In [1]:

```python
#Imports needed packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns


from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import itertools

from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
```

```
2024-02-23 01:07:15.911790: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable
to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already bee
n registered
2024-02-23 01:07:15.911943: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable t
o register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been
registered
2024-02-23 01:07:16.088922: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable
to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already b
een registered
```

In [2]:

```python
#Creates a configuration class with the model parameters
class CFG:
    verbose=1
    seed = 42
    classes = 10
    shape = (-1,28,28,1)
    epochs = 25
    batch_size = 128
    patience=3
```

In [3]:

```python
#Loads the data
df_train = pd.read_csv("/kaggle/input/digit-recognizer/train.csv")
x_test = pd.read_csv("/kaggle/input/digit-recognizer/test.csv")
```

In [4]:

```python
print("df_train:", df_train.shape)
print("df_test:", x_test.shape)
df_train.head()
```

```
df_train: (42000, 785)
df_test: (28000, 784)
```

Out[4]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel7
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	

5 rows × 785 columns

In [5]:

```
#Checks for any null values in the data frames
print(df_train.isnull().sum().sum())
print(x_test.isnull().sum().sum())
```

```
0
0
```

In [6]:

```
#Creates the X and Y traind data sets
y_train = df_train["label"]
x_train = df_train.drop(columns=['label'])
#gives count of each digit label in the training set
y_train.value_counts()
```

Out[6]:

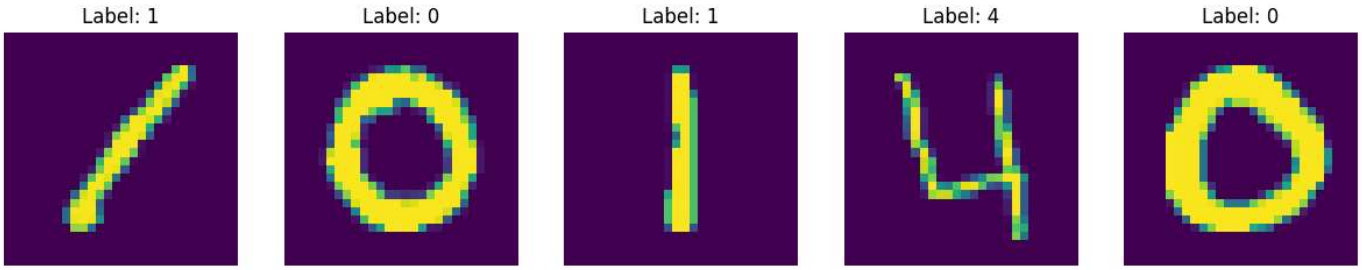
```
label
1    4684
7    4401
3    4351
9    4188
2    4177
6    4137
0    4132
4    4072
8    4063
5    3795
Name: count, dtype: int64
```

In [7]:

```
#Normalizes the data by dividing by 255 so no pixel is larger than 1
x_train = x_train / 255.0
x_test = x_test / 255.0

#Reshapes the images to 28x28x1
x_train = x_train.values.reshape(CFG.shape)
x_test = x_test.values.reshape(CFG.shape)

# Encodes the labels to one hot vectors
oh_y_train = to_categorical(y_train, CFG.classes)
```

In [8]:

```
#show sthe first five images and labels in the training DS
fig, axes = plt.subplots(nrows=1, ncols=5, figsize=(15, 3))

for i in range(5):
    axes[i].imshow(x_train[i][:,:,0])
    axes[i].axis('off')
    label = y_train.iloc[i]
    axes[i].set_title(f'Label: {label}')
```



In [9]:

```
x_train, x_val, y_train, y_val = train_test_split(x_train, oh_y_train, test_size = 0.1, random_state
```

In [10]:

```
#augments the data to create additional images to train with.
data_generator = ImageDataGenerator(
        rotation_range=15,
        zoom_range = 0.15,
        width_shift_range=0.15,
        height_shift_range=0.15,
        horizontal_flip=False,
        vertical_flip=False)


data_generator.fit(x_train)
```

In [11]:

```
#Creates the CNN model
model = Sequential()

#Creates "conv2d", "maxpool", and "dropout", and layers
model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',
                 activation ='relu'))
model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.15))


model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.15))

model.add(Conv2D(filters = 128, kernel_size = (6,6),padding = 'Same',
                 activation ='relu'))
model.add(Conv2D(filters = 128, kernel_size = (6,6),padding = 'Same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.15))


model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(10, activation = "softmax"))
```

In [12]:

```
#Compiles the model and sets the optimizer to "adam"
model.compile(optimizer = "adam" , loss = "categorical_crossentropy", metrics=["accuracy"])
```

In [13]:

```
#Sets a learning rate annealer
lr_reduction = ReduceLROnPlateau(monitor='val_acc', patience=CFG.patience, verbose=CFG.verbose,
                                 factor=0.5, min_lr=0.00001)
```

In [14]:

```
history = model.fit(data_generator.flow(x_train,y_train, batch_size=CFG.batch_size),
                    epochs = CFG.epochs, validation_data = (x_val,y_val),verbose = CFG.verbose,
                    steps_per_epoch=x_train.shape[0] // CFG.batch_size,callbacks=[lr_reduction])
```

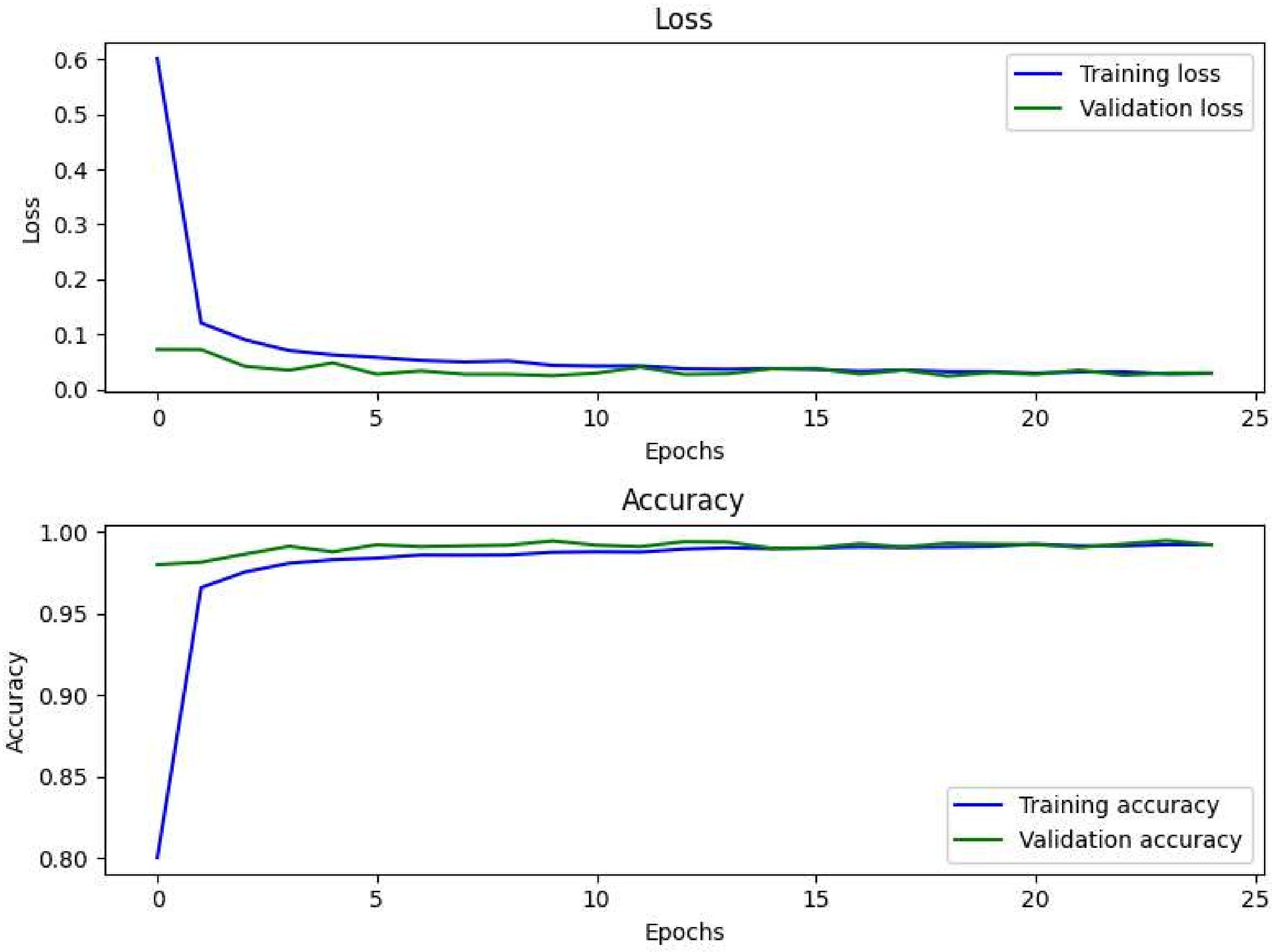
```
Epoch 1/25

2024-02-23 01:07:40.160680: E tensorflow/core/grappler/optimizers/meta_optimizer.cc:961] layout fai
led: INVALID_ARGUMENT: Size of values 0 does not match size of permutation 4 @ fanin shape insequen
tial/dropout/dropout/SelectV2-2-TransposeNHWCToNCHW-LayoutOptimizer
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1708650462.131969      72 device_compiler.h:186] Compiled cluster using XLA!  This line
is logged at most once for the lifetime of the process.
```

```
295/295 [==============================] - 23s 58ms/step - loss: 0.6012 - accuracy: 0.8002 - val_lo
ss: 0.0723 - val_accuracy: 0.9800 - lr: 0.0010
Epoch 2/25
295/295 [==============================] - 15s 52ms/step - loss: 0.1204 - accuracy: 0.9659 - val_lo
ss: 0.0721 - val_accuracy: 0.9814 - lr: 0.0010
Epoch 3/25
295/295 [==============================] - 16s 53ms/step - loss: 0.0898 - accuracy: 0.9755 - val_lo
ss: 0.0416 - val_accuracy: 0.9864 - lr: 0.0010
Epoch 4/25
295/295 [==============================] - 15s 52ms/step - loss: 0.0704 - accuracy: 0.9808 - val_lo
ss: 0.0345 - val_accuracy: 0.9912 - lr: 0.0010
Epoch 5/25
295/295 [==============================] - 16s 53ms/step - loss: 0.0626 - accuracy: 0.9829 - val_lo
ss: 0.0479 - val_accuracy: 0.9879 - lr: 0.0010
Epoch 6/25
295/295 [==============================] - 15s 51ms/step - loss: 0.0579 - accuracy: 0.9840 - val_lo
ss: 0.0273 - val_accuracy: 0.9921 - lr: 0.0010
Epoch 7/25
295/295 [==============================] - 16s 53ms/step - loss: 0.0524 - accuracy: 0.9859 - val_lo
ss: 0.0335 - val_accuracy: 0.9910 - lr: 0.0010
Epoch 8/25
295/295 [==============================] - 15s 52ms/step - loss: 0.0497 - accuracy: 0.9858 - val_lo
ss: 0.0273 - val_accuracy: 0.9914 - lr: 0.0010
Epoch 9/25
295/295 [==============================] - 15s 52ms/step - loss: 0.0515 - accuracy: 0.9859 - val_lo
ss: 0.0272 - val_accuracy: 0.9919 - lr: 0.0010
Epoch 10/25
295/295 [==============================] - 15s 52ms/step - loss: 0.0436 - accuracy: 0.9875 - val_lo
ss: 0.0247 - val_accuracy: 0.9945 - lr: 0.0010
Epoch 11/25
295/295 [==============================] - 15s 52ms/step - loss: 0.0422 - accuracy: 0.9878 - val_lo
ss: 0.0293 - val_accuracy: 0.9919 - lr: 0.0010
Epoch 12/25
295/295 [==============================] - 15s 51ms/step - loss: 0.0423 - accuracy: 0.9877 - val_lo
ss: 0.0403 - val_accuracy: 0.9910 - lr: 0.0010
Epoch 13/25
295/295 [==============================] - 15s 51ms/step - loss: 0.0375 - accuracy: 0.9895 - val_lo
ss: 0.0272 - val_accuracy: 0.9940 - lr: 0.0010
Epoch 14/25
295/295 [==============================] - 15s 52ms/step - loss: 0.0367 - accuracy: 0.9903 - val_lo
ss: 0.0282 - val_accuracy: 0.9938 - lr: 0.0010
Epoch 15/25
295/295 [==============================] - 15s 52ms/step - loss: 0.0379 - accuracy: 0.9897 - val_lo
ss: 0.0375 - val_accuracy: 0.9900 - lr: 0.0010
Epoch 16/25
295/295 [==============================] - 15s 51ms/step - loss: 0.0360 - accuracy: 0.9901 - val_lo
ss: 0.0380 - val_accuracy: 0.9902 - lr: 0.0010
Epoch 17/25
295/295 [==============================] - 15s 52ms/step - loss: 0.0328 - accuracy: 0.9908 - val_lo
ss: 0.0275 - val_accuracy: 0.9929 - lr: 0.0010
Epoch 18/25
295/295 [==============================] - 15s 52ms/step - loss: 0.0350 - accuracy: 0.9905 - val_lo
ss: 0.0349 - val_accuracy: 0.9907 - lr: 0.0010
Epoch 19/25
295/295 [==============================] - 15s 51ms/step - loss: 0.0322 - accuracy: 0.9908 - val_lo
ss: 0.0240 - val_accuracy: 0.9931 - lr: 0.0010
Epoch 20/25
295/295 [==============================] - 15s 51ms/step - loss: 0.0321 - accuracy: 0.9911 - val_lo
ss: 0.0303 - val_accuracy: 0.9926 - lr: 0.0010
Epoch 21/25
295/295 [==============================] - 15s 52ms/step - loss: 0.0290 - accuracy: 0.9926 - val_lo
ss: 0.0270 - val_accuracy: 0.9924 - lr: 0.0010
Epoch 22/25
295/295 [==============================] - 15s 52ms/step - loss: 0.0318 - accuracy: 0.9915 - val_lo
ss: 0.0348 - val_accuracy: 0.9905 - lr: 0.0010
Epoch 23/25
295/295 [==============================] - 15s 51ms/step - loss: 0.0320 - accuracy: 0.9914 - val_lo
ss: 0.0257 - val_accuracy: 0.9926 - lr: 0.0010
```

```
Epoch 24/25
295/295 [==============================] - 16s 53ms/step - loss: 0.0276 - accuracy: 0.9923 - val_lo
ss: 0.0292 - val_accuracy: 0.9948 - lr: 0.0010
Epoch 25/25
295/295 [==============================] - 15s 52ms/step - loss: 0.0288 - accuracy: 0.9922 - val_lo
ss: 0.0295 - val_accuracy: 0.9921 - lr: 0.0010
```

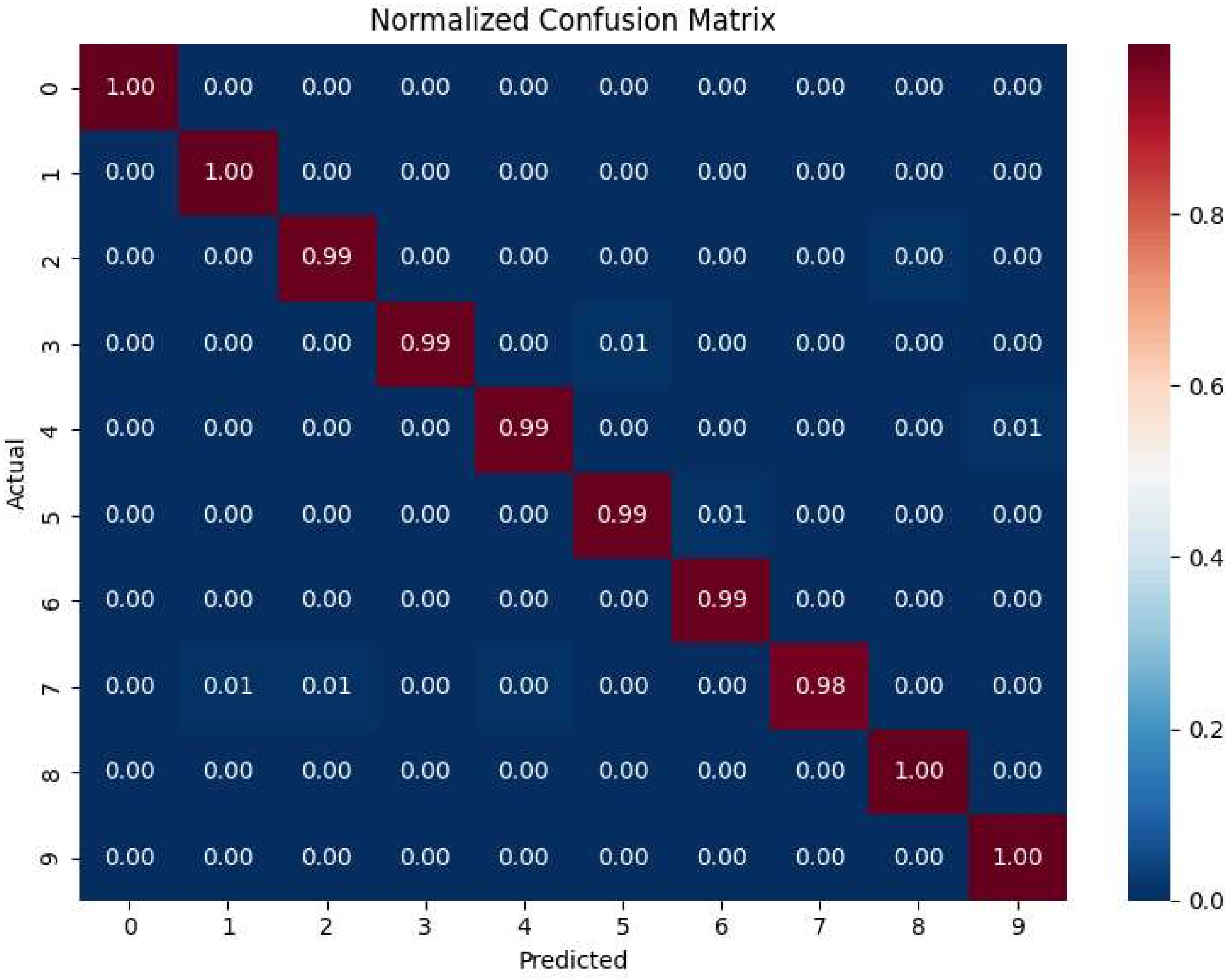
In [15]:

```
#Turns off warnings
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

#Creates figure and axes
fig, ax = plt.subplots(2, 1, figsize=(8, 6))

#Plots the loss of the model epochs
sns.lineplot(x=range(len(history.history['loss'])), y=history.history['loss'], ax=ax[0], color='b',
sns.lineplot(x=range(len(history.history['val_loss'])), y=history.history['val_loss'], ax=ax[0], col
ax[0].set_title('Loss')
ax[0].set_xlabel('Epochs')
ax[0].set_ylabel('Loss')

#Plots the accuracy of the model epochs
sns.lineplot(x=range(len(history.history['accuracy'])), y=history.history['accuracy'], ax=ax[1], col
sns.lineplot(x=range(len(history.history['val_accuracy'])), y=history.history['val_accuracy'], ax=ax
ax[1].set_title('Accuracy')
ax[1].set_xlabel('Epochs')
ax[1].set_ylabel('Accuracy')

plt.tight_layout()
plt.show()
```



In [16]:
```
y_pred = model.predict(x_val)
y_pred_classes = np.argmax(y_pred,axis = 1)
y_true = np.argmax(y_val,axis = 1)

#Creates the confusion matrix
cm = confusion_matrix(y_true, y_pred_classes)
cm_normalized = cm / cm.sum(axis=1, keepdims=True)

# Plot the confusion matrix using Seaborn's heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm_normalized, annot=True, fmt=".2f", cmap='RdBu_r', center=.5)
plt.title('Normalized Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.xticks(np.arange(10) + 0.5, range(10))
plt.yticks(np.arange(10) + 0.5, range(10))
plt.tight_layout()
plt.show()
```

```
132/132 [==============================] - 1s 3ms/step
```



In [17]:
```
#Predicts results
results = model.predict(x_test)

# selects the indix with the maximum probability
results = np.argmax(results,axis = 1)
results = pd.Series(results,name="Label")
```

```
875/875 [==============================] - 2s 2ms/step
```

In [18]:

```
submission = pd.concat([pd.Series(range(1,28001),name = "ImageId"),results],axis = 1)
submission.to_csv("cnn_mnist_datagen.csv",index=False)
```