Detection of Cardio Vascular Diseases of MIT_BIH Dataset using CNN

In [3]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout
from keras.utils import to_categorical
```

In [4]:

```
!pip install wfdb
```

```
Collecting wfdb
  Downloading wfdb-4.1.2-py3-none-any.whl (159 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 160.0/160.0 kB 3.4 MB/s eta 0:00:00
Requirement already satisfied: SoundFile>=0.10.0 in /usr/local/lib/python3.10/dist-packag
es (from wfdb) (0.12.1)
Requirement already satisfied: matplotlib>=3.2.2 in /usr/local/lib/python3.10/dist-packag
es (from wfdb) (3.7.1)
Requirement already satisfied: numpy>=1.10.1 in /usr/local/lib/python3.10/dist-packages (
from wfdb) (1.25.2)
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.10/dist-packages (
from wfdb) (2.0.3)
Requirement already satisfied: requests>=2.8.1 in /usr/local/lib/python3.10/dist-packages
(from wfdb) (2.31.0)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (f
rom wfdb) (1.11.4)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-package
s (from matplotlib>=3.2.2->wfdb) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (f
rom matplotlib>=3.2.2->wfdb) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packag
es (from matplotlib>=3.2.2->wfdb) (4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packag
es (from matplotlib>=3.2.2->wfdb) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages
(from matplotlib>=3.2.2->wfdb) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (
from matplotlib>=3.2.2->wfdb) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-package
s (from matplotlib>=3.2.2->wfdb) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-pac
kages (from matplotlib>=3.2.2->wfdb) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (f
rom pandas>=1.3.0->wfdb) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages
(from pandas>=1.3.0->wfdb) (2024.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist
-packages (from requests>=2.8.1->wfdb) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (f
rom requests>=2.8.1->wfdb) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packa
ges (from requests>=2.8.1->wfdb) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packa
ges (from requests>=2.8.1->wfdb) (2024.6.2)
Requirement already satisfied: cffi>=1.0 in /usr/local/lib/python3.10/dist-packages (from
SoundFile>=0.10.0->wfdb) (1.16.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from
cffi>=1.0->SoundFile>=0.10.0->wfdb) (2.22)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from
python-dateutil>=2.7->matplotlib>=3.2.2->wfdb) (1.16.0)
Installing collected packages: wfdb
Successfully installed wfdb-4.1.2
```

In [5]:

```
import wfdb
from wfdb import processing


wfdb.dl_database('mitdb', dl_dir='mitdb')


record = wfdb.rdrecord('mitdb/100', sampfrom=0, sampto=10000)
annotation = wfdb.rdann('mitdb/100', 'atr', sampfrom=0, sampto=10000)
```

```
Generating record list for: 100
Generating record list for: 101
Generating record list for: 102
Generating record list for: 103
Generating record list for: 104
Generating record list for: 105
Generating record list for: 106
Generating record list for: 107
Generating record list for: 108
Generating record list for: 109
Generating record list for: 111
Generating record list for: 112
Generating record list for: 113
Generating record list for: 114
Generating record list for: 115
Generating record list for: 116
Generating record list for: 117
Generating record list for: 118
Generating record list for: 119
Generating record list for: 121
Generating record list for: 122
Generating record list for: 123
Generating record list for: 124
Generating record list for: 200
Generating record list for: 201
Generating record list for: 202
Generating record list for: 203
Generating record list for: 205
Generating record list for: 207
Generating record list for: 208
Generating record list for: 209
Generating record list for: 210
Generating record list for: 212
Generating record list for: 213
Generating record list for: 214
Generating record list for: 215
Generating record list for: 217
Generating record list for: 219
Generating record list for: 220
Generating record list for: 221
Generating record list for: 222
Generating record list for: 223
Generating record list for: 228
Generating record list for: 230
Generating record list for: 231
Generating record list for: 232
Generating record list for: 233
Generating record list for: 234
Generating list of all files for: 100
Generating list of all files for: 101
Generating list of all files for: 102
Generating list of all files for: 103
Generating list of all files for: 104
Generating list of all files for: 105
Generating list of all files for: 106
Generating list of all files for: 107
Generating list of all files for: 108
Generating list of all files for: 109
Generating list of all files for: 111
Generating list of all files for: 112
```

```
Generating list of all files for: 113
Generating list of all files for: 114
Generating list of all files for: 115
Generating list of all files for: 116
Generating list of all files for: 117
Generating list of all files for: 118
Generating list of all files for: 119
Generating list of all files for: 121
Generating list of all files for: 122
Generating list of all files for: 123
Generating list of all files for: 124
Generating list of all files for: 200
Generating list of all files for: 201
Generating list of all files for: 202
Generating list of all files for: 203
Generating list of all files for: 205
Generating list of all files for: 207
Generating list of all files for: 208
Generating list of all files for: 209
Generating list of all files for: 210
Generating list of all files for: 212
Generating list of all files for: 213
Generating list of all files for: 214
Generating list of all files for: 215
Generating list of all files for: 217
Generating list of all files for: 219
Generating list of all files for: 220
Generating list of all files for: 221
Generating list of all files for: 222
Generating list of all files for: 223
Generating list of all files for: 228
Generating list of all files for: 230
Generating list of all files for: 231
Generating list of all files for: 232
Generating list of all files for: 233
Generating list of all files for: 234
Created local base download directory: mitdb
Downloading files...
Finished downloading files
```

In [6]:

```
signal = record.p_signal[:, 0]
ann_samples = annotation.sample

window_size = 180  # Number of samples per segment
X = []
y = []
for ann in ann_samples:
    start = ann - window_size // 2
    end = ann + window_size // 2
    if start >= 0 and end < len(signal):
        X.append(signal[start:end])
        y.append(annotation.symbol[np.where(annotation.sample == ann)[0][0]])

# Convert lists to numpy arrays
X = np.array(X)
y = np.array(y)

# Filter out non-beat annotations (e.g., labels like 'N', 'L', 'R')
valid_labels = ['N', 'L', 'R']  # Adjust based on desired labels
X = X[np.isin(y, valid_labels)]
y = y[np.isin(y, valid_labels)]

# Convert labels to integers
label_mapping = {label: idx for idx, label in enumerate(valid_labels)}
y = np.array([label_mapping[label] for label in y])

# Normalize the data
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

```
# One-hot encode the labels
y = to_categorical(y)
```

In [7]:

```
# Cell 5: Split the data into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
)
```

In [8]:

```
# Cell 6: Reshape data for the CNN
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
```

In [9]:

```
# Cell 7: Define the CNN model
model = Sequential([
    Conv1D(64, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)),
    MaxPooling1D(pool_size=2),
    Dropout(0.5),
    Conv1D(128, kernel_size=3, activation='relu'),
    MaxPooling1D(pool_size=2),
    Dropout(0.5),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(y_train.shape[1], activation='softmax')
])

# Compile model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

In [10]:

```
history = model.fit(X_train, y_train, validation_split=0.2, epochs=20, batch_size=32)
```

```
Epoch 1/20
```

```
/usr/local/lib/python3.10/dist-packages/tensorflow/python/util/dispatch.py:1260: SyntaxWa
rning: In loss categorical_crossentropy, expected y_pred.shape to be (batch_size, num_cla
sses) with num_classes > 1. Received: y_pred.shape=(None, 1). Consider using 'binary_cros
sentropy' if you only have 2 classes.
  return dispatch_target(*args, **kwargs)
```

```
1/1 [==============================] - 2s 2s/step - loss: 0.0000e+00 - accuracy: 1.0000 -
val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 2/20
1/1 [==============================] - 0s 109ms/step - loss: 0.0000e+00 - accuracy: 1.000
0 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 3/20
1/1 [==============================] - 0s 103ms/step - loss: 0.0000e+00 - accuracy: 1.000
0 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 4/20
1/1 [==============================] - 0s 106ms/step - loss: 0.0000e+00 - accuracy: 1.000
0 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 5/20
1/1 [==============================] - 0s 63ms/step - loss: 0.0000e+00 - accuracy: 1.0000
- val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 6/20
1/1 [==============================] - 0s 75ms/step - loss: 0.0000e+00 - accuracy: 1.0000
- val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 7/20
1/1 [==============================] - 0s 76ms/step - loss: 0.0000e+00 - accuracy: 1.0000
- val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 8/20
1/1 [==============================] - 0s 71ms/step - loss: 0.0000e+00 - accuracy: 1.0000
- val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 9/20
```

```
1/1 [==============================] - 0s 61ms/step - loss: 0.0000e+00 - accuracy: 1.0000
- val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 10/20
1/1 [==============================] - 0s 77ms/step - loss: 0.0000e+00 - accuracy: 1.0000
- val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 11/20
1/1 [==============================] - 0s 73ms/step - loss: 0.0000e+00 - accuracy: 1.0000
- val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 12/20
1/1 [==============================] - 0s 155ms/step - loss: 0.0000e+00 - accuracy: 1.000
0 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 13/20
1/1 [==============================] - 0s 102ms/step - loss: 0.0000e+00 - accuracy: 1.000
0 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 14/20
1/1 [==============================] - 0s 114ms/step - loss: 0.0000e+00 - accuracy: 1.000
0 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 15/20
1/1 [==============================] - 0s 96ms/step - loss: 0.0000e+00 - accuracy: 1.0000
- val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 16/20
1/1 [==============================] - 0s 121ms/step - loss: 0.0000e+00 - accuracy: 1.000
0 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 17/20
1/1 [==============================] - 0s 136ms/step - loss: 0.0000e+00 - accuracy: 1.000
0 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 18/20
1/1 [==============================] - 0s 108ms/step - loss: 0.0000e+00 - accuracy: 1.000
0 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 19/20
1/1 [==============================] - 0s 135ms/step - loss: 0.0000e+00 - accuracy: 1.000
0 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 20/20
1/1 [==============================] - 0s 124ms/step - loss: 0.0000e+00 - accuracy: 1.000
0 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
```

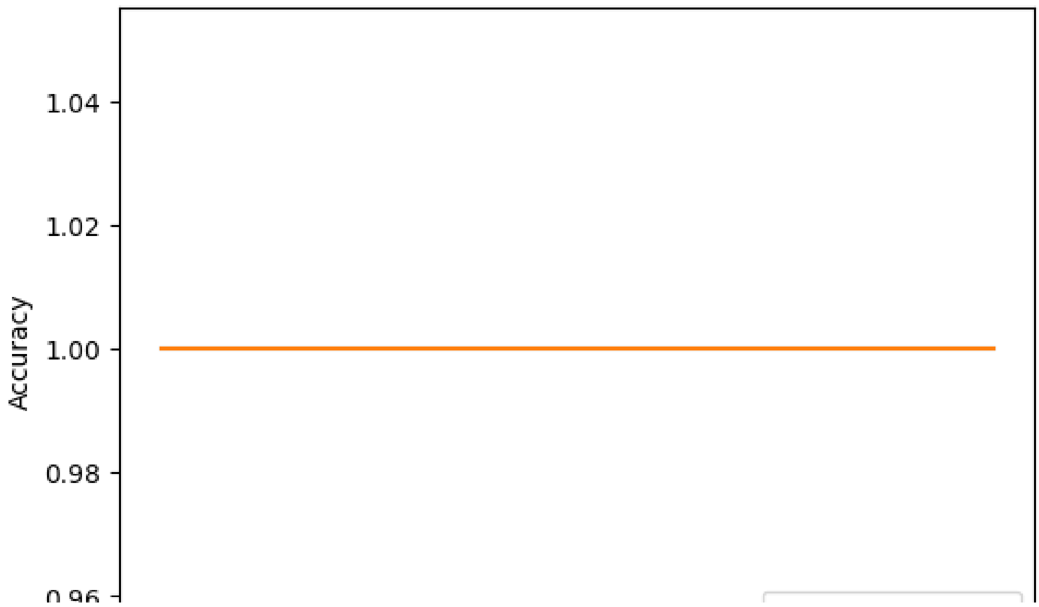
In [11]:

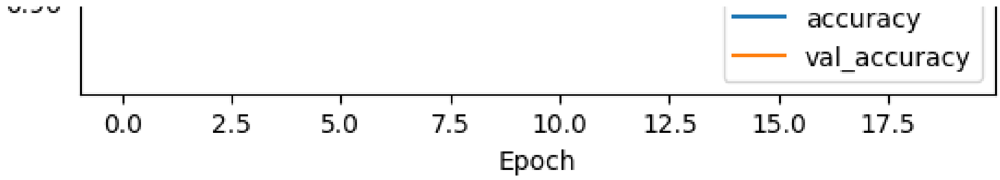
```
score = model.evaluate(X_test, y_test, verbose=0)
print(f'Test accuracy: {score[1]*100:.2f}%')
```

```
Test accuracy: 100.00%
```

In [12]:

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.show()
```

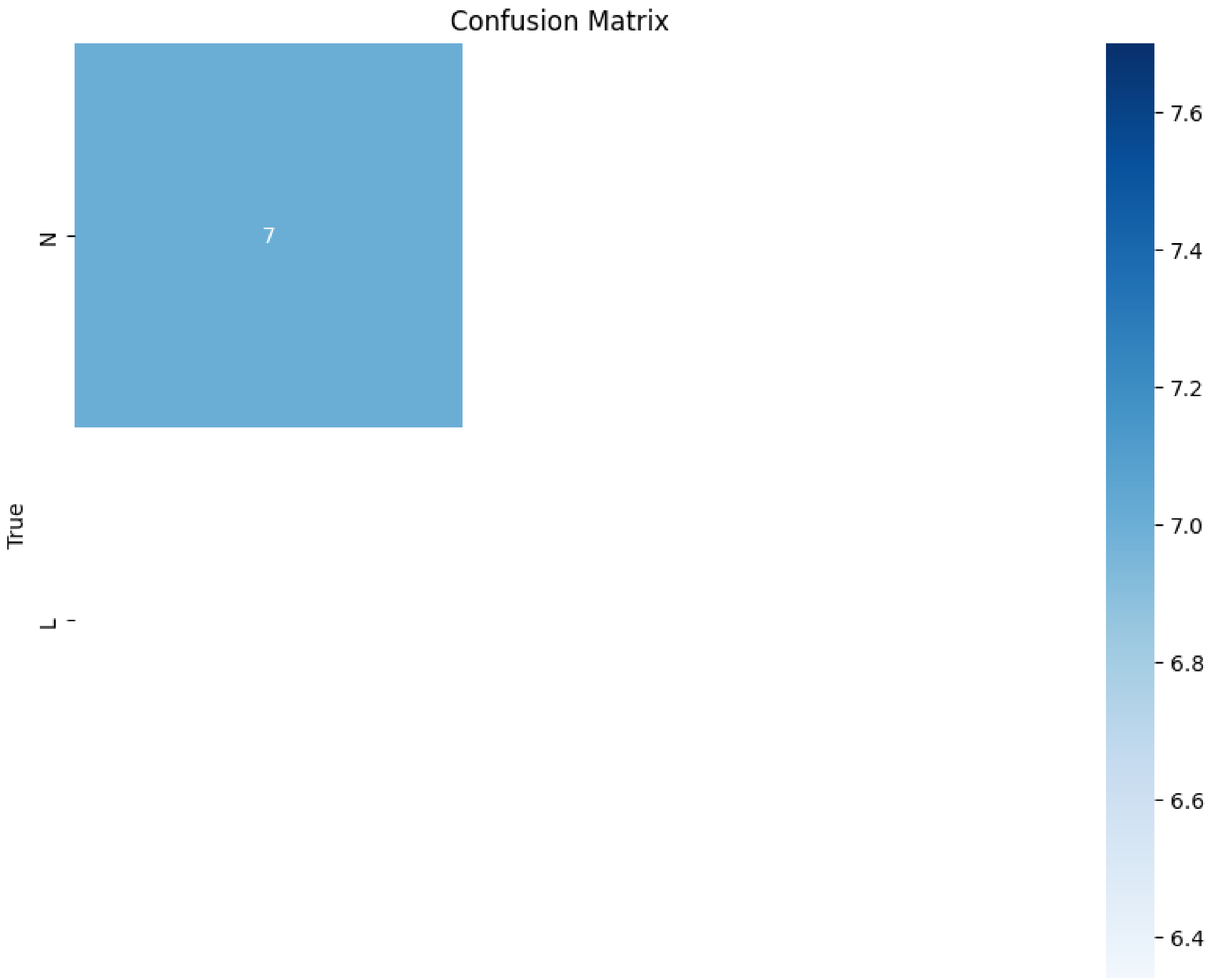




In [14]:

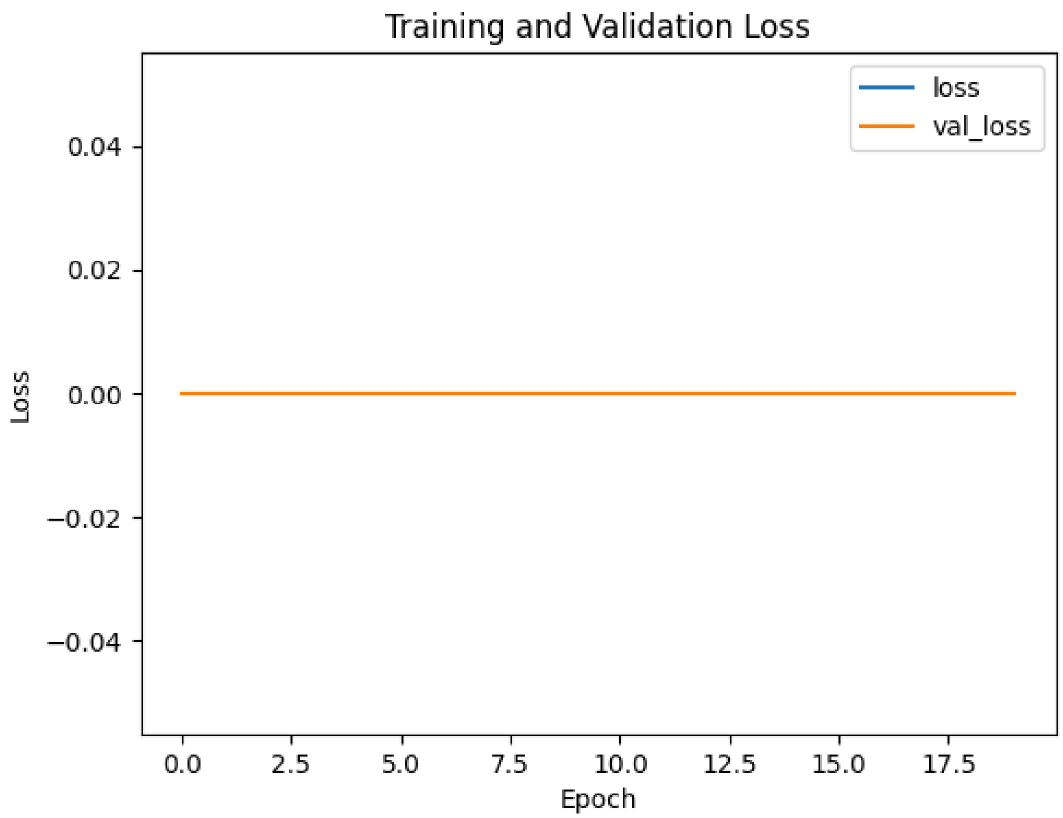
```
# Cell 11: Import additional libraries for confusion matrix and metrics
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns

# Predict the labels for the test set
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)

# Ensure the number of classes matches the valid_labels
unique_labels = np.unique(y_true)

# Compute confusion matrix
cm = confusion_matrix(y_true, y_pred_classes)

# Plot confusion matrix heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=valid_labels, yticklabels
=valid_labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```
1/1 [==============================] - 0s 23ms/step
```



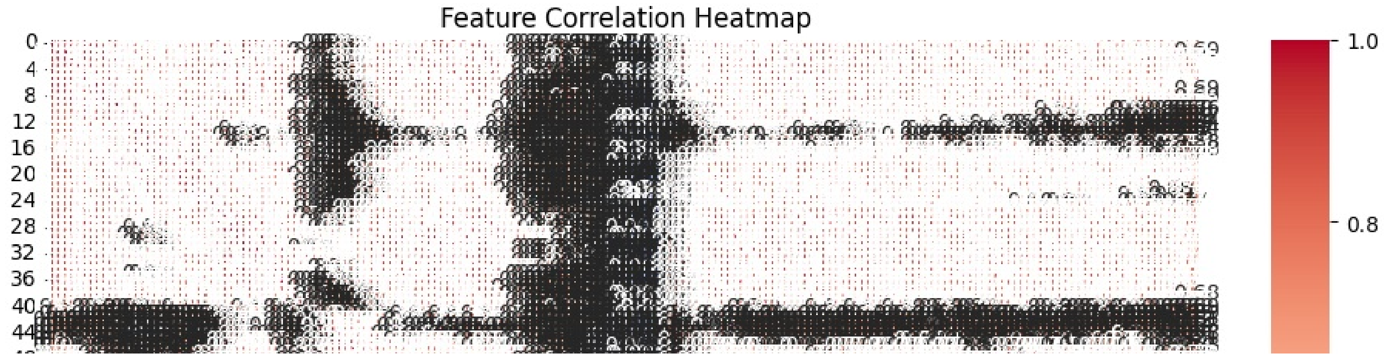
In [ ]:

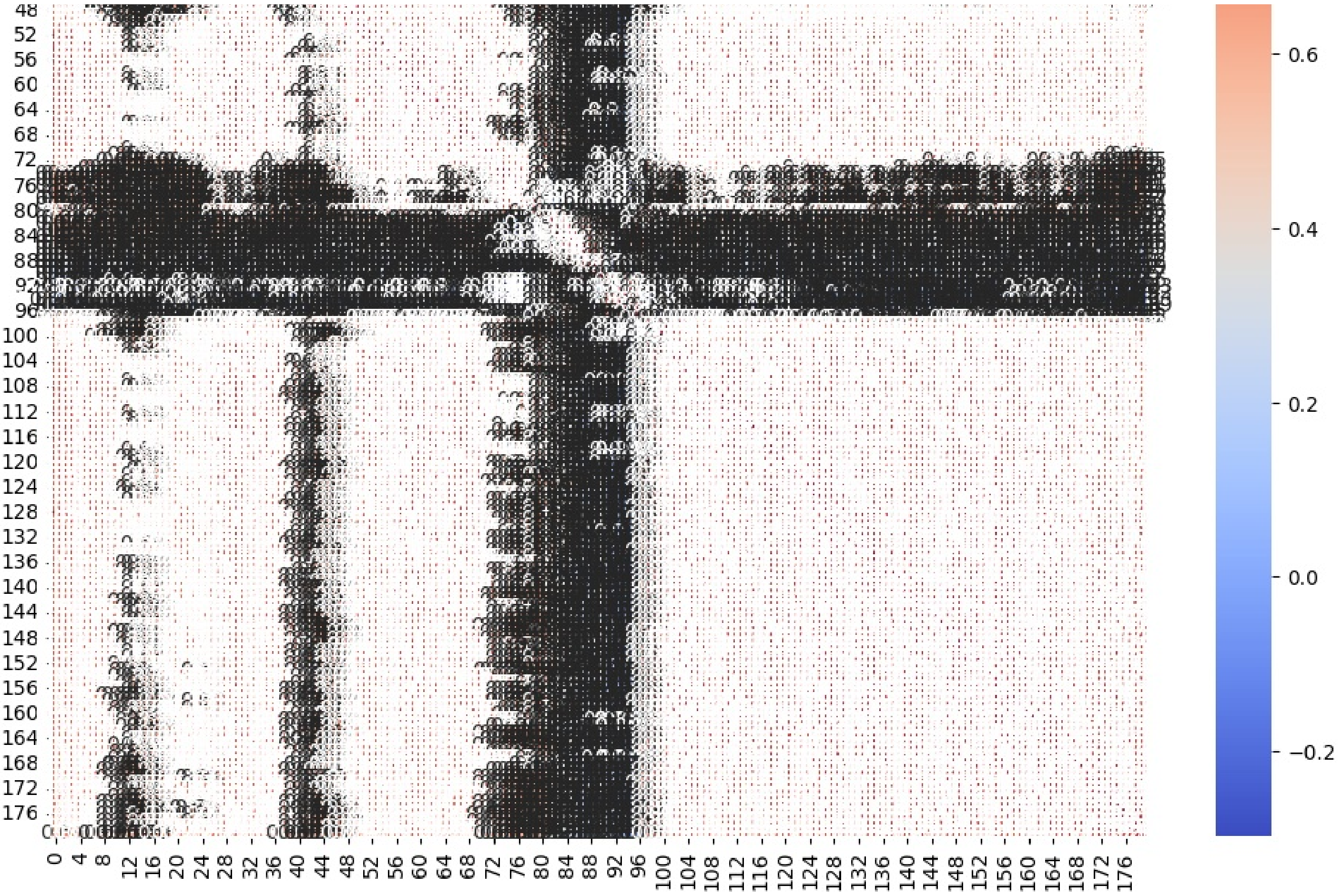
```python
# Cell 13: Plot training and validation loss
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



In [17]:

```python
# Cell 15: Heatmap of Feature Correlations (Optional)
# Note: This is more relevant if you have multiple features, e.g., using multiple channel
s or extracted features.

# Assuming `X` is the original feature set (before train/test split)
# Compute the correlation matrix
correlation_matrix = pd.DataFrame(X).corr()

# Plot heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Feature Correlation Heatmap')
plt.show()
```





48
52
56
60
64
68
72
76
80
84
88
92
96
100
104
108
112
116
120
124
128
132
136
140
144
148
152
156
160
164
168
172
176
0 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80 84 88 92 96 100 104 108 112 116 120 124 128 132 136 140 144 148 152 156 160 164 168 172 176
0.6
0.4
0.2
0.0
−0.2

In [ ]: