Autoencoder-based Hybrid Neural EEG/ERP Data Compression

Case-Study for 256-channel Binocular Rivalry Dataset

Contents

1. Introduction

- **Goal –** test the ENCODER-DECODER ANN architecture for EEG/ERP data compression
- **Lossless** or **almost lossless** compression
- **Compression from the point of ML** $\Longrightarrow$ as knowledge derived from data

Work still in progress!

1.1. Used Tools

PyTorch

NumPy

MNE
MEG + EEG ANALYSIS & VISUALIZATION

statsmodels

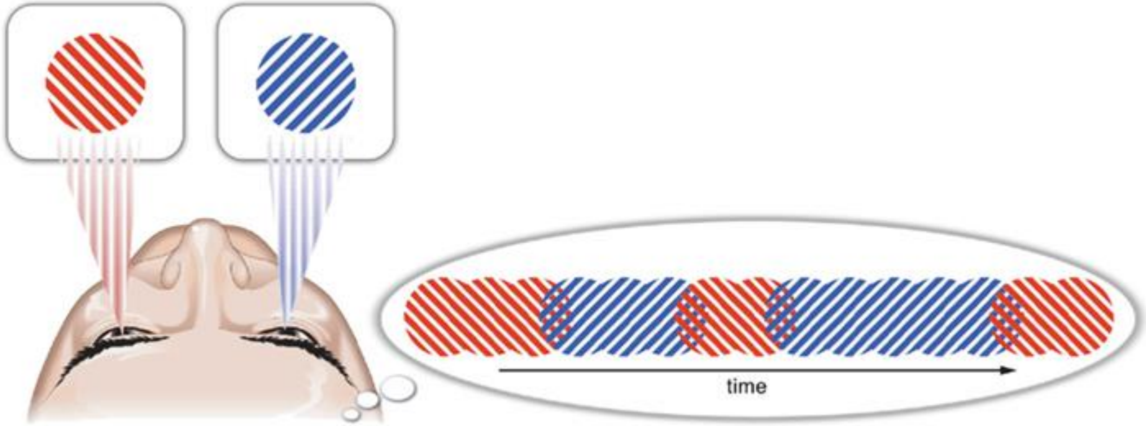
matplotlib

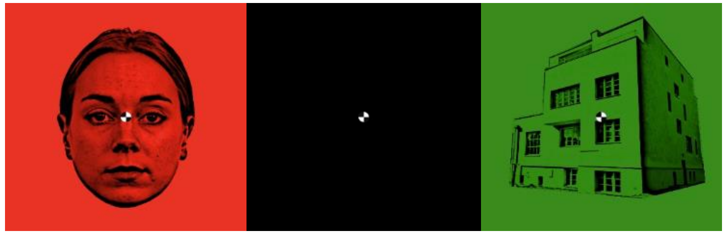
seaborn

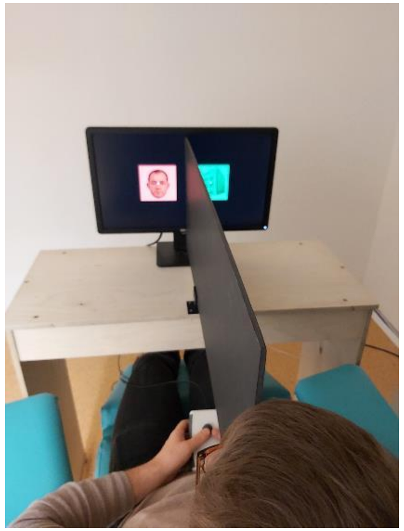
NVIDIA
CUDA

2. Dataset

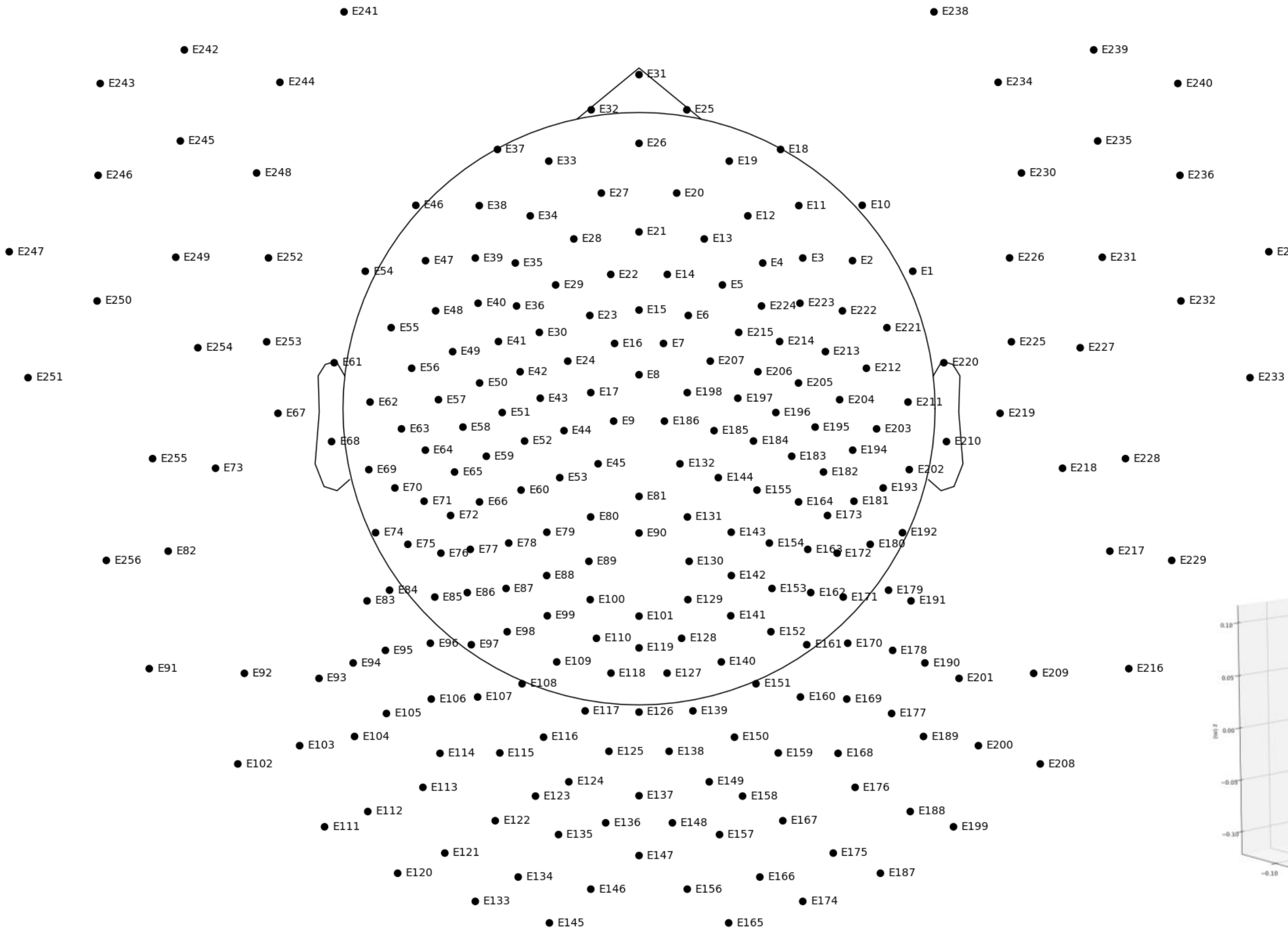
- **NIMH** (National Institute of Mental Health)
- **256**-electrode EEG
- **25 subjects**
- **Binocular rivalry** – two stimuli, one in each eye

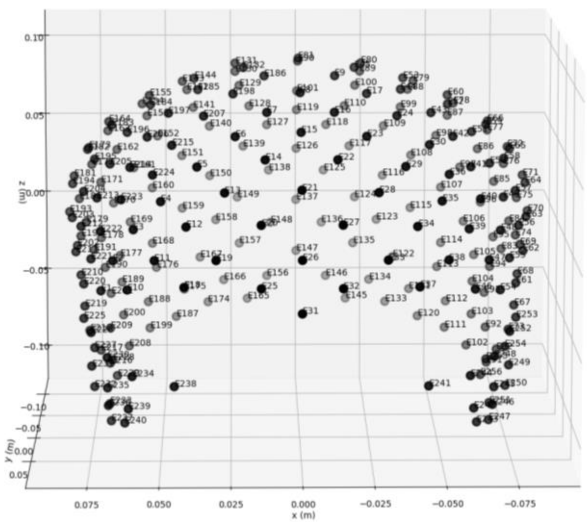


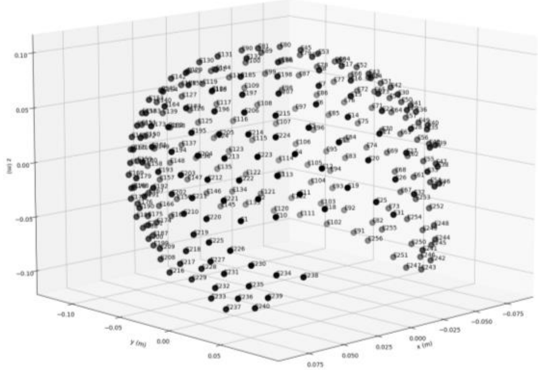


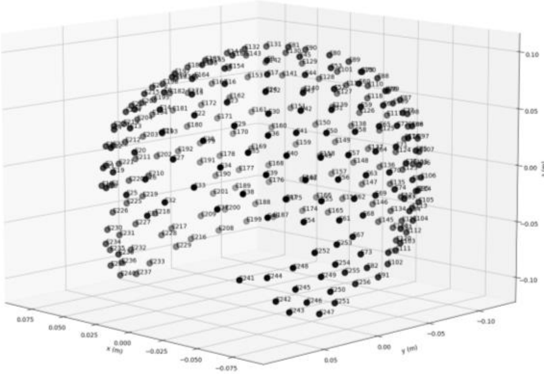


2.1. Electrode Placement









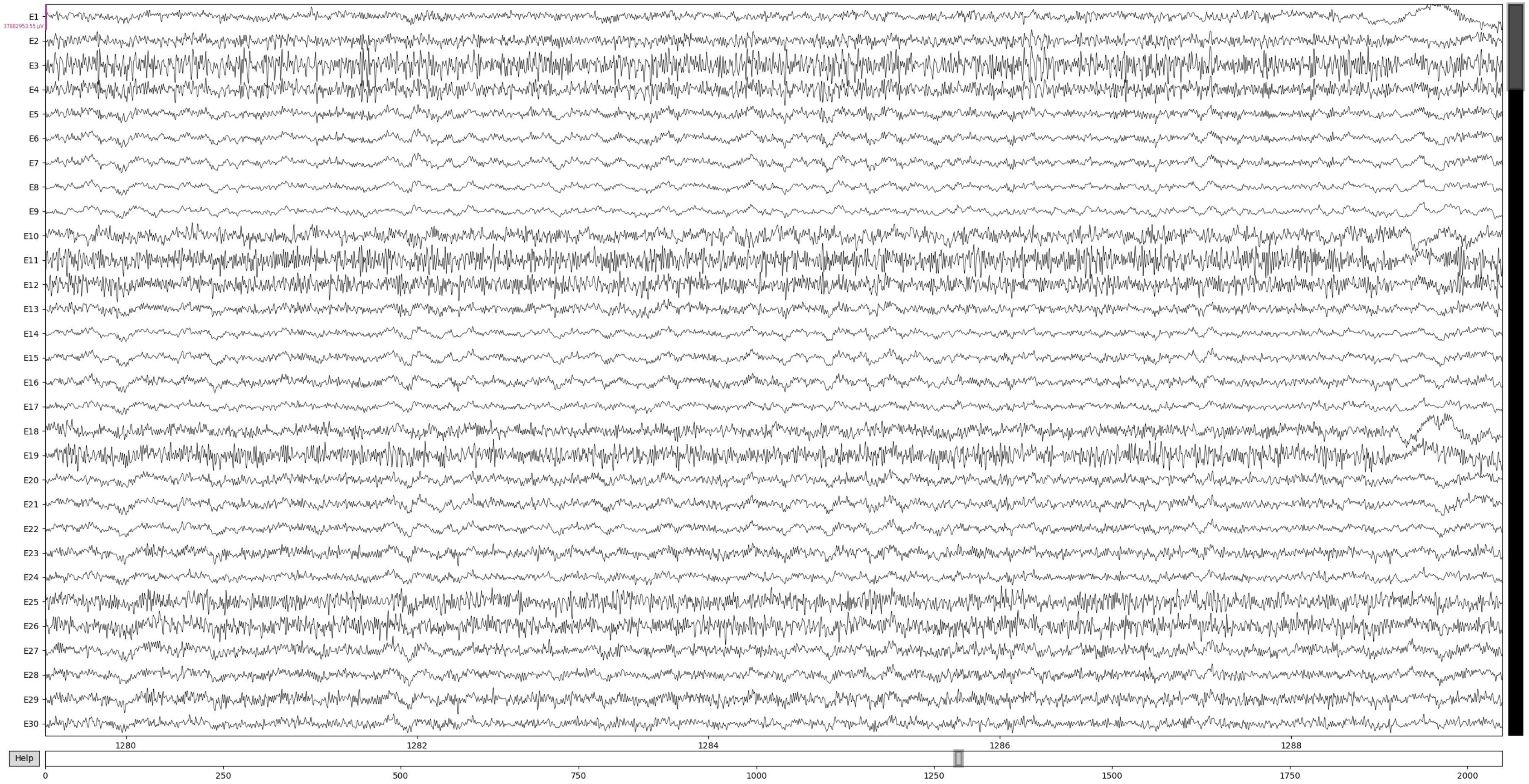
2.2. File Details

- **56.3 GB** of **.MAT** files $\Longrightarrow$ HDF5 already compressed with GZIP
- Contains extra 4 channels and 5 sec of zeroed segments at the start/end
- After **loading and extracting EEG signals**:

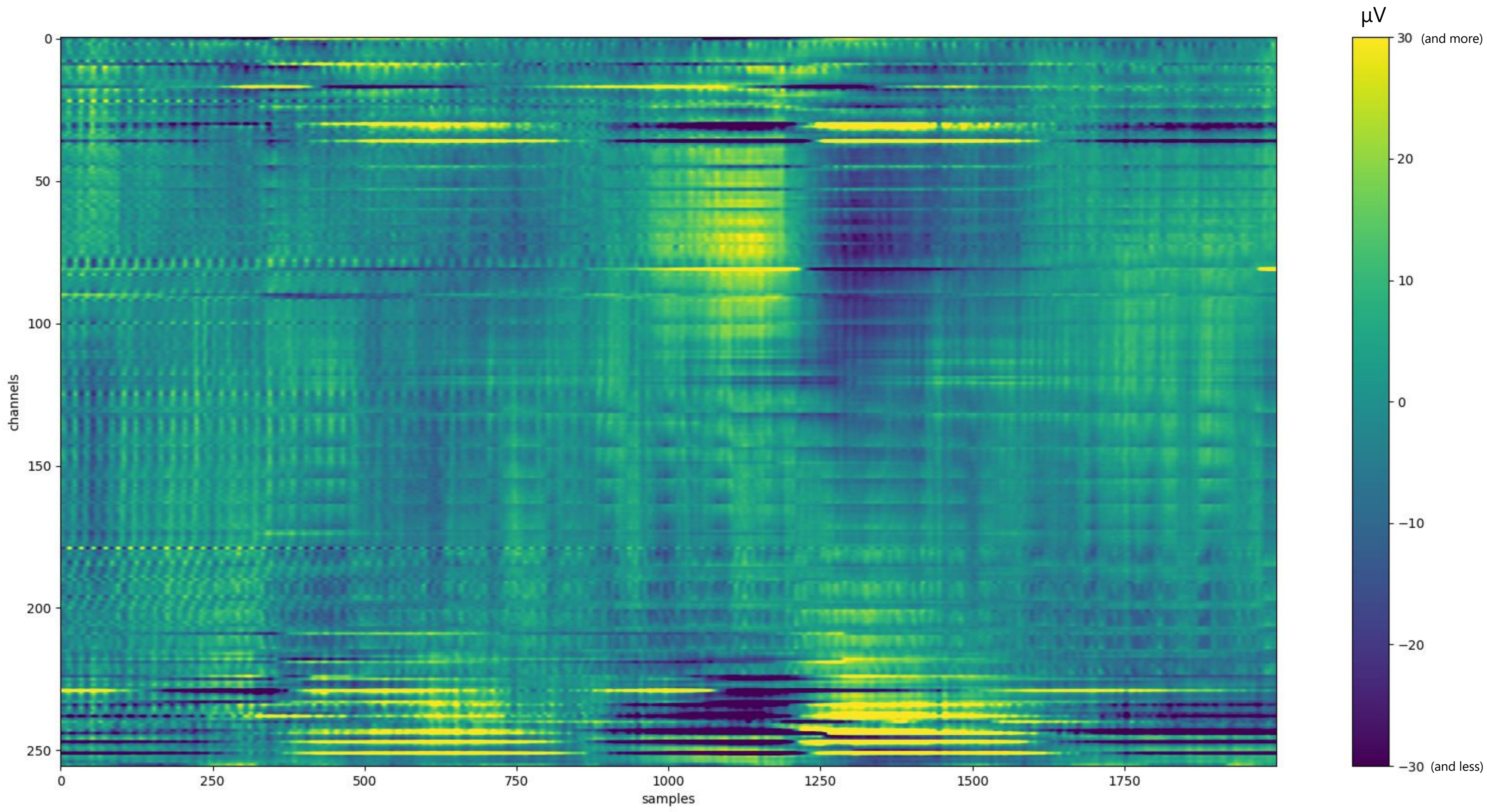
$$25 \cdot 3.98 \text{ GB} \approx \mathbf{99.5 \text{ GB of .NPY arrays (float64)}}$$

- Metadata is separated from the array
- e. g. array for the 1^{st} subject has a size of **260 × 2 057 837**

2.3. EEG Signal



2.3. EEG Signal (as an image)



3. Data Preprocessing

- Data has **specific behavior** that needs to be addressed
- In raw form, it is **too unwieldy**
- Preparing datasets for the future **ANN training**

CLEANUP

STATISTICS

SPLITS

TENSOR DATASETS

DATA TRANSFORM

3.1. Cleanup

1. **Removal of muscle artifacts etc.** (not by me)
2. **Removal of 4 extra channels** (not actually EEG)
3. **Clipping the zeroed segments** (not actually EEG)

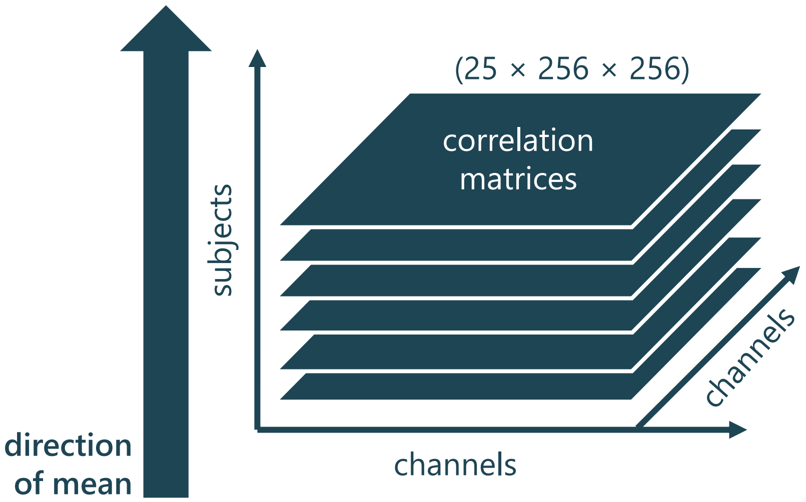
CLEANUP › STATISTICS › SPLITS › TENSOR DATASETS › DATA TRANSFORM

3.2. Statistics

1. **Correlation between channels** – is it reasonable to compress?
2. **Autocorrelation of the samples** – size of splits (explained later)

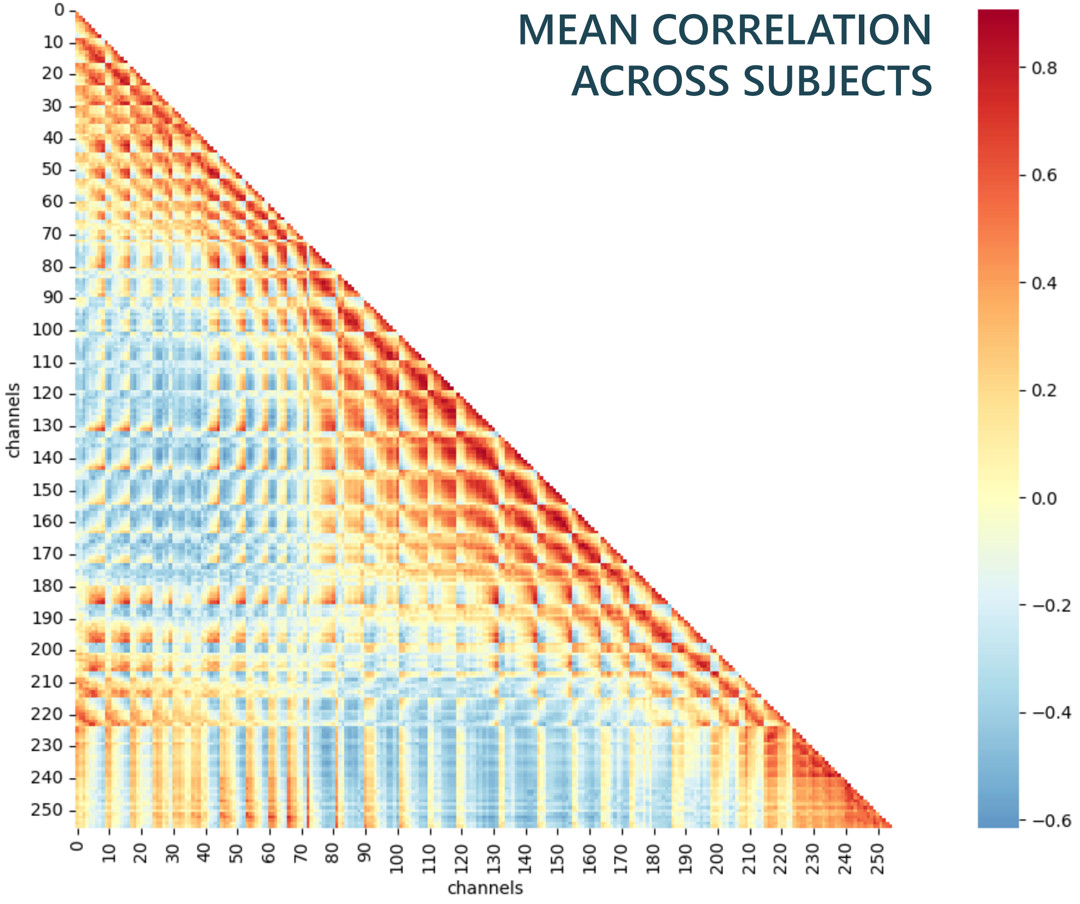
3.2.1. Correlation

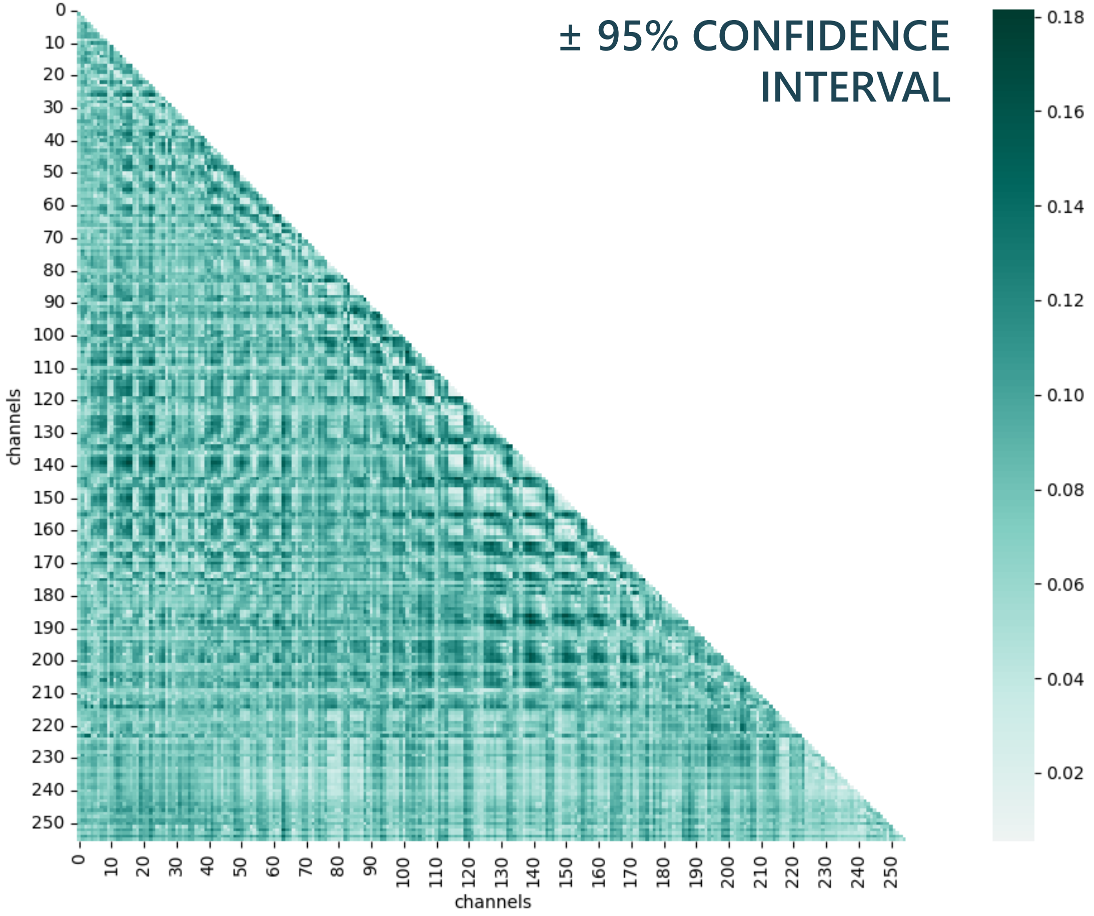
- **High/low correlation** ⟹ redundant information ⟹ better compression
- **Electrodes are close** ⟹ we expect them to correlate
- Similar across subjects?



CLEANUP | **STATISTICS** | SPLITS | TENSOR DATASETS | DATA TRANSFORM

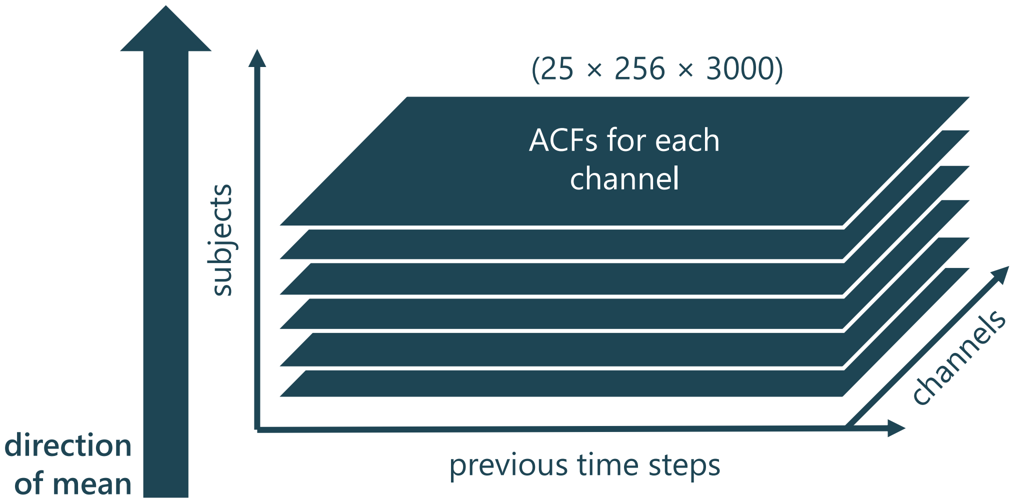
3.2.1. Correlation





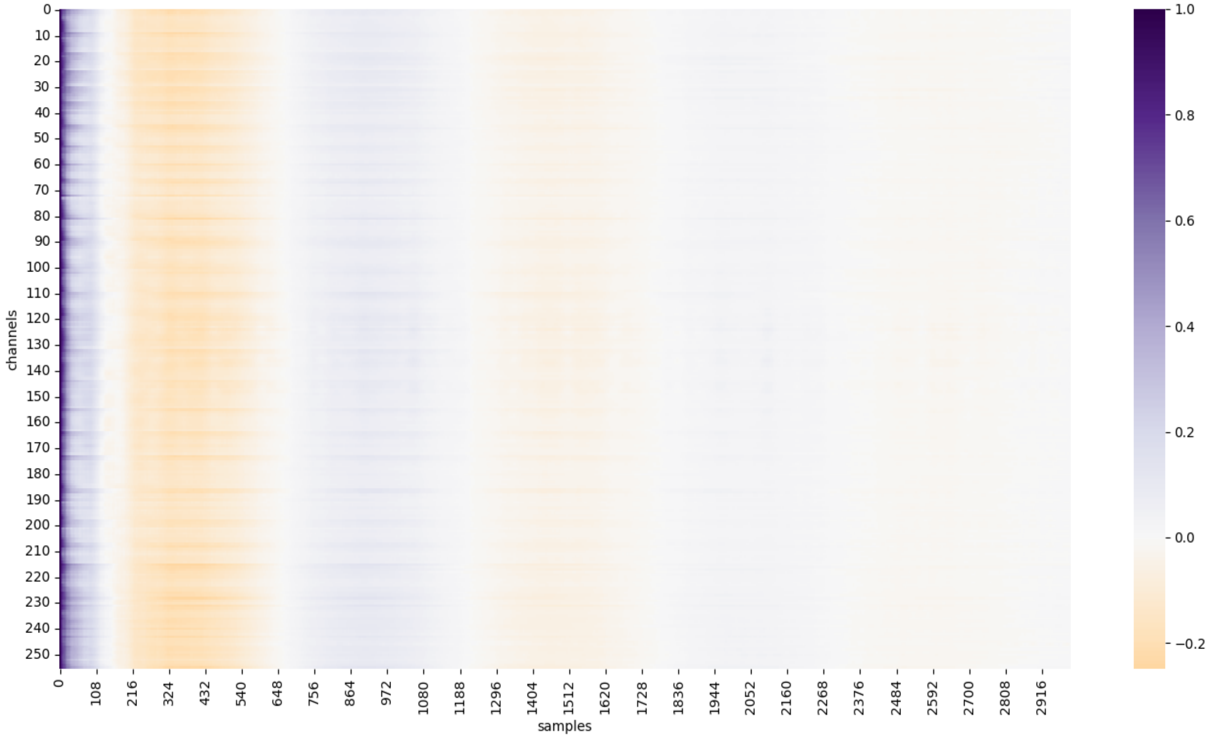
3.2.2. Autocorrelation

- **Impossible (and undesirable) to load the entire matrices at once**
- **Somehow determine the optimal split size** ⟹ ACF
- Similar across subjects?

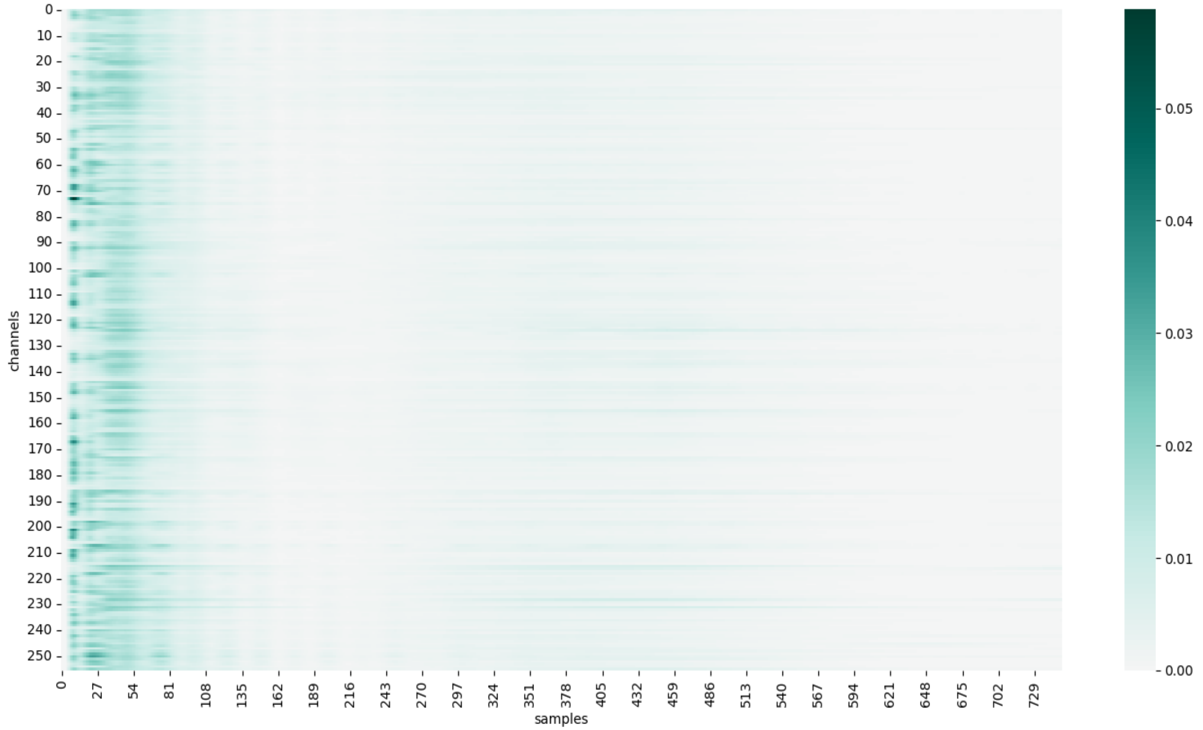


3.2.2. Autocorrelation

MEAN AUTOCORRELATION ACROSS SUBJECTS

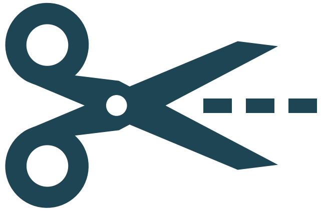


± 95% CONFIDENCE INTERVAL



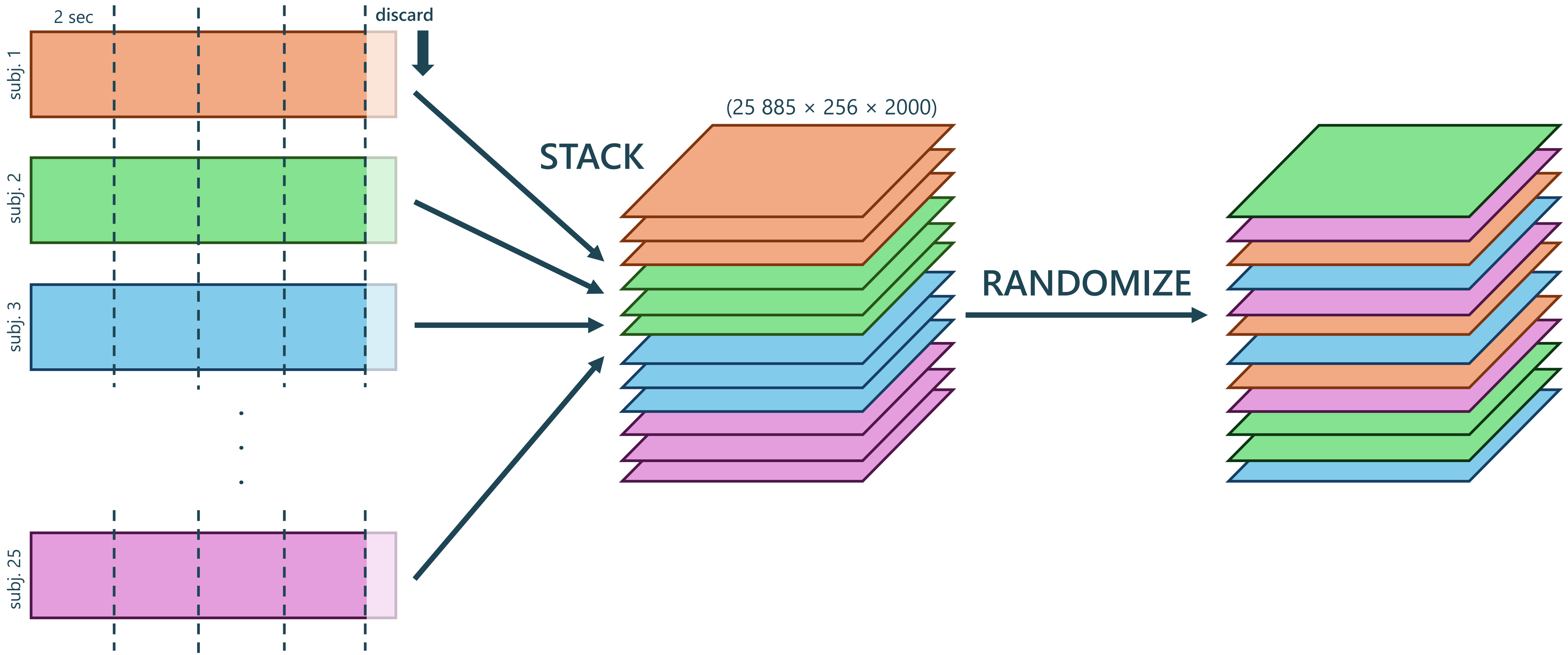
3.3. Splits

- Using the ACF, we determine split length as **2000 samples (2 sec)**
- **Must be uniform** $\Longrightarrow$ the “tail” gets discarded (OK here)
- Overall, we get **25 885 EEG chunks**



CLEANUP → STATISTICS → **SPLITS** → TENSOR DATASETS → DATA TRANSFORM

3.3. Splits



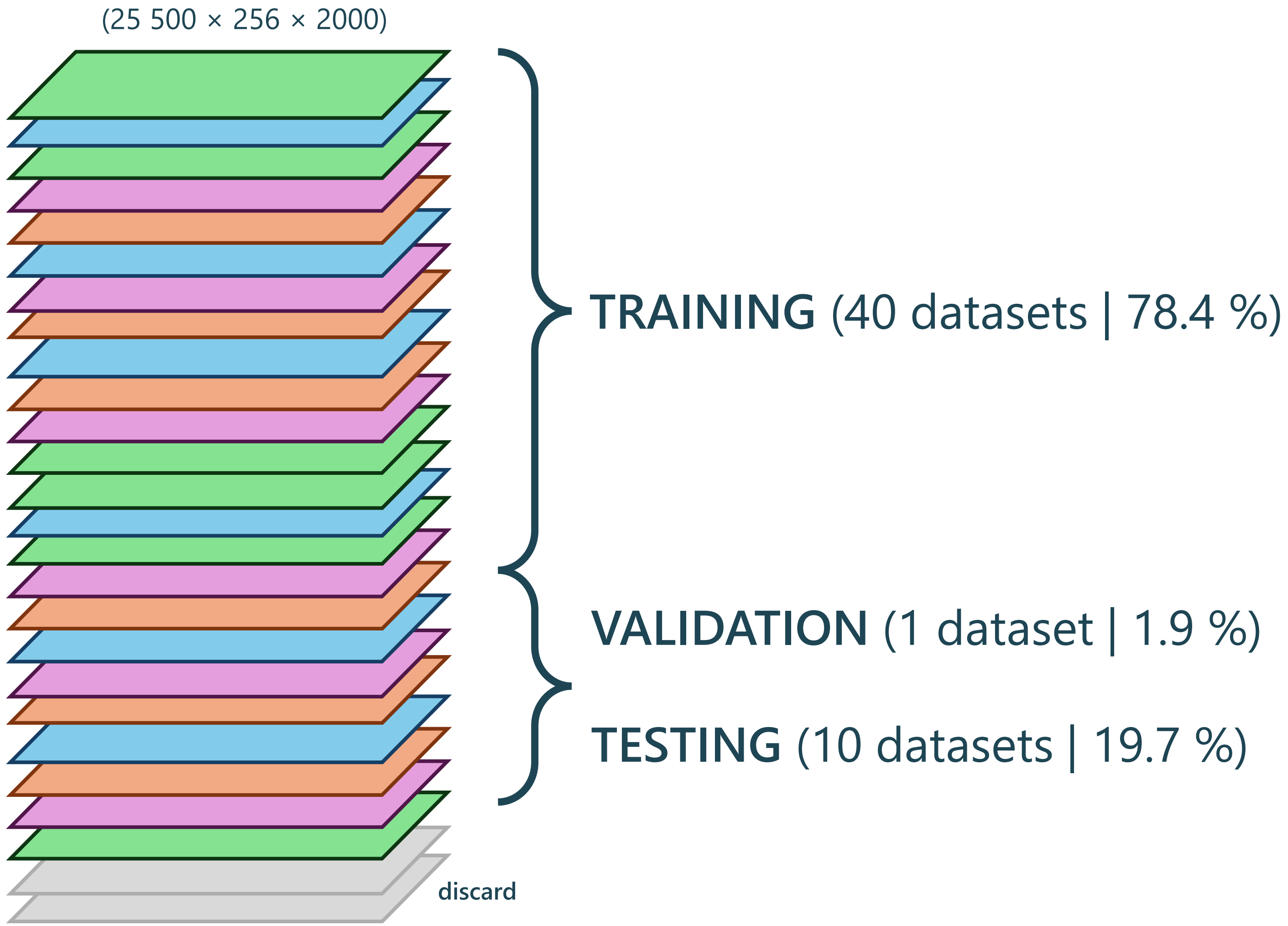
2 sec
discard
subj. 1
subj. 2
subj. 3
subj. 25
STACK
(25 885 × 256 × 2000)
RANDOMIZE

3.3. Tensor Datasets

- We need to make **training, validation, and testing datasets**
- Desired ratio is about **80 % for training, 20 % for validation/testing**
- Cannot fit massive datasets into memory $\Longrightarrow$ **multiple datasets**
- We want to evaluate learning across datasets $\Longrightarrow$ **same size**

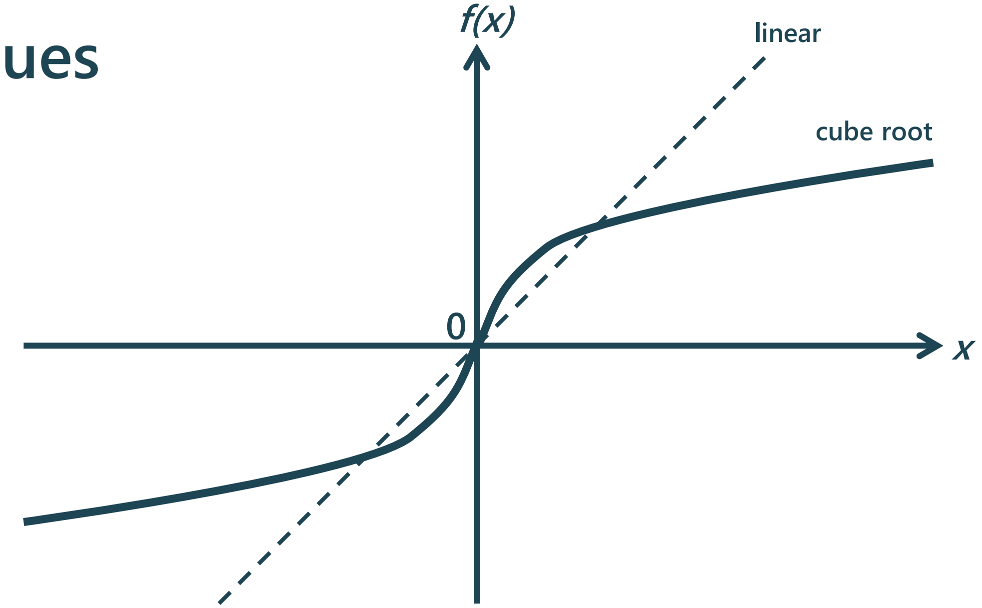
$(500 \times 256 \times 2\,000)$

3.3. Tensor Datasets



3.4. Data Transform

- **Extreme anomalies** ⟹ extreme errors when training (esp. with MSE)
- The majority of data has **relatively small values**
- Both **positive and negative values**
- **Solution** – cube root



4. Artificial Neural Network

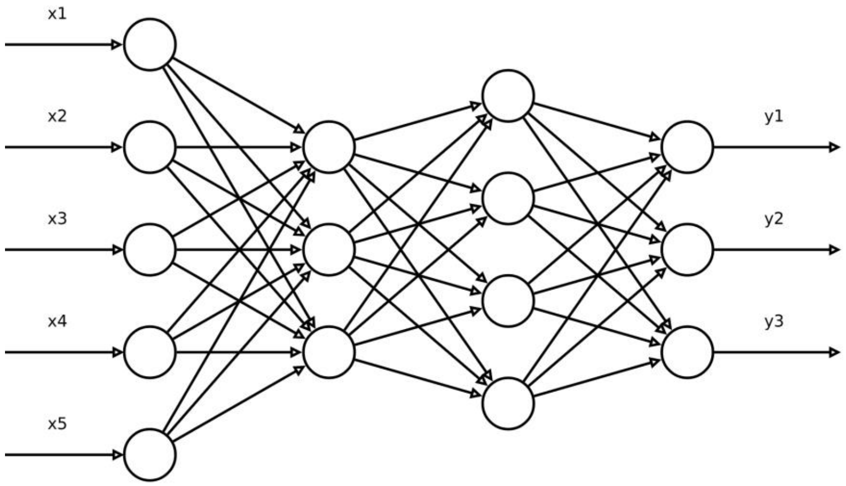
- Reduction of images ⟹ **Convolutional Autoencoder (CNN-AE)**
- **Unsupervised learning, regression**
- **Deterministic behavior** ⟹ sender knows how the data will be reconstructed by the receiver

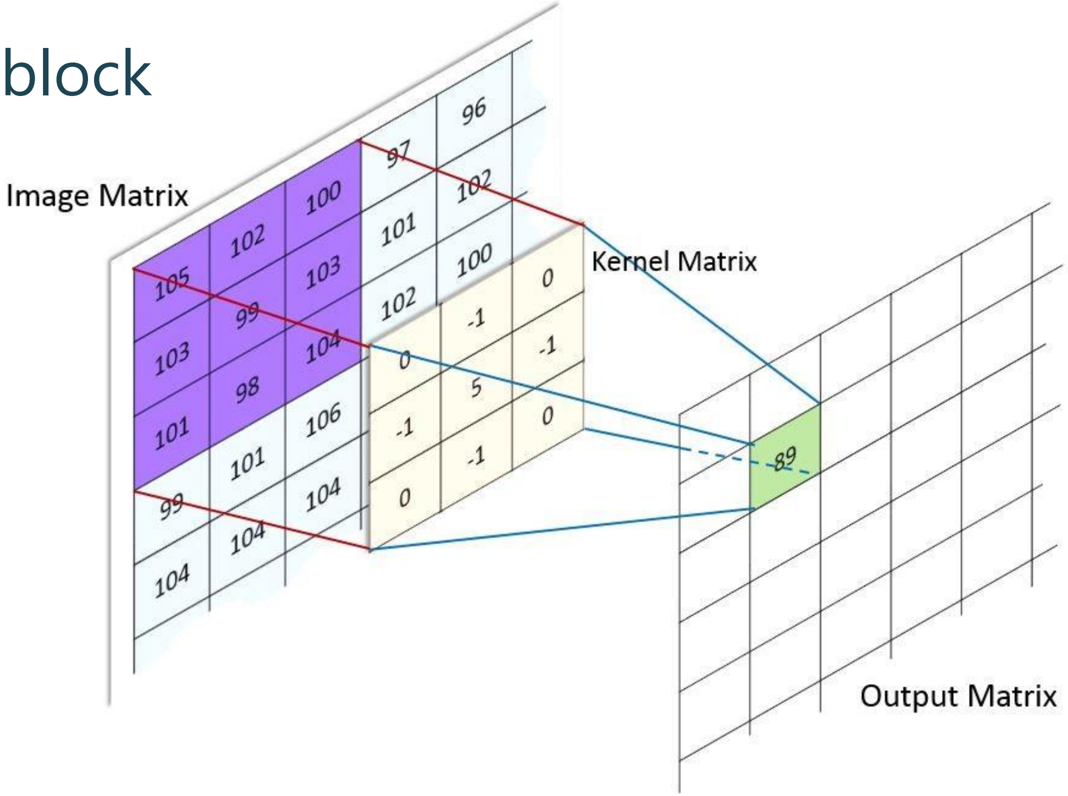
ARCHITECTURE → TRAINING → TRAINED MODEL

4.1. Architecture

1. **Convolutional layers** – main building block

2. **Linear layers** – in the bottleneck

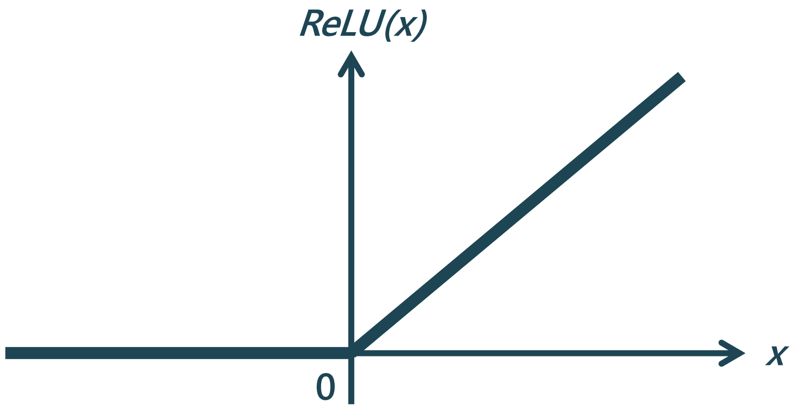


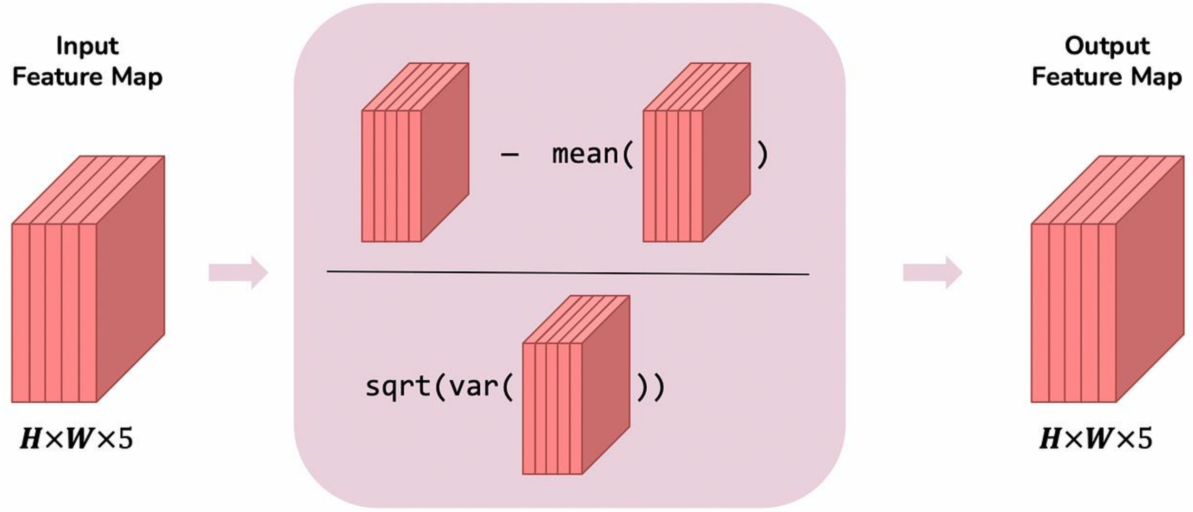


4.1. Architecture

3. **Batch Norm** – used to keep the learning stable

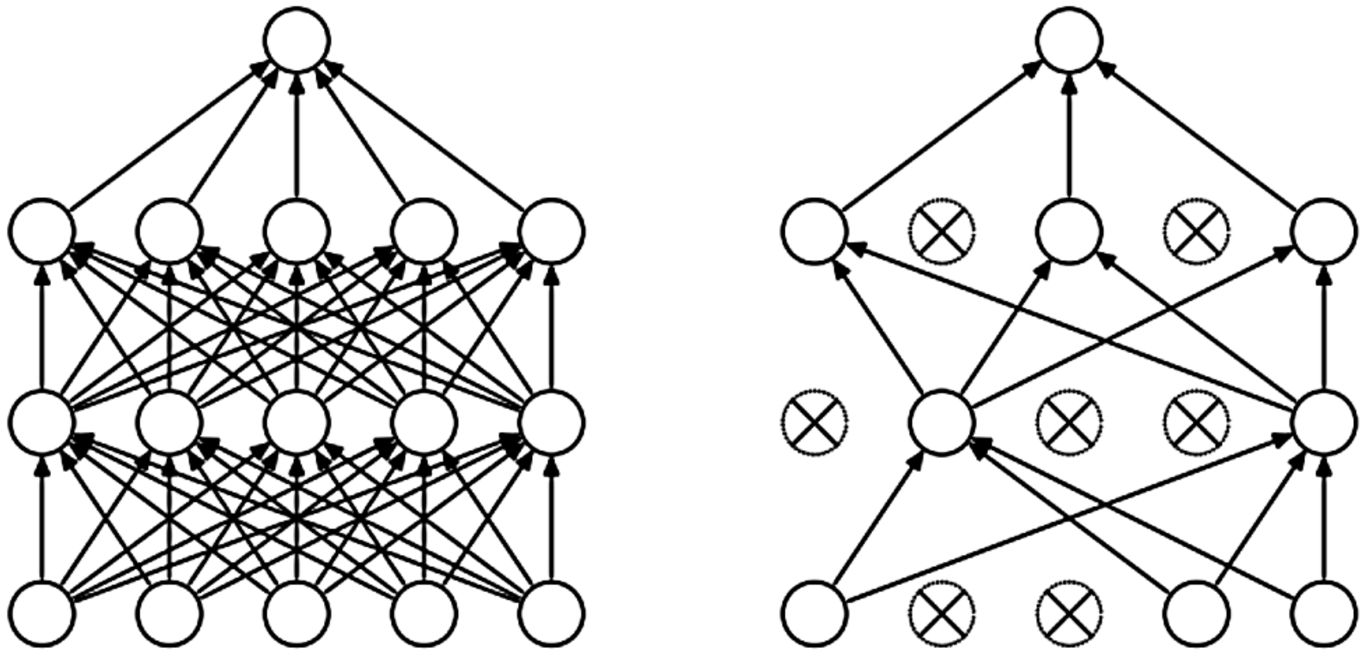
4. **ReLU** – activation function





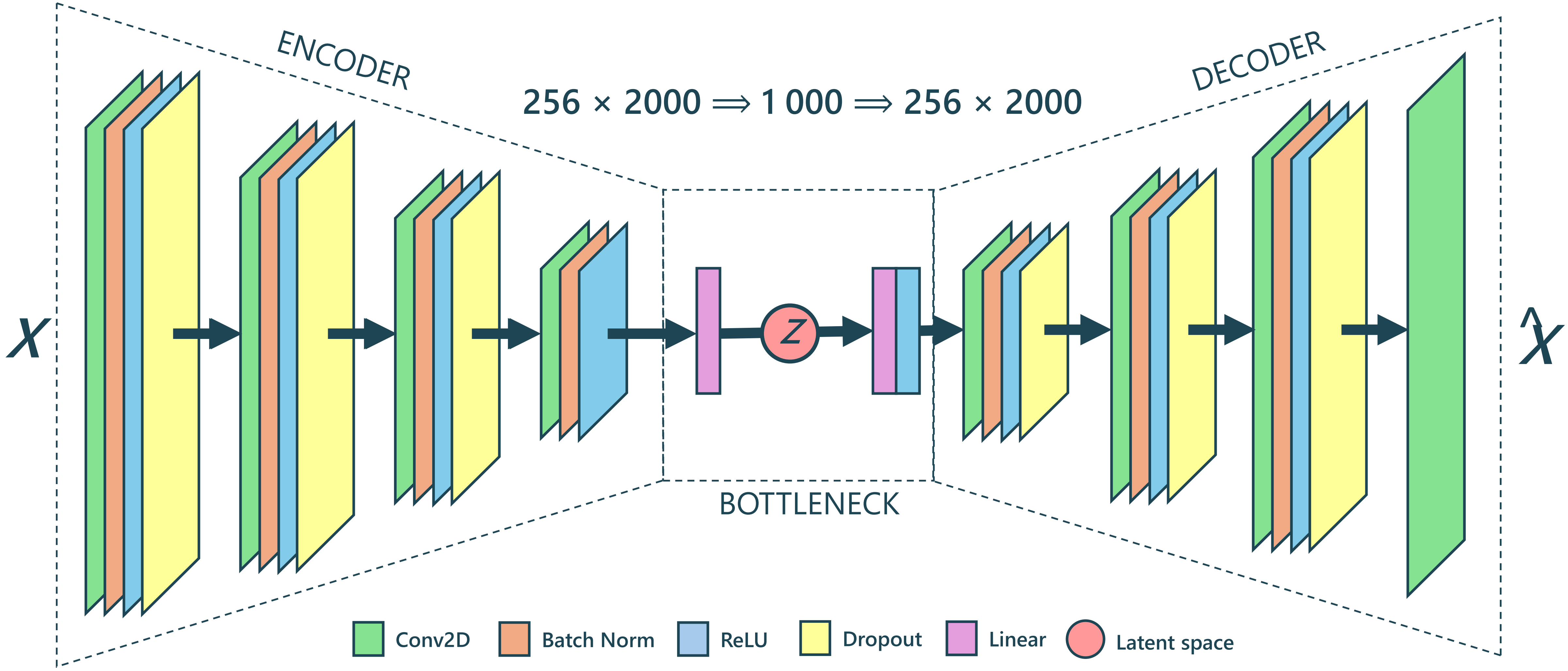
4.1. Architecture

5. **Dropout (25 %)** – to avoid overfitting



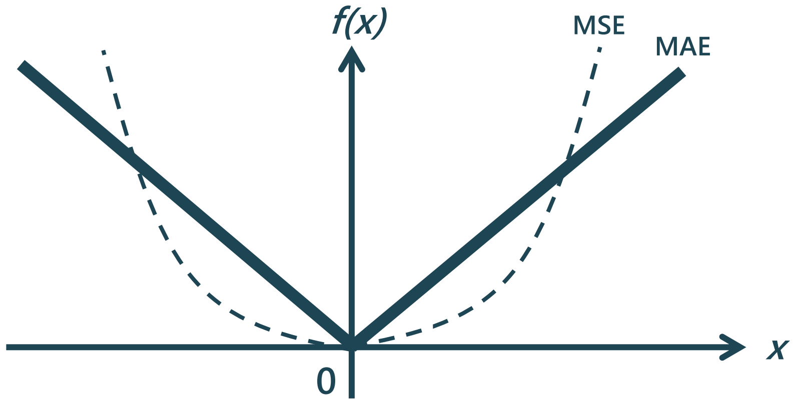
ARCHITECTURE | TRAINING | TRAINED MODEL

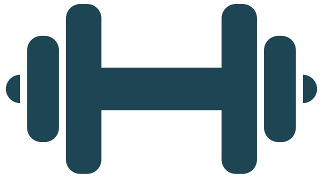
4.1. Architecture



4.2. Training

1. **Batch size** – 10 (50 batches per dataset)
2. **Epochs** – 20, each comprised of **2 000 parameter updates** (50 · 40)
3. **Loss function** – MAE, because of small numbers (cube root)

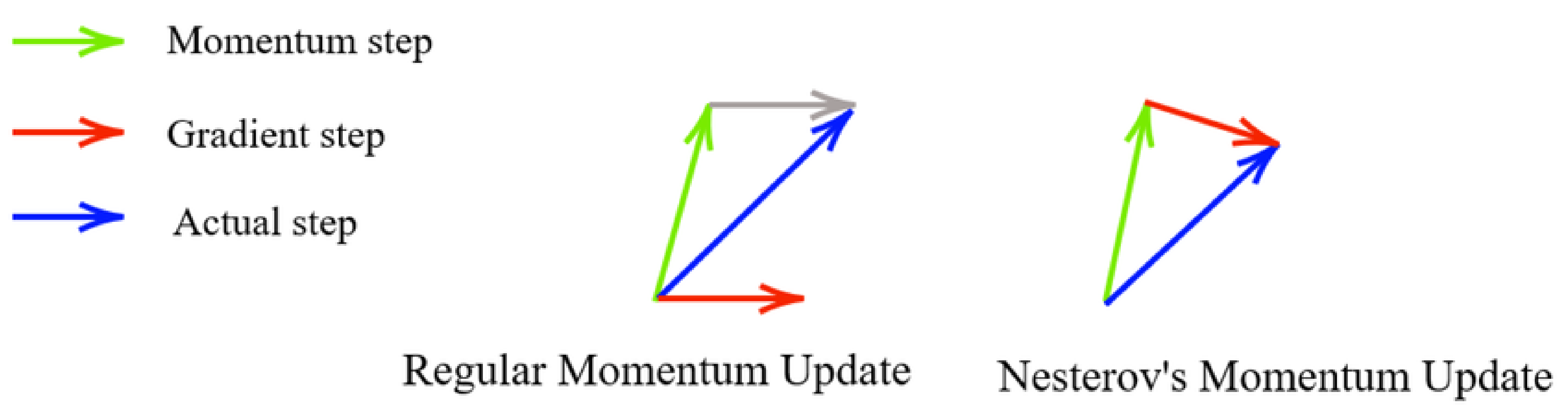




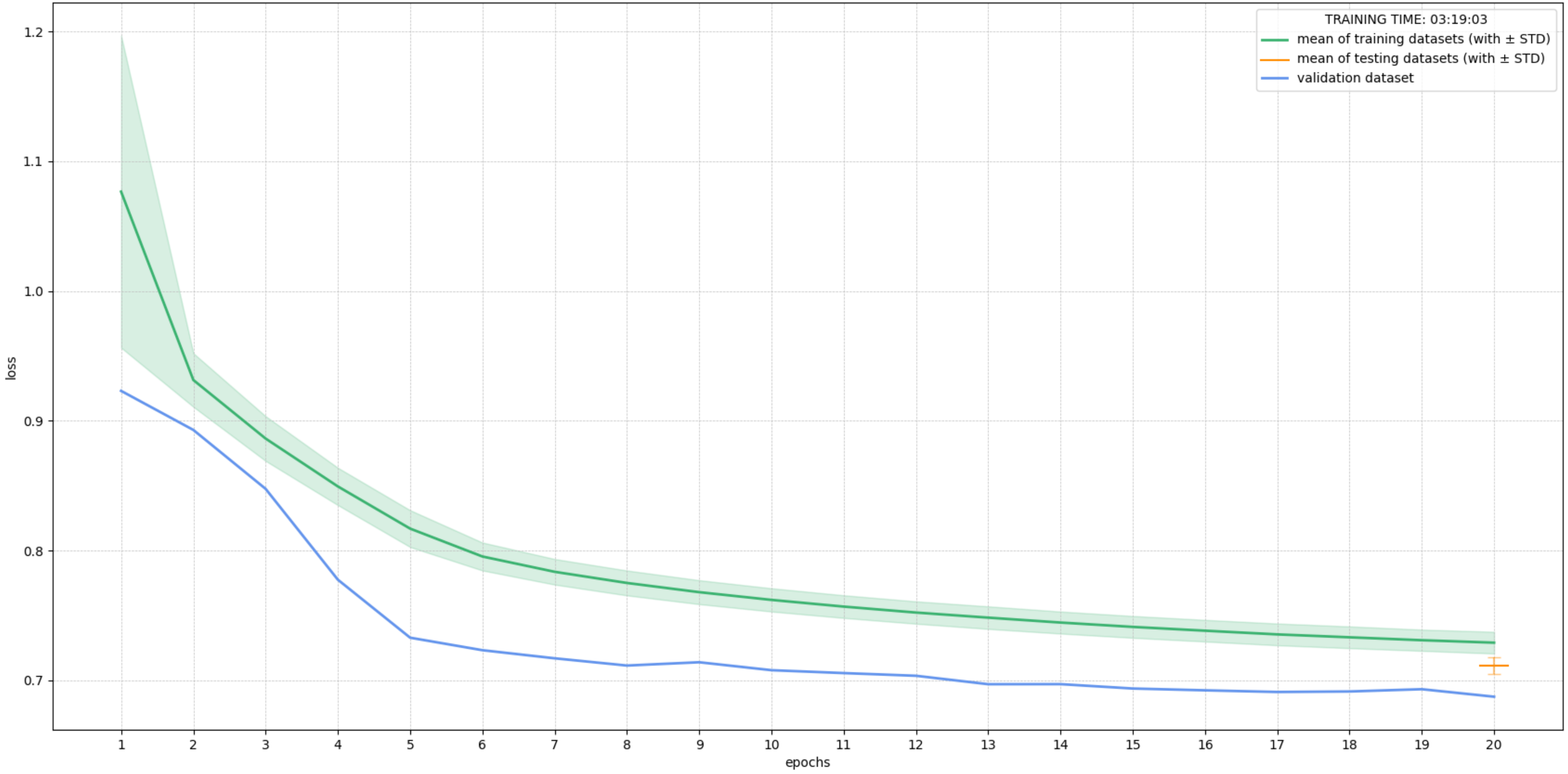
ARCHITECTURE → **TRAINING** → TRAINED MODEL

4.2. Training

4. **Optimizer** – NAdam (LR = 0.001)



4.2. Training


TRAINING TIME: 03:19:03
mean of training datasets (with ± STD)
mean of testing datasets (with ± STD)
validation dataset
loss
epochs
1.2
1.1
1.0
0.9
0.8
0.7
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

4.3. Trained Model

- Total **training time 3:18:03**
- Parameters saved as a **488 MB .PT file**
- For the model to be used, it **requires both CNN-AE code and parameters**
- **Expects 256-channel EEG as input**
- **Usable on other EEG data or similar ERP experiments?** (lack of data)

ARCHITECTURE | TRAINING | TRAINED MODEL

5. Compression Mechanism

- Combination of lossy and lossless compression
- **Lossy** – reconstruction by the CNN-AE
- **Lossless** – original values for pixels with error higher than threshold

5.1. Sender

1. **Passes the data through CNN-AE** $\Longrightarrow$ gets both Z and $\hat{X}$
2. **A threshold is chosen** (5 µV in the following images)
3. Based on the reconstruction error $AE(X, \hat{X})$, **coordinates and differences get saved** (if they exceed the threshold)
4. Both Z **and value corrections get send to the receiver**

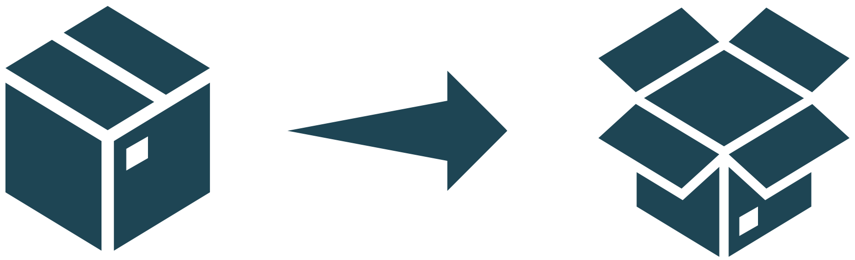
SENDER RECEIVER RESULTS

5.2. Receiver

1. **Passes Z through the decoder** $\Longrightarrow$ gets $\hat{X}$
2. **Updates values** based on the corrections

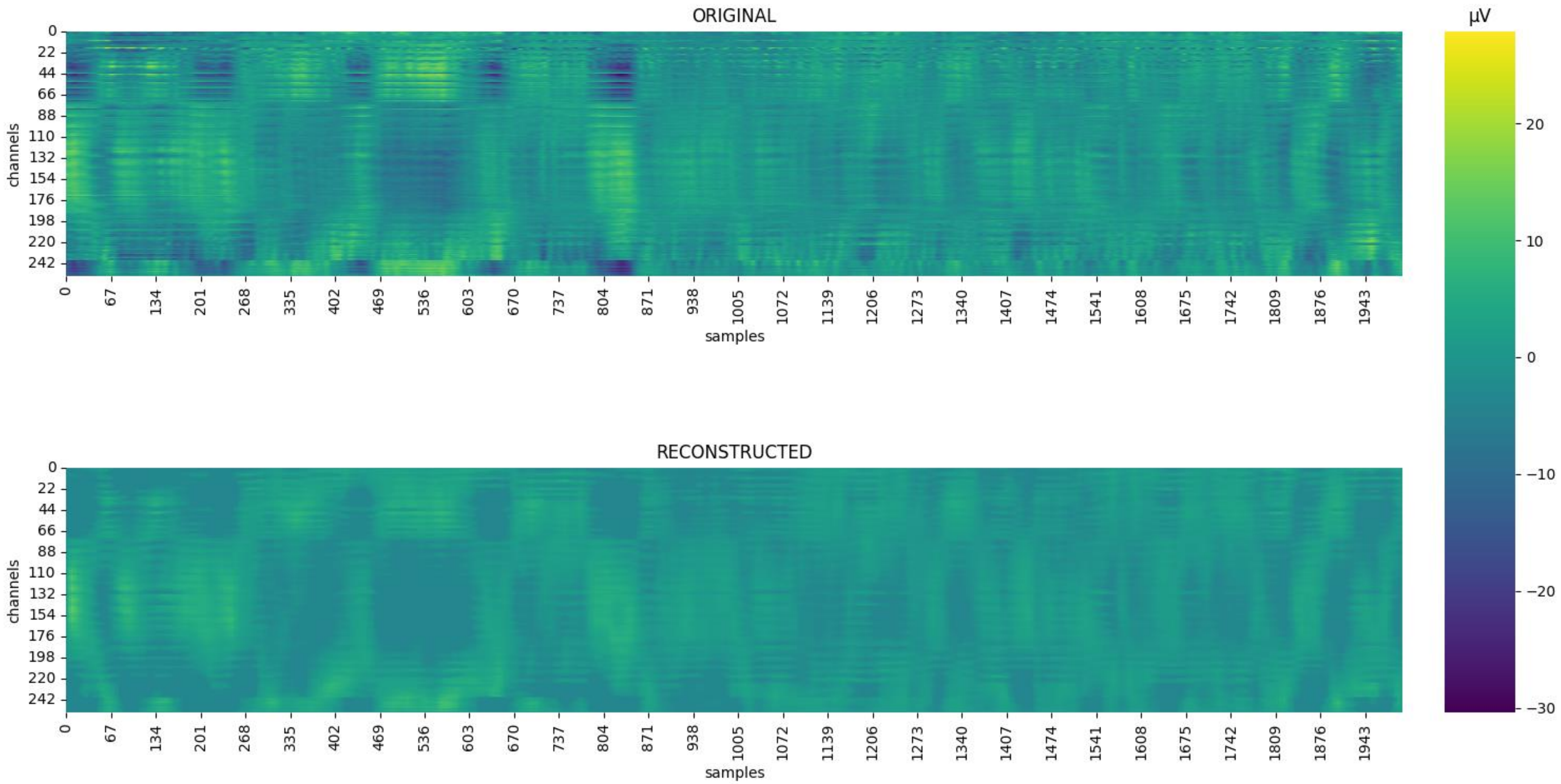
5.3. Results

- Tested on one testing dataset $\Longrightarrow$ **1.9 GB (float64)**
- Folder with compressed data $\Longrightarrow$ **481 MB (float32)** $\Longrightarrow$ **25.3 %**
- Adding GZIP on top of that $\Longrightarrow$ **106 MB** $\Longrightarrow$ **5.6 %**
- **Compression time 18.5 sec, decompression time 3.2 min**



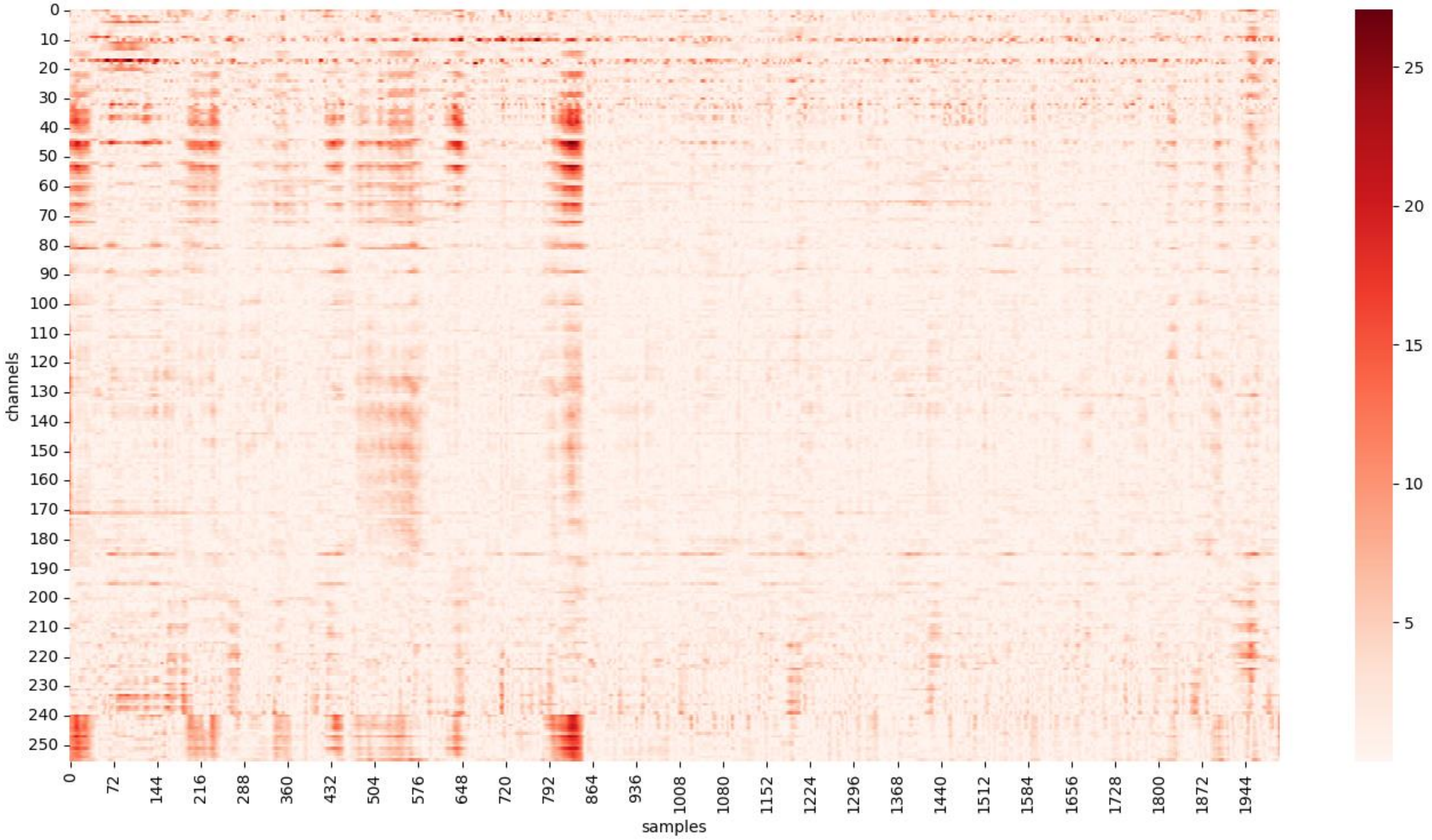
SENDER
RECEIVER
RESULTS

5.3. Results (1)



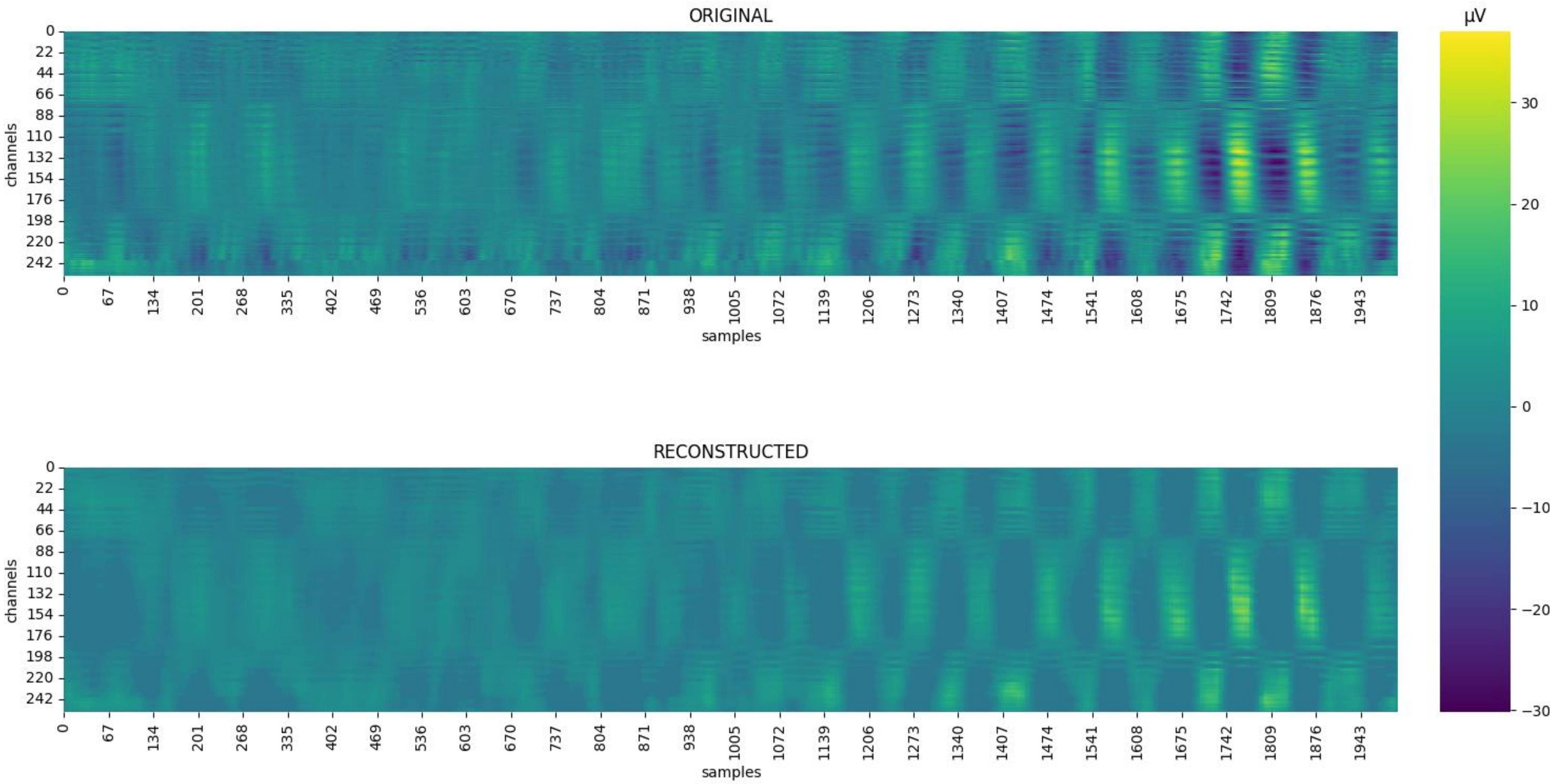
ORIGINAL
RECONSTRUCTED
channels
samples
µV
0
22
44
66
88
110
132
154
176
198
220
242
0
67
134
201
268
335
402
469
536
603
670
737
804
871
938
1005
1072
1139
1206
1273
1340
1407
1474
1541
1608
1675
1742
1809
1876
1943
20
10
0
−10
−20
−30

5.3. Results (1)



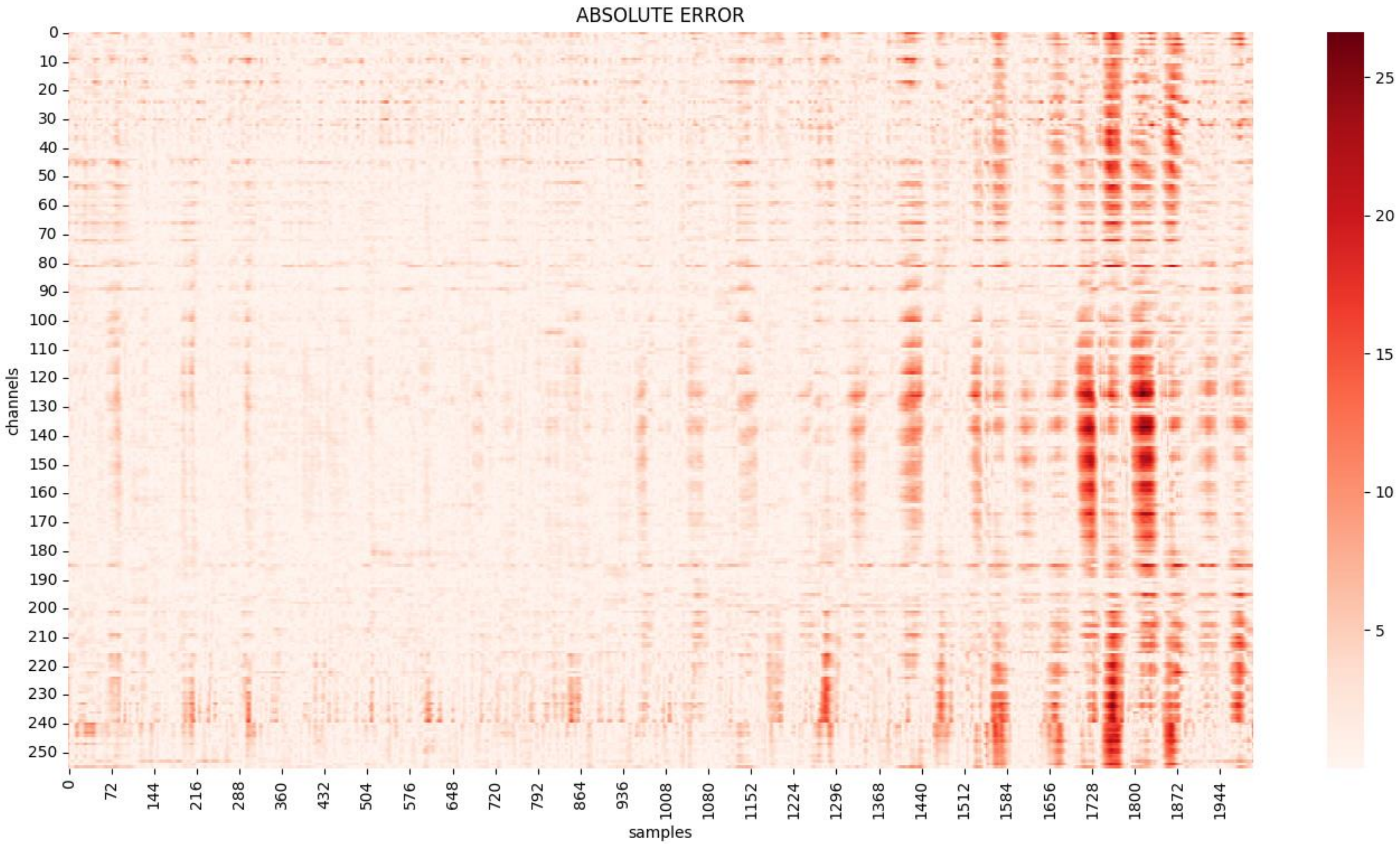
channels
samples
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
0
72
144
216
288
360
432
504
576
648
720
792
864
936
1008
1080
1152
1224
1296
1368
1440
1512
1584
1656
1728
1800
1872
1944
25
20
15
10
5

5.3. Results (2)

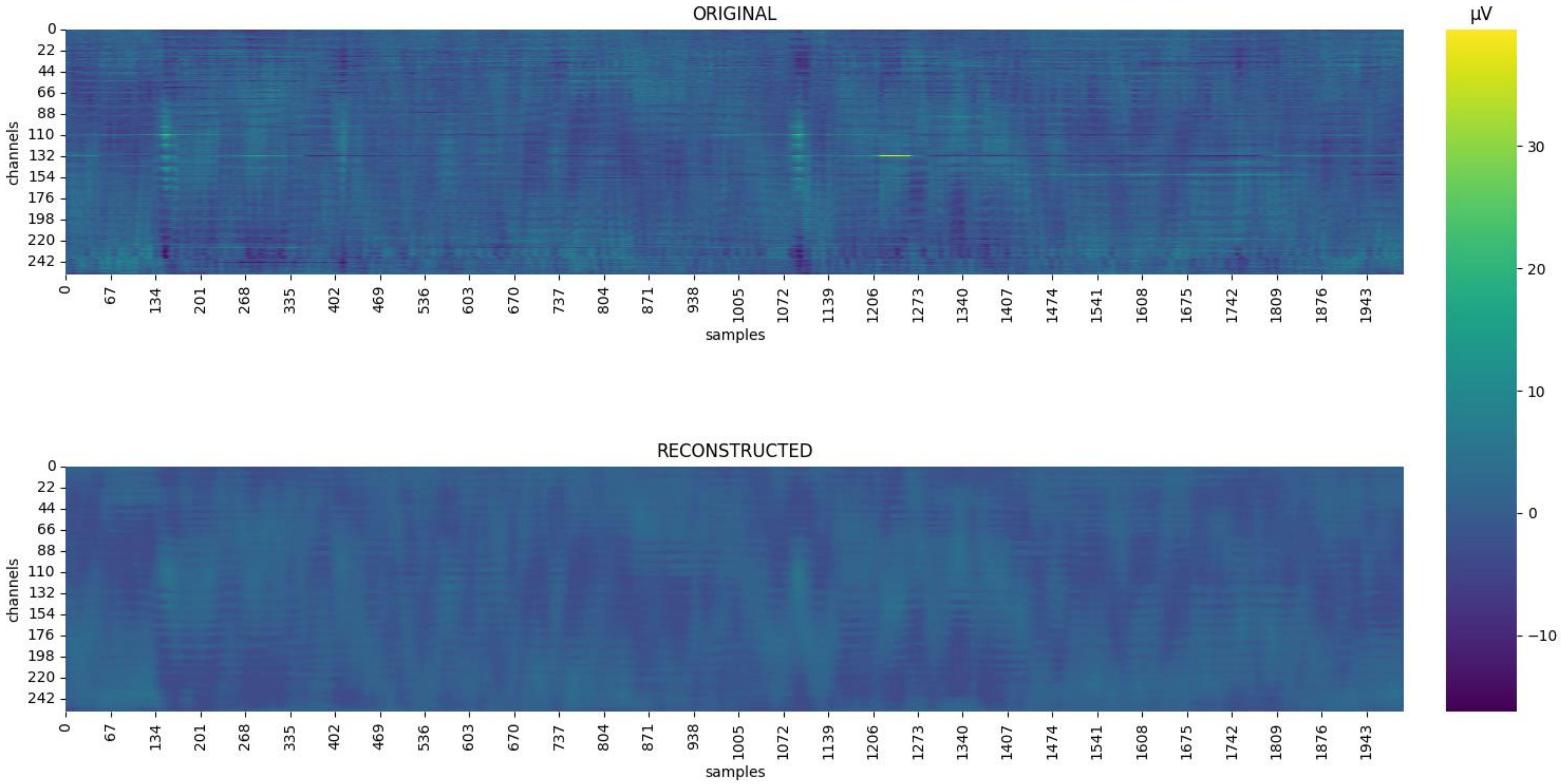


ORIGINAL
RECONSTRUCTED
channels
samples
µV
30
20
10
0
−10
−20
−30
0
22
44
66
88
110
132
154
176
198
220
242
0
67
134
201
268
335
402
469
536
603
670
737
804
871
938
1005
1072
1139
1206
1273
1340
1407
1474
1541
1608
1675
1742
1809
1876
1943

5.3. Results (2)

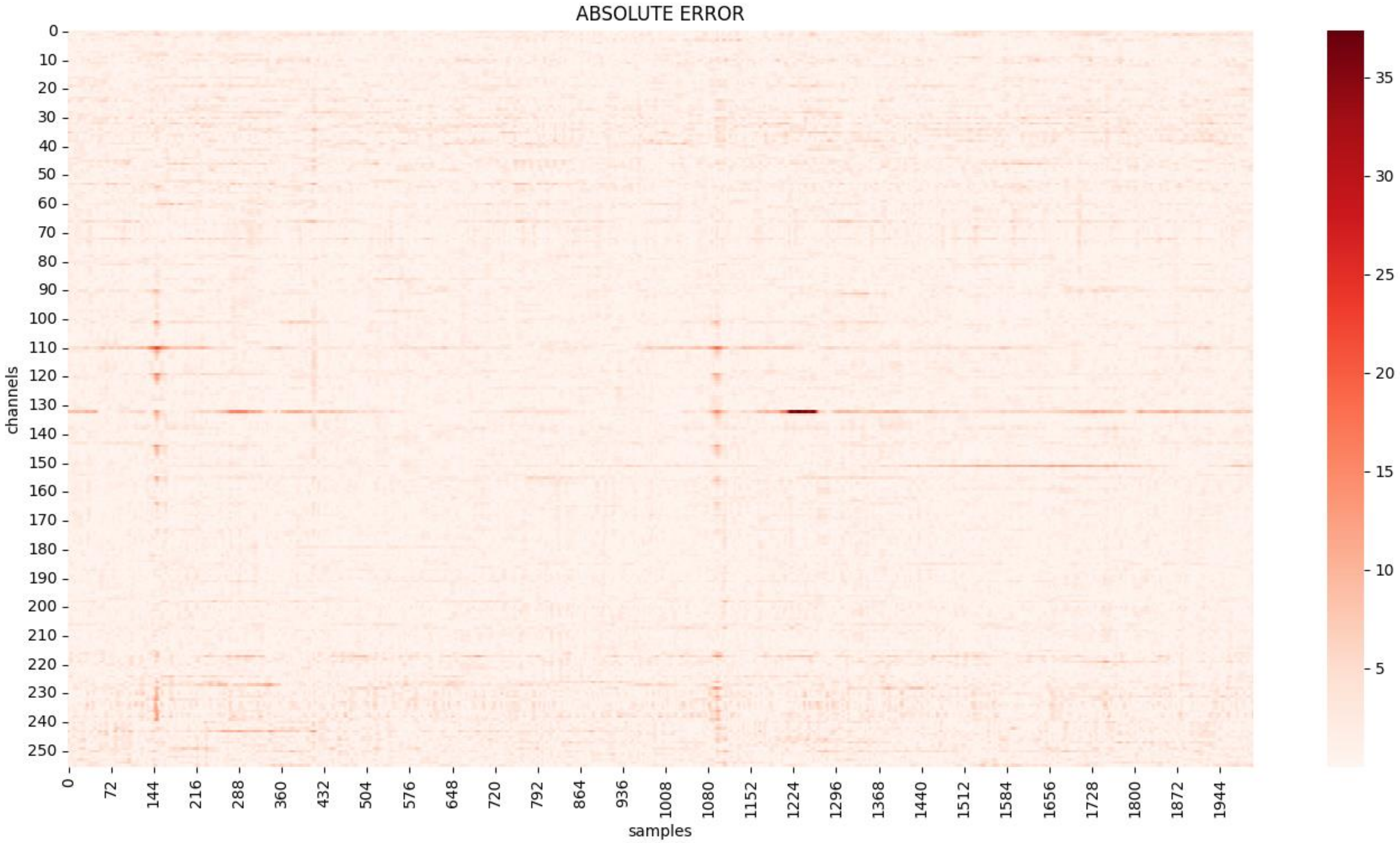

ABSOLUTE ERROR
channels
samples
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
0
72
144
216
288
360
432
504
576
648
720
792
864
936
1008
1080
1152
1224
1296
1368
1440
1512
1584
1656
1728
1800
1872
1944
25
20
15
10
5

5.3. Results (3)



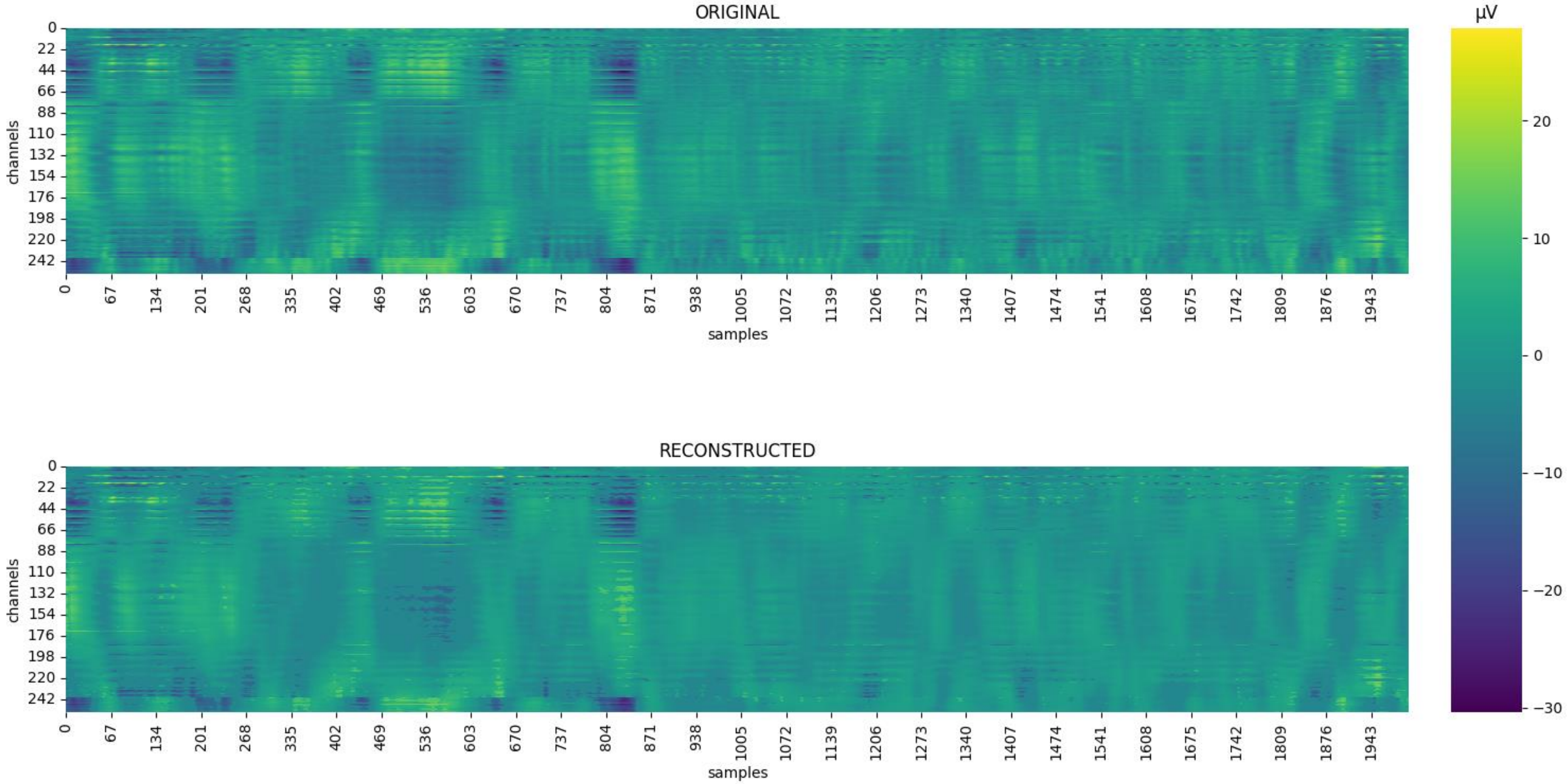
ORIGINAL
RECONSTRUCTED
channels
samples
µV
0
22
44
66
88
110
132
154
176
198
220
242
0
67
134
201
268
335
402
469
536
603
670
737
804
871
938
1005
1072
1139
1206
1273
1340
1407
1474
1541
1608
1675
1742
1809
1876
1943
30
20
10
0
−10

5.3. Results (3)

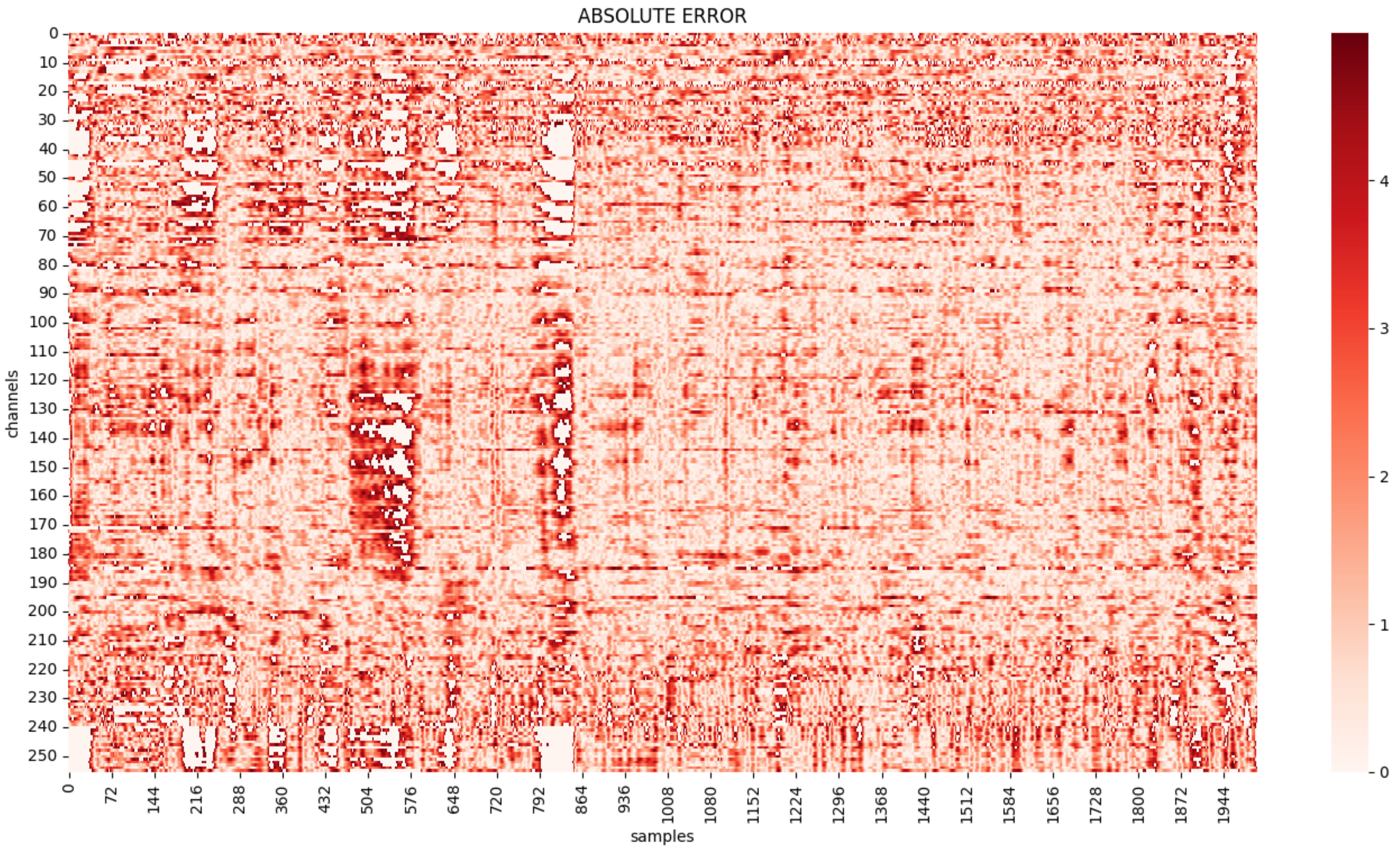


ABSOLUTE ERROR
channels
samples
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
0
72
144
216
288
360
432
504
576
648
720
792
864
936
1008
1080
1152
1224
1296
1368
1440
1512
1584
1656
1728
1800
1872
1944
35
30
25
20
15
10
5

5.3. Results (decompression)



5.3. Results (decompression)



6. Future prospects

1. **Optimize used data types**
2. **Smarter representation of the corrections**
3. **Lossless compression?** (threshold set to 0)
4. **Test model performance on the data from different EEG/ERP experiments?**

7. Conclusion

- **Method** – the better the ANN (i.e. knowledge representation), the better the compression
- **Positives** – high compression rate, possible follow up research (e.g. latent space analysis), tailored to specific data, unsupervised learning
- **Negatives** – limited use, uncertain generalization, not fully lossless (currently), computational complexity, large datasets

Thank You For Your Attention!

Martin KUKRÁL

kukrma@students.zcu.cz

FACULTY OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA