Project Description

Project Description:

- **Datasets (taken from kaggle)**:
  - **Arrhythmia Dataset:**
    - two different files (mitbih_train and mitbih_test)
    - 109446 samples and 5 classes ['N': 0, 'S': 1, 'V': 2, 'F': 3, 'Q': 4]
    - [N: Normal beat, S: Supraventricular premature beat, V: Premature ventricular contraction, F: Fusion of ventricular and normal beat, Q: Unclassifiable beat]
  - **The PTB Diagnostic ECG Database:**
    - two different files (ptbdb_abnormal and ptbdb_normal)
    - 14552 samples and 2 classes (MI or normal)
  - All the samples are cropped, downsampled and padded with zeros if necessary to the fixed dimension of 188.
  - This dataset consists of a series of CSV files. Each of these CSV files contain a matrix, with each row representing an example in that portion of the dataset. The final element of each row denotes the class to which that example belongs.

Project Description:

- **Baselines**:
  - https://github.com/CVxTz/ECG_Heartbeat_Classification/blob/master/code/baseline_mitbih.py
  - https://github.com/CVxTz/ECG_Heartbeat_Classification/blob/master/code/baseline_ptbdb.py
  - Unlike proposed model, these implementations doesn't contain residual blocks.
  - Don't take the transfer learning code of that repository.
- **Training / Testing split:**
  - **Important:** Use code of the baselines, see next slide

Train/Test split:

Arrhythmia Dataset

```
df_train = pd.read_csv("../input/mitbih_train.csv", header=None)
df_train = df_train.sample(frac=1)
df_test = pd.read_csv("../input/mitbih_test.csv", header=None)

Y = np.array(df_train[187].values).astype(np.int8)
X = np.array(df_train[list(range(187))].values)[..., np.newaxis]

Y_test = np.array(df_test[187].values).astype(np.int8)
X_test = np.array(df_test[list(range(187))].values)[..., np.newaxis]
```

https://github.com/CVxTz/ECG_Heartbeat_Classification/blob/master/code/baseline_mitbih.py

The PTB Diagnostic ECG Database

```
df_1 = pd.read_csv("../input/ptbdb_normal.csv", header=None)
df_2 = pd.read_csv("../input/ptbdb_abnormal.csv", header=None)
df = pd.concat([df_1, df_2])

df_train, df_test = train_test_split(df, test_size=0.2, random_state=1337, stratify=df[187])


Y = np.array(df_train[187].values).astype(np.int8)
X = np.array(df_train[list(range(187))].values)[..., np.newaxis]

Y_test = np.array(df_test[187].values).astype(np.int8)
X_test = np.array(df_test[list(range(187))].values)[..., np.newaxis]
```

https://github.com/CVxTz/ECG_Heartbeat_Classification/blob/master/code/baseline_ptbdb.py

TASKS:

1. Solve both datasets with vanilla **RNNs** and **CNNs**.
    - For the binary one, report accuracy, AUROC and AUPRC.
    - For the non-binary one, report accuracy.
    - Compare with baseline.
2. Implement between one to three **additional models** for both datasets (bidirectional LSTM, CNN with residual blocks, autoencoder, transformers etc), be creative!
   How many? Depends on the complexity!
3. Implement (two) **ensemble approaches** (e.g. average of the outputs, logistic regression on the outputs, etc).

4. OPTIONAL: Use **Transfer Learning** for the selected models.

Transfer learning options

1. Transfer learning with RNNs, frozen base model.
   - Train RNN model with MIT-BIH Arrhythmia Database.
   - Remove output layer(s) from model (https://keras.io/getting-started/faq/#how-can-i-remove-a-layer-from-a-sequential-model) and feed it with the PTB Diagnostic ECG Database. The outputs are the vector representation of the PTB Diagnostic ECG Database samples.
   - Take the representations of the samples of PTB Diagnostic ECG Database and train a new small feedforward neural network equivalent to the layers you removed from the first model.

2. Transfer learning with RNNs, re-training whole model.
   - Train RNN model with MIT-BIH Arrhythmia Database.
   - Remove output layer(s) from model (https://keras.io/getting-started/faq/#how-can-i-remove-a-layer-from-a-sequential-model)
   - Add output layer(s) for the PTB Diagnostic ECG Database and train the whole model.

Transfer learning options

3. Transfer learning with RNNs, first frozen base model, then re-training whole model.
   - Train RNN model with MIT-BIH Arrhythmia Database.
   - Remove output layer(s) from model (https://keras.io/getting-started/faq/#how-can-i-remove-a-layer-from-a-sequential-model).
   - Freeze the layers of the model (https://keras.io/getting-started/faq/#how-can-i-freeze-keras-layers)
   - Add output layer(s) for the PTB Diagnostic ECG Database and train the output layers.
   - Unfreeze the whole model.
   - Train the whole model.

What you are given

- Datasets (download from kaggle).
- Baseline code, containing both baseline results and data splits you have to use.

Deliverables

- Solve all Tasks.
- **Report** of max. 2 pages (+ 1 or 2 pages for figures if needed).
- **Code** with conda environment and README (you can also use jupyter notebooks).
  - Advice: Use Keras functional API
- Do not hardcode any results! We will run your code.
- Again: sequential execution and **reproducibility**
- **Do not copy solutions from previous projects!** We are aware of all existing solutions on github. We run code similarity checks and check for plagiarism in the reports from previous years solutions. Any plagiarism will result in a 0 grade for all projects.
- **Deadline**: 29.03.2022

Grade

- To grade the project we will focus (on equal part) to:
  - the content, organisation, clarity, quality and writing of the **final report**
  - the quality of the **implementation** (reproducibility and clarity)
  - the **performance/creativity** of the methods used to solve the tasks, and the reasons behind the choice.
- The **prerequisites** to get the maximum grade are:
  - write a clear and **good report**
  - submit a **clean code** with **easy instructions** on how to reproduce each result of the report
  - solve every task and at least one **optional task** with **well-justified** methods
  - bonus: implement **creative** models for one task (see below).