Project Design Phase-I
Solution Architecture

Date	19 September 2022
Team ID	PNT2022TMID43406
Project Name	Project - Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation
Maximum Marks	4 Marks

PROJECT DESCRIPTION

An "ambulatory electrocardiogram" or an ECG) about the size of a postcard or digital camera that the patient will be using for 1 to 2 days, or up to 2 weeks. The test measures the movement of electrical signals or waves through the heart. These signals tell the heart to contract (squeeze) and pump blood. The patient will have electrodes taped to your skin. It's painless, although some people have mild skin irritation from the tape used to attach the electrodes to the chest. They can do everything but shower or bathe while wearing the electrodes. After the test period, patient will go back to see your doctor. They will be downloading the information.

TECHNICAL ARCHITECTURE

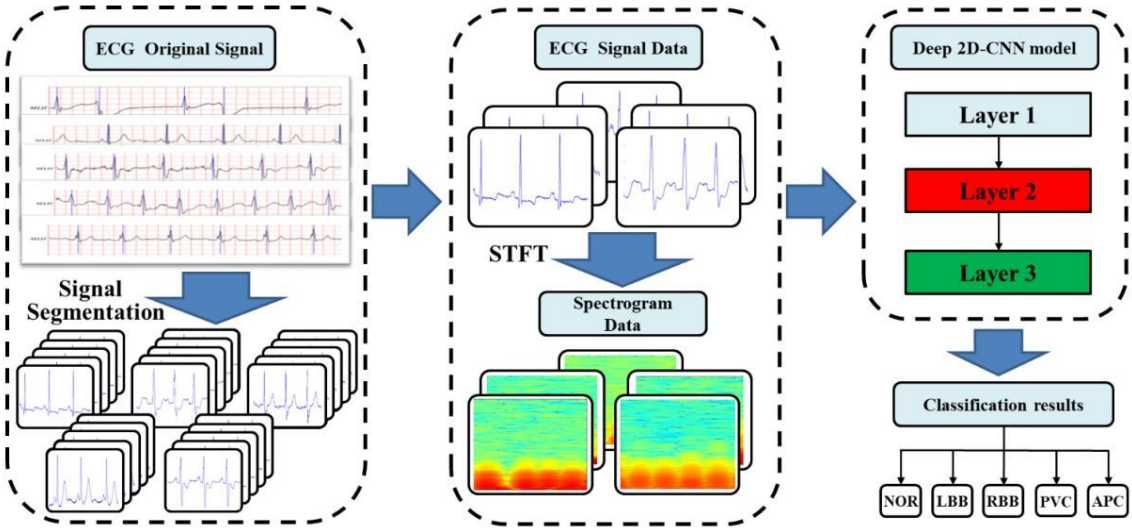

FIGURE 1. Overall procedures in ECG arrhythmia classification based on proposed 2D-CNN

SOLUTION

Hardware Components used: Since we are using the IBM cloud as a platform to execute this project we don't need any hardware components other than our system. Software Components Used: We will be using Anaconda Navigator which is installed in our system and Watson studio from the IBM cloud to complete the project. Anaconda Navigator Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook.

WATSON STUDIO

Watson Studio is one of the core services in Cloud Pak for Data as a Service. Watson Studio provides you with the environment and tools to solve your business problems by collaboratively working with

data. You can choose the tools you need to analyse and visualize data, to cleanse and shape data, or to build machine learning models. This illustration shows how the architecture of Watson Studio is centred around the project. A project is a workspace where you organize your resources and work with data. Watson Studio projects fully integrate with the catalogues and deployment spaces: Deployment spaces are provided by the Watson Machine Learning Service You can easily move assets between projects and deployment spaces.

Experimental Investigations:

In this project, we have deployed our training model using CNN on IBM Watson studio and in our local machine. We are deploying 4 types of CNN layers in a sequential manner, starting from:

Convolutional layer 2D

A 2-D convolutional layer applies sliding convolutional filters to 2-D input. The layer convolves the input by moving the filters along the input vertically and horizontally and computing the dot product of the weights and the input, and then adding a bias term.

Pooling Layer

Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. The pooling layer summarises the features present in a region of the feature map generated by a convolution layer.

Fully-Connected layer

After extracting features from multiple convolution layers and pooling layers, the fully-connected layer is used to expand the connection of all features. Finally, the SoftMax layer makes a logistic regression classification. Fully-connected layer transfers the weighted sum of the output of the previous layer to the activation function.

Dropout Layer

There is usually a dropout layer before the fully-connected layer. The dropout layer will temporarily disconnect some neurons from the network according to the certain probability during the training of the convolution neural network, which reduces the joint adaptability between neuron nodes, reduces overfitting, and enhances the generalization ability of the network. Flow Chart & Results with Screenshots:

Flow Chart & Results by training model in local machine: Dataset Collection: The dataset contains six classes: Left Bundle Branch Block Normal Premature Atrial Contraction Premature Ventricular Contractions Right Bundle Branch Block Ventricular Fibrillation Image Pre-processing: Image Pre-processing includes the following main tasks Import ImageDataGenerator Library: Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class.

Image shifts via the width_shift_range and height_shift_range arguments. Image flips via the horizontal_flip and vertical_flip arguments. Image rotates via the rotation_range argument Image brightness via the brightness_range argument. Image zooms via the zoom_range argument.

An instance of the ImageDataGenerator class can be constructed for train and test.

Applying ImageDataGenerator functionality to the trainset and test set: We will apply ImageDataGenerator functionality to Trainset and Testset by using the following code

This function will return batches of images from the subdirectories Left Bundle Branch Block, Normal, Premature Atrial Contraction, Premature Ventricular Contractions, Right Bundle Branch Block and Ventricular Fibrillation, together with labels 0 to 5{'Left Bundle Branch Block': 0, 'Normal': 1, 'Premature Atrial Contraction': 2, 'Premature Ventricular Contractions': 3, 'Right Bundle Branch Block': 4, 'Ventricular Fibrillation': 5}

We can see that for training there are 15341 images belonging to 6 classes and for testing there are 6825 images belonging to 6 classes.

Model Building We are ready with the augmented and pre-processed image data, we will begin our build our model by following the below steps: Import the model building Libraries:

Initializing the model: Keras has 2 ways to define a neural network: Sequential Function API The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model. In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add () method. Now, will initialize our model. Adding CNN Layers: We are adding a convolution layer with an activation function as “relu” and with a small filter size (3,3) and a number of filters as (32) followed by a max-pooling layer.

The Max pool layer is used to down sample the input.

The flatten layer flattens the input.

Adding Hidden Layers: Dense layer is deeply connected neural network layer. It is most common and frequently used layer.

Adding Output Layer:

Understanding the model is very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the model and its layers.

Configure the Learning Process: The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find error or deviation in the learning process. Keras requires loss function during the model compilation process. Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer Metrics is used to evaluate the performance of your model. It is similar to loss function, but not used in the training process.

Training the model: We will train our model with our image dataset. fit_generator functions used to train a deep learning neural network.

Saving the model: The model is saved with .h5 extension as follows An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

Testing the model: Load necessary libraries and load the saved model using load_model Taking an image as input and checking the results Note: The target size should for the image that is should be the same as the target size that you have used for training.

The unknown image uploaded is:

Here the output for the uploaded result is normal.

Application Building: In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has uploaded an image. The uploaded image is given to the saved model and prediction is showcased on the UI. This section has the following tasks Building HTML Pages: We use HTML to create the front-end part of the web page. Here, we created 4 html pages- home.html, predict_base.html, predict.html, information.html.

home.html displays the home page.

information.html displays all important details to be known about ECG.

predict-base.html and predict.html accept input from the user and predicts the values.

Building server-side script: We will build the flask file 'app.py' which is a web framework written in python for server-side scripting. The app starts running when the "**name**" constructor is called in main. render_template is used to return HTML file. "GET" method is used to take input from the user. "POST" method is used to display the output to the user.

Running The App:

Navigate to the localhost you can view your web page.

5.2 Flow Chart & Results by training model in IBM WATSON STUDIO: Creating IBM cloud account: We have to create an IBM Cloud Account and should log in. Creating Watson Studio Service & Machine Learning Service:

Create a Project & Deployment space in the Watson studio:

d. Upload The dataset and create a jupyter source file in the created project:

e. Apply CNN algorithm and save the model and deploy it using API key generated:

f. For downloading the model we have to run the last part of the above code in the local jupyter notebook:

g. Now we will extract the .h5 model file and will do the app deployment using flask as done in the previous training:

Hence, we trained the model using IBM Watson.

Advantages & Disadvantages

Advantages: The proposed model predicts Arrhythmia in images with a high accuracy rate of nearly 96% The early detection of Arrhythmia gives better understanding of disease causes, initiates therapeutic interventions and enables developing appropriate treatments. 6.2 Disadvantages: Not useful for identifying the different stages of Arrhythmia disease. Not useful in monitoring motor symptoms

Applications:

It is useful for identifying the arrhythmia disease at an early stage. It is useful in detecting cardiovascular disorders. Conclusion: Cardiovascular disease is a major health problem in today's world. The early diagnosis of cardiac arrhythmia highly relies on the ECG. Unfortunately, the expert level of medical resources is rare, visually identify the ECG signal is challenging and time-consuming. The advantages of the proposed CNN network have been put to evidence. It is endowed with an ability to effectively process the non-filtered dataset with its potential anti-noise features. Besides

that, ten-fold cross-validation is implemented in this work to further demonstrate the robustness of the network.