,

Denoising Autoencoders for Biosignals

Rasmus Hjelm Rasmussen RHRA17@STUDENT.AAU.DK

Simon Anielski Barsøe Jensen SABJ17@STUDENT.AAU.DK

Abstract

This paper introduces a self-supervised framework for pre-training on EEG data. The goal is to create a model that produces good features, such that the model can be used for transfer learning. Our framework is based on a denoising autoencoder architecture. We have a model that receives an input that is augmented using token masking, and tries to reconstruct the original input. The pre-training is done using a subset of the *Temple University Hospital EEG Data Corpus (TUEG)* datasets. Our model is a transformer based model, inspired by the likes of BERT. For our results we ran benchmarks on the 3 different datasets we did fine tuning on. Here we compare to some supervised methods. The results show that our model, though not the best, is comparable to the supervised models. The benchmark proves that the model has learned transferable features.

1. Introduction

Brain-computer interfaces (BCIs) are devices that enable computers to read information from the brain, and are generally used as a method of analyzing and understanding the brain. One of the most commonly used BCIs is electroencephalography (EEG)(1). EEG is a method of measuring the electrical activity from the brain. EEG is non-evasive, and works by recording voltage potential with electrodes placed on the scalp of a subject (2). EEG is widely used due to of its ease of use and it's relatively low costs. The main advantage of EEG compared to other BCI systems is its relatively low cost and ease of use, due to its non-invasive application. Furthermore, EEG can be recorded with high temporal resolution in comparison with the temporal patterns of the brain, unlike most other BCIs(2). EEG's non-invasive nature also comes with some disadvantages. The spatial resolution of EEGs is quite limited, since only the outer layers of the brain can be recorded(3). Moreover, In comparison to the amount of neurons present in the brain, the number of electrodes is also very low, and each electrode therefore records a single signal over a large number of neurons, but produces only a single signal. The result is a signal full of artifacts and noise with a high noise-to-signal ratio. (4)

EEG data has shown to be very helpful as a medical tool. EEG has been successfully applied to classify a range of different mental disorders. Some examples of this are Epilepsy (5), Parkinson's Disease (6), Attention Deficit Hyperactivity Disorder (ADHD) (7), and Schizophrenia (8). It has also been used to help people with disabilities and movement impairments(9)(10). Traditionally, to analyze these signals, a lot of signal processing methods have been applied to extract useful features from EEG waves, including Common Spatial Pattern and PCA. (11) Once feature extraction has been done, you would use classifiers such as SVMs and Random Forests to classify the signal on using those features.

In recent years a surge in popularity of deep learning, and in this field as well. Deep learning makes it possible learn features that would previously have to be handcrafted. The promise of deep learning, is that we create end-to-end systems that learn all features, making thorough preprocessing and filtering unnecessary (12). Mainly convolutional networks (CNNs), and recurrent network (RNNs) have dominated the space(3). Recently, the newer transformer models have also been adopted for EEG decoding. Sarker et al.(13) suggests that deep learning outperforms more traditional machine learning algorithms, however this only applies above a critical number of data points. This poses a problem in the context of EEG decoding, since EEG datasets are hard to produce and are therefore often small in comparison to datasets in other domains within machine learning. The studies are also likely to have a limited amount of subjects, resulting in the data being highly biased. A review of Deep learning studies in EEG(3) shows that half the 154 studies reviewed are conducted on less than 13 subjects. As a result, the best performing models are most often those, that are trained for each of the subjects in a study, in an intra-subject fashion (3). There is a great deal of inefficiency with this method, since whenever a new subject is introduced, a lot more data must be collected and a new model must be trained. It would be preferable, if only a small amount of data would be required to calibrate a model to a new subject. Moreover, different EEG datasets are made to tackle very different problems, and as suggested by Kostas et al. (14), the variability of the signals is significant, both inter- and intra-personally, and for the same tasks at different sessions. They also suggest that supervised learning is not likely to create latent features useful for other EEG tasks.

Creating latent features has been achieved in other machine learning domains already. For vision, you can find a pre-trained model that can be fine-tuned to your particular task. Within the field of EEG research, this has not been studied to the same degree. Within Computer Vision and Natural language processing, there has been an effort to create models that produce good features that can be used for a range of different problems(15)(16). This is the problem self-supervised learning attempts to tackle.

In this paper we explore different architectures and setups of self-supervised training for EEG data, inspired by Kostas et al. (14). We attempt to show that good latent representations can be created with a smaller model compared to the BENDR model(14). The goal is to create a model that produces good features of EEG signals which can transfer to work for different tasks.We present a self-supervised framework for pre-training, which produces EEG features that can be transferred on to other datasets. We argue that with more data and computation the framework has potential given scaling of three factors. We compare our small scale model against popular supervised learning models.

2. Related works

2.1 Self-supervised Learning

Self-supervised learning (SSL) is a machine learning approach that learns from unlabeled data, similar to an unsupervised learning approach. However, the difference between these two approaches is that SSL approaches still seek to minimize the distance between an output of a network and a target, making them supervised in essence. However, in SSL, the targets are not created from hand-labeling data, but are created directly from the data itself. The target is often the actual input itself, where the model receives an augmented version of the input. Learning useful features of a dataset is the main objective of self-supervised learning. It is particularly desirable to learn from unlabeled data, since most data in the world is not labeled. Labeling would involve a considerable amount of effort.

Traditionally, self-supervised learning has been accomplished using autoencoders. There are two components to an autoencoder: an encoder and a decoder. The encoder creates a latent representation of the input, while the decoder recreates the input from that latent representation. After the autoencoder has been trained, the encoder can be used to create a useful feature representation over an input. A linear classifier can be trained on top of the encoder. There are also Denoising autoencoders that accept an input with noise applied and have to reproduce the original input. A popular type of autoencoder is a variational autoencoder, which predicts a distribution over the latent space rather than the latent vector itself.

More recently, contrastive learning techniques such as MoCo and SimCLR have shown promising results(15)(17). Contrastive learning is based on the idea of creating a representation space where similar samples are close in representation and samples that are dissimilar are far away in representation. This is achieved through the use of positive and negative samples. Positive samples are samples with close similarities to the input, and negative samples are samples with little or no similarities. By corrupting or augmenting your sample a bit, you can produce positive samples with similar contents, but not identical. How you create negative samples varies according to the approaches you employ in contrastive learning.

Work has been done exploring the efficacy of contrastive learning within the EEG domain, notably Mohsenvand et al. demonstrates that a useable feature representation is achievable with an extension of the SimCLR framework (18). They employ two different models specialized for EEG data, one of them a CNN based model, the other a Bidirectional RNN. They demonstrate that by using a pre-trained network, you can achieve better results than training from a random initialization. Further more they show that the pre-trained model is much more label efficient, and in some cases perform better when only trained on 50% of the labels compared to results using random initialization.

Another interesting article is from Joseph Y. Cheng et al., wherein they do contrastive learning, but make use of the subject information as part of the pre-training task (19). They do this by trying to predict the subject from the learned features. They demonstrate for smaller datasets, including the subject as part of pre-training can lessen the impact of the subject variability on learning features.

They conclude, however, that for datasets with a larger amount of subjects (over 64), the subject aware part of pre-training become redundant.

Building on the work of contrastive learning approaches, non-contrastive learning approaches have recently shown positive results. Among them are BYOL and DINO (20)(21), which have demonstrated promising results. Different from contrastive methods, non-contrastive methods avoid using negative samples. The models would normally suffer from mode collapse as a result of this, however, these methods avoid this problem by using some tricks. Most recently, Data2vec has shown promise in creating a non-contrastive approach that works across different modalities. We are unaware of any implementation of non-contrastive learning for EEG data.

Recently there is a trend moving towards transformer based self-supervised frameworks, introduced by the paper "Attention is all you need"(22). Some evidence of this is the recent publication from Meta AI: Data2vec(23). In the paper they demonstrate how a non-constrative framework using transformers can be applied successfully to multiple modalities namely: Vision, NLP, and Audio processing.

Before this the Transformer had already become dominant throughout the field of Natural Language Processing, having been incorporated into large language models such as BERT and OpenAI's GPT models (16)(24). Transformers have also been gaining traction across other fields of research, and they are now applied across many modalities(25)(26).

Recently we are seeing transformers applied to EEG data as well, although research in this area is still in its infancy. The efficacy of transformer based architectures are, among others, demonstrated by Lee and Lee (27) and Tao et al.(28). Both these papers create an their on take on the transformer model and demonstrate its effectiveness.

As for using the transformer model in a self-supervised context, the most notable contribution is the BENDR model(14). They demonstrate both the efficacy of self-supervised training on EEG-data, along with transferability. It shows promising results, however the BENDR model is significantly larger than any other EEG we are aware of, and likewise requires a large amount of data. This brings forth the concern that fine tuning the BENDR model for specific tasks is very resource demanding.

3. Method and Experimental setup

3.1 Proposed method

We propose a simple self-supervised method for pre-training on EEG signals. The method is a BERT(16) inspired de-noising Autoencoder, which has been adapted to work for EEG signals. The proposed setup is shown in figure 3. For the pre-training task, the model is trained to predict the original EEG-input. With a de-noising autoencoder you apply noise/augmentation to the input signals and letting the model reconstruct the original input. In our case we apply the augmentation in the form of token masking as well as some EEG-specific augmentations. After the pre-training task, we save the encoder. The pre-trained encoder is then used for transferring learning to new tasks with a different dataset. This requires training a new embedding layer that fits the dimensions of the dataset, along with a new head to perform classification.

The architecture for this system consists of three primary components: the embedding layer, transformer encoder, and projection head.

To create tokens from input, we utilized the approach that the Vision Trasformer(25) uses, which involves splitting the input into non-overlapping patches. The patch has the shape of (number of channels x patch size), where the patch size determines how many time steps are included in the patch. Once the patches have been created, they are inserted into the embedding layer. With trans-

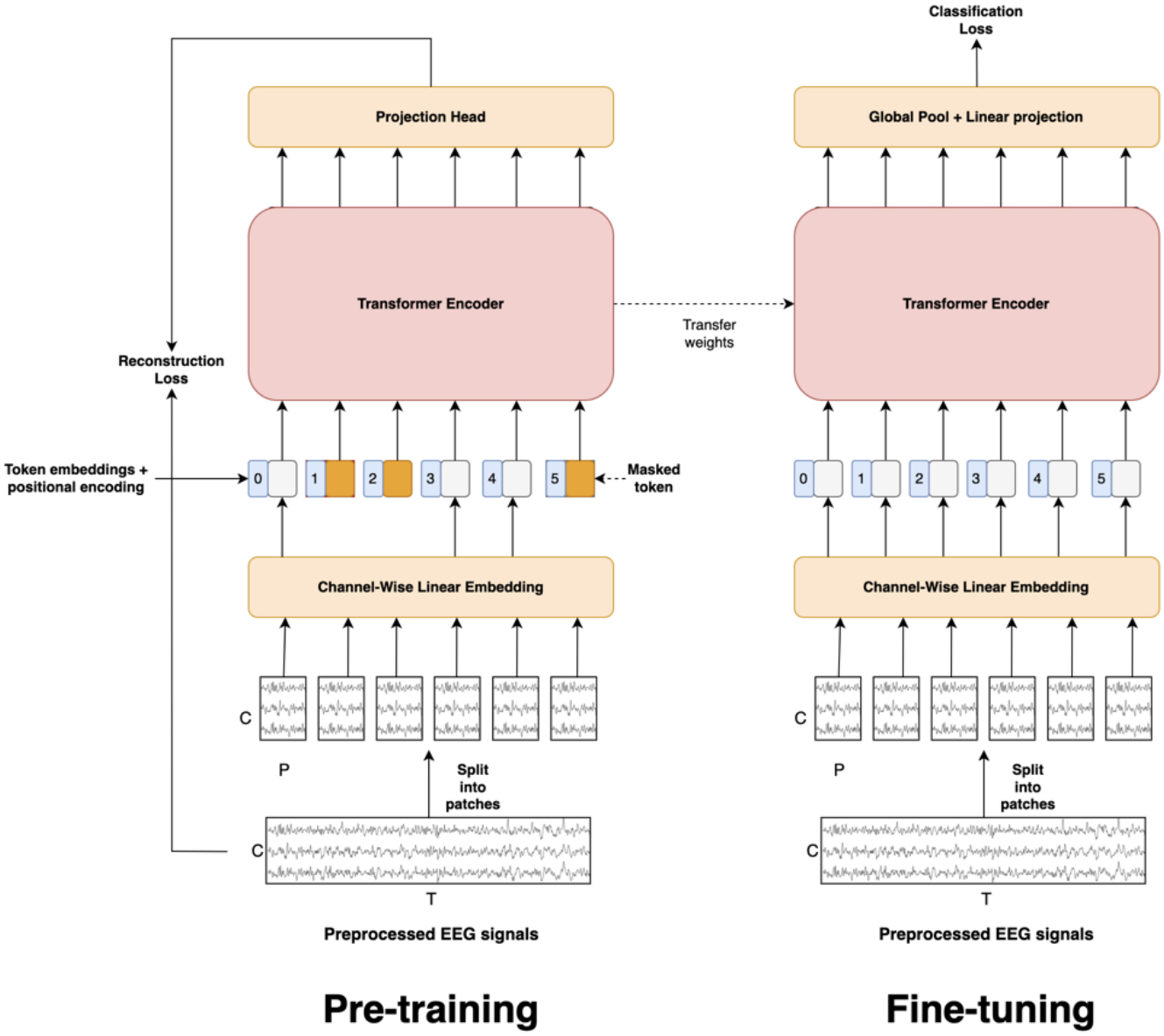

Figure 1: Architecture of our transformer model, both for pre-training (L), and transfer (R)

former models, it is common practice to tokenize the raw input and pass it through and embedding layer before passing it through the encoder(22).

The original transformer paper(22) uses an embedding that maps tokens to learned vectors. This works for domains with discrete inputs, such as NLP, but it is not applicable to domains with continuous inputs. In continuous domains such as time series data, or in more complex discrete domains, such as vision, it is commonplace to simply flatten the raw input and pass it through

a linear feedforward layer instead (25). The embedding of other types of data, such as audio, is also sometimes carried out using successive convolutional layers(29). We do not consider the dimensionality of EEG data to be high enough to justify the use of CNNs to compress the data dimension, and in our initial tests, linear embeddings proved to be more effective.

We found that, rather than flattening the patches and passing through a linear layer, it is better for fine tuning, to pass the patches through a channel-wise linear layer followed by flattening. A channel-wise linear layer, being a linear layer applied over the channels for all time steps in the patch. This simplifies the fine tuning steps, since channel-wise embedding effective functions as a learned channel mapping, and also has a lot fewer parameters and is therefore easier to pre-train. Following the embedding layer, we encode each of the tokens using a sinusoidal positional encoding (22), rather than a learned positional encoding. The benefit of sinusoidal positional encoding is that it is static and does not require retraining during transfer learning.

Our encoder resembles the default transformer encoder as presented by the "Attention is all you need" paper(22), consisting of n multihead attention layers, using a layernorm and the GELU activation.

On top of the transformer encoder we have a projection head, which servers to reconstruct the original raw EEG waves. The projection head consists of 2 consecutive blocks with a Linear layer, a layer normalization, and the GELU activation function.

3.2 Pre-training

The pretraining setup is based around a selected part of the *Temple University Hospital EEG Data Corpus (TUEG)* datasets(30). The TUEG dataset is a large assembly of EEG recordings going as far back as 2002. The dataset consists mainly of recordings of patients, whom experience different types of mental problems such as seizures, and has a binary target class on whether any abnormalities are present in a sample.

We downloaded 204GB of the TUEG data, yielding 13380 datasubsets, which was further filtered by a recording frequency of 250Hz and 30 channels, finally yielding 1490 datasubsets. This represents 833 subjects. The frequency was selected on basis of the recording frequency of the BCI dataset, and selecting 30 channels yields the highest number of datasubsets. We narrowed down the TUEG dataset based on these parameters to limit the otherwise excessive prepossessing steps required for transfer learning.

For both pre-training and fine tuning all the datasets are re-sampled to a frequency of 250hz, matching the subset of the TUEG dataset. For all datasets we use exponential moving standardization to normalize the data (31). For pre-training we use no band-pass filtering.

Our augmentation strategy involves masking out tokens similar to the way likes of BERT(16). During pre-training we employ a mix of two different masking approaches: random masking and block masking. With random masking, each token is assigned a probability 0.4 of being masked, meaning that on average 40% of the tokens will be masked out. With block masking a contiguous block of 40% of the tokens are masked. We assign a 50% probability of using either random or block masking for any given batch.

On top of masking we explore few different EEG based augmentations. We make use of the augmentation library from the braindecode library (32).

In pretraining we make use of the AdamW optimizer with a learning rate of 1e-4, with the beta values set to (0.5, 0.99). We also use a weight decay of 0.01 similar to the BERT paper(16).

3.2.1 Reconstruction loss

For the reconstruction loss we calculate a smoothed L1 distance between the predicted reconstruction and the raw EEG signal input. In testing we found that a smooth L1 loss worked better than a simple L1 or L2 loss, which are widely used for auto encoders. Figure 3.2.1 shows the smooth L1 loss.

$$l_n = \begin{cases} 0.5(x_n - y_n)^2/beta, & \text{if } |x_n - y_n| < beta \\ |x_n - y_n| - 0.5 * beta, & \text{otherwise} \end{cases}$$

Figure 2: Reconstruction loss

3.3 Finetuning

To test how successful the model is at learning features we fine-tune on 3 other datasets. These datasets can be seen in table 1.

Dataset	Type	Subjects	Channels	Frequency	Targets
BCIC 4(33) - dataset 2a	Motor Imagery	9	22	250	4
P300 - dataset 1(34)	ERP	8	8	256	2
P300 - dataset 2(35)	ERP	10	16	256	2

Table 1: Datasets used for fine-tuning

After pre-training the encoder copied to be used for transfer learning. The encoder itself is the only thing saved from pre-training. Since the shapes number of channels are different across the dataset, we train a new embedding layer for each dataset.

For some of our experimentation we leave the encoder frozen, while only training the embedding layer and the classification head of the model. This is the a good way to evaluate the usefulness of the features output by the encoder. It is common in self-supervised learning to evaluate the quality of representation by linear evaluation. In linear evaluation the encoder is left frozen and a linear layer is trained on top of the encoder for a specific classification task.

For fine-tuning, the encoder is also initially also left frozen in order to bootstrap the new embedding layer. For this part we use a relatively high learning rate of 0.01 or 0.001. After bootstraping for certain number of steps, depending on the dataset, we unfreeze the encoder and train the whole model again. This time we use a much lower learning rate of 1e-4, same as used in pre-training.

3.4 Training Setup

We train the models on Google Colab(36), using a Tesla 100 GPU, while we log the experiments using WeightsAndBiases(WandB)(37). The models are saved to WandB throughout the training and after training is complete. Our setup is based on Pytorch, and relies on the Pytorch Lightning(38) library.

4. Experiments

The models we explored are all of a significant complexity, and in order to select and optimize the right architecture, we have run several experiments to determine the efficacy of different approaches

and hyper-parameter settings. We examine different model architectures w.r.t loss types, different data augmentations, size of the models and how scaling of the number of subjects affect the performance. To get an accuracy estimate we make a few runs over a single subject of the BCI4-2a and average the scores on the testing set.

4.1 Study of Frameworks

In the initial part testing, we experimented with different self-supervised approaches. With these test we decided which approach would be worth exploring further. The test includes three types of losses: a reconstruction loss, a contrastive loss and a non-contrastive loss. The testing is done using a model of embedding dimension of 256 and a depth of 6. In this initial test of framework, we evaluated performance by both pre-training and fine tuning on the BCI4-2a dataset and comparing using the test accuracy.

4.1.1 Contrastive framework

The we apply the SimCLR framework(15) to the same transformer model we use. In the SimCLR framework you create two version of the same input using augmentation. Those two versions are then fed through a model, where the contrastive loss NT-Xent, is applied between the two outputs.

4.1.2 Non-contrastive framework

The non-contrastive loss is modeled after Data2Vec(23), which is basically a transformer based version of BYOL(20). The setup here is having 2 models: a student model and a teacher model, where the teacher is an exponential moving average of the student. The teacher receives the raw input and the student gets a masked version. We then try to minimize the distance between the student's and teacher's outputs. Data2Vec also uses a smooth L1 loss.

4.1.3 Autoencoder framework

The autoencoder is inspired by the BERT(16) family of models. With this setup, the input is augmented, then passed through the model. The model then tries to reconstruct the original, non-augmented input. Here we again use a smooth L1 loss.

Frameworks	Linear evaluation - accuracy
Constrastive	0.684
Non-contrastive	0.7674
Autoencoder	0.8546

Table 2: Performance comparison of different frameworks

Table 2 shows the performance of different types of frameworks. The Autoencoder framework clearly outperforms both the contrastive and non-contrastive frameworks, and as such we choose to focus on the Autoencoder framework.

4.2 Augmentation

We set out the test different masking and augmentations methods. When using the braindecode augmentation library with their *AugmentedDataloader*, individual datapoints are replaced by an augmented version based on the given probability. We experimented with shuffling the channels, and doing frequency shifts, time reversal, along with Gaussian noise. The experiments were conducted with only one augmentation at a time, to see the effects in isolation. The effects on these augmentations are all based on a given input probability. In the experiments with the braindecode

augmentations, we strived to use the same parameters for the model, as to isolate the effects of each augmentation, and make comparing these different augmentations easier. Here, we also set the masking probability to 0%, as masking has shown to have a notable impact on the loss. We trained a model with no augmentation or masking to use as a baseline.

Augmentation	25%	33%	50%	80%	100%
Channel shuffle	0.6725	0.6591	**0.6824**	0.6406	0.6436
Frequency shift	**0.7014**	0.641	0.6122	0.5706	0.6458
Time reverse	0.6334	0.6064	**0.6488**	0.5657	0.5958
Gaussian noise	0.5517	0.5385	0.6181	**0.6629**	0.643
Baseline	**0.6571**				

Table 3: Results from experiments with data augmentation

As seen in table 3, the different augmentation methods have very different "optimal" probability-values. Trends can be seen among Gaussian noise and Frequency shift, but for channel shuffle and time reverse, the values seem more random. Pre-training the model with a combination of these augmentations at their "optimal" probabilities yielded poorer performance than the baseline, with a maximum validation accuracy of 0.6011. Fine-tuning the model yielded an accuracy of 0.7618. It is likely that the optimal probabilities for the augmentations when used in combination are very different. Shuffling channels with a 50%, along with frequency shift with a 25% probability, yielded better results than the baseline, suggesting that these augmentation techniques are effective w.r.t pre-training.

Mask type	Random	Cut-out	Mixed
Validation acc	0.7326	0.7244	0.724

Table 4: Results of different masking techniques

Earlier experiments suggested that generally a masking percentage of 40% yielded the best performance. The experiments with masking techniques, seen in figure 4, shows a low discrepancy between the performance of the different applied techniques. Random masking seems to perform slightly better than cut-out and mixed.

4.3 Patch size

We set out to test for the most effective patch size for self-supervised training. The patch size determines how many time steps are included for each token. The patch size also has an impact of the embedding, because of the way the channel projection is done. Since the the number of values has to add up to the embed size set for the model, whenever you increase the patch size, the number of neurons in the channel projection has to be decreased as a result.

We test with a embedding dimension of 512 and a depth of 8 on the model. We test three different patch sizes: 8, 16, and 32.

With a patch size of 8, 16 and 32, the channel projection size will be 64, 32, and 16 respectively.

patch_size: 4	patch_size: 8	patch_size: 16	patch_size: 32
0.6964	0.6976	0.6907	0.7146

Table 5: Results of experiments with patch size

We train a linear evaluate on all subjects of the BCI4-2a dataset with the metric being accuracy. The results clearly show that a patch size of 8 is subpar. Interestingly we do not see a large discrepancy, between the patch sizes of 16 and 32.

4.4 Model size

We test how the size of the model affects performance. The size of the model in our setup is mainly determined by the *embed_size* parameter

Train type	Embed_size: 128	Embed_size: 256	Embed_size: 512
Pre-train	0.6938	0.7306	0.7146
Fine tuning	0.6732	0.6338	0.7415

Table 6: Results of experiments with model size

Table 6 shows a trend with model size. There is some variation in the results, but overall the results suggest that performance increases with the model size.

4.5 Scaling

For this experiment we evaluate the impact of having more samples to train on. The model embed_size is set to 512 with a patch_size of 16. Our hypothesis is that training on a higher number of subjects results in better performance.

10	50	100	300	500
0.6595	0.6692	0.7114	0.6932	0.7212

Table 7: Results of experiments with different amount of subjects

As seen in table 7, as the number of subjects increases, the model performs better. It is interesting to note that the performance is lower with 300 subjects compared to both 100 and 500 subjects, but we contribute this to random variations.

5. Results

We evaluate the capability of our pretraining framework by finetuning it on a number of EEG datasets and comparing it against popular supervised learning models. We eavulate on the BCI4-2a, P300-1 and P300-2 datasets from table 1.

BCI4- 2a	S1	S2	S3	S4	S5	S6	S7	S8	S9
EEGResnet(39)(40)	**0.7768**	**0.4506**	0.7712	0.5027	0.4185	0.4295	0.7599	**0.7198**	0.6858
TCN(41)(42)	0.5258	0.4075	0.6836	0.3877	0.4243	0.4519	0.543	0.7141	0.6494
ShallowFBCSPNet(39)(43)	0.7292	0.434	0.7867	**0.5908**	**0.5438**	**0.4899**	**0.7955**	0.7126	0.7053
EEGNetv4(44)(45)	0.7069	0.4022	**0.8918**	0.4836	0.4671	0.4793	0.7507	0.6069	**0.7702**
Ours - Linear Eval	0.5864	0.348	0.6007	0.3649	0.3922	0.3819	0.5871	0.5101	0.5738
Ours - Finetune	0.7429	0.4363	0.8448	0.3926	0.4505	0.4298	0.7146	0.6309	0.7009

Table 8: Results of benchmarks using the BCI4 Test set

P300 - 1	S1	S2	S3	S4	S5	S6	S7	S8
EEGResnet(39)(40)	0.8067	0.8449	0.8362	0.7837	0.8338	0.8033	0.8495	0.859
TCN(41)(42)	0.8343	0.8362	0.8361	0.829	0.8357	0.8357	0.8333	0.8329
ShallowFBCSPNet(39)(43)	**0.8484**	**0.8656**	**0.8813**	**0.8672**	0.8516	**0.8797**	0.8766	**0.9**
EEGNetv4(44)(45)	0.8067	0.8633	0.8362	0.819	**0.8552**	0.8286	**0.8776**	0.8695
Ours - Linear Eval	0.8438	0.8271	0.8367	0.8367	0.8336	08219	0.839	0.8325
Ours - Finetune	0.8333	0.8524	0.84	0.8533	0.857	0.8533	0.86	0.8905

Table 9: Results of benchmarks using the p300-1 dataset

P300 -2	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
EEGResnet	0.8426	0.86	0.8206	0.8148	0.8866	0.809	0.8067	0.8183	0.8993	0.912
TCN	0.8047	0.8229	0.8385	0.8099	0.8359	0.849	**0.8724**	0.8438	0.875	0.8963
ShallowFBCSPNet	0.838	0.8552	0.8229	0.8356	0.8704	0.8125	0.7882	0.8391	0.8669	0.8867
EEGNetv4	**0.8889**	**0.9144**	0.86	0.8854	**0.9248**	0.8588	0.8669	**0.897**	**0.9109**	0.9282
Ours - Linear Eval	0.8391	0.8519	0.8275	0.8738	0.9132	0.8519	0.8368	0.8414	0.853	0.8553
Ours - Finetune	0.8704	0.8634	**0.8935**	**0.8935**	0.919	**0.8866**	0.8206	0.8762	0.9005	**0.9306**

Table 10: Results of benchmarks using the p300-2 dataset

6. Discussion

The constraints of this project, both in time as well as computing resources, meant that we were unable to scale the framework further. However, our experiments in sections 4.4 and 4.5 suggest that the framework is able to scale further. It is clear from our scaling experiments, and it is also our intuition from testing a variation of different models, that using samples from more subjects yields better results. This of course is not surprising, deep learning tends to perform better with more data. We also believe that there are gains to be had by increasing model size. The model size experiments indicate that our biggest model of embed size 512 performs best out of those tested. We would like to be able to train an even bigger model. Because of the time constraints, we have also been limited in the length of the runs we could do. The performance graphs from the different experiments suggest that better performance could have been achieved if we let the models pre-train for longer. Fine-tuning on our proposed model at intermediary train-step intervals confirms that more pre-training increases the performance.

Augmentation is usually quite important, but our experiments shows a limited impact of the augmentation techniques we used. Further experiments are needed to verify whether it is simply the techniques themselves, or whether we have not succeeded in finding the optimal values. Further augmentation techniques should also be explored.

In the experiments we saw some variability, for instance with scaling the number of subjects during pre-training. We attribute these variations to random factors. To confirm this, we would have to re-run the respective experiments, however this has not been possible due to time constraints.

For the framework we utilize a channel-wise embedding, which we retrain in order to transfer to other datasets. This has proven effective, though it would be interesting to test this against the traditional way of channel mapping.

For the benchmark we ended up doing evaluation on a single subject. This was a measure to save time and is clearly less than ideal. Evaluating in an intra-subject manner would have been preferable, since that is how we conduct the benchmarks. Even doing inter-subject, where we train a classifier

over all subject could have been better in terms of evaluating how well our pre-trained encoder produces features.

Considering our constraints, we still have managed to achieve performance comparable to supervised methods, which given the comparatively small size of our model compared to the BENDR model, we find promising.

7. Conclusion

In this paper we present a self-supervised framework for pre-training on EEG data. We pre-train on the subset of the large TUEG dataset. We then evaluate the trained models on 4 different datasets. We show that our framework is able to learn useful features from pre-training on the TUEG dataset. We demonstrate that it is possible to transfer a pre-trained model to dataset it has never seen before. The framework can produce comparable results to what supervised models showcases. Our approach however, would not compare favourably to newer state-of-the-art models. We argue, however that the framework should be able to scale with more data and potentially with bigger models. Three factors can be scaled with more compute: model size, number of samples and the length of pre-training.

Acknowledgements

Special thanks to Dalin Zhang for supervising this project, and providing feedback.

References

[1] Simanto Saha, Khondaker A. Mamun, Khawza Ahmed, Raqibul Mostafa, Ganesh R. Naik, Sam Darvishi, Ahsan H. Khandoker, and Mathias Baumert. Progress in brain computer interface: Challenges and opportunities. *Frontiers in Systems Neuroscience*, 15, 2021. https://www.frontiersin.org/article/10.3389/fnsys.2021.578875.

[2] Andrea Biasiucci, Benedetta Franceschiello, and Micah M. Murray. Electroencephalography. *Current Biology*, 29(3):R80–R85, 2019. https://www.sciencedirect.com/science/article/pii/S0960982218315513.

[3] Yannick Roy, Hubert Banville, Isabela Albuquerque, Alexandre Gramfort, Tiago H Falk, and Jocelyn Faubert. Deep learning-based electroencephalography analysis: a systematic review. *Journal of Neural Engineering*, 16(5):051001, aug 2019. https://doi.org/10.1088/1741-2552/ab260c.

[4] Neuromarketing Science Business Association. Signal to noise in eeg. https://nmsba.com/neuromarketing-companies/neuromarketing-2021/signal-to-noise-in-eeg, 2021. Last visited 16-06-2021.

[5] U. Rajendra Acharya, S. Vinitha Sree, G. Swapna, Roshan Joy Martis, and Jasjit S. Suri. Automated eeg analysis of epilepsy: A review. *Knowledge-Based Systems*, 45:147–165, 2013. https://www.sciencedirect.com/science/article/pii/S0950705113000798.

[6] B.T. Klassen, J.G. Hentz, H.A. Shill, E. Driver-Dunckley, V.G.H. Evidente, M.N. Sabbagh, C.H. Adler, and J.N. Caviness. Quantitative eeg as a predictive biomarker for parkinson disease dementia. *Neurology*, 77(2):118–124, 2011. https://n.neurology.org/content/77/2/118.

[7] M. M. Lansbergen, M. van Dongen-Boomsma, J. K. Buitelaar, and D. Slaats-Willemse. Adhd and eeg-neurofeedback: a double-blind randomized placebo-controlled feasibility study. *Journal of Neural Transmission*, 118, 2011. https://link.springer.com/article/10.1007/s00702-010-0524-2.

[8] Turan M. Itil. Qualitative and Quantitative EEG Findings in Schizophrenia. *Schizophrenia Bulletin*, 3(1):61–79, 01 1977. https://doi.org/10.1093/schbul/3.1.61.

[9] Krupal Sureshbai Mistry, Pablo Pelayo, Divya Geethakumari Anil, and Kiran George. An ssvep based brain computer interface system to control electric wheelchairs. pages 1–6, 2018. https://ieeexplore.ieee.org/document/8409632.

[10] Dr. Rima E. Laibow MD, Albert N. Stubblebine MSc, Henry Sandground, and Michel Bounias DSc. Eeg-neurobiofeedback treatment of patients with brain injury: Part 2: Changes in eeg parameters versus rehabilitation. *Journal of Neurotherapy*, 5(4):45–71, 2002. https://doi.org/10.1300/J184v05n04_04.

[11] William Yaputra Budiman, Handayani Tjandrasa, and Dini Adni Navastara. Classification of eeg signals using common spatial pattern-principle component analysis and interval type-2 fuzzy logic system. In *2016 International Conference on Information Communication Technology and Systems (ICTS)*, pages 85–89, 2016. https://ieeexplore.ieee.org/document/7910278.

[12] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015. https://www.nature.com/articles/nature14539.

[13] Iqbal H. Sarker. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2(6):420, Aug 2021. https://doi.org/10.1007/s42979-021-00815-1.

[14] Demetres Kostas, Stephane Aroca-Ouellette, and Frank Rudzicz. Bendr: using transformers and a contrastive self-supervised learning task to learn from massive amounts of eeg data. https://arxiv.org/abs/2101.12037, 2021.

[15] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020. https://arxiv.org/abs/2002.05709.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. http://arxiv.org/abs/1810.04805.

[17] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. 2019. https://arxiv.org/abs/1911.05722.

[18] Mostafa 'Neo' Mohsenvand, Mohammad Rasool Izadi, and Pattie Maes. Contrastive representation learning for electroencephalogram classification. *MLH4H*, 2020. http://proceedings.mlr.press/v136/mohsenvand20a/mohsenvand20a.pdf.

[19] Joseph Y. Cheng, Hanlin Goh, Kaan Dogrusoz, Oncel Tuzel, and Erdrin Azemi. Subject-aware contrastive learning for biosignals. *CoRR*, abs/2007.04871, 2020. https://arxiv.org/abs/2007.04871.

[20] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. 2020. https://arxiv.org/abs/2006.07733.

[21] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. 2021. https://arxiv.org/abs/2104.14294.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. http://arxiv.org/abs/1706.03762.

[23] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. data2vec: A general framework for self-supervised learning in speech, vision and language. *CoRR*, abs/2202.03555, 2022. https://arxiv.org/abs/2202.03555.

[24] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. https://arxiv.org/abs/2005.14165.

[25] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. https://arxiv.org/abs/2010.11929.

[26] Xie Chen, Yu Wu, Zhenghao Wang, Shujie Liu, and Jinyu Li. Developing real-time streaming transformer transducer for speech recognition on large-scale dataset. *CoRR*, abs/2010.11395, 2020. https://arxiv.org/abs/2010.11395.

[27] Seo-Hyun Lee Young-Eun Lee. *EEG-Transformer: Self-attention from Transformer Architecture for Decoding EEG of Imagined Speech*. https://arxiv.org/pdf/2112.09239.pdf.

[28] Yunzhe Tao, Tao Sun, Aashiq Muhamed, Sahika Genc, Dylan Jackson, Ali Arsanjani, Suri Yaddanapudi, Liang Li, and Prachi Kumar. Gated transformer for decoding human brain eeg signals. 2021:125–130, 11 2021. https://ieeexplore.ieee.org/document/9630210.

[29] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *CoRR*, abs/2006.11477, 2020. https://dblp.org/rec/journals/corr/abs-2006-11477.

[30] Iyad Obeid and Joseph Picone. The temple university hospital eeg data corpus. *Frontiers in Neuroscience*, 10, 2016. https://www.frontiersin.org/article/10.3389/fnins.2016.00196.

[31] Zhaowei Cai, Avinash Ravichandran, Subhransu Maji, Charless C. Fowlkes, Zhuowen Tu, and Stefano Soatto. Exponential moving average normalization for self-supervised and semi-supervised learning. *CoRR*, abs/2101.08482, 2021. https://arxiv.org/abs/2101.08482.

[32] Braindecode. Augmentation. https://braindecode.org/api.html#augmentation.

[33] Bci competitions. https://bbci.de/competition/iv/.

[34] Angela Riccio, Luca Simione, Francesca Schettini, Alessia Pizzimenti, Maurizio Inghilleri, Marta Olivetti Belardinelli, Donatella Mattia, and Febo Cincotti. Attention and p300-based bci performance in people with amyotrophic lateral sclerosis. *Frontiers in Human Neuroscience*, 7, 2013. https://www.frontiersin.org/article/10.3389/fnhum.2013.00732.

[35] Aricò P, Aloise F, Schettini F, Salinari S, and Mattia Dand Cincotti F. Influence of p300 latency jitter on event related potential-based brain-computer interface performance. *Neural Eng*, 2014. https://pubmed.ncbi.nlm.nih.gov/24835331.

[36] Google. Colab. https://colab.research.google.com/.

[37] WandB. Wandb. https://wandb.ai.

[38] Pytorch Lightning. Pytorch lightning. https://www.pytorchlightning.ai/.

[39] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human Brain Mapping*, 38(11):5391–5420, 2017. https://onlinelibrary.wiley.com/doi/abs/10.1002/hbm.23730.

[40] Braindecode. Eegresnet. https://braindecode.org/generated/braindecode.models.EEGResNet.html.

[41] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. https://arxiv.org/abs/1803.01271, 2018.

[42] Braindecode. Temporal convolutional network (tcn). https://braindecode.org/generated/braindecode.models.TCN.html.

[43] Braindecode. Shallowfbcspnet. https://braindecode.org/generated/braindecode.models.ShallowFBCSPNet.html.

[44] Vernon J Lawhern, Amelia J Solon, Nicholas R Waytowich, Stephen M Gordon, Chou P Hung, and Brent J Lance. EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces. *Journal of Neural Engineering*, 15(5):056013, jul 2018. https://doi.org/10.1088%2F1741-2552%2Faace8c.

[45] Braindecode. Eegnetv4. https://braindecode.org/generated/braindecode.models.EEGNetv4.html.