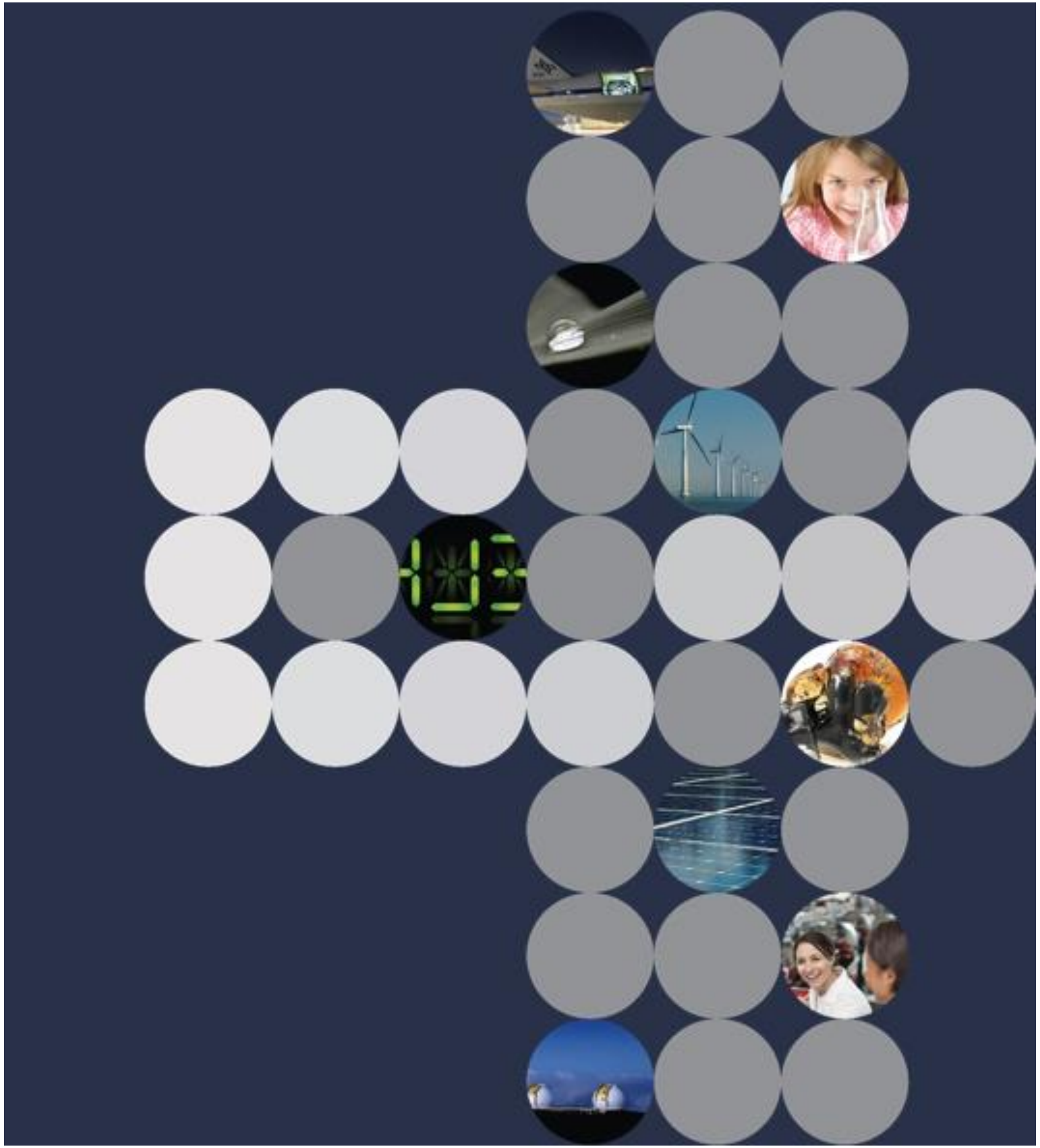
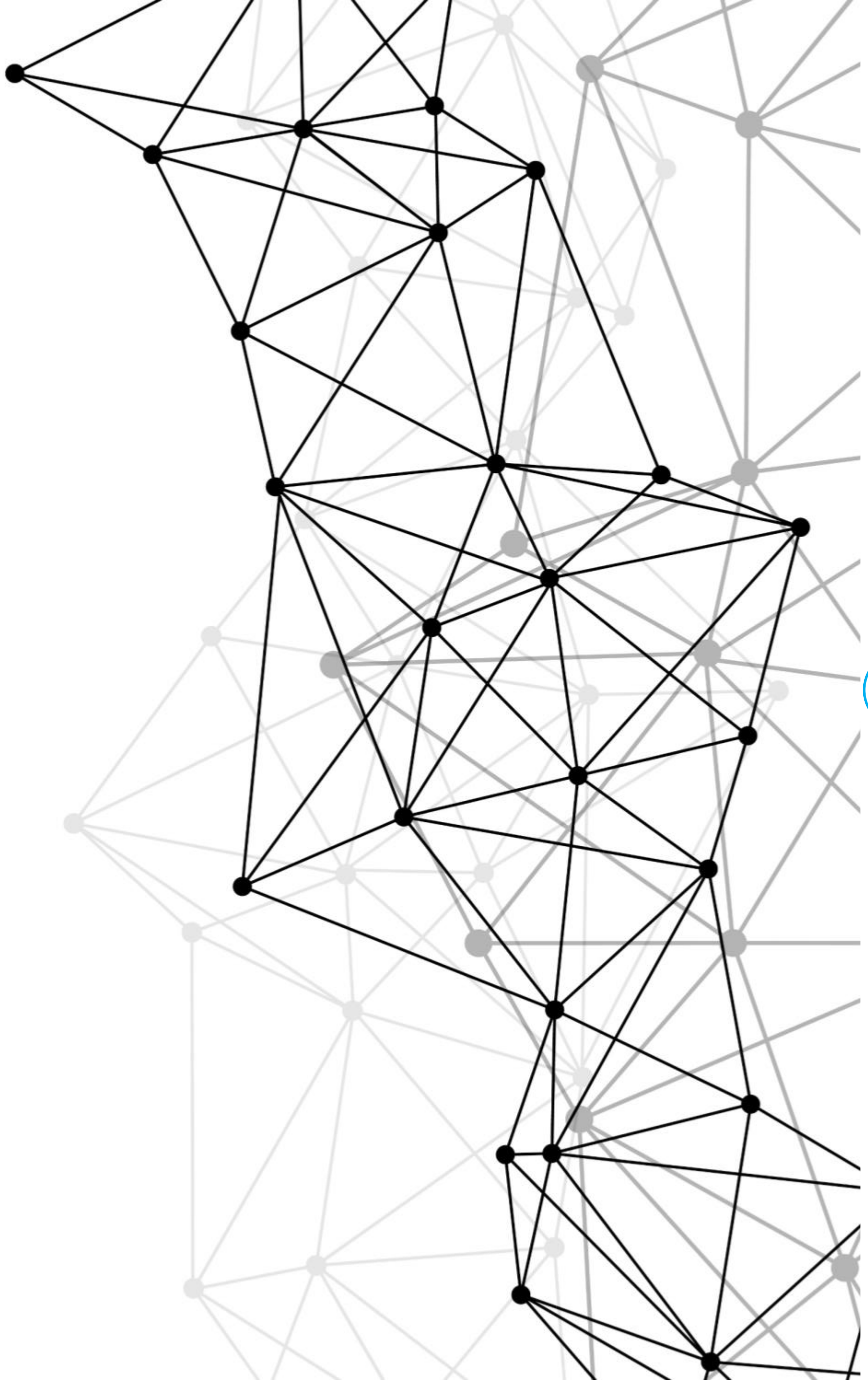
EE512 – Applied Biomedical Signal Processing

Neural network architecture

Clémentine AGUET
CSEM Signal Processing Group



- Labs



Lab – Instructions

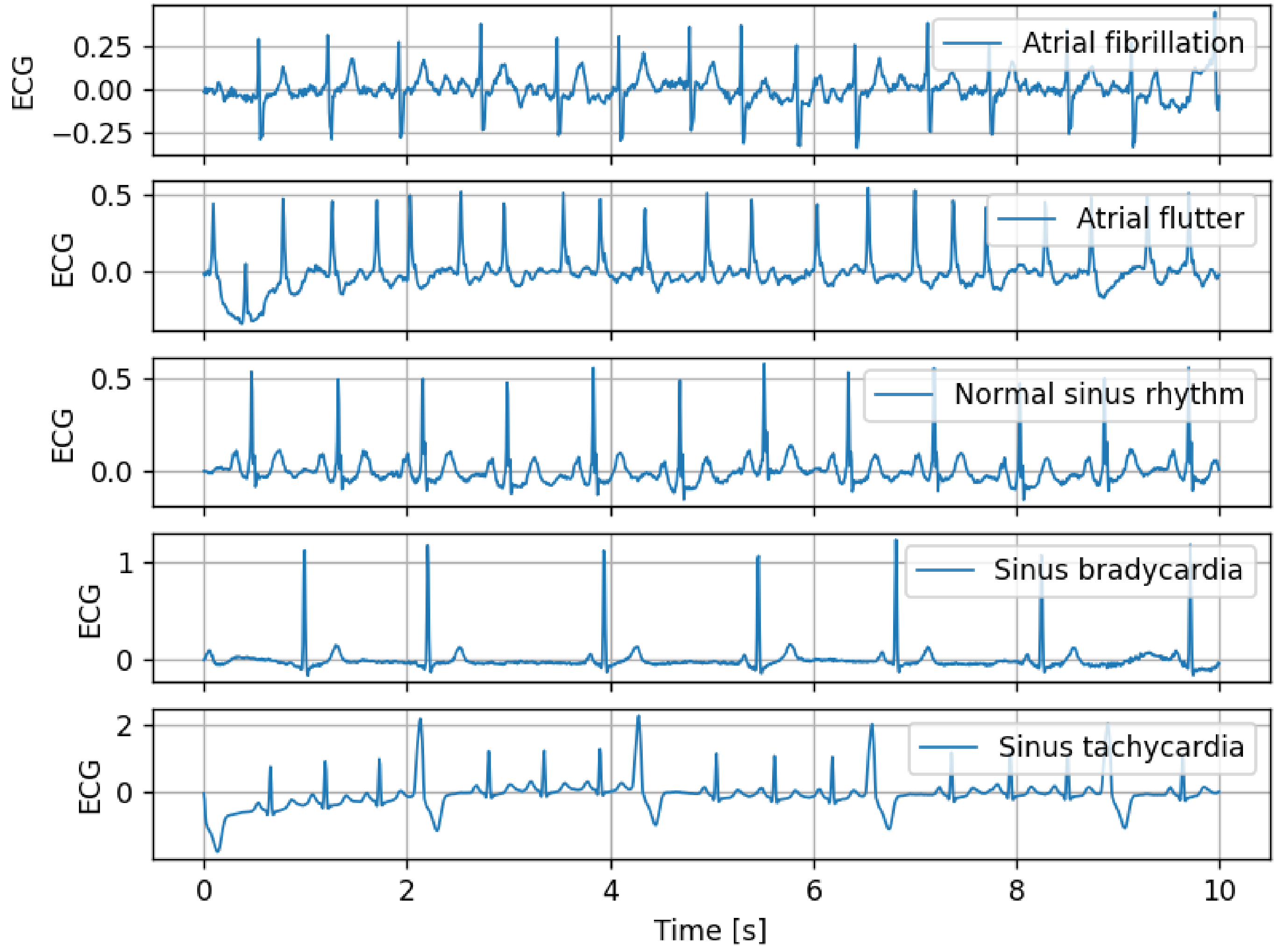
- Submit report as **single PDF file**
- Recommended to work in groups of **3 students**
- You can prepare one single report for the group (name1_name2_name3_lab_NN.pdf)
- But every member must upload the file on Moodle
- Python code is given and provided as **Jupyter notebooks**
- This practical session is not focused on coding but on questions testing your understanding and interpretation of the results.
- **2 exercises** in this lab session on real-life biomedical problems

Labs – Exercise 1

- **ECG rhythm classification**
  - GOAL: Train NN to classify different cardiac rhythms from single-lead ECG signals
  - Data: https://physionet.org/content/ecg-arrhythmia/1.0.0/
  - Cardiac rhythms:
    - Atrial fibrillation
    - Atrial flutter
    - Normal sinus rhythm
    - Sinus bradycardia
    - Sinus tachycardia
  - 1500 single-lead (lead II) ECG signals of each rhythm

Labs – Exercise 1

- **ECG rhythm classification**



Different in HR compared to normal sinus rhythm?

HR irregularity?

Labs – Exercise 1

- **ECG rhythm classification**
  - Split data into training, validation and testing subsets stratified by rhythms
  - 5 folds
    - 3 in training
    - 1 in validation
    - 1 in testing

Subset	Total	Atrial fibrillation	Atrial flutter	Normal sinus	Sinus bradycardia	Sinus tachycardia
Training	4500	900	900	900	900	900
Validation	1500	300	300	300	300	300
Test	1500	300	300	300	300	300

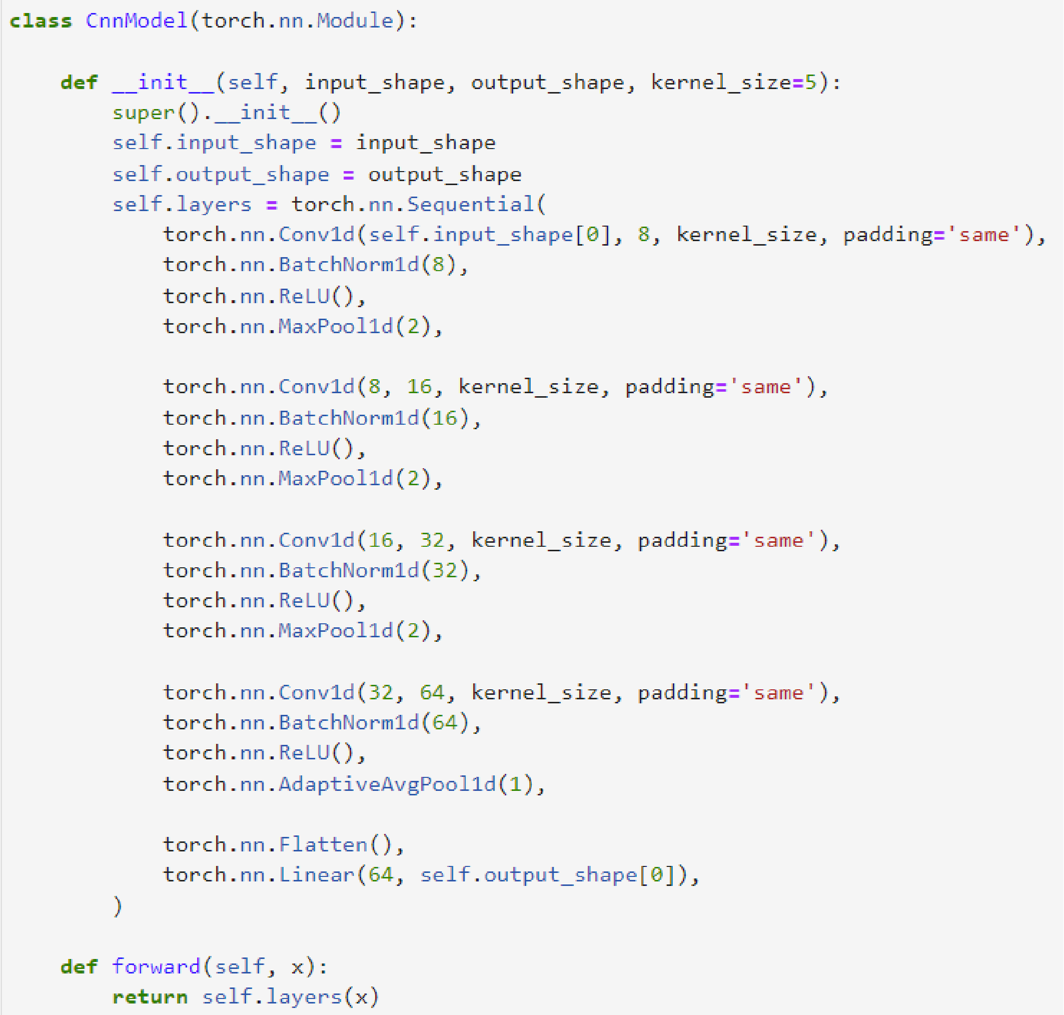
Labs – Exercise 1

- **ECG rhythm classification**
  - Scale ECG signals to have approximately unit variance
  - Encode rhythm labels with one-hot encoding

	Atrial fibrillation	Atrial flutter	Normal sinus	Sinus bradycardia	Sinus tachycardia
Atrial_fibrillation	1.0	0.0	0.0	0.0	0.0
Sinus_bradycardia	0.0	0.0	0.0	1.0	0.0
Sinus_bradycardia	0.0	0.0	0.0	1.0	0.0
Atrial_flutter	0.0	1.0	0.0	0.0	0.0
Atrial_fibrillation	1.0	0.0	0.0	0.0	0.0
Normal_sinus	0.0	0.0	1.0	0.0	0.0
Sinus_bradycardia	0.0	0.0	0.0	0.0	1.0
Atrial_flutter	0.0	1.0	0.0	0.0	0.0

Labs – Exercise 1

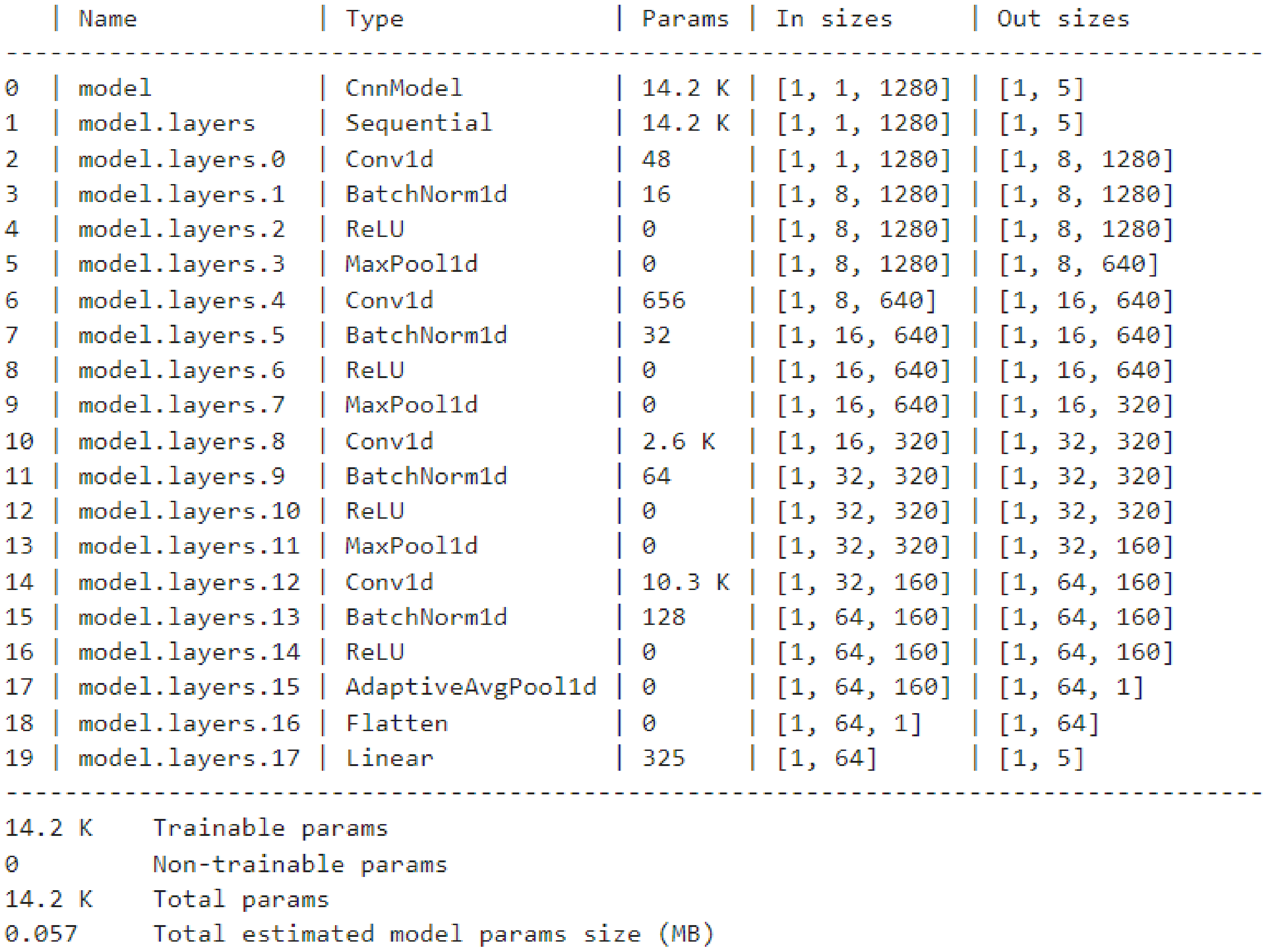
- **ECG rhythm classification**
  - CNN



```python
class CnnModel(torch.nn.Module):

    def __init__(self, input_shape, output_shape, kernel_size=5):
        super().__init__()
        self.input_shape = input_shape
        self.output_shape = output_shape
        self.layers = torch.nn.Sequential(
            torch.nn.Conv1d(self.input_shape[0], 8, kernel_size, padding='same'),
            torch.nn.BatchNorm1d(8),
            torch.nn.ReLU(),
            torch.nn.MaxPool1d(2),

            torch.nn.Conv1d(8, 16, kernel_size, padding='same'),
            torch.nn.BatchNorm1d(16),
            torch.nn.ReLU(),
            torch.nn.MaxPool1d(2),

            torch.nn.Conv1d(16, 32, kernel_size, padding='same'),
            torch.nn.BatchNorm1d(32),
            torch.nn.ReLU(),
            torch.nn.MaxPool1d(2),

            torch.nn.Conv1d(32, 64, kernel_size, padding='same'),
            torch.nn.BatchNorm1d(64),
            torch.nn.ReLU(),
            torch.nn.AdaptiveAvgPool1d(1),

            torch.nn.Flatten(),
            torch.nn.Linear(64, self.output_shape[0]),
        )

    def forward(self, x):
        return self.layers(x)
```

Labs – Exercise 1

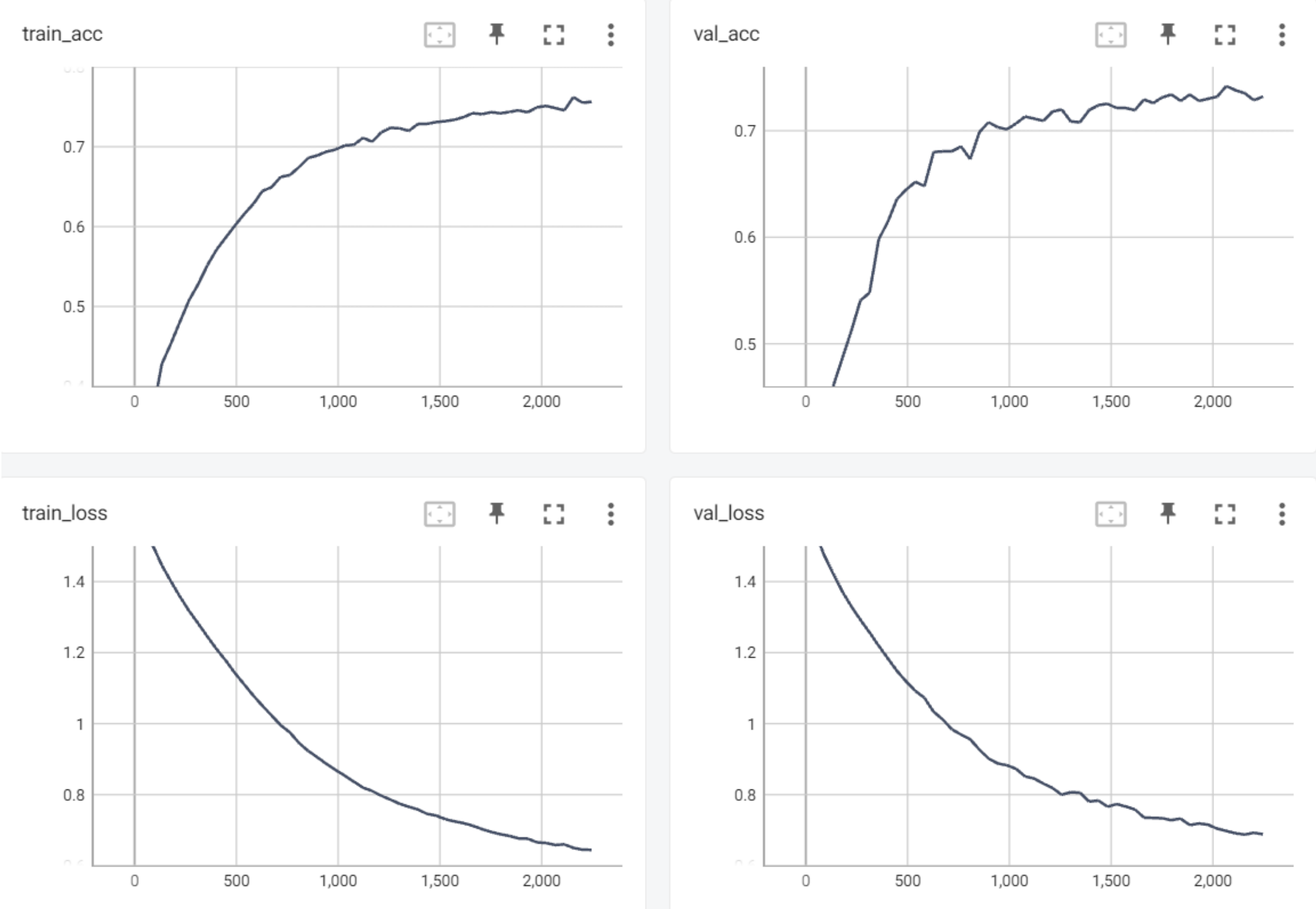
- **ECG rhythm classification**
  - CNN



```
   | Name             | Type                | Params | In sizes      | Out sizes
-----------------------------------------------------------------------------------------
0  | model            | CnnModel            | 14.2 K | [1, 1, 1280]  | [1, 5]
1  | model.layers     | Sequential          | 14.2 K | [1, 1, 1280]  | [1, 5]
2  | model.layers.0   | Conv1d              | 48     | [1, 1, 1280]  | [1, 8, 1280]
3  | model.layers.1   | BatchNorm1d         | 16     | [1, 8, 1280]  | [1, 8, 1280]
4  | model.layers.2   | ReLU                | 0      | [1, 8, 1280]  | [1, 8, 1280]
5  | model.layers.3   | MaxPool1d           | 0      | [1, 8, 1280]  | [1, 8, 640]
6  | model.layers.4   | Conv1d              | 656    | [1, 8, 640]   | [1, 16, 640]
7  | model.layers.5   | BatchNorm1d         | 32     | [1, 16, 640]  | [1, 16, 640]
8  | model.layers.6   | ReLU                | 0      | [1, 16, 640]  | [1, 16, 640]
9  | model.layers.7   | MaxPool1d           | 0      | [1, 16, 640]  | [1, 16, 320]
10 | model.layers.8   | Conv1d              | 2.6 K  | [1, 16, 320]  | [1, 32, 320]
11 | model.layers.9   | BatchNorm1d         | 64     | [1, 32, 320]  | [1, 32, 320]
12 | model.layers.10  | ReLU                | 0      | [1, 32, 320]  | [1, 32, 320]
13 | model.layers.11  | MaxPool1d           | 0      | [1, 32, 320]  | [1, 32, 160]
14 | model.layers.12  | Conv1d              | 10.3 K | [1, 32, 160]  | [1, 64, 160]
15 | model.layers.13  | BatchNorm1d         | 128    | [1, 64, 160]  | [1, 64, 160]
16 | model.layers.14  | ReLU                | 0      | [1, 64, 160]  | [1, 64, 160]
17 | model.layers.15  | AdaptiveAvgPool1d   | 0      | [1, 64, 160]  | [1, 64, 1]
18 | model.layers.16  | Flatten             | 0      | [1, 64, 1]    | [1, 64]
19 | model.layers.17  | Linear              | 325    | [1, 64]       | [1, 5]
-----------------------------------------------------------------------------------------
14.2 K    Trainable params
0         Non-trainable params
14.2 K    Total params
0.057     Total estimated model params size (MB)
```

Labs – Exercise 1

- **ECG rhythm classification**
    - Tensorboard

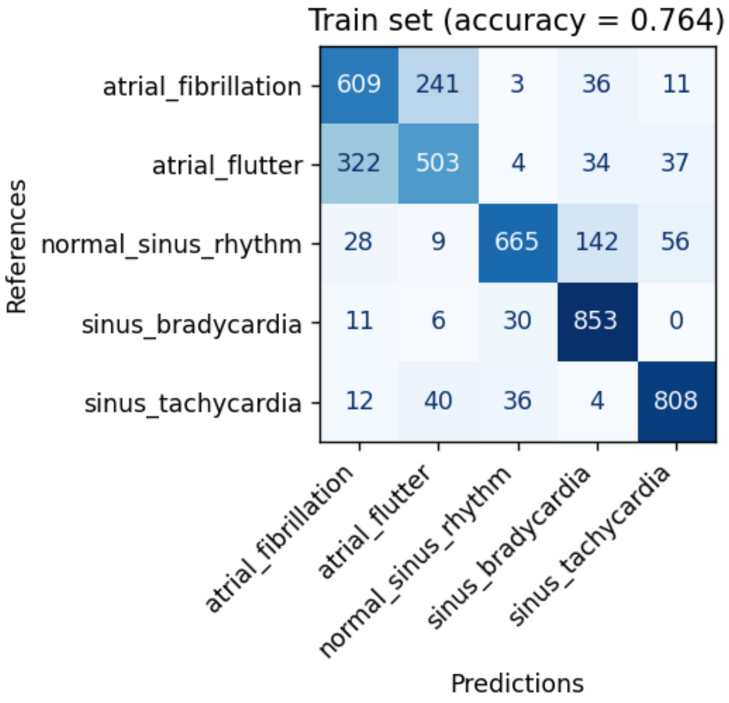


Overfitting?

Benefit for training longer?

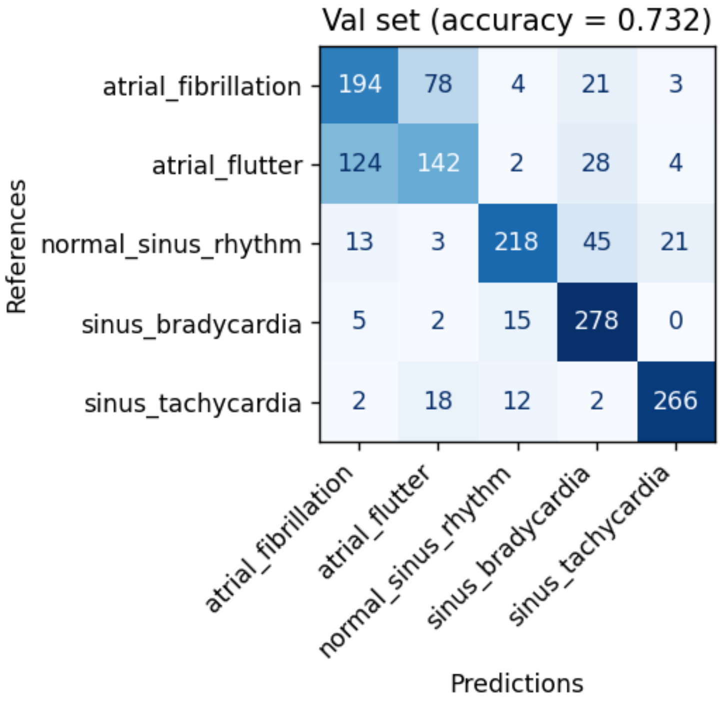
Labs – Exercise 1

- **ECG rhythm classification**
  - Evaluation with confusion matrices



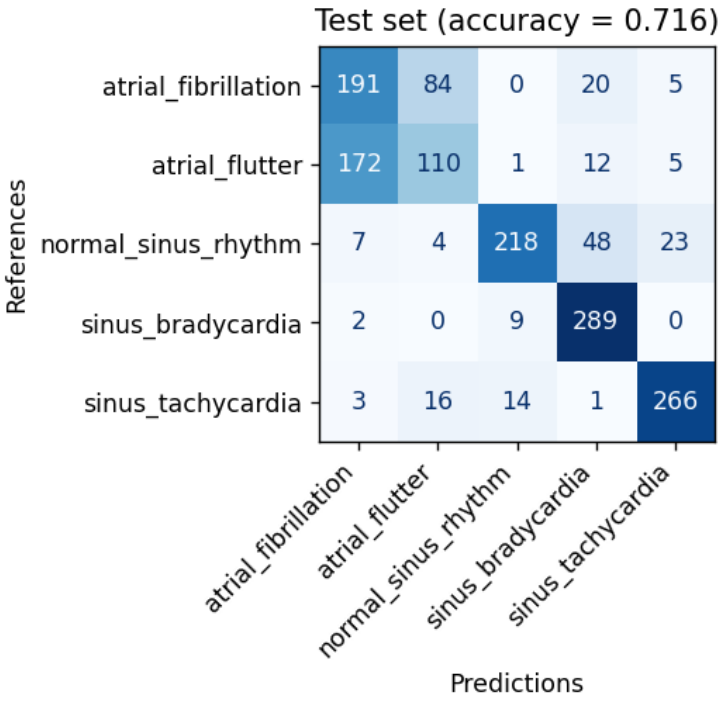
Train set (accuracy = 0.764)

References \ Predictions	atrial_fibrillation	atrial_flutter	normal_sinus_rhythm	sinus_bradycardia	sinus_tachycardia
atrial_fibrillation	609	241	3	36	11
atrial_flutter	322	503	4	34	37
normal_sinus_rhythm	28	9	665	142	56
sinus_bradycardia	11	6	30	853	0
sinus_tachycardia	12	40	36	4	808



Val set (accuracy = 0.732)

References \ Predictions	atrial_fibrillation	atrial_flutter	normal_sinus_rhythm	sinus_bradycardia	sinus_tachycardia
atrial_fibrillation	194	78	4	21	3
atrial_flutter	124	142	2	28	4
normal_sinus_rhythm	13	3	218	45	21
sinus_bradycardia	5	2	15	278	0
sinus_tachycardia	2	18	12	2	266



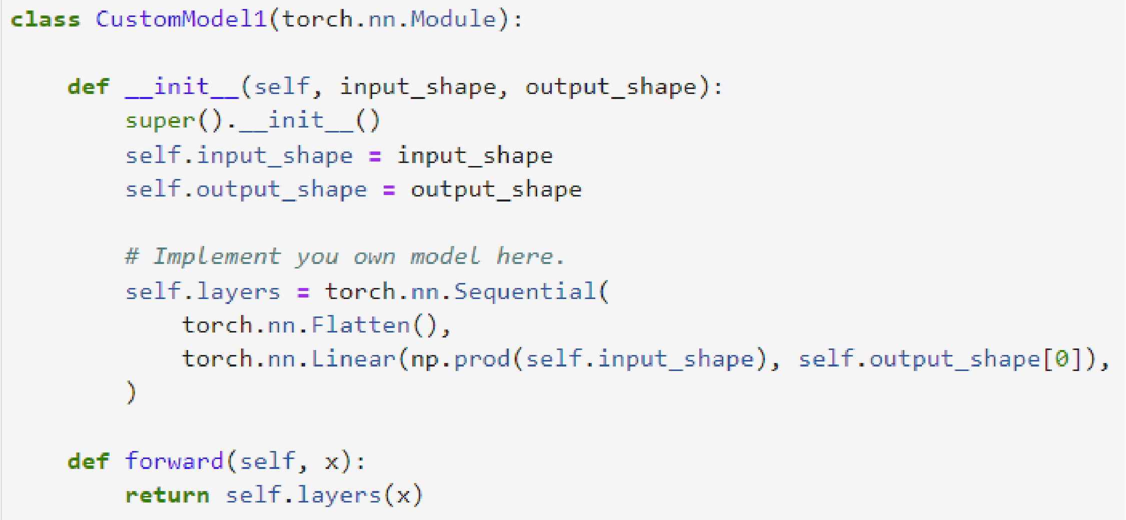
Test set (accuracy = 0.716)

References \ Predictions	atrial_fibrillation	atrial_flutter	normal_sinus_rhythm	sinus_bradycardia	sinus_tachycardia
atrial_fibrillation	191	84	0	20	5
atrial_flutter	172	110	1	12	5
normal_sinus_rhythm	7	4	218	48	23
sinus_bradycardia	2	0	9	289	0
sinus_tachycardia	3	16	14	1	266

Are some classes more difficult to classify?

Labs – Exercise 1

- **ECG rhythm classification**
  - 2 custom architectures
  - Add layers: convolutional, pooling, FC, etc.



```
class CustomModel1(torch.nn.Module):

    def __init__(self, input_shape, output_shape):
        super().__init__()
        self.input_shape = input_shape
        self.output_shape = output_shape

        # Implement you own model here.
        self.layers = torch.nn.Sequential(
            torch.nn.Flatten(),
            torch.nn.Linear(np.prod(self.input_shape), self.output_shape[0]),
        )

    def forward(self, x):
        return self.layers(x)
```

Labs – Exercise 2

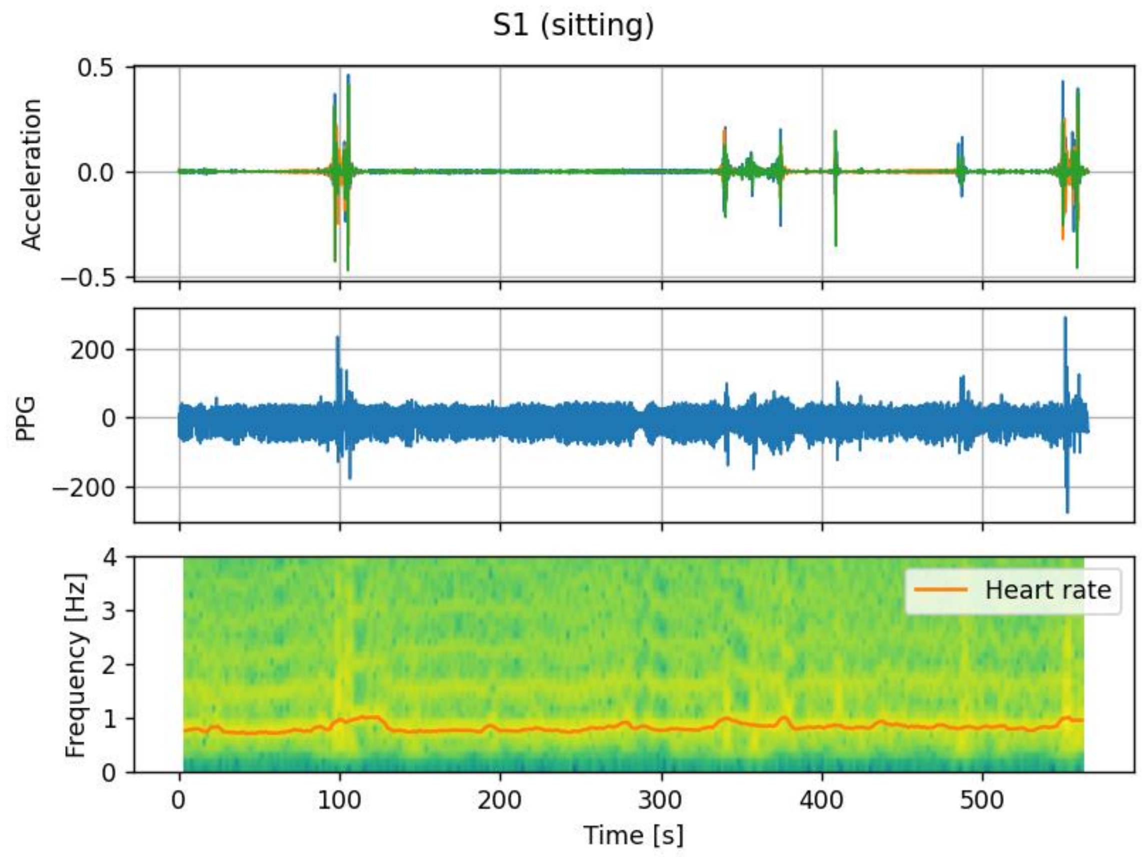
- **Heart rate classification**
  - GOAL: Estimate heart rate (HR) from PPG and acceleration signal
  - Data: https://archive.ics.uci.edu/ml/datasets/PPG-DaLiA
    - PPG signals
    - Acceleration signals
    - Reference HR computed from ECG signal
  - Data collected during various activities but focus on:
    - Sitting
    - Walking

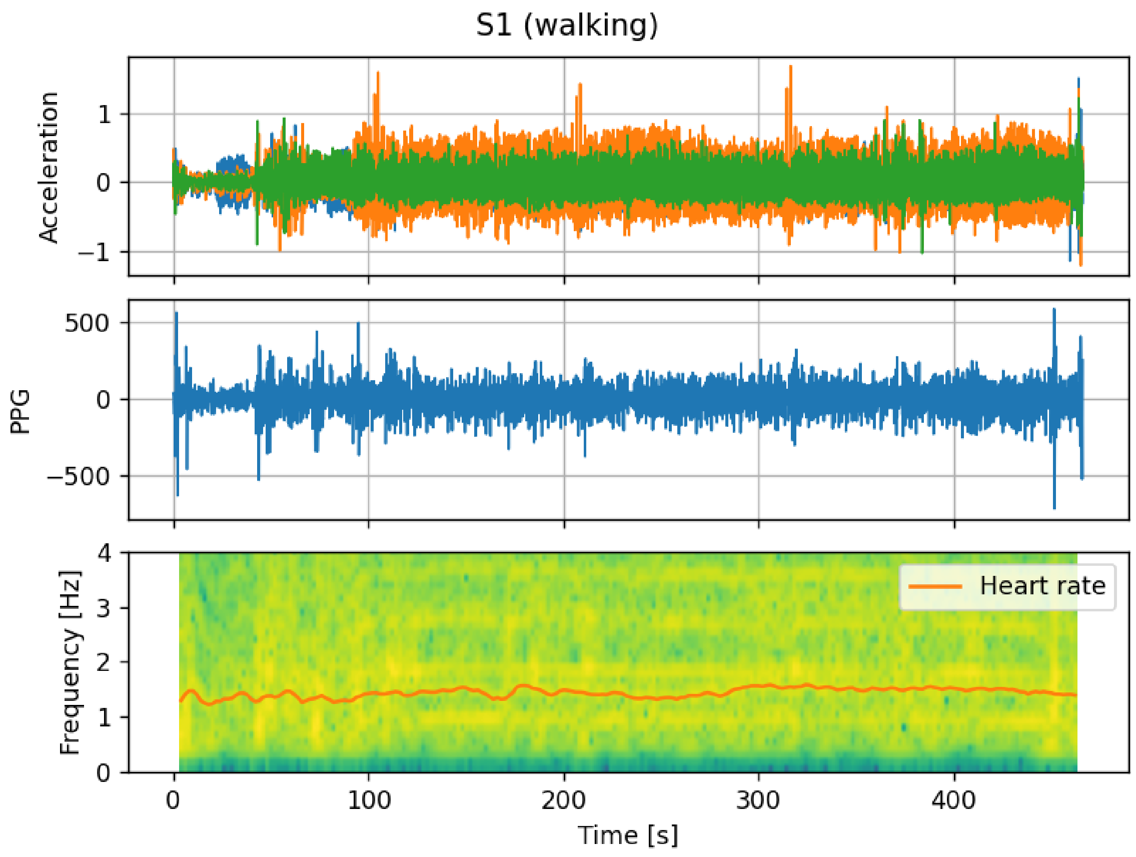
Labs – Exercise 2

- **Heart rate classification**
  - Preprocessing (already done)
    - Band-pass filtering between 0.4 and 4.0 Hz (24 – 240 bpm)
    - Resampling to 25 Hz
  - Split signal into windows with overlap
    - Window length: 8 s
    - Shift length: 2 s

Labs – Exercise 2

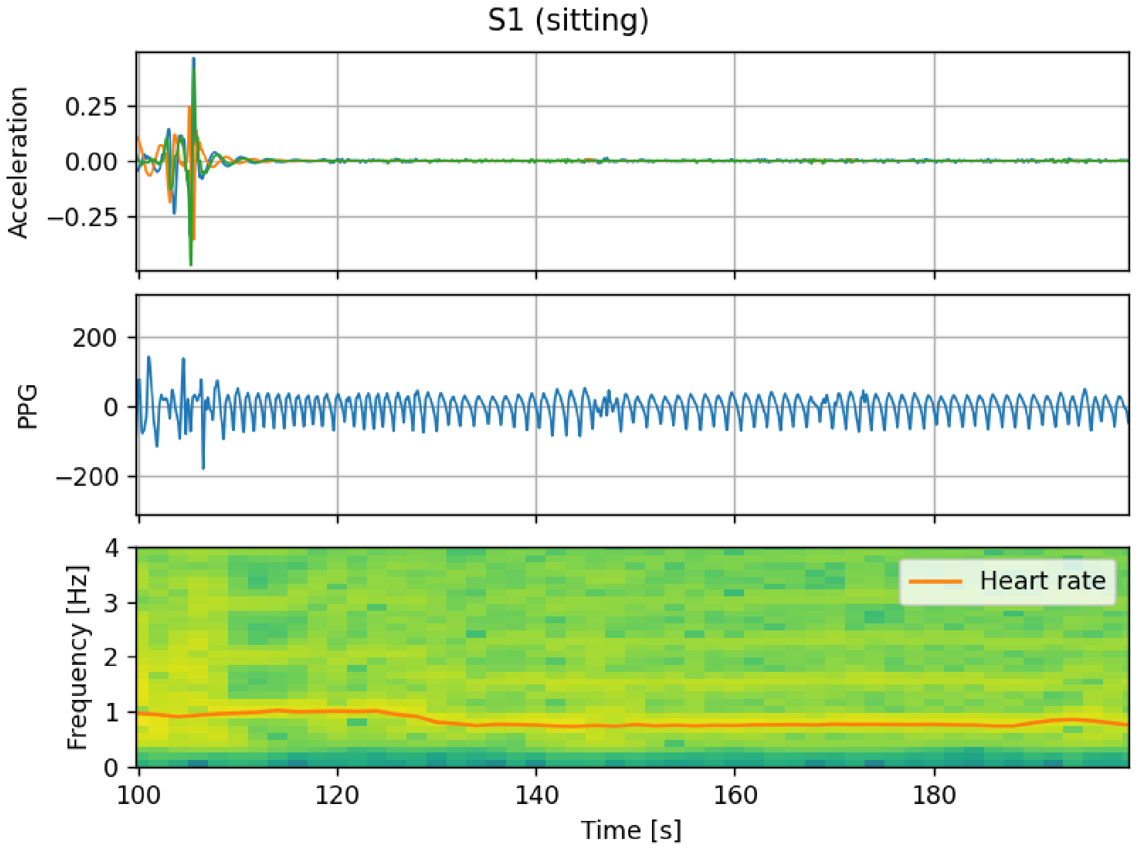
- **Heart rate classification**

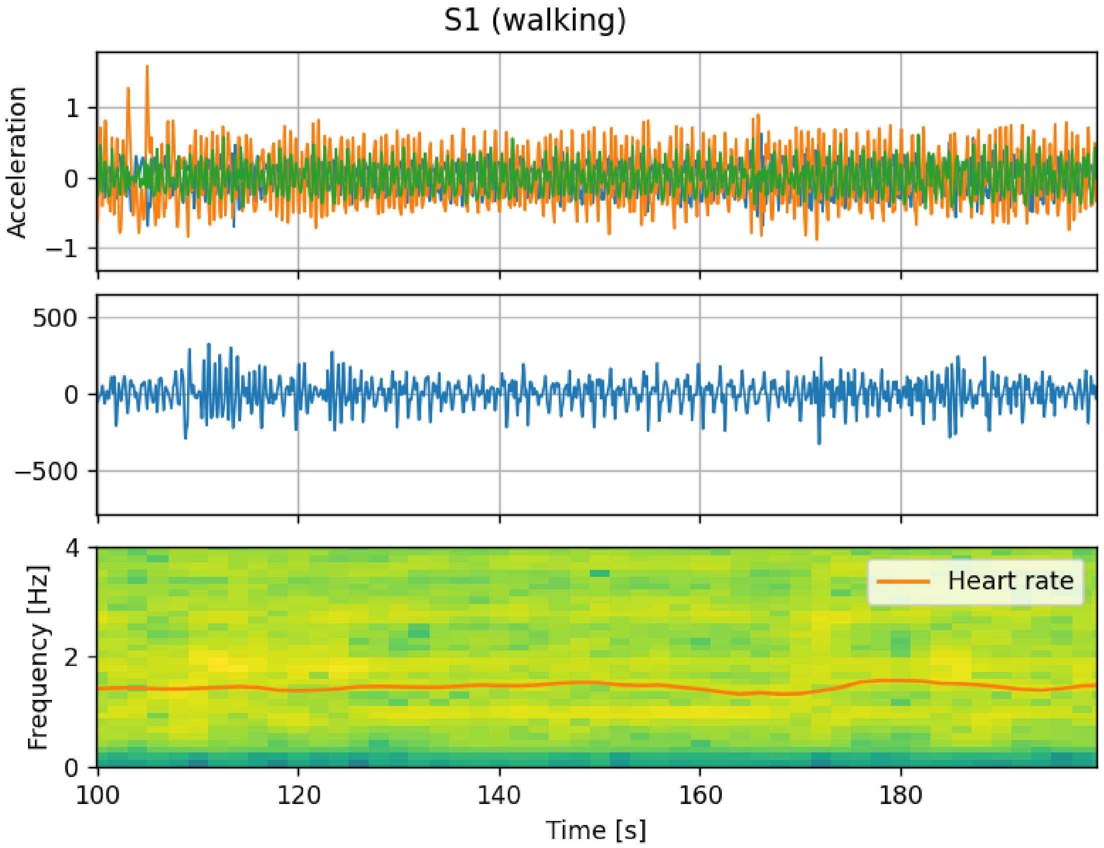




Labs – Exercise 2

- **Heart rate classification**





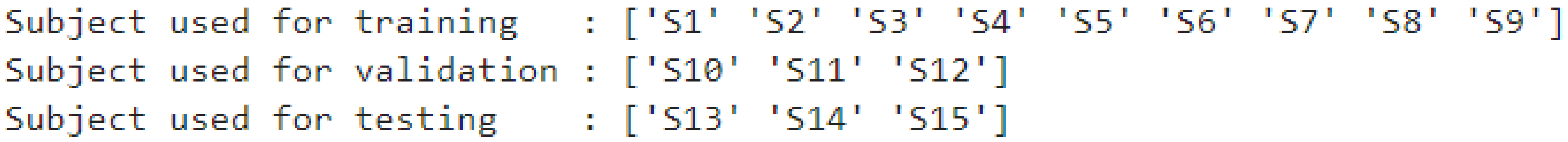
Labs – Exercise 2

- **Heart rate classification**
  - Extract signal windows with overlaps
    - Window length: 8 s
    - Shift length: 2 s

  - 7420 windows with 1 or 4 channels
  - Each window includes 200 samples (8 seconds at 25 Hz)
    - Input PPG (7420, 1, 200)
    - Input PPG + accelero (7420, 4, 200)

Labs – Exercise 2

- **Heart rate classification**
  - Split data into training, validation and testing subsets by subjects
    - 9 subjects in training set
    - 3 subjects in validation set
    - 3 subject in testing set

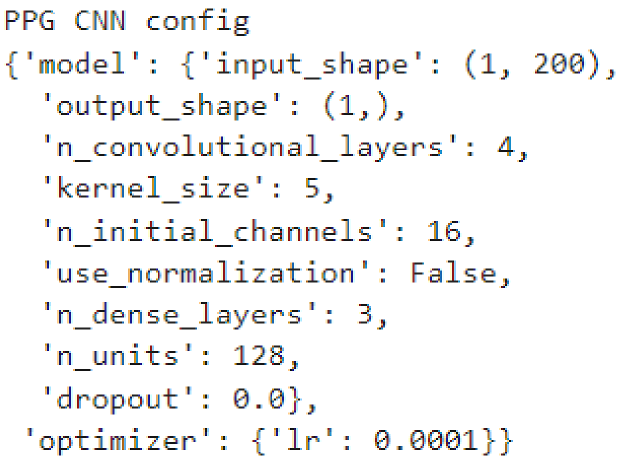


```
Subject used for training   : ['S1' 'S2' 'S3' 'S4' 'S5' 'S6' 'S7' 'S8' 'S9']
Subject used for validation : ['S10' 'S11' 'S12']
Subject used for testing    : ['S13' 'S14' 'S15']
```

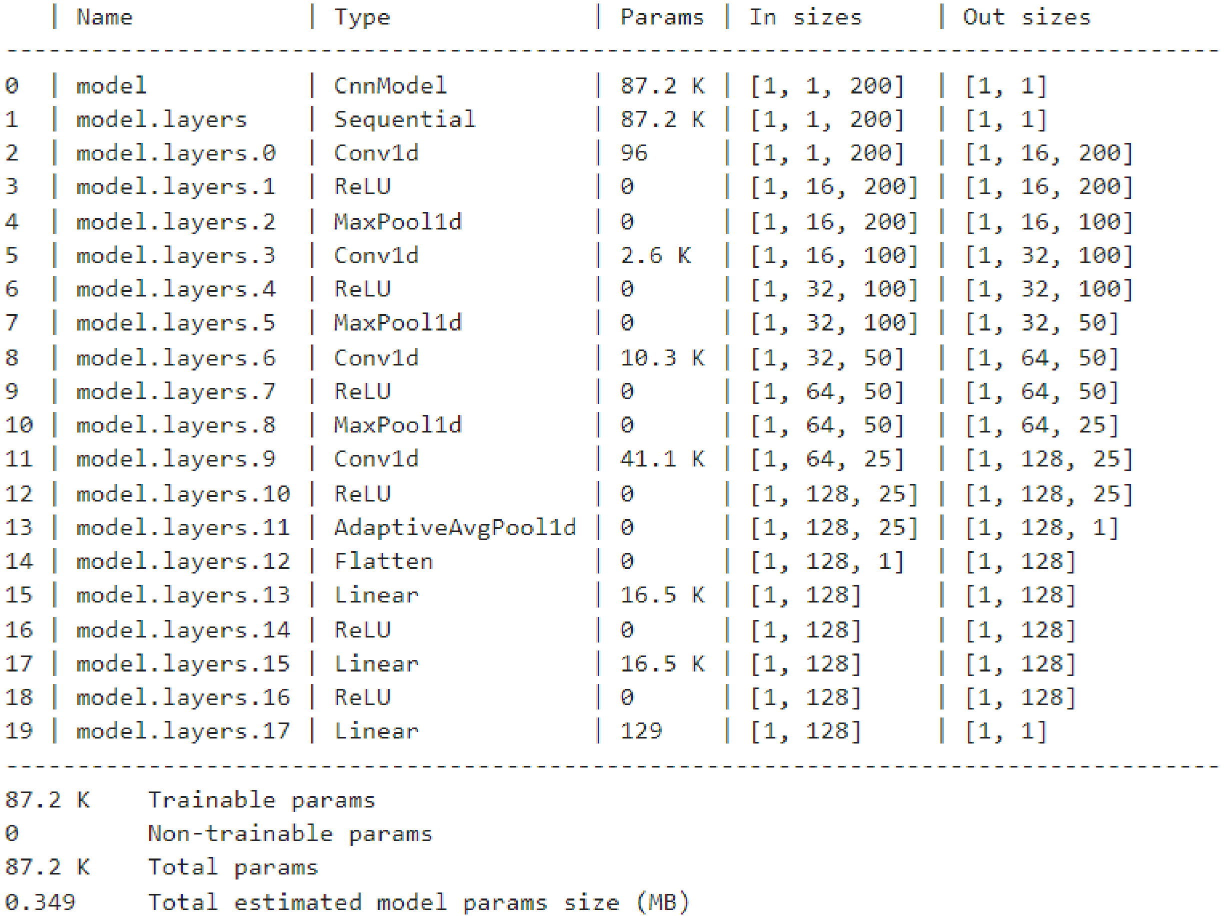
  - Scale windows to have approximately unit variance

Labs – Exercise 2

- **Heart rate classification**
  - CNN
  - Input: PPG
  - MSE loss function



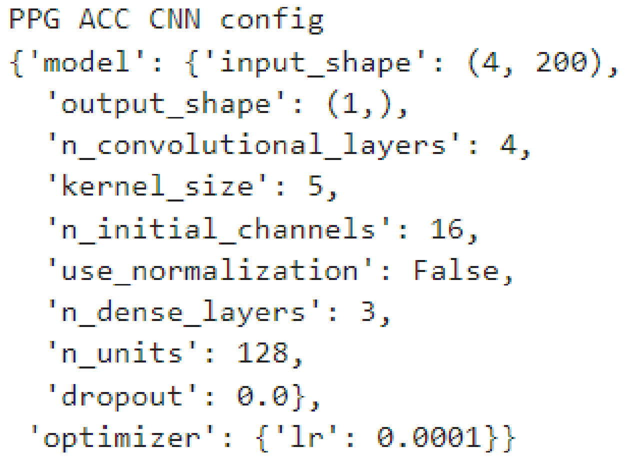
```
PPG CNN config
{'model': {'input_shape': (1, 200),
  'output_shape': (1,),
  'n_convolutional_layers': 4,
  'kernel_size': 5,
  'n_initial_channels': 16,
  'use_normalization': False,
  'n_dense_layers': 3,
  'n_units': 128,
  'dropout': 0.0},
 'optimizer': {'lr': 0.0001}}
```



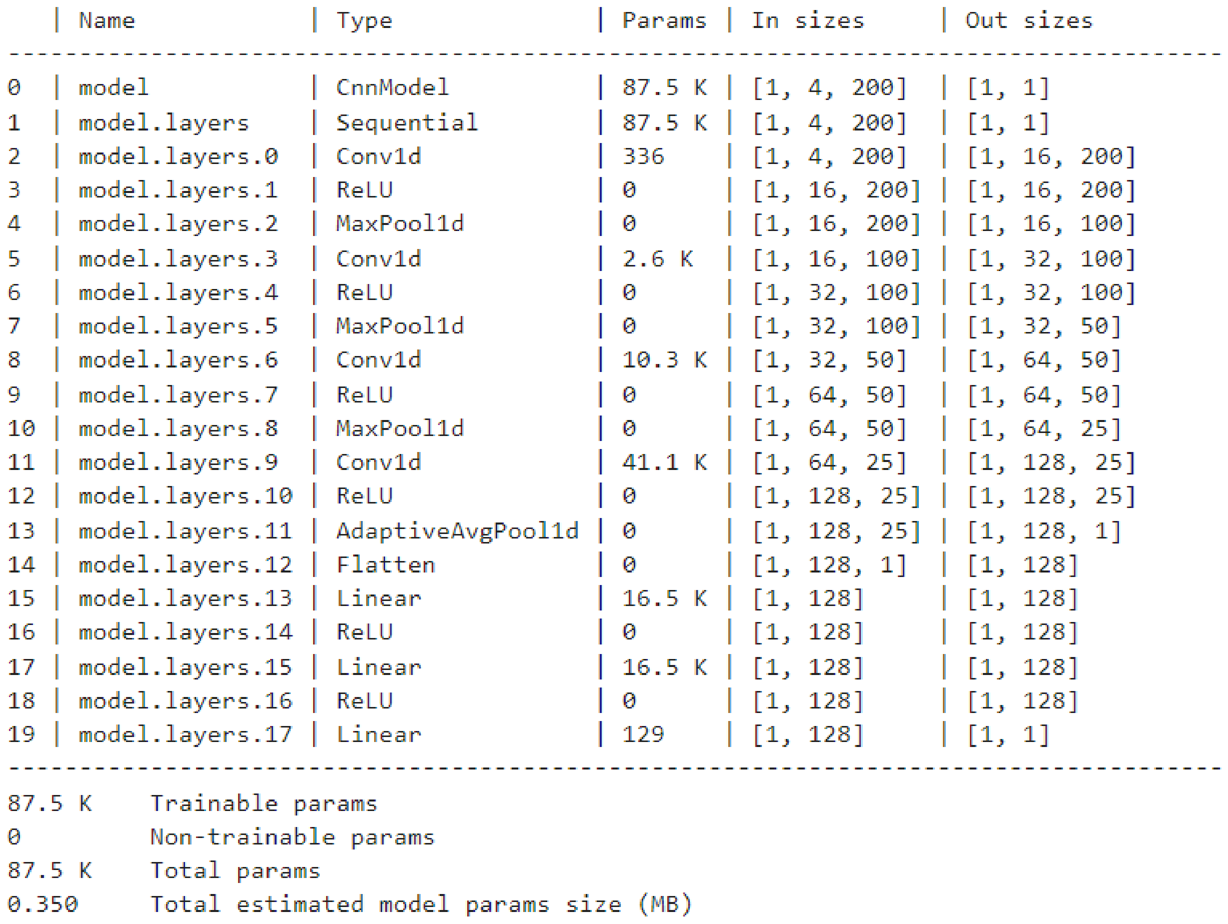
```
   | Name             | Type              | Params | In sizes       | Out sizes
----------------------------------------------------------------------------------
0  | model            | CnnModel          | 87.2 K | [1, 1, 200]    | [1, 1]
1  | model.layers     | Sequential        | 87.2 K | [1, 1, 200]    | [1, 1]
2  | model.layers.0   | Conv1d            | 96     | [1, 1, 200]    | [1, 16, 200]
3  | model.layers.1   | ReLU              | 0      | [1, 16, 200]   | [1, 16, 200]
4  | model.layers.2   | MaxPool1d         | 0      | [1, 16, 200]   | [1, 16, 100]
5  | model.layers.3   | Conv1d            | 2.6 K  | [1, 16, 100]   | [1, 32, 100]
6  | model.layers.4   | ReLU              | 0      | [1, 32, 100]   | [1, 32, 100]
7  | model.layers.5   | MaxPool1d         | 0      | [1, 32, 100]   | [1, 32, 50]
8  | model.layers.6   | Conv1d            | 10.3 K | [1, 32, 50]    | [1, 64, 50]
9  | model.layers.7   | ReLU              | 0      | [1, 64, 50]    | [1, 64, 50]
10 | model.layers.8   | MaxPool1d         | 0      | [1, 64, 50]    | [1, 64, 25]
11 | model.layers.9   | Conv1d            | 41.1 K | [1, 64, 25]    | [1, 128, 25]
12 | model.layers.10  | ReLU              | 0      | [1, 128, 25]   | [1, 128, 25]
13 | model.layers.11  | AdaptiveAvgPool1d | 0      | [1, 128, 25]   | [1, 128, 1]
14 | model.layers.12  | Flatten           | 0      | [1, 128, 1]    | [1, 128]
15 | model.layers.13  | Linear            | 16.5 K | [1, 128]       | [1, 128]
16 | model.layers.14  | ReLU              | 0      | [1, 128]       | [1, 128]
17 | model.layers.15  | Linear            | 16.5 K | [1, 128]       | [1, 128]
18 | model.layers.16  | ReLU              | 0      | [1, 128]       | [1, 128]
19 | model.layers.17  | Linear            | 129    | [1, 128]       | [1, 1]
----------------------------------------------------------------------------------
87.2 K    Trainable params
0         Non-trainable params
87.2 K    Total params
0.349     Total estimated model params size (MB)
```

Labs – Exercise 2

- **Heart rate classification**
  - CNN
  - Input: PPG + accelero
  - MSE loss function



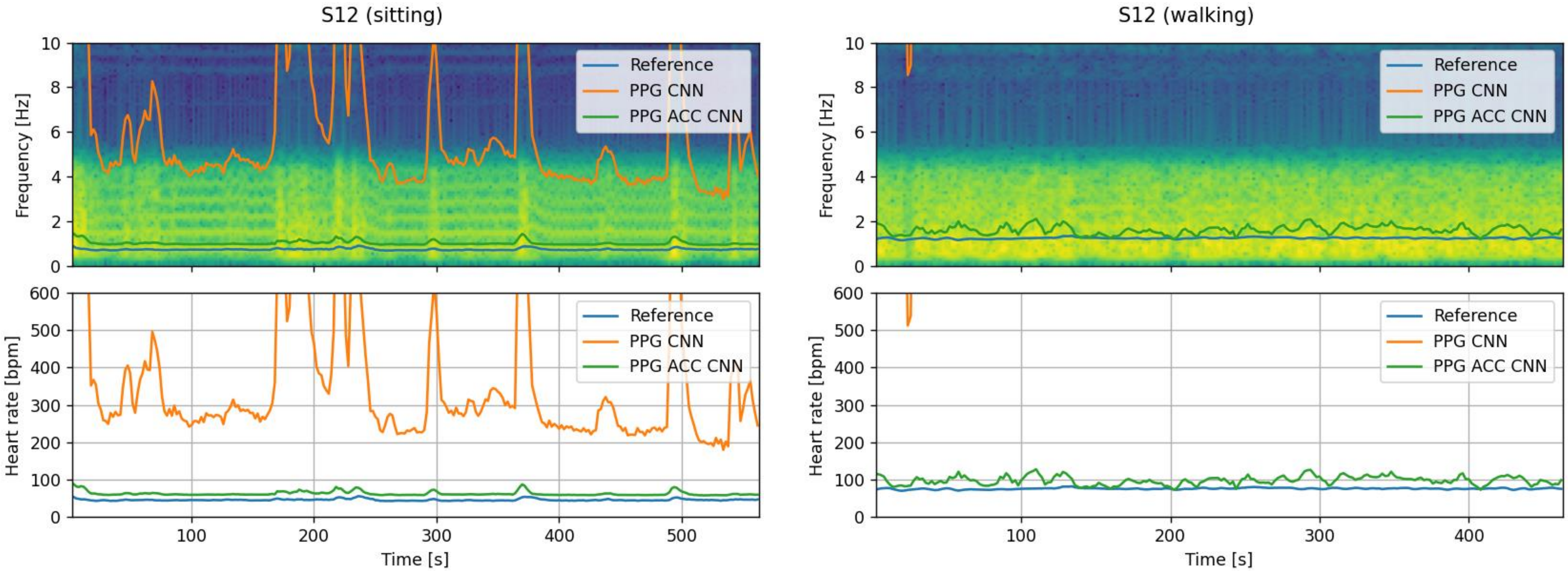
```
PPG ACC CNN config
{'model': {'input_shape': (4, 200),
  'output_shape': (1,),
  'n_convolutional_layers': 4,
  'kernel_size': 5,
  'n_initial_channels': 16,
  'use_normalization': False,
  'n_dense_layers': 3,
  'n_units': 128,
  'dropout': 0.0},
 'optimizer': {'lr': 0.0001}}
```



```
   | Name            | Type                | Params | In sizes       | Out sizes
-----------------------------------------------------------------------------------------
0  | model           | CnnModel            | 87.5 K | [1, 4, 200]    | [1, 1]
1  | model.layers    | Sequential          | 87.5 K | [1, 4, 200]    | [1, 1]
2  | model.layers.0  | Conv1d              | 336    | [1, 4, 200]    | [1, 16, 200]
3  | model.layers.1  | ReLU                | 0      | [1, 16, 200]   | [1, 16, 200]
4  | model.layers.2  | MaxPool1d           | 0      | [1, 16, 200]   | [1, 16, 100]
5  | model.layers.3  | Conv1d              | 2.6 K  | [1, 16, 100]   | [1, 32, 100]
6  | model.layers.4  | ReLU                | 0      | [1, 32, 100]   | [1, 32, 100]
7  | model.layers.5  | MaxPool1d           | 0      | [1, 32, 100]   | [1, 32, 50]
8  | model.layers.6  | Conv1d              | 10.3 K | [1, 32, 50]    | [1, 64, 50]
9  | model.layers.7  | ReLU                | 0      | [1, 64, 50]    | [1, 64, 50]
10 | model.layers.8  | MaxPool1d           | 0      | [1, 64, 50]    | [1, 64, 25]
11 | model.layers.9  | Conv1d              | 41.1 K | [1, 64, 25]    | [1, 128, 25]
12 | model.layers.10 | ReLU                | 0      | [1, 128, 25]   | [1, 128, 25]
13 | model.layers.11 | AdaptiveAvgPool1d   | 0      | [1, 128, 25]   | [1, 128, 1]
14 | model.layers.12 | Flatten             | 0      | [1, 128, 1]    | [1, 128]
15 | model.layers.13 | Linear              | 16.5 K | [1, 128]       | [1, 128]
16 | model.layers.14 | ReLU                | 0      | [1, 128]       | [1, 128]
17 | model.layers.15 | Linear              | 16.5 K | [1, 128]       | [1, 128]
18 | model.layers.16 | ReLU                | 0      | [1, 128]       | [1, 128]
19 | model.layers.17 | Linear              | 129    | [1, 128]       | [1, 1]
-----------------------------------------------------------------------------------------
87.5 K    Trainable params
0         Non-trainable params
87.5 K    Total params
0.350     Total estimated model params size (MB)
```

Labs – Exercise 2

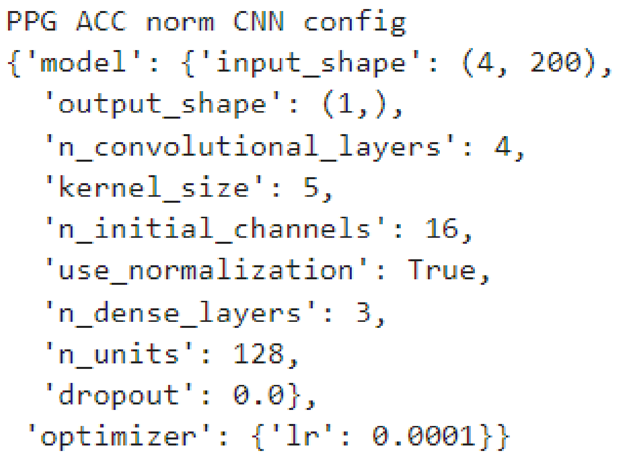
- **Heart rate classification**
  - HR prediction for validation set (with or without accelero)



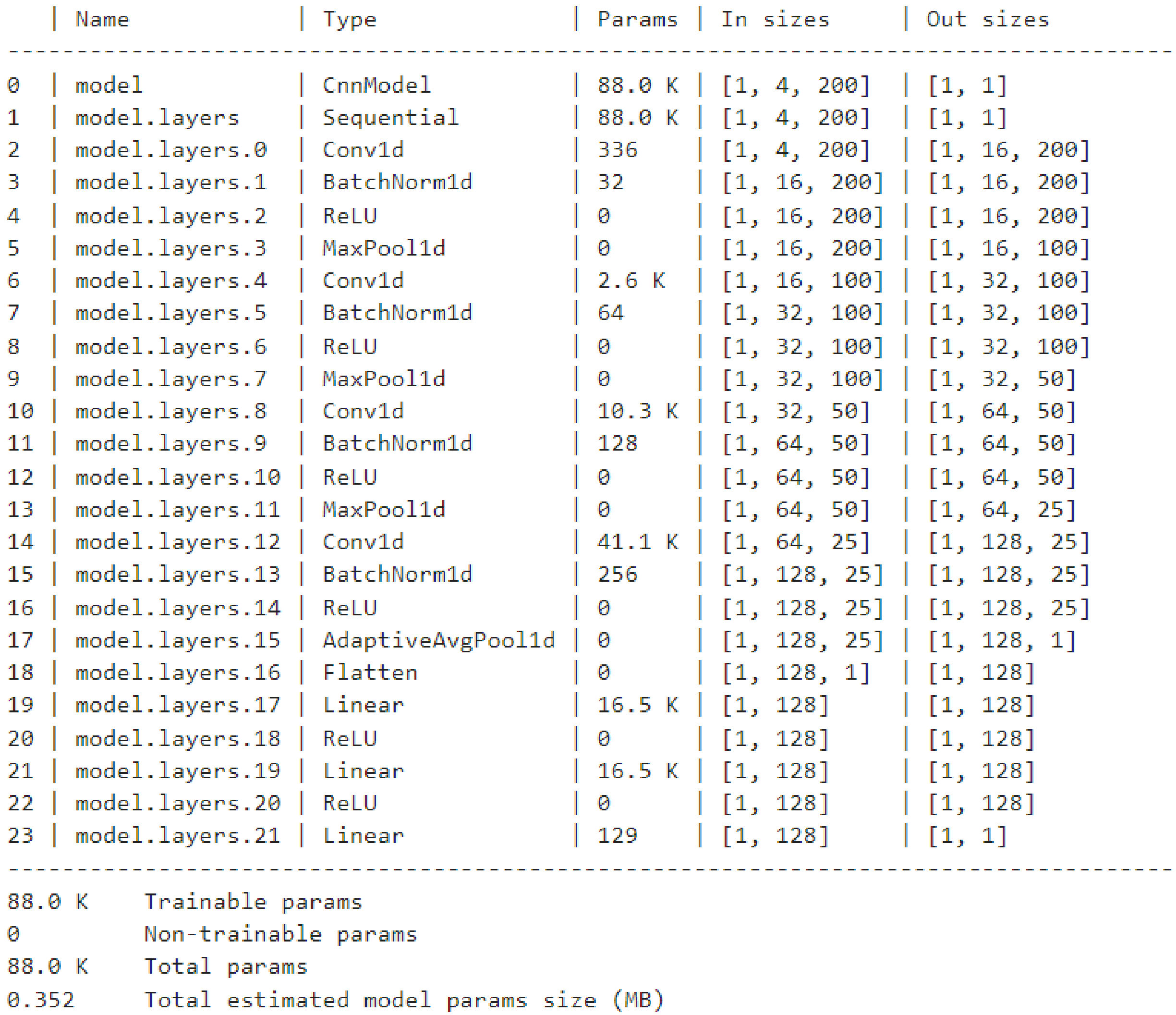
Difference between with or without acceleration signals?

Labs – Exercise 2

- **Heart rate classification**
  - CNN
  - Input: PPG + accelero
  - MSE loss function
  - Add batch normalization after each convolution layer



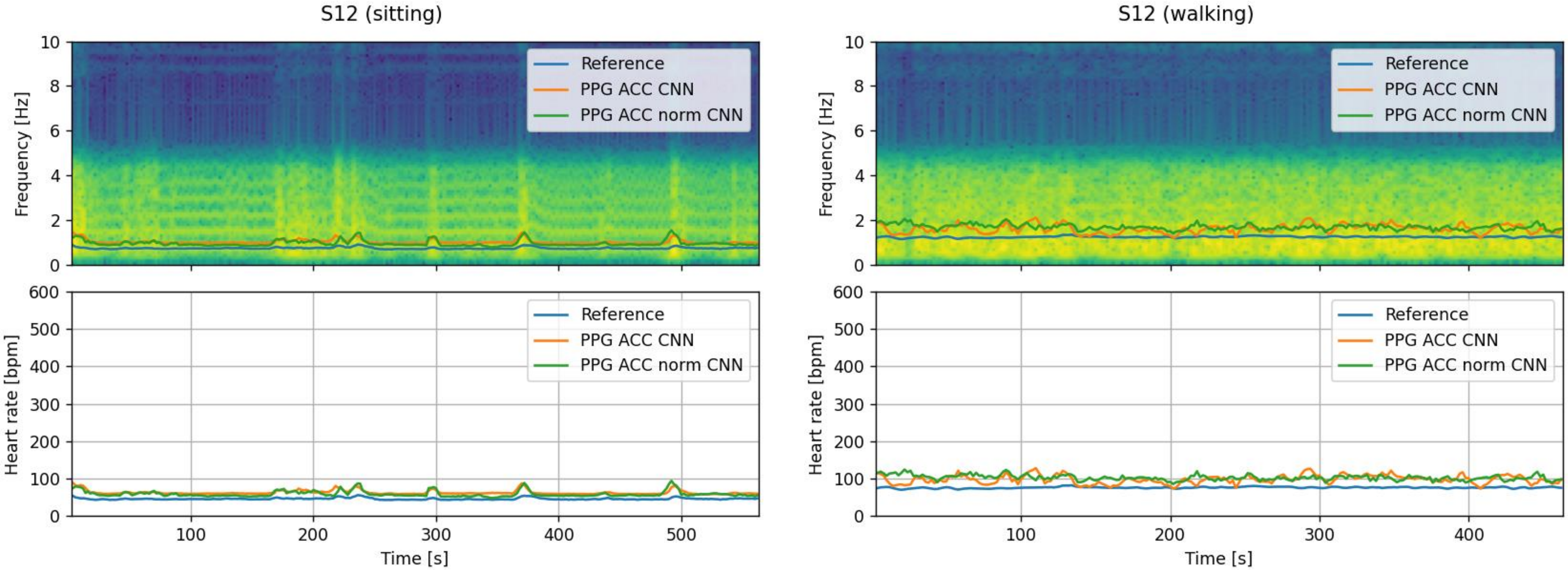
```
PPG ACC norm CNN config
{'model': {'input_shape': (4, 200),
  'output_shape': (1,),
  'n_convolutional_layers': 4,
  'kernel_size': 5,
  'n_initial_channels': 16,
  'use_normalization': True,
  'n_dense_layers': 3,
  'n_units': 128,
  'dropout': 0.0},
 'optimizer': {'lr': 0.0001}}
```



```
   | Name            | Type              | Params | In sizes       | Out sizes
-------------------------------------------------------------------------------------
0  | model           | CnnModel          | 88.0 K | [1, 4, 200]    | [1, 1]
1  | model.layers    | Sequential        | 88.0 K | [1, 4, 200]    | [1, 1]
2  | model.layers.0  | Conv1d            | 336    | [1, 4, 200]    | [1, 16, 200]
3  | model.layers.1  | BatchNorm1d       | 32     | [1, 16, 200]   | [1, 16, 200]
4  | model.layers.2  | ReLU              | 0      | [1, 16, 200]   | [1, 16, 200]
5  | model.layers.3  | MaxPool1d         | 0      | [1, 16, 200]   | [1, 16, 100]
6  | model.layers.4  | Conv1d            | 2.6 K  | [1, 16, 100]   | [1, 32, 100]
7  | model.layers.5  | BatchNorm1d       | 64     | [1, 32, 100]   | [1, 32, 100]
8  | model.layers.6  | ReLU              | 0      | [1, 32, 100]   | [1, 32, 100]
9  | model.layers.7  | MaxPool1d         | 0      | [1, 32, 100]   | [1, 32, 50]
10 | model.layers.8  | Conv1d            | 10.3 K | [1, 32, 50]    | [1, 64, 50]
11 | model.layers.9  | BatchNorm1d       | 128    | [1, 64, 50]    | [1, 64, 50]
12 | model.layers.10 | ReLU              | 0      | [1, 64, 50]    | [1, 64, 50]
13 | model.layers.11 | MaxPool1d         | 0      | [1, 64, 50]    | [1, 64, 25]
14 | model.layers.12 | Conv1d            | 41.1 K | [1, 64, 25]    | [1, 128, 25]
15 | model.layers.13 | BatchNorm1d       | 256    | [1, 128, 25]   | [1, 128, 25]
16 | model.layers.14 | ReLU              | 0      | [1, 128, 25]   | [1, 128, 25]
17 | model.layers.15 | AdaptiveAvgPool1d | 0      | [1, 128, 25]   | [1, 128, 1]
18 | model.layers.16 | Flatten           | 0      | [1, 128, 1]    | [1, 128]
19 | model.layers.17 | Linear            | 16.5 K | [1, 128]       | [1, 128]
20 | model.layers.18 | ReLU              | 0      | [1, 128]       | [1, 128]
21 | model.layers.19 | Linear            | 16.5 K | [1, 128]       | [1, 128]
22 | model.layers.20 | ReLU              | 0      | [1, 128]       | [1, 128]
23 | model.layers.21 | Linear            | 129    | [1, 128]       | [1, 1]
-------------------------------------------------------------------------------------
88.0 K    Trainable params
0         Non-trainable params
88.0 K    Total params
0.352     Total estimated model params size (MB)
```

Labs – Exercise 2

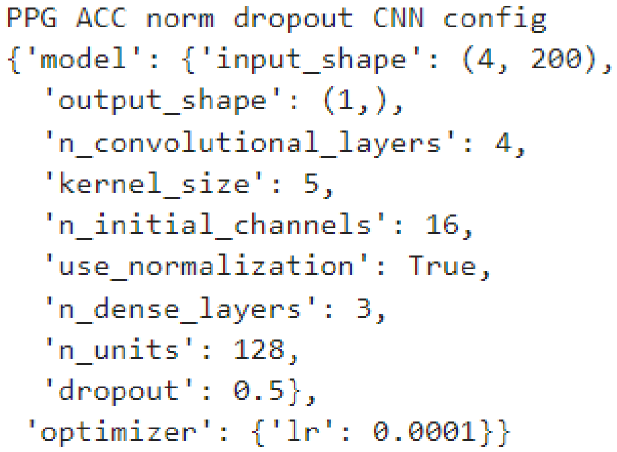
- **Heart rate classification**
  - HR prediction for validation set (with or without batch normalization)



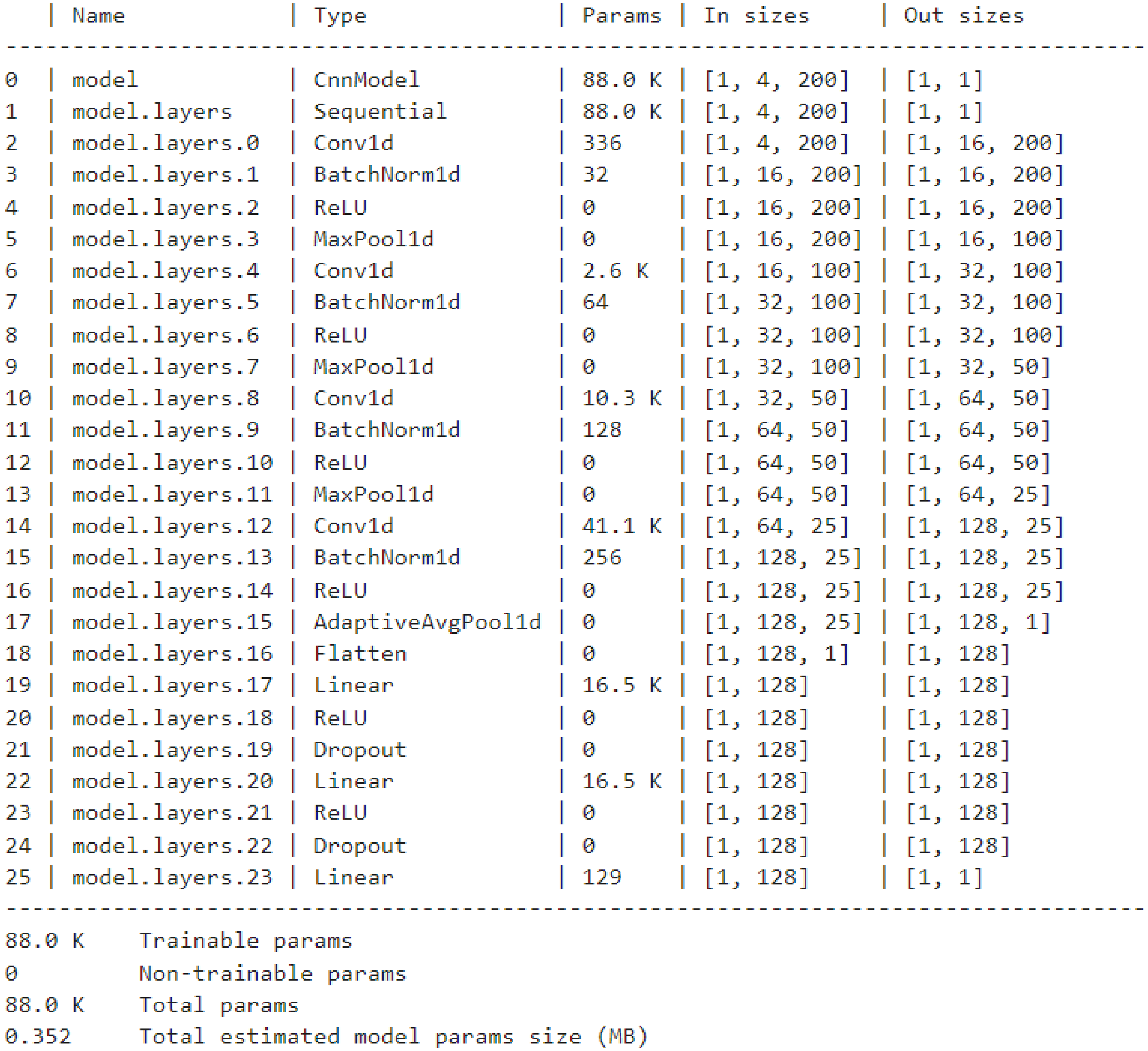
Difference between with or without batch normalization?

Labs – Exercise 2

- **Heart rate classification**
  - CNN
  - Input: PPG + accelero
  - MSE loss function
  - Batch normalization after each convolution layer
  - Add dropout after each dense layer (except last one)



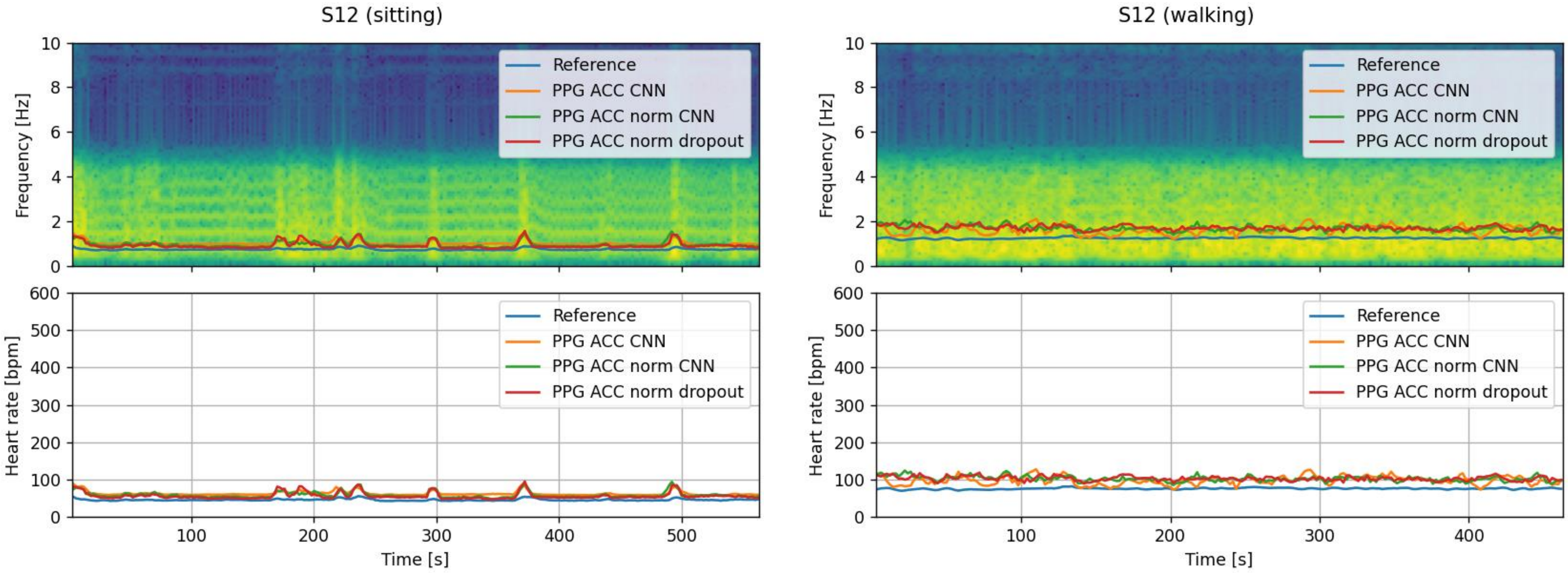
```
PPG ACC norm dropout CNN config
{'model': {'input_shape': (4, 200),
  'output_shape': (1,),
  'n_convolutional_layers': 4,
  'kernel_size': 5,
  'n_initial_channels': 16,
  'use_normalization': True,
  'n_dense_layers': 3,
  'n_units': 128,
  'dropout': 0.5},
 'optimizer': {'lr': 0.0001}}
```



```
   | Name            | Type              | Params | In sizes       | Out sizes
---------------------------------------------------------------------------------------
0  | model           | CnnModel          | 88.0 K | [1, 4, 200]    | [1, 1]
1  | model.layers    | Sequential        | 88.0 K | [1, 4, 200]    | [1, 1]
2  | model.layers.0  | Conv1d            | 336    | [1, 4, 200]    | [1, 16, 200]
3  | model.layers.1  | BatchNorm1d       | 32     | [1, 16, 200]   | [1, 16, 200]
4  | model.layers.2  | ReLU              | 0      | [1, 16, 200]   | [1, 16, 200]
5  | model.layers.3  | MaxPool1d         | 0      | [1, 16, 200]   | [1, 16, 100]
6  | model.layers.4  | Conv1d            | 2.6 K  | [1, 16, 100]   | [1, 32, 100]
7  | model.layers.5  | BatchNorm1d       | 64     | [1, 32, 100]   | [1, 32, 100]
8  | model.layers.6  | ReLU              | 0      | [1, 32, 100]   | [1, 32, 100]
9  | model.layers.7  | MaxPool1d         | 0      | [1, 32, 100]   | [1, 32, 50]
10 | model.layers.8  | Conv1d            | 10.3 K | [1, 32, 50]    | [1, 64, 50]
11 | model.layers.9  | BatchNorm1d       | 128    | [1, 64, 50]    | [1, 64, 50]
12 | model.layers.10 | ReLU              | 0      | [1, 64, 50]    | [1, 64, 50]
13 | model.layers.11 | MaxPool1d         | 0      | [1, 64, 50]    | [1, 64, 25]
14 | model.layers.12 | Conv1d            | 41.1 K | [1, 64, 25]    | [1, 128, 25]
15 | model.layers.13 | BatchNorm1d       | 256    | [1, 128, 25]   | [1, 128, 25]
16 | model.layers.14 | ReLU              | 0      | [1, 128, 25]   | [1, 128, 25]
17 | model.layers.15 | AdaptiveAvgPool1d | 0      | [1, 128, 25]   | [1, 128, 1]
18 | model.layers.16 | Flatten           | 0      | [1, 128, 1]    | [1, 128]
19 | model.layers.17 | Linear            | 16.5 K | [1, 128]       | [1, 128]
20 | model.layers.18 | ReLU              | 0      | [1, 128]       | [1, 128]
21 | model.layers.19 | Dropout           | 0      | [1, 128]       | [1, 128]
22 | model.layers.20 | Linear            | 16.5 K | [1, 128]       | [1, 128]
23 | model.layers.21 | ReLU              | 0      | [1, 128]       | [1, 128]
24 | model.layers.22 | Dropout           | 0      | [1, 128]       | [1, 128]
25 | model.layers.23 | Linear            | 129    | [1, 128]       | [1, 1]
---------------------------------------------------------------------------------------
88.0 K    Trainable params
0         Non-trainable params
88.0 K    Total params
0.352     Total estimated model params size (MB)
```

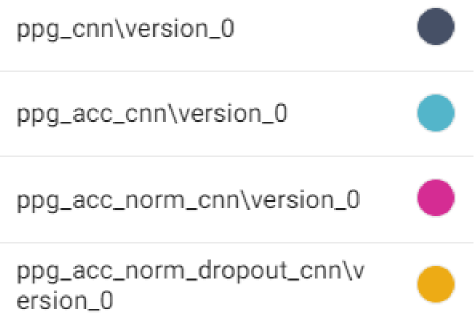
Labs – Exercise 2

- **Heart rate classification**
  - HR prediction for validation set (with or without dropout)

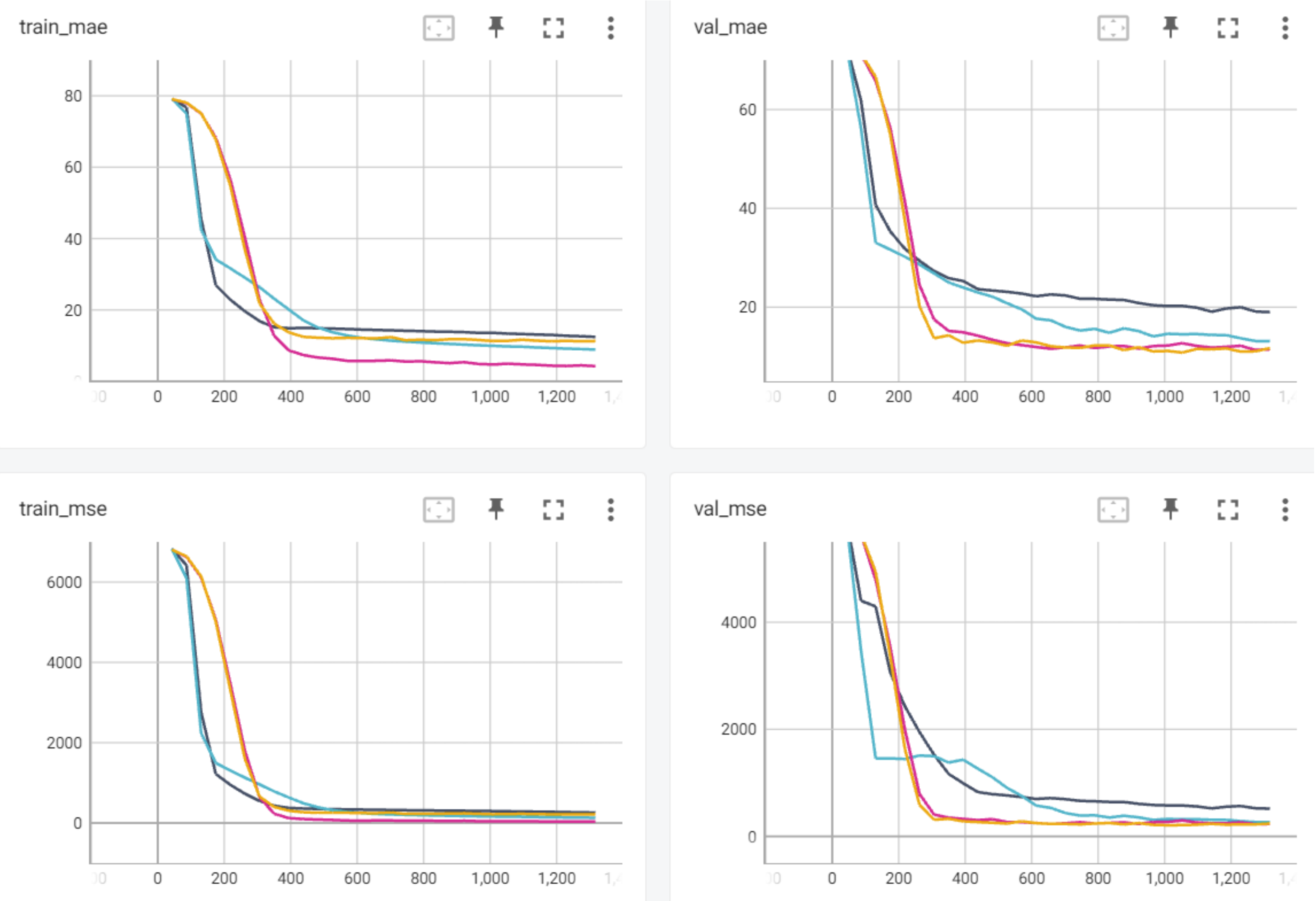


Labs – Exercise 2

- **Heart rate classification**
  - Tensorboard



ppg_cnn\version_0	●
ppg_acc_cnn\version_0	●
ppg_acc_norm_cnn\version_0	●
ppg_acc_norm_dropout_cnn\version_0	●



Labs – Exercise 2

- **Heart rate classification**
  - Evaluation

	Model	Training	Validation	Test
MSE	PPG	253.15	518.95	199.11
	PPG + ACC	127.88	263.46	198.26
	PPG + ACC + batch norm	26.12	232.19	71.65
	PPG + ACC + batch norm + dropout	39.89	239.53	49.74

	Model	Training	Validation	Test
MAE	PPG	12.37	19.01	12.43
	PPG + ACC	8.78	13.10	12.05
	PPG + ACC + batch norm	3.75	11.48	6.16
	PPG + ACC + batch norm + dropout	4.68	11.72	5.07