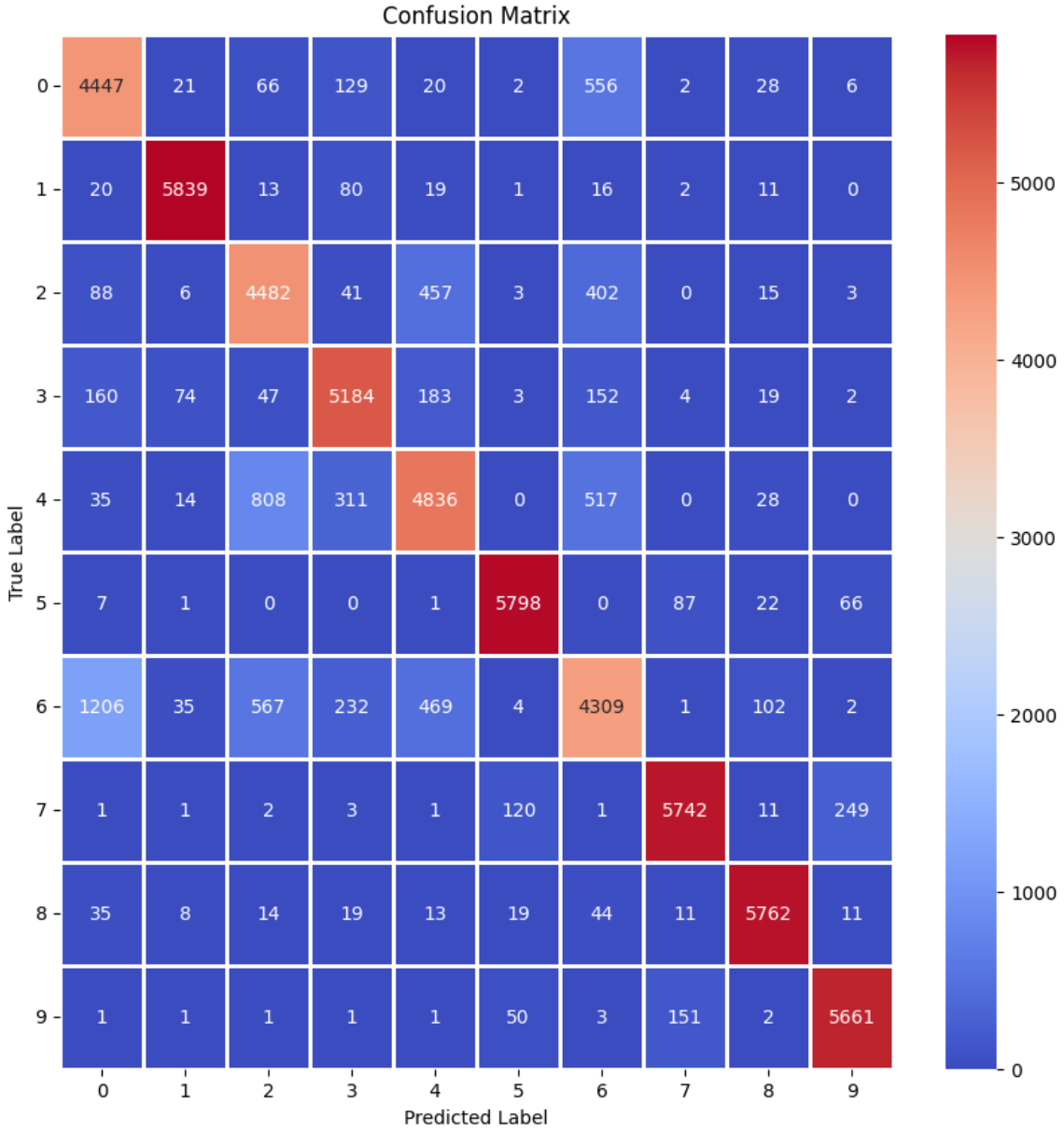
```
313/313 [==============================] - 0s 986us/step
y_pred (10000,) (10000,)
```

Print results

In [59]:
```
for k in results:
  if k == 'confusion_matrix' or k=='model':
    continue
  rlist=cv_results.get(k,[])
  rlist.append(results[k])
  cv_results[k]=rlist


print("Average accuracy %.3f" % df['accuracy'].mean())
print("Average precision %.3f" % df['precision'].mean())
print("Average recall %.3f" % df['recall'].mean())
print("Average F1 %.3f" % df['f1'].mean())
```

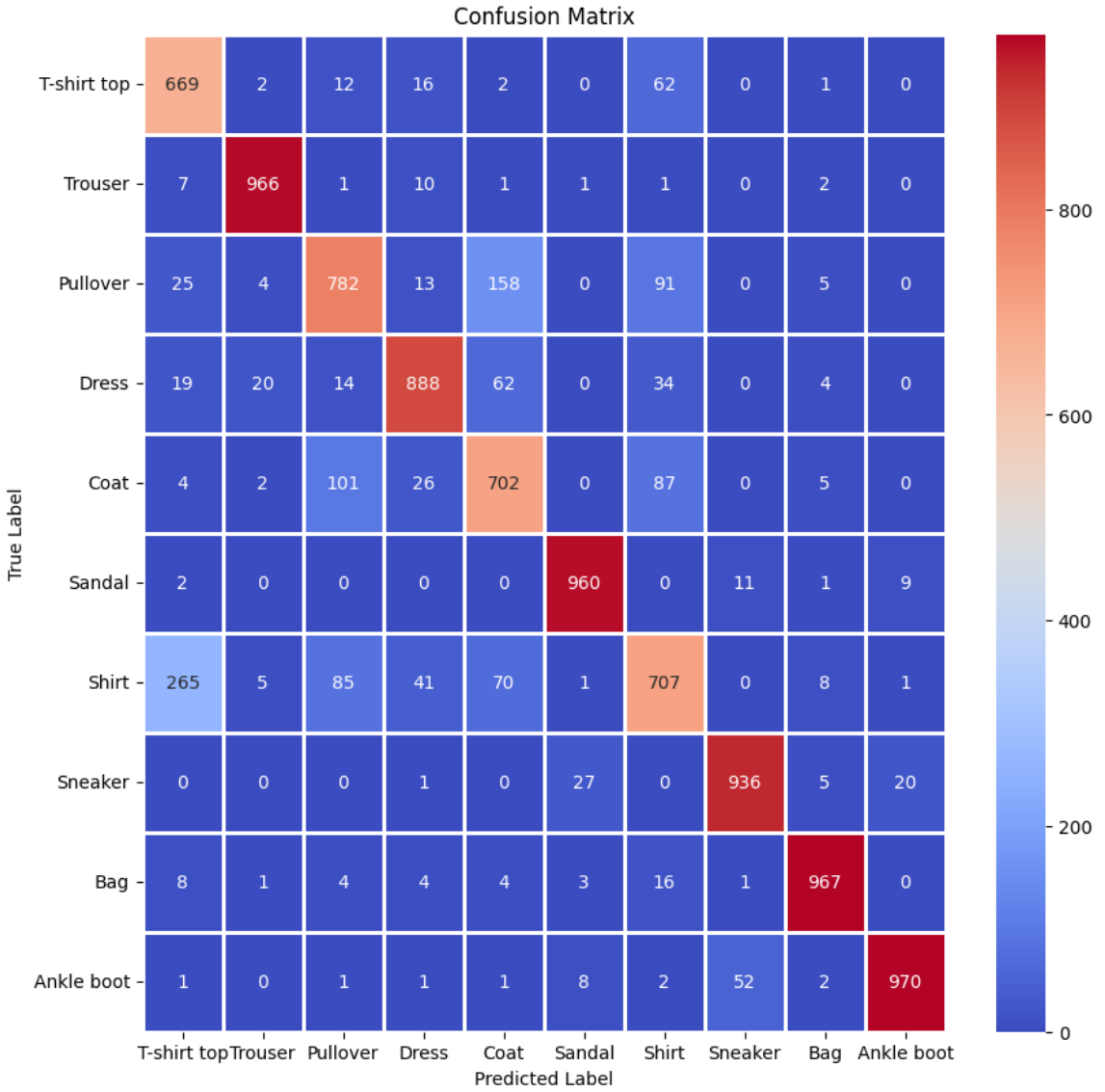
```
Average accuracy 0.868
Average precision 0.868
Average recall 0.874
Average F1 0.869
```

In [60]:
```
cm = np.sum (df['confusion_matrix'])
show_confusion_matrix(cm)
```



Display the confusion matrix

In [61]:
```
labels = ['T-shirt top','Trouser','Pullover','Dress','Coat','Sandal','Shi
show_confusion_matrix(results['confusion_matrix'],labels)
```



3.5 Cifar dataset

We will process Cifar dataset using a resource demanding CNN model.

In [98]:
```
from keras.datasets import cifar10
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

In [99]:
```
print(X_train.shape)
```

```
(50000, 32, 32, 3)
```

In [100…
```
# Show a few sample images from the training set
import matplotlib.pyplot as plt

plt.rcParams['figure.figsize'] = (2.5, 2.5) # set default size of plots
col1 = 10
row1 = 1
fig = plt.figure(figsize=(col1, row1))
for index in range(0, col1*row1):
    fig.add_subplot(row1, col1, index + 1)
    plt.axis('off')
    plt.imshow(X_train[index]) # index of the sample picture
    plt.title("Class " + str(y_train[index]))
```

```
plt.show()

# Show a few sample digits from the training set
plt.rcParams['figure.figsize'] = (1.0, 1.0) # set default size of plots
col2 = 20
row2 = 10
fig = plt.figure(figsize=(col2, row2))
for index in range(col1*row1, col1*row1 + col2*row2):
    fig.add_subplot(row2, col2, index - col1*row1 + 1)
    plt.axis('off')
    plt.imshow(X_train[index]) # index of the sample picture
plt.show()
```

Some basic preprocessing and calculation of the input shape

In [101…

```
X_train = X_train.astype('float32')   # Copy this array and cast it to a
X_test = X_test.astype('float32')     # Copy this array and cast it to a
X_train /= 255 # Transfrom the training data from the range of 0 and 255
X_test /= 255  # Transfrom the validating data from the range of 0 and 25
input_shape = X_train.shape[1:]
print(f'input shape={input_shape}')
```

```
input shape=(32, 32, 3)
```

Definition of CNN model

In [105…

```
from keras import layers
from keras import models
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import regularizers


num_classes = y_train.max()+1

# Define the sequential Keras model composed of a few layers
```

```
model3 = Sequential()   # establishes the type of the network model
# Conv2D - creates a convolutional layer (https://keras.io/layers/convolu
# filters - specified number of convolutional filters
# kernel_size - defines the frame (sliding window) size where the convolu
# activation - sets the activation function for this layers, here ReLU
# input_shape - defines the shape of the input matrix (vector), here inpu
model3.add(Conv2D(64, kernel_size=(3, 3),activation='relu', input_shape=i
model3.add(Conv2D(64, (3, 3), activation='relu'))
# MaxPooling2D pools the max value from the frame (sliding window) of 2 x
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(Dropout(0.25)) # Implements the drop out with the probability
model3.add(Conv2D(128, (3, 3), activation='relu',padding='same'))
#model3.add(Conv2D(128, (3, 3), activation='relu',padding='same'))
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(Dropout(0.35))
model3.add(Conv2D(256, (3, 3), activation='relu',padding='same'))
#model3.add(Conv2D(256, (3, 3), activation='relu',padding='same'))
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(Dropout(0.35))
# Finish the convolutional model and flatten the layer which does not aff
model3.add(Flatten())
# Use a dense layer (MLP) consisting of 256 neurons with relu activation
model3.add(Dense(512, activation='relu', kernel_regularizer=regularizers.
model3.add(Dropout(0.5))
model3.add(Dense(256, activation='relu', kernel_regularizer=regularizers.
model3.add(Dropout(0.3))
model3.add(Dense(num_classes, activation='softmax'))

# Now print the full model and notice the number of all trainable paramte
print(model3.summary())
```

```
Model: "sequential_40"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_54 (Conv2D)          (None, 30, 30, 64)        1792

 conv2d_55 (Conv2D)          (None, 28, 28, 64)        36928

 max_pooling2d_53 (MaxPoolin  (None, 14, 14, 64)       0
 g2D)

 dropout_5 (Dropout)         (None, 14, 14, 64)        0

 conv2d_56 (Conv2D)          (None, 14, 14, 128)       73856

 max_pooling2d_54 (MaxPoolin  (None, 7, 7, 128)        0
 g2D)

 dropout_6 (Dropout)         (None, 7, 7, 128)         0

 conv2d_57 (Conv2D)          (None, 7, 7, 256)         295168

 max_pooling2d_55 (MaxPoolin  (None, 3, 3, 256)        0
 g2D)

 dropout_7 (Dropout)         (None, 3, 3, 256)         0

 flatten_38 (Flatten)        (None, 2304)              0

 dense_76 (Dense)            (None, 512)               1180160

 dropout_8 (Dropout)         (None, 512)               0

 dense_77 (Dense)            (None, 256)               131328

 dropout_9 (Dropout)         (None, 256)               0

 dense_78 (Dense)            (None, 10)                2570

=================================================================
Total params: 1,721,802
Trainable params: 1,721,802
Non-trainable params: 0
_________________________________________________________________
None
```

As the number of parameters is high, training the model requires a few dozen of epochs. As an epoch may last as long as 160 sec, the training phase may last over 2 hours

To accelerate a bit and make the whole process reliable:

- We will train a model on a subsample of the training set and we will check scores
- Then we will train using the full training set, in a few increments
- Models will be saved to be restored in the case of the seession crach

Subsampling

We will use only 10% of data for training

In [106…
```
import numpy as np

probs = np.random.rand(y_train.shape[0])

X_train_sample = X_train[probs<0.1]
y_train_sample =y_train[probs<0.1]

print(X_train_sample.shape)
```

```
(5072, 32, 32, 3)
```

Model preparation

In [107…
```
# from tensorflow import keras
from keras import optimizers
# dir(optimizers)
optimizer = optimizers.adam_v2.Adam(learning_rate=0.001)
model3.compile(loss="sparse_categorical_crossentropy",optimizer=optimizer
```

In [109…
```
import os
os.mkdir('models')
```

Training

In [110…
```
batch_size = 512
epochs = 50

for i in range(10):
  print(f'--- Increment {i} on a subsampled dataset ---')
  model3.fit(X_train_sample, y_train_sample,batch_size=batch_size,epochs=
  path=f'models/CNN_CIFAR-10_sub{i}.h5'
  model3.save(path)
```

```
--- Increment 0 on a subsampled dataset ---
Epoch 1/5
10/10 [==============================] - 6s 555ms/step - loss: 3.3609 -
accuracy: 0.1025
Epoch 2/5
10/10 [==============================] - 5s 506ms/step - loss: 3.0295 -
accuracy: 0.1337
Epoch 3/5
10/10 [==============================] - 5s 497ms/step - loss: 2.7467 -
accuracy: 0.1893
Epoch 4/5
10/10 [==============================] - 5s 496ms/step - loss: 2.4925 -
accuracy: 0.2163
Epoch 5/5
10/10 [==============================] - 5s 499ms/step - loss: 2.3363 -
accuracy: 0.2348
--- Increment 1 on a subsampled dataset ---
Epoch 1/5
10/10 [==============================] - 5s 499ms/step - loss: 2.2257 -
accuracy: 0.2721
Epoch 2/5
10/10 [==============================] - 5s 499ms/step - loss: 2.1687 -
accuracy: 0.2784
Epoch 3/5
10/10 [==============================] - 5s 500ms/step - loss: 2.0427 -
accuracy: 0.3117
Epoch 4/5
10/10 [==============================] - 5s 501ms/step - loss: 1.9607 -
accuracy: 0.3263
Epoch 5/5
10/10 [==============================] - 5s 503ms/step - loss: 1.8713 -
accuracy: 0.3653
--- Increment 2 on a subsampled dataset ---
Epoch 1/5
10/10 [==============================] - 5s 544ms/step - loss: 1.8752 -
accuracy: 0.3677
Epoch 2/5
10/10 [==============================] - 6s 561ms/step - loss: 1.7926 -
accuracy: 0.3813
Epoch 3/5
10/10 [==============================] - 6s 572ms/step - loss: 1.7607 -
accuracy: 0.3967
Epoch 4/5
10/10 [==============================] - 6s 570ms/step - loss: 1.6948 -
accuracy: 0.4180
Epoch 5/5
10/10 [==============================] - 5s 515ms/step - loss: 1.6605 -
accuracy: 0.4322
--- Increment 3 on a subsampled dataset ---
Epoch 1/5
10/10 [==============================] - 5s 511ms/step - loss: 1.6508 -
accuracy: 0.4381
Epoch 2/5
10/10 [==============================] - 5s 505ms/step - loss: 1.6258 -
accuracy: 0.4541
Epoch 3/5
10/10 [==============================] - 5s 504ms/step - loss: 1.5574 -
accuracy: 0.4702
Epoch 4/5
10/10 [==============================] - 5s 546ms/step - loss: 1.5223 -
```

```
accuracy: 0.4961
Epoch 5/5
10/10 [==============================] - 5s 526ms/step - loss: 1.4878 -
accuracy: 0.4990
--- Increment 4 on a subsampled dataset ---
Epoch 1/5
10/10 [==============================] - 5s 510ms/step - loss: 1.4616 -
accuracy: 0.5187
Epoch 2/5
10/10 [==============================] - 5s 508ms/step - loss: 1.4309 -
accuracy: 0.5319
Epoch 3/5
10/10 [==============================] - 5s 511ms/step - loss: 1.4225 -
accuracy: 0.5331
Epoch 4/5
10/10 [==============================] - 5s 511ms/step - loss: 1.3730 -
accuracy: 0.5519
Epoch 5/5
10/10 [==============================] - 5s 514ms/step - loss: 1.3334 -
accuracy: 0.5718
--- Increment 5 on a subsampled dataset ---
Epoch 1/5
10/10 [==============================] - 5s 513ms/step - loss: 1.3088 -
accuracy: 0.5814
Epoch 2/5
10/10 [==============================] - 5s 510ms/step - loss: 1.2790 -
accuracy: 0.5938
Epoch 3/5
10/10 [==============================] - 5s 508ms/step - loss: 1.2858 -
accuracy: 0.5935
Epoch 4/5
10/10 [==============================] - 5s 508ms/step - loss: 1.2350 -
accuracy: 0.6078
Epoch 5/5
10/10 [==============================] - 5s 515ms/step - loss: 1.1986 -
accuracy: 0.6337
--- Increment 6 on a subsampled dataset ---
Epoch 1/5
10/10 [==============================] - 5s 509ms/step - loss: 1.1862 -
accuracy: 0.6280
Epoch 2/5
10/10 [==============================] - 5s 511ms/step - loss: 1.1510 -
accuracy: 0.6431
Epoch 3/5
10/10 [==============================] - 5s 513ms/step - loss: 1.1280 -
accuracy: 0.6629
Epoch 4/5
10/10 [==============================] - 5s 506ms/step - loss: 1.1446 -
accuracy: 0.6544
Epoch 5/5
10/10 [==============================] - 5s 519ms/step - loss: 1.1313 -
accuracy: 0.6605
--- Increment 7 on a subsampled dataset ---
Epoch 1/5
10/10 [==============================] - 5s 539ms/step - loss: 1.0926 -
accuracy: 0.6772
Epoch 2/5
10/10 [==============================] - 5s 509ms/step - loss: 1.0652 -
accuracy: 0.6918
Epoch 3/5
```

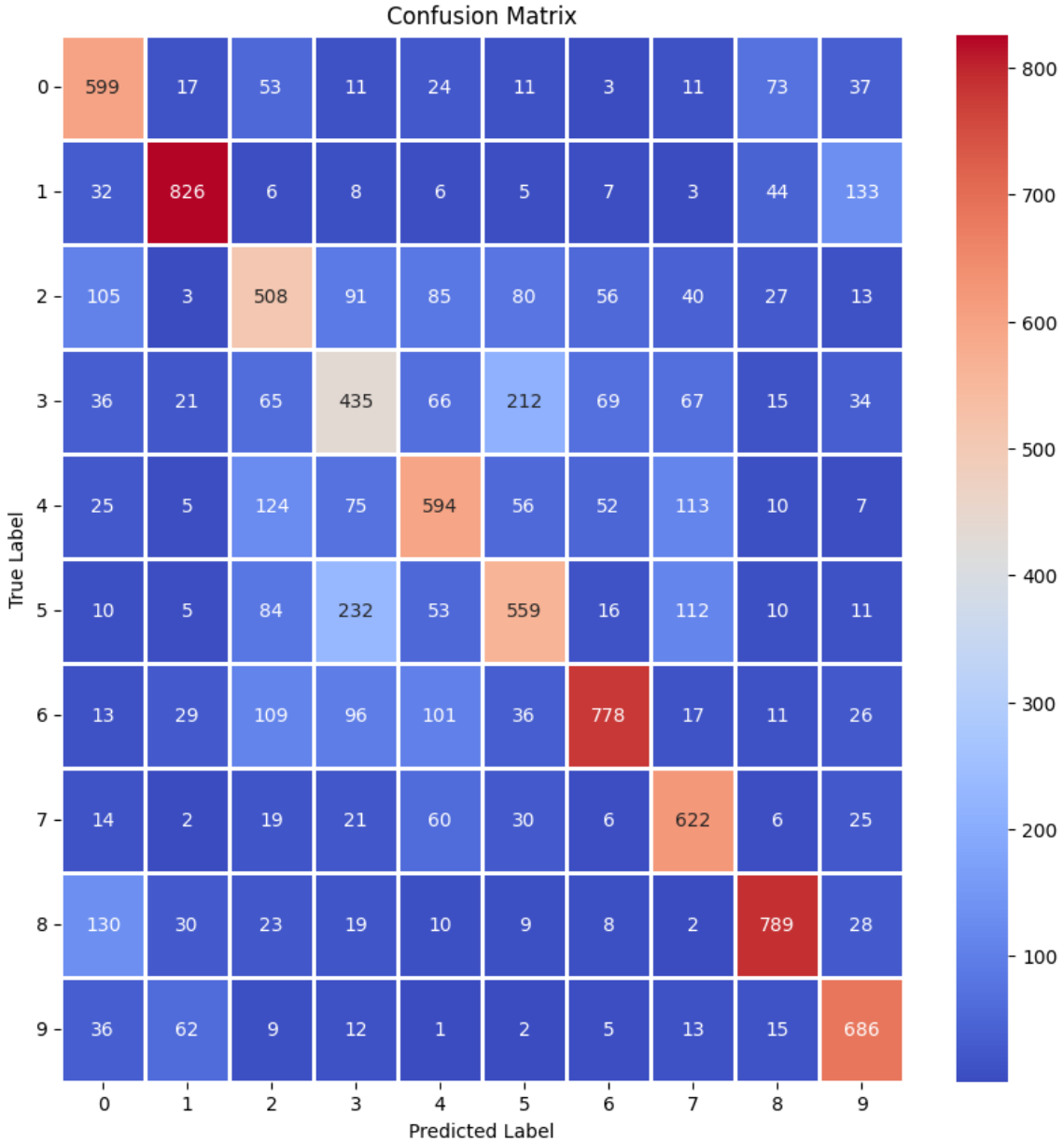
```
10/10 [==============================] - 5s 507ms/step - loss: 1.0404 -
accuracy: 0.7082
Epoch 4/5
10/10 [==============================] - 5s 501ms/step - loss: 1.0081 -
accuracy: 0.7104
Epoch 5/5
10/10 [==============================] - 5s 506ms/step - loss: 0.9743 -
accuracy: 0.7267
--- Increment 8 on a subsampled dataset ---
Epoch 1/5
10/10 [==============================] - 5s 504ms/step - loss: 0.9418 -
accuracy: 0.7336
Epoch 2/5
10/10 [==============================] - 5s 500ms/step - loss: 0.9574 -
accuracy: 0.7348
Epoch 3/5
10/10 [==============================] - 5s 503ms/step - loss: 0.9363 -
accuracy: 0.7401
Epoch 4/5
10/10 [==============================] - 5s 505ms/step - loss: 0.9217 -
accuracy: 0.7502
Epoch 5/5
10/10 [==============================] - 5s 506ms/step - loss: 0.8789 -
accuracy: 0.7715
--- Increment 9 on a subsampled dataset ---
Epoch 1/5
10/10 [==============================] - 5s 500ms/step - loss: 0.8551 -
accuracy: 0.7750
Epoch 2/5
10/10 [==============================] - 5s 505ms/step - loss: 0.8570 -
accuracy: 0.7743
Epoch 3/5
10/10 [==============================] - 5s 501ms/step - loss: 0.8537 -
accuracy: 0.7739
Epoch 4/5
10/10 [==============================] - 5s 502ms/step - loss: 0.8328 -
accuracy: 0.7863
Epoch 5/5
10/10 [==============================] - 5s 501ms/step - loss: 0.8155 -
accuracy: 0.7951
```

Validation on the test set

TODO 3.5.1 Display scores and the confussion matrix

In [72]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score

preds = model3.predict(X_test)
y_pred = np.argmax(preds,axis=1)

# print scores
print('Accuracy:', accuracy_score(y_pred,y_test))
print('Precision:', precision_score(y_pred,y_test,average='macro'))
print('Recall:', recall_score(y_pred,y_test,average='macro'))
print('F1:', f1_score(y_pred,y_test,average='macro'))

# display confusion matrix
show_confusion_matrix(confusion_matrix(y_pred,y_test))
```

```
313/313 [==============================] - 5s 16ms/step
Accuracy: 0.6396
Precision: 0.6396
Recall: 0.6469132227552359
F1: 0.6404131539967299
```



Continue training on the whole dataset

In [111…

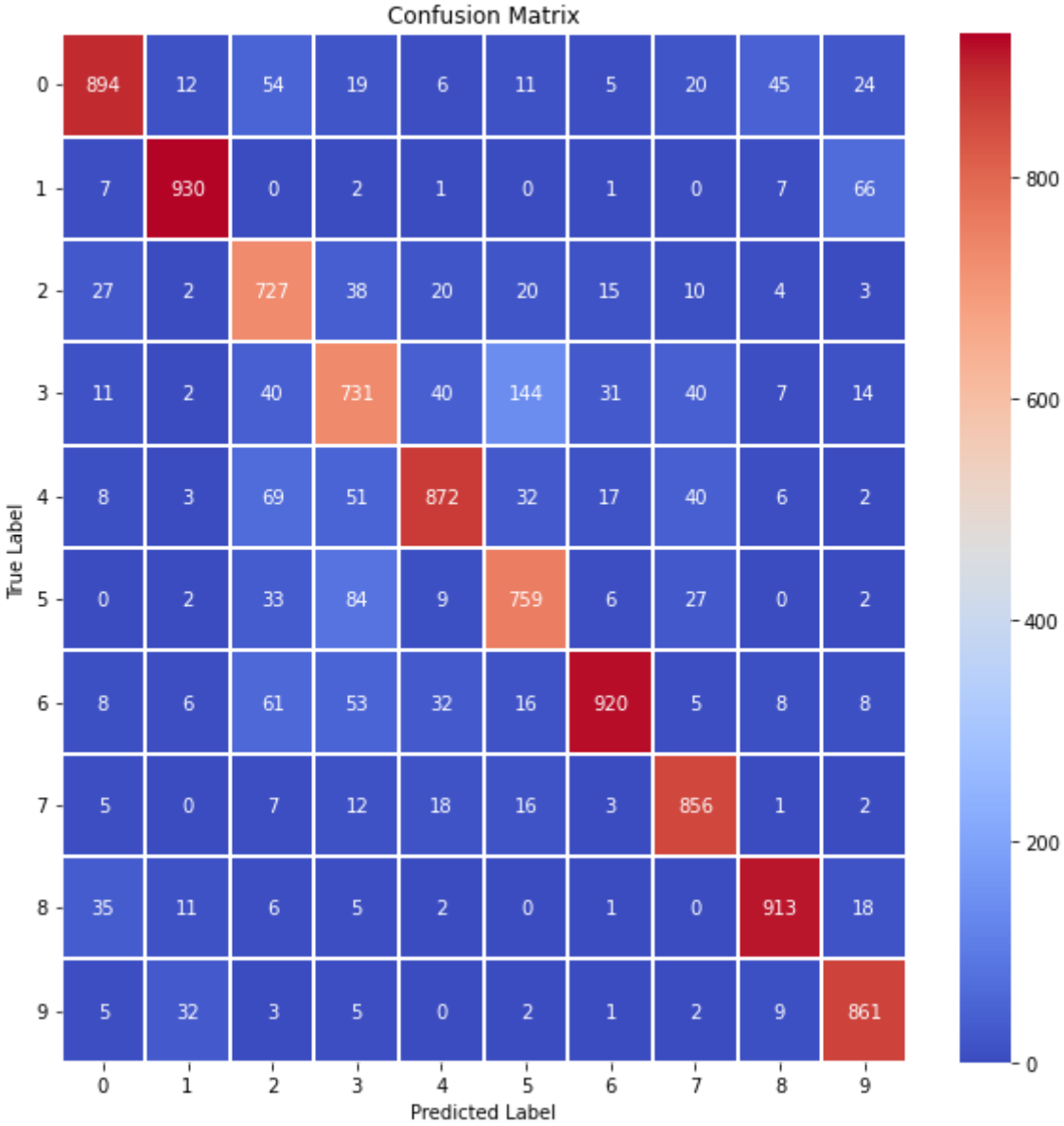
```
i=i+1
print(f'--- Increment {i} on a full dataset ---')
model3.fit(X_train, y_train,batch_size=batch_size,epochs=10,verbose=1)
path=f'models/CNN_CIFAR-10_full{i}.h5'
model3.save(path)
```

```
--- Increment 10 on a full dataset ---
Epoch 1/10
98/98 [==============================] - 50s 511ms/step - loss: 1.3293 -
accuracy: 0.6203
Epoch 2/10
98/98 [==============================] - 49s 503ms/step - loss: 1.1641 -
accuracy: 0.6635
Epoch 3/10
98/98 [==============================] - 49s 501ms/step - loss: 1.0745 -
accuracy: 0.6897
Epoch 4/10
98/98 [==============================] - 51s 522ms/step - loss: 1.0269 -
accuracy: 0.6999
Epoch 5/10
98/98 [==============================] - 49s 499ms/step - loss: 0.9785 -
accuracy: 0.7203
Epoch 6/10
98/98 [==============================] - 50s 512ms/step - loss: 0.9522 -
accuracy: 0.7280
Epoch 7/10
98/98 [==============================] - 51s 516ms/step - loss: 0.9178 -
accuracy: 0.7412
Epoch 8/10
98/98 [==============================] - 49s 500ms/step - loss: 0.9002 -
accuracy: 0.7507
Epoch 9/10
98/98 [==============================] - 49s 502ms/step - loss: 0.8804 -
accuracy: 0.7553
Epoch 10/10
98/98 [==============================] - 50s 510ms/step - loss: 0.8591 -
accuracy: 0.7626
```

TODO 3.5.2 Print scores and display confusion matrix

In [ ]:
```
preds = model3.predict(X_test)
y_pred = np.argmax(preds,axis=1)

# print scores
print('Accuracy:', accuracy_score(y_pred,y_test))
print('Precision:', precision_score(y_pred,y_test,average='macro'))
print('Recall:', recall_score(y_pred,y_test,average='macro'))
print('F1:', f1_score(y_pred,y_test,average='macro'))

# display confusion matrix
show_confusion_matrix(confusion_matrix(y_pred,y_test))
```

```
Accuracy: 0.8463
Precision: 0.8463
Recall: 0.849362620114289
F1: 0.8463054925140712
```



TODO 3.5.3 Repeat the two above steps (training the model and testing) until satisfying results are reached. You may consider writing a loop, which terminates upon certain conditions, e.g. no improvement or reaching a desired score value

In [112…

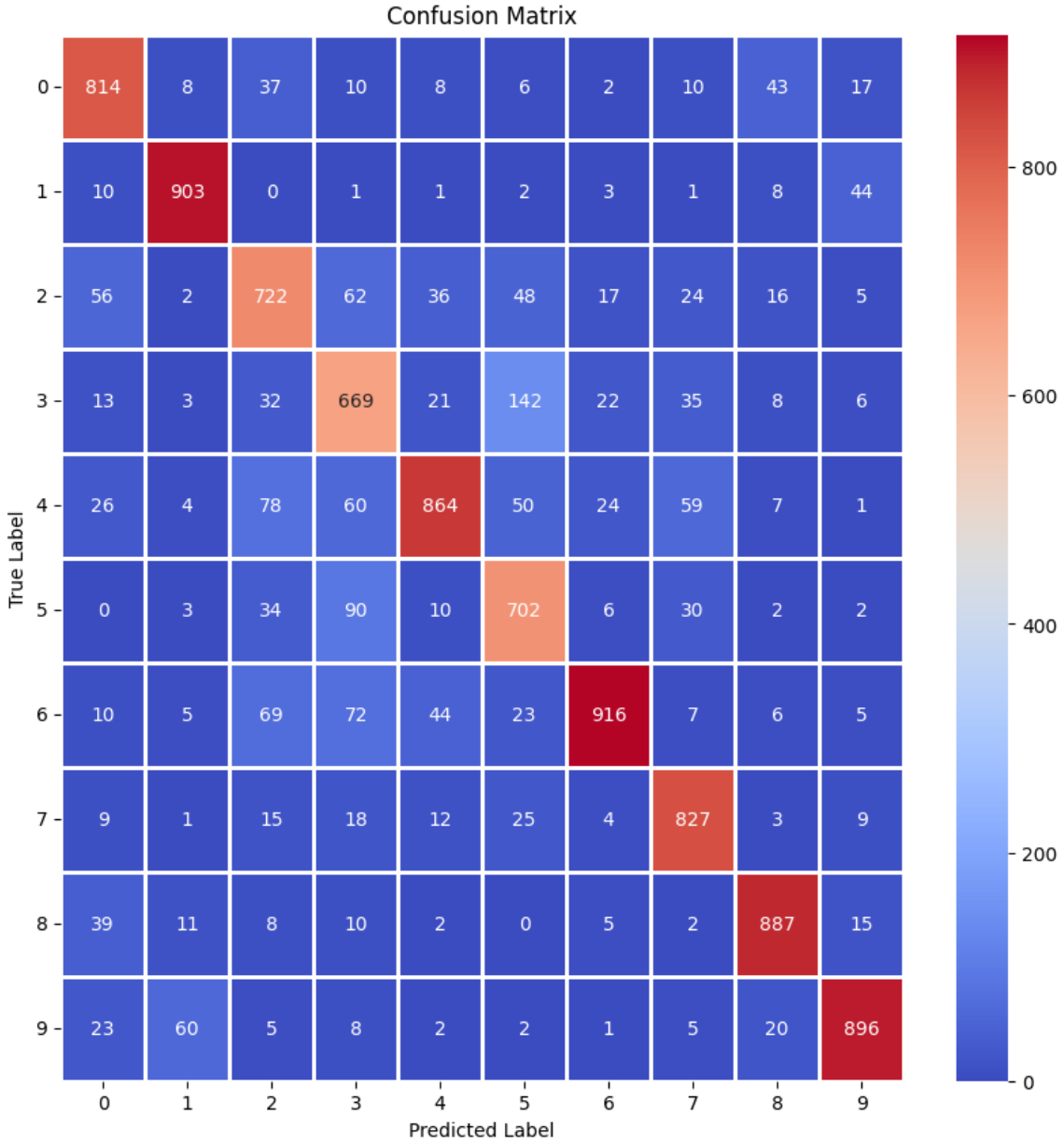
```
probs = np.random.rand(y_train.shape[0])

X_train_sample = X_train[probs<0.5]
y_train_sample =y_train[probs<0.5]

batch_size = 512
epochs = 50

for i in range(10):
  print(f'--- Increment {i} on a subsampled dataset ---')
  model3.fit(X_train_sample, y_train_sample,batch_size=batch_size,epochs=
  path=f'models/CNN_CIFAR-10_sub{i}.h5'
  model3.save(path)

preds = model3.predict(X_test)
y_pred = np.argmax(preds,axis=1)

# print scores
```

```
print('Accuracy:', accuracy_score(y_pred,y_test))
print('Precision:', precision_score(y_pred,y_test,average='macro'))
print('Recall:', recall_score(y_pred,y_test,average='macro'))
print('F1:', f1_score(y_pred,y_test,average='macro'))

# display confusion matrix
show_confusion_matrix(confusion_matrix(y_pred,y_test))
```

```
--- Increment 0 on a subsampled dataset ---
Epoch 1/5
49/49 [==============================] - 25s 508ms/step - loss: 0.8376 -
accuracy: 0.7751
Epoch 2/5
49/49 [==============================] - 25s 506ms/step - loss: 0.8053 -
accuracy: 0.7847
Epoch 3/5
49/49 [==============================] - 25s 509ms/step - loss: 0.7967 -
accuracy: 0.7874
Epoch 4/5
49/49 [==============================] - 25s 507ms/step - loss: 0.7745 -
accuracy: 0.7966
Epoch 5/5
49/49 [==============================] - 25s 504ms/step - loss: 0.7801 -
accuracy: 0.7940
--- Increment 1 on a subsampled dataset ---
Epoch 1/5
49/49 [==============================] - 25s 506ms/step - loss: 0.7521 -
accuracy: 0.8059
Epoch 2/5
49/49 [==============================] - 25s 518ms/step - loss: 0.7450 -
accuracy: 0.8080
Epoch 3/5
49/49 [==============================] - 28s 576ms/step - loss: 0.7408 -
accuracy: 0.8111
Epoch 4/5
49/49 [==============================] - 26s 531ms/step - loss: 0.7280 -
accuracy: 0.8190
Epoch 5/5
49/49 [==============================] - 26s 540ms/step - loss: 0.7252 -
accuracy: 0.8210
--- Increment 2 on a subsampled dataset ---
Epoch 1/5
49/49 [==============================] - 25s 505ms/step - loss: 0.7113 -
accuracy: 0.8280
Epoch 2/5
49/49 [==============================] - 25s 501ms/step - loss: 0.7070 -
accuracy: 0.8276
Epoch 3/5
49/49 [==============================] - 25s 505ms/step - loss: 0.7087 -
accuracy: 0.8281
Epoch 4/5
49/49 [==============================] - 25s 505ms/step - loss: 0.6931 -
accuracy: 0.8359
Epoch 5/5
49/49 [==============================] - 25s 505ms/step - loss: 0.6970 -
accuracy: 0.8323
--- Increment 3 on a subsampled dataset ---
Epoch 1/5
49/49 [==============================] - 25s 508ms/step - loss: 0.6789 -
accuracy: 0.8420
Epoch 2/5
49/49 [==============================] - 25s 506ms/step - loss: 0.6873 -
accuracy: 0.8394
Epoch 3/5
49/49 [==============================] - 25s 505ms/step - loss: 0.6716 -
accuracy: 0.8442
Epoch 4/5
49/49 [==============================] - 25s 504ms/step - loss: 0.6740 -
```

```
accuracy: 0.8416
Epoch 5/5
49/49 [==============================] - 25s 513ms/step - loss: 0.6672 -
accuracy: 0.8471
--- Increment 4 on a subsampled dataset ---
Epoch 1/5
49/49 [==============================] - 25s 504ms/step - loss: 0.6645 -
accuracy: 0.8464
Epoch 2/5
49/49 [==============================] - 25s 506ms/step - loss: 0.6582 -
accuracy: 0.8509
Epoch 3/5
49/49 [==============================] - 25s 514ms/step - loss: 0.6626 -
accuracy: 0.8485
Epoch 4/5
49/49 [==============================] - 25s 505ms/step - loss: 0.6421 -
accuracy: 0.8554
Epoch 5/5
49/49 [==============================] - 25s 506ms/step - loss: 0.6404 -
accuracy: 0.8585
--- Increment 5 on a subsampled dataset ---
Epoch 1/5
49/49 [==============================] - 25s 502ms/step - loss: 0.6337 -
accuracy: 0.8597
Epoch 2/5
49/49 [==============================] - 25s 503ms/step - loss: 0.6375 -
accuracy: 0.8573
Epoch 3/5
49/49 [==============================] - 25s 504ms/step - loss: 0.6203 -
accuracy: 0.8650
Epoch 4/5
49/49 [==============================] - 25s 503ms/step - loss: 0.6113 -
accuracy: 0.8686
Epoch 5/5
49/49 [==============================] - 25s 514ms/step - loss: 0.6173 -
accuracy: 0.8641
--- Increment 6 on a subsampled dataset ---
Epoch 1/5
49/49 [==============================] - 25s 502ms/step - loss: 0.6276 -
accuracy: 0.8632
Epoch 2/5
49/49 [==============================] - 25s 503ms/step - loss: 0.6097 -
accuracy: 0.8687
Epoch 3/5
49/49 [==============================] - 25s 503ms/step - loss: 0.6020 -
accuracy: 0.8696
Epoch 4/5
49/49 [==============================] - 25s 505ms/step - loss: 0.6085 -
accuracy: 0.8700
Epoch 5/5
49/49 [==============================] - 25s 506ms/step - loss: 0.5926 -
accuracy: 0.8748
--- Increment 7 on a subsampled dataset ---
Epoch 1/5
49/49 [==============================] - 25s 502ms/step - loss: 0.5897 -
accuracy: 0.8769
Epoch 2/5
49/49 [==============================] - 25s 505ms/step - loss: 0.5892 -
accuracy: 0.8745
Epoch 3/5
```

```
49/49 [==============================] - 25s 502ms/step - loss: 0.5955 -
accuracy: 0.8731
Epoch 4/5
49/49 [==============================] - 25s 506ms/step - loss: 0.5849 -
accuracy: 0.8789
Epoch 5/5
49/49 [==============================] - 25s 507ms/step - loss: 0.5848 -
accuracy: 0.8784
--- Increment 8 on a subsampled dataset ---
Epoch 1/5
49/49 [==============================] - 25s 505ms/step - loss: 0.5867 -
accuracy: 0.8780
Epoch 2/5
49/49 [==============================] - 25s 503ms/step - loss: 0.5851 -
accuracy: 0.8790
Epoch 3/5
49/49 [==============================] - 25s 503ms/step - loss: 0.5826 -
accuracy: 0.8797
Epoch 4/5
49/49 [==============================] - 25s 505ms/step - loss: 0.5602 -
accuracy: 0.8872
Epoch 5/5
49/49 [==============================] - 25s 505ms/step - loss: 0.5704 -
accuracy: 0.8858
--- Increment 9 on a subsampled dataset ---
Epoch 1/5
49/49 [==============================] - 25s 503ms/step - loss: 0.5719 -
accuracy: 0.8815
Epoch 2/5
49/49 [==============================] - 25s 506ms/step - loss: 0.5659 -
accuracy: 0.8841
Epoch 3/5
49/49 [==============================] - 25s 508ms/step - loss: 0.5712 -
accuracy: 0.8835
Epoch 4/5
49/49 [==============================] - 25s 516ms/step - loss: 0.5602 -
accuracy: 0.8883
Epoch 5/5
49/49 [==============================] - 27s 540ms/step - loss: 0.5560 -
accuracy: 0.8890
313/313 [==============================] - 5s 16ms/step
Accuracy: 0.82
Precision: 0.8200000000000001
Recall: 0.8220293437959733
F1: 0.8194779889377048
```



TODO 3.5.4 Perform an experiment, during which models are trained using:

- 10% of data (done)
- 25% of data
- 50% of data
- 75% of data
- 100% of data (done)

Display plots showing differences between the classification scores

We will use only 25% of data for training

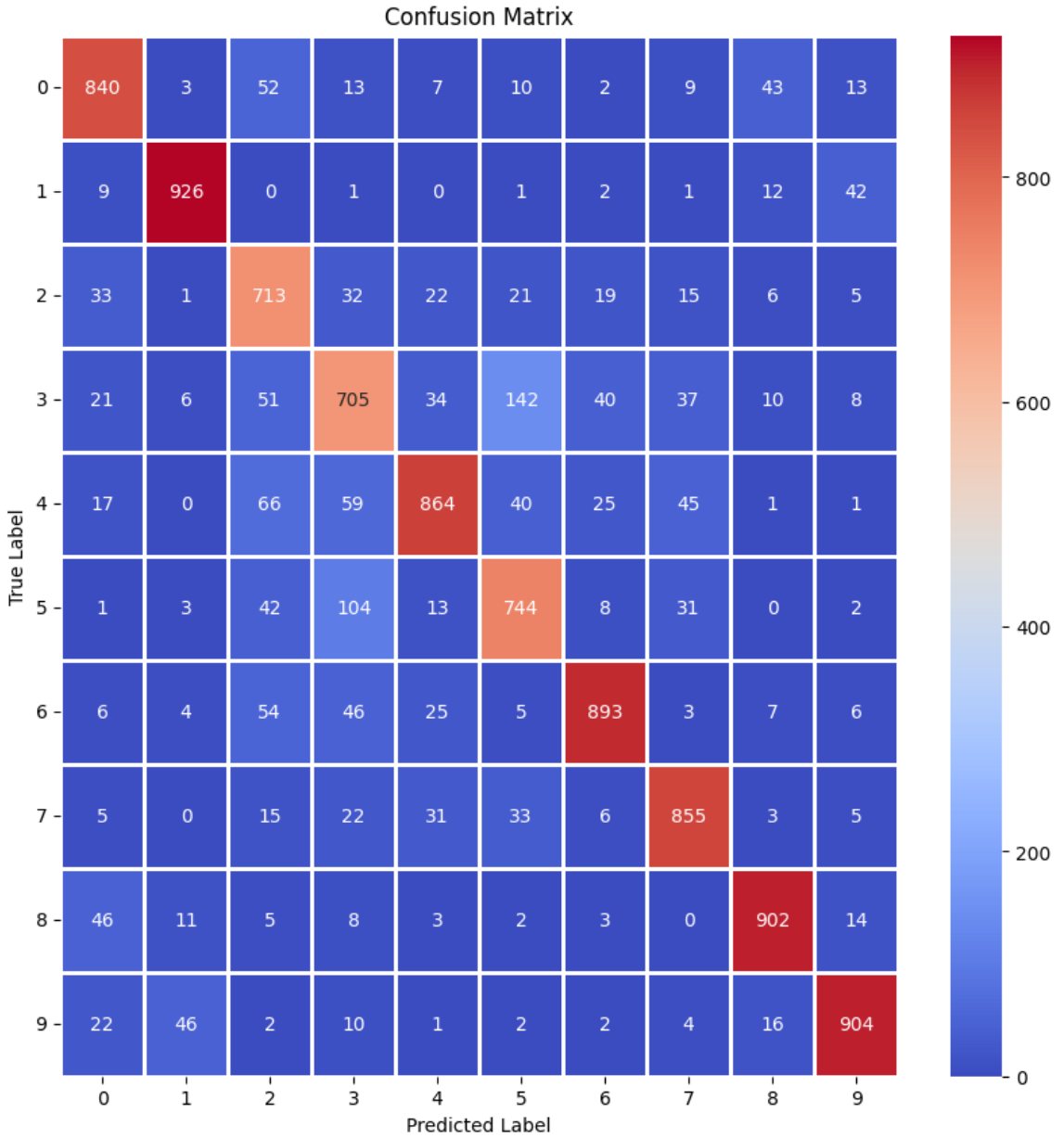
In [77]:
```
probs = np.random.rand(y_train.shape[0])

X_train_sample = X_train[probs<0.25]
y_train_sample =y_train[probs<0.25]

print(X_train_sample.shape)

for i in range(10):
```

```
  print(f'--- Increment {i} on a subsampled dataset ---')
  model3.fit(X_train_sample, y_train_sample,batch_size=batch_size,epochs=
  path=f'models/CNN_CIFAR-10_sub{i}.h5'
  model3.save(path)

preds = model3.predict(X_test)
y_pred = np.argmax(preds,axis=1)

# print scores
print('Accuracy:', accuracy_score(y_pred,y_test))
print('Precision:', precision_score(y_pred,y_test,average='macro'))
print('Recall:', recall_score(y_pred,y_test,average='macro'))
print('F1:', f1_score(y_pred,y_test,average='macro'))

# display confusion matrix
show_confusion_matrix(confusion_matrix(y_pred,y_test))
```

```
(12521, 32, 32, 3)
--- Increment 0 on a subsampled dataset ---
Epoch 1/5
25/25 [==============================] - 13s 520ms/step - loss: 0.6231 -
accuracy: 0.8656
Epoch 2/5
25/25 [==============================] - 13s 511ms/step - loss: 0.5919 -
accuracy: 0.8721
Epoch 3/5
25/25 [==============================] - 13s 514ms/step - loss: 0.5724 -
accuracy: 0.8812
Epoch 4/5
25/25 [==============================] - 13s 518ms/step - loss: 0.5681 -
accuracy: 0.8809
Epoch 5/5
25/25 [==============================] - 14s 568ms/step - loss: 0.5350 -
accuracy: 0.8919
--- Increment 1 on a subsampled dataset ---
Epoch 1/5
25/25 [==============================] - 13s 528ms/step - loss: 0.5153 -
accuracy: 0.8993
Epoch 2/5
25/25 [==============================] - 14s 552ms/step - loss: 0.5076 -
accuracy: 0.9026
Epoch 3/5
25/25 [==============================] - 14s 567ms/step - loss: 0.5118 -
accuracy: 0.9014
Epoch 4/5
25/25 [==============================] - 14s 546ms/step - loss: 0.5002 -
accuracy: 0.9066
Epoch 5/5
25/25 [==============================] - 13s 518ms/step - loss: 0.4972 -
accuracy: 0.9028
--- Increment 2 on a subsampled dataset ---
Epoch 1/5
25/25 [==============================] - 13s 524ms/step - loss: 0.4934 -
accuracy: 0.9082
Epoch 2/5
25/25 [==============================] - 14s 542ms/step - loss: 0.4821 -
accuracy: 0.9132
Epoch 3/5
25/25 [==============================] - 13s 531ms/step - loss: 0.4687 -
accuracy: 0.9166
Epoch 4/5
25/25 [==============================] - 15s 588ms/step - loss: 0.4720 -
accuracy: 0.9149
Epoch 5/5
25/25 [==============================] - 14s 579ms/step - loss: 0.4706 -
accuracy: 0.9139
--- Increment 3 on a subsampled dataset ---
Epoch 1/5
25/25 [==============================] - 14s 544ms/step - loss: 0.4751 -
accuracy: 0.9153
Epoch 2/5
25/25 [==============================] - 14s 541ms/step - loss: 0.4727 -
accuracy: 0.9182
Epoch 3/5
25/25 [==============================] - 13s 527ms/step - loss: 0.4618 -
accuracy: 0.9221
Epoch 4/5
```

```
25/25 [==============================] - 13s 508ms/step - loss: 0.4593 -
accuracy: 0.9228
Epoch 5/5
25/25 [==============================] - 12s 494ms/step - loss: 0.4462 -
accuracy: 0.9269
--- Increment 4 on a subsampled dataset ---
Epoch 1/5
25/25 [==============================] - 12s 495ms/step - loss: 0.4531 -
accuracy: 0.9234
Epoch 2/5
25/25 [==============================] - 13s 508ms/step - loss: 0.4505 -
accuracy: 0.9275
Epoch 3/5
25/25 [==============================] - 14s 547ms/step - loss: 0.4623 -
accuracy: 0.9221
Epoch 4/5
25/25 [==============================] - 13s 524ms/step - loss: 0.4502 -
accuracy: 0.9292
Epoch 5/5
25/25 [==============================] - 13s 512ms/step - loss: 0.4645 -
accuracy: 0.9238
--- Increment 5 on a subsampled dataset ---
Epoch 1/5
25/25 [==============================] - 13s 501ms/step - loss: 0.4536 -
accuracy: 0.9283
Epoch 2/5
25/25 [==============================] - 12s 491ms/step - loss: 0.4429 -
accuracy: 0.9302
Epoch 3/5
25/25 [==============================] - 12s 489ms/step - loss: 0.4566 -
accuracy: 0.9228
Epoch 4/5
25/25 [==============================] - 13s 513ms/step - loss: 0.4442 -
accuracy: 0.9300
Epoch 5/5
25/25 [==============================] - 12s 498ms/step - loss: 0.4458 -
accuracy: 0.9282
--- Increment 6 on a subsampled dataset ---
Epoch 1/5
25/25 [==============================] - 12s 497ms/step - loss: 0.4326 -
accuracy: 0.9331
Epoch 2/5
25/25 [==============================] - 12s 494ms/step - loss: 0.4243 -
accuracy: 0.9372
Epoch 3/5
25/25 [==============================] - 12s 494ms/step - loss: 0.4343 -
accuracy: 0.9328
Epoch 4/5
25/25 [==============================] - 13s 510ms/step - loss: 0.4344 -
accuracy: 0.9336
Epoch 5/5
25/25 [==============================] - 12s 490ms/step - loss: 0.4335 -
accuracy: 0.9320
--- Increment 7 on a subsampled dataset ---
Epoch 1/5
25/25 [==============================] - 12s 494ms/step - loss: 0.4392 -
accuracy: 0.9316
Epoch 2/5
25/25 [==============================] - 12s 494ms/step - loss: 0.4329 -
accuracy: 0.9326
```

```
Epoch 3/5
25/25 [==============================] - 12s 491ms/step - loss: 0.4322 -
accuracy: 0.9337
Epoch 4/5
25/25 [==============================] - 12s 492ms/step - loss: 0.4299 -
accuracy: 0.9347
Epoch 5/5
25/25 [==============================] - 12s 490ms/step - loss: 0.4132 -
accuracy: 0.9397
--- Increment 8 on a subsampled dataset ---
Epoch 1/5
25/25 [==============================] - 12s 492ms/step - loss: 0.4167 -
accuracy: 0.9366
Epoch 2/5
25/25 [==============================] - 12s 495ms/step - loss: 0.4349 -
accuracy: 0.9336
Epoch 3/5
25/25 [==============================] - 13s 505ms/step - loss: 0.4155 -
accuracy: 0.9371
Epoch 4/5
25/25 [==============================] - 13s 500ms/step - loss: 0.4248 -
accuracy: 0.9375
Epoch 5/5
25/25 [==============================] - 13s 504ms/step - loss: 0.4296 -
accuracy: 0.9359
--- Increment 9 on a subsampled dataset ---
Epoch 1/5
25/25 [==============================] - 12s 497ms/step - loss: 0.4153 -
accuracy: 0.9385
Epoch 2/5
25/25 [==============================] - 12s 499ms/step - loss: 0.4232 -
accuracy: 0.9368
Epoch 3/5
25/25 [==============================] - 12s 498ms/step - loss: 0.4164 -
accuracy: 0.9383
Epoch 4/5
25/25 [==============================] - 12s 498ms/step - loss: 0.4135 -
accuracy: 0.9405
Epoch 5/5
25/25 [==============================] - 13s 512ms/step - loss: 0.4106 -
accuracy: 0.9408
313/313 [==============================] - 4s 14ms/step
Accuracy: 0.8346
Precision: 0.8346
Recall: 0.8358830164487439
F1: 0.8344273714558081
```



We will use only 50% of data for training

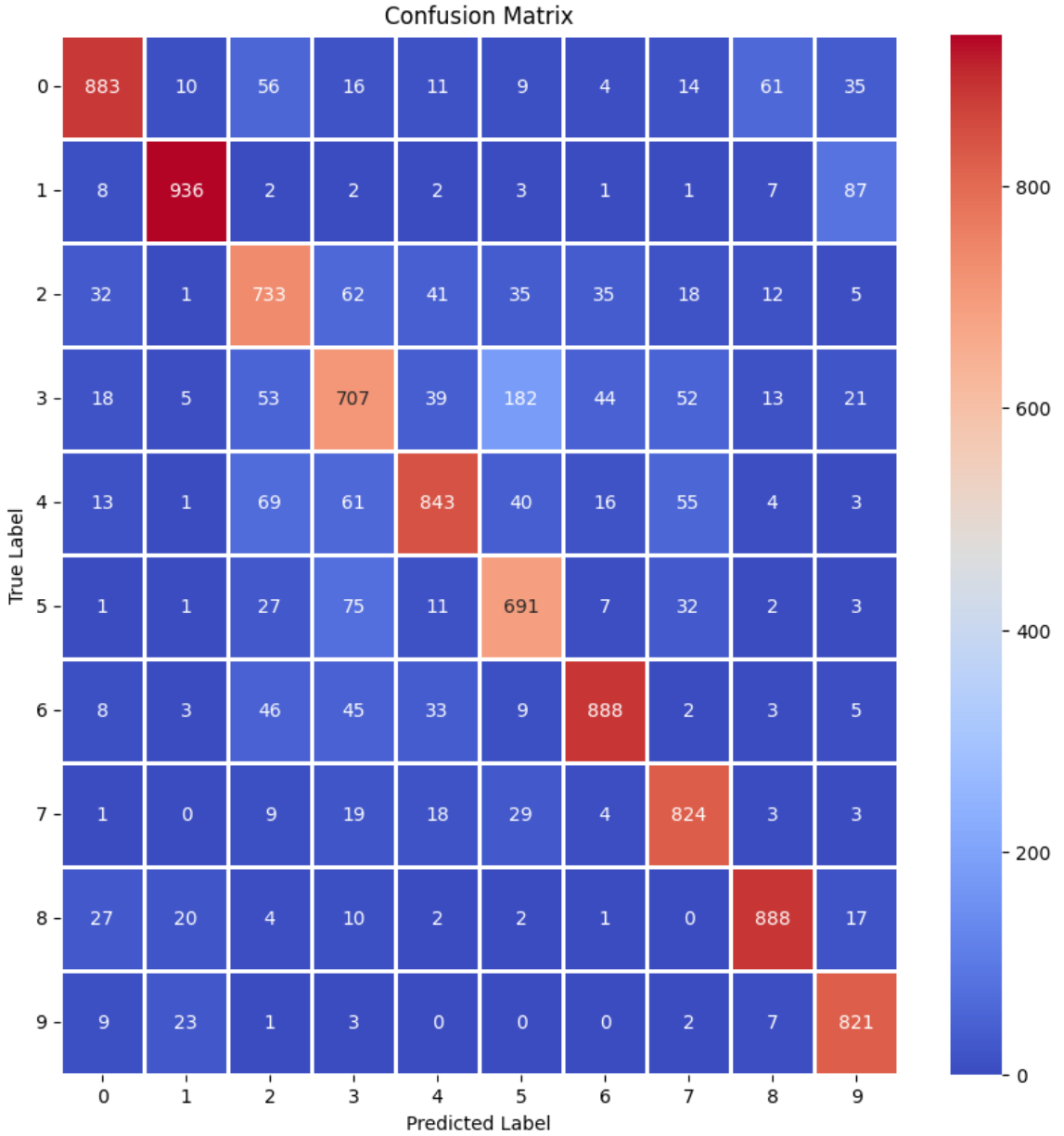
In [75]:
```
probs = np.random.rand(y_train.shape[0])

X_train_sample = X_train[probs<0.5]
y_train_sample =y_train[probs<0.5]

print(X_train_sample.shape)

for i in range(10):
  print(f'--- Increment {i} on a subsampled dataset ---')
  model3.fit(X_train_sample, y_train_sample,batch_size=batch_size,epochs=
  path=f'models/CNN_CIFAR-10_sub{i}.h5'
  model3.save(path)

preds = model3.predict(X_test)
y_pred = np.argmax(preds,axis=1)

# print scores
print('Accuracy:', accuracy_score(y_pred,y_test))
print('Precision:', precision_score(y_pred,y_test,average='macro'))
print('Recall:', recall_score(y_pred,y_test,average='macro'))
```

```
print('F1:', f1_score(y_pred,y_test,average='macro'))

# display confusion matrix
show_confusion_matrix(confusion_matrix(y_pred,y_test))
```

```
(25092, 32, 32, 3)
--- Increment 0 on a subsampled dataset ---
Epoch 1/5
50/50 [==============================] - 26s 527ms/step - loss: 0.9503 -
accuracy: 0.7719
Epoch 2/5
50/50 [==============================] - 25s 510ms/step - loss: 0.8529 -
accuracy: 0.7962
Epoch 3/5
50/50 [==============================] - 26s 513ms/step - loss: 0.8234 -
accuracy: 0.7998
Epoch 4/5
50/50 [==============================] - 26s 513ms/step - loss: 0.8814 -
accuracy: 0.7809
Epoch 5/5
50/50 [==============================] - 26s 512ms/step - loss: 0.7767 -
accuracy: 0.8181
--- Increment 1 on a subsampled dataset ---
Epoch 1/5
50/50 [==============================] - 25s 508ms/step - loss: 0.7820 -
accuracy: 0.8123
Epoch 2/5
50/50 [==============================] - 26s 515ms/step - loss: 0.7469 -
accuracy: 0.8245
Epoch 3/5
50/50 [==============================] - 28s 558ms/step - loss: 0.7181 -
accuracy: 0.8297
Epoch 4/5
50/50 [==============================] - 27s 535ms/step - loss: 0.7142 -
accuracy: 0.8300
Epoch 5/5
50/50 [==============================] - 26s 521ms/step - loss: 0.7569 -
accuracy: 0.8175
--- Increment 2 on a subsampled dataset ---
Epoch 1/5
50/50 [==============================] - 26s 516ms/step - loss: 0.7842 -
accuracy: 0.8099
Epoch 2/5
50/50 [==============================] - 26s 514ms/step - loss: 0.7095 -
accuracy: 0.8361
Epoch 3/5
50/50 [==============================] - 26s 514ms/step - loss: 0.6857 -
accuracy: 0.8447
Epoch 4/5
50/50 [==============================] - 26s 530ms/step - loss: 0.6788 -
accuracy: 0.8452
Epoch 5/5
50/50 [==============================] - 26s 511ms/step - loss: 0.6715 -
accuracy: 0.8488
--- Increment 3 on a subsampled dataset ---
Epoch 1/5
50/50 [==============================] - 25s 509ms/step - loss: 0.6594 -
accuracy: 0.8531
Epoch 2/5
50/50 [==============================] - 26s 527ms/step - loss: 0.6636 -
accuracy: 0.8498
Epoch 3/5
50/50 [==============================] - 25s 509ms/step - loss: 0.6631 -
accuracy: 0.8479
Epoch 4/5
```

```
50/50 [==============================] - 25s 509ms/step - loss: 0.6494 -
accuracy: 0.8554
Epoch 5/5
50/50 [==============================] - 26s 511ms/step - loss: 0.6485 -
accuracy: 0.8540
--- Increment 4 on a subsampled dataset ---
Epoch 1/5
50/50 [==============================] - 26s 510ms/step - loss: 0.6435 -
accuracy: 0.8585
Epoch 2/5
50/50 [==============================] - 26s 516ms/step - loss: 0.6352 -
accuracy: 0.8583
Epoch 3/5
50/50 [==============================] - 25s 510ms/step - loss: 0.6366 -
accuracy: 0.8621
Epoch 4/5
50/50 [==============================] - 25s 509ms/step - loss: 0.6316 -
accuracy: 0.8631
Epoch 5/5
50/50 [==============================] - 25s 509ms/step - loss: 0.6236 -
accuracy: 0.8636
--- Increment 5 on a subsampled dataset ---
Epoch 1/5
50/50 [==============================] - 26s 522ms/step - loss: 0.6232 -
accuracy: 0.8649
Epoch 2/5
50/50 [==============================] - 26s 524ms/step - loss: 0.6268 -
accuracy: 0.8649
Epoch 3/5
50/50 [==============================] - 25s 508ms/step - loss: 0.6102 -
accuracy: 0.8676
Epoch 4/5
50/50 [==============================] - 26s 512ms/step - loss: 0.6531 -
accuracy: 0.8551
Epoch 5/5
50/50 [==============================] - 27s 544ms/step - loss: 0.6437 -
accuracy: 0.8619
--- Increment 6 on a subsampled dataset ---
Epoch 1/5
50/50 [==============================] - 26s 520ms/step - loss: 0.6330 -
accuracy: 0.8643
Epoch 2/5
50/50 [==============================] - 26s 517ms/step - loss: 0.6093 -
accuracy: 0.8729
Epoch 3/5
50/50 [==============================] - 26s 518ms/step - loss: 0.6095 -
accuracy: 0.8733
Epoch 4/5
50/50 [==============================] - 26s 520ms/step - loss: 0.5969 -
accuracy: 0.8754
Epoch 5/5
50/50 [==============================] - 26s 520ms/step - loss: 0.6031 -
accuracy: 0.8726
--- Increment 7 on a subsampled dataset ---
Epoch 1/5
50/50 [==============================] - 26s 518ms/step - loss: 0.5946 -
accuracy: 0.8780
Epoch 2/5
50/50 [==============================] - 26s 517ms/step - loss: 0.5962 -
accuracy: 0.8767
```

```
Epoch 3/5
50/50 [==============================] - 26s 520ms/step - loss: 0.6167 -
accuracy: 0.8687
Epoch 4/5
50/50 [==============================] - 26s 515ms/step - loss: 0.5844 -
accuracy: 0.8805
Epoch 5/5
50/50 [==============================] - 26s 519ms/step - loss: 0.5914 -
accuracy: 0.8784
--- Increment 8 on a subsampled dataset ---
Epoch 1/5
50/50 [==============================] - 26s 517ms/step - loss: 0.5843 -
accuracy: 0.8818
Epoch 2/5
50/50 [==============================] - 26s 517ms/step - loss: 0.5758 -
accuracy: 0.8822
Epoch 3/5
50/50 [==============================] - 26s 518ms/step - loss: 0.5712 -
accuracy: 0.8845
Epoch 4/5
50/50 [==============================] - 26s 513ms/step - loss: 0.5937 -
accuracy: 0.8755
Epoch 5/5
50/50 [==============================] - 27s 539ms/step - loss: 0.5768 -
accuracy: 0.8824
--- Increment 9 on a subsampled dataset ---
Epoch 1/5
50/50 [==============================] - 26s 523ms/step - loss: 0.5662 -
accuracy: 0.8861
Epoch 2/5
50/50 [==============================] - 26s 510ms/step - loss: 0.5629 -
accuracy: 0.8873
Epoch 3/5
50/50 [==============================] - 26s 510ms/step - loss: 0.5705 -
accuracy: 0.8845
Epoch 4/5
50/50 [==============================] - 25s 509ms/step - loss: 0.5656 -
accuracy: 0.8843
Epoch 5/5
50/50 [==============================] - 26s 527ms/step - loss: 0.5610 -
accuracy: 0.8859
313/313 [==============================] - 5s 15ms/step
Accuracy: 0.8214
Precision: 0.8213999999999999
Recall: 0.8267856541377077
F1: 0.8221791868764443
```



We will use only 75% of data for training

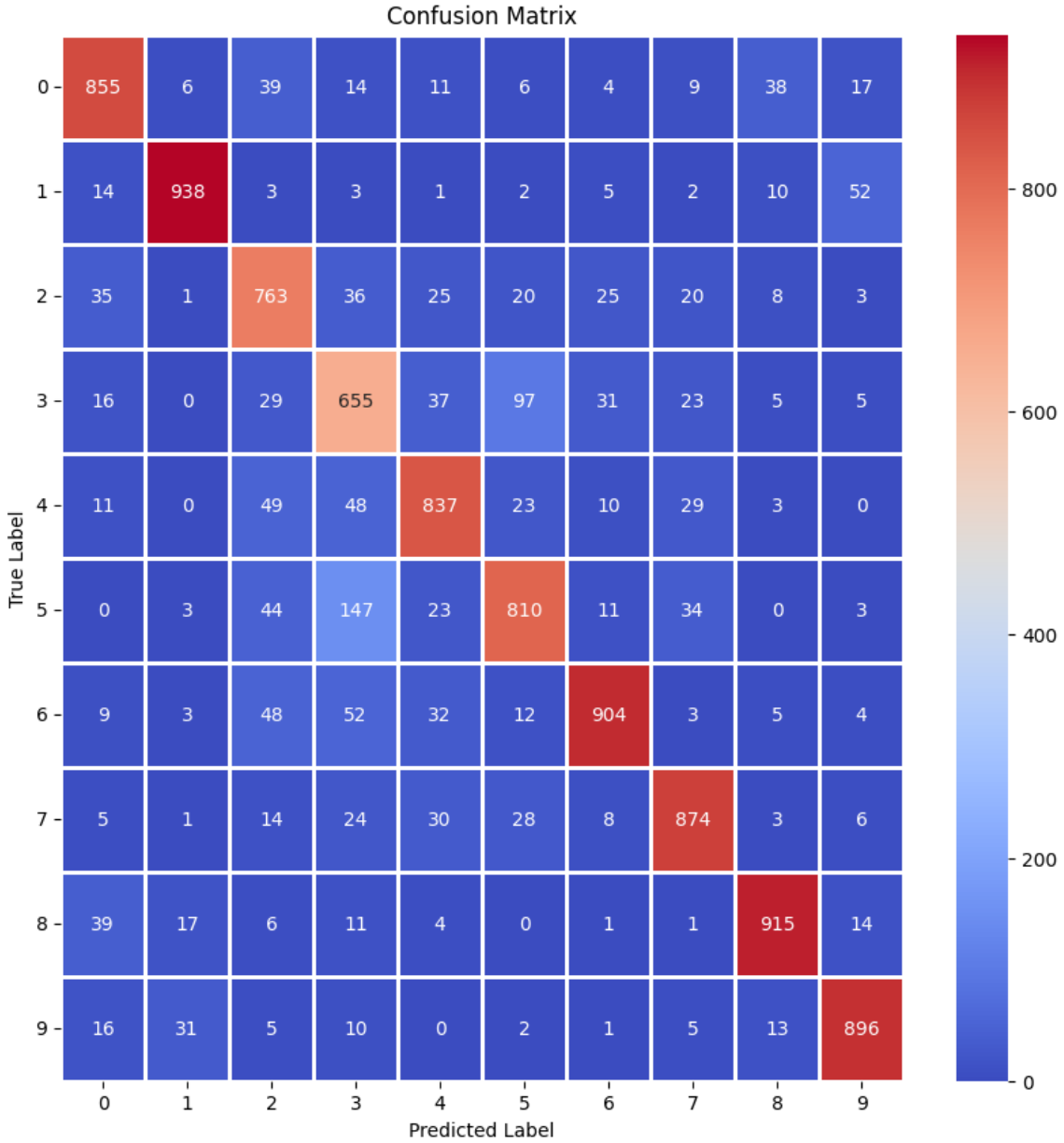
In [76]:
```
probs = np.random.rand(y_train.shape[0])

X_train_sample = X_train[probs<0.75]
y_train_sample =y_train[probs<0.75]

print(X_train_sample.shape)

for i in range(10):
  print(f'--- Increment {i} on a subsampled dataset ---')
  model3.fit(X_train_sample, y_train_sample,batch_size=batch_size,epochs=
  path=f'models/CNN_CIFAR-10_sub{i}.h5'
  model3.save(path)

preds = model3.predict(X_test)
y_pred = np.argmax(preds,axis=1)

# print scores
print('Accuracy:', accuracy_score(y_pred,y_test))
print('Precision:', precision_score(y_pred,y_test,average='macro'))
print('Recall:', recall_score(y_pred,y_test,average='macro'))
```

```
print('F1:', f1_score(y_pred,y_test,average='macro'))

# display confusion matrix
show_confusion_matrix(confusion_matrix(y_pred,y_test))
```

```
(37593, 32, 32, 3)
--- Increment 0 on a subsampled dataset ---
Epoch 1/5
74/74 [==============================] - 40s 533ms/step - loss: 0.7558 -
accuracy: 0.8294
Epoch 2/5
74/74 [==============================] - 38s 520ms/step - loss: 0.7121 -
accuracy: 0.8366
Epoch 3/5
74/74 [==============================] - 38s 520ms/step - loss: 0.6904 -
accuracy: 0.8427
Epoch 4/5
74/74 [==============================] - 40s 536ms/step - loss: 0.6822 -
accuracy: 0.8422
Epoch 5/5
74/74 [==============================] - 40s 534ms/step - loss: 0.6695 -
accuracy: 0.8477
--- Increment 1 on a subsampled dataset ---
Epoch 1/5
74/74 [==============================] - 41s 553ms/step - loss: 0.6610 -
accuracy: 0.8475
Epoch 2/5
74/74 [==============================] - 39s 520ms/step - loss: 0.6448 -
accuracy: 0.8515
Epoch 3/5
74/74 [==============================] - 39s 533ms/step - loss: 0.6444 -
accuracy: 0.8543
Epoch 4/5
74/74 [==============================] - 39s 529ms/step - loss: 0.6435 -
accuracy: 0.8519
Epoch 5/5
74/74 [==============================] - 38s 518ms/step - loss: 0.6407 -
accuracy: 0.8562
--- Increment 2 on a subsampled dataset ---
Epoch 1/5
74/74 [==============================] - 38s 515ms/step - loss: 0.6317 -
accuracy: 0.8585
Epoch 2/5
74/74 [==============================] - 38s 517ms/step - loss: 0.6230 -
accuracy: 0.8591
Epoch 3/5
74/74 [==============================] - 40s 535ms/step - loss: 0.6198 -
accuracy: 0.8627
Epoch 4/5
74/74 [==============================] - 38s 515ms/step - loss: 0.6210 -
accuracy: 0.8617
Epoch 5/5
74/74 [==============================] - 38s 515ms/step - loss: 0.6216 -
accuracy: 0.8617
--- Increment 3 on a subsampled dataset ---
Epoch 1/5
74/74 [==============================] - 38s 514ms/step - loss: 0.6155 -
accuracy: 0.8620
Epoch 2/5
74/74 [==============================] - 38s 513ms/step - loss: 0.6270 -
accuracy: 0.8594
Epoch 3/5
74/74 [==============================] - 38s 516ms/step - loss: 0.6149 -
accuracy: 0.8644
Epoch 4/5
```

```
74/74 [==============================] - 38s 516ms/step - loss: 0.6111 -
accuracy: 0.8649
Epoch 5/5
74/74 [==============================] - 38s 516ms/step - loss: 0.6104 -
accuracy: 0.8670
--- Increment 4 on a subsampled dataset ---
Epoch 1/5
74/74 [==============================] - 38s 512ms/step - loss: 0.6048 -
accuracy: 0.8678
Epoch 2/5
74/74 [==============================] - 38s 512ms/step - loss: 0.6028 -
accuracy: 0.8666
Epoch 3/5
74/74 [==============================] - 39s 530ms/step - loss: 0.5970 -
accuracy: 0.8702
Epoch 4/5
74/74 [==============================] - 39s 523ms/step - loss: 0.5966 -
accuracy: 0.8707
Epoch 5/5
74/74 [==============================] - 39s 524ms/step - loss: 0.5933 -
accuracy: 0.8713
--- Increment 5 on a subsampled dataset ---
Epoch 1/5
74/74 [==============================] - 38s 516ms/step - loss: 0.5860 -
accuracy: 0.8739
Epoch 2/5
74/74 [==============================] - 38s 517ms/step - loss: 0.5948 -
accuracy: 0.8702
Epoch 3/5
74/74 [==============================] - 40s 543ms/step - loss: 0.5898 -
accuracy: 0.8706
Epoch 4/5
74/74 [==============================] - 39s 531ms/step - loss: 0.5831 -
accuracy: 0.8752
Epoch 5/5
74/74 [==============================] - 39s 522ms/step - loss: 0.5768 -
accuracy: 0.8748
--- Increment 6 on a subsampled dataset ---
Epoch 1/5
74/74 [==============================] - 38s 517ms/step - loss: 0.5810 -
accuracy: 0.8747
Epoch 2/5
74/74 [==============================] - 38s 515ms/step - loss: 0.5784 -
accuracy: 0.8757
Epoch 3/5
74/74 [==============================] - 38s 511ms/step - loss: 0.5776 -
accuracy: 0.8744
Epoch 4/5
74/74 [==============================] - 38s 512ms/step - loss: 0.5751 -
accuracy: 0.8776
Epoch 5/5
74/74 [==============================] - 38s 513ms/step - loss: 0.5716 -
accuracy: 0.8764
--- Increment 7 on a subsampled dataset ---
Epoch 1/5
74/74 [==============================] - 38s 515ms/step - loss: 0.5745 -
accuracy: 0.8764
Epoch 2/5
74/74 [==============================] - 38s 513ms/step - loss: 0.5694 -
accuracy: 0.8797
```

```
Epoch 3/5
74/74 [==============================] - 38s 515ms/step - loss: 0.5675 -
accuracy: 0.8803
Epoch 4/5
74/74 [==============================] - 38s 516ms/step - loss: 0.5659 -
accuracy: 0.8818
Epoch 5/5
74/74 [==============================] - 38s 516ms/step - loss: 0.5716 -
accuracy: 0.8784
--- Increment 8 on a subsampled dataset ---
Epoch 1/5
74/74 [==============================] - 38s 514ms/step - loss: 0.5636 -
accuracy: 0.8794
Epoch 2/5
74/74 [==============================] - 38s 515ms/step - loss: 0.5516 -
accuracy: 0.8820
Epoch 3/5
74/74 [==============================] - 40s 537ms/step - loss: 0.5601 -
accuracy: 0.8816
Epoch 4/5
74/74 [==============================] - 41s 555ms/step - loss: 0.5588 -
accuracy: 0.8833
Epoch 5/5
74/74 [==============================] - 41s 555ms/step - loss: 0.5573 -
accuracy: 0.8835
--- Increment 9 on a subsampled dataset ---
Epoch 1/5
74/74 [==============================] - 43s 574ms/step - loss: 0.5619 -
accuracy: 0.8806
Epoch 2/5
74/74 [==============================] - 43s 585ms/step - loss: 0.5512 -
accuracy: 0.8847
Epoch 3/5
74/74 [==============================] - 41s 550ms/step - loss: 0.5516 -
accuracy: 0.8854
Epoch 4/5
74/74 [==============================] - 38s 520ms/step - loss: 0.5518 -
accuracy: 0.8831
Epoch 5/5
74/74 [==============================] - 39s 521ms/step - loss: 0.5464 -
accuracy: 0.8890
313/313 [==============================] - 5s 16ms/step
Accuracy: 0.8447
Precision: 0.8447000000000001
Recall: 0.8439708731826279
F1: 0.8438066739200272
```



ACCURACY: Results for 10% : 0.6396 Results for 25% : 0.9431 Results for 50% : 0.8214 Results for 75% : 0.8447 Results for 100%: 0.7473