****

**ECG Arrhythmia Classification using Deep Learning**

In this tutorial, you will learn to apply image classification using Convolutional Neural Networks(CNNs), trained on the DBC network.

For executing the steps in the tutorial, you will only be required to install DBC on your computing device since all other dependencies including keras and OpenCV will be covered by a pre-configured, GPU-compatible [docker container image.](https://www.docker.com/resources/what-container)

The instructions assume a Linux environment.

Prerequisites:

* Knowledge of basic Linux command line instructions
* High level understanding of supervised learning and CNNs

**1) Motivation**

According to the World Health Organization (WHO), cardiovascular diseases (CVDs) are the number one cause of death today. Over 17.7 million people died from CVDs which is an about 31% of all deaths, and over 75% of these deaths occur in low and middle income countries. Arrhythmia is a representative type of CVDs that refers to any irregular change from the normal heart rhythms. There are several types of arrhythmia including atrial fibrillation, premature contraction, ventricular fibrillation, and tachycardia. Although single arrhythmia heartbeat may not have a serious impact on life, continuous arrhythmia beats can result in fatal circumstances. For example, prolonged premature ventricular contractions (PVCs) beats occasionally turn into a ventricular tachycardia (VT) or a ventricular fibrillation (VF) beats which can immediately lead to the heart failure. Thus, it is important to periodically monitor the heart rhythms to manage and prevent the CVDs. Electrocardiogram (ECG) is a non-invasive medical tool that displays the rhythm and status of the heart. Therefore, automatic detection of irregular heart rhythms from ECG signals is a significant task in the field of cardiology. [[1]](https://arxiv.org/abs/1804.06812)

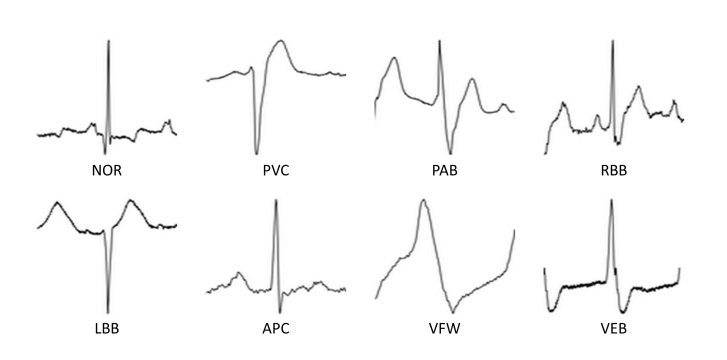
**2) Approach**

We will be using the approach used in: 'ECG arrhythmia classification using a 2-D convolutional neural network' (from here on, referred to as 'the paper')

This approach converts a dataset of ECG signals from the **MIT-BIH Arrhythmia database** into a set of 128x128 grayscale ECG images. A 2D Convolutional Neural Network is then trained to classify the images between 1 of 8 classes of arrhythmias.

**3) Dataset**

In the interest of time, the conversion from signal to over a 100K, 128x128 images has been done for you. A few example images are shown below:



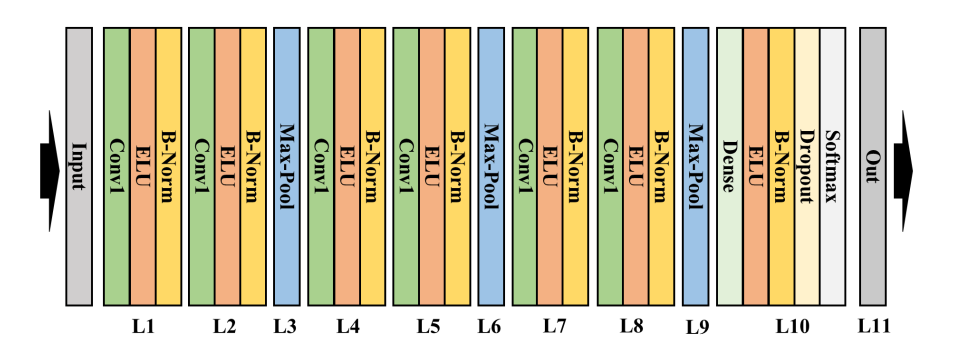
As you can see, there are a total of 8 classes, consisting of 7 arrhythmias and a normal beat class (NOR). Each class is in a separate folder in the dataset.

Download the dataset by running the following command from a terminal window on your computing device:

**$ wget www.deepbrainchain.com/ecg\_dataset**

**4) The Model**

The model we are using is a 2D CNN consisting of stacked layers including convolutional layers, pooling layers and Batch Normalization Layers, among others. The paper shows it outperforms AlexNet and VGG16 architectures.



**5) Training the Model using the DBC network**

We will be training the model using the DBC network of GPUs.

1) Firstly, download the latest version of dbc using the instructions provided in the DBC [AI Manual](https://github.com/DeepBrainChain/deepbrainchain-release/releases/download/0.3.4.0/DBC.AI.Users.Manual_En.pdf)

2) Login to the DBC website and purchase a machine for the desired duration. We used 2 GPUs for 32 hours

3) Enter the machine id for your GPU machine in the task.conf file as per the AI Manual

# **6) Uploading the dataset and code directory**

Run the following in your terminal:

**$ cd /path\_to\_dbc\_directory/dbc\_repo/tool**

**$ bash ./dbc\_upload /path\_to\_data\_directory/**

Copy the resulting DIR\_HASH and enter it in the 'dataset\_dir' entry in the task.conf file.

Next run the following:

**$ bash ./dbc\_upload /path\_to\_code\_directory/**

# **7) Submitting the task file for training**

# In your DBC terminal, submit your completed task file by executing the following:

**$ start -c /path/to/file/task.conf**

The task file specifies an execution file run.sh that executes the training script train\_keras.py when the DBC task is initiated.

**8) Checking the status of the project**

Copy the node\_id for your task. It can be used to check the status via the latest standard output prints that are generated.

Run the following in your dbc terminal to check the status:

**$ logs -t node\_id**

# **9) Downloading and reviewing the results**

When the list -a command shows 'job completed', it means that your task has been completed. Well done!

Now use the following command to download the results of your task for analysis:

**$ result -t node\_id -o /custom\_save\_path**

**10) Assignments**

**[10 pts] Q1 - Visualize results**

Visualizations offer a human readable views to understand the behavour of a model. They are

often used in papers for comparisions between several iterations or different models.

After the downloading the results from training, you will find the file ‘trainHistory.pkl’ which is a record of a [History](https://keras.io/callbacks/" \l "history) object created by Keras.

Plot the loss vs epochs from the history object using the python library Matplotlib, or any other library you prefer.

**[10 pts] Q2 - CNN Layers**

Section ’The Model’ in this tutorial names 3 types of layers used in the chosen architecure. Describe the function of each layer.

**[30 pts] Q3 - Data Augmentation**

Data augmentation is a tried and tested method to expand the size of a dataset, and improve

performance of a trained network.

The paper [1] suggests using 9 different cropping methods, randomly during the training session.

Modify the function generator() in DataGenerator.py to augment the dataset randomly during

training.

You may use the [medium](https://medium.com/datadriveninvestor/ecg-arrhythmia-classification-using-a-2-d-convolutional-neural-network-33aa586bad67) post as a hint on impolementing the 9 cropping functions

**[50 pts] Q3 - Different Models (Challenge Question!)**

Modify the model model\_keras.py and test an different model for training.

Here are a few examples:

* AlexNet
* VGG16
* ResNet50
* InceptionV3

An important skill in deep learning is to try different approaches and benchmark results.

Train your custom model and answer the following questions:

Did your results improve or not? What could be the reason for improvement or deterioration

of the results?