SPP-EEGNET: An Input-Agnostic Self-supervised EEG Representation Model for Inter-Dataset Transfer Learning

Xiaomin Li, Vangelis Metsis

Texas State University, San Marcos TX 78666, USA,
x_l30@txstate.edu, vmetsis@txstate.edu

Abstract. There is currently a scarcity of labeled Electroencephalography (EEG) recordings, and different datasets usually have incompatible setups (e.g., various sampling rates, number of channels, event lengths, etc.). These issues hinder machine learning practitioners from training general-purpose EEG models that can be reused on specific EEG classification tasks through transfer learning. We present a deep convolutional neural network architecture with a spatial pyramid pooling layer that is able to take in EEG signals of varying dimensionality and extract their features to fixed-size vectors. The model is trained with a contrastive self-supervised learning task on a large unlabelled dataset. We introduce a set of EEG signal augmentation techniques to generate large amounts of sample pairs to train the feature extractor. We then transfer the trained feature extractor to new downstream tasks. The experimental results[1] show that the first few convolutional layers of the feature extractor learn the general features of the EEG data, which can significantly improve the classification performance on new datasets.

Keywords: EEG, self-supervised learning, contrastive learning, transfer learning, spatial pyramid pooling

1 Introduction

Electroencephalography (EEG) has enabled numerous applications both in and out of the clinical domain. For example, it has been used for diagnosing epilepsy [1], monitoring sleep stages [2], emotion recognition [3], human-computer interaction [4], etc. However, EEG signals often have a low signal-to-noise ratio, high variability among subjects, and are collected in confugurations that are incompatible across datasets, making them hard to interpret and challenging to use in an end-to-end automated processing pipeline (i.e., applying machine learning models on raw signals).

In the past, most machine learning EEG applications relied on hand-crafted features extracted by neuroscience experts. They usually required designing specific solutions for one particular task and are hard to be re-used on other

[1] The source code on Github: https://github.com/imics-lab/eeg-transfer-learning

tasks. Due to the difficulty of manually extracting useful features, only a limited amount of tasks with clear EEG patterns can benefit from automatically trained machine learning models. Deep learning allows end-to-end signal processing, feature extraction, and model training, which has enabled significant performance improvements in the EEG field [5, 6, 7] compared to traditional machine learning methods. Despite the success of deep learning in classifying EEG signals, the majority of these approaches are subject to the limited amounts of labeled EEG recordings that often lead to relatively shallow deep neural networks, trained in supervised ways, and on a single dataset.

Applications of deep learning in other domains (e.g. Computer Vision and NLP), have vastly benefited from transfer learning, i.e. starting with a neural network that has been pre-trained on big datasets and then fine-tuned on a specific task to achieve better classification accuracy rates and training performance speedup on smaller datasets.

Self-supervised Learning (SSL) is an approach for learning useful representations from unlabelled data. SSL allows deep neural networks to first learn general-purpose features from large unlabeled datasets and then fine-tune the representations for the supervised downstream tasks. Contrastive learning has been used to learn the general features of a dataset without labels by teaching the model which data points are similar or different. It has achieved great success in many fields. For example, image patches reordering [8] in Computer Vision, sentence embedding [9] in Natural Language Processing, and Wave2Vec [10] in time series data.

A few works have also implemented contrastive learning on EEG data. Banville et al. [11] developed three solutions mainly by defining negative windows and positive windows located in different places in an EEG recording and sample positive and negative data pairs from those windows. However, it only implements the downstream tasks on the same dataset trained for the pre-training task but did not verify if the learning features also fit other datasets. Mohsenvand et al. [12] developed a single-channel EEG representation model which fused multiple datasets and recombined multiple channels in one to boost the variety of channels. The work of Kostas et al. [13] is the most similar work compared to ours. They designed a CNN feature extractor and pass features to a transformer model to conduct contrastive learning tasks. However, the datasets used for the pre-training task and downstream tasks all need to be resampled to the same sampling rate.

In this paper, we present a new framework that allows us to, 1) learn EEG representations in a self-supervised manner via contrastive learning without requiring external labels, and 2) transfer the pre-trained feature extractor to other downstream tasks without modifying target dataset dimensions to match the properties of the original datasets used for the self-supervised learning task.

We design a contrastive learning task to train the feature extractor. To generate contrastive learning data pairs, we introduce a few EEG data augmentation methods. For example, adding noise, downsampling, random cropping, pooling, and quantizing signal voltage values, etc. To transfer the trained feature extrac-

tor without modifying the downstream dataset's input dimensions, we design a densely connected convolutional neural network with spatial pyramid pooling layer [14] to map any size of input data to fixed size vectors. Two data samples in a data pair both pass through the feature extractor and the absolute difference value are calculated in the latent space. Then a linear classifier is used to distinguish whether it is a positive data pair or negative one.

While transferring the pre-trained feature extractor to downstream tasks, we load the parameters of different numbers of convolutional layers to target classification models. The experimental results show that the contrastive learning task helps the feature extractor learn general EEG signal features that can be re-used on downstream tasks to improve the model's classification performance.

Our contributions can be summarized as follows:

- We exploit an EEG signal feature extractor that is agnostic of the input data dimension and can be re-used for any downstream task without modifying the target dataset's properties.
- We conduct a contrastive self-supervised learning task to extract general features from the EEG signals from massive EEG data without external labels.
- We perform transfer learning from the pre-trained feature extractor to other smaller EEG datasets. The experimental results show significant classification performance improvements compared to training smaller datasets on deep learning models from scratch.

The rest of the paper is organized as follows. Section 2 introduces our methodology. The datasets, data preprocessing steps, experiments setup, and results are described in Section 3. Section 4 concludes the paper.

2 Method

2.1 SPP-EEG Feature Extractor

We design an input dimension agnostic feature extractor, SPP-EEG network. The architecture is shown in Fig. 1. The convolution layers all have the kernel size $(1, 3)$ and filter count $F = 32$. They are applied to all input signal channels to learn the representations of each specific 10-20 EEG montage system channel. There are two convolution layers which have stride size $(1, 2)$ used to downsample the feature map by a factor of $1/2$. A dropout layer is added after every three convolution layers to alleviate the over-fitting problem.

The Spatial Pyramid Pooling layer ensures every input signal length and shape be extracted to the exact same length of feature vectors. By choosing a pooling bin $[A, B, C]$, a vector of $size = F * (A^2 + B^2 + C^2)$ will be extracted after the spatial pyramid pooling layer for all shapes of input data. Then a dense layer with the embedding size 100 is added in this model. The spatial pyramid pooling layer enables the feature extractor to be trained with multiple EEG datasets with various data dimensions and learns the general EEG data features that co-exist on all recordings. And it can be transferred to any other downstream

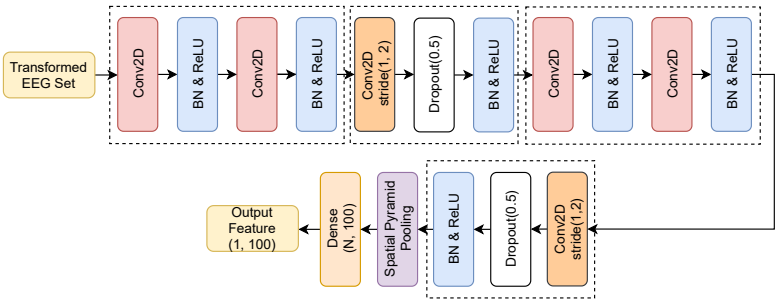

Fig. 1: SPP-EEG feature extractor architecture. Any dimensional input signals will be mapped to the same size 1D vectors. The Conv2D layers reserve general EEG signal features that can be transferred to downstream tasks.

EEG classification task without modifying the original signal collection settings (e.g. sampling rate, event window size, etc).

2.2 Contrastive learning

Fig. 2 shows the contrastive learning pipeline. We denote the input multivariate time-series $S \in R^{1\times C\times N}$, 1 represents that we view each EEG input as an 2D image with only one color channel. C is the total number of EEG data channels following the 10-20 EEG Montage system [15]. N is the total time-points of each input data sample. Here N can vary depending on various event window sizes, data transformations, and signal sampling rates, etc.

To produce labeled samples from the multivariate time-series S, we propose to sample pairs of time windows $(x_t, x_{t'})$ where each window x_t, $x_{t'}$ is in $R^{1\times C\times N}$. The first window x_t is referred to as the "anchor window". The second window $x_{t'}$ can be a transformation of the anchor window, where $(x_t, x_{t'})$ has the label '1'. $x_{t'}$ can also be a transformation of a random time window which is far away from the anchor window for a distance of at least 10 times the window size, where $(x_t, x_{t'})$ has the label '0'. Our assumption is that an anchor window and its transformed counterparts still have some similar features and should have the same label, whereas the distant window should have a different label.

In order to learn how to discriminate pairs of time windows based on their sampled position, we use the feature extractor mentioned in section 2.1 to map a window x to its representation in the latent space, $f_\Theta : R^{1\times C\times N} \rightarrow R^D$. Ultimately we expect f_Θ to learn an informative representation of the raw EEG input which can be reused in different downstream tasks. A contrastive module g_{aug} is then used to aggregate the feature representations of each window. For our task, $g_{aug} : R^D \times R^D \rightarrow R^D$ combines representation form pairs of windows by computing an element-wise absolute difference, $g_{aug}\left(f_\Theta\left(x\right) - f_\Theta\left(x'\right)\right) = \left|f_\Theta\left(x\right) - f_\Theta\left(x'\right)\right| \in R^D$.

Then we use the binary cross entropy loss with a Sigmoid function on the predictions of g_{aug}, we can write a joint loss function as:

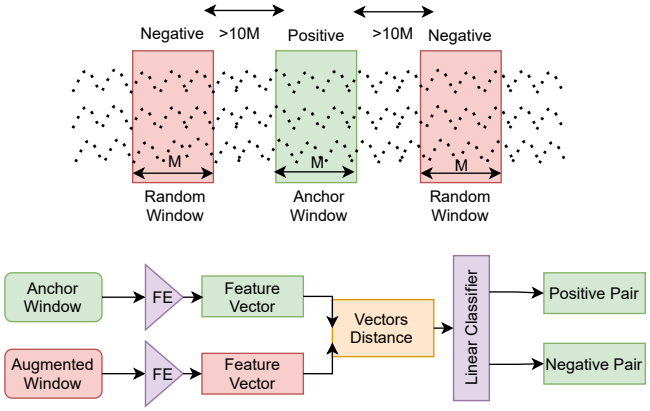

Fig. 2: Contrastive learning pipeline. FE represents the SPP-EEG feature extractor shown in Fig. 1. The absolute differences of two feature vectors are computed and a linear classifier is used to classify whether two vectors are similar or not in the latent space.

Transformation	min	max
Add Noise (scale)	0.02	0.1
Downsample (Hz)	100	256
Random Crop (Time Points)	600	1200
Pooling (level)	2	5
Quantize (level)	10	30

Table 1: Transformation Ranges

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} y_i log\sigma\left(g_{aug}\right) + \left(1-y_i\right) log\left(1-\sigma\left(g_{aug}\right)\right)$$

2.3 Augmentation methods

Data augmentation serves a dual purpose in our methodology. It allows us to generate more training data and it enables the self-supervised contrastive learning process, as explained in section 2.2. Five different signal augmentation methods are used. Table 1 shows the range of each transformation. **Add Noise:** Add independent and identically distributed random noise to the input signal with the standard deviation of the random noise equal to a random generated scale. **Downsample:** Downsample input signals to a random sampling rate within a range but not violate the Nyquist–Shannon sampling theorem [16]. **Random Crop:** Select random sub-sequences in a range from the input signal. **Pooling:** Reduce the temporal resolution to original input 1/level without changing the input signal length. **Quantize:** Round each timepoint value to the nearest level in a level set.

2.4 Transfer Learning

After training the feature extractor with the contrastive self-supervised learning task, we transfer the feature extractor to different downstream EEG classification tasks and fine-tune it on the new dataset. The downstream classification model has a similar architecture with the pre-trained feature extractor, except we switch the spatial pyramid pooling layer to a maxpooling layer. The pooling size is decided upon each dataset's input window size. Such a decision is made empirically as the experimental results show the downstream tasks cannot benefit from the spatial pyramid pooling layer and it is also unnecessary to use it for single dataset classification tasks. Then we add a few dense layers and an output layer. The output dimension depends on each downstream task.

3 Experiments

3.1 Datasets

We conduct a self-supervised contrastive learning task on the TUH Normal / Abnormal dataset [17]. This dataset contains 2,993 recordings of around 15 minutes from 2,329 different patients who underwent a clinical EEG in a hospital setting. The training set contains 1,371 recordings labeled as normal and 1,346 labeled as abnormal. The evaluation set contains 150 and 126 recordings respectively.

We conduct transfer learning to three downstream datasets. The first one is from the same dataset used for feature extractor training. We select 30 normal and 30 abnormal patients' recordings that are not been used formerly. The second one is the EEG Motor Movement/Imagery Dataset [18, 19], which consists of over 1500 one- and two-minute EEG recordings, obtained from 109 volunteers. The recordings are labeled as $0, 1$ for four different binary motor imagery tasks. We used the first 50 subjects recordings in transfer learning experiments. The third one is the subset the first one, chooses 10 EEG recordings from the TUH normal/abnormal dataset in each training round to make up the training set and testing set.

3.2 Data preprocessing

For the TUH Normal/Abnormal dataset, we use the same train/test split, to train and test the feature extraction model, as in the original paper [17]. For the transfer learning datasets, we use KFold = 5 cross-validation.

To preprocess the TUH dataset, we follow the EEG montage 10-20 system to select and reorder all recording channels to [FP1, FP2, F7, F3, FZ, F4, F8, T3, C3, CZ, C4, T4, T5, P3, PZ, P4, T6, O1, O2]. We discard the first 1-minute time points from all recordings and crop the total length of a recording to no more than 20 minutes. We apply a (0.3-80Hz) fifth-order band-pass Butterworth filter [20] to all recordings. 6-second time windows are extracted and each time window has the dimension $size = (1, 19, 6 * Hz)$. Then all time windows are channel-wise normalized.

For the transfer learning datasets, we select and reorder recording channels the same as the TUH dataset. We follow the original event time and the sampling rate of each dataset to divide time windows, which yields different sizes of time windows for different datasets. To all recordings we apply the filter bands suggested in their manuals. And similarly, all time windows are normalized in each channel.

3.3 Training details

We conduct all experiments on an Intel server with 377GB CPU memory and 2 Nvidia 1080 GPUs. The feature extraction model is trained for 100 epochs and saved at the best performance checkpoint. We use Adam optimizer [21] with the initial learning rate of 0.001 and batch size of 32. The learning rate degrades to its 95% after every 10 epochs. Since the contrastive data pairs we generated have different window sizes, we pad all time windows to the same size in a training batch to maximize the GPU's parallelization ability. For transfer learning tasks, we still use Adam optimizer with the learning rate 0.0001 or 0.00001 depending on the dataset. The training epochs are set to 50 in each fold and the batch size is set to 32 or 64 depending on the available GPU memory.

3.4 Transfer Learning Results

We first transfer the feature extractor to the TUH Normal/Abnormal dataset. As the first dataset described in section 3.1. We segment the recordings to 6-second time windows and each is labeled as normal or abnormal. Then reload the parameters of different numbers of convolutional layers and train the classifier mentioned in Section 2.4. The experiment results are shown in Fig. 3. The red line is the classification accuracy obtained when training the same network architecture on the downstream dataset from scratch. The blue line shows the classification accuracy obtained on the downstream task when transferring the first n convolutional layers trained on the self-supervised learning task. From the plot, we can see the more than 6% accuracy improvement is achieved when transferring the first three layers on the TUH dataset. Overall, the accuracies obtained using our transfer learning approach are better than training from scratch except when transferring all six layers, which expected as in that case the original model is highly specialized on the initial self-supervised learning task. In other words, the last layers extract the patterns that particularly fit the former task which are not useful for downstream tasks.

Similar results can be found in Fig.4 when transferring the feature extractor to EEGBCI dataset which is a new dataset, not seen during the self-supervised pre-training process. The best performance appears when transferring the first four convolutional layers with around 4-5% accuracy improvement compared to training from scratch.

We conduct another experiment to show how the pre-trained feature extractor could benefit the downstream task on very small training sets. In this case, due to shortage of training data, the deep learning model quickly over-fits the

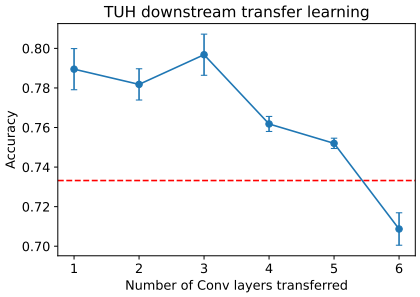

Fig. 3: TUH Downstream transfer learning results. The red line is the classification accuracy obtained when training the same network architecture on the downstream dataset from scratch. The blue line shows the classification accuracy obtained on the downstream task when transferring the first n pre-trained convolutional layers.

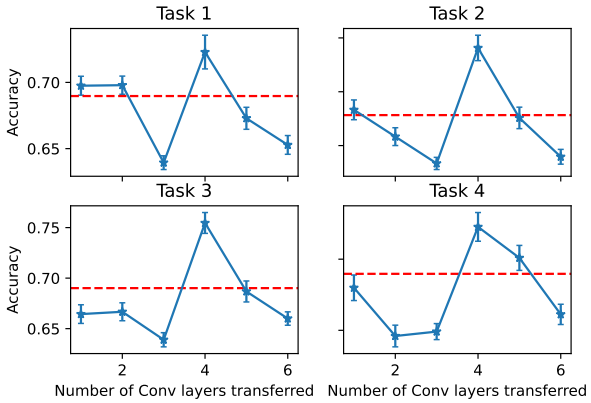

Fig. 4: EEGBCI transfer learning results for the four different tasks.

training set and performs very poorly on testing set. The experiment uses only 10 EEG recordings from the TUH normal/abnormal dataset to make up the training set and testing set. We transfer the pre-trained feature extractor, training on three different training sets, and test on a single testing set. Fig. 5 shows the transfer learning results. The smaller the difference between training and testing accuracy in the downstream deep learning model indicates less over-fitting, and the testing accuracy improves. From the plots, (1) we can observe that transferring the first three convolution layers parameters to the downstream task and freezing the first layer's parameters give the best practical result. (2) On such a small training set, the pre-trained feature extractor overall performs better than training from scratch. Our findings show that the number of layers to be transferred and the number of layers to be frozen when training on the downstream task matters. Transferring too many or freezing too many layers adds

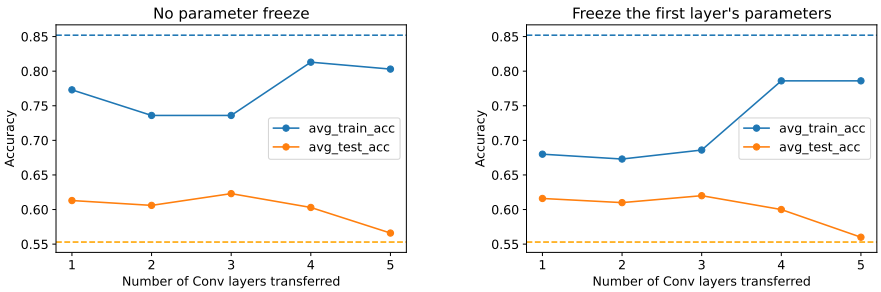

Fig. 5: Transfer learning results on a very small dataset. The upper and lower borderlines are the training from scratch training and testing accuracy.

stricter restrictions on the downstream model which makes it performs worse on the testing set.

4 Conclusion

In this work, we introduce an input data shape agnostic deep convolutional neural network for learning EEG data representations. We train the model with a self-supervised contrastive learning task and transfer the model to other smaller datasets with different recording setups. The experimental results verify the ability of contrastive learning to learn EEG data inner representation without external labels. And the pre-trained parameters can be used in other small EEG datasets as the initial training parameters to improve the downstream classification performance.

References

[1] U Rajendra Acharya et al. "Automated EEG analysis of epilepsy: a review". In: *Knowledge-Based Systems* 45 (2013), pp. 147–165.

[2] Khald Ali I Aboalayon et al. "Sleep stage classification using EEG signal analysis: a comprehensive survey and new investigation". In: *Entropy* 18.9 (2016), p. 272.

[3] Abeer Al-Nafjan et al. "Review and classification of emotion recognition based on EEG brain-computer interface system research: a systematic review". In: *Applied Sciences* 7.12 (2017), p. 1239.

[4] Fabien Lotte, Laurent Bougrain, and Maureen Clerc. *Electroencephalography (EEG)-based brain-computer interfaces*. 2015.

[5] Robin Tibor Schirrmeister et al. "Deep learning with convolutional neural networks for EEG decoding and visualization". In: *Human brain mapping* 38.11 (2017), pp. 5391–5420.

[6] Vernon J Lawhern et al. “EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces”. In: *Journal of neural engineering* 15.5 (2018), p. 056013.

[7] Dongrui Wu et al. “Transfer Learning for Motor Imagery Based Brain-Computer Interfaces: A Complete Pipeline”. In: *arXiv preprint arXiv:2007.03746* (2020).

[8] William Falcon and Kyunghyun Cho. “A framework for contrastive self-supervised learning and designing a new approach”. In: *arXiv preprint arXiv:2009.00104* (2020).

[9] Tianyu Gao, Xingcheng Yao, and Danqi Chen. “SimCSE: Simple Contrastive Learning of Sentence Embeddings”. In: *arXiv preprint arXiv:2104.08821* (2021).

[10] Ye Yuan et al. “Wave2vec: Learning deep representations for biosignals”. In: *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2017, pp. 1159–1164.

[11] Hubert Banville et al. “Uncovering the structure of clinical EEG signals with self-supervised learning”. In: *Journal of Neural Engineering* 18.4 (2021), p. 046020.

[12] Mostafa Neo Mohsenvand, Mohammad Rasool Izadi, and Pattie Maes. “Contrastive representation learning for electroencephalogram classification”. In: *Machine Learning for Health*. PMLR. 2020, pp. 238–253.

[13] 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”. In: *arXiv preprint arXiv:2101.12037* (2021).

[14] Kaiming He et al. “Spatial pyramid pooling in deep convolutional networks for visual recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), pp. 1904–1916.

[15] Robert Oostenveld and Peter Praamstra. “The five percent electrode system for high-resolution EEG and ERP measurements”. In: *Clinical neurophysiology* 112.4 (2001), pp. 713–719.

[16] Jonathan Nemirovsky and Efrat Shimron. “Utilizing Bochners Theorem for Constrained Evaluation of Missing Fourier Data”. In: *arXiv preprint arXiv:1506.03300* (2015).

[17] Silvia López, I Obeid, and J Picone. “Automated interpretation of abnormal adult electroencephalograms”. PhD thesis. Temple University Graduate Board, 2017.

[18] JR Wolpaw et al. “BCI2000: a general purpose brain–computer interface system”. In: *Soc Neurosci Abstr 2000c*. Vol. 26, p. 1229.

[19] AL Goldberger et al. “Components of a new research resource for complex physiologic signals”. In: *PhysioBank, PhysioToolkit, and Physionet* (2000).

[20] Stephen Butterworth et al. “On the theory of filter amplifiers”. In: *Wireless Engineer* 7.6 (1930), pp. 536–541.

[21] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).