ECG Classification based Time Frequency Analysis and Deep Learning.

By: Mosab Aboidrees Altraifi Yousif and Abdelrahman Mohammed (group 12)

Notes About implementation:

- We did the first implementation using Matlab to investigate data and take the benefits of digital signal processing toolbox.
- We used the transfer learning to investigate the data, and now we build our CNN model.

Data Preparation/Feature Engineering

1. Overview

Data preparation and feature engineering are critical stages in machine learning projects, especially when dealing with time-series data. In this project, we focus on ECG signals, transforming the time-series data into time-frequency representations using Continuous Wavelet Transform (CWT) to leverage the power of deep learning for classification.

2. Data Collection

The dataset is obtained from three PhysioNet databases:

- MIT-BIH Arrhythmia Database (ARR)[1][2]
- MIT-BIH Normal Sinus Rhythm Database (NSR)[1]
- BIDMC Congestive Heart Failure Database (CHF) [1] [3]

The raw ECG signals were sampled at 128 Hz and are stored in a `.mat` file (`ECGData.mat`). Each row corresponds to an ECG signal, and the dataset contains 162 rows, representing 96 arrhythmia (ARR), 30 congestive heart failure (CHF), and 36 normal sinus rhythm (NSR) recordings.

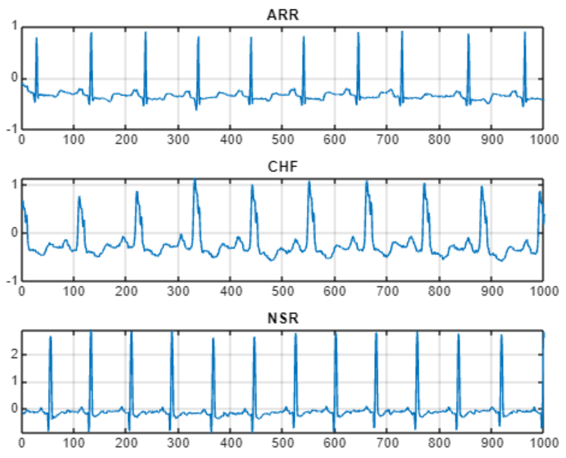
3. Data Cleaning

Since ECG signals are continuous, there were no missing values. The signals were preprocessed to remove noise, and each signal was standardized to have the same length for consistent CWT analysis[4]. The dataset was organized into labeled categories ('ARR', 'CHF', 'NSR').

4. Exploratory Data Analysis (EDA)

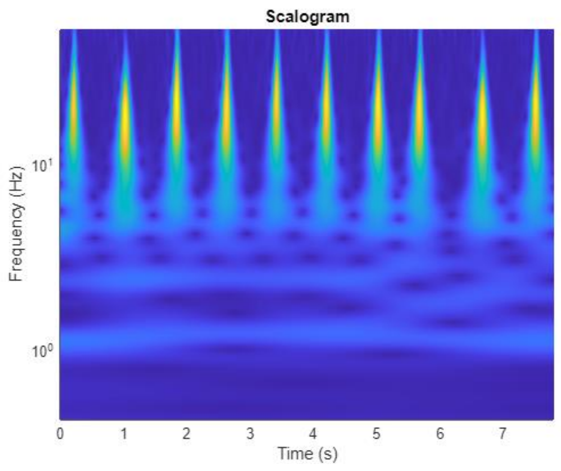
Exploratory Data Analysis was performed by visualizing the time-series signals and their corresponding scalograms (time-frequency representations).

```
% Plotting representative signals from each category
folderLabels = unique(ECGData.Labels);
for k=1:3
    ecgType = folderLabels{k};
    ind = find(ismember(ECGData.Labels,ecgType));
    subplot(3,1,k)
    plot(ECGData.Data(ind(1),1:1000));
    grid on
    title(ecgType)
end
```



This code generates time-series plots of the ECG signals from each class, and then we created scalograms to visualize the time-frequency characteristics.

```
% Visualizing a scalogram for one ECG signal
Fs = 128; % Sampling frequency
sig = ECGData.Data(1,1:1000); % Taking a 1000-sample ECG segment
fb = cwtfilterbank(SignalLength=1000, SamplingFrequency=Fs,
VoicesPerOctave=12);
[cfs,frq] = wt(fb,sig);
t = (0:999)/Fs;
figure;
pcolor(t,frq,abs(cfs)); shading interp; set(gca, 'yscale', 'log');
title("Scalogram of ECG Signal");
xlabel('Time (s)'); ylabel('Frequency (Hz)');
```



The scalograms[4] allow us to capture both time-domain and frequency-domain features, which are crucial for accurate ECG classification.

5. Feature Engineering

The ECG time-series data were transformed into time-frequency representations using CWT[4]. The scalograms were saved as RGB images (224x224x3) to be compatible with deep learning models like GoogLeNet.

```
% Convert ECG signals to scalograms and save as images
imageRoot = fullfile(parentFolder,childFolder);

data = ECGData.Data;
labels = ECGData.Labels;

[~,signalLength] = size(data);

fb = cwtfilterbank(SignalLength=signalLength,VoicesPerOctave=12);
r = size(data,1);

for ii = 1:r
    cfs = abs(fb.wt(data(ii,:)));
    im = ind2rgb(round(rescale(cfs,0,255)),jet(128));

    imgLoc = fullfile(imageRoot,char(labels(ii)));
    imFileName = char(labels(ii))+"_"+num2str(ii)+".jpg";
    imwrite(imresize(im,[224 224]),fullfile(imgLoc,imFileName));
end
```

6. Data Transformation

No further scaling or normalization was necessary since the CNN models handle image inputs, which were standardized to 224x224x3.

Dividing the Data into Training and Validation Sets

```
% Define the path to the directory where the scalogram images are saved
parentDir = data_dir;
dataDir = "data";

allImages = imageDatastore(fullfile(parentDir, dataDir), ...
    "IncludeSubfolders", true, ...
    "LabelSource", "foldernames");

[imgsTrain, imgsValidation] = splitEachLabel(allImages, 0.8, "randomized");

disp("Number of training images: " + num2str(numel(imgsTrain.Files)));
disp("Number of validation images: " + num2str(numel(imgsValidation.Files)));
```

Output:

```
Number of training images: 130
Number of validation images: 32
```

Model Exploration

1. Model Selection

We used **GoogLeNet** pre-trained CNN architecture. These models is well-suited for transfer learning, where we fine-tune it to classify ECG scalograms. GoogLeNet has a deep architecture, making it effective at capturing intricate patterns in the scalograms.

2. Model Training

We replaced the final layers of GoogLeNet with new layers that fit the ECG classification task. We used stochastic gradient descent with momentum (SGDM) and fine-tuned the networks with an initial learning rate of 0.0001 for 20 epochs for GoogLeNet.

```
% Load pretrained GoogLeNet
net = imagePretrainedNetwork("googlenet");

% Modify last layers
numClasses = numel(categories(imgsTrain.Labels));
newConnectedLayer   =   fullyConnectedLayer(numClasses,   "Name",   "new_fc",
"WeightLearnRateFactor", 5, "BiasLearnRateFactor", 5);
net = replaceLayer(net, "loss3-classifier", newConnectedLayer);

% Set training options
options = trainingOptions("sgdm", "MaxEpochs", 20, "InitialLearnRate", 1e-4,
"MiniBatchSize",  15,  "ValidationData",  imgsValidation,  "Plots",  "training-
progress");

% Train the network
trainedGN = trainnet(imgsTrain, net, "crossentropy", options);
```

3. Model Evaluation

The model was evaluated using standard classification metrics. We achieved an accuracy of **94%** on GoogLeNet. A confusion matrix and accuracy plot were generated.

```
% Evaluate model accuracy
accuracy = mean(YPred == imgsValidation.Labels);
disp("GoogLeNet Accuracy: " + num2str(100 * accuracy) + "%");
```

4. Code Implementation

Below are code snippets for model training.

Model Training:

```
% Load and fine-tune GoogLeNet
net = imagePretrainedNetwork("googlenet");
newDropoutLayer = dropoutLayer(0.6, "Name", "new_Dropout");
net = replaceLayer(net, "pool5-drop_7x7_s1", newDropoutLayer);
```

```
% Replace fully connected layer for 3-class classification
newConnectedLayer      =      fullyConnectedLayer(3,      "Name",      "new_fc",
"WeightLearnRateFactor", 5, "BiasLearnRateFactor", 5);
net = replaceLayer(net, "loss3-classifier", newConnectedLayer);

% Set training options and train
options = trainingOptions("sgdm", "InitialLearnRate", 1e-4, "MaxEpochs", 20,
"ValidationData", imgsValidation);
trainedNet = trainnet(imgsTrain, net, "crossentropy", options);
```

This approach provides an effective way to classify ECG signals by leveraging pre-trained deep learning models and wavelet analysis for feature extraction. Let me know if you need any further assistance!

References:

[1] Moody, G. B., and R. G. Mark. "The impact of the MIT-BIH Arrhythmia Database." IEEE Engineering in Medicine and Biology Magazine. Vol. 20. Number 3, May-June 2001, pp. 45–50. (PMID: 11446209)

[2] Goldberger A. L., L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. "PhysioBank, PhysioToolkit,and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals." Circulation. Vol. 101, Number 23: e215–e220. [Circulation Electronic Pages; http://circ.ahajournals.org/content/101/23/e215.full]; 2000 (June 13). doi: 10.1161/01.CIR.101.23.e215.

[3] Baim, D. S., W. S. Colucci, E. S. Monrad, H. S. Smith, R. F. Wright, A. Lanoue, D. F. Gauthier, B. J. Ransil, W. Grossman, and E. Braunwald. "Survival of patients with severe congestive heart failure treated with oral milrinone." Journal of the American College of Cardiology. Vol. 7, Number 3, 1986, pp. 661–670.

[4] Zhao, Q., and L. Zhang. "ECG feature extraction and classification using wavelet transform and support vector machines." In IEEE International Conference on Neural Networks and Brain, 1089–1092. Beijing, China: IEEE, 2005.