Deep Learning for Sensing: Emotion Recognition

Why Deep Learning?

- Availability of bigger and better-quality datasets (e.g., ImageNet)
- Better compute available; i.e., faster and cheaper GPUs
- Better algorithms (e.g., model architecture, optimizer, and training procedure) and tools (e.g., Keras)
- Availability of pretrained models that have taken months to train but can be quickly reused

Model Zoo

Table 2-1. Architectural details of select pretrained ImageNet models

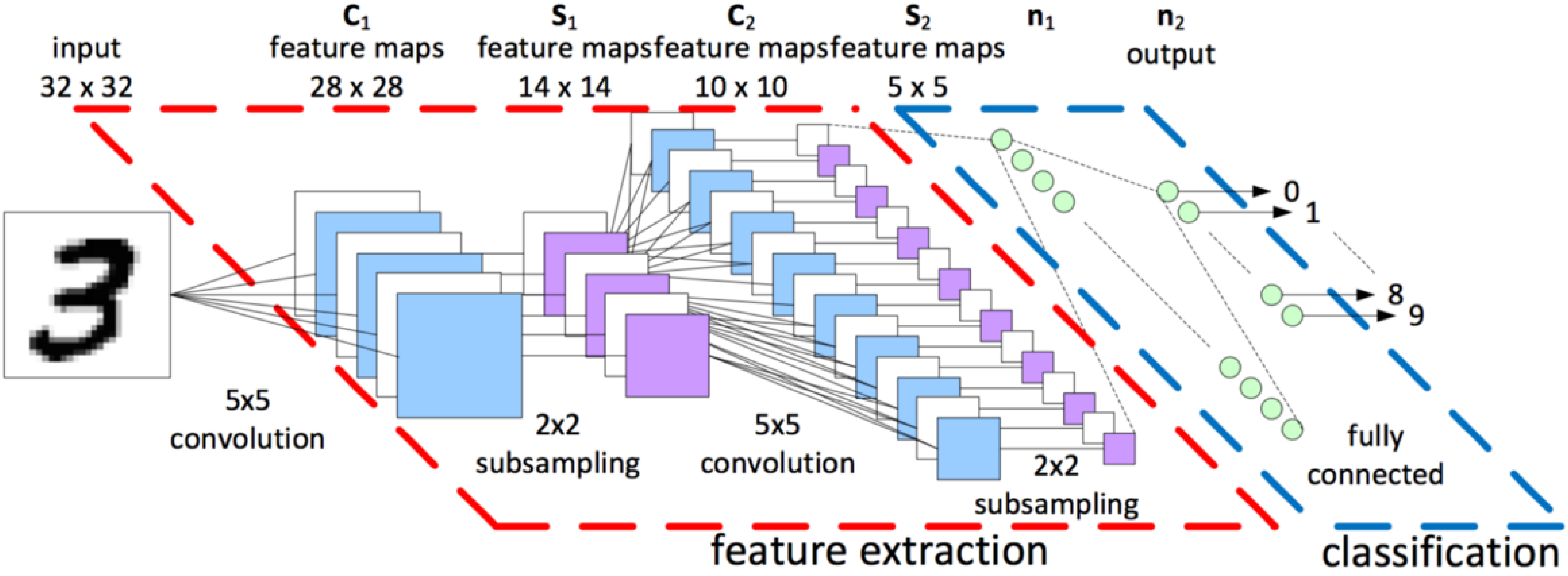
Model	Size	Top-1 accuracy	Top-5 accuracy	Parameters	Depth
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.9	143,667,240	26
ResNet-50	98 MB	0.749	0.921	25,636,712	50
ResNet-101	171 MB	0.764	0.928	44,707,176	101
ResNet-152	232 MB	0.766	0.931	60,419,944	152
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
NASNetMobile	23 MB	0.744	0.919	5,326,716	—
NASNetLarge	343 MB	0.825	0.96	88,949,818	—
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88


Learning Melodica From Scratch?
Effort = 3 months
Already Play Piano?
Fine-tune Skills
Effort = 3 days

Transfer Learning

- Training a deep learning model from scratch on a multimillion-image database requires weeks of training time and lots of GPU computational energy, making it a difficult task
- The model zoo *(https://keras.io/applications/)* in Keras is a collection of various architectures trained using the Keras framework on the ImageNet dataset
- Possible to perform transfer learning that simply modifies existing models by training our own classifier in minutes

Feature Extraction vs. Classification



https://medium.com/analytics-vidhya/convolutional-neural-networks-cnn-explained-step-by-step-69137a54e5e7

Generic -> Task specific layers (left to right)

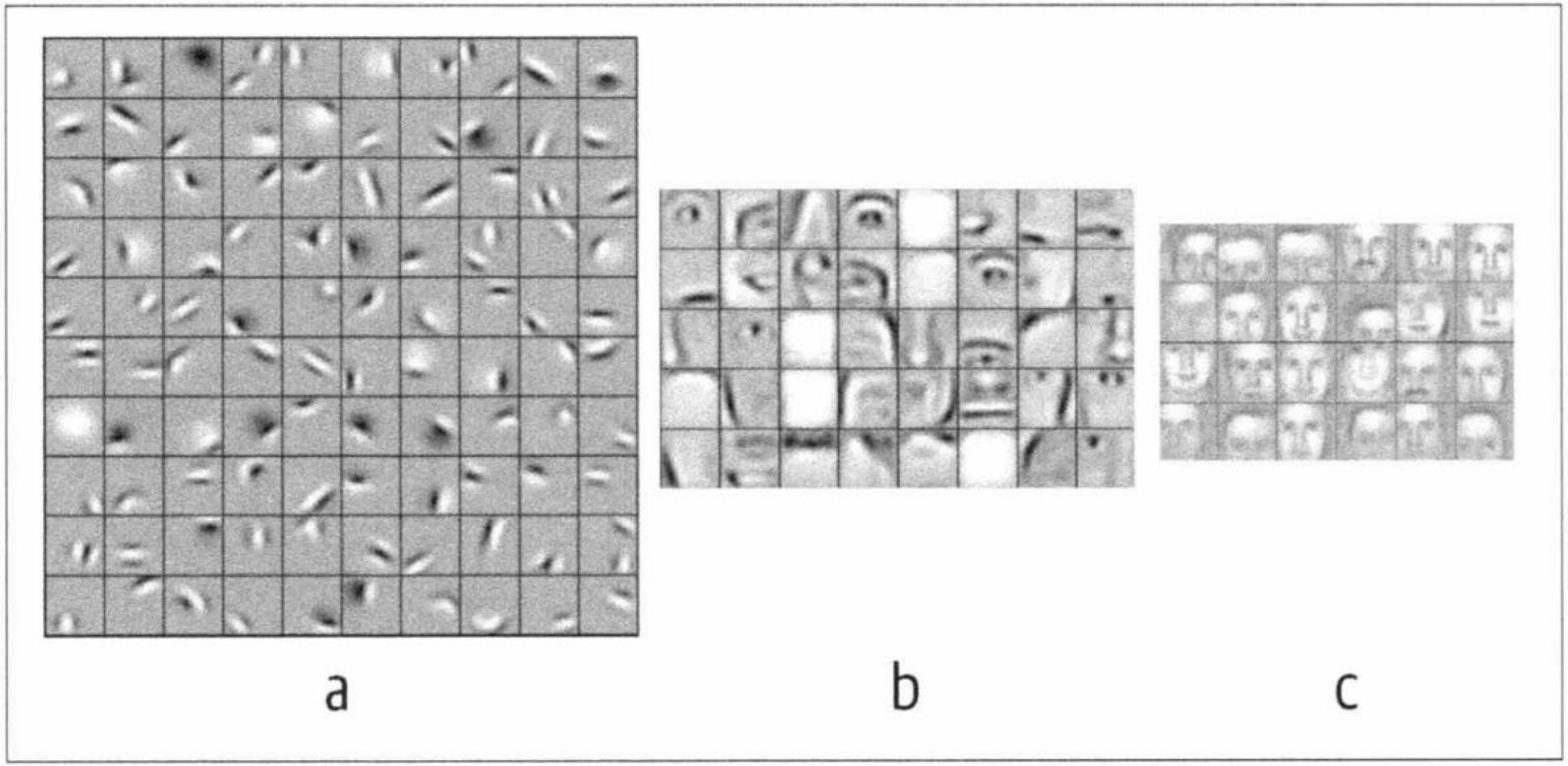
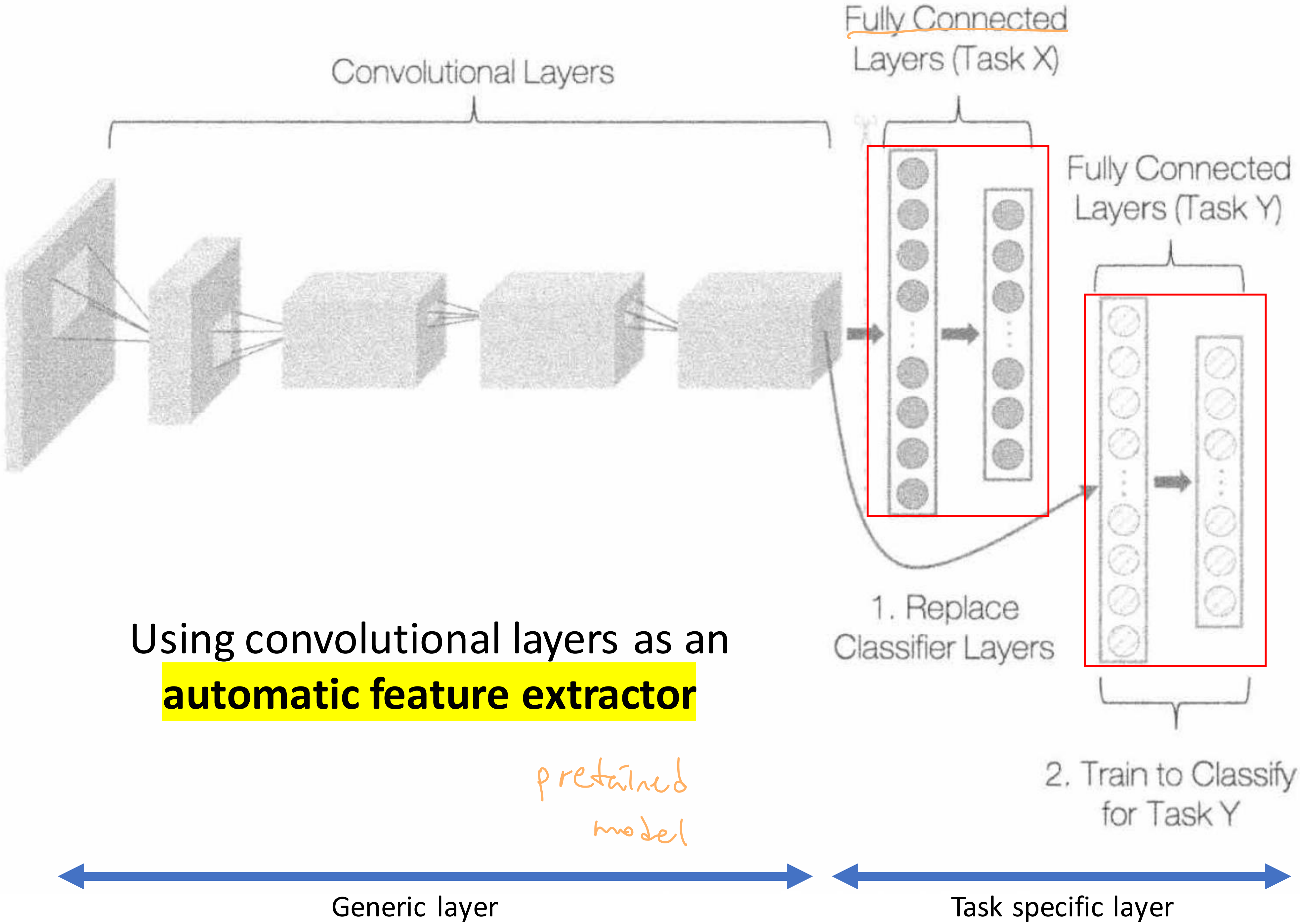


Figure 3-3. (a) Lower-level activations, followed by (b) midlevel activations and (c) upper-layer activations (image source: Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations, Lee et al., ICML 2009)

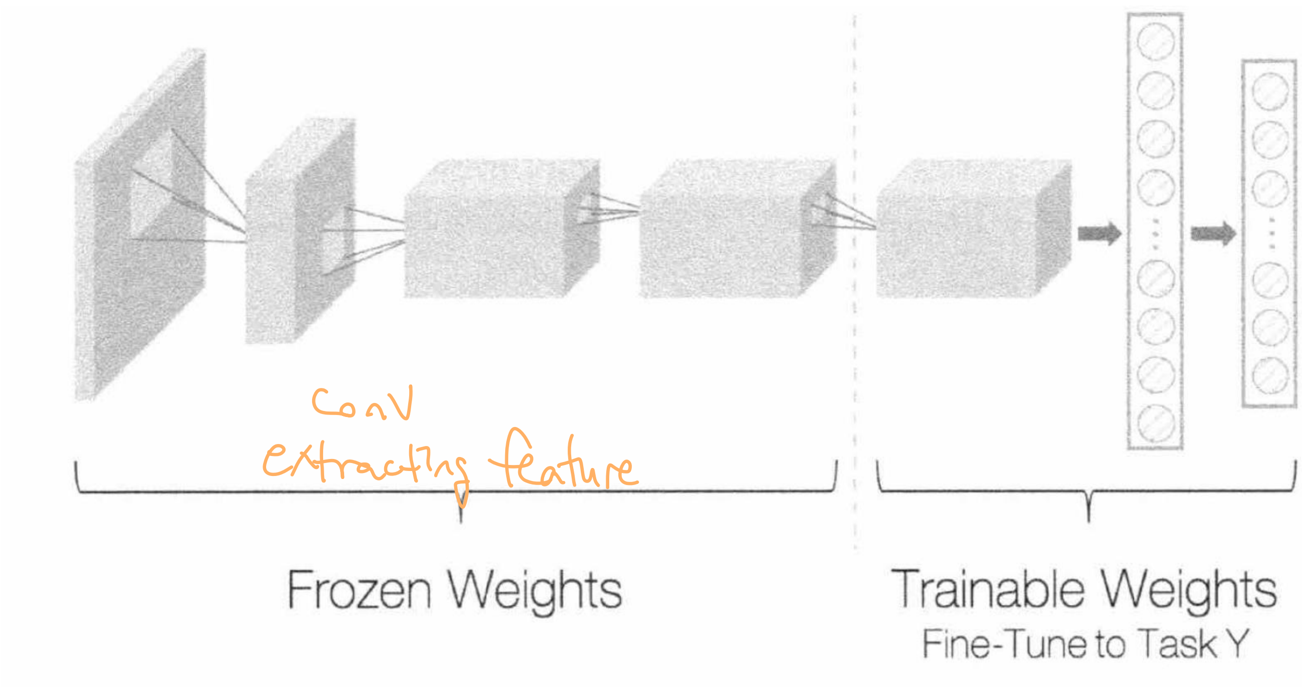


Using convolutional layers as an
automatic feature extractor

pretrained
model

Transfer Learning

- Neural networks are commonly used for task adaptation; that's why it's called "fine-tuning"
- But other classifiers could be also used such as decision trees, gradient boost, and SVM



When to use Transfer Learning

Task Similarity

Training Dataset size		**High similarity (task & datasets)**	**Low similarity (task & datasets)**
	Large amount of training data	Fine tune all layers	Train from scratch, or fine tune all layers
	Small amount of training data	Fine tune last few layers	Tough luck! Train on a smaller network with heavy data augmentation or somehow get more data

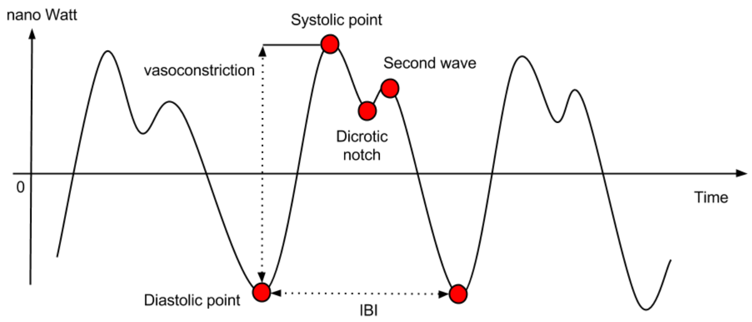
Emotion Recognition w/ K-EmoCon Dataset

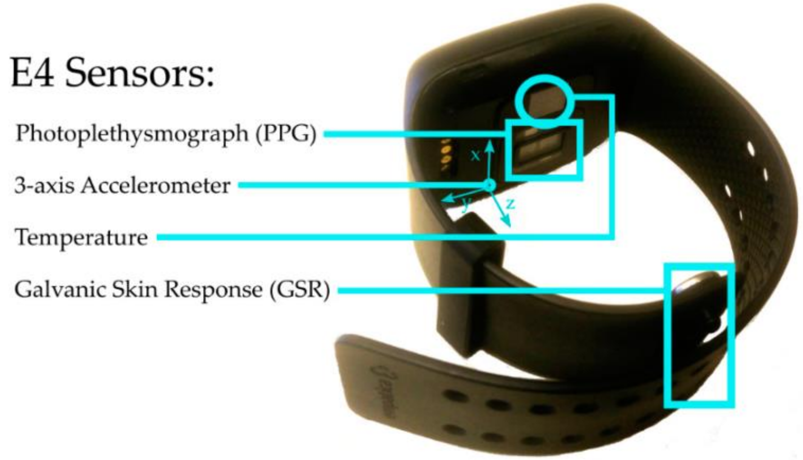
Signals

https://support.empatica.com/hc/en-us/articles/360029719792-E4-data-BVP-expected-signal

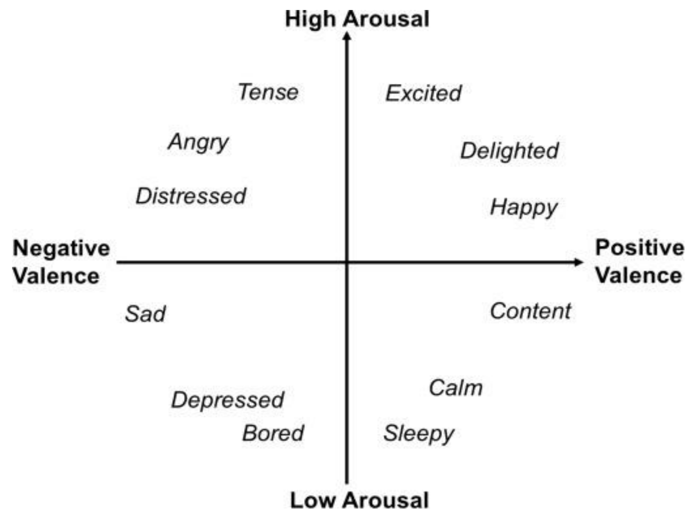
- Empatica E4
  - BVP – PPG raw data
    - Blood Volume Pulse (BVP) is the primary output from the PPG sensor
  - EDA – GSR data (skin)
  - Skin temperature data
- Polar H10
  - ECG - heart rate data

BVP





Ground Truth

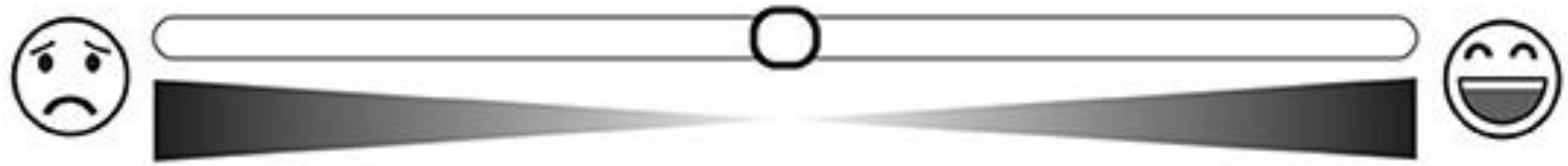


Arousal – scale (1-5) neutral = 3

Low arousal/alertness

High arousal/alertness

Valence – scale (1-5) neutral = 3

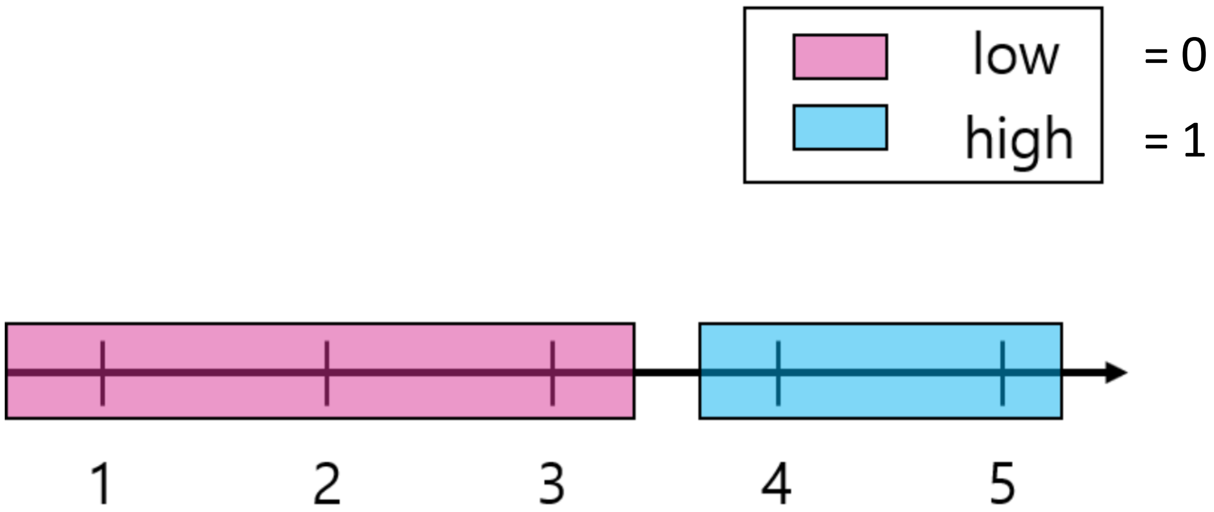


Negative feeling
(Sad)

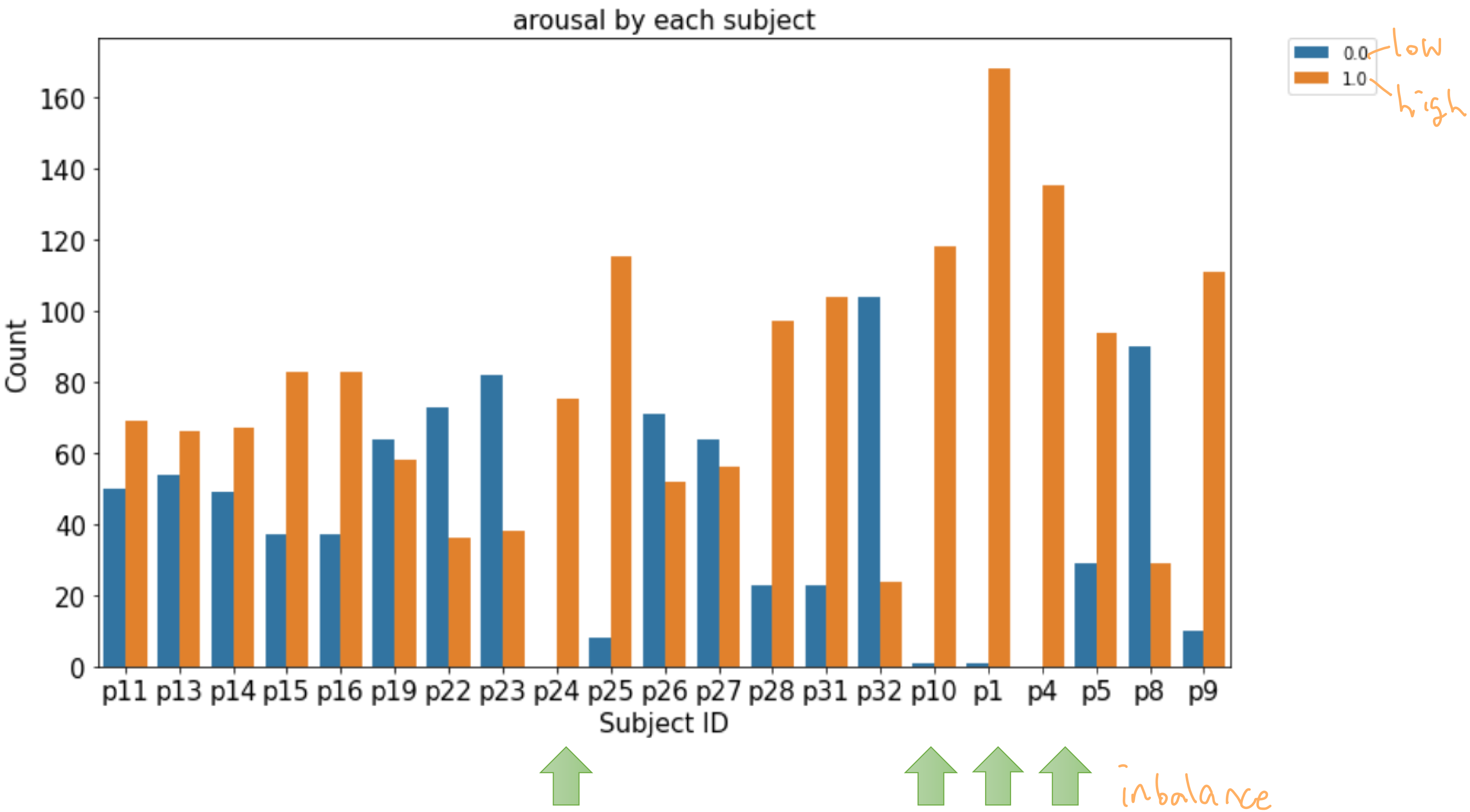
Positive feeling
(Happy)

Ground Truth – Binary Label

- Low vs. High
  - Low: 1, 2, 3
  - High: 4, 5



Skewed Labels



Signals - Preparation

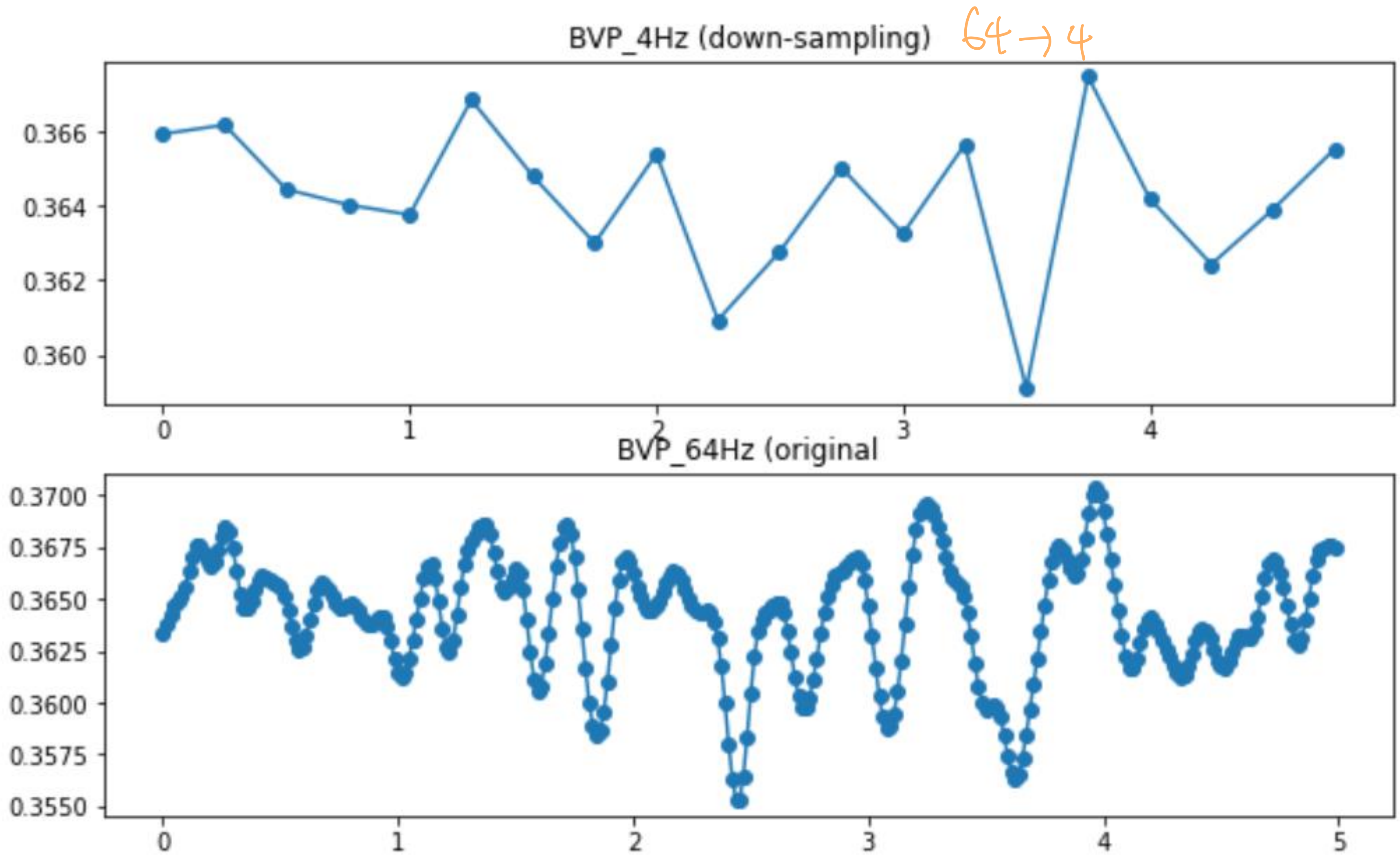
- Different sampling rates for each sensor type
  - BVP/PPG: 64Hz ↓
  - ECG: 1Hz ↑
  - EDA: 4Hz
  - Skin Temperature: 4Hz
- Problem: can’t feed these four items together to the deep learning model
  - Solution – setting a uniform sampling rate of 4 Hz
    - BVP => down sampling (to 4 Hz) via Rolling Mean
    - ECG => up sampling (to 4 Hz) via Linear Interporation

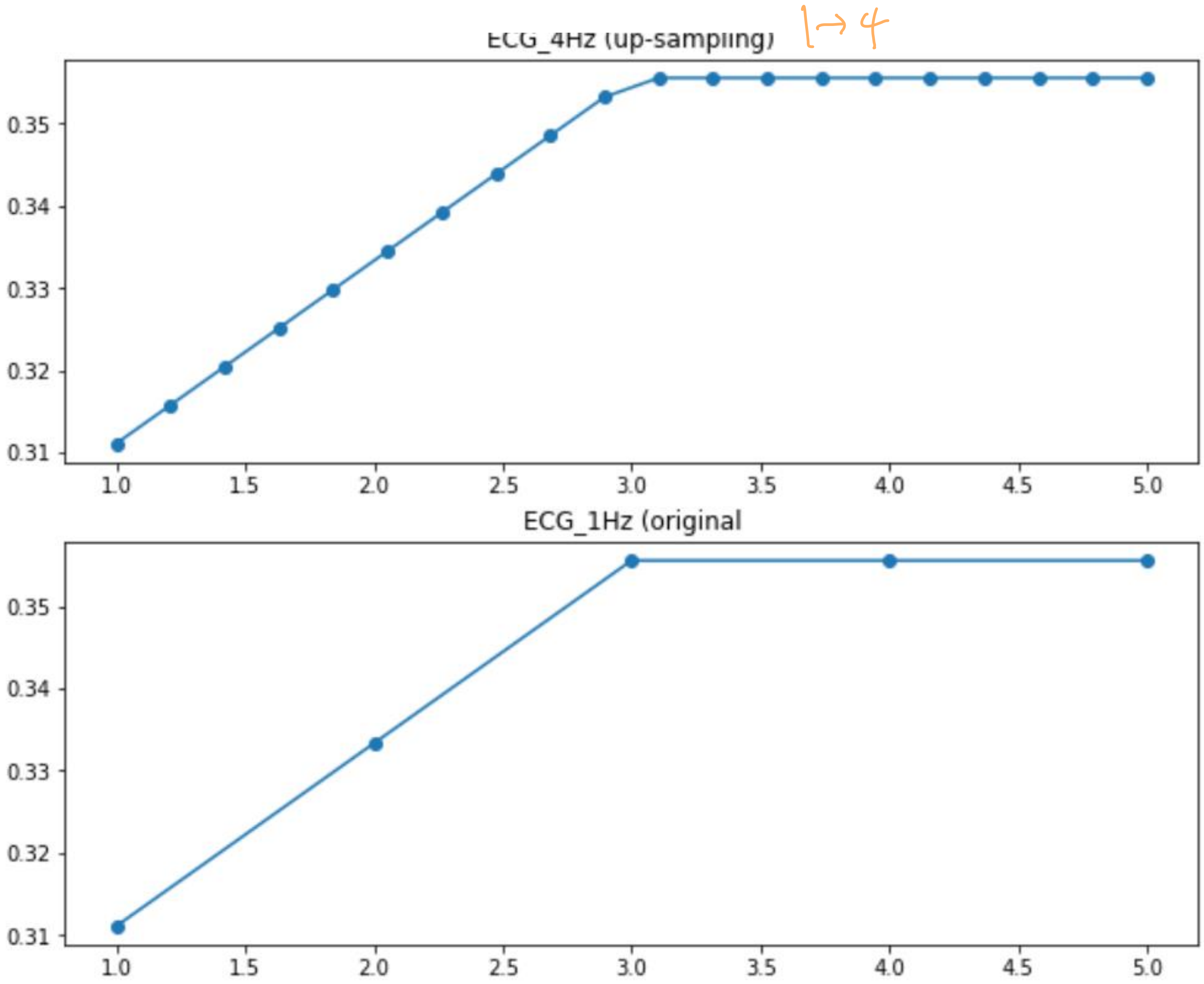
Rolling average as low pass filtering

	bvp_64Hz
0	0.363333
1	0.363816
2	0.364284
3	0.364627
4	0.364861
5	0.365141
6	0.365624
7	0.366310
8	0.367027
9	0.367526
10	0.367604
11	0.367277
12	0.366840
13	0.366591
14	0.366762
15	0.367339
16	0.368025
17	0.368414
18	0.368259
19	0.367479

	bvp_64Hz
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN
5	NaN
6	NaN
7	NaN
8	NaN
9	NaN
10	NaN
11	NaN
12	NaN
13	NaN
14	NaN
15	0.365935
16	0.366228
17	0.366516
18	0.366764
19	0.366942

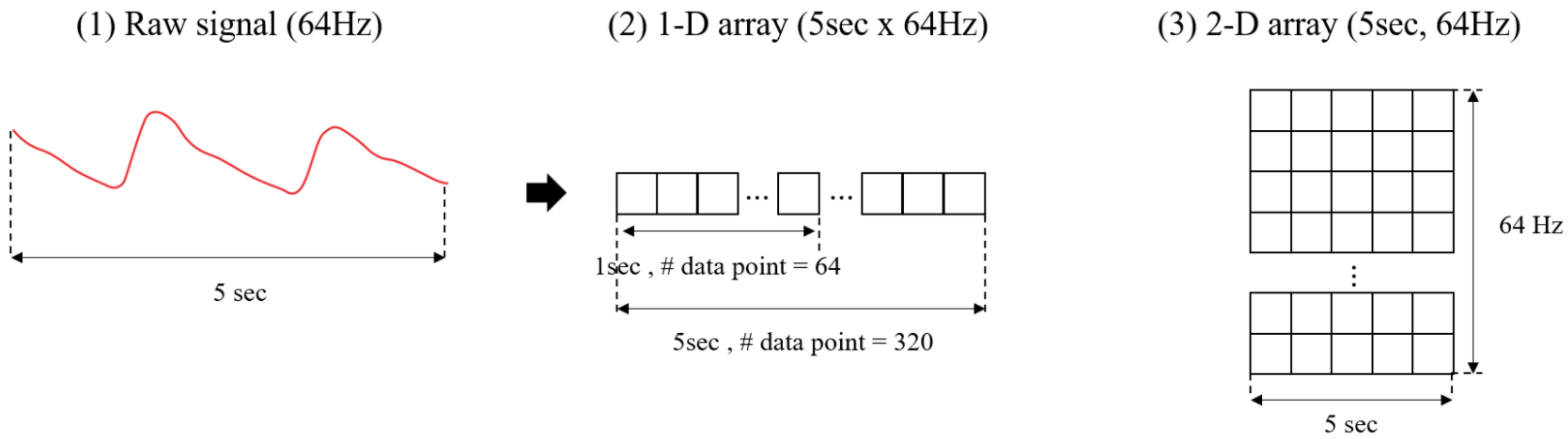
	bvp_4Hz
15	0.365935
31	0.366178
47	0.364449
63	0.364034
79	0.363763
95	0.366854
111	0.364801
127	0.363002
143	0.365395
159	0.360940
175	0.362771
191	0.365032
207	0.363256
223	0.365649
239	0.359100
255	0.367475
271	0.364199
287	0.362433
303	0.363910
319	0.365536


BVP_4Hz (down-sampling)
64 → 4
0.366
0.364
0.362
0.360
0
1
2
3
4
BVP_64Hz (original
0.3700
0.3675
0.3650
0.3625
0.3600
0.3575
0.3550
0
1
2
3
4
5

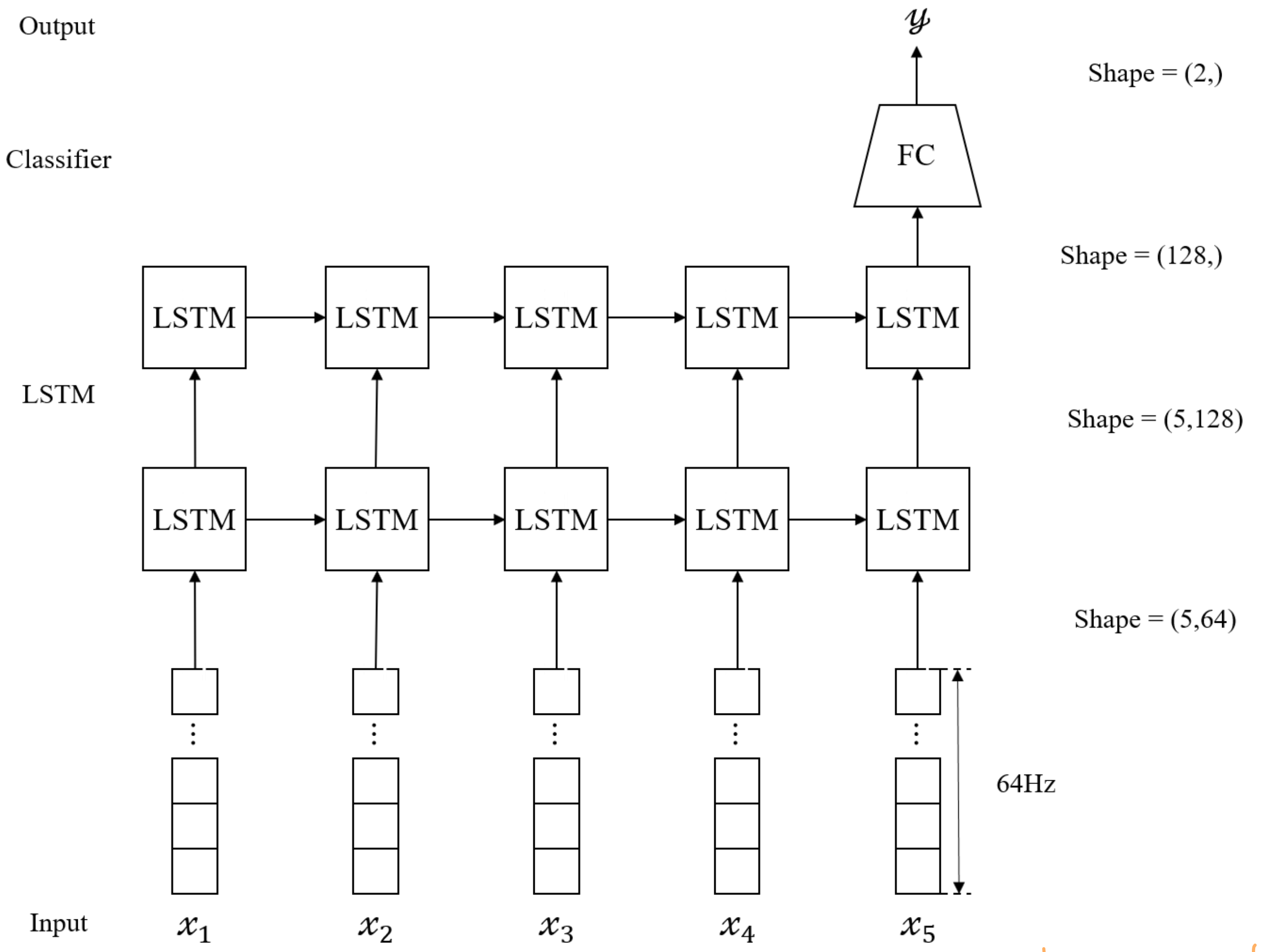

ECG_4Hz (up-sampling)
1→4
0.35
0.34
0.33
0.32
0.31
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
ECG_1Hz (original
0.35
0.34
0.33
0.32
0.31
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

BVP Example (64Hz)

- How to feed into a neural network?
  - Sample at 1 second – keep 5 seconds



LSTM - BVP Example (64Hz)



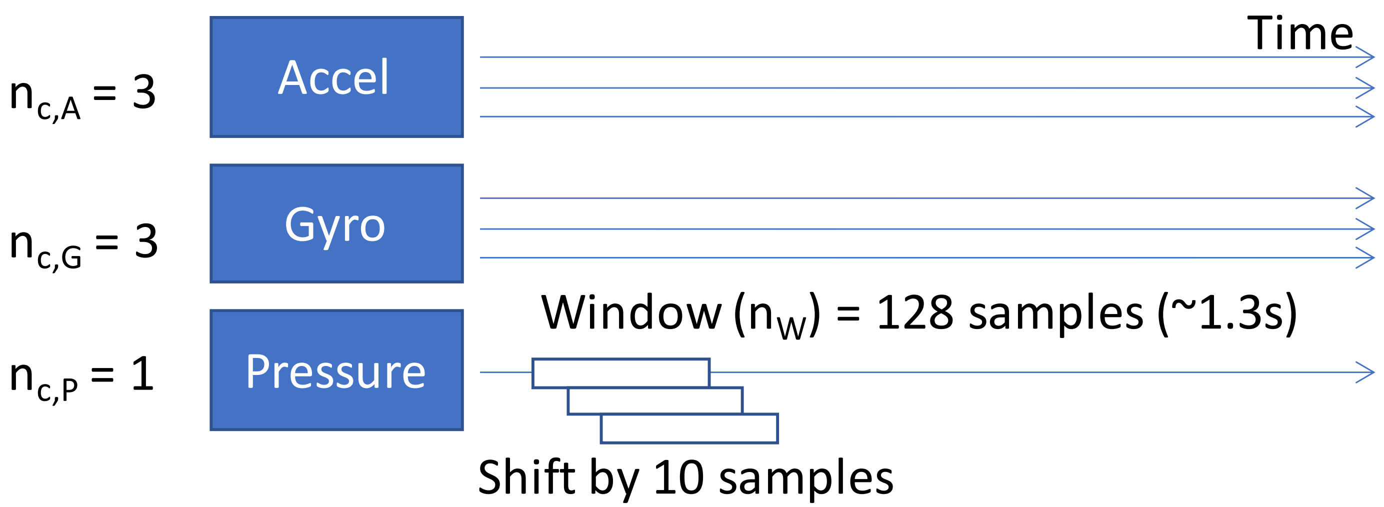
5sec data → 1 input

LSTM - BVP Example (64Hz)

- What will happen if we increase the time period? (from 5 seconds to 20 seconds or even 1 minute?)

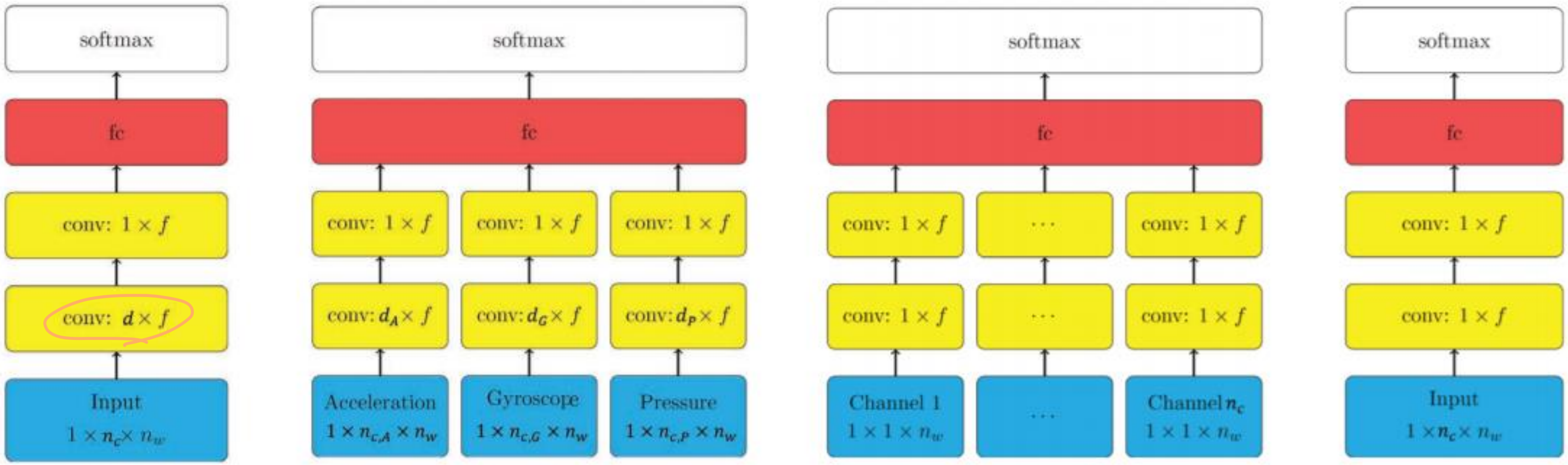
Sensor Fusion w/ Deep Learning

- Example scenario: Activity Recognition
  - 3D Accelerometer = 3 streams (X, Y, Z)
    - 14 bit resolution per sample; sampling rate: 100 Hz
  - 3D Gyroscope = 3 streams (X, Y, Z)
    - 16 bit resolution per sample; sampling rate: 100 Hz
  - 1D Pressure = 1 stream
    - 16 bit resolution per sample; sampling rate: 100 Hz



How to feed into neural networks?

Sensor Fusion w/ Deep Learning



(a) Early fusion

(b) Sensor-based late fusion

LF

(c) Channel-based late fusion

(d) Shared filters hybrid fusion

Early fusion (EF)

$d = n_c$

Sensor-based late fusion (SB-LF)

$d_A = n_{c,A} = 3$
$d_G = n_{c,G} = 3$
$d_P = n_{c,P} = 1$

Channel-based late fusion (CB-LF)

Shared filters hybrid fusion (SF-HF)

It's CB-LF, but it ***uses the same filters*** *are used for all input channels (best performing)*

CNN-based sensor fusion techniques for multimodal human activity recognition, ISWC 2017

Layer	conv1	conv2	FC	Soft-max
Parameters	W^1, b^1	W^2, b^2	W^{fc}, b^{fc}	W^{sm}, b^{sm}
EF	704	4,096	1,016,064	1,799
SB-LF	768	12,288	3,047,680	
CB-LF	896	28,672	7,110,912	
SF-HF	128	4,096	7,110,912	

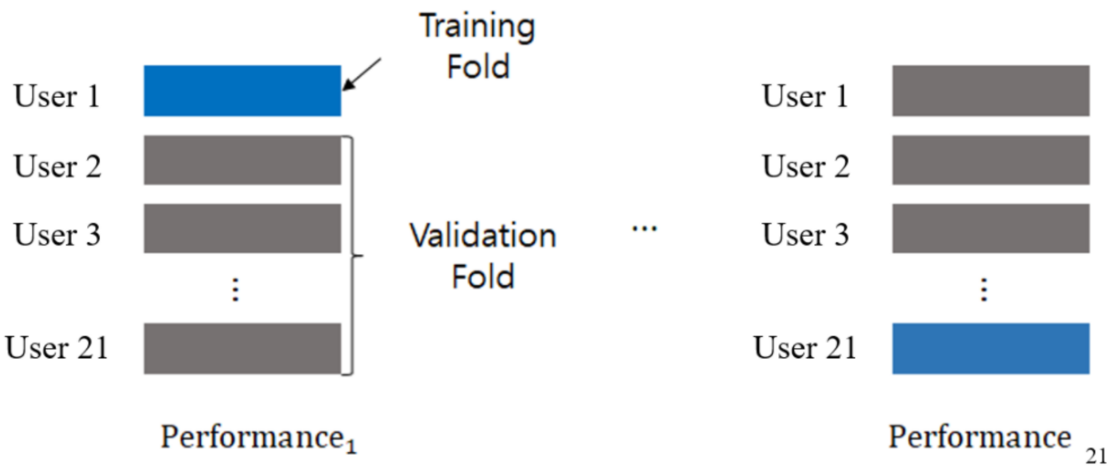
Table 2. Overview of number of parameters per layer in different sensor fusion techniques, exemplary shown for the optimal two layered CNN architecture with $n_f = 32$ in both convolutional layers (denoted as conv1 and conv2) and $n_n = 256$ in the fully connected layer (FC).

Fusion technique		EF	SB-LF	CB-LF	SF-HF
2L-CNN	mean	0.74	0.76	0.81	**0.86**
	std	0.013	0.043	0.045	0.034
3L-CNN	mean	0.74	0.81	0.81	**0.85**
	std	0.016	0.052	0.05	0.029

Table 3. Average F_1-scores achieved on PAMAP2 for different sensor fusion models (EF, SB-LF, CB-LF and SF-HF) with two and three layered CNNs using zNorm+BN normalization.

Validation

- Generalized model : LOSO



Result= mean ± SD

$$\text{mean}= \frac{1}{21} \sum_{i=1}^{21} \text{Performance}_i$$

$$\text{SD} = \sqrt{\frac{1}{21} \sum_{i=1}^{21} (\text{Performance}_i - \text{mean})^2}$$