import liberaies

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
import matplotlib.cm as cm
from keras.models import Sequential,load_model
from keras.layers import Dense,Dropout,Flatten,Conv2D,MaxPooling2D
from keras.callbacks import ReduceLROnPlateau,ModelCheckpoint

import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [14]:
```
train = pd.read_csv('/kaggle/input/digit-recognizer/train.csv')
test = pd.read_csv('/kaggle/input/digit-recognizer/test.csv')
```

In [15]:
```
test.shape
train.shape
```

Out[15]: (42000, 785)

In [16]:
```
test = test.values.astype('float32')
test = test.reshape(-1, 28, 28, 1)
X_val = train.iloc[:, 1:].values.astype('float32')
X_val = X_val.reshape(-1, 28, 28, 1)
y_val = train.iloc[:, 0].values.astype('int32')
print(X_val.shape, y_val.shape)
```

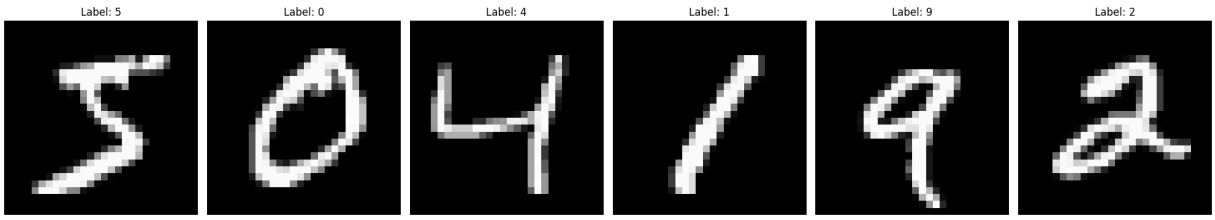
```
(42000, 28, 28, 1) (42000,)
```

In [17]:
```
(X_train,y_train),(X_test,y_test)=mnist.load_data()
X_train=np.vstack((X_train,X_test))
y_train=np.concatenate([y_train,y_test])
X_train=X_train.reshape(-1,28,28,1)
X_train.shape,y_train.shape
```

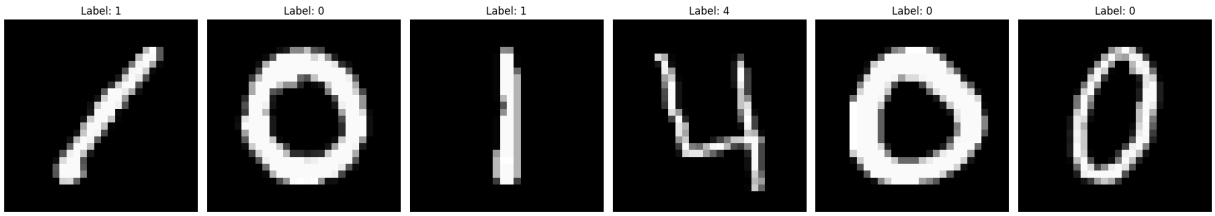
Out[17]: ((70000, 28, 28, 1), (70000,))

In [18]:
```
import matplotlib.pyplot as plt

# Number of images to display
num_images = 6

# Create subplots
fig, axes = plt.subplots(1, num_images, figsize=(20, 20))

# Display each image with its label
for i in range(num_images):
    axes[i].imshow(X_train[i].reshape(28, 28), cmap='gray')  # Reshape and display
```

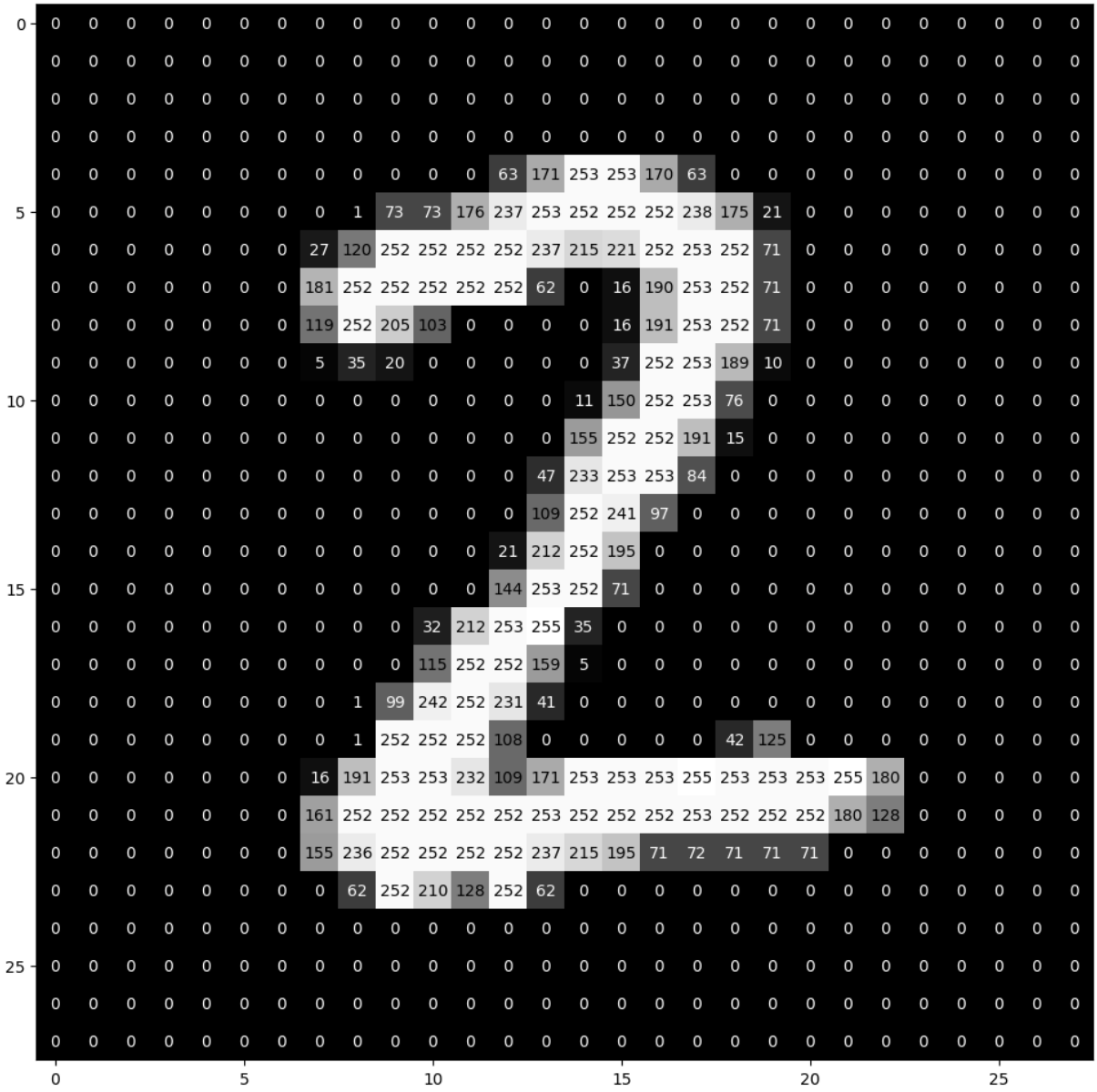
```
    axes[i].set_title(f"Label: {y_train[i]}")  # Set title as label
    axes[i].axis('off')  # Hide axis for clarity

plt.tight_layout()  # Adjust layout for better spacing
plt.show()
```



In [19]:
```
fig, ax = plt.subplots(1, 6, figsize=(20, 20))

# Loop through the first 6 images
for u in range(6):
    ax[u].imshow(X_val[u].reshape(28, 28), cmap='gray')  # Reshape to 28x28 and dis
    ax[u].set_title(f"Label: {y_val[u]}")  # Set title as label for each image
    ax[u].axis('off')  # Hide axis for better visualization

plt.tight_layout()  # Adjust layout for spacing
plt.show()
```



In [20]:
```
def img_to_num(img, ax):
    ax.imshow(img, cmap='gray')
    w, h = img.shape
    thresh = img.max() / 2.5
    for x in range(w):
        for y in range(h):
            ax.annotate(
                str(round(img[x, y])),
                xy=(y, x),
                horizontalalignment='center',
                verticalalignment='center',
                color='white' if img[x, y] < thresh else 'black'
            )

fig = plt.figure(figsize=(12, 12))
ax = fig.add_subplot(111)
img_to_num(X_val[22].reshape(28, 28), ax)
plt.show()
```



normalization

In [21]:
```
X_train = X_train/255
test = test/255
X_val = X_val/255
```

label encoding

In [22]:
```
print("Before catagorizing ",y_train[77])
y_train=to_categorical(y_train,10)
y_val=to_categorical(y_val,10)
print("Before catagorizing " ,y_train[77])
```

```
Before catagorizing  1
Before catagorizing  [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```

In [23]:
```
model = Sequential()

model.add(Conv2D(filters=64,kernel_size=3,padding='same',activation='relu',input_sh
model.add(Conv2D(filters=128,kernel_size=3,padding='same',activation='relu'))
model.add(Conv2D(filters=128,kernel_size=3,padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=128,kernel_size=3,padding='same',activation='relu'))
model.add(Conv2D(filters=192,kernel_size=3,padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=192,kernel_size=5,padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=2,padding='same'))
model.add(Flatten())
model.add(Dense(256,activation='relu'))
model.add(Dense(10,activation='softmax'))

model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 28, 28, 64)	640
conv2d_7 (Conv2D)	(None, 28, 28, 128)	73,856
conv2d_8 (Conv2D)	(None, 28, 28, 128)	147,584
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 128)	0
conv2d_9 (Conv2D)	(None, 14, 14, 128)	147,584
conv2d_10 (Conv2D)	(None, 14, 14, 192)	221,376
max_pooling2d_4 (MaxPooling2D)	(None, 7, 7, 192)	0
conv2d_11 (Conv2D)	(None, 7, 7, 192)	921,792
max_pooling2d_5 (MaxPooling2D)	(None, 4, 4, 192)	0
flatten_1 (Flatten)	(None, 3072)	0
dense_2 (Dense)	(None, 256)	786,688
dense_3 (Dense)	(None, 10)	2,570

Total params: 2,302,090 (8.78 MB)

Trainable params: 2,302,090 (8.78 MB)

Non-trainable params: 0 (0.00 B)

In [24]:
```
model.compile(loss = 'categorical_crossentropy',optimizer='adam',metrics=['accuracy

lr = ReduceLROnPlateau(monitor='loss',factor = 0.3,verbose = 1,patience = 2, min_lr
```

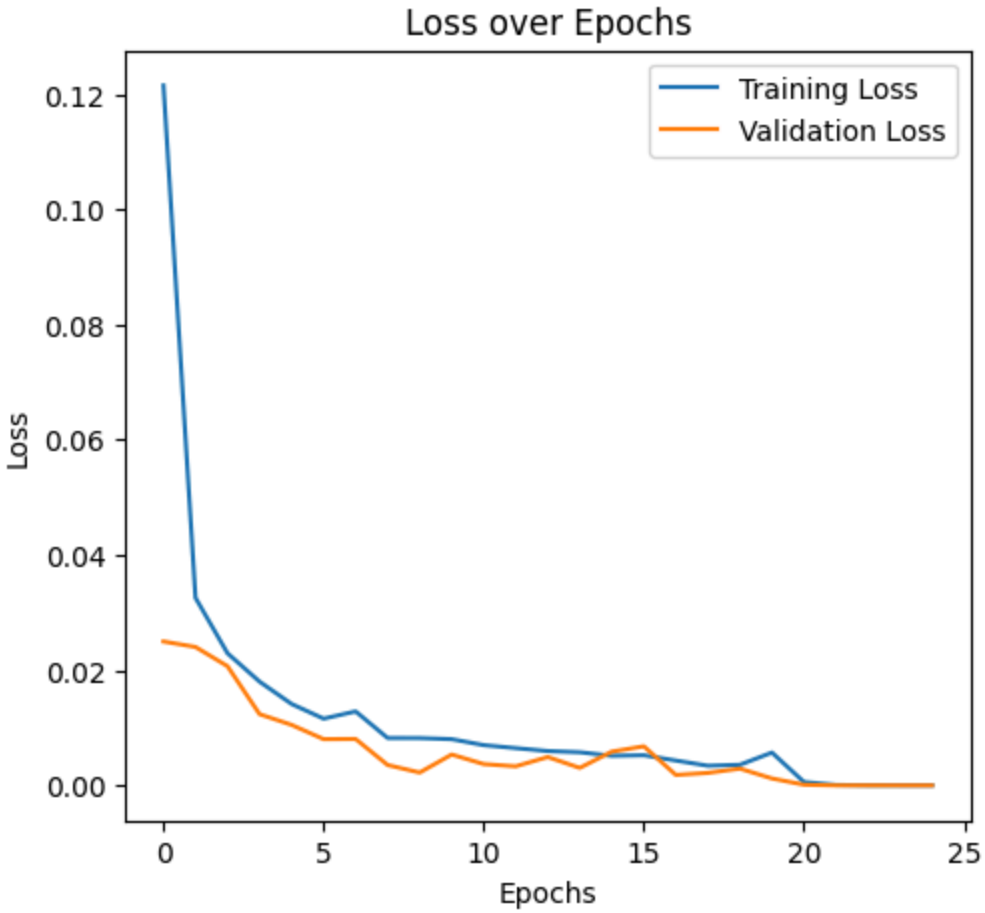
```
history=model.fit(X_train,y_train,batch_size = 128,epochs = 25,validation_data=(X_v

model.save('my_model.keras')
```

```
Epoch 1/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 19s 28ms/step - accuracy: 0.8965 - loss: 0.3222 - val_a
ccuracy: 0.9927 - val_loss: 0.0250 - learning_rate: 0.0010
Epoch 2/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9895 - loss: 0.0340 - val_a
ccuracy: 0.9926 - val_loss: 0.0240 - learning_rate: 0.0010
Epoch 3/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9929 - loss: 0.0224 - val_a
ccuracy: 0.9943 - val_loss: 0.0207 - learning_rate: 0.0010
Epoch 4/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9953 - loss: 0.0163 - val_a
ccuracy: 0.9962 - val_loss: 0.0124 - learning_rate: 0.0010
Epoch 5/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9957 - loss: 0.0129 - val_a
ccuracy: 0.9967 - val_loss: 0.0105 - learning_rate: 0.0010
Epoch 6/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9966 - loss: 0.0104 - val_a
ccuracy: 0.9974 - val_loss: 0.0080 - learning_rate: 0.0010
Epoch 7/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9963 - loss: 0.0113 - val_a
ccuracy: 0.9974 - val_loss: 0.0081 - learning_rate: 0.0010
Epoch 8/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9971 - loss: 0.0082 - val_a
ccuracy: 0.9990 - val_loss: 0.0035 - learning_rate: 0.0010
Epoch 9/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9979 - loss: 0.0070 - val_a
ccuracy: 0.9994 - val_loss: 0.0022 - learning_rate: 0.0010
Epoch 10/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9978 - loss: 0.0064 - val_a
ccuracy: 0.9982 - val_loss: 0.0054 - learning_rate: 0.0010
Epoch 11/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9976 - loss: 0.0069 - val_a
ccuracy: 0.9989 - val_loss: 0.0037 - learning_rate: 0.0010
Epoch 12/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9980 - loss: 0.0061 - val_a
ccuracy: 0.9989 - val_loss: 0.0033 - learning_rate: 0.0010
Epoch 13/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9985 - loss: 0.0041 - val_a
ccuracy: 0.9986 - val_loss: 0.0049 - learning_rate: 0.0010
Epoch 14/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9981 - loss: 0.0060 - val_a
ccuracy: 0.9990 - val_loss: 0.0030 - learning_rate: 0.0010
Epoch 15/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9988 - loss: 0.0037 - val_a
ccuracy: 0.9980 - val_loss: 0.0059 - learning_rate: 0.0010
Epoch 16/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9985 - loss: 0.0045 - val_a
ccuracy: 0.9977 - val_loss: 0.0068 - learning_rate: 0.0010
Epoch 17/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9985 - loss: 0.0046 - val_a
ccuracy: 0.9994 - val_loss: 0.0018 - learning_rate: 0.0010
Epoch 18/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9997 - loss: 0.0013 - val_a
ccuracy: 0.9994 - val_loss: 0.0022 - learning_rate: 0.0010
Epoch 19/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9989 - loss: 0.0032 - val_a
```

```
ccuracy: 0.9991 - val_loss: 0.0029 - learning_rate: 0.0010
Epoch 20/25
546/547 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.9985 - loss: 0.0063
Epoch 20: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 25ms/step - accuracy: 0.9985 - loss: 0.0063 - val_a
ccuracy: 0.9996 - val_loss: 0.0012 - learning_rate: 0.0010
Epoch 21/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 0.9998 - loss: 8.2900e-04 - v
al_accuracy: 1.0000 - val_loss: 1.3161e-04 - learning_rate: 3.0000e-04
Epoch 22/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 25ms/step - accuracy: 1.0000 - loss: 1.4943e-04 - v
al_accuracy: 1.0000 - val_loss: 1.1703e-05 - learning_rate: 3.0000e-04
Epoch 23/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 1.0000 - loss: 1.0145e-05 - v
al_accuracy: 1.0000 - val_loss: 6.5937e-06 - learning_rate: 3.0000e-04
Epoch 24/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - accuracy: 1.0000 - loss: 5.6928e-06
Epoch 24: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 1.0000 - loss: 5.6930e-06 - v
al_accuracy: 1.0000 - val_loss: 4.5018e-06 - learning_rate: 3.0000e-04
Epoch 25/25
547/547 ━━━━━━━━━━━━━━━━━━━━ 13s 24ms/step - accuracy: 1.0000 - loss: 4.3830e-06 - v
al_accuracy: 1.0000 - val_loss: 4.0213e-06 - learning_rate: 9.0000e-05
```

In [31]:
```python
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

Out[31]:
```
<matplotlib.legend.Legend at 0x7bb3266d2860>
```



In [32]:
```
pred  = model.predict(test,verbose=2)
pred_classes = np.argmax(pred, axis=1)
```

```
875/875 - 2s - 2ms/step
```

In [33]:
```
submission = pd.read_csv('/kaggle/input/digit-recognizer/sample_submission.csv')
submission["Label"] = pred_classes
submission.to_csv('submission.csv',index=False)
print("Submission file created")
```

```
Submission file created
```

In [34]:
```
submission
```

Out[34]:

	ImageId	Label
0	1	2
1	2	0
2	3	9
3	4	0
4	5	3
...	...	...
27995	27996	9
27996	27997	7
27997	27998	3
27998	27999	9
27999	28000	2

28000 rows × 2 columns

now lets evaluate our model ofrom train data

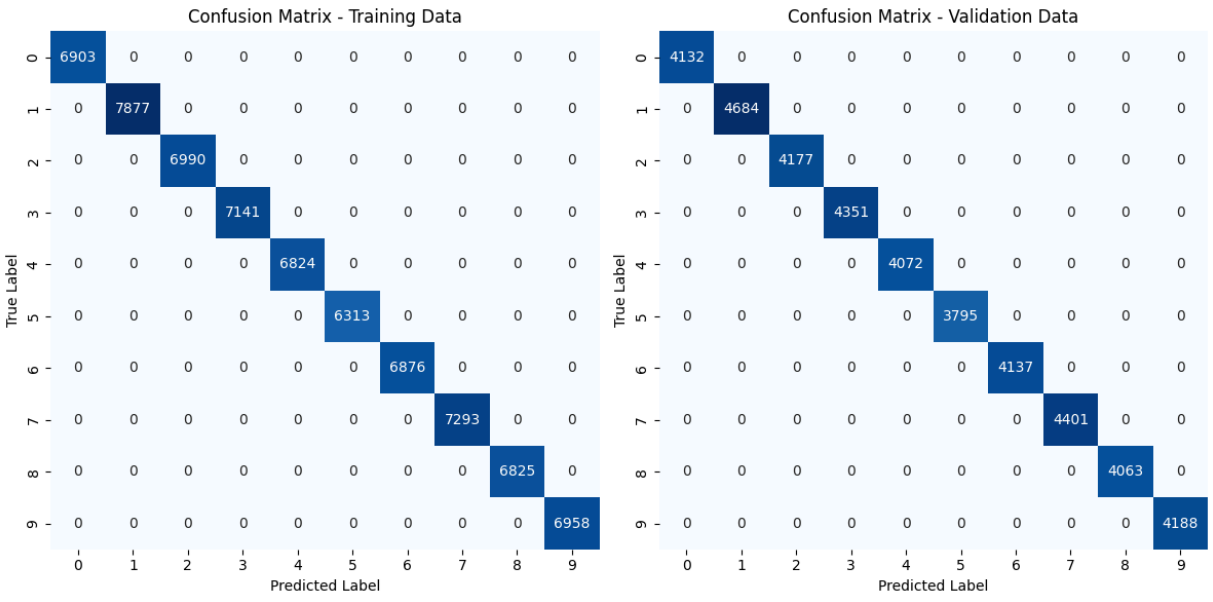
In [35]:

```
from sklearn.metrics import classification_report, accuracy_score, precision_score,
import numpy as np

# Make predictions on the training and validation sets
pred_tr = model.predict(X_train).argmax(axis=1)  # Predict on training data and get
pred_val = model.predict(X_val).argmax(axis=1)   # Predict on validation data and g

# Convert one-hot encoded y_train and y_val to label format for metric calculations
y_train_labels = np.argmax(y_train, axis=1)
y_val_labels = np.argmax(y_val, axis=1)

# Training Evaluation Metrics
print('Training Evaluation Metrics:\n')

print('Classification Report (Training):\n', classification_report(y_train_labels,
print('Accuracy (Training):', accuracy_score(y_train_labels, pred_tr))
print('Precision (Training):', precision_score(y_train_labels, pred_tr, average='mi
print('Recall (Training):', recall_score(y_train_labels, pred_tr, average='micro'))
print('F1 Score (Training):', f1_score(y_train_labels, pred_tr, average='micro'))

print('\nTesting Evaluation Metrics:\n')

# Validation (Testing) Evaluation Metrics
print('Classification Report (Testing):\n', classification_report(y_val_labels, pre
print('Accuracy (Testing):', accuracy_score(y_val_labels, pred_val))
print('Precision (Testing):', precision_score(y_val_labels, pred_val, average='micr
```

```
print('Recall (Testing):', recall_score(y_val_labels, pred_val, average='micro'))
print('F1 Score (Testing):', f1_score(y_val_labels, pred_val, average='micro'))
```

```
2188/2188 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step
Training Evaluation Metrics:

Classification Report (Training):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      6903
           1       1.00      1.00      1.00      7877
           2       1.00      1.00      1.00      6990
           3       1.00      1.00      1.00      7141
           4       1.00      1.00      1.00      6824
           5       1.00      1.00      1.00      6313
           6       1.00      1.00      1.00      6876
           7       1.00      1.00      1.00      7293
           8       1.00      1.00      1.00      6825
           9       1.00      1.00      1.00      6958

    accuracy                           1.00     70000
   macro avg       1.00      1.00      1.00     70000
weighted avg       1.00      1.00      1.00     70000

Accuracy (Training): 1.0
Precision (Training): 1.0
Recall (Training): 1.0
F1 Score (Training): 1.0

Testing Evaluation Metrics:

Classification Report (Testing):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4132
           1       1.00      1.00      1.00      4684
           2       1.00      1.00      1.00      4177
           3       1.00      1.00      1.00      4351
           4       1.00      1.00      1.00      4072
           5       1.00      1.00      1.00      3795
           6       1.00      1.00      1.00      4137
           7       1.00      1.00      1.00      4401
           8       1.00      1.00      1.00      4063
           9       1.00      1.00      1.00      4188

    accuracy                           1.00     42000
   macro avg       1.00      1.00      1.00     42000
weighted avg       1.00      1.00      1.00     42000

Accuracy (Testing): 1.0
Precision (Testing): 1.0
Recall (Testing): 1.0
F1 Score (Testing): 1.0
```

In [36]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Generate confusion matrices for training and validation predictions
conf_matrix_train = confusion_matrix(y_train_labels, pred_tr)
conf_matrix_val = confusion_matrix(y_val_labels, pred_val)

# Plot confusion matrix for training data
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.heatmap(conf_matrix_train, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.title("Confusion Matrix - Training Data")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")

# Plot confusion matrix for validation data
plt.subplot(1, 2, 2)
sns.heatmap(conf_matrix_val, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.title("Confusion Matrix - Validation Data")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")

plt.tight_layout()
plt.show()
plt.savefig('Confusion matrix')
```




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

In [ ]: