In [1]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout,
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

In [2]:
```
# Load the ECG data from the text file
file_path = r'C:\Users\diyu2\OneDrive - AUT University\AUT YEAR 4\INDUSTRIAL PROJEC
```

In [3]:
```
# Initialize a list to hold the ECG signals
data = []

# Temporary storage for reading lines
temp_signal = []

# Read the file and process each line
with open(file_path, 'r') as file:
    for line in file:
        line = line.strip()
        if line.startswith("Rhythm signal:") or line == '':
            continue  # Skip header line and empty lines

        print(f'Reading line: {line}')  # Debug: Show each line being read

        try:
            # Convert the line into a list of integers
            numbers = list(map(int, line.split()))  # Convert to integers
            temp_signal.extend(numbers)  # Append to the temporary list

            # Check if we have gathered enough data
            if len(temp_signal) >= 5000:
                # If we have enough, slice it to 5000 and append
                data.append(temp_signal[:5000])
                temp_signal = temp_signal[5000:]  # Keep any excess for the next si
                print(f'Added 5000 elements, remaining: {len(temp_signal)}')  # Deb

        except ValueError:
            print(f'ValueError: could not convert line to integers: {line}')  # Deb
            continue
```

```
Reading line: 19 -19 -39 0 29 -29 9 9 9 9 -9 -19
Reading line: 19 -19 -39 0 29 -29 9 9 9 14 -9 -14
Reading line: 14 -24 -39 4 26 -31 9 14 9 19 -4 -9
Reading line: 14 -24 -39 4 26 -31 9 14 9 24 -4 -4
Reading line: 9 -29 -39 9 24 -34 9 19 9 29 0 0
Reading line: 9 -34 -43 12 26 -39 9 19 14 29 0 0
Reading line: 4 -39 -43 17 24 -41 9 19 19 29 0 0
Reading line: 4 -43 -48 19 26 -46 9 19 24 29 0 0
Reading line: 0 -48 -48 24 24 -48 9 19 29 29 0 0
Reading line: 0 -34 -34 17 17 -34 9 19 24 24 -4 -4
Reading line: 0 -19 -19 9 9 -19 4 14 14 14 -9 -9
Reading line: 0 -4 -4 2 2 -4 4 14 9 9 -14 -14
Reading line: 0 9 9 -4 -4 9 0 9 0 0 -19 -19
Reading line: 0 0 0 0 0 0 0 9 0 0 -19 -19
Reading line: 0 -9 -9 4 4 -9 0 9 0 0 -19 -19
Reading line: 0 -19 -19 9 9 -19 0 9 0 0 -19 -19
Reading line: 0 -29 -29 14 14 -29 0 9 0 0 -19 -19
Reading line: 0 -24 -24 12 12 -24 0 9 0 0 -19 -19
Reading line: 0 -14 -14 7 7 -14 0 9 0 0 -19 -24
```

In [4]:
```
# Convert the list of ECG signals to a NumPy array
data = np.array(data)
```

In [5]:
```
# Check the shape of the data
print(f'Data shape: {data.shape}')  # This should now show (number_of_samples, 5000
```

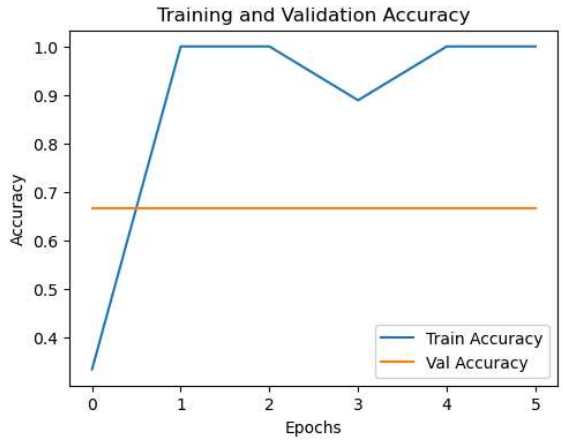
```
Data shape: (12, 5000)
```

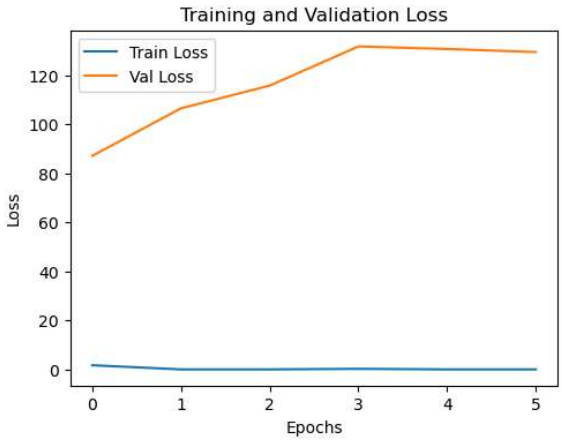
In [6]:

```
# Proceed only if we have valid data
if data.size > 0:
    # Assuming each ECG recording is one sample, and the number of time steps is eq
    X = data.reshape(data.shape[0], data.shape[1], 1)  # Reshape to (samples, time_

    # Generate dummy labels for demonstration (binary classification)
    num_samples = X.shape[0]
    y = np.random.randint(0, 2, num_samples)  # Replace with actual labels

    # Split the dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random

    # Build the CNN model
    model = Sequential()

    # Add the first Conv1D layer
    model.add(Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(X_t
    model.add(BatchNormalization())  # Normalize after conv layer
    model.add(MaxPooling1D(pool_size=2))

    # Add the second Conv1D layer
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling1D(pool_size=2))

    # Flatten the data
    model.add(Flatten())

    # Add a Dense layer
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))  # Dropout for regularization

    # Output layer for binary classification
    model.add(Dense(1, activation='sigmoid'))

    # Compile the model
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'

    # Callbacks for early stopping and reducing learning rate on plateau
    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_wei
    reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3)

    # Train the model
    history = model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data
                        callbacks=[early_stopping, reduce_lr])

    # Plot training and validation accuracy and loss
    plt.figure(figsize=(12, 4))

    # Accuracy plot
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Train Accuracy')
    plt.plot(history.history['val_accuracy'], label='Val Accuracy')
    plt.title('Training and Validation Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()

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

```
    plt.show()

    # Evaluate the model
    test_loss, test_accuracy = model.evaluate(X_test, y_test)
    print(f'Test Accuracy: {test_accuracy}')
else:
    print("No valid ECG data found.")
```

```
C:\Users\diyu2\anaconda3\Lib\site-packages\keras\src\layers\convolutional\base_con
v.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a laye
r. When using Sequential models, prefer using an `Input(shape)` object as the firs
t layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

Epoch 1/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 8s 8s/step - accuracy: 0.3333 - loss: 1.6925 - val_accura
cy: 0.6667 - val_loss: 87.1388 - learning_rate: 0.0010
Epoch 2/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 228ms/step - accuracy: 1.0000 - loss: 1.9229e-14 - val
_accuracy: 0.6667 - val_loss: 106.5229 - learning_rate: 0.0010
Epoch 3/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 214ms/step - accuracy: 1.0000 - loss: 2.8588e-18 - val
_accuracy: 0.6667 - val_loss: 115.7828 - learning_rate: 0.0010
Epoch 4/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 215ms/step - accuracy: 0.8889 - loss: 0.2384 - val_acc
uracy: 0.6667 - val_loss: 131.7661 - learning_rate: 0.0010
Epoch 5/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 220ms/step - accuracy: 1.0000 - loss: 2.5679e-28 - val
_accuracy: 0.6667 - val_loss: 130.7282 - learning_rate: 5.0000e-04
Epoch 6/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 225ms/step - accuracy: 1.0000 - loss: 0.0000e+00 - val
_accuracy: 0.6667 - val_loss: 129.4653 - learning_rate: 5.0000e-04
```






```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - accuracy: 0.6667 - loss: 87.1388
Test Accuracy: 0.6666666865348816
```

In [ ]: