Supplementary materials

CSLP-AE: A Contrastive Split-Latent Permutation Autoencoder Framework for Zero-Shot Electroencephalography Signal Conversion

Anders Vestergaard Nørskov **Alexander Neergaard Zahid** **Morten Mørup**
Department of Applied Mathematics and Computer Science
Technical University of Denmark
andersxa@gmail.com {aneol,mmor}@dtu.dk
https://github.com/andersxa/CSLP-AE

Contents

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

A Structural encoding and generalization of split-latent permutation loss

We observed a tendency in the SLP-AE model trained using only the split-latent permutation loss, in which the model would simply learn identical latent spaces. We discussed how this tendency stems from the information-propagation through one of the latents during training. Split-latent permutation is trained using the self-reconstruction loss where one of the latents given as input to the decoder is swapped with that of another sample where both samples either belong to the same task or the same subject. Given that only one latent is swapped at a time, the model can learn a permutation-invariant encoding of the signal not specific to either the subject or task content of the signal, and it could rely on this information during (S.s., S.t.), (D.s., S.t.), (S.s., D.t.) conversion.

In this material we generalize the split-latent permutation using a quadruplet sampling method instead to instances where the input sample and the reconstruction target (output sample) is not the same, but which in special cases becomes identical to both the split-latent permutation and the self-reconstruction loss.

During training we sample a batch of K quadruplets, $\{(X_k^{(A)}, X_k^{(B)}, X_k^{(C)}, X_k^{(D)})\}_{k=1}^{K}$. For each k we choose two random subjects, U_k and V_k, and two random tasks, M_k and N_k. The quadruplet samples are sampled such that

$$X_k^{(A)} \text{ has subject and task } (U_k, M_k) \tag{1}$$
$$X_k^{(B)} \text{ has subject and task } (V_k, M_k) \tag{2}$$
$$X_k^{(C)} \text{ has subject and task } (U_k, N_k) \tag{3}$$
$$X_k^{(D)} \text{ has subject and task } (V_k, N_k) \tag{4}$$

Similar to the split-latent permutation, the encoding of these quadruplets should match and disentangle the subject and task content into their respective latent spaces, such that a latent-swap between two latents (which ideally encode the same information) has minimal impact on the reconstruction/conversion. With these quadruplets we can now generalize the split-latent permutation such that both latents are swapped with latents from other samples which should encode the same information. The samples encode the following latents

$$X_k^{(A)} \text{ encodes } (\boldsymbol{z}_k^{(\mathcal{S},a)}, \boldsymbol{z}_k^{(\mathcal{T},a)}) \tag{5}$$
$$X_k^{(B)} \text{ encodes } (\boldsymbol{z}_k^{(\mathcal{S},b)}, \boldsymbol{z}_k^{(\mathcal{T},b)}) \tag{6}$$
$$X_k^{(C)} \text{ encodes } (\boldsymbol{z}_k^{(\mathcal{S},c)}, \boldsymbol{z}_k^{(\mathcal{T},c)}) \tag{7}$$
$$X_k^{(D)} \text{ encodes } (\boldsymbol{z}_k^{(\mathcal{S},d)}, \boldsymbol{z}_k^{(\mathcal{T},d)}) \tag{8}$$

where

$$\boldsymbol{z}_k^{(\mathcal{S},a)} \text{ and } \boldsymbol{z}_k^{(\mathcal{S},c)} \text{ both ideally encode } U_k \tag{9}$$
$$\boldsymbol{z}_k^{(\mathcal{S},b)} \text{ and } \boldsymbol{z}_k^{(\mathcal{S},d)} \text{ both ideally encode } V_k \tag{10}$$
$$\boldsymbol{z}_k^{(\mathcal{T},a)} \text{ and } \boldsymbol{z}_k^{(\mathcal{T},b)} \text{ both ideally encode } M_k \tag{11}$$
$$\boldsymbol{z}_k^{(\mathcal{T},c)} \text{ and } \boldsymbol{z}_k^{(\mathcal{T},d)} \text{ both ideally encode } N_k \tag{12}$$

All of these pairs of latents which ideally encode the same information are swapped in the generalized split-latent permutation loss, which we will refer to as the *quadruplet permutation loss* (QP-loss). With this full swap of latents, there is no direct path between the input sample and the output sample, and the model is forced to encode the subject and task content into their respective latents. The reconstructions are as follows

$$\hat{X}_k^{(A)} = D_\phi(\boldsymbol{z}_k^{(\mathcal{S},c)}, \boldsymbol{z}_k^{(\mathcal{T},b)}) \text{ should reconstruct } X_k^{(A)} \tag{13}$$
$$\hat{X}_k^{(B)} = D_\phi(\boldsymbol{z}_k^{(\mathcal{S},d)}, \boldsymbol{z}_k^{(\mathcal{T},a)}) \text{ should reconstruct } X_k^{(B)} \tag{14}$$
$$\hat{X}_k^{(C)} = D_\phi(\boldsymbol{z}_k^{(\mathcal{S},a)}, \boldsymbol{z}_k^{(\mathcal{T},d)}) \text{ should reconstruct } X_k^{(C)} \tag{15}$$
$$\hat{X}_k^{(D)} = D_\phi(\boldsymbol{z}_k^{(\mathcal{S},b)}, \boldsymbol{z}_k^{(\mathcal{T},c)}) \text{ should reconstruct } X_k^{(D)} \tag{16}$$

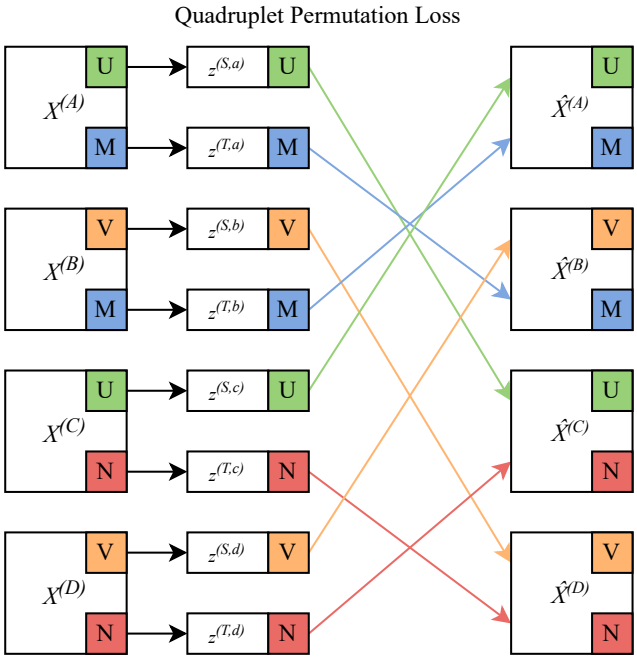


Figure 1: Illustration of the quadruplet permutation loss. The quadruplet permutation loss is a generalization of the split-latent permutation loss, where the latents are swapped in pairs such that there is no direct path between the input sample and the reconstruction. The quadruplet permutation loss is illustrated with the quadruplet $(X_k^{(A)}, X_k^{(B)}, X_k^{(C)}, X_k^{(D)})$ where the latents are swapped such that $\boldsymbol{z}_k^{(\mathcal{S},a)}$ and $\boldsymbol{z}_k^{(\mathcal{S},c)}$ are swapped, and $\boldsymbol{z}_k^{(\mathcal{T},a)}$ and $\boldsymbol{z}_k^{(\mathcal{T},b)}$ are swapped, etc., before decoding. This is done for all quadruplet samples yielding the quadruplet permutation loss as the MSE loss between the input sample and the reconstruction.

The swap of latents is illustrated in Figure 1.

The quadruplet permutation loss is defined as

$$\mathcal{L}_{\text{QP}} = \frac{1}{4K} \sum_{k=1}^{K} \left(\|X_k^{(A)} - \hat{X}_k^{(A)}\|_2^2 + \|X_k^{(B)} - \hat{X}_k^{(B)}\|_2^2 + \|X_k^{(C)} - \hat{X}_k^{(C)}\|_2^2 + \|X_k^{(D)} - \hat{X}_k^{(D)}\|_2^2 \right) \tag{17}$$

The quadruplet permutation loss collapses to the split-latent permutation loss in some special cases. When input samples $X_k^{(A)}$ and $X_k^{(C)}$ are the same, then it becomes the same-subject permutation loss, and when input samples $X_k^{(A)}$ and $X_k^{(B)}$ are the same, then it becomes the same-task permutation loss. In the case where all input samples are the same, then the quadruplet permutation loss becomes the self-reconstruction loss.

A.1 Quadruplet permutation loss results

We provide here results using the same training and testing setup as in the paper. We conduct an ablation study on contrastive learning, latent-permutation and quadruplet permutation loss. The following four models are trained in five repetitions each:

- **SQP-AE**: Quadruplet permutation loss only.
- **CSQP-AE**: Quadruplet permutation loss and contrastive loss in conjunction.
- **SQLP-AE**: Quadruplet permutation loss and latent-permutation loss in conjunction.
- **CSQLP-AE**: Quadruplet permutation loss, contrastive loss and latent-permutation loss in conjunction.

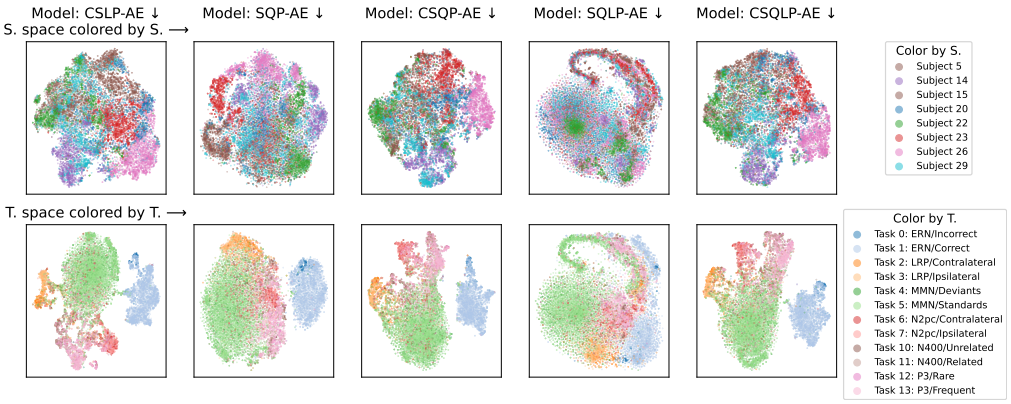
Table 1: Single-trial balanced subject classification accuracy (S.acc%), task-on-subject classification accuracy (T⊢S.acc%), task classification accuracy (T.acc%), subject-on-task classification accuracy (S⊢T.acc%), and zero-shot same-subject same-task ERP conversion MSE (S.s., S.t.), different-subject different-task ERP conversion MSE (D.s, D.t.), different-subject same-task ERP conversion MSE (D.s., S.t.), same-subject different-task ERP conversion MSE (S.s., D.t.). All ERP conversion MSE values have scales of 10^{-11}V^2. Epoch window was 1s.

Model	S.acc%	T⊢S.acc%	T.acc%	S⊢T.acc%	(S.s., S.t.)	(D.s, D.t.)	(D.s., S.t.)	(S.s., D.t.)
CSQLP-AE	76.10 ± 0.76	43.36 ± 0.46	46.17 ± 0.25	76.47 ± 0.46	1.91 ± 0.08	6.90 ± 0.05	3.43 ± 0.05	3.94 ± 0.16
SQLP-AE	69.26 ± 0.50	46.86 ± 1.35	44.80 ± 0.77	69.60 ± 0.25	$\mathbf{1.48 \pm 0.05}$	$\mathbf{6.44 \pm 0.03}$	$\mathbf{2.87 \pm 0.10}$	$\mathbf{2.97 \pm 0.05}$
CSQP-AE	73.04 ± 0.50	35.89 ± 0.42	44.83 ± 0.21	71.56 ± 0.64	6.20 ± 0.12	7.00 ± 0.08	6.60 ± 0.11	6.59 ± 0.10
SQP-AE	73.44 ± 0.33	43.92 ± 0.29	$\mathbf{48.88 \pm 0.13}$	70.42 ± 0.39	5.58 ± 0.07	6.49 ± 0.04	6.07 ± 0.04	5.99 ± 0.06
CSLP-AE	$\mathbf{80.32 \pm 0.28}$	45.41 ± 0.37	48.48 ± 0.34	79.64 ± 0.37	4.21 ± 0.12	20.06 ± 0.10	5.80 ± 0.15	6.65 ± 0.23
SLP-AE	74.63 ± 0.74	47.23 ± 0.31	47.00 ± 0.32	74.70 ± 0.73	3.82 ± 0.04	19.92 ± 0.10	6.12 ± 0.09	5.02 ± 0.08
C-AE	79.42 ± 0.48	37.34 ± 0.45	46.59 ± 0.23	73.27 ± 0.25	4.28 ± 0.06	20.28 ± 0.07	11.33 ± 0.47	10.64 ± 0.30

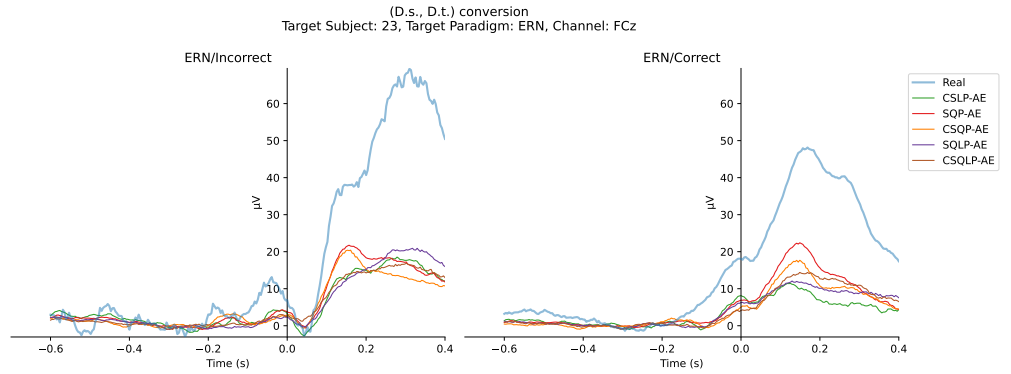
Here we see that the quadruplet permutation loss degrades performance on subject classification accuracy considerably, but with similar performance on task classification accuracy. We also see that the (D.s., D.t.) conversion loss now is on the same level as (D.s., S.t.) and (S.s., D.t.) conversion for the models without latent permutation. Adding the latent-permutation, although it introduces the structural encoding pathway to the model again, considerably decreases both (S.s., S.t.) conversion compared to CSLP-AE and SLP-AE from the paper.

We provide t-SNE [1, 14, 20] plots of the generalized models here and the CSLP-AE model from the paper in Figure 2a. Furthermore, we provide (D.s., D.t.) conversion examples in Figure 2b for the generalized models and the CSLP-AE model from the paper.

The SQP-AE model achieves specialized latent spaces with disentangled subject and task content as evident from the t-SNE plots in Figure 2a. Notably, the latent space is similar to the CSLP-AE model, but using simply the quadruplet permutation loss. In this sense, the quadruplet permutation loss is similar to the contrastive loss in that it encourages the model to learn a disentangled latent space while also directly learning all conversion schemes. Further research could focus on this quadruplet permutation loss and its relation to the contrastive loss. We view it as a contrastive loss that relates input samples to the output sample (the reconstruction), i.e. an auto-encoder contrastive loss, whereas a standard contrastive loss operates in the latent space itself. When we add the latent-permutation loss back to the model, we see that the structural encoding property occurs again in the SQLP-AE model, while the CSQLP-AE model does not have this property due to the contrastive loss used in specializing the latent space. Therefore, the SQP-AE model might be most comparable to the CSLP-AE model from the paper, as it both optimizes for conversion and latent disentanglement.



(a)



(b)

Figure 2: (a) *t*-SNE plots of split-latents as encoded on the test set (unseen subjects), colored by true labels. Rows show *subject* and *task* latent spaces, respectively, while columns indicate models CSLP-AE, SQP-AE, CSQP-AE, SQLP-AE, and CSQLP-AE respectively. **(b)** (D.s., D.t.) converted ERPs from the same five models for a random target subject and target paradigm. All latents used in conversion were from unseen subjects on the test set, i.e. unseen to unseen conversion.

A.2 Smoothness, disentanglement and conversion

In this section we will discuss how we can modify and guide the latent space to achieve the desired conversion. Meta-priors are auxiliary priors which, when present, can relieve strain on the model and provide guidance for the model to learn good representations.

Given the ith pair of input samples $(\boldsymbol{X}_i^a, \boldsymbol{X}_i^b)$, we use the encoder (E_θ) to split the latent space into two parts yielding $(\boldsymbol{z}_i^{(\mathcal{S},a)}, \boldsymbol{z}_i^{(\mathcal{T},a)})$ and $(\boldsymbol{z}_i^{(\mathcal{S},b)}, \boldsymbol{z}_i^{(\mathcal{T},b)})$. These two input samples have some factor in common, e.g. they are samples from the same task, and as such, we would like the latent space to be locally smooth, i.e. $\boldsymbol{z}_i^{(\mathcal{T},a)} \approx \boldsymbol{z}_i^{(\mathcal{T},b)}$ since they should encode the same task content (information) and a swap should have minimal effect on the output in the ideal conversion case. This can be thought of as the smoothness meta-prior proposed by Bengio et al. [2] where a representation should be invariant to local perturbations. Here the local perturbations are the other factors of variance in the data which do not belong to the given latent space, e.g. the subject variability in the case of the task latent space, for which we desire these representations to be invariant to. This is illustrated in Figure 3 where the desired property is the task content of the signal and the other factor of variation is the subject variability.

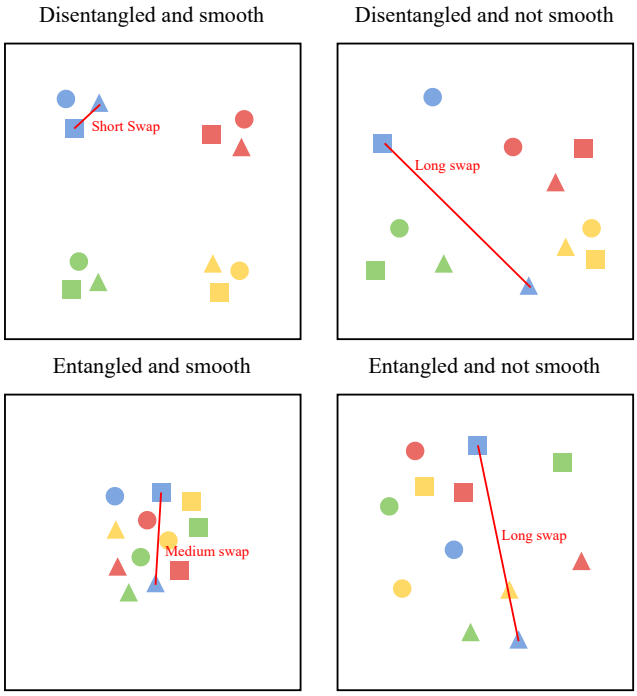


Figure 3: Illustration of latent space with and without the disentanglement and smoothness meta-priors present. Here colors denote the desired property of the latent space (e.g. the task content of the signal) and the shapes denote another factor of variation in the data (e.g. subject variability in the case of desired property task content) The disentangled and smooth latent space is the goal of the meta-priors. It is possible to perform conversion in the disentangled and not smooth latent space, however, there is a strain on the decoder since it might need to map multiple regions of the latent space to the same output. Smoothness itself is not enough as shown in the entangled and smooth latent space since there is no way to distinguish between classes in the conversion.

This is a property we wish to achieve by using the latent-permutation loss where we first encode a pair of samples

$$E_\theta(\boldsymbol{X}_i^a) = (\boldsymbol{z}_i^{(\mathcal{S},a)}, \boldsymbol{z}_i^{(\mathcal{T},a)}) \tag{18}$$

$$E_\theta(\boldsymbol{X}_i^b) = (\boldsymbol{z}_i^{(\mathcal{S},b)}, \boldsymbol{z}_i^{(\mathcal{T},b)}) \tag{19}$$

and then swap the latents corresponding to the desired property of the latent space, here the task latents, such that the reconstructed samples become

$$\hat{\boldsymbol{X}}_i^{(\mathcal{T},a)} = D_\phi(\boldsymbol{z}_i^{(\mathcal{S},a)}, \boldsymbol{z}_i^{(\mathcal{T},b)}) \tag{20}$$

$$\hat{\boldsymbol{X}}_i^{(\mathcal{T},b)} = D_\phi(\boldsymbol{z}_i^{(\mathcal{S},b)}, \boldsymbol{z}_i^{(\mathcal{T},a)}) \tag{21}$$

the latent-permutation loss used to guide the smoothness meta-prior is then the reconstruction loss between the input sample and the reconstructed sample

$$L_{LP}(\mathcal{L}; \boldsymbol{X}_i^a, \boldsymbol{X}_i^b) = \frac{1}{N}\sum_{i=1}^{N}\left(\|\boldsymbol{X}_i^a - \hat{\boldsymbol{X}}_i^{(\mathcal{L},a)}\|_2^2 + \|\boldsymbol{X}_i^b - \hat{\boldsymbol{X}}_i^{(\mathcal{L},b)}\|_2^2\right) \tag{22}$$

in this case the desired property is the latent space and the loss becomes

$$L_{LP}(\mathcal{T}; \boldsymbol{X}_i^a, \boldsymbol{X}_i^b) = \frac{1}{N}\sum_{i=1}^{N}\left(\|\boldsymbol{X}_i^a - \hat{\boldsymbol{X}}_i^{(\mathcal{T},a)}\|_2^2 + \|\boldsymbol{X}_i^b - \hat{\boldsymbol{X}}_i^{(\mathcal{T},b)}\|_2^2\right) \tag{23}$$

Consider the first term of the sum in Eq. (23) where we have the L_2-norm between the input sample $\boldsymbol{X}_i^a$ and the reconstructed sample $\hat{\boldsymbol{X}}_i^{(\mathcal{T},a)}$. When local smoothness is obtained then

$$\boldsymbol{z}_i^{(\mathcal{T},a)} \approx \boldsymbol{z}_i^{(\mathcal{T},b)} \tag{24}$$

which then implies that

$$D_\phi(\boldsymbol{z}_i^{(\mathcal{S},a)}, \boldsymbol{z}_i^{(\mathcal{T},b)}) \approx D_\phi(\boldsymbol{z}_i^{(\mathcal{S},a)}, \boldsymbol{z}_i^{(\mathcal{T},a)}) \tag{25}$$

$$\hat{\boldsymbol{X}}_i^{(\mathcal{T},a)} \approx \hat{\boldsymbol{X}}_i^a \tag{26}$$

as this local approximation in Eq. (24) becomes more accurate where $D_\phi(\boldsymbol{z}_i^{(\mathcal{S},a)}, \boldsymbol{z}_i^{(\mathcal{T},a)}) = \hat{\boldsymbol{X}}_i^a$ is the self-reconstruction. This means that the reconstruction loss between the input sample and the reconstructed sample becomes

$$\|\boldsymbol{X}_i^a - \hat{\boldsymbol{X}}_i^{(\mathcal{T},a)}\|_2^2 \approx \|\boldsymbol{X}_i^a - \hat{\boldsymbol{X}}_i^a\|_2^2 \tag{27}$$

$$\tag{28}$$

Therefore, as local smoothness becomes more exact, as achieved by a consistent encoder and decoder, the latent-permutation loss approximates the self-reconstruction loss, i.e. the default auto-encoder reconstruction loss.

However, the local smoothness meta-prior is not enough to achieve the desired conversion, and a trivial solution where all latents are identical is a possible outcome of this smoothness meta-prior. Therefore, we need to introduce a disentanglement method to ensure that the latents are not identical. We do this by introducing contrastive learning to specialize each latent space. This is beneficial in the following ways: (a) it ensures that the latents are not identical, i.e. different classes of the desired property are represented by different groupings/clusters of latents, and (b) contrastive learning itself provides in-cluster smoothness as the similarity between samples in the same class is maximized.

B Conversion schemes

B.1 Conversion schemes in-depth

Given the kth sample $\boldsymbol{X}_k$ with corresponding task t_k and subject s_k from the test set, we can encode the samples into task and subject latents

$$E_\theta(\boldsymbol{X}_k) = (\boldsymbol{z}_k^{(\mathcal{S})}, \boldsymbol{z}_k^{(\mathcal{T})}) \quad \forall k \in \{1, \dots, N\} \tag{29}$$

where $\boldsymbol{z}_k^{(\mathcal{S})}$ and $\boldsymbol{z}_k^{(\mathcal{T})}$ are the kth subject and task latents respectively. We can then use these latents to perform conversion using the conversion method described in the paper. Using different conversion schemes we can produce different levels of abstraction from the input sample by choosing which latents we sample from depending on a target task and subject class.

Conversion is only valid if the latents correspond to input samples from the same subject and task class as the target subject and task class.

For this purpose, given *target* subject and task label σ and γ respectively, a valid conversion pair of subject and task latents can be sampled from the test set in the following proposed ways:

Same subject, same task (S.s., S.t.): $\boldsymbol{z}_k^{(\mathcal{T})}$ where $t_k = \gamma \wedge s_k = \sigma, \boldsymbol{z}_k^{(\mathcal{S})}$ where $t_k = \gamma \wedge s_k = \sigma$

Same subject, different task (S.s., D.t.): $\boldsymbol{z}_k^{(\mathcal{T})}$ where $t_k = \gamma \wedge s_k = \sigma, \boldsymbol{z}_k^{(\mathcal{S})}$ where $t_k \neq \gamma \wedge s_k = \sigma$

Different subject, same task (D.s., S.t.): $\boldsymbol{z}_k^{(\mathcal{T})}$ where $t_k = \gamma \wedge s_k \neq \sigma, \boldsymbol{z}_k^{(\mathcal{S})}$ where $t_k = \gamma \wedge s_k = \sigma$

Different subject, different task (D.s., D.t.): $\boldsymbol{z}_k^{(\mathcal{T})}$ where $t_k = \gamma \wedge s_k \neq \sigma, \boldsymbol{z}_k^{(\mathcal{S})}$ where $t_k \neq \gamma \wedge s_k = \sigma$

where k is the index of the sample in the test set, $\boldsymbol{z}_k^{(\mathcal{T})}$ and $\boldsymbol{z}_k^{(\mathcal{S})}$ are the kth task and subject latents respectively. In the results we sample N of valid pairs for a given conversion scheme. The effect of the number of samples used in the construction of a converted ERP is analyzed in Section B.2.

The conversion schemes are illustrated in Figure 4.

Figure 4: The four different conversion schemes. The colors represent the subject of the latent, and the shapes represent the task of the latent. Each input sample has both a subject and task latent. The marked regions (red ellipse) indicate the latents from the target task or subject condition, while the directly marked latents (red outline) represent the latents from the condition scheme. The target task and subject conditions are the same for all conversion schemes because the latents need to be valid for conversion. The target task is represented by triangle shape and the target subject is represented by blue color. The conversion schemes differentiate between using latents from the same or different out-of-space target. For example, in the (S.s., D.t.) conversion scheme, since the target subject is blues, we only sample from the region in the subject latent space that corresponds to blues. Additionally, the D.t. conversion notation means that we only sample from latents that are not the target subject, which in this case is every shape but the triangles. Since the target task is triangles, we only sample from the region in the task latent space that corresponds to triangles. Furthermore, the S.s. conversion notation means that we only sample from latents that are the same as the target task, which in this case is the blue latents. So, from the subject latent space, we sample latents that are blue but not triangles, and from the task latent space, we sample latents that are both triangles and blue in the (S.s., D.t.) conversion scheme.

B.2 Number of samples for latent pair sampling

In this section we show the effect of the number of samples N used for sampling the latent pairs for conversion. We show the results for the C-AE, SLP-AE, and CSLP-AE models for all conversion schemes. The results are shown in Figure 5. We see that the conversion loss decreases as the number of samples increases. However, the decrease is not very significant for $N > 1000$ samples. We choose $N = 2000$ samples for the experiments in the paper. The ERP conversion loss provided here is the mean of the conversion loss across all subject and paradigm combinations on the test set for each conversion scheme and for a given number of latent pair samples N.

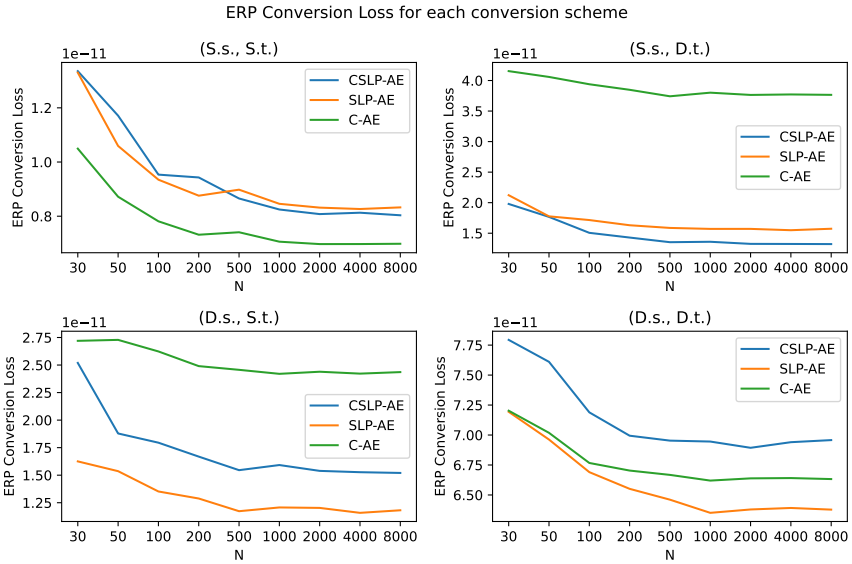


Figure 5: Mean ERP conversion loss across all subject and paradigm combinations on the test set for each conversion scheme. Note that the x-axis is not linear.

C Checklist of items to include in a DL-EEG study

We fill in a checklist of items to include in a DL-EEG study as proposed and provided by Roy et al. [19].

C.1 Data

- **Number of subjects:** 40 in total; 15 male, 25 female, mean age and SD 21.5 ± 2.87, age range 18-30. All had normal color perception, normal/corrected-to-normal vision, no history of neurological injury/disease indicated by self-report. For more details, see Kappenman et al. [12].
- **Electrode montage including reference(s):** International 10/20 System with 30 scalp electrodes. Single-ended mode without the use of a reference. Channels were referenced offline to the average of EEG P9 and EEG P10 for the MMN, N2pc, N400, P3, LRP and ERN component paradigms, and to the EEG average for the N170 paradigm. For more details, see Kappenman et al. [12].
- **Shape of one example:** 256 samples $\times$ 30 channels.
- **Data augmentation technique:** None.
- **Number of examples in training, validation and test sets:** In total: 49943 examples in training set, 7161 examples in evaluation (validation) set, and 14485 examples in test set. 28 subjects in training set, 4 subjects in evaluation (validation) set, and 8 subjects in test set.

C.2 EEG Processing

- **Temporal filtering:** Before digitization all signals were low-pass filtered using a 5th order sinc filter, 204.8 Hz half-power cutoff. After digitization and down-sampling all signals were high-pass filtered using a non-causal Butterworth impulse response function, 0.1 Hz half-amplitude cut-off, and 12 dB/oct roll-off.[1]
- **Spatial filtering:** None.
- **Artifact handling techniques:** None.
- **Resampling:** Digitized at 1024 Hz with 24 bits. Downsampled to 256 Hz.

C.3 Neural network architecture

- **Architecture type:** Autoencoder with residual CNNs and transformer bottleneck. Split latent space.
- **Number of layers**: 69 layers in total. 3 layers per ConvBlock. 4 layers per transformer. 24 layers in encoder: 1 input convolution, 5 ConvBlocks, 4 strided convolutions, and 1 transformer. 21 layers in bottleneck: 7 ConvBlocks. 24 layers in decoder: 1 transformer, 5 ConvBlocks, 4 transposed convolutions, and 1 output convolution. Illustrated in Figure 1 in the paper.
- **Number of learnable parameters:** ≈ 10 million in total. ≈ 5 million in encoder (including pre-latent space bottleneck). ≈ 5 million in decoder (including post-latent space bottleneck).

C.4 Training hyperparameters

- **Parameter initialization:** All layer initialization is done using the default PyTorch package [16] initialization methods. Convolutions are initialized using He initialization [11]. The linear layers of the transformer are initialized using Glorot initialization [8]. The learned temperature-scales for similarity matrices in the contrastive loss were initialized to $\tau_{\mathcal{S}} = \frac{1}{0.07} \approx 14.2857143$ and $\tau_{\mathcal{T}} = \frac{1}{0.07} \approx 14.2857143$ following the implementation of Radford et al. [18][2].

[1] Procedure after digitization available here: `https://github.com/lucklab/ERP_CORE/blob/master/N400/N400%20Analysis%20Procedures.pdf` Note: we only used Script #1 of this procedure.

[2] Their initialization is available here: `https://github.com/openai/CLIP/blob/a9b1bf5920416aaeaec965c25dd9e8f98c864f16/clip/model.py#LL295C65-L295C65` Note: the

- **Loss function:** Latent permutation, contrastive loss, autoencoder reconstruction loss, supervised cross-entropy.
- **Batch size:** 256.
- **Number of epochs:** 200.
- **Stopping criterion:** None.
- **Regularization:** None.
- **Optimization algorithm:** Adam optimizer [13].
- **Learning rate schedule and optimizer parameters:** Learning rate cosine annealing starting at 100th epoch with a final divide-factor of 10. Learning rate starts at $\eta_0 = 1.0 \times 10^{-4}$ and ends at $\eta_T = 1.0 \times 10^{-5}$. Running average coefficients were $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Epsilon was $\epsilon = 1.0 \times 10^{-8}$.
- **Values of all hyperparameters for the results that are presented in the paper:**
  - Number of channels per layer: 256.
  - Latent dimension: 64.
  - Number of downsampling layers: 4.
  - Latent example size: $\frac{256}{2^4} \times 64 = 16 \times 64$. Where 16 is the time-resolution in the latent space and 64 is the number of channels. The effective latent size was $16 \times 64 = 1024$.
  - Loss weights: All losses were weighed equally (by 1).
- **Hyperparameter search method:** Non-critical hyperparameters were found during development by performance on the evaluation set (validation). Critical hyperparameters; latent dimension size and number of repeated time-resolution downsamples in the encoder and decoder, were validated on the evaluation set (validation) using grid-search.

C.5 Performance and model comparison

- **Performance metrics:** Balanced classification accuracy, ERP Conversion loss and t-SNE plots in the paper. Per-subject and per-task F1-score, precision, recall provided here in supplementary materials, together with confusion matrices and training/test t-SNE plots. Paradigm classification repurposed from task-classification also reported here in supplementary.
- **Type of validation scheme:** 5-fold stratified cross-validation on unseen subjects.
- **Description of baseline models:** All models used the same autoencoder model architecture with split latent spaces, although some models did not have a decoder. CSLP-AE used same-task and same-subject latent permutation in conjunction with subject and task contrastive loss. SLP used same-task and same-subject latent permutation. C-AE used standard autoencoder reconstruction loss in conjunction with subject and task contrastive loss. CL used subject and task contrastive loss (and no decoder). CE used supervised task and subject cross-entropy loss (and no decoder). CE(t) used supervised task cross-entropy loss (and no decoder). C-AE was considered the baseline for solving the conversion problem using contrastive learning and auto-encoders.

parameter is initialized as the logarithm, and when the similarity matrices are scaled the exponential is used to constrain the parameter to the positive numbers domain.

D Data and training

D.1 Model overview

Here is provided a quick overview of the models used in the paper. All models used the same architecture and hyperparameters except the common spatial pattern (CSP) method [4, 15] which used no model.

- **CSLP-AE**: Contrastive split-latent permutation over both subjects and latents ($L_{LP}(\mathcal{S};\cdot,\cdot)$, $L_{LP}(\mathcal{T};\cdot,\cdot)$, $L_{\text{CLIP}}(\mathcal{S};\cdot,\cdot)$ and $L_{\text{CLIP}}(\mathcal{T};\cdot,\cdot)$)
- **SLP-AE**: Split-latent permutation over both subjects and latents ($L_{LP}(\mathcal{S};\cdot,\cdot)$ and $L_{LP}(\mathcal{T};\cdot,\cdot)$)
- **C-AE**: Contrastive over both subjects and latents and the default auto-encoder reconstruction loss ($L_{\text{CLIP}}(\mathcal{S};\cdot,\cdot)$, $L_{\text{CLIP}}(\mathcal{T};\cdot,\cdot)$ and self-reconstruction loss)
- **AE**: Default auto-encoder reconstruction loss (self-reconstruction loss)
- **CL**: Contrastive learning with no decoder ($L_{\text{CLIP}}(\mathcal{S};\cdot,\cdot)$ and $L_{\text{CLIP}}(\mathcal{T};\cdot,\cdot)$)
- **CE**: Supervised cross-entropy with no decoder (supervised cross-entropy loss in both latent spaces)
- **CE(t)**: Supervised cross-entropy with no decoder (supervised cross-entropy loss in the task latent space)
- **CSP**: Common spatial pattern [4, 15] using the multi-class generalization from Grosse-Wentrup and Buss [10] trained to discriminate on tasks, where latent-space from transform into "CSP space" is used as the task-latent space.

D.2 Data, training and evaluation details

We use the ERP Core dataset[3] from Kappenman et al. [12] providing a standardized ERP dataset containing data from 40 subjects across six different tasks based on seven widely used ERP components:

- **N170**: Face Perception Paradigm
- Mismatch Negativity (**MMN**): Passive Auditory Oddball Paradigm
- **N2pc**: Simple Visual Search Paradigm
- **N400**: Word Pair Judgement Paradigm
- **P3**: Active Visual Oddball Paradigm
- Lateralized Readiness Potential (**LRP**): Flankers Paradigm
- Error-related Negativity (**ERN**): Flankers Paradigm

However, since the N170 paradigm used a different offline reference than the other paradigms, we excluded it from the data set. We extracted epochs according to the time-locking events from Kappenman et al. [12] yielding two ERP components per paradigm.

We wanted to perform minimal pre-processing on the data, and as such we only followed the ERP procedure in Script #1 of each ERP component, i.e. up until ICA preparation[4], since ICA preparation and artifact rejection requires expert knowledge. Following this procedure we pre-process the data as follows:

- Shift the event codes of paradigms with LCD monitor delay by 26 ms
- Downsample from 1024 Hz to 256 Hz
- Re-reference to P9 and P10 for all included paradigms
- Create bipolar HEOG and VEOG channel from `HEOG_left - HEOG_right` and `VEOG_lower - FP2` respectively

[3] ERP Core info: `https://erpinfo.org/erp-core`. Data available at: `https://osf.io/thsqg/`

[4] See the full ERP Core procedure here: `https://github.com/lucklab/ERP_CORE/blob/master/ERN/ERN%20Analysis%20Procedures.pdf`

- Remove DC offsets
- Apply high-pass filter (non-causal Butterworth impulse response function, half-amplitude cutoff at 0.1 Hz, 12 dB/oct roll-off)
- Normalization using the per-channel mean and standard deviation over all examples

Following the pre-processing we split the data into a training set, an evaluation (validation) set and a test set based on subjects, such that the training set had 70% (28 subjects) of the data, the evaluation (validation) set had 10% (4 subjects) and the test set had 20% (8 subjects). The subjects were distributed across the data sets as follows

- Training set: {1, 2, 3, 6, 8, 9, 10, 11, 12, 13, 16, 17, 18, 19, 21, 24, 25, 28, 30, 31, 32, 34, 35, 36, 37, 38, 39, 40}
- Evaluation (validation) set: {4, 7, 27, 33}
- Test set: {5, 14, 15, 20, 22, 23, 26, 29}

with total samples across the data sets as follows

- Training set: 49943 examples
- Evaluation (validation) set: 7161 examples
- Test set: 14485 examples

The ERP component distribution across all samples was as follows

- ERN/Correct: 14083
- ERN/Incorrect: 1679
- LRP/Contralateral: 3160
- LRP/Ipsilateral: 3860
- MMN/Standards: 23270
- MMN/Deviants: 7967
- N2pc/Ipsilateral: 2790
- N2pc/Contralateral: 2835
- N400/Related: 2237
- N400/Unrelated: 2260
- P3/Rare: 1340
- P3/Frequent: 6108

Notice the class-imbalance between MMN/Standards and MMN/Deviants and to other components as well. We performed undersampling in the cross-validation procedure to account for this.

All models used the same architecture as described in the paper with the following hyperparameters:

- Number of downsampling blocks: 4
- Latent dimension size: 64
- Channels: 256
- Softmax Temperature: Initialized as ≈ 14.29 for models using contrastive loss

The encoder had ≈ 5 million parameters and the decoder had ≈ 5 million parameters, for a total of ≈ 10 million parameters.

The models were all trained using the PyTorch package [16] and logging was done using the Weights & Biases MLOps platform[5] [3]. The models were trained on a single NVIDIA A100 GPU (Nvidia Corporation, Santa Clara, CA, USA) with 40 GB of memory. In total ≈ 600 total GPU hours were

[5]Main site: `https://wandb.ai/site`

used for training, with ≈ 100 GPU hours for the main results provided in the paper. One model takes about 2 hours to train on a single A100 GPU.

The training procedure used the following hyperparameters:

- Batch size: 256
- Epochs: 200
- Optimizer: Adam [13] with learning rate 0.001
- Learning rate scheduler: Cosine annealing schedule[6] with a `div_factor=10` starting at epoch 100
- Loss weighting: Equal weighting across all losses

It is worth mentioning that the notion of epochs used here is fluid since the dataset was sampled from instead of exhaustively traversed. An epoch was counted as the number of times the model was trained on a number of samples equal to the size of the dataset. Since the same-task batch construction requires two samples for each task, it is expected that the underrepresented classes are repeated more than the overrepresented classes.

For models using supervised cross-entropy a softmax layer (a linear-layer with softmax activation) was applied before the loss function. For models using contrastive loss the softmax temperature is a learned parameter which was initialized as ≈ 14.29 following the implementation of Radford et al. [18][7]. The value is initialized as the logarithm and the exponential is taken before scaling the similarity matrix to constrain it to the positive numbers domain.

During training some loss methods required special batch construction while some did not. For methods not requiring a special method we simply sampled a batch of examples from the training set. For methods requiring special batch reconstruction we sampled from the training set with conditions as follows:

- **Same-task permutation and task latent space contrastive loss**: A pair of examples is sampled from each task class to yield $N_{\#T}$ pairs of examples, where $\#T$ is the number of unique tasks in the training set. Each pair has a different task class. The batch size is then $N_{\#T} \times 2$.
- **Same-subject permutation and subject latent space contrastive loss**: A pair of examples is sampled from each subject class to yield $N_{\#S}$ pairs of examples, where $\#S$ is the number of unique subjects in the training set. Each pair has a different subject class. The batch size is then $N_{\#S} \times 2$.

Given batch size B, to match the batch size of the other methods we then complete $\lfloor B/(N_{\#T} \times 2) \rfloor$ or $\lfloor B/(N_{\#S} \times 2) \rfloor$ of such batch cosntructions, where the loss is now the mean over the inner-batch and then the mean over the outer-batch. In this case where there are 28 different subjects in the training set, the batch size for special subject batch construction is achieved by performing $\lfloor 256/(28 \times 2) \rfloor = 4$ batch constructions. There are 12 different tasks in the training set and so the batch size for special task batch construction is achieved by performing $\lfloor 256/(12 \times 2) \rfloor = 10$ batch constructions.

During training the models were evaluated on the evaluation set every epoch using a simple stratified train/test-split of the dataset with a 20% test ratio. The training split was undersampled using the least represented class of the set. A XGBoost[8] [5] classifier was trained on the training split to classify tasks and evaluated on the test split to yield the task classification accuracy.

Once training was complete, the models were evaluated on the test set. Here two 5-fold crossvalidations were performed, one for the subject latents and one for the task latents. This is illustrated in Figure 6.

[6] `https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.CosineAnnealingLR.html`

[7] Their initialization is available here: `https://github.com/openai/CLIP/blob/a9b1bf5920416aaeaec965c25dd9e8f98c864f16/clip/model.py#LL295C65-L295C65`

[8] Documentation: `https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBClassifier`

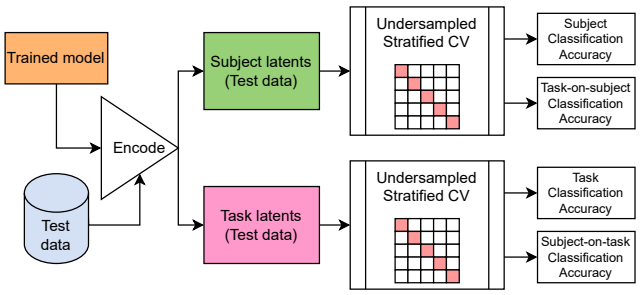


Figure 6: Illustration of the two cross-validations performed for each latent space.

Evaluation was performed by first encoding every sample in the test set yielding a subject and task latent for each example. The subject latent crossvalidation was stratified on subjects and the task latent crossvalidation was stratified on tasks. In each crossvalidation the training split of the fold was undersampled according to the least represented class of the set. Two XGBoost classifiers were trained on the training split of each crossvalidation fold to classify subjects and tasks respectively and evaluated on the test split to yield the subject and task classification accuracy respectively on each latent space. From this we obtain subject classification on subject latents, task classification on subject latents, subject classification on task latents, and task classification on task latents. We used the balanced accuracy metric[9] from the `scikit-learn` package [17] to account for the class imbalance in the dataset. This is equivalent to the average of the per-class recall.

The XGBoost classifier had the following hyperparameters:

- `n_estimators`: 300
- `max_bin`: 100
- `learning_rate`: 0.3
- `grow_policy`: `depthwise`
- `objective`: `multi:softmax`
- `tree_method`: `gpu_hist`

The CSP method used the following hyperparameters:

- `cov_est`: `epoch`
- `n_components`: 16
- `norm_trace`: `True`
- `reg`: `shrinkage`
- `shrinkage`: 0.1

We used the MNE Package [9] implementation of CSP[10].

Both sets of hyperparameters for the CSP and XGBoost methods were found using the Bayesian optimization sweep implementation of the Weights & Biases MLOps platform[11] [3] on the evaluation (validation) set.

We provide the task classification accuracies on the evaluation (validation) set during training in Figure 7.

[9] Documentation: `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html`

[10] `https://mne.tools/stable/generated/mne.decoding.CSP.html`

[11] Main site: `https://wandb.ai/site` Bayesian Optimization sweep: `https://docs.wandb.ai/guides/sweeps`

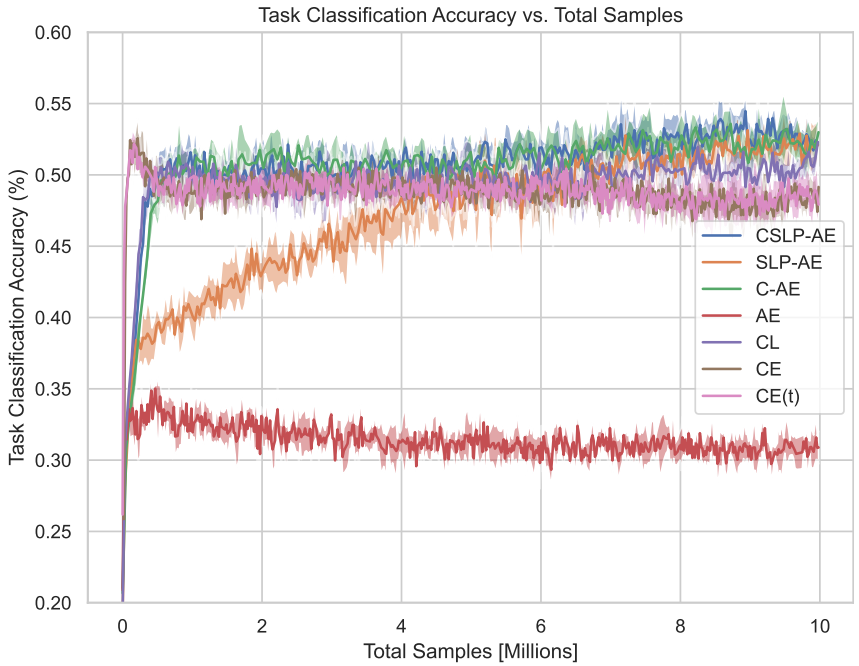


Figure 7: Task classification accuracy progression on evaluation (validation) set over training for all deep learning models. The standard deviation over the five repeats is illustrated as the corresponding area around the mean.

D.3 Additional datasets

EEG Motor Movement/Imagery Dataset We used the raw waveform data from the EEGMMI dataset using only a 3 second epoch window similarly to [21]. We used the same training and evaluation methodology as for the ERP Core for the EEGMMI dataset using a train/test/validation split with the following division of subjects:

- Training set: {1, 2, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 25, 26, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 44, 45, 46, 47, 49, 50, 51, 53, 54, 55, 56, 57, 59, 62, 63, 64, 69, 71, 73, 74, 75, 77, 79, 82, 83, 84, 85, 86, 90, 93, 94, 97, 98, 99, 102, 103, 104, 105, 107, 109}
- Validation set: {23, 24, 27, 52, 61, 68, 72, 78, 80, 87}
- Test set: {3, 7, 29, 42, 43, 48, 58, 60, 65, 66, 67, 70, 76, 81, 91, 95, 96, 101, 108}

The samples had 3 seconds duration at a sampling rate of 160 Hz, and we therefore needed to use five blocks instead of four to match the latent dimension of ERP Core models. The models were run to convergence for 800 epochs. No other tweaks were made to conform the framework to this dataset.

No data-specific hyperparameter tuning was performed here, and all hyperparameters, except the number of blocks, were reused from the ERP Core models.

Sleep-EDF Expanded Database Since the SleepEDFx dataset contains a limited number of EEG channels, we only considered a single EEG channel and applied a short-time Fourier transform to fit the data to the same setup as used for ERP Core. We have employed the conventional 30s time series windows with associated sleep stage labels as common in the sleep stage literature (as well as being the conventional clinical procedure). The single channel EEG time information sampled at 100 Hz is transformed using a 128-point short-time-Fourier transform with a 100-point window size and 15-point step size.

We applied our framework (from the ERP Core setup) "as is" with the following minor modifications: To match the latent dimension of the models from ERP Core, we opted to only use three blocks instead of four and resized the latent dimension's temporal resolution from 64 to 40 to end up with

latents of size 25×40 compared to the ERP Core models which had 16×64 latents. Finally, these models were only run for 50 epochs.

Instead of using a train/test/dev split evaluation method, we performed a 5-fold cross-validation where we trained a new model with the same architecture on each fold. For each fold, however, we used the first seven subjects for run-time validation.

No data-specific hyperparameter tuning was performed here, and all hyperparameters, except the number of blocks, were reused from the ERP Core models.

E Impact of latent dimension and number of blocks

In this material we provide a surface-level grid search for the latent dimension size and the number of layers in the encoder downsample block. We provide results with half the latent dimension and double the latent dimension of the default model. We also provide results with two less and two more layers than the default model. The latent dimension size is the number of channels for each time step in the latent representation. The number of layers is the number of downsample blocks in the encoder, which subsequently determines the time resolution of the latent representation. Here the time resolution is reduced by $2^{N_{\text{layers}}}$ where N_{layers} is the number of layers. As such, the size of the latent representation is $L_{\text{eff.size}} = C_{\text{latent}} \times T_{\text{latent}}$ where C_{latent} is the latent dimension size, $T_{\text{latent}} = 256/2^{N_{\text{layers}}}$ is the time resolution and $L_{\text{eff.size}}$ is the effective size of the latent representation.

The metric of comparison is the task classification accuracy on the evaluation (validation) set of the data. A cross-validation is not performed here, but rather a simple train/test split of the data to yield a training portion and a testing portion. We used a 20% test split ratio.

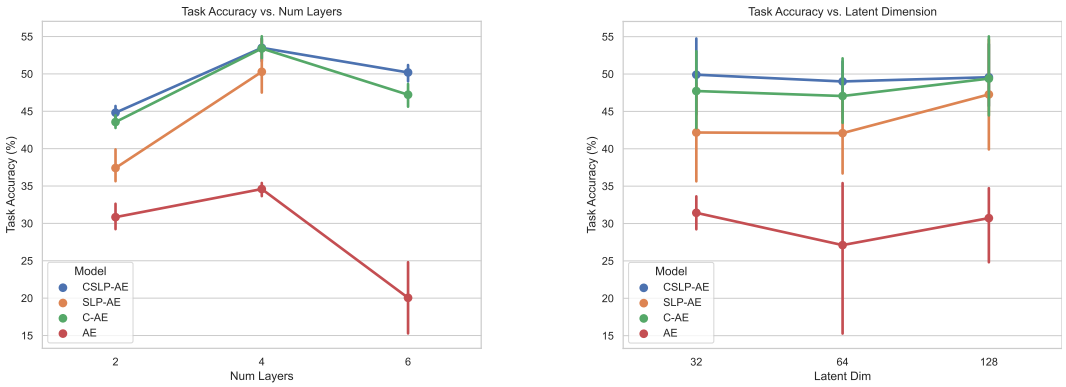
It is worth noting that the largest latent representations here ($C_{\text{latent}} = 128$ and $N_{\text{layers}} = 2$) had an effective size of $L_{\text{eff.size}} = 8192$ that is larger than the input sample size, $30 \times 256 = 7680$.

Some models did not converge/diverged/failed to train yielding a degenerate model. These models were excluded from the grid search results. The results are shown in Table 2 and Figure 8.

Latent dimension seemed to have little to no overall effect on the task classification accuracy on the evaluation (validation) set, whereas $N_{\text{layers}} = 4$ had the best performance of the three values tested.

Table 2: Grid search results for latent dimension and number of blocks in terms of task classification accuracy. N/A indicates degenerate model. Bold denotes the best set of latent dimension and number of downsampling blocks per model.

Model	Latent Dim / Num Layers	32	64	128
CSLP-AE	2	44.58	44.13	45.71
	4	**54.76**	51.69	54.01
	6	50.39	51.19	48.99
SLP-AE	2	35.64	36.69	39.91
	4	48.72	47.50	**54.63**
	6	N/A	N/A	N/A
C-AE	2	42.76	43.46	44.46
	4	53.05	52.11	**55.04**
	6	47.37	45.61	48.67
AE	2	29.22	30.65	32.63
	4	33.64	**35.41**	34.73
	6	N/A	15.28	24.81



(a) **(b)**

Figure 8: Grid search results for latent dimension and number of layers. Degenerate models were excluded. **(a)** Task accuracy vs. number of layers for models with latent dimensions of 32, 64, and 128. **(b)** Task accuracy vs. latent dimension for models with 2, 4, and 6 layers.

F Additional results

F.1 Per-subject and per-task results

In this material we provide per-subject and per-task F1-scores, precision, and recall from the cross-validation on the test set, and (S.s., S.t.) conversion error, (S.s., D.t.) conversion error, (D.s., S.t.) conversion error, and (D.s., D.t.) conversion error sampled from the test set with $N = 2000$. See Section B.2 for an analysis of how this parameter affects the conversion error.

In the below result tables *sem* denotes the standard error of the mean over the five repeats.

Table 3: Per-subject single-trial F1 score (%) from cross-validation on unseen subjects (test data)

	name	CE(t)	CE	CL	AE	C-AE	SLP-AE	CSLP-AE
all	mean	42.75	79.20	78.73	60.28	79.20	74.46	**80.15**
	sem	0.70	0.39	0.52	0.26	0.51	0.92	0.30
S5	mean	54.77	81.28	81.56	69.76	80.40	78.62	**83.45**
	sem	0.70	0.79	0.73	0.63	1.09	0.71	1.22
S14	mean	38.93	85.61	85.48	60.15	**87.47**	83.98	86.04
	sem	1.18	1.14	0.83	1.25	0.70	3.27	0.47
S15	mean	38.92	75.29	**79.26**	58.33	78.04	70.10	77.68
	sem	0.70	0.67	1.31	0.63	2.04	1.15	0.74
S20	mean	35.32	70.01	70.18	49.01	70.14	64.99	**71.61**
	sem	0.70	0.69	0.60	0.54	1.18	0.98	0.79
S22	mean	50.65	79.68	77.74	66.28	80.01	**82.41**	80.84
	sem	0.74	0.41	0.98	0.58	0.69	0.75	0.33
S23	mean	50.15	**80.98**	77.68	68.67	77.40	77.57	80.57
	sem	0.85	1.14	0.48	0.29	1.29	0.38	0.63
S26	mean	41.54	91.43	91.32	63.79	91.82	76.12	**92.15**
	sem	0.88	0.73	0.40	0.48	0.43	0.89	0.24
S29	mean	31.74	**69.36**	66.66	46.31	68.36	61.99	68.92
	sem	0.93	0.98	1.49	0.29	0.63	0.93	0.80

Table 4: Per-task single-trial F1 score (%) from cross-validation on unseen subjects (test data)

	name	CSP	CE(t)	CE	CL	AE	C-AE	SLP-AE	CSLP-AE
all	mean	36.90	50.80	49.74	50.91	30.38	52.46	52.26	**53.31**
	sem	0.09	0.16	0.26	0.44	0.24	0.24	0.24	0.23
ERN/Incorrect	mean	21.80	42.47	40.80	38.62	23.11	41.17	44.42	**46.19**
	sem	0.14	0.78	0.57	0.45	0.24	0.65	0.95	0.97
ERN/Correct	mean	59.70	89.00	87.76	88.61	42.73	89.43	89.33	**91.17**
	sem	0.43	0.44	0.29	0.30	0.68	0.17	0.36	0.31
LRP/Contralateral	mean	26.05	**46.55**	42.58	42.89	26.92	45.19	40.90	44.77
	sem	0.82	0.40	0.88	0.62	0.43	0.71	0.57	1.16
LRP/Ipsilateral	mean	26.45	46.75	44.74	46.32	26.97	47.51	45.27	**48.78**
	sem	0.63	1.17	0.57	0.58	0.59	0.99	0.22	0.67
MMN/Deviants	mean	23.87	30.13	29.80	30.64	22.06	30.92	**32.14**	31.28
	sem	0.25	0.08	0.41	0.22	0.34	0.58	0.30	0.48
MMN/Standards	mean	38.45	49.31	48.22	50.17	33.64	52.09	**52.29**	51.75
	sem	0.45	0.30	0.52	0.74	0.70	0.66	0.68	0.21
N2pc/Contralateral	mean	28.52	32.21	32.74	33.08	24.21	35.10	36.57	**37.43**
	sem	0.16	1.32	0.80	1.49	0.50	0.37	0.55	1.30
N2pc/Ipsilateral	mean	30.91	29.22	28.29	31.91	20.06	**36.24**	32.99	35.21
	sem	0.47	0.49	0.38	1.12	0.30	0.76	0.95	1.12
N400/Unrelated	mean	19.19	24.97	25.42	27.07	18.95	29.98	28.00	**31.23**
	sem	0.27	0.91	0.85	0.95	0.59	0.69	0.74	1.30
N400/Related	mean	21.90	27.30	25.92	28.33	17.47	27.87	28.55	**29.63**
	sem	0.54	0.38	0.73	0.88	0.65	0.48	0.59	0.48
P3/Rare	mean	16.91	23.04	22.89	23.26	13.84	24.70	**26.13**	25.48
	sem	0.16	0.56	0.87	0.80	0.76	0.87	0.86	0.64
P3/Frequent	mean	32.14	41.20	40.62	39.75	24.93	41.65	41.42	**44.85**
	sem	0.50	1.08	0.54	1.26	0.92	1.02	0.31	1.21

Table 5: Per-subject single-trial precision (%) from cross-validation on unseen subjects (test data)

	name	CE(t)	CE	CL	AE	C-AE	SLP-AE	CSLP-AE
all	mean	42.85	79.17	78.70	60.43	79.17	74.47	**80.15**
	sem	0.70	0.39	0.53	0.28	0.51	0.92	0.30
S5	mean	55.30	81.92	81.68	71.19	80.56	79.34	**83.46**
	sem	0.95	0.54	0.96	1.03	1.06	0.63	1.06
S14	mean	39.49	83.91	83.07	60.66	**84.87**	81.64	83.77
	sem	1.08	1.09	0.95	1.49	0.66	3.28	0.50
S15	mean	40.30	75.59	**79.90**	56.22	77.19	69.67	76.74
	sem	0.66	0.71	1.35	0.49	1.81	0.97	0.99
S20	mean	33.45	72.15	71.68	49.35	73.51	64.55	**74.94**
	sem	0.59	0.57	0.84	0.61	0.96	1.02	0.58
S22	mean	48.30	79.57	78.78	61.74	81.12	79.97	**81.62**
	sem	0.92	0.42	0.80	0.43	0.41	0.67	0.42
S23	mean	50.43	**81.66**	76.84	71.34	76.37	78.39	79.82
	sem	0.52	1.08	0.51	0.29	1.02	0.56	0.68
S26	mean	42.96	88.50	88.12	63.72	89.18	77.33	**89.26**
	sem	1.17	0.80	0.56	0.81	0.63	0.72	0.30
S29	mean	32.54	70.09	69.56	49.05	70.69	64.97	**71.60**
	sem	1.05	0.78	1.13	0.17	0.48	0.74	1.08

Table 6: Per-task single-trial precision (%) from cross-validation on unseen subjects (test data)

	name	CSP	CE(t)	CE	CL	AE	C-AE	SLP-AE	CSLP-AE
all	mean	45.96	57.36	56.79	57.32	39.47	58.35	58.63	**59.03**
	sem	0.13	0.11	0.19	0.35	0.25	0.23	0.21	0.27
ERN/Incorrect	mean	13.52	28.70	27.28	25.85	14.83	27.95	30.42	**32.36**
	sem	0.12	0.71	0.47	0.39	0.14	0.50	0.81	0.82
ERN/Correct	mean	73.19	96.13	95.40	96.85	57.62	97.09	96.61	**97.29**
	sem	0.54	0.29	0.17	0.27	0.46	0.08	0.33	0.14
LRP/Contralateral	mean	22.55	42.35	38.39	40.51	22.15	**43.08**	37.24	42.29
	sem	0.74	0.35	0.89	0.59	0.37	0.33	0.40	1.42
LRP/Ipsilateral	mean	23.00	43.71	41.05	45.05	22.74	45.84	41.84	**46.67**
	sem	0.65	1.13	0.59	0.78	0.49	1.17	0.55	0.72
MMN/Deviants	mean	21.40	24.24	24.22	23.85	20.32	24.12	**25.81**	24.42
	sem	0.25	0.07	0.27	0.19	0.39	0.41	0.15	0.40
MMN/Standards	mean	61.78	68.42	68.68	66.88	56.59	67.82	**70.50**	68.31
	sem	0.47	0.24	0.26	0.41	0.62	0.35	0.35	0.44
N2pc/Contralateral	mean	24.47	29.32	29.63	30.79	19.29	33.88	33.63	**35.40**
	sem	0.10	1.11	0.89	1.54	0.42	0.45	0.87	1.17
N2pc/Ipsilateral	mean	25.23	25.75	24.98	28.49	15.77	**32.52**	29.58	32.00
	sem	0.37	0.39	0.34	0.82	0.29	0.77	0.83	0.91
N400/Unrelated	mean	15.03	20.96	20.88	23.71	14.43	26.19	24.13	**27.46**
	sem	0.28	0.75	0.62	0.93	0.52	0.65	0.78	1.30
N400/Related	mean	16.56	23.77	21.76	25.67	13.23	26.32	24.70	**27.37**
	sem	0.54	0.34	0.49	0.95	0.52	0.63	0.39	0.61
P3/Rare	mean	11.15	16.56	16.31	16.79	8.87	18.10	18.80	**18.85**
	sem	0.14	0.36	0.69	0.62	0.47	0.53	0.71	0.56
P3/Frequent	mean	36.00	50.34	49.67	51.54	26.90	52.11	52.24	**54.97**
	sem	0.47	1.03	0.81	1.55	0.87	1.35	0.50	1.48

Table 7: Per-subject single-trial recall (%) from cross-validation on unseen subjects (test data)

	name	CE(t)	CE	CL	AE	C-AE	SLP-AE	CSLP-AE
all	mean	42.79	79.32	78.90	60.38	79.38	74.58	**80.32**
	sem	0.69	0.39	0.51	0.26	0.51	0.93	0.32
S5	mean	54.35	80.70	81.47	68.47	80.30	77.97	**83.50**
	sem	0.51	1.17	0.57	0.34	1.21	0.79	1.48
S14	mean	38.43	87.41	88.05	59.75	**90.26**	86.48	88.49
	sem	1.30	1.19	0.71	1.11	0.84	3.27	0.80
S15	mean	37.70	75.03	78.68	60.67	**78.97**	70.61	78.70
	sem	0.79	0.65	1.40	0.85	2.28	1.34	0.75
S20	mean	37.43	68.04	**68.79**	48.73	67.11	65.49	68.60
	sem	0.84	0.89	0.57	0.62	1.46	1.07	1.07
S22	mean	53.30	79.83	76.76	71.60	78.99	**85.05**	80.13
	sem	0.63	0.51	1.20	0.85	1.11	1.04	0.72
S23	mean	49.93	80.38	78.59	66.23	78.51	76.82	**81.37**
	sem	1.25	1.43	0.57	0.37	1.65	0.46	0.65
S26	mean	40.27	94.58	94.79	63.93	94.66	74.98	**95.26**
	sem	0.71	0.65	0.25	0.40	0.32	1.14	0.31
S29	mean	31.03	**68.69**	64.06	43.92	66.28	59.40	66.54
	sem	0.86	1.18	1.86	0.45	1.06	1.30	0.87

Table 8: Per-task single-trial recall (%) from cross-validation on unseen subjects (test data)

	name	CSP	CE(t)	CE	CL	AE	C-AE	SLP-AE	CSLP-AE
all	mean	34.31	48.70	47.57	48.72	28.26	50.37	50.18	**51.38**
	sem	0.09	0.16	0.27	0.41	0.21	0.25	0.22	0.24
ERN/Incorrect	mean	56.65	82.20	81.45	76.75	52.59	78.54	**83.07**	81.10
	sem	0.51	1.25	1.21	1.10	0.89	0.93	0.85	0.75
ERN/Correct	mean	50.47	82.87	81.27	81.69	34.00	82.92	83.10	**85.79**
	sem	0.45	0.66	0.51	0.48	0.79	0.35	0.66	0.48
LRP/Contralateral	mean	31.05	**51.90**	47.97	45.72	34.45	47.74	45.55	47.77
	sem	1.01	0.69	1.07	0.80	0.63	1.35	0.82	1.03
LRP/Ipsilateral	mean	31.22	50.40	49.39	47.82	33.31	49.55	49.47	**51.20**
	sem	0.60	1.27	0.60	0.42	0.76	1.04	0.48	0.81
MMN/Deviants	mean	27.11	39.85	38.81	42.96	24.21	43.16	42.66	**43.57**
	sem	0.35	0.35	0.71	0.42	0.32	1.05	0.65	0.60
MMN/Standards	mean	27.99	38.57	37.21	40.23	23.97	**42.37**	41.63	41.71
	sem	0.43	0.32	0.55	0.88	0.61	0.88	0.79	0.24
N2pc/Contralateral	mean	34.38	35.88	36.83	36.21	32.58	36.56	**40.28**	39.96
	sem	0.28	1.62	0.81	1.85	0.66	0.48	0.65	1.77
N2pc/Ipsilateral	mean	40.11	33.97	32.74	36.47	27.73	**41.22**	37.44	39.42
	sem	0.87	0.72	0.66	1.69	0.29	1.06	1.11	1.54
N400/Unrelated	mean	26.71	31.25	32.60	31.77	27.72	35.26	33.56	**36.39**
	sem	0.35	1.38	1.24	0.98	0.71	1.07	0.69	1.32
N400/Related	mean	32.52	32.24	32.25	31.85	25.89	29.81	**34.09**	32.43
	sem	0.65	0.83	1.21	0.91	0.91	0.66	1.03	0.43
P3/Rare	mean	35.33	38.09	38.51	38.17	31.66	39.35	**43.11**	39.83
	sem	0.59	1.19	1.14	1.53	1.86	1.92	1.12	0.98
P3/Frequent	mean	29.09	34.98	34.49	32.52	23.28	34.82	34.38	**38.06**
	sem	0.55	1.12	0.46	1.12	0.97	0.86	0.34	1.25

Table 9: Per-subject zero-shot (S.s., S.t.) ERP conversion error ($10^{-11}\mathrm{V}^2$) on the test data

	name	AE	C-AE	SLP-AE	CSLP-AE
all	mean	**3.54**	4.28	3.82	4.21
	sem	0.14	0.06	0.04	0.14
S5	mean	**9.04**	12.29	10.13	10.48
	sem	0.34	0.59	0.17	0.61
S14	mean	**0.77**	1.11	1.17	1.13
	sem	0.01	0.03	0.05	0.02
S15	mean	**0.36**	0.57	0.56	0.65
	sem	0.01	0.01	0.01	0.02
S20	mean	**1.11**	1.71	1.30	1.51
	sem	0.05	0.09	0.03	0.05
S22	mean	**0.62**	0.78	0.84	0.86
	sem	0.02	0.02	0.01	0.02
S23	mean	13.52	13.99	**13.35**	15.31
	sem	0.66	0.76	0.23	0.72
S26	mean	**1.51**	1.98	1.56	1.92
	sem	0.06	0.12	0.06	0.08
S29	mean	**1.41**	1.80	1.65	1.81
	sem	0.02	0.03	0.05	0.01

Table 10: Paradigm zero-shot (S.s., S.t.) ERP conversion error ($10^{-11}\mathrm{V}^2$) on the test data

	name	AE	C-AE	SLP-AE	CSLP-AE
all	mean	**3.54**	4.28	3.82	4.21
	sem	0.14	0.06	0.04	0.14
ERN	mean	**6.39**	7.31	6.73	7.74
	sem	0.27	0.21	0.13	0.15
LRP	mean	**1.13**	1.50	1.33	1.70
	sem	0.07	0.05	0.02	0.06
MMN	mean	**0.81**	1.10	0.95	0.99
	sem	0.05	0.04	0.01	0.02
N2pc	mean	**1.24**	1.73	1.71	1.95
	sem	0.03	0.04	0.04	0.01
N400	mean	**7.51**	8.70	7.81	7.96
	sem	0.26	0.18	0.09	0.38
P3	mean	**4.17**	5.34	4.37	4.90
	sem	0.17	0.18	0.06	0.30

Table 11: Per-subject zero-shot (S.s., D.t.) ERP conversion error ($10^{-11}\mathrm{V}^2$) on the test data

	name	AE	C-AE	SLP-AE	CSLP-AE
all	mean	11.20	11.33	6.12	**5.80**
	sem	0.35	0.53	0.10	0.17
S5	mean	23.20	24.23	14.26	**13.18**
	sem	0.50	1.48	0.12	0.70
S14	mean	1.48	1.87	1.51	**1.25**
	sem	0.03	0.08	0.06	0.03
S15	mean	1.19	1.69	1.01	**0.93**
	sem	0.06	0.07	0.03	0.05
S20	mean	4.21	5.08	3.53	**2.90**
	sem	0.21	0.35	0.12	0.14
S22	mean	1.48	1.90	1.22	**1.18**
	sem	0.10	0.08	0.04	0.03
S23	mean	49.70	46.55	23.21	**22.98**
	sem	1.80	2.46	0.61	0.93
S26	mean	6.35	6.82	2.20	**2.00**
	sem	0.28	0.31	0.07	0.08
S29	mean	2.01	2.52	2.04	**1.95**
	sem	0.02	0.09	0.04	0.06

Table 12: Paradigm zero-shot (S.s., D.t.) ERP conversion error ($10^{-11}\mathrm{V}^2$) on the test data

	name	AE	C-AE	SLP-AE	CSLP-AE
all	mean	11.20	11.33	6.12	**5.80**
	sem	0.35	0.53	0.10	0.17
ERN	mean	24.56	23.06	**9.98**	10.04
	sem	0.95	1.21	0.25	0.27
LRP	mean	5.89	5.95	**2.74**	2.97
	sem	0.28	0.39	0.09	0.11
MMN	mean	**0.80**	1.13	1.05	0.95
	sem	0.04	0.04	0.03	0.03
N2pc	mean	2.38	3.14	2.49	**2.15**
	sem	0.05	0.08	0.05	0.05
N400	mean	19.03	19.99	13.00	**11.64**
	sem	0.50	0.80	0.20	0.40
P3	mean	14.56	14.74	7.48	**7.03**
	sem	0.41	0.87	0.30	0.35

Table 13: Per-subject zero-shot (D.s., S.t.) ERP conversion error ($10^{-11}\mathrm{V}^2$) on the test data

	name	AE	C-AE	SLP-AE	CSLP-AE
all	mean	10.74	10.64	**5.02**	6.65
	sem	0.54	0.34	0.08	0.25
S5	mean	21.99	23.75	**12.94**	14.43
	sem	1.28	0.84	0.23	0.91
S14	mean	1.71	1.77	**1.23**	1.37
	sem	0.08	0.07	0.03	0.05
S15	mean	1.09	0.95	**0.58**	0.66
	sem	0.05	0.04	0.01	0.01
S20	mean	3.57	3.78	**1.64**	3.22
	sem	0.22	0.19	0.08	0.07
S22	mean	0.89	1.00	**0.82**	0.93
	sem	0.03	0.04	0.01	0.02
S23	mean	45.01	42.18	**18.22**	27.47
	sem	2.21	1.92	0.43	1.03
S26	mean	8.84	8.88	2.62	**2.35**
	sem	0.46	0.37	0.08	0.11
S29	mean	2.78	2.82	**2.11**	2.74
	sem	0.08	0.08	0.04	0.02

Table 14: Paradigm zero-shot (D.s., S.t.) ERP conversion error ($10^{-11}\mathrm{V}^2$) on the test data

	name	AE	C-AE	SLP-AE	CSLP-AE
all	mean	10.74	10.64	**5.02**	6.65
	sem	0.54	0.34	0.08	0.25
ERN	mean	23.30	22.91	**8.09**	9.13
	sem	1.26	1.06	0.19	0.43
LRP	mean	4.83	5.14	**2.10**	3.26
	sem	0.32	0.34	0.06	0.10
MMN	mean	2.91	2.28	**1.20**	2.13
	sem	0.17	0.16	0.01	0.14
N2pc	mean	3.12	3.44	**2.68**	3.12
	sem	0.18	0.18	0.06	0.08
N400	mean	17.18	17.43	**10.15**	12.75
	sem	0.74	0.23	0.15	0.52
P3	mean	13.07	12.64	**5.90**	9.48
	sem	0.60	0.33	0.12	0.58

Table 15: Per-subject zero-shot (D.s., D.t.) ERP conversion error ($10^{-11}\mathrm{V}^2$) on the test data

	name	AE	C-AE	SLP-AE	CSLP-AE
all	mean	20.82	20.28	**19.92**	20.06
	sem	0.08	0.08	0.11	0.11
S5	mean	42.07	40.50	**35.94**	39.50
	sem	0.54	0.39	0.17	0.58
S14	mean	**2.68**	2.98	3.74	3.12
	sem	0.06	0.13	0.11	0.05
S15	mean	**2.33**	2.52	4.28	2.74
	sem	0.03	0.07	0.09	0.09
S20	mean	**8.32**	9.06	11.67	9.93
	sem	0.09	0.20	0.07	0.24
S22	mean	**2.33**	2.63	4.32	2.73
	sem	0.10	0.12	0.12	0.07
S23	mean	89.47	85.50	**81.26**	82.72
	sem	0.38	0.57	0.67	0.50
S26	mean	15.72	15.29	**13.79**	15.44
	sem	0.11	0.07	0.03	0.32
S29	mean	**3.64**	3.74	4.37	4.29
	sem	0.04	0.13	0.04	0.08

Table 16: Paradigm zero-shot (D.s., D.t.) ERP conversion error ($10^{-11}V^2$) on the test data

	name	AE	C-AE	SLP-AE	CSLP-AE
all	mean	20.82	20.28	**19.92**	20.06
	sem	0.08	0.08	0.11	0.11
ERN	mean	46.84	45.46	44.67	**44.03**
	sem	0.22	0.26	0.33	0.54
LRP	mean	11.50	**11.43**	11.82	11.47
	sem	0.08	0.07	0.12	0.09
MMN	mean	2.76	2.17	**1.65**	4.07
	sem	0.17	0.15	0.06	0.25
N2pc	mean	**4.97**	5.23	5.53	5.79
	sem	0.10	0.13	0.03	0.11
N400	mean	31.56	31.07	30.26	**29.74**
	sem	0.14	0.29	0.13	0.16
P3	mean	27.28	26.31	25.62	**25.26**
	sem	0.11	0.29	0.15	0.09

F.2 Confusion matrices

In this material we provide confusion matrices as the summary of the predictions of each fold in the cross-validation, i.e. through the cross-validation each sample in the test-set is predicted once (by leaving it out in one of the folds), the confusion matrices below are the sum over all these prediction and normalized by true instances of the label (row-wise normalization). In more technical terms: we train the classifier on the train-split of a fold and then predict on the test-split of a fold, these predictions on each fold are then concatenated and the confusion matrix is simply created from the concatenated predictions and normalized row-wise.

We did not train any classifiers to directly classify paradigm. Instead we repurposed the task-predictions such that ERP-components from the same paradigm were relabelled to a paradigm label, i.e. ERN/Incorrect and ERN/Correct predictions and truth labels were all relabelled to just ERN, etc. This is equivalent to simply summing over the task confusion matrix with 2×2-block patches. Note: the classifier was not directly trained on paradigm-classification, and therefore, this might not be the most useful or accurate representation of the paradigm content of the latent since most of the misclassifications were in-paradigm, i.e. the classifier had a hard time distinguishing between in-paradigm ERP components. However, it is clear that task (ERP-component) classification is much harder than paradigm classification.

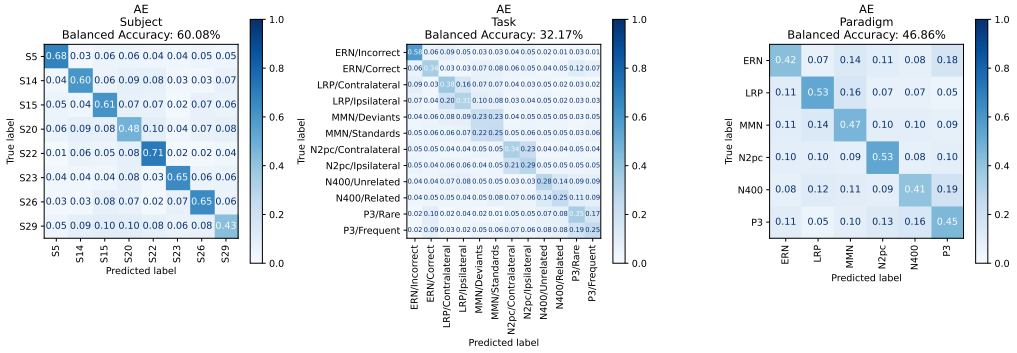


Figure 9: Confusion matrices on the cross-validation of the test set for the AE model. Left: Subject classification. Middle: Task classification. Right: Repurposed task- to paradigm-classification

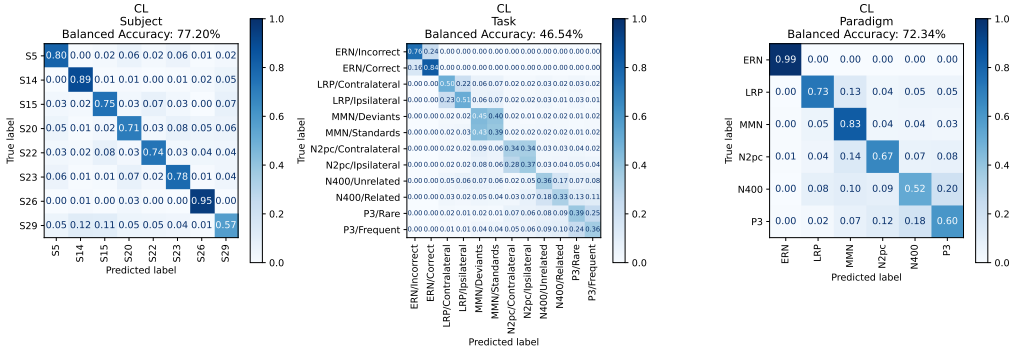


Figure 10: Confusion matrices on the cross-validation of the test set for the CL model. Left: Subject classification. Middle: Task classification. Right: Repurposed task- to paradigm-classification

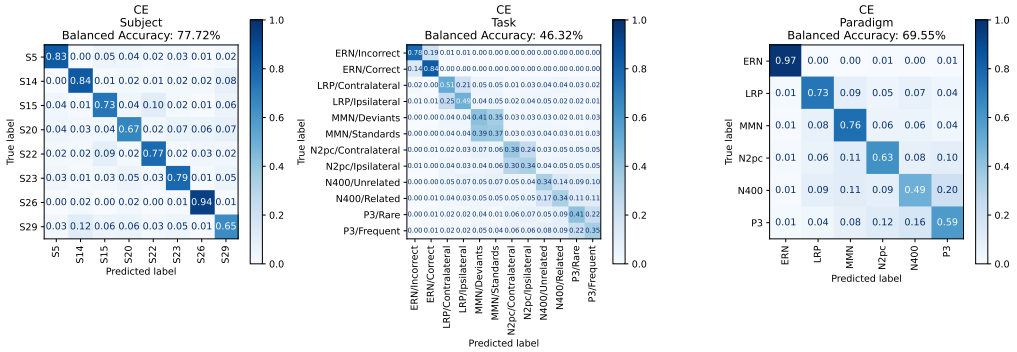


Figure 11: Confusion matrices on the cross-validation of the test set for the CE model. Left: Subject classification. Middle: Task classification. Right: Repurposed task- to paradigm-classification

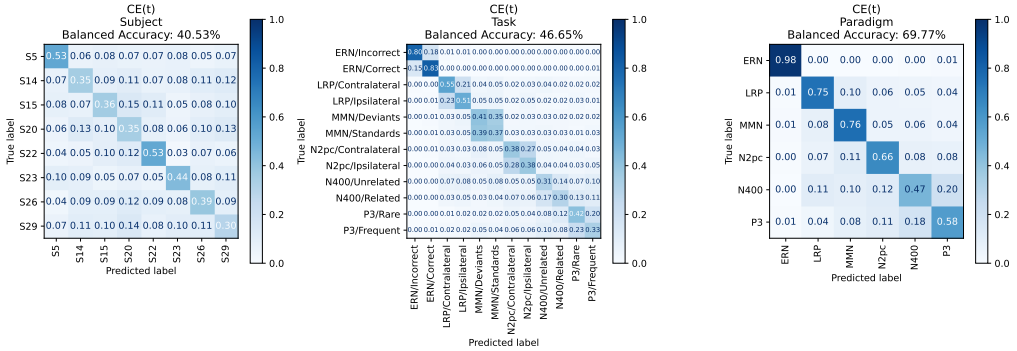


Figure 12: Confusion matrices on the cross-validation of the test set for the CE(t) model. Left: Subject classification. Middle: Task classification. Right: Repurposed task- to paradigm-classification

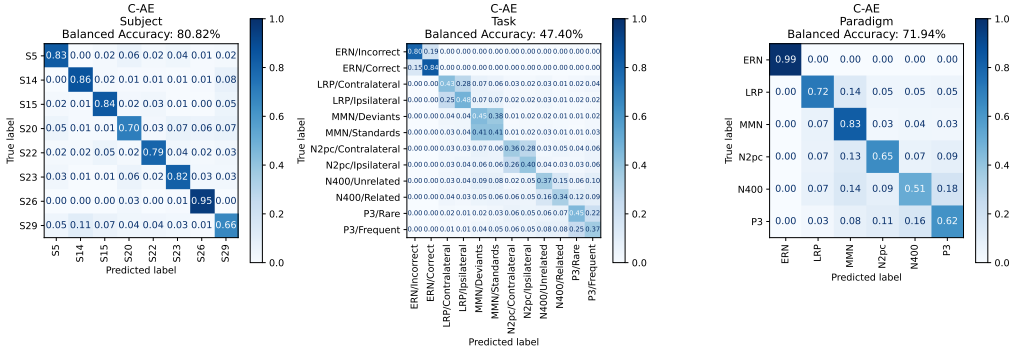


Figure 13: Confusion matrices on the cross-validation of the test set for the C-AE model. Left: Subject classification. Middle: Task classification. Right: Repurposed task- to paradigm-classification

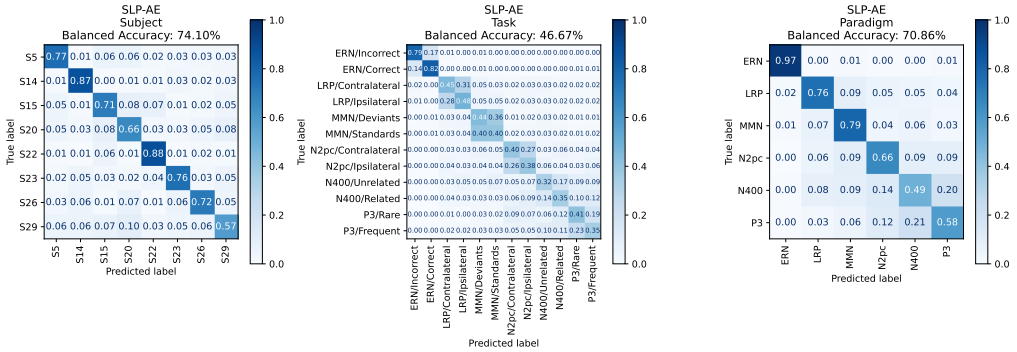


Figure 14: Confusion matrices on the cross-validation of the test set for the SLP-AE model. Left: Subject classification. Middle: Task classification. Right: Repurposed task- to paradigm-classification

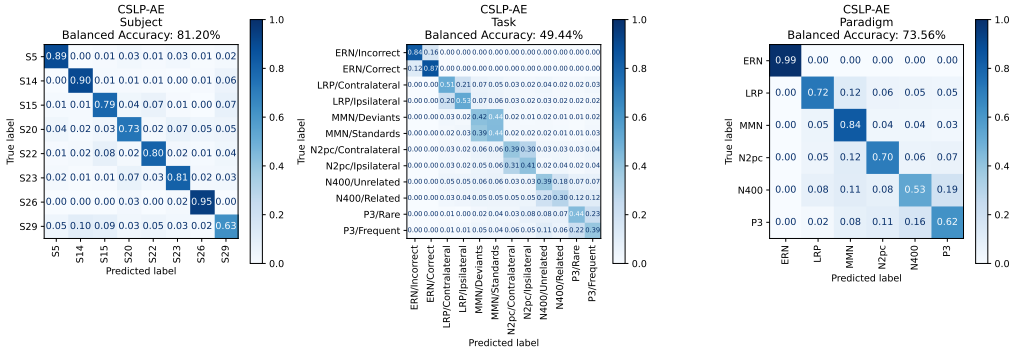


Figure 15: Confusion matrices on the cross-validation of the test set for the CSLP-AE model. Left: Subject classification. Middle: Task classification. Right: Repurposed task- to paradigm-classification

F.3 Additional t-SNE plots

In this material we provide t-SNE plots [1, 14, 20] on the training and test set. For each set (training or test) we run t-SNE on each latent space (subject or task latent space) and then we color each of these either by subject or task.

We end up with training set subject latent space colored by subjects, training set subject latent space colored by tasks, training set task latent space colored by tasks, and training set task latent space colored by subjects. These are available in Figure 16. Furthermore, we provide test set subject latent space colored by subjects, test set subject latent space colored by tasks, test set task latent space colored by tasks, and test set task latent space colored by subjects. These are available in Figure 17. All t-SNE applications used `perplexity=30` and the Barnes-hut approximation using the `scikit-learn` implementation[12] [17]. All initializations used `random_state=1968125571`. We used the default parameters of the `scikit-learn` implementation since these were found to give good clustering on the evaluation (validation) set.

[12]Documentation here: `https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html`

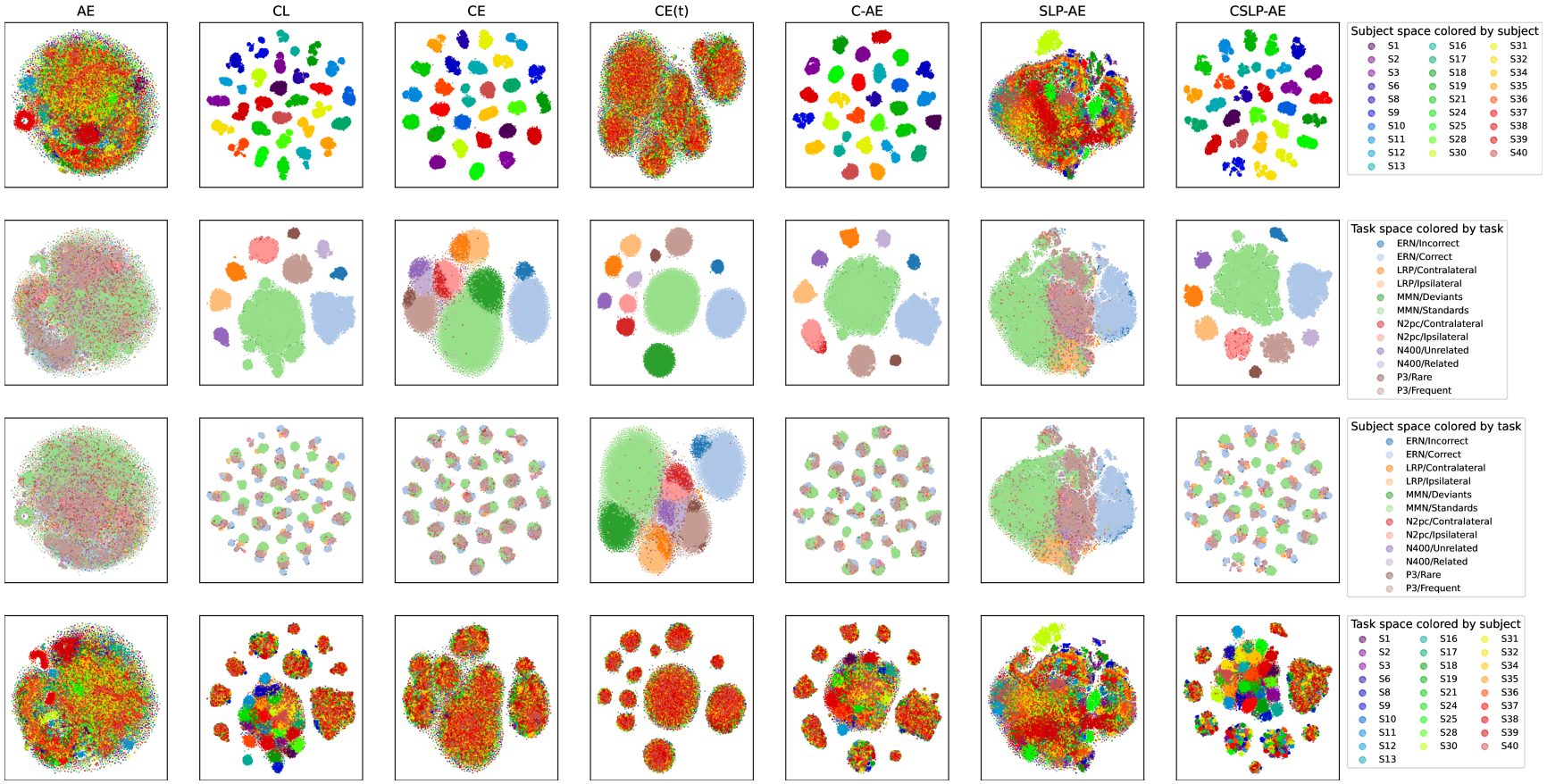


Figure 16: *t*-SNE plots of the subject and task latent space of each model on the training data colored by both subject and task. Legend on the right corresponds to each row. Note that the subject latent space for the CE(t) model is not trained (it is part of the model architecture, but no losses backpropagate through it)

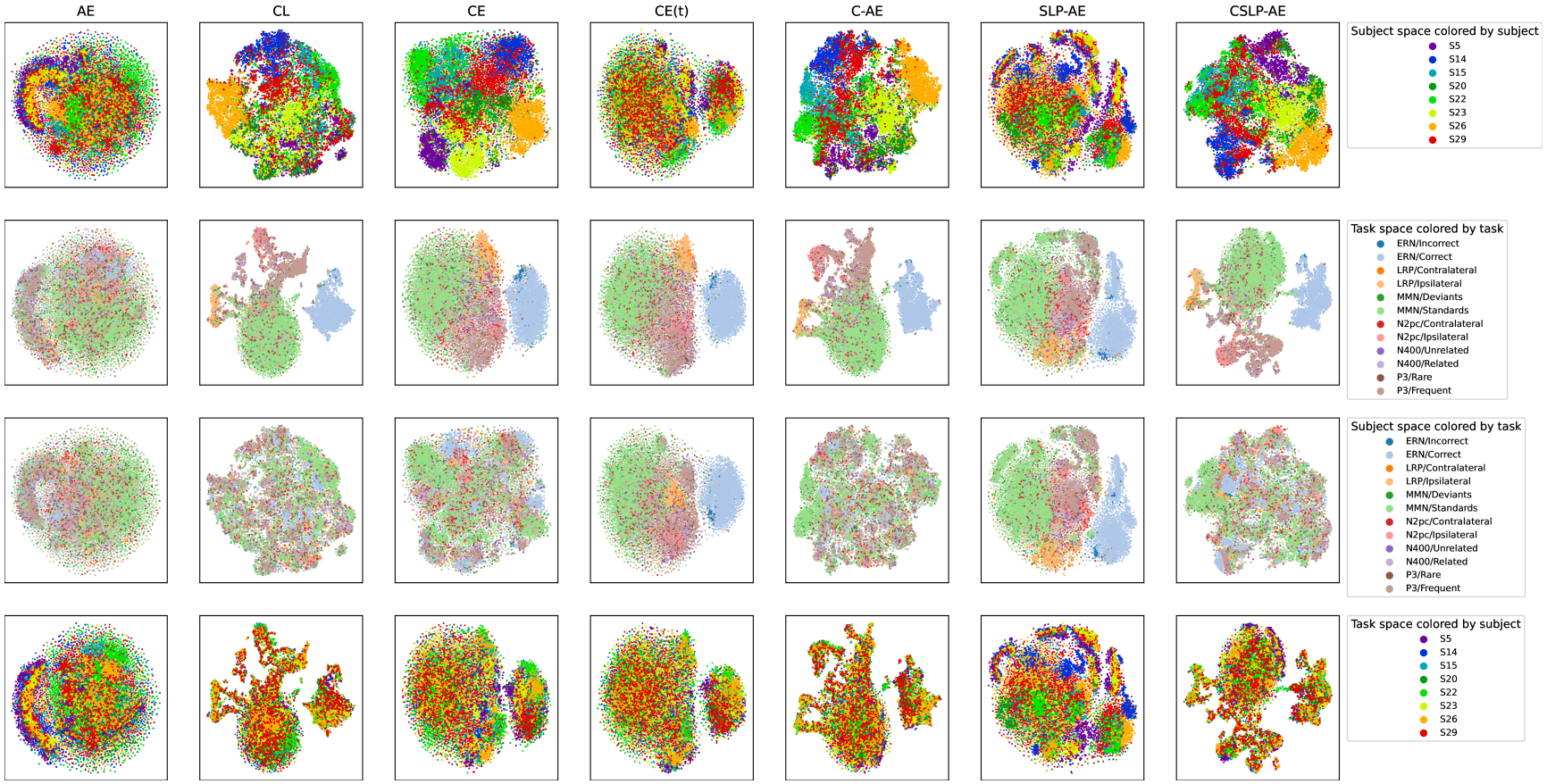


Figure 17: *t*-SNE plots of the subject and task latent space of each model on the test data colored by both subject and task. Legend on the right corresponds to each row. Note that the subject latent space for the CE(t) model is not trained (it is part of the model architecture, but no losses backpropagate through it)

G Other classifiers

In Table 17 we provide cross-validation results using multiple classifiers, here XGBoost[13] [5], KNN[14] [6], and ExtraTrees [15] [7]. The K-nearest neighbor classifier and ExtraTrees classifier were trained using the `scikit-learn` [17] package.

For K-nearest neighbor classification we used $K = 15$ and cosine distance as the distance metric to match the similarity metric used in the contrastive loss.

For ExtraTrees we used `n_estimators=200`, `max_features=0.2`, `max_depth=60`, and `class_weight="balanced"`. These were chosen by using the Bayesian optimization sweep implementation of the Weights & Biases MLOps platform[16] [3]. The hyperparameter search was done over the evaluation (validation) set.

Table 17: Results using other classifiers on the test data. Best classifier and model is marked in bold.

	XGBoost [5]		KNN [6]		ExtraTrees [7]	
	S.acc%	T.acc%	S.acc%	T.acc%	S.acc%	T.acc%
CSLP-AE	**80.32 ± 0.28**	**48.10 ± 0.32**	72.65 ± 0.69	44.66 ± 0.55	72.18 ± 0.35	46.55 ± 0.37
SLP-AE	74.63 ± 0.74	47.00 ± 0.32	59.08 ± 0.16	45.84 ± 0.03	60.30 ± 1.00	45.71 ± 0.17
C-AE	79.42 ± 0.48	46.59 ± 0.23	71.21 ± 0.63	43.85 ± 0.29	72.09 ± 0.53	45.58 ± 0.25
AE	60.68 ± 0.16	31.43 ± 0.28	60.76 ± 0.38	29.86 ± 0.35	41.53 ± 0.22	26.24 ± 0.24
CL	78.82 ± 0.46	45.36 ± 0.37	71.53 ± 0.59	43.75 ± 0.35	73.05 ± 0.64	44.88 ± 0.25
CE	79.25 ± 0.37	45.22 ± 0.23	75.18 ± 0.48	43.39 ± 0.82	73.48 ± 0.46	44.76 ± 0.22
CE(t)	-	45.80 ± 0.24	-	43.00 ± 0.60	-	45.38 ± 0.27

[13]Documentation: `https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBClassifier`

[14]Documentation: `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`

[15]Documentation: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html`

[16]Main site: `https://wandb.ai/site` Bayesian Optimization sweep: `https://docs.wandb.ai/guides/sweeps`

H Conversion examples

In this material we provide examples from ten random pairs of target subject and target paradigms realized using the C-AE, SLP-AE, and CSLP-AE models over all conversion method schemes. Since the subjects are different in the training set and test set, five random subjects from the training set and five random subjects from the test set were chosen with corresponding random paradigms. These are all sampled using $N = 2000$ samples from the conversion scheme.

H.1 Conversion examples for train data (seen subjects)

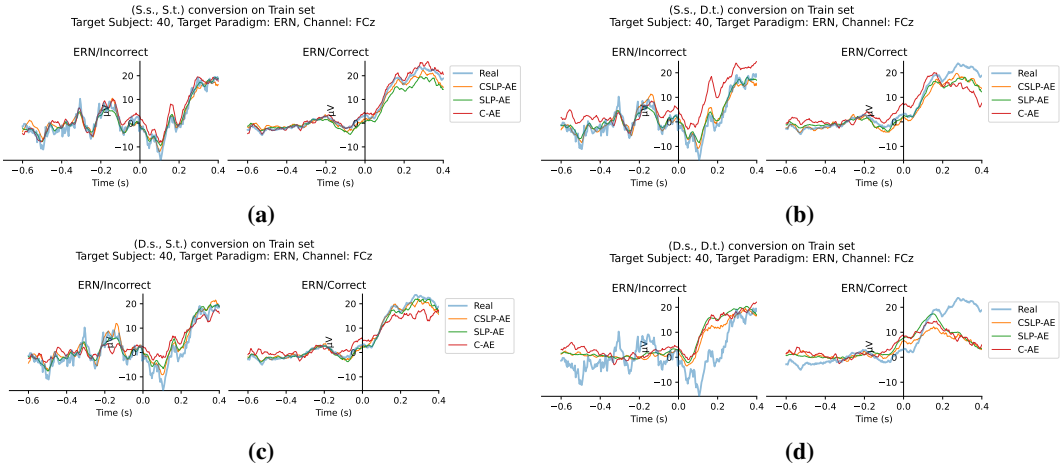


Figure 18: Example ERPs for Train set, Target Subject: 40, Target Paradigm: ERN, Channel: FCz.

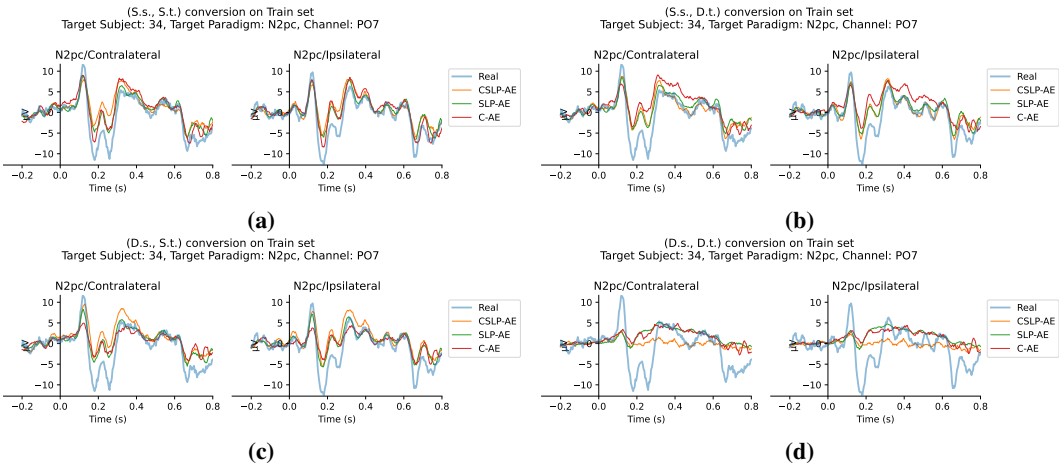


Figure 19: Example ERPs for Train set, Target Subject: 34, Target Paradigm: N2pc, Channel: PO7.

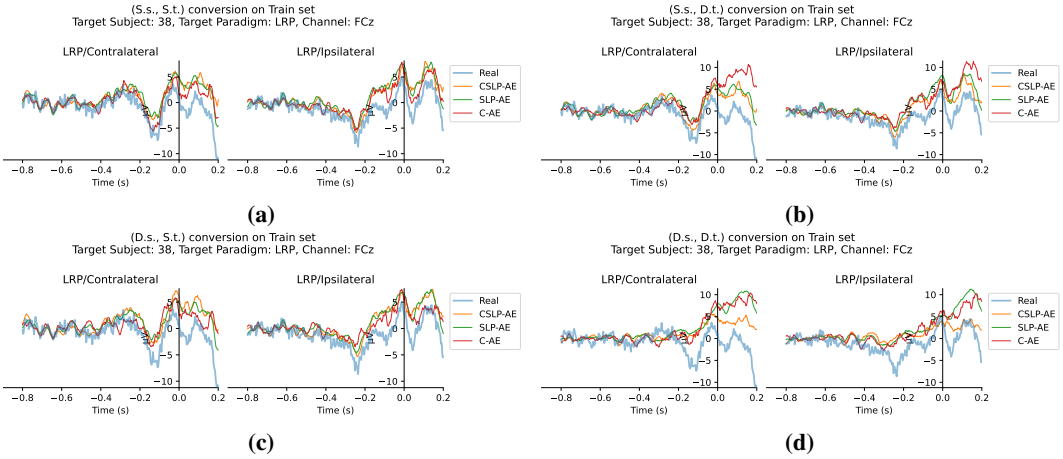


Figure 20: Example ERPs for Train set, Target Subject: 38, Target Paradigm: LRP, Channel: FCz.

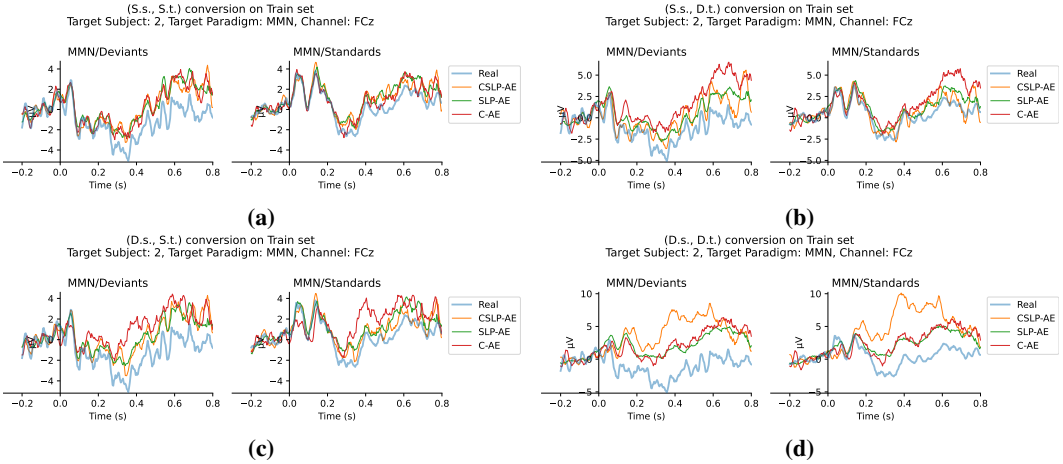


Figure 21: Example ERPs for Train set, Target Subject: 2, Target Paradigm: MMN, Channel: FCz.

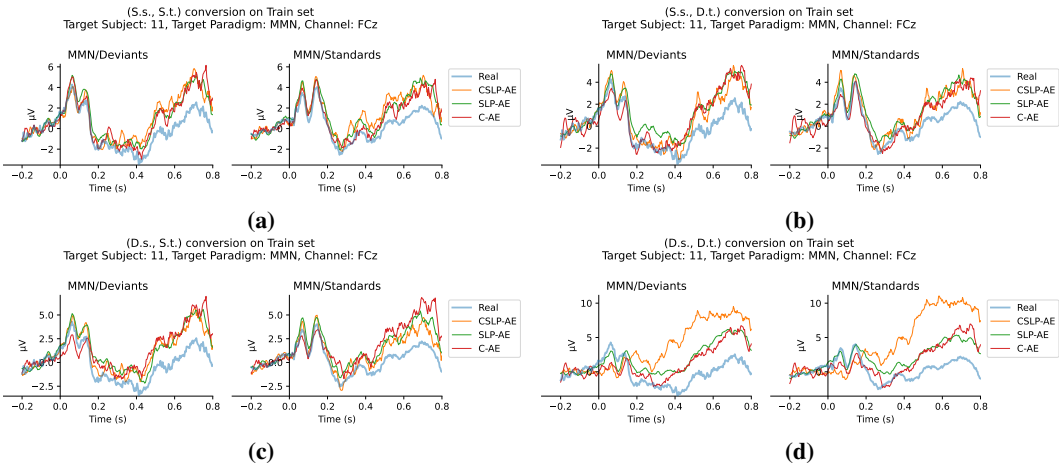


Figure 22: Example ERPs for Train set, Target Subject: 11, Target Paradigm: MMN, Channel: FCz.

H.2 Conversion examples for test data (unseen subjects)

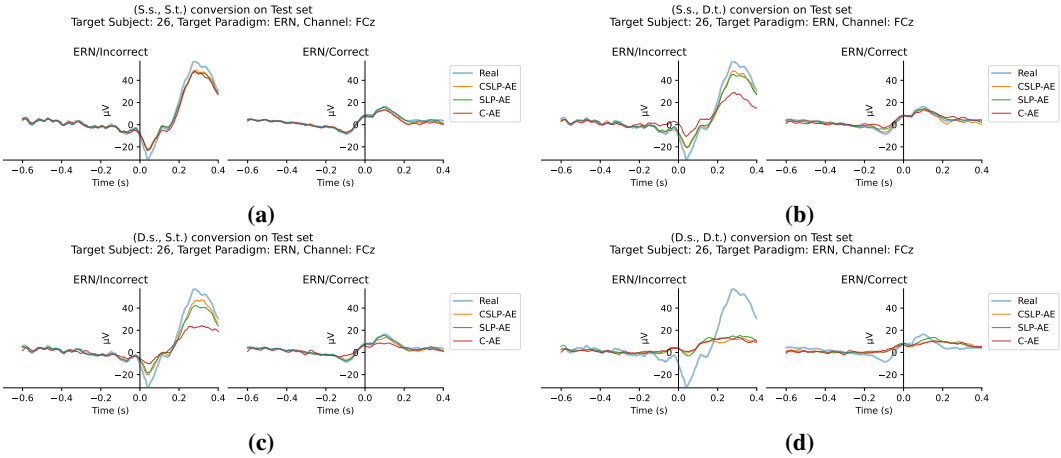


Figure 23: Example ERPs for Test set, Target Subject: 26, Target Paradigm: ERN, Channel: FCz.

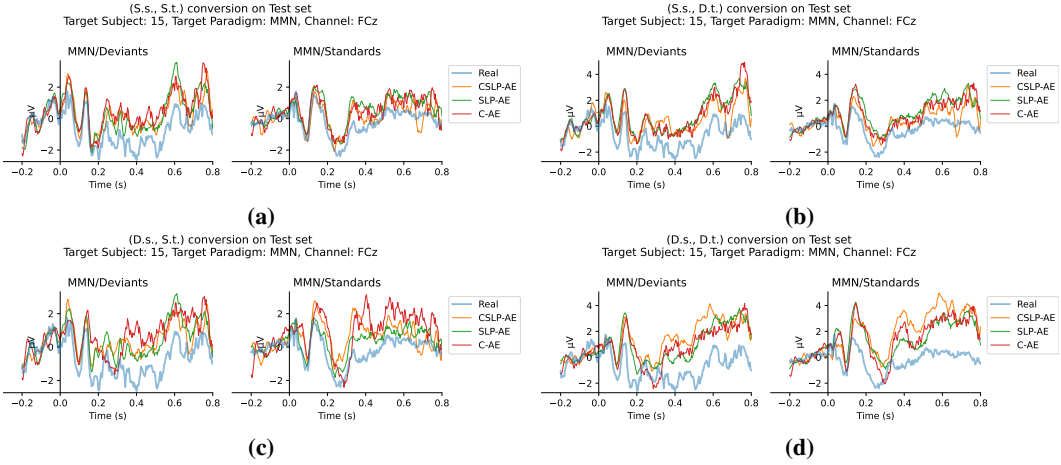


Figure 24: Example ERPs for Test set, Target Subject: 15, Target Paradigm: MMN, Channel: FCz.

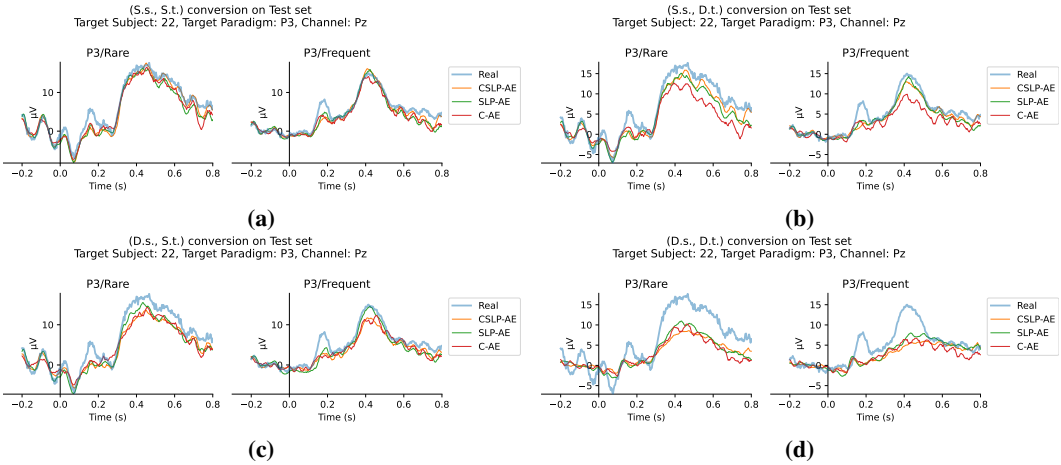


Figure 25: Example ERPs for Test set, Target Subject: 22, Target Paradigm: P3, Channel: Pz.

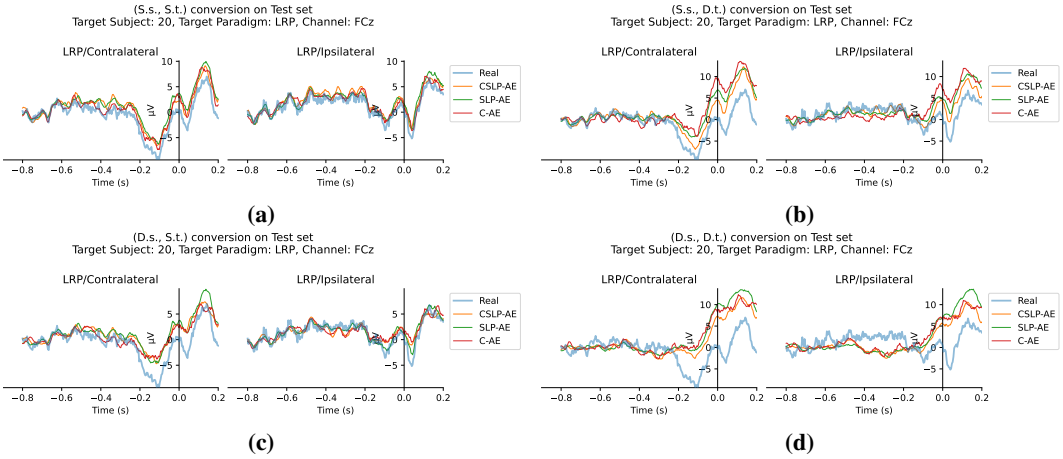


Figure 26: Example ERPs for Test set, Target Subject: 20, Target Paradigm: LRP, Channel: FCz.

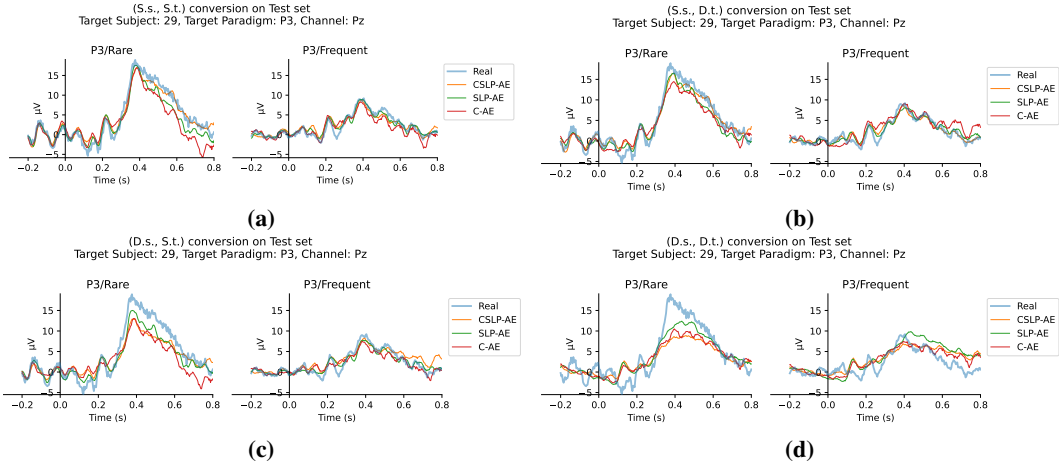


Figure 27: Example ERPs for Test set, Target Subject: 29, Target Paradigm: P3, Channel: Pz.

References

[1] A. C. Belkina, C. O. Ciccolella, R. Anno, R. Halpert, J. Spidlen, and J. E. Snyder-Cappione. Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. *Nature communications*, 10(1):5415, 2019.

[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[3] L. Biewald. Experiment tracking with weights and biases, 2020. URL `https://www.wandb.com/`. Software available from wandb.com.

[4] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K.-R. Muller. Optimizing spatial filters for robust eeg single-trial analysis. *IEEE Signal processing magazine*, 25(1):41–56, 2007.

[5] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[6] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[7] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63:3–42, 2006.

[8] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[9] A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, and M. S. Hämäläinen. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7(267):1–13, 2013. doi: 10.3389/fnins.2013.00267.

[10] M. Grosse-Wentrup and M. Buss. Multiclass common spatial patterns and information theoretic feature extraction. *IEEE transactions on Biomedical Engineering*, 55(8):1991–2000, 2008.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[12] E. S. Kappenman, J. L. Farrens, W. Zhang, A. X. Stewart, and S. J. Luck. Erp core: An open resource for human event-related potential research. *NeuroImage*, 225:117465, 2021.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[14] D. Kobak and P. Berens. The art of using t-sne for single-cell transcriptomics. *Nature communications*, 10(1):5416, 2019.

[15] Z. J. Koles, M. S. Lazar, and S. Z. Zhou. Spatial patterns underlying population differences in the background eeg. *Brain topography*, 2(4):275–284, 1990.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[19] Y. Roy, H. Banville, I. Albuquerque, A. Gramfort, T. H. Falk, and J. Faubert. Deep learning-based electroencephalography analysis: a systematic review. *Journal of neural engineering*, 16 (5):051001, 2019.

[20] L. Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[21] X. Wang, M. Hersche, B. Tömekce, B. Kaya, M. Magno, and L. Benini. An accurate eegnet-based motor-imagery brain–computer interface for low-power edge computing. In *2020 IEEE international symposium on medical measurements and applications (MeMeA)*, pages 1–6. IEEE, 2020.