
Automatic pathology detection from PCG signals

BARIS BOZKURT

University of Crete
June 2017

Overview

- 1. Problem definition*
- 2. Database collection*
- 3. Processes involved in building an automatic system*
- 4. Software tool developed*

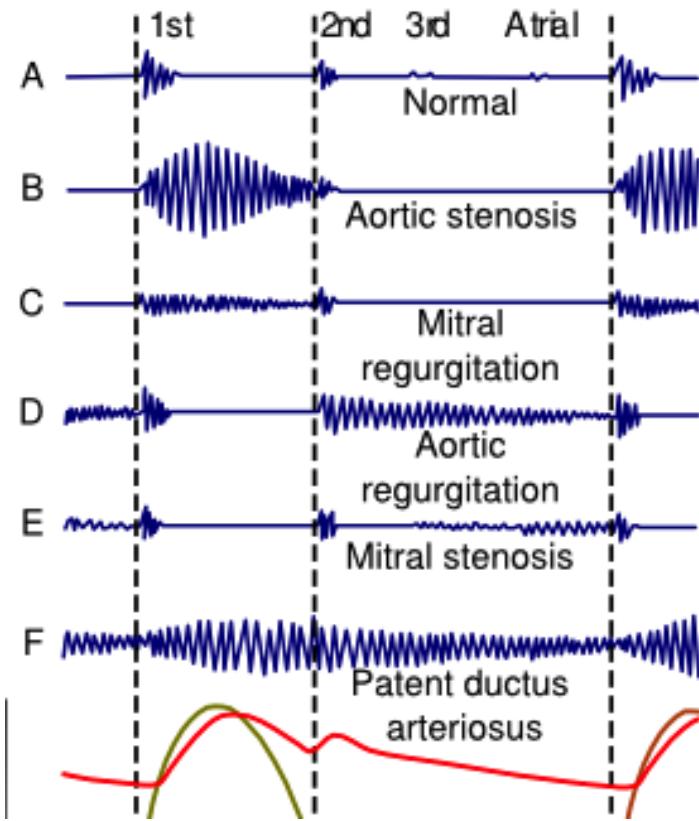
Problem definition

Detection of heart defect risk via processing heart sounds (Phonocardiogram (PCG))

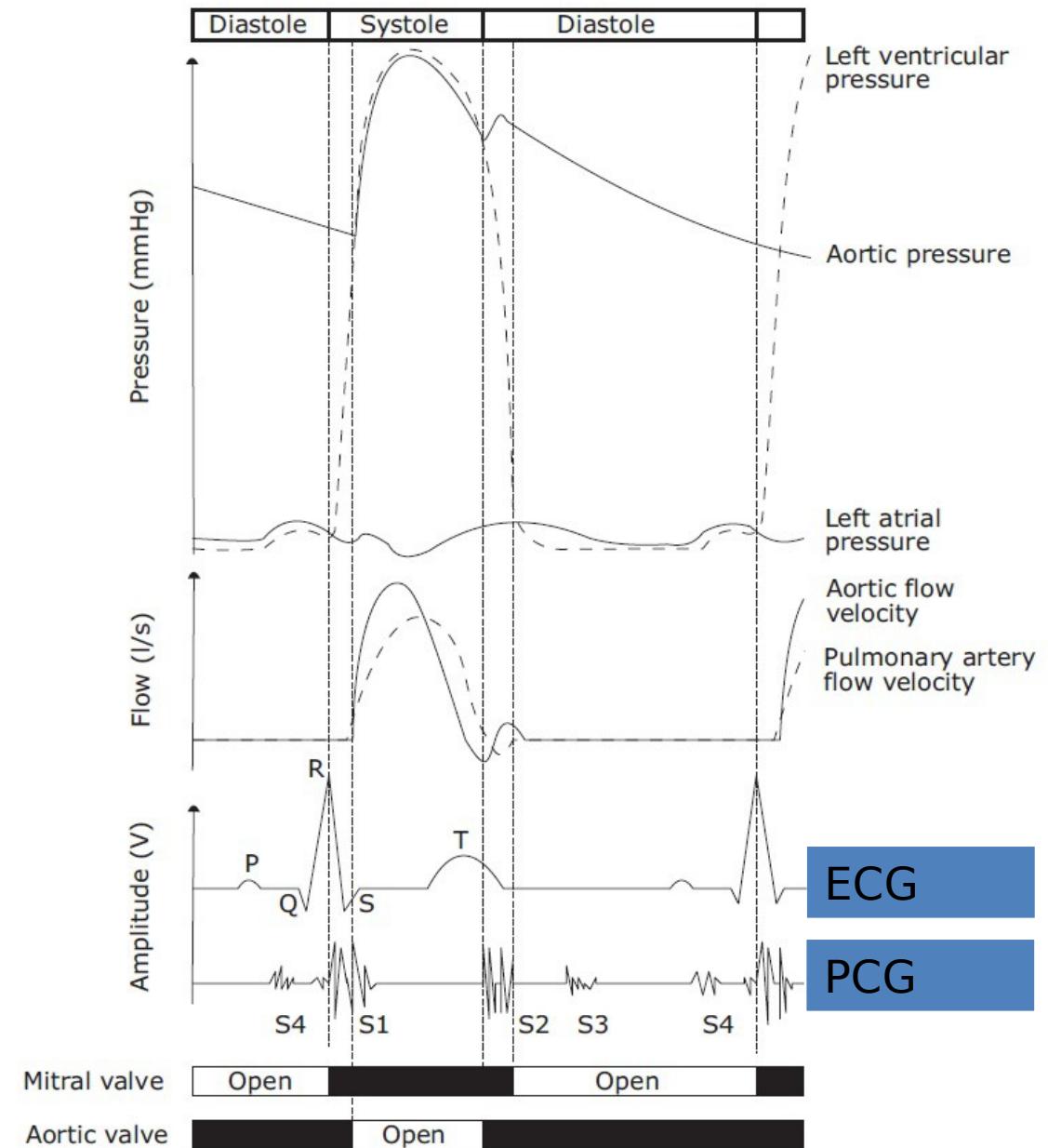
Classes:

- Normal and abnormal
- With murmur, without murmur
- Innocent murmur, pathological murmur

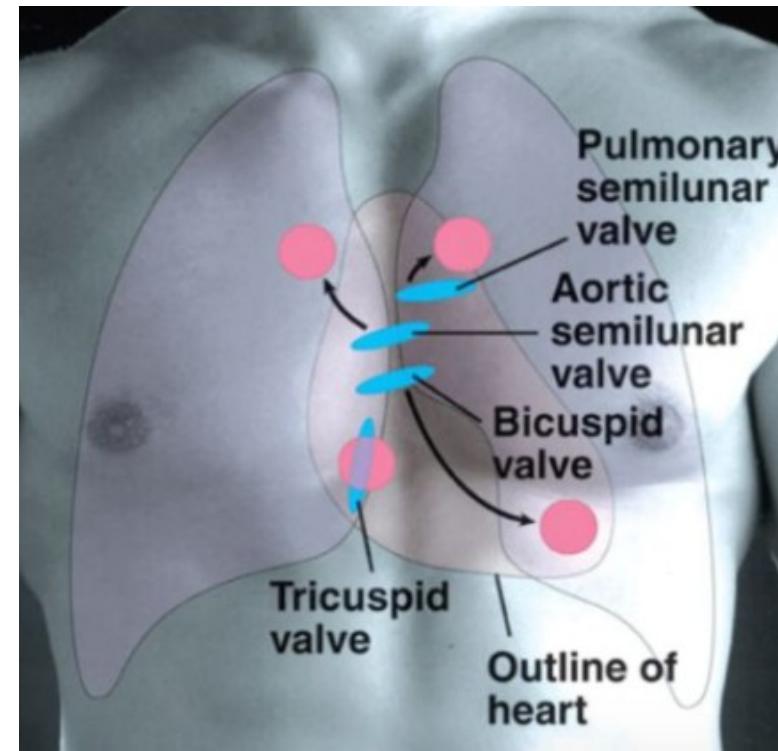
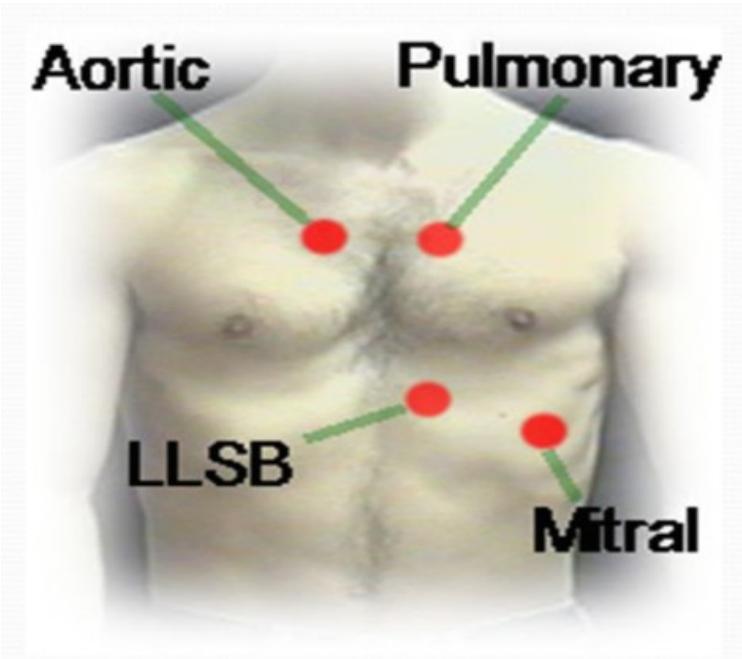
Heart sounds versus murmurs



<http://www.textbookofcardiology.org>



Database collection



**Murmurs are not audible in all recordings.
Expert labels 'single best'**

...our database includes one more recording location



UoC Pcg database

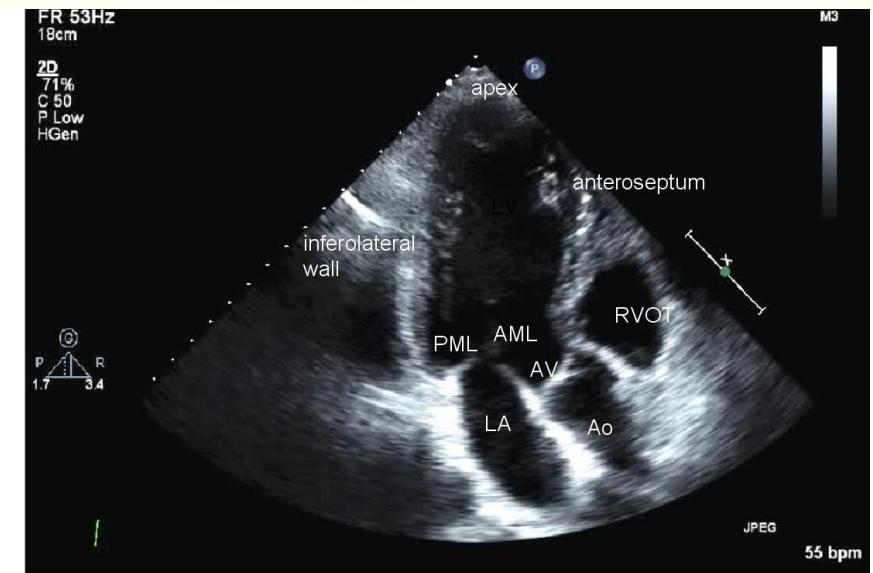
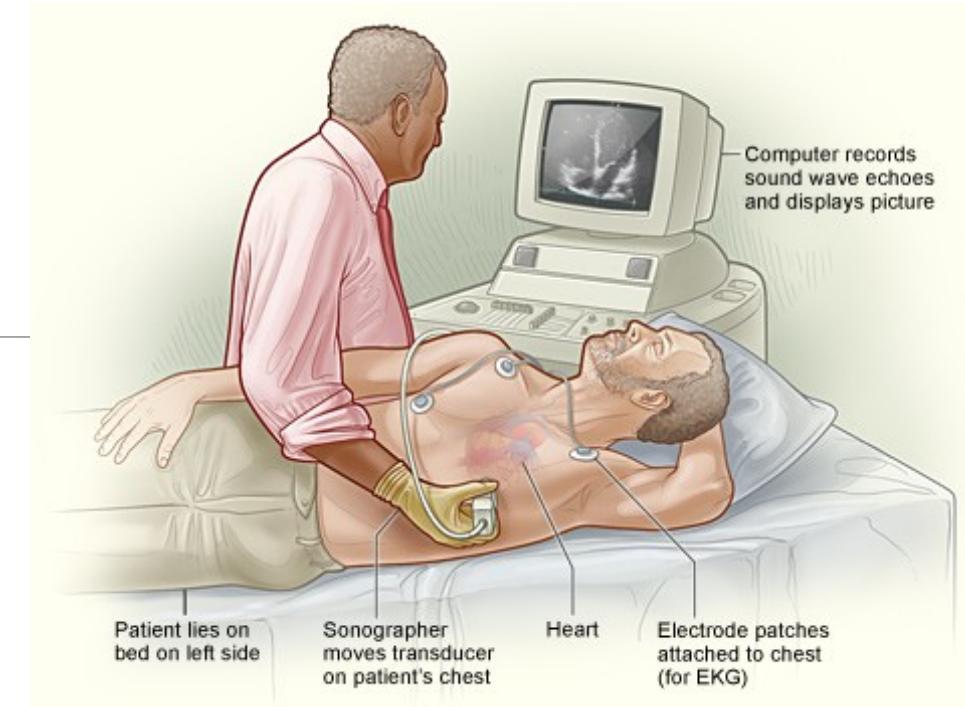
Three categories:

No murmur, innocent/musical murmur,
pathological murmur

Recordings from 5 locations, 1
selected

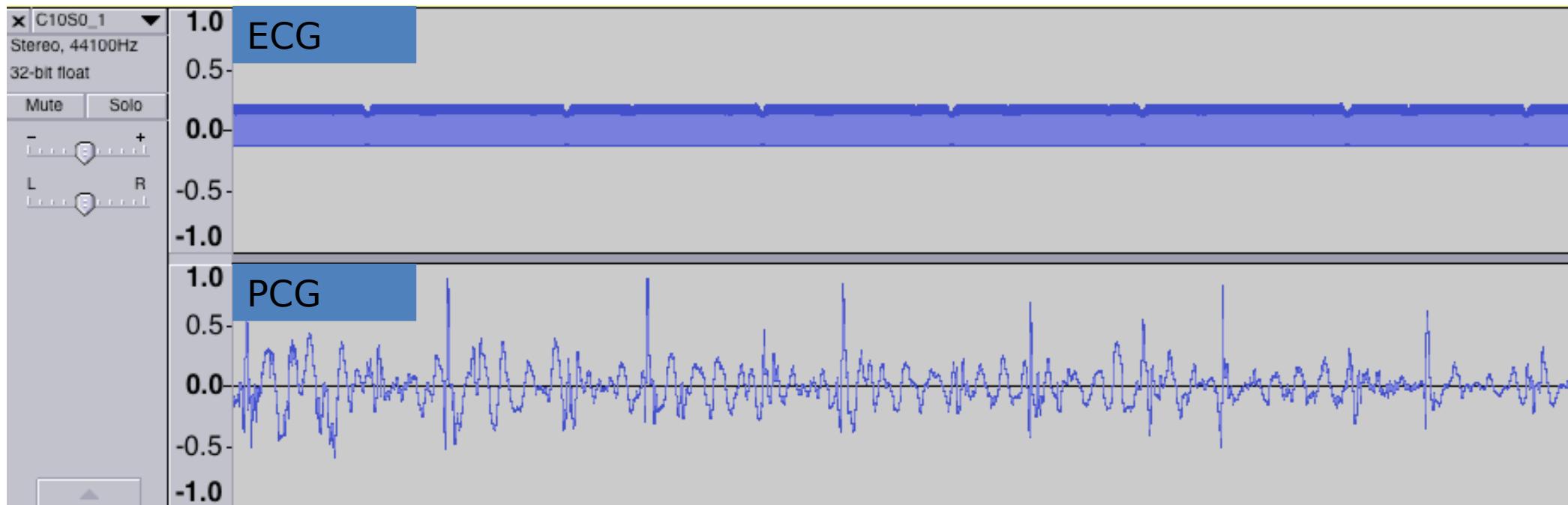
Validation:

- Echocardiogram
- Audio validation during consultation

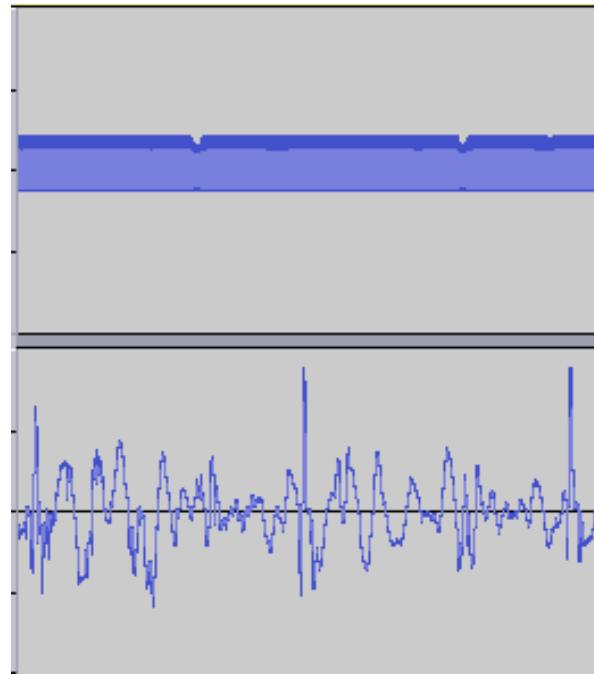


UoC Pcg database

Most devices would use proprietary coding schemes and bound users to a specific user interface and data analysis tool

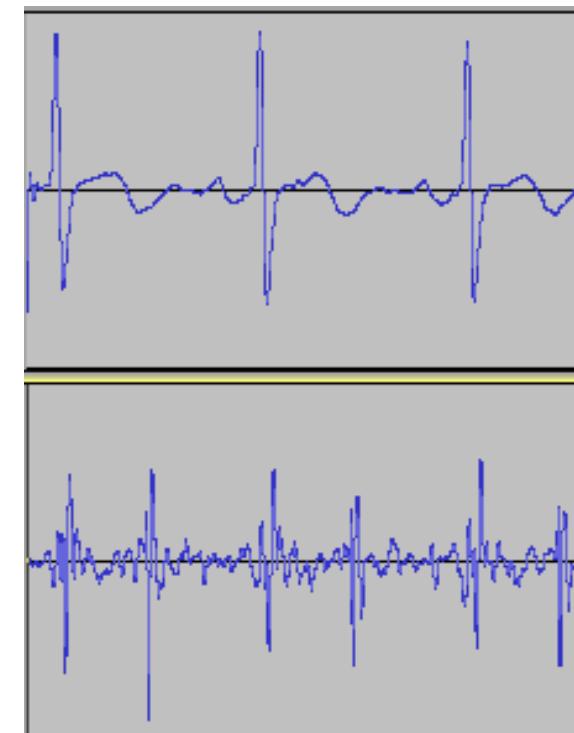


Processes involved in building an automatic system



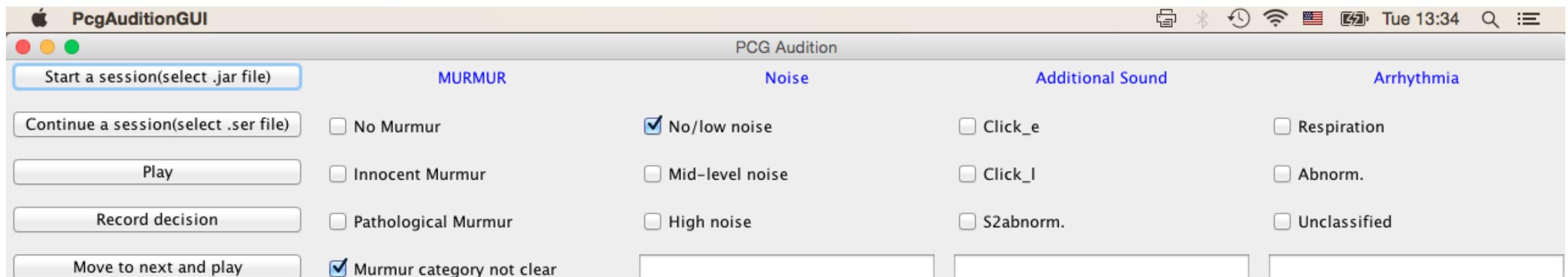
Pre-processing

- Separate channels
 - Down-sample
 - LP Filter
 - Re-group for classification tests



!! All Phyton libraries are not reliable for signal processing. Scipy-resampling introduces artifacts

Blind validation



Tool
implemented
in Java

report_validation.txt			
\WaveFiles\C13S7_mono.wav	Mur:1	Noise:0	AddSound:000
\WaveFiles\C11S54_mono.wav	Mur:0	Noise:1	AddSound:000
\WaveFiles\C15S7_mono.wav	Mur:0	Noise:1	AddSound:000
\WaveFiles\C12S16_mono.wav	Mur:0	Noise:1	AddSound:000
\WaveFiles\C11S18_mono.wav	Mur:0	Noise:1	AddSound:000
\WaveFiles\C15S23_mono.wav	Mur:1	Noise:1	AddSound:000
\WaveFiles\C12S37_mono.wav	Mur:0	Noise:1	AddSound:000
\WaveFiles\C13S28_mono.wav	Mur:1	Noise:1	AddSound:000
\WaveFiles\C13S33_mono.wav	Mur:0	Noise:1	AddSound:000
\WaveFiles\C16S12_mono.wav	Mur:0	Noise:1	AddSound:000
\WaveFiles\C13S21_mono.wav	Mur:0	Noise:2	AddSound:000
\WaveFiles\C15S12_mono.wav	Mur:1	Noise:0	AddSound:000
\WaveFiles\C11S43_mono.wav	Mur:1	Noise:0	AddSound:000

Experts
feedback:
sounds
including high-
noise and/or
clicks should be
excluded

Final database after duplicate removals

TRAIN

	NoMur	MurMus	MurPat	Total
NumWav	719	336	130	1185
NumPer	252	327	117	696

Files re-grouped and stored in folders automatically:
trainNoMur, trainMurMus,
trainMurPat ... as two channels: ecg, pcg

TEST

(other position recordings of patients in TRAIN)

	NoMur	MurMus	MurPat	Total
NumWav	24	569	170	763
NumPer	8	183	58	249

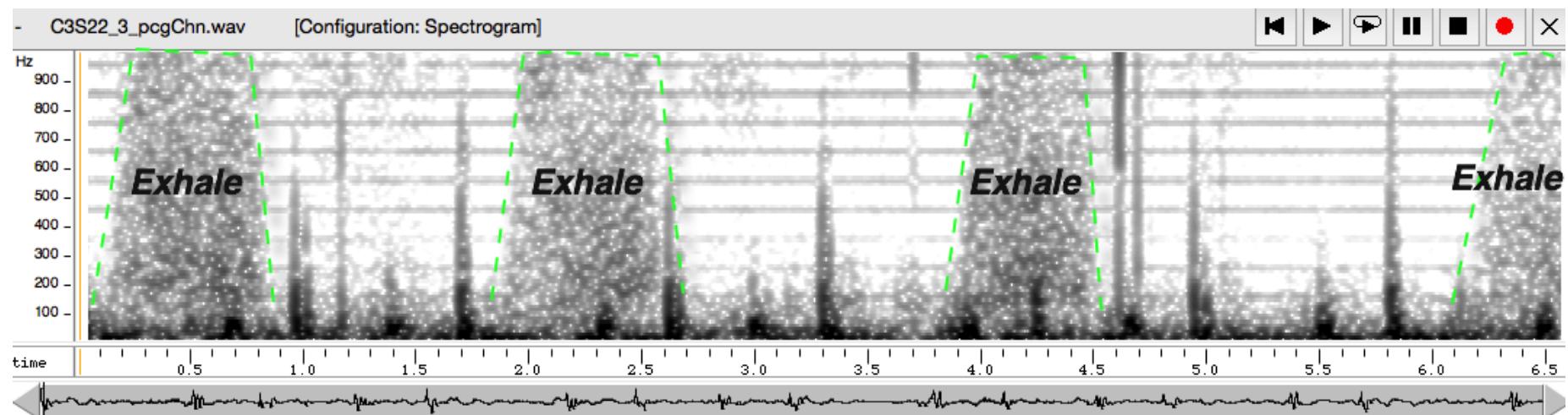
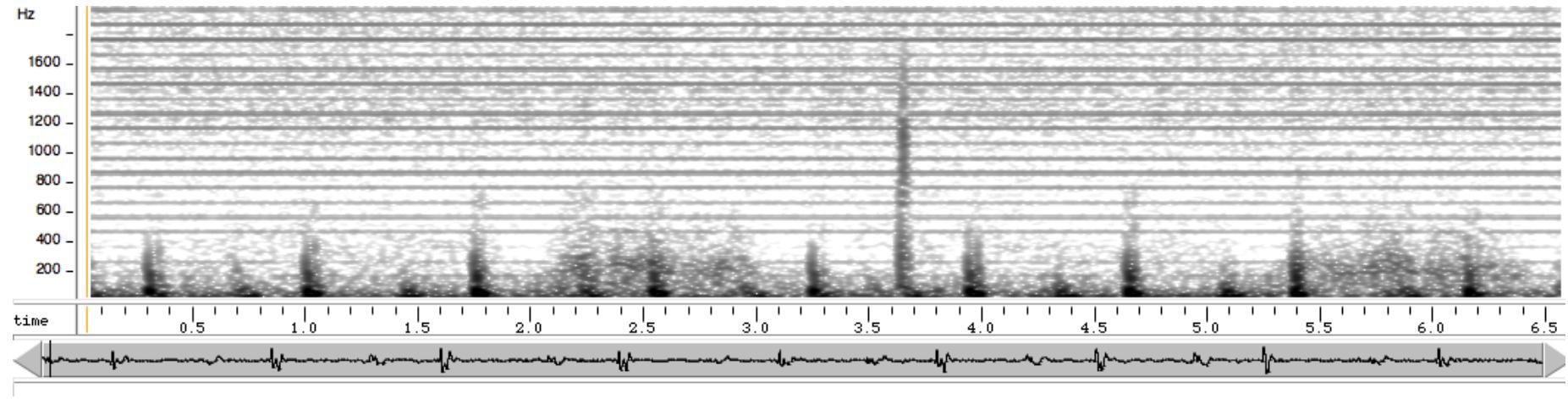
Most reliable part of the database:
31.8Mbytes
(after re-samp.)

Not used due to lack of blind audio validation

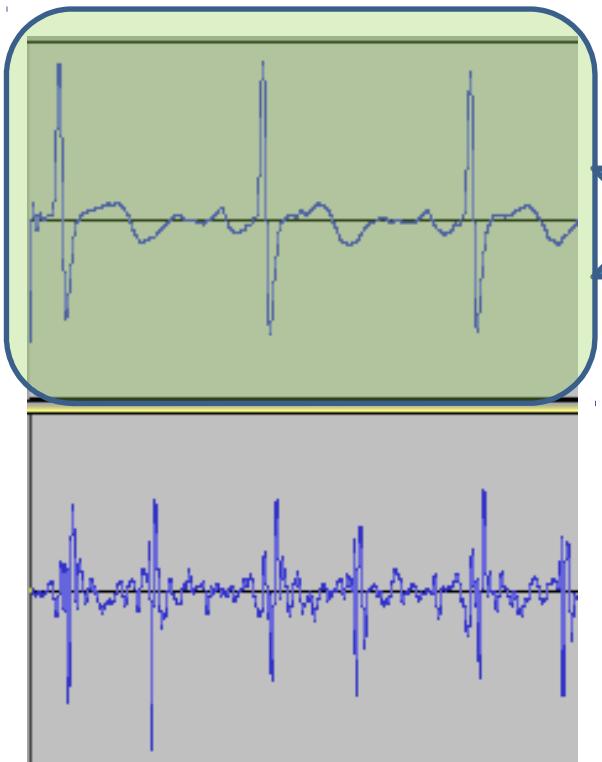
Noise

Various types
of noise exist

In existence of
breath,
murmurs may
be inaudible.
Breath has
similar signal
characteristics
as murmur

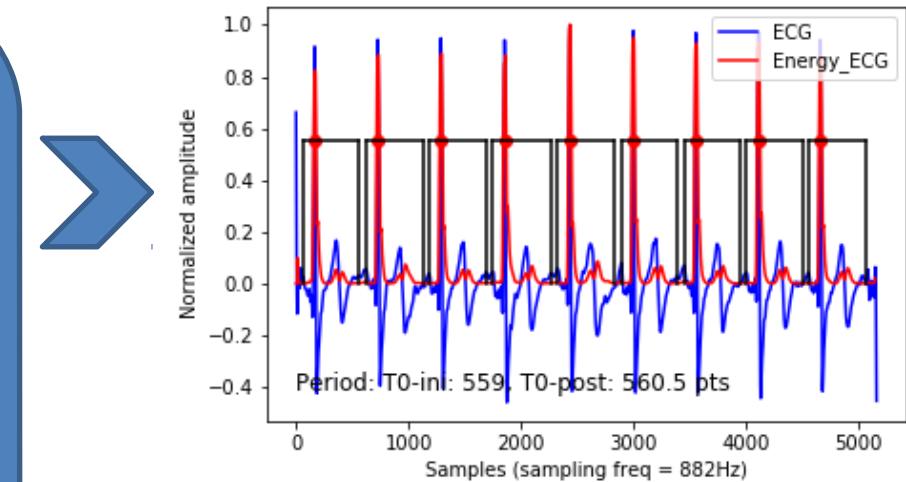


Segmentation using ECG signals



Segmentation

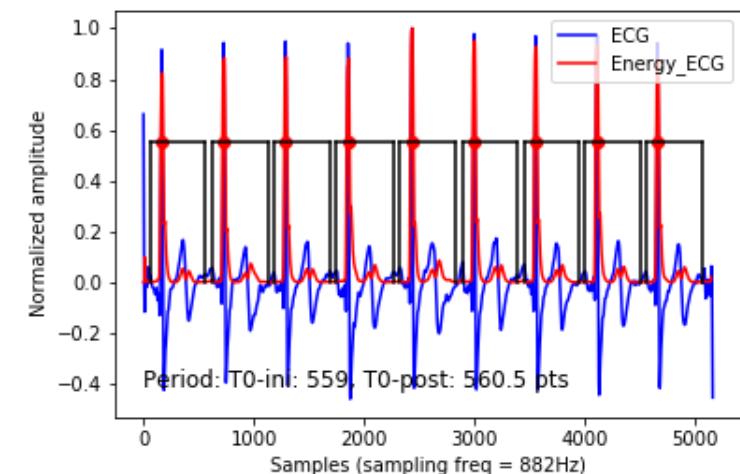
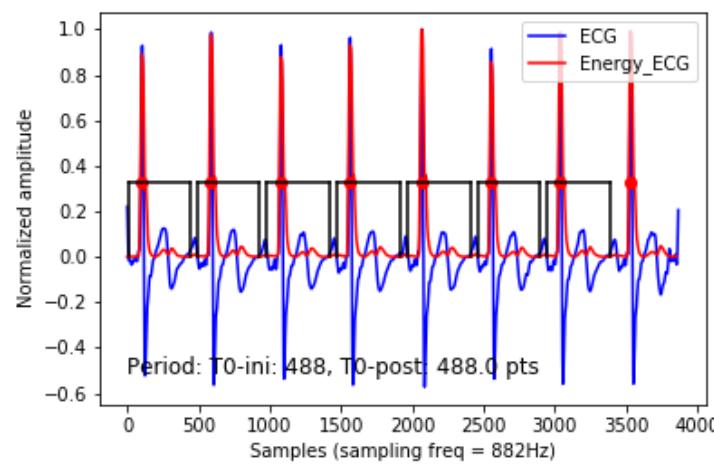
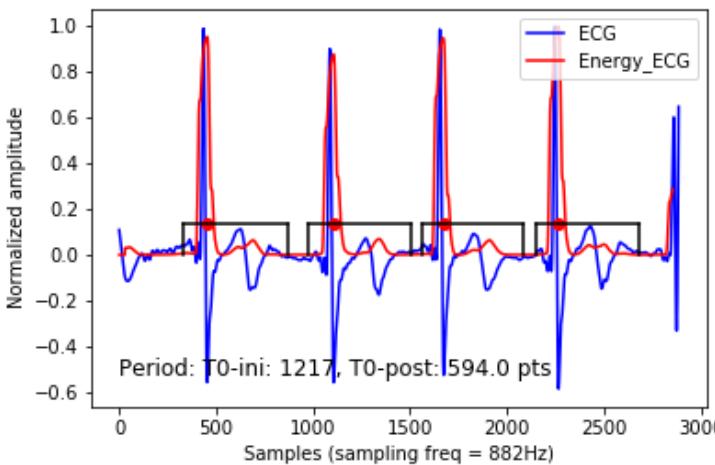
- Compute energy
- Re-shape to enhance peaks
- Detect period (auto-corr.)
 - Estimate # of periods
- Detect $3 * \#$ peaks via thresh.
 - Filter out spurious peaks



Different strategies may be applied for framing:

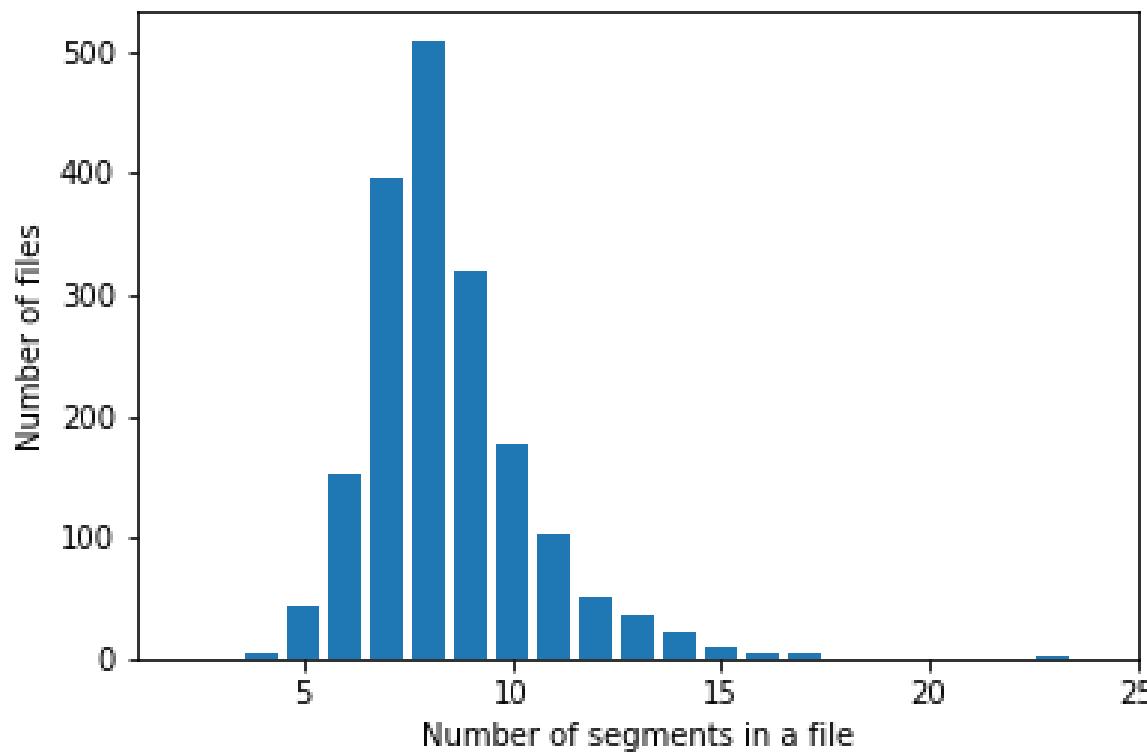
- Syncr. Frames (1/2/3 period len)
- Syncr. Frames (0.5/1/2 sec. len)
- ASyncr. Frames (0.5/1/2 sec. len)

Processes involved in building an automatic system: Period marking from ECG signals [examples]



- Window-based amplitude normalization applied to get rid-off sudden energy jumps
- Reliable in most of the cases

Processes involved in building an automatic system: Period marking from ECG signals [examples]



On average there are 8 periods per file

Feature extraction

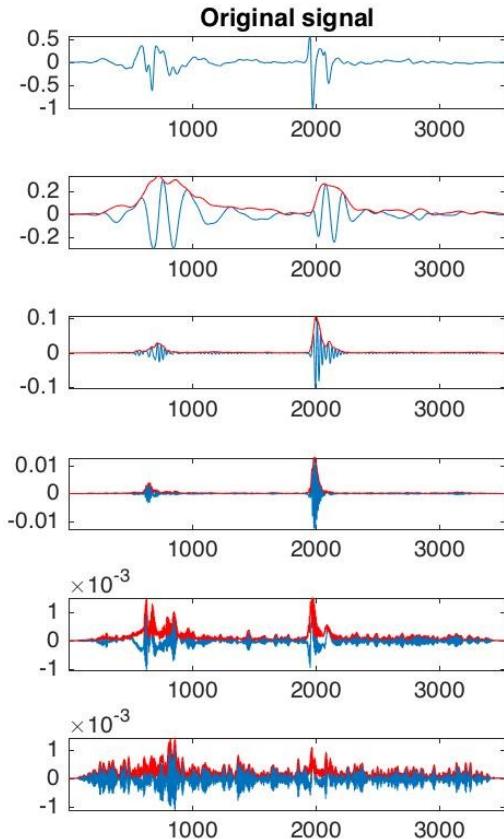
Frame-level feature computation:

- MFCC (with/without delta coefficients) [librosa]
- Mel-Spectrum (with/without delta coefficients) [librosa]
- Sub-band envelopes:
 - Use auditory filter banks to create sub-band filters
 - Filter the signal to get sub-band signals
 - Compute (Hilbert)envelopes of sub-band signals and re-sample
- Sub-band signals
 - The first two steps above

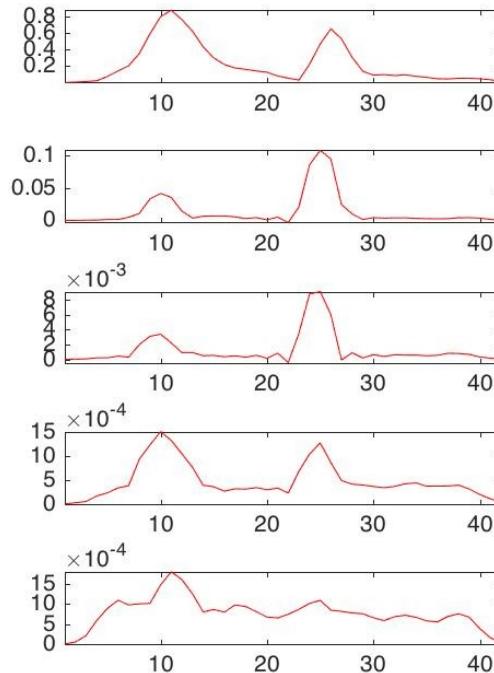
Processes involved in building an automatic system

Feature computation: Sub-band signal envelopes

Sub-band 1
Sub-band 2
Sub-band 3



Re-sampled envelopes



In a later step,
logarithmic
compression is
applied since
energy is generally
too low in higher
bands

CNN architectures tested

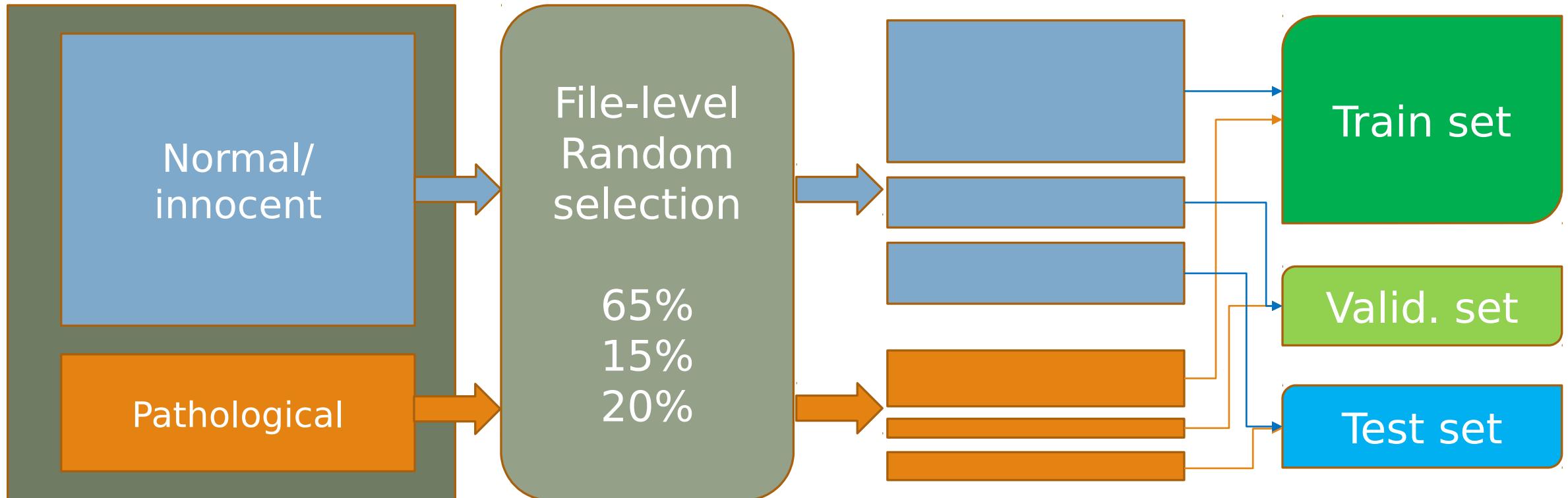
```
if modelName=="uocSeq0":#Model with single convolutional layer
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1), padding='valid', activation='relu', input_shape=input_shape))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(32,kernel_regularizer=regularizers.l1(0.0005), activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))
    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.Adam(lr=0.0003, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0),
                  metrics=['accuracy'])#learning rate default: 0.001
```

```
elif modelName=="uocSeq1":#Model with two convolutional layers
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1), padding='valid', activation='relu', input_shape=input_shape))
    model.add(Conv2D(32, (3, 3), strides=(1, 1), padding='valid', activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(16,kernel_regularizer=regularizers.l1(0.0002)))
    model.add(Dense(num_classes, activation='softmax'))
    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0),
                  metrics=['accuracy'])#learning rate default: 0.001

elif modelName=="uocSeq2":
    model = Sequential()#Model with 4 convolutional layers
    model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1), padding='valid', activation='relu', input_shape=input_shape))
    model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1), padding='valid', activation='relu', input_shape=input_shape))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.5))
    model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1), padding='valid', activation='relu', input_shape=input_shape))
    model.add(Conv2D(32, kernel_size=(3, 3), strides=(1, 1), padding='valid', activation='relu', input_shape=input_shape))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(32,kernel_regularizer=regularizers.l1(0.0001), activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(16,kernel_regularizer=regularizers.l1(0.0001), activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True, clipnorm=5)
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer=sgd)
```

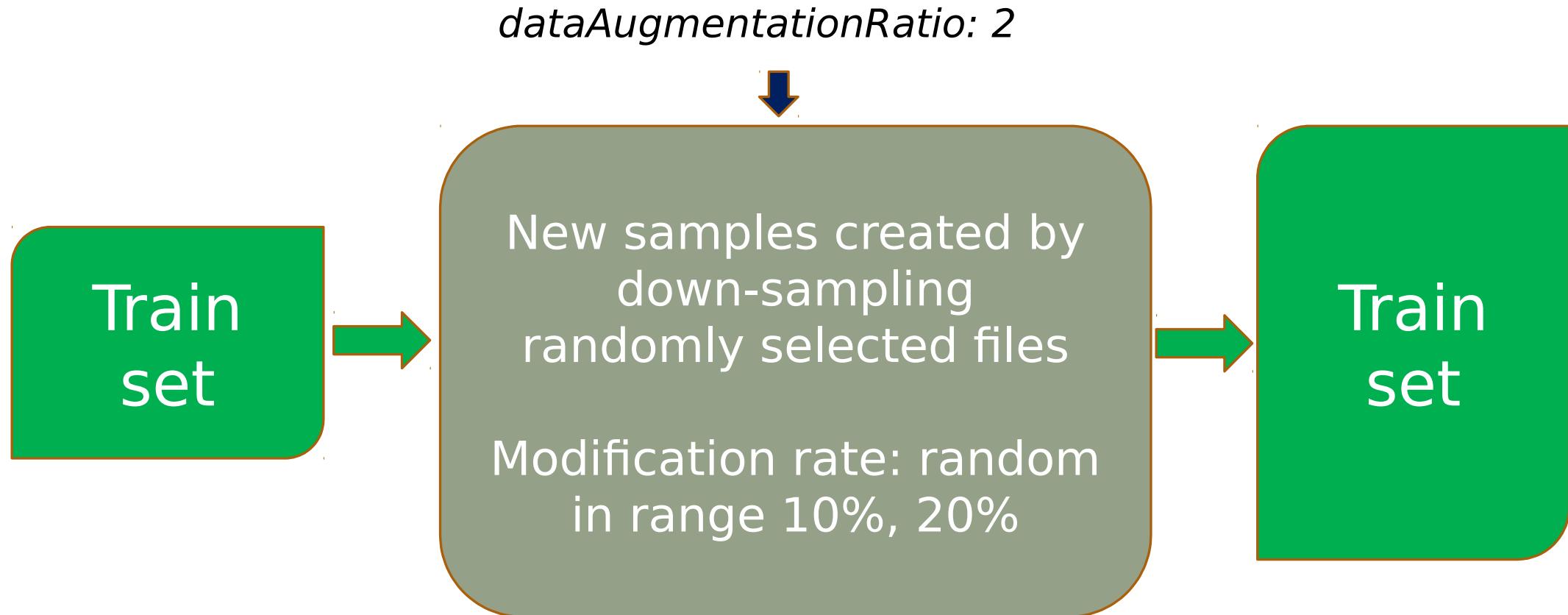
Test design: Splitting data into train/validation/test



Selection results in ~25 pathological samples in test set. Each random selection results in varying numbers of 'difficult to classify cases'. So, tests should be repeated and results averaged.

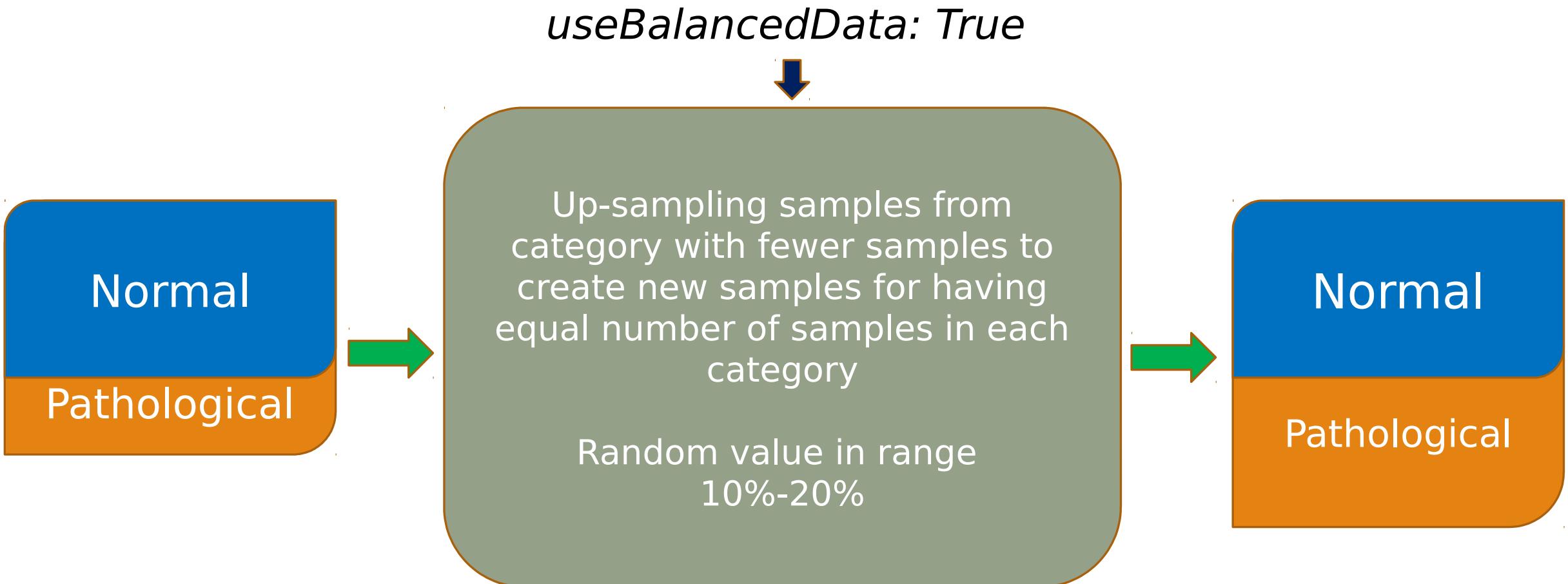
Test design:

Data augmentation **[Applied on train set]**

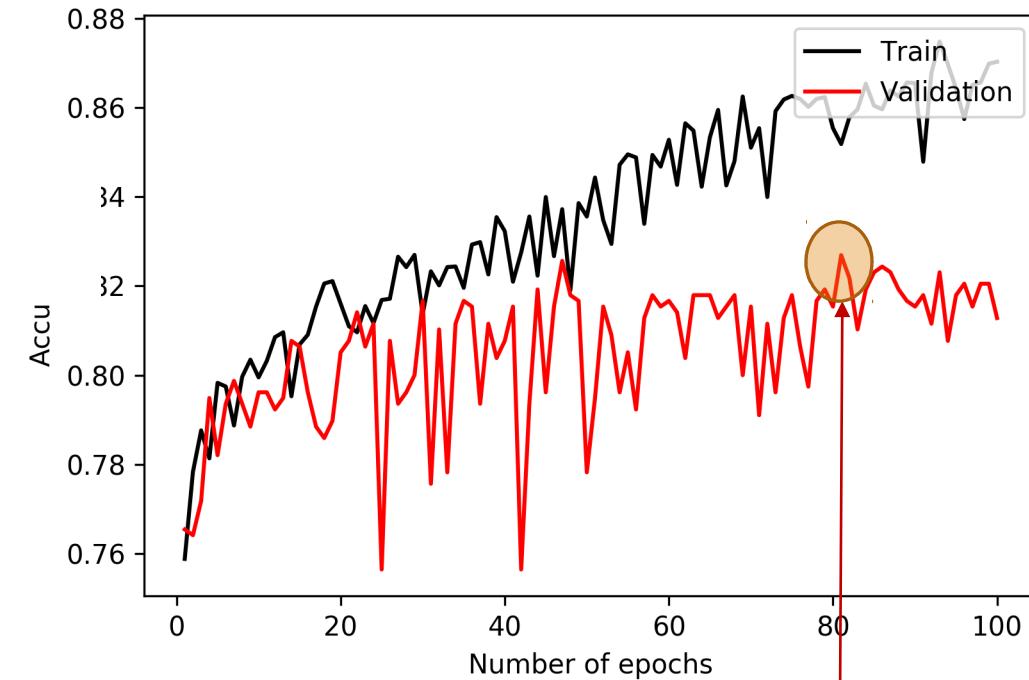
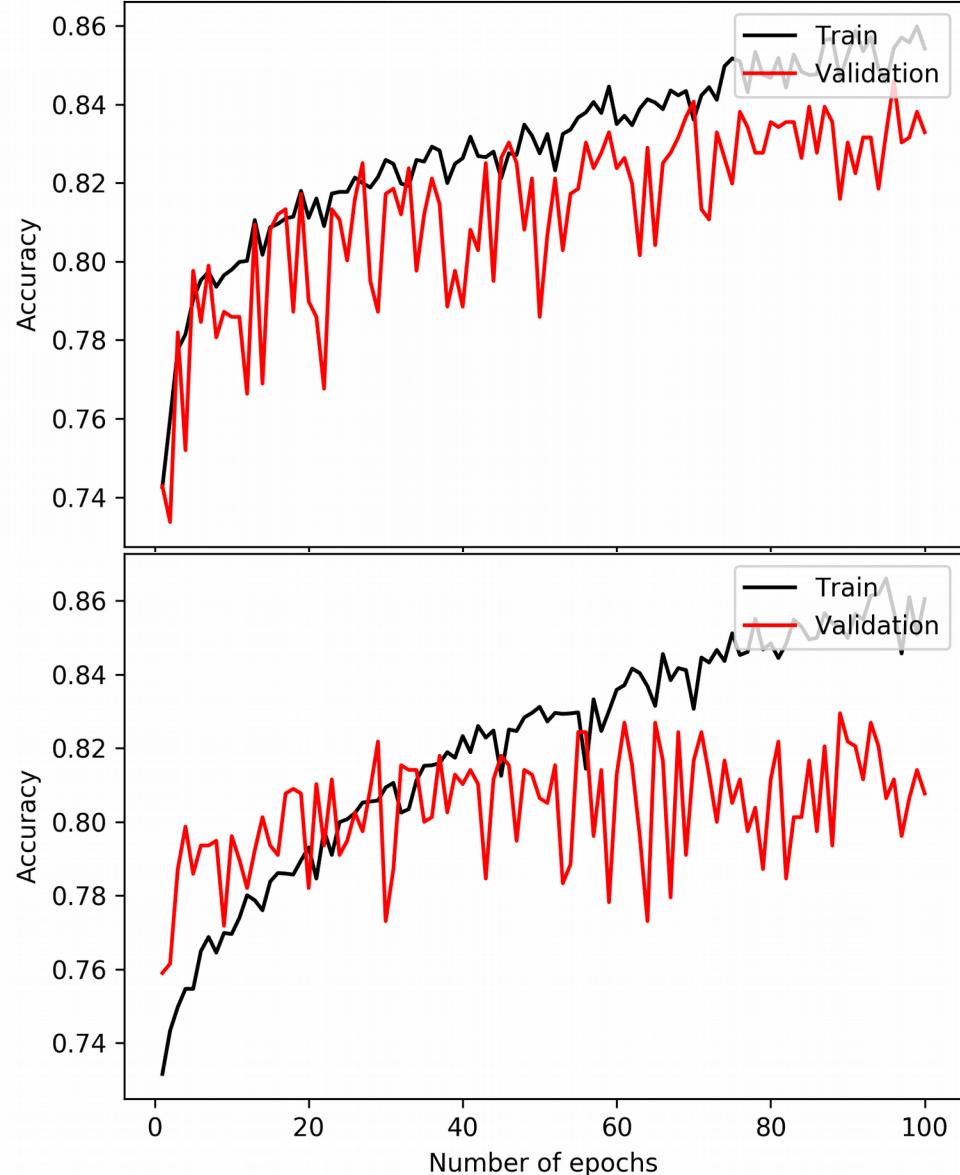


Test design:

Balancing the data via creating new samples for the small sized category
[Applied on train and validation sets]



Sample learning curves



We store the system with highest validation accuracy as the final model

User defined number of epochs=100 for these tests

Extensive testing with various settings is unavoidable for arriving at a good design.

Designing systematic tests and reporting the results may be our contribution to the community.

Database

Uoc: 2 categories with Mus. Mur and Path. Mur
Train-set 2 categories with (No Mur + Mus. Mur) and Path. Mur
3 categories
Physionet - 2016

=> 4 distinct cases

Segmentation

Ecg-based segmentation :
Pcg-based segmentation [needed for Physionet 2016]
Fixed-size segmentation/windowing
Sizes: 1 period, 2 periods, 3 periods, 1 sec., 2 sec, 3 sec

{ 6 options for Uoc
=> { 6 options for Physionet
6 options / dba

Features

Sub-band envelopes, Spectrogram,
Mel-spectr. / MFCC with/without delta
File level global features

Sizes: Freq. bands: 8, 16, 24
Time resolution: 24, 32, 48/period

~ 20-30 options

Neural Network Model

CNN, RNN, LSTM
Various numbers of layers , Regularisation, Drop-out
, Batch norm, Zero-pad, ...

~ 10 models

4800 - 6000 tests

After several attempts, I figured out this is impractical and decided to:

- Reduce number of tests
- Apply on a single dataset
- Pick limited number of best(?) systems
- Re-run experiments several times (%20->5)
- Test these few systems on several databases

Database

Uoc: 2 categories with Mus. Mur and Path. Mur
Train-set 2 categories with (No Mur + Mus. Mur) and Path. Mur
3 categories
Physionet - 2016

=> 4 distinct cases

Segmentation

Ecg-based segmentation :
Pcg-based segmentation ~~needed for Physionet 2016]~~
Fixed-size segmentation/windowing
Sizes: 1 period, 2 periods, 3 periods, 1 sec., 2 sec, 3 sec

{ 6 options for Uoc
=> { 6 options for Physionet
6 options / dba

Features

Sub-band envelopes, Spectrogram,
Mel-spectr./MFCC with/without delta
File level ~~global~~ features

Sizes: Freq. bands: 8, 16, 24
Time resolution: 24, 32, 48/period

~ 20-30 options

Neural Network Model

CNN, ~~RNN~~, ~~LSTM~~,
Various numbers of layers

, Regularisation, Drop-out
, Batch norm, Zero-pad, ...

~ 10 models

4800 - 6000 tests

Segmentation

ecg, periodSync, 170

Feature

MelSpec, 48x16, frame-level
segmentation, involve delta

features

MelSpec

Sub-band Env.

PNCC

...

Data

'Physionet', waves folder, features folder

features, use BalancedData, split ratios

TEST

NN model name = "voc1.0"

Data, mergeType = "majorityVote"
results folder, batch size,
number of training epochs

test.run() → Data.loadSplittedData()

.. loadModel()

.. train() → saves model file

.. test() → loads model file
and applies on
test data

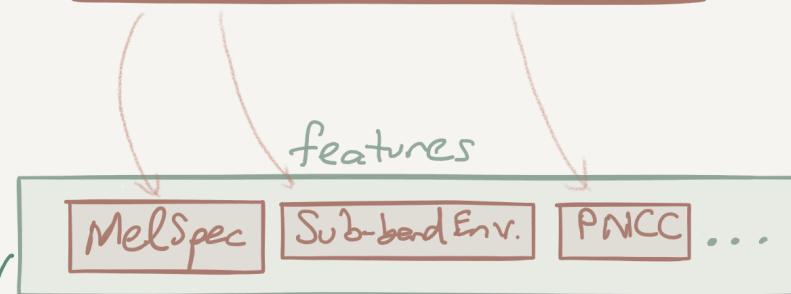
Result

confMatr., F1, acc, prec. ...
printing and plotting functions

**Designed an
Object-
oriented tool
to carry
combinational
testing in a
simple way**

Segmentation
ecg, periodSync, 170

Feature
MelSpec, 48x16, frame-level
segmentation, involve delta



```
#Define features to be used
features=[]
for featName in ['SubEnv', 'SubSig', 'MelSpec', 'MFCC']:
    for segType in segStrategies:
        for timeDim in [32, 64, 128, 256]:
            for freqDim in [8, 16, 24, 32]:
                features.append(Feature(featName, [timeDim, freqDim], "frame", segType, involveDelta=False))
                if featName=='MelSpec' or featName=='MFCC':#for MFCC and MelSpec, also add involve data coeffs.
                    features.append(Feature(featName, [timeDim, freqDim], "frame", segType, involveDelta=True))
```

Sample code lines

```
## Define segmentation strategies
ecg1perSegments=Segmentation("ecg",periodSync=True,sizeType="1period")
ecg2perSegments=Segmentation("ecg",periodSync=True,sizeType="2periods")
ecg3perSegments=Segmentation("ecg",periodSync=True,sizeType="3periods")
pcg1perSegments=Segmentation("pcg",periodSync=True,sizeType="1period")
```



```
#Segmentation strategies to be tested for the SubEnv feature:
segStrategies=[ecg1perSegments,ecg2perSegments,ecg3perSegments]
```

288 different feature settings
defined here

For a single experiment:
~10 lines of code

For hundreds of experiments:
~20-30 lines of code

TEST
NN model name = "uoc1.0"
→ Data , mergeType= "majorityVote"
results folder, batch size ,
number of training epochs

- Does not re-compute features if feature-files are available.
- The user can specify train/valid./test sets or use random splits

```
#Define data specifications
data=Data(dbName,wavFolder,featureFolder,features,useBalancedData,splitRatios,info)

for i in range(3):#this loop for setting models by name
    #Define test specifications
    CnnModelName="uocSeq"+str(i)
    test4SubEnv=Test(CnnModelName,data,mergeType,resultsFolder,batch_size=128,num_epochs=200)
    #Run the tests
    test4SubEnv.run()
```

Data
'Phystone+' , wavFolder , features Folder
features , useBalancedData , split ratios

Result
confMatr., f1, acc, prec. ...
printing and plotting functions

Frame-based decision to file-level decision

Probabilities for normal and pathological for frames of a file:

```
[ 0.21741118, 0.78258884] -> pathological  
[ 0.17369123, 0.82630873] -> pathological  
[ 0.59241056, 0.4075895 ] -> normal  
[ 0.23323412, 0.76676589] -> pathological  
[ 0.57892609, 0.42107388] -> normal  
[ 0.52060699, 0.47939304] -> normal  
[ 0.33756566, 0.6624344 ] -> pathological  
[ 0.18149786, 0.81850213] -> pathological  
[ 0.208248 , 0.79175204] -> pathological  
[ 0.22317596, 0.77682412] -> pathological
```

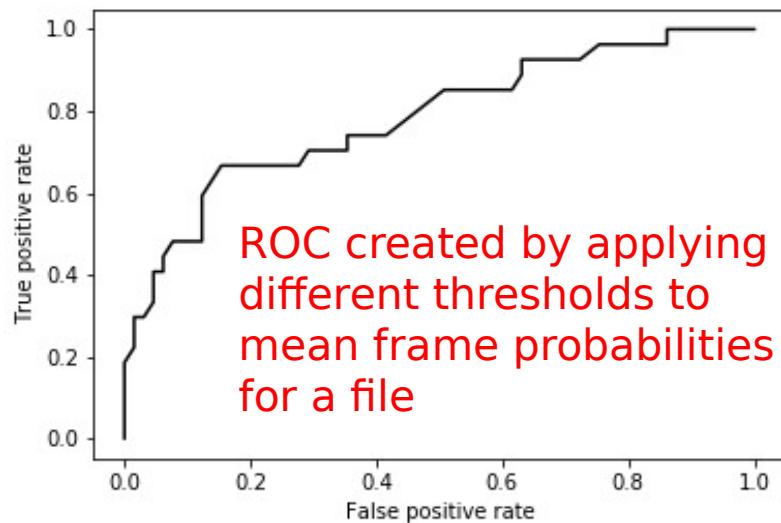
Majority voting

-> pathological

Other option: Computing mean of all frame based probabilities and applying a threshold (which may be lower than 0.5 for increasing sensitivity)

Comparing systems

How to compare hundreds of systems in a reliable way?



results

Search

results

Name

Date Modified

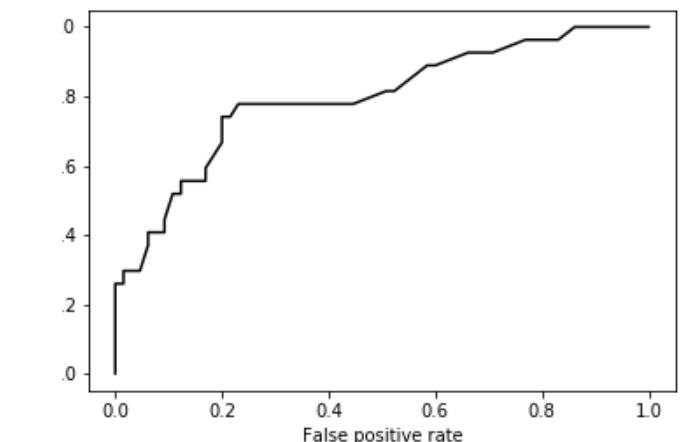
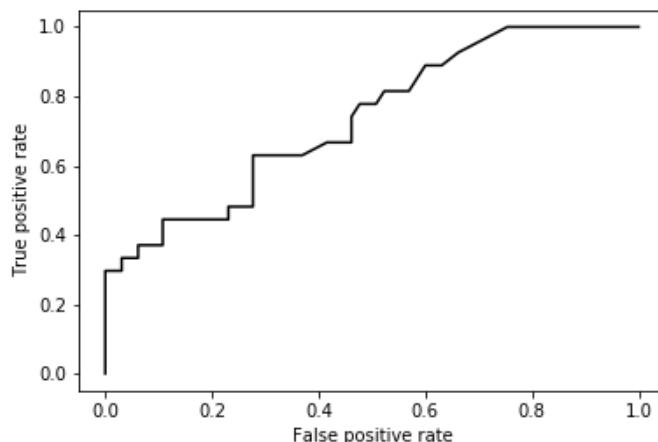
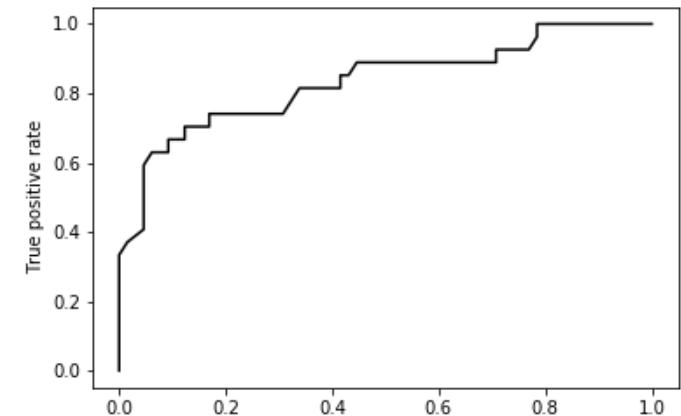
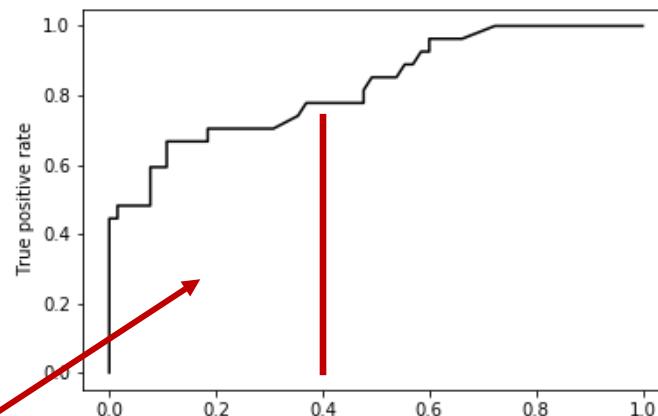
M_uocSeq1MFCC48by16_eS	53.0	12.0	
M_uocSeq1MFCC48by16_eS	9.0	18.0	
M_uocSeq1MFCC48by16_eS	Sensitivity =	0.6666666666666666	
M_uocSeq1MFCC48by16_eS	Specificity =	0.8153846153846154	
M_uocSeq1MFCC48by16_eS	Accuracy =	0.7717391304347826	
M_uocSeq1MFCC48by16_eS	F1 =	0.631578947368421	
M_uocSeq1MFCC48by16_eS	Matthews CC =	0.46824735250351485	
M_uocSeq1MFCC48by16_eS			
M_uocSeq1MFCC48by16_eS	50.0	15.0	
M_uocSeq1MFCC48by16_eS	10.0	17.0	
M_uocSeq1MFCC48by16_eS	Sensitivity =	0.6296296296296297	
M_uocSeq1MFCC48by16_eS	Specificity =	0.7692307692307693	
M_uocSeq1MFCC48by16_eS	Accuracy =	0.7282608695652174	
M_uocSeq1MFCC64by16_eS	F1 =	0.576271186440678	
M_uocSeq1MFCC64by16_eS	Matthews CC =	0.38133692943535785	
M_uocSeq1MFCC64by16_eSyn21_10t_maj_results_ROC.png			3 Jun 2017 03:54
M_uocSeq1MFCC64by16_eSyn21_10t_maj_results.txt			3 Jun 2017 03:54
M_uocSeq1MFCC64by16_eSyn21_10t.h5maj.pkl			3 Jun 2017 03:54
M_uocSeq1MFCC64by16_eSynf2_10t_maj_acc.png			3 Jun 2017 04:00
M_uocSeq1MFCC64by16_eSynf2_10t_maj_loss.png			3 Jun 2017 04:00
M_uocSeq1MFCC64by16_eSynf2_10t_maj_results_ROC.png			3 Jun 2017 04:00
M_uocSeq1MFCC64by16_eSynf2_10t_maj_results.txt			3 Jun 2017 04:00
M_uocSeq1MFCC64by16_eSynf2_10t.h5maj.pkl			3 Jun 2017 04:00
M_uocSeq1SubEnv32by32_eSyn11_10h_maj_acc.png			3 Jun 2017 01:30
M_uocSeq1SubEnv32by32_eSyn11_10h_maj_loss.png			3 Jun 2017 01:30

Selecting a system among many using the ROC curves

Main Criteria:

High sensitivity:

We don't want to miss any pathological cases, it is ok to mis classify some normal cases as pathological.



Area under the left part of ROC curve (obtained by thresholding mean probabilities) may be a good measure together with sensitivity and specificity of the majority voting based decision

Selected systems sorted with respect to area under the curve

uocSeq2SubEnv32by16_eSyn500len: Sequential model 2, Sub-band envelopes size 32*16
computed on Ecg-synchronous 500 msec length frames.

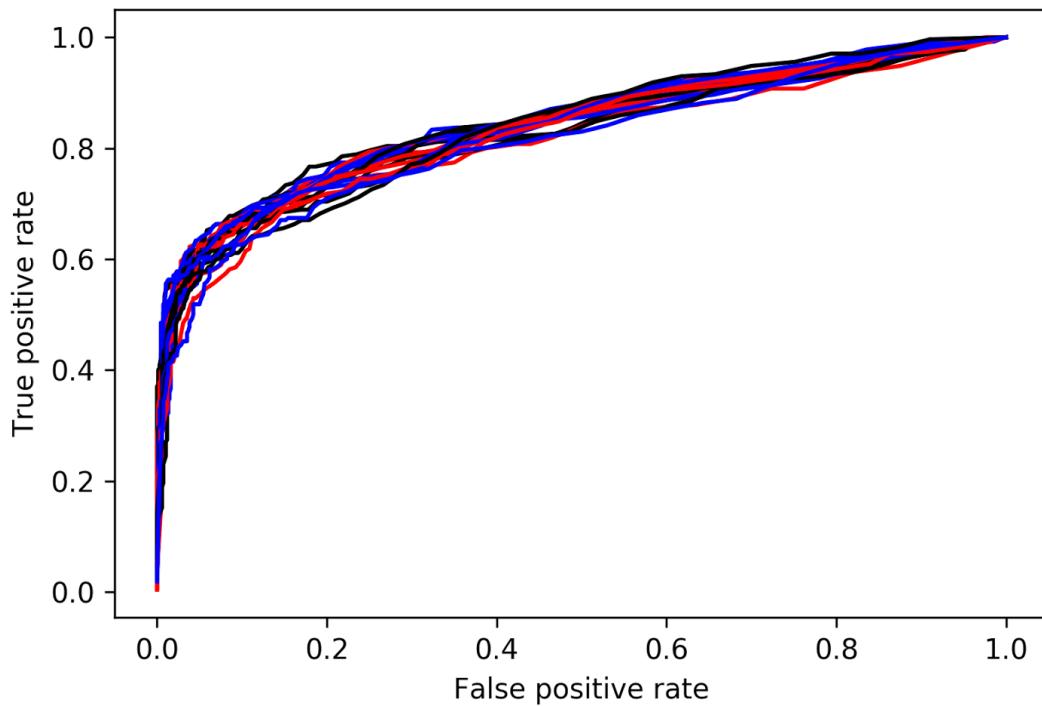
uocSeq2SubEnv128by16_eSyn2000len
uocSeq2SubEnv64by16_eSyn1000len
uocSeq2SubEnv32by16_eSyn1000len
uocSeq2SubEnv128by16_nASyn2000len
uocSeq0SubEnv128by8_eSyn2000len
uocSeq2MelSpec128by16_nASyn2000len
uocSeq1SubEnv32by8_eSyn1per
uocSeq1SubEnv64by8_eSyn1per
uocSeq2SubEnv32by16_eSyn1per
uocSeq0SubEnv64by8_eSyn1000len
....

Observations:

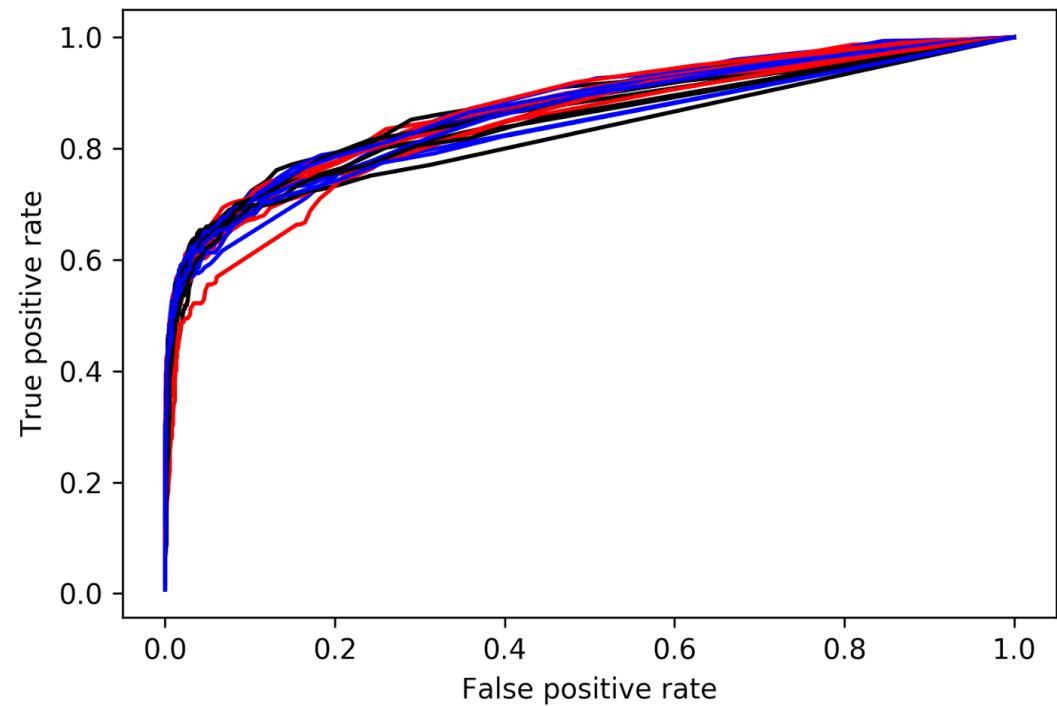
- Most of the selected systems use Sub-band envelopes
- More complex sequential models have higher performance
- While most of the selected systems use Ecg-synchronous frames, there are a few good systems selected to use asynchronous frames. This is worth dedicating time.

ROC curves of best 20 systems (selected with respect to area under ROC)

Grp1: innocent vs pathological murmur



Grp2: normal vs pathological



Preliminary test results for Physionet 2016

M_uocSeq1SubEnv64by16_nASyn2000len_1000hopt

File-level measures:

Confusion matrix:

Normal: 485 | 31

Pathological: 17 | 117

Sensitivity = 0.873, Specificity = 0.940,

Accuracy = 0.926, F1 = 0.830

Frame-level measures:

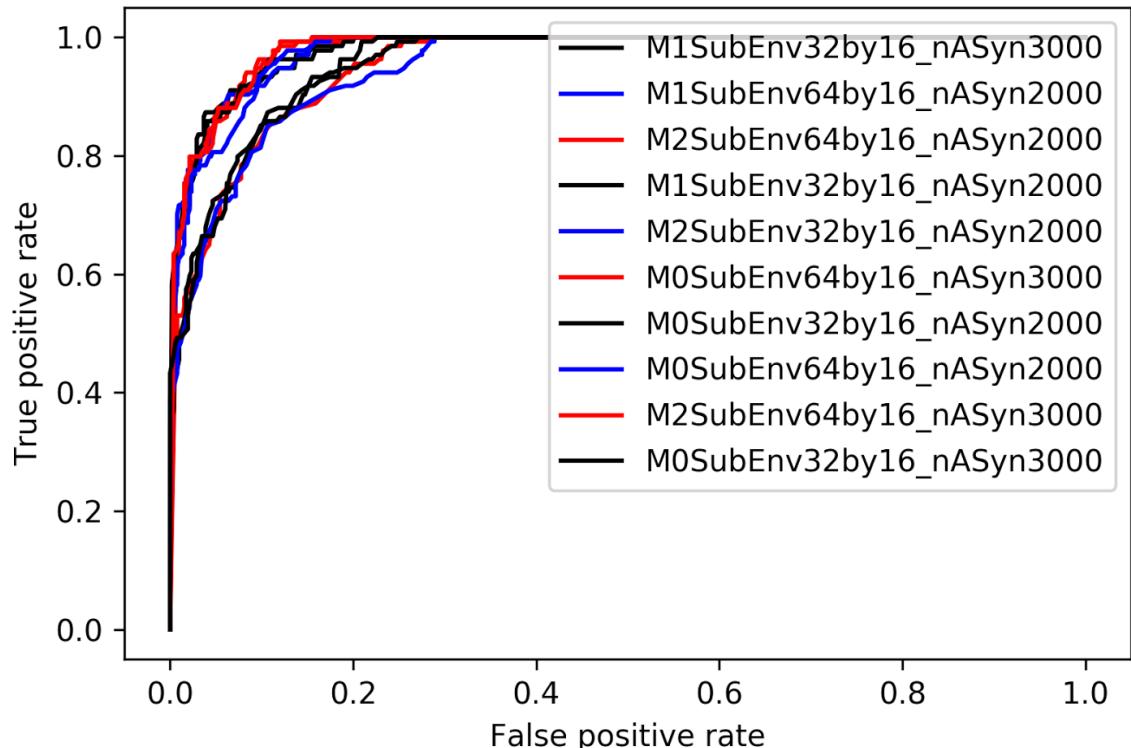
Confusion matrix:

Normal: 10045 | 663

Pathological: 424 | 2665

Sensitivity = 0.863, Specificity = 0.938

Best 10 systems ROCs for Physionet 2016



Result from literature [Potes et al, 2016]

Classifier	Sensitivity	Specificity	Overall Score
AdaBoost-abstain	0.70 (0.88)	0.88 (0.82)	0.79 (0.85)
CNN	0.79 (0.88)	0.86 (0.80)	0.82 (0.84)
Classifier ensemble	0.88 (0.96)	0.82 (0.80)	0.85 (0.89)