Sparsity for Efficient Electroencephalography Control

Samuel Neumann
Department of Computer Science
University of Alberta
sfneuman@ualberta.ca

Laura Petrich
Department of Computer Science
University of Alberta
laurapetrich@ualberta.ca

Abstract

Brain-computer interfaces (BCI) acquire, process, and translate brain signals to provide individuals with severe motor impairments a means of communication and control. Electroencephalography (EEG) is a common method for measuring brain signals due to it's low cost and high temporal resolution. In BCI systems, neural networks are often used to translate the user's intent from such EEG signals. However, densely connected neural networks can be computationally expensive – a severe problem for real-time BCI systems. In this paper we investigate sparse neural networks for EEG-based motor classification, with the goal of reducing computational load without sacrificing model performance. We compare two sparsity-inducing algorithms, weight pruning and sparse evolutionary training, with a dense neural network baseline under three experimental conditions. Overall, our results show that sparse neural networks can achieve higher performance accuracy and generalization than their densely-connected counterparts. Our results also suggest that in order to achieve high performance accuracy it is important to include data from the end-user during training, regardless of the network configuration. We found that sparse evolutionary training achieves the highest and most stable performance across all experiments. Introducing sparsity into the network is a viable option for efficient EEG-based control.

1 Introduction and related work

Brain-computer interfaces (BCIs) are an emerging technology aimed at enabling the control of external devices via acquired brain signals [Maiseli et al., 2023]. BCIs have shown great promise in improving the quality of life for individuals living with physical disabilities by increasing their personal autonomy [Belkacem et al., 2020]. The overarching goal of a BCI system is to offer an alternative means of communication in order for the user to interact with their environment. A BCI system is composed of three general components, as shown in Figure 1: signal acquisition, signal processing (involving feature extraction, classification, and translation), and the control application [Maiseli et al., 2023]. The overall performance and success of a BCI system relies on the quality of signals acquired from the brain as these are used to interpret the user's intent.

Electroencephalography (EEG) is a method of capturing the brain's electrical activity (i.e., signal acquisition) through wet or dry electrodes placed on the scalp [Biasiucci et al., 2019]. It is a popular method used for real-time control as EEG systems are relatively low-cost, non-invasive, and portable [Al-Quraishi et al., 2018]. EEG signals have a high temporal resolution, however, they are inherently noisy and have poor spatial resolution [Bamdad et al., 2015].

Signal processing is used to extract a clean signal from a noisy EEG recording and then generate a mapping to the user's intent [Habibzadeh Tonekabony Shad et al., 2020]. First, a preprocessing step is generally performed, involving channel selection (i.e., what features are relevant to the task of interest), signal frequency filtering (i.e., what filters should we apply to remove noise), and artifact removal (i.e., how do we identify and remove irrelevant noise artifacts) [Altaheri et al., 2023]. Once

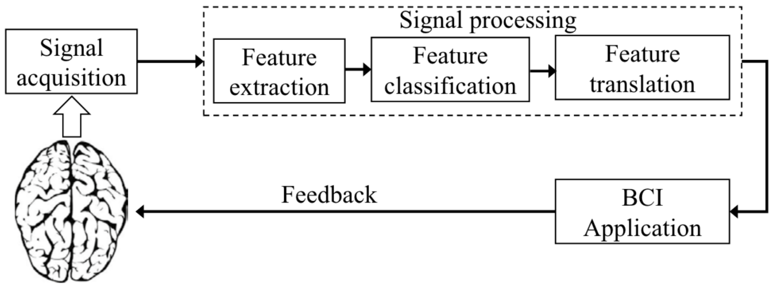


Figure 1: Figure taken from Maiseli et al. [2023] showing the three key components of a brain-computer interface system: signal acquisition, signal processing, and BCI application.

the relevant features are extracted they can be classified and translated into motor commands to send to an external device (i.e., the control application), such as to control a smart wheelchair or robotic arm [Maiseli et al., 2023].

Due to the high-dimensionality, non-stationarity, and complexity of EEG data, machine learning techniques are often used for classification [Lotte et al., 2007]. However, these machine learning algorithms, specifically deep learning with dense neural networks, can be *computationally expensive* and have *large memory requirements*. For practical, real-time control with EEG-based BCI systems, low memory and computation requirements are necessary as the entire system will need to be portable for improved usability. Yet, BCI systems should predict the user's intent accurately and react quickly to a user's input or feedback. Sparse neural networks are an emerging research area in machine learning that address both of these constraints, making such networks particularly well suited for BCI control systems.

In a sparse neural network, each neuron is connected to a subset of neurons in the proceeding layer. This is in contrast to densely or fully connected neural networks where each neuron is connected to all neurons in the proceeding layer. Sparse neural networks can be more energy efficient than fully connected neural networks, using a fraction of the memory/compute while reaching higher levels of performance and generalization [Mocanu et al., 2018, Bellec et al., 2018, Liu, 2020]. In a recent survey on sparsity in deep learning, Hoefler et al. [2021] found “that today's sparsification methods can lead to a 10-100x reduction in model size, and to corresponding theoretical gains in computational, storage, and energy efficiency, all without significant loss of accuracy." (p. 2-3). Furthermore, sparse neural networks may be more resistant to interference and catastrophic forgetting than their fully connected counterparts [Liu et al., 2019b, Ghiassian et al., 2020, Pan et al., 2021]. Finally, fully connected neural networks generally require large amounts of training data to generalize well, an important consideration in the application of BCI systems where training datasets are usually small [Lotte et al., 2007].

In this work, we investigate how sparsity affects the performance and generalization capabilities of neural networks in a classification task with EEG data. We hypothesize that, when training and testing on a fixed pool of subjects, sparse neural networks can attain higher levels of prediction accuracy and generalization compared to fully connected neural networks – with fewer neuronal connections. We further hypothesize that sparse neural networks can learn and adapt to specific users more quickly than their fully connected counterparts.

2 Background

Sparsity can be introduced into neural networks either *statically* or *dynamically*[1]. Static sparse networks are neural networks which have fixed sparse connections between neurons. These connections are not adapted during training [Grooten et al., 2023]. Sparsity in these networks is due to deliberate architecture choices. In contrast, dynamic sparse networks have sparse connections between neurons which are learned through a gradual prune-and-regrow procedure [Hoefler et al., 2021]. The goal here is to learn a network architecture which improves some performance metric [Grooten et al.,

[1]The terminology hereon is taken from Grooten et al. [2023].

2023]. Dynamic sparse networks may be more biologically plausible than static sparse networks, as the human brain is known to dynamically adjust its neuronal connections [Holtmaat et al., 2005, Stettler et al., 2006, Chambers and Rumpel, 2017].

Dynamic Sparse Training (DST) refers to the simultaneous optimization of a dynamic sparse network's weights and neuronal connections. The Sparse Evolutionary Training (SET) algorithm, shown in Algorithm 1, is one of the seminal works in the DST line of research [Mocanu et al., 2018]. SET initializes a neural network with a sparse Erdős-Rényi random graph topology [Erdős and Rényi, 1959] with sparsity level $\varepsilon \in (0,1)$. Every n epochs, SET prunes and re-grows the connections in this neural network. To do so, it first removes the $\frac{100\zeta\%}{2}$ smallest positive and largest negative weights for $\zeta \in (0,1)$. Next, an equal number of random connections are added back into the network, preserving the Erdős-Rényi random graph topology. After this, conventional training continues until the next pruning epoch. Sparse networks trained with SET can match the performance of densely connected networks with a fraction of the number of neurons and far less memory and computation [Mocanu et al., 2018].

Algorithm 1: Sparse Evolutionary Training

```
Input: prune fraction $\zeta$, prune frequency $n$, $\varepsilon$,
       optimiser $\mathbb{O}$, loss $\mathscr{L}$
Initialize network with Erdős-Rényi topology with
  sparsity level $\varepsilon$
for each training epoch $e$ do
    Train network parameters using $\mathbb{O}$ on $\mathscr{L}$
    if $e \equiv 0 \pmod{n}$ then
        Remove $\frac{100\zeta\%}{2}$ smallest positive weights
        Remove $\frac{100\zeta\%}{2}$ largest negative weights
        if e is not the last epoch then
            Add random new connections
              (weights) between consecutive layers
              equal to the amount removed
        end
    end
end
```

SET follows a *sparse-to-sparse* training regime since it initializes a neural network with a sparse topology and subsequently learns the weights and connections. In contrast, *dense-to-sparse* training algorithms begin by initializing a dense network then introduce sparsity by pruning network connections during training [Janowsky, 1989, Han et al., 2015, Zhu and Gupta, 2018, Frankle and Carbin, 2019]. Such methods generally follow a three stage process. In the first stage, a densely connected network is trained following conventional training procedures. During this stage, the learned network weights are considered to represent the importance of each connection. In the second stage, the $100\varepsilon\%$ of unimportant connections are pruned at epoch n for $\varepsilon \in (0,1)$. Typically, the importance of a weight is measured by its magnitude. Finally, the pruned, sparse network is re-trained from either the current weight values or after re-initializing the weights. A typical weight pruning algorithm (WP), and the one we consider in our empirical study, is shown in Algorithm 2.

Algorithm 2: Weight Pruning

```
Input: prune fraction $\varepsilon$, prune epoch $n$, optimiser
       $\mathbb{O}$, loss $\mathscr{L}$, final network topology $\mathscr{T}$
for each training epoch $e$ do
    Train network parameters using $\mathbb{O}$ on $\mathscr{L}$
    if $e = n$ then
        Remove $100\varepsilon\%$ of weights of lowest
          magnitude, ensuring that the resulting
          network has topology $\mathscr{T}$
    end
end
```

3 Data

For this study we chose to work with the WAY-EEG-GAL dataset collected by Luciw et al. [2014]. As this dataset is well established in the community and has been used in classification tasks to achieve a high classification accuracy, it is the perfect benchmark for our sparsity investigation. The authors gathered this dataset with the intent of evaluating whether EEG signals can be used for upper-limb prosthetic device control. The data was collected with twelve right-handed participants and contains a total of 3,936 grasp and lift events (328 events per participant). Each participant completed ten sessions consisting of many trials. For each trial the participant reached out, grasped an object, lifted the object and held it in the air for a few seconds before placing it back down and releasing the object. For more detailed information on the data collection protocol we refer the reader to the original article introducing the dataset by Luciw et al. [2014].

3.1 Data Characteristics and Considerations

When working with EEG data, it is important to consider that EEG activity may be initiated prior to or continue after actual motor movement. We also need to take into consideration the temporal sequence of the event classes (i.e., that they occur sequentially over time and always in the same order) and that some events overlap with each other.

In order to concentrate on the effect of introducing sparsity, we decided to use a subset of the WAY-EEG-GAL dataset previously extracted for a Kaggle competition[2]. This subset contains the raw EEG data (32 channels from wet electrodes with a 500Hz sampling rate) labelled with six hand motion events: HandStart, FirstDigitTouch, BothStartLoadPhase, LiftOff, Replace, and BothReleased [Edin et al., 2015]. An event label was assigned to a sample if the event occurred within +-150ms of the EEG recording. Since the events occur sequentially, each EEG sample may be labelled with multiple events. We account for this in Section 4.1.

EEG data is known to be noisy and requires filtering in order to be sure that artifacts are not interfering with model predictions [Jiang et al., 2019]. Head or wire movements as well as perspiration can cause low frequency noise artifacts. High frequency artifacts can be caused by muscle contractions in the vicinity of the EEG recording sites (electromyography artifacts), eye blinking or movement (electrooculography artifacts), and electromagnetic interference. Since the EEG data was gathered during the execution of a grasp and lift motor task, we decided to apply a conservative bandpass filter to accommodate for unintentional artifacts caused by the movement itself.

4 Methods

In this section we provide details on data pre-processing, network configurations, experimental pipeline, and statistical analyses.

4.1 Data pre-processing

Noise filtering During data collection for the original WAY-EEG-GAL dataset, no pre-processing (e.g., artifact rejection) steps were applied to the EEG signals other than the removal of "unneeded or extra samples at the beginning and end of each series." (p. 5) [Luciw et al., 2014]. To ensure that noise artifacts did not interfere with our model predictions while maintaining as many potentially meaningful signals as possible, we applied a bandpass filter with a low frequency cutoff of 0.2 Hz and a high frequency cutoff of 50 Hz.

Classification labels We decided to concentrate on training our classifiers to detect four distinct events that would not overlap with one another in the EEG data: HandStart (class 0), LiftOff (class 1), Replace (class 2), and BothReleased (class 3). The class label, class number, and number of samples for each class are summarized in Table 1. We refer to this reduced dataset as the *GAL-EEG dataset*.

Class Label	Class Number	Number of Samples
HandStart	0	468000
LiftOff	1	388279
Replace	2	321905
BothReleased	3	321905

Table 1: Class details for the GAL-EEG dataset used in our multi-class classification problem.

4.2 Algorithms and hyperparameter tuning

To better understand how sparse neural networks can perform on an EEG-based classification task, we studied the learning process of two dynamic sparse training (DST) algorithms. The first algorithm we studied was sparse evolutionary training (SET), shown in Algorithm 1. The second algorithm was a weight pruning (WP) algorithm similar to those used by Han et al. [2015] and Frankle and Carbin

[2] `https://www.kaggle.com/competitions/grasp-and-lift-eeg-detection/overview`

[2019] and shown in Algorithm 2. We did not re-initialize weights after pruning, rather training continued from the current weight values. We compared these two DST algorithms with two baselines. The first baseline was a densely connected network (DN) following standard training procedures. The second baseline randomly predicted class labels according to their relative occurrence frequency in the dataset. This baseline served as a sanity check to ensure that each algorithm learned reasonable behaviour. All algorithms were implemented in `Flax` [Heek et al., 2023] using the `jaxpruner` library [Lee et al., 2023].

Due to the nature of our study, we were not interested in attaining state-of-the-art performance on the GAL-EEG dataset. Rather, we wanted to characterize the learning processes of sparsity-inducing algorithms. We therefore decided to use small networks. For all experiments, neural networks were composed of two hidden layers. We performed a small sweep of hidden layer sizes in $\{128, 256, 512, 1024\}$ and found that all networks performed best with hidden layers of size 1024 and that the computation time for all network sizes was similar. For the rest of the experiments we set the hidden layers to size 1024. Each network predicted a softmax distribution over class labels. All neural networks were trained using the Adam optimizer [Kingma and Ba, 2014] with the default hyperparameters, except for the stepsize.

To tune the hyperparameters, we conducted a grid search for multiple runs with different random seeds. We selected the tuned hyperparameters based on the highest average prediction accuracy achieved throughout training. For all experiments we swept stepsizes in $\{0.01, 0.001, 0.0001\}$. In our preliminary experiments, we evaluated the prediction accuracy of SET and WP for sparsity levels $\varepsilon \in \{0.25, 0.5, 0.75\}$ and percentage of dropped weights (SET) $\zeta \in \{0.15, 0.3, 0.5\}$. For both the sparsity level and percentage of dropped weights, we found that prediction accuracy was highest for the lowest two values and swept over those values for the final experiments. We further performed a sweep of batch sizes and found that full-batch updating resulted in highest prediction accuracy across all algorithms. However, in a number of experiments, full-batch updating was not possible due to the immense size of the training data. We therefore used a large batch size of 8192, further discussed in Appendix B. We tested pruning weights every $\{1, 2, 5\}$ epochs for SET and $\{167, 333, 400\}$ epochs for WP. We used the Erdős-Rényi random graph topology for both SET and WP. The final best performing tuned hyperparameters are listed in Appendix C.

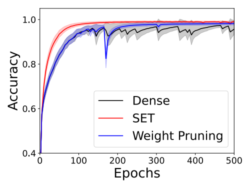
For all experiments we used a weighted cross-entropy loss similar to that used by Pedregosa et al. [2011]. The loss term for each class y was weighted by $\frac{N}{Cn_y}$, where N is the total number of data samples in the dataset, C is the number of classes, and n_y is the number of data points with label y. We began our study using Nested K-Fold Cross Validation, but quickly found that this was computationally intractable due to the size of our dataset. We then switched to a stratified train, test, and validation split for each experiment. Unless otherwise stated, training sets were composed of 80% of the full data while validation and test sets were each composed of 10% of the data.

4.3 Experiment details

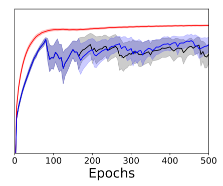
We conducted three different experiments to test the efficacy of sparsity in different settings. First, we evaluated each algorithm when training and testing data were from the same pool of subjects, referred to as the *within-subject pool* setting. Next, we evaluated how each algorithm performed when classifying out-of-distribution data, where the training and testing datasets came from different pools of subjects. We refer to this setting with the term *between-subject pool*. Finally, we investigated how increasing amounts of training data from a single subject affected each algorithm's predictions on the same subject and refer to this setting with the term *within-subject*.

Experiment 1 Our first experiment seeks to address the question *how does each model perform within the subject pool it was trained on?* To answer this question, we trained and tested each algorithm on data from $N \in \{1, 3, 6, 9\}$ subjects out of the first ten subjects. The N subjects were randomly sampled without replacement. We tuned hyperparameters over three random seeds. We then conducted an additional 20 experimental runs using the tuned hyperparameters and report performance metrics for these final 20 independent runs.

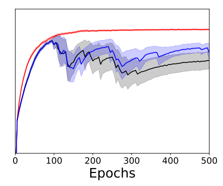
Due to the nature of the EEG data, we would expect that increasing the number of subjects in the training dataset would decrease prediction accuracy. This is because the model needs to account not only for variability that exists within an individual subject but also for variability between different subjects. We are interested to see whether sparsity can help to counteract this effect. Indeed, for many



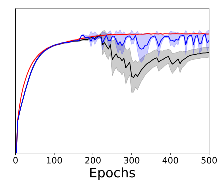
(a) 1 Subject



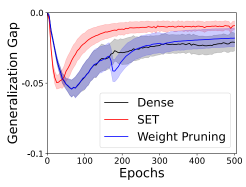
(b) 3 Subjects



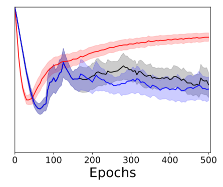
(c) 6 Subjects



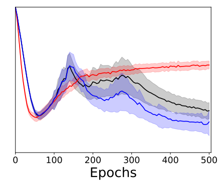
(d) 9 Subjects



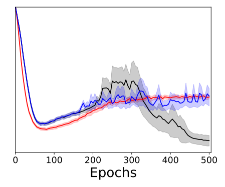
(e) 1 Subject



(f) 3 Subjects



(g) 6 Subjects



(h) 9 Subjects

Figure 2: Experiment 1 within-subject pool results. **Top Row**: Test accuracy over 20 independent runs. As more subjects are added to the training dataset, the performance of SET stays consistently high with low variance. **Bottom Row**: Generalization gap (i.e., test - train accuracy) over 20 independent runs. SET is able to generalize more efficiently than either DN or WP. The final performance of SET is always statistically significantly higher than that of the baseline DN. **Both**: Solid lines denote mean performance with shaded regions denoting 95% bootstrapped confidence intervals.

of the highest performing submissions in the Kaggle competition, a separate network was trained for each subject[3]. Nevertheless, a usable and reliable BCI system should be capable of being trained on multiple subjects data in order to reduce the training burden on any one individual.

Experiment 2 Our second experiment addresses the question *can each model efficiently generalize to the data of new, held-out subjects?* For this experiment we took the tuned-and-trained networks from experiment 1 and tested how they performed on data from subjects 11 and 12. We would expect each model to perform worse in this setting compared to experiment 1, but we are interested in quantifying this discrepancy and determining if sparsity helps to negate it. This study will allow us to better understand who training data should be collected from for real-world BCI systems – the end user or other participants. Improving generalization to new participants would be of great benefit to BCI systems, as it is often challenging to have the end-user come in for multiple training sessions.

Experiment 3 Our final experiment seeks to address the question *how does each algorithm perform with differing levels of training data for a single subject?* This experiment allows us to better understand how each algorithm may behave in a real-world setting where the system could potentially learn and adapt to the behaviour of the end-user during deployment. For this experiment, data from a single randomly sampled subject was split into training, validation, and test sets. The training set was composed of $p \in \{10\%, 20\%, 30\%, 50\%, 70\%, 80\%\}$ of the subject's data. The validation set was composed of 10% of the training set, and the test set was composed of the remaining data. An increasing amount of training data should improve the performance of each algorithm, with diminishing returns. A well-performing algorithm, especially one suited for a deployable BCI system, should reach reasonable performance with a moderate amount of training data. We repeated this experiment 15 times with different random seeds, tuning hyperparameters and reporting performance over all 15 seeds.

Evaluation metrics We evaluated model performance using prediction accuracy, precision and recall. Similar to Novak et al. [2018] and Liu et al. [2019a], we quantified generalization using the *generalization gap* – the difference between accuracy on the testing and training datasets. If a model were to perfectly generalize to unseen data, the generalization gap would be zero. We report mean evaluation metrics across runs with bootstrap confidence intervals. We used 10,000 resamples, batch sizes of 10, and 5% significance to construct confidence intervals, unless otherwise specified.

[3]`https://www.kaggle.com/competitions/grasp-and-lift-eeg-detection/leaderboard`

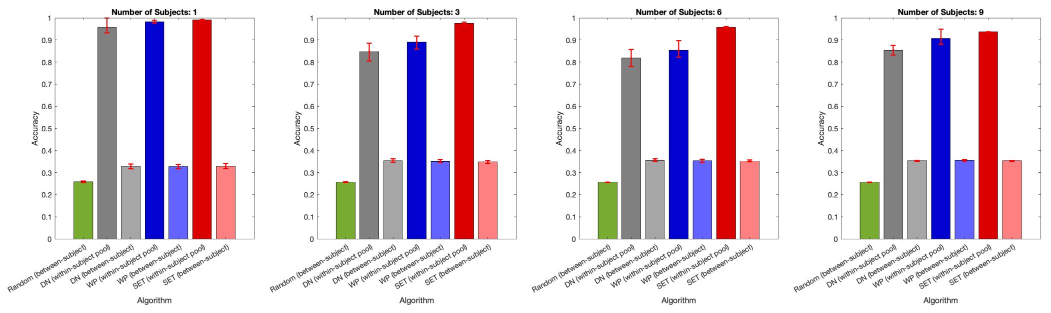


Figure 3: Accuracy of the random predictions, dense network (DN), weight pruning (WP), and sparse evolutionary training (SET) algorithms for experiment 1 (within-subject pool) and experiment 2 (between-subject pools) over 20 random seeds. All network architectures show a steep decline in accuracy when predicting on out-of-distribution data from held-out subjects (i.e., the between-subject bars). Error bars denote 95% bootstrapped confidence intervals with 10,000 resamples. The random (green bar) data is pulled from our experiment 2 results (Table 7), although experiment 1 obtained similar results (Table 4). Corresponding tables are outlined in Appendix C.1 and C.2

5 Results

In this section we present the results of our empirical study for each experimental setting. Our code repository is open-sourced and available at `https://github.com/lpetrich/SparsEEG`.

5.1 Experiment 1: within-subject pool

The top row of Figure 2 shows the test accuracy for each algorithm over 20 independent runs with training/test data from 1, 3, 6, and 9 randomly sampled subjects. WP performed similarly to DN in terms of prediction accuracy. In contrast, SET generally improved prediction accuracy compared to DN and learned noticeably faster than both DN and WP. SET also exhibited statistically significantly higher precision and recall than DN, discussed in Appendix C.1. Accuracy decreased for all algorithms when more subjects were added into the training data. We hypothesize that this is because each algorithm had to better generalize among all subjects' conditional data distributions[4].

Whereas SET exhibited low variance in prediction accuracy with tight confidence intervals, DN and WP exhibited severe degradation in prediction accuracy at differing intervals throughout training as seen in the top row of Figure 2. Notably, these dips are not apparent when using full-batch updating and happen across all classes (that is, the dips are not due to poor prediction accuracy for a single class). Such a phenomenon reduces the utility of both these algorithms to real-life BCI systems and is further discussed in Appendix B.

The bottom row of Figure 2 outlines the generalization gap for each algorithm. SET often exhibited a statistically significantly lower magnitude generalization gap than either DN or WP. WP and DN exhibited a similar magnitude generalization gap, except when training on 9 subjects. For all algorithms, the generalization gap increased with an increasing number of training subjects. Overall, SET obtains the best prediction accuracy and performance stability across all settings suggesting that it is a helpful training regime when adding more subjects data into the training pool.

5.2 Experiment 2: between-subject pools

Figure 3 shows the test accuracy of random predictions, DN, WP, and SET over 20 random seeds for both experiment 1 and experiment 2. Overall, we see a steep decline in accuracy when making predictions on held-out subject data, with algorithms performing only marginally better than the random baseline. For the 1 subject case, we see a decrease in prediction accuracy of around two-thirds for all algorithms (comparing the within-subject pool bars with corresponding between-subject bars).

[4]Denote the theoretical distribution of EEG data (X) and labels (y) on this grasp-and-lift task as $p(X, y)$. Then the distribution of EEG data from a single subject S is $p(X, y \mid S)$.

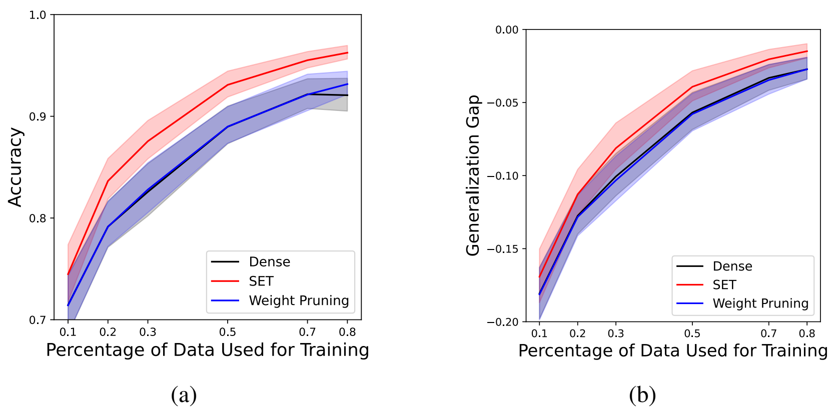


Figure 4: Experiment 3 within-subject results. Prediction **(a)** accuracy and **(b)** generalization gap when trained on increasing amounts of training data. Solid lines denote mean performance with shaded regions denoting 95% bootstrap confidence intervals. **(a)** SET attained the highest accuracy for increasing dataset sizes; **(b)** each algorithm exhibited a similar generalization gap.

All algorithms also exhibited a significant decrease in precision and recall (see Tables 5 and 6 in Appendix C.2). In many cases DN did not exhibit precision/recall statistically significantly higher than the random baseline. WP never exhibited precision/recall statistically significantly higher than DN, and SET only exhibited higher precision/recall than DN for one dataset-label pair.

Notably, this decline in prediction accuracy is not due to the held-out subjects having poor quality data. We verified that each algorithm could learn on these subjects' data, each reaching over 90% prediction accuracy (see Appendix C.2). Overall, DN, SET, and WP each performed approximately equally when predicting on held-out subject data, with a slight increase in accuracy moving from the 1 to 3 subject setting (approximately 32% to 35%).

Considering these results in tandem with those of experiment 1, we conclude that data from the end-user should be included when training a BCI control system, otherwise there may be a significant loss in model performance. Adding sparsity into the network does not appear to help reduce the drop in performance when testing on subjects not included in the training set. Hence, BCI systems may be useful when learning in an online manner during deployment, where the system learns and adapts to it's specific user. This is the motivation for our next experiment.

5.3 Experiment 3: within-subject

A successful BCI system should be able to continually learn and adapt to the dynamic nature of its user. This experiment allowed us to quantify the gains in performance with increasing data from the end-user. Ideally, a BCI system would be useful even with a low amount of training data, although what defines a model as *useful* will ultimately depend on the end application.

Figure 4a shows the accuracy during learning for increasing amounts of training data. Even with a low amount of data, all algorithms attained reasonable performance. With just 10% of the training data (approximately 27 grasp and lift trials), all algorithms reached a prediction accuracy around 75%. As more training data was introduced, the accuracy of all algorithms continued to increase, as expected. The prediction accuracy of SET increased more quickly than that of DN and WP and was statistically significantly higher for datasets composed of more than 10% of a subject's data. For the largest dataset sizes, SET attained statistically significantly higher levels of precision and recall than DN (with a significance level of 0.1), further discussed in Appendix C.3.

Figure 4b shows the generalization gap for DN, SET, and WP. Overall, the gap shrinks with increasing training data, indicating that all algorithms can better generalize with more training data. When 10% of the data was used for training, the generalization gap was around 20% across algorithms – indicating overfitting. When more than 70% of the data was used for training, the generalization gap was less than 5%, indicating that the algorithms could generalize to unseen data effectively. All three algorithms exhibited a similar generalization gap for increasing dataset sizes.

6 Discussion

In this work we evaluated the performance of two sparse neural network algorithms for an EEG-based upper-limb motion classification task. It is well-known that sparse neural networks can reduce the computational and memory requirements of deep-learning systems Hoefler et al. [2021], and our goal was to determine whether such networks are viable for adaptive BCI systems. This is towards our long-term goal of developing feasible brain-computer interface systems. Overall, our results show that sparse neural networks can achieve higher performance and generalization than dense neural networks on one EEG dataset. Furthermore, our experiments provide evidence that sparse neural networks can achieve high performance with low variance during learning, making these networks ideal for real-time BCI systems which adapt to the end-user during deployment.

An open question in the BCI community is *from whom should data be collected* in order to reduce the effects of intra- and inter-subject variability when training a BCI control model [Saha and Baumert, 2020]. We showed how the algorithms under consideration were able to make accurate predictions for subject data on which each algorithm was trained. But, when testing on held-out subject data, each algorithm exhibited a statistically significant, and considerably large, drop in prediction accuracy, precision, and recall. This suggests that such computational models should include some training data from the end-user. Pre-trained models should be either fine-tuned to the end-user prior to deployment, for example through transfer learning [Fahimi et al., 2019], or include data from the end-user while training the model. For best performance, we suggest training a model on as much data as possible from solely the end-user, although we do acknowledge this may not always be a viable option. When training on data from multiple subjects, SET can help to minimize the decline in prediction accuracy.

Overall, our empirical study provides support for our hypothesis that sparsifying a neural network is a viable option for reducing the computational and memory requirements of training an EEG-based model without significant loss of performance. Sparse neural networks can also adapt more quickly during training than their densely connected counterparts. This brings us one step closer towards making a *direct impact on the lives of people living with severe motor impairments through the use of machine learning models trained on brain data.*

6.1 Ethical considerations

As machine learning models become more prevalent in every day life, they are being used to make decisions that affect not only individuals but also society as a whole. Thus, it is important to understand the potential sources of harm that could arise during different stages of the machine learning life cycle. Suresh and Guttag [2021] present a framework for identifying seven sources of harm, three of which we take into consideration for this study.

The WAY-GAL-EEG dataset was gathered for the intent of evaluating whether EEG signals can be used for upper-limb prosthetic device control, however it suffers from representation bias. The twelve participants were all right-handed university students, age 19-35, and the target population (study participants) does not reflect the use population (BCI users). The sample size is also quite small. This means that the population that would end up requiring and using a model such as this for BCI control is underrepresented and the model will likely fail to generalize well for the use population.

The second source of harm we consider is measurement bias. The four classification labels we chose as a proxy is an oversimplification of a much more complex construct - EEG signals. Furthermore, EEG recordings are in themselves an oversimplification of the complex construct that is the human brain. By oversimplifying a complex motor action recorded via EEG signals into four simple labels, we run the risk of our model being a poor reflection of what is actually occurring in the brain.

Lastly, a key goal of this work is to develop efficient and accurate prediction models for use in a real-world BCI control system. Such a system would involve a human user controlling a neuroprosthetic device via real-time EEG signals. Thus, it is imperative that future work takes into consideration any potential harm due to deployment bias. We must ensure that there is a match between the prediction problem the model solves and how it is being used during deployment.

Reproducibility and transparency are an important part of ethical research practices. To this end, we open-sourced all of our code for dataset loading, model training, and evaluation. We also report our final tuned hyperparameters for each model in Appendix C along with tables containing our numerical results for ease of comparison.

7 Limitations and Future Work

Our empirical analysis has a number of limitations. First, we did not correct confidence levels for family-wise error rates. Second, our baseline dense network was at a disadvantage in many of our experiments, due to having a significantly fewer number of hyperparameters than either WP or SET. Third, our analysis is based on the best-case performance with respect to hyperparameters. This is likely an optimistic characterization of algorithm performance during deployment. Lastly, for experiment 2, we only used two subjects as held-out data for testing. A better estimate of prediction accuracy could have been attained with cross-validation.

A limitation of our data pre-processing step is that we removed all samples with overlapping or no event labels. This would make it more difficult to transfer to a real-time control system as there will be a significant amount of out-of-distribution EEG signals being fed into the trained model. One intermediary step before moving to a BCI application should be to re-train these models using the full set of original event labels to see if our conclusions still hold. Future work should also test different cutoff values for the bandpass filter in order to make sure that important information about the motor execution is not being filtered out.

In addition to addressing the aforementioned limitations, future work should focus on studying these algorithms on larger BCI datasets. Many of our conclusions are a direct consequence of training on a small dataset of twelve participants with a reduced number of class labels, and it is unclear if these conclusions would hold with larger datasets. Future work should also consider systematic hyperparameter sensitivity analyses of these algorithms, to understand better how they would perform in real-world scenarios where tuning is impossible. Finally, a long-term goal of this work is direct control of a BCI system, such as a robotic arm, using EEG or similar technology. We hope that our research has laid the groundwork necessary for conducting such an ambitious study in the future.

References

Maged S. Al-Quraishi, Irraivan Elamvazuthi, Siti Asmah Daud, S. Parasuraman, and Alberto Borboni. EEG-Based Control for Upper and Lower Limb Exoskeletons and Prostheses: A Systematic Review. *Sensors*, 18, 2018. doi: 10.3390/s18103342. URL `https://www.mdpi.com/1424-8220/18/10/3342`.

Hamdi Altaheri, Ghulam Muhammad, Mansour Alsulaiman, Syed Umar Amin, Ghadir Ali Altuwaijri, Wadood Abdul, Mohamed A. Bencherif, and Mohammed Faisal. Deep Learning Techniques for Classification of Electroencephalogram (EEG) Motor Imagery (MI) Signals: A Review. *Neural Computing and Applications*, 35:14681–14722, 2023. doi: 10.1007/s00521-021-06352-5. URL `https://link.springer.com/10.1007/s00521-021-06352-5`.

Mahdi Bamdad, Homayoon Zarshenas, and Mohammad A. Auais. Application of BCI Systems in Neurorehabilitation: A Scoping Review. *Disability and Rehabilitation: Assistive Technology*, 10: 355–364, 2015. doi: 10.3109/17483107.2014.961569. URL `http://www.tandfonline.com/doi/full/10.3109/17483107.2014.961569`.

Abdelkader Nasreddine Belkacem, Nuraini Jamil, Jason A. Palmer, Sofia Ouhbi, and Chao Chen. Brain Computer Interfaces for Improving the Quality of Life of Older Adults and Elderly Patients. *Frontiers in Neuroscience*, 14, 2020. URL `https://www.frontiersin.org/articles/10.3389/fnins.2020.00692`.

Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep Rewiring: Training Very Sparse Deep Networks. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Andrea Biasiucci, Benedetta Franceschiello, and Micah M. Murray. Electroencephalography. *Current Biology*, 29:R80–R85, 2019. doi: 10.1016/j.cub.2018.11.052. URL `https://www.sciencedirect.com/science/article/pii/S0960982218315513`.

Anna Chambers and Simon Rumpel. A Stable Brain from Unstable Components: Emerging Concepts, Implications for Neural Computation. *Neuroscience*, 357, 2017. doi: 10.1016/j.neuroscience.2017.06.005.

Benoni Edin, Klaus Greff, and Will Cukierski. Grasp-and-Lift EEG Detection, 2015. URL `https://kaggle.com/competitions/grasp-and-lift-eeg-detection`.

P Erdős and A Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

Fatemeh Fahimi, Zhuo Zhang, Wooi Boon Goh, Tih-Shi Lee, Kai Keng Ang, and Cuntai Guan. Inter-subject transfer learning with an end-to-end deep convolutional neural network for EEG-based BCI. *Journal of Neural Engineering*, 16(2):026007, January 2019. ISSN 1741-2552. doi: 10.1088/1741-2552/aaf3f6. URL `https://dx.doi.org/10.1088/1741-2552/aaf3f6`. Publisher: IOP Publishing.

Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *Proceedings of the 7th International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJl-b3RcF7`.

Sina Ghiassian, Banafsheh Rafiee, Yat Long Lo, and Adam White. Improving Performance in Reinforcement Learning by Breaking Generalization in Neural Networks. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2020.

Bram Grooten, Ghada Sokar, Shibhansh Dohare, Elena Mocanu, Matthew E. Taylor, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Automatic Noise Filtering with Dynamic Sparse Training in Deep Reinforcement Learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2023.

Erwin Habibzadeh Tonekabony Shad, Marta Molinas, and Trond Ytterdal. Impedance and Noise of Passive and Active Dry EEG Electrodes: A Review. *IEEE Sensors Journal*, 20:14565–14577, 2020. doi: 10.1109/JSEN.2020.3012394. URL `https://ieeexplore.ieee.org/document/9149885`.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning Both Weights and Connections for Efficient Neural Networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2015.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2023. URL `http://github.com/google/flax`.

Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks, January 2021. URL `http://arxiv.org/abs/2102.00554`. arXiv:2102.00554 [cs].

Anthony J. G. D. Holtmaat, Joshua T. Trachtenberg, Linda Wilbrecht, Gordon M. Shepherd, Xiaoqun Zhang, Graham W. Knott, and Karel Svoboda. Transient and Persistent Dendritic Spines in the Neocortex In Vivo. *Neuron*, 45:279–291, 2005. doi: https://doi.org/10.1016/j.neuron.2005.01.003. URL `https://www.sciencedirect.com/science/article/pii/S0896627305000048`.

Steven A. Janowsky. Pruning versus clipping in neural networks. *Phys. Rev. A*, 39:6600–6603, Jun 1989. doi: 10.1103/PhysRevA.39.6600. URL `https://link.aps.org/doi/10.1103/PhysRevA.39.6600`.

Xiao Jiang, Gui-Bin Bian, and Zean Tian. Removal of Artifacts from EEG Signals: A Review. *Sensors (Basel, Switzerland)*, 19(5):987, February 2019. ISSN 1424-8220. doi: 10.3390/s19050987. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6427454/`.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations*, 2014.

Joo Hyung Lee, Wonpyo Park, Nicole Mitchell, Jonathan Pilault, Johan S. Obando-Ceron, Han-Byul Kim, Namhoon Lee, Elias Frantar, Yun Long, Amir Yazdanbakhsh, Shivani Agrawal, Suvinay Subramanian, Xin Wang, Sheng-Chun Kao, Xingyao Zhang, Trevor Gale, Aart J. C. Bik, Woohyun Han, Milen Ferev, Zhonglin Han, Hong-Seok Kim, Yann Dauphin, Karolina Dziugaite, Pablo Samuel Castro, and Utku Evci. Jaxpruner: A concise library for sparsity research, 2023. URL `https://github.com/google-research/jaxpruner`.

Shiwei Liu. Learning Sparse Neural Networks for Better Generalization. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2020. URL `https://doi.org/10.24963/ijcai.2020/735`.

Shiwei Liu, Decebal Constantin Mocanu, and Mykola Pechenizkiy. On Improving Deep Learning Generalization with Adaptive Sparse Connectivity. *ICML 2019 Workshop on Understanding and Improving Generalization in Deep Learning*, 2019a. URL `http://arxiv.org/abs/1906.11626`.

Vincent Liu, Raksha Kumaraswamy, Lei Le, and Martha White. The utility of sparse representations for control in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4384–4391, 2019b.

Fabien Lotte, Marco Congedo, Anatole Lécuyer, Fabrice Lamarche, and Bruno Arnaldi. A Review of Classification Algorithms for EEG-Based Brain–Computer Interfaces. *Journal of Neural Engineering*, 4:R1–R13, 2007. doi: 10.1088/1741-2560/4/2/R01. URL `https://iopscience.iop.org/article/10.1088/1741-2560/4/2/R01`.

Matthew D. Luciw, Ewa Jarocka, and Benoni B. Edin. Multi-Channel EEG Recordings During 3,936 Grasp and Lift Trials with Varying Weight and Friction. *Scientific Data*, 1, 2014. doi: 10.1038/sdata.2014.47. URL `https://www.nature.com/articles/sdata201447`.

Baraka Maiseli, Abdi T. Abdalla, Libe V. Massawe, Mercy Mbise, Khadija Mkocha, Nassor Ally Nassor, Moses Ismail, James Michael, and Samwel Kimambo. Brain–Computer Interface: Trend, Challenges, and Threats. *Brain Informatics*, 10, 2023. doi: 10.1186/s40708-023-00199-3. URL `https://braininformatics.springeropen.com/articles/10.1186/s40708-023-00199-3`.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable Training of Artificial Neural Networks with Adaptive Sparse Connectivity Inspired by Network Science. *Nature Communications*, 9, 2018. doi: 10.1038/s41467-018-04316-3. URL `http://arxiv.org/abs/1707.04780`.

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and Generalization in Neural Networks: An Empirical Study. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Yangchen Pan, Kirby Banman, and Martha White. Fuzzy Tiling Activations: A Simple Approach to Learning Sparse Representations Online. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

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.

Simanto Saha and Mathias Baumert. Intra- and Inter-subject Variability in EEG-Based Sensorimotor Brain Computer Interface: A Review. *Frontiers in Computational Neuroscience*, 13, 2020. ISSN 1662-5188. URL `https://www.frontiersin.org/articles/10.3389/fncom.2019.00087`.

Dan Stettler, Homare Yamahachi, Wu Li, Winfried Denk, and Charles Gilbert. Axons and Synaptic Boutons Are Highly Dynamic in Adult Visual Cortex. *Neuron*, 49:877–87, 2006. doi: 10.1016/j.neuron.2006.02.018.

Harini Suresh and John V. Guttag. A Framework for Understanding Sources of Harm throughout the Machine Learning Life Cycle. In *Equity and Access in Algorithms, Mechanisms, and Optimization*, pages 1–9, October 2021. doi: 10.1145/3465416.3483305. URL `http://arxiv.org/abs/1901.10002`. arXiv:1901.10002 [cs, stat].

Michael Zhu and Suyog Gupta. To Prune, or Not to Prune: Exploring the Efficacy of Pruning for Model Compression. In *Procedings of the 6th International Conference on Learning Representations Workshop Track*, 2018. URL `https://openreview.net/forum?id=Sy1iIDkPM`.

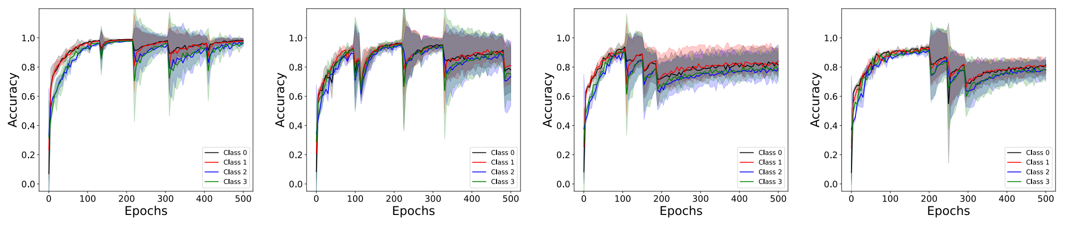
A Work Distribution

- ☑ Process dataset and extract event labels (Laura)
- ☑ Exploratory data analysis (Laura)
- ☑ Plan experimental setup (Samuel and Laura)
- ☑ Plan codebase architecture (Samuel)
- ☑ Implement codebase (Samuel)
    - ☑ Experiment and evaluation pipelines (Samuel)
    - ☑ Data loading and pre-processing (Laura)
    - ☑ Implementation of baseline algorithms (Samuel)
    - ☑ Implementation of Weight pruning and SET algorithms (Samuel)
    - ☑ Experimental configs for Compute Canada (Samuel)
- ☑ Run experiments as outlined in the *Methods* section (Samuel and Laura)
- ☑ Analyze data as it comes back from experiments (Samuel and Laura)
- ☑ Paper planning and outline (Samuel and Laura)
- ☑ Create figures for final project report (Samuel and Laura)
- ☑ Prepare final project presentation (Samuel and Laura)
- ☑ Write final project report (Samuel and Laura)

B Analysis of Batch Size

As discussed in the main text, both the dense network baseline (DN) and the weight pruning network (WP) exhibited behaviour indicative of interference or similar phenomena when trained with mini-batch style updates. Other works have noticed a similar trend, where dense networks seem to be affected by catastrophic forgetting and interference more severely than sparse networks are [Ghiassian et al., 2020, Pan et al., 2021, Liu et al., 2019b]. During training, we noticed that the performance of DN and WP consistently improved. Suddenly, the performance would drastically drop, after which it continued to improve again. This tended to happen multiple times during training.

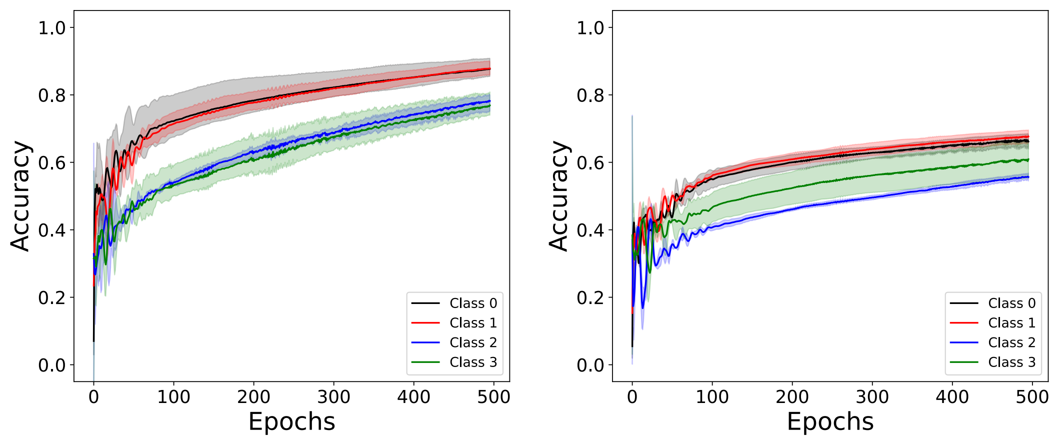
We first suspected that this behaviour was due to the algorithm ignoring one or more classes entirely and focusing its predictions on other classes. Figure 5 shows the prediction accuracy of DN across class labels. Here, the DN was trained using mini-batch style gradient updates for datasets consisting of 1, 3, 6, and 9 subjects across 3 random seeds, with hyperparameter tuning as described in Section 4.2. As can be seen, our hypothesis was left unsubstantiated. The dips in performance happen across all classes and are not due to the algorithm focusing its predictions on a subset of classes.



(a) 1 Subject (b) 3 Subjects (c) 6 Subjects (d) 9 Subjects

Figure 5: Per-class prediction accuracy for DN using mini-batch updates of size 8192 on training data generated from 1, 3, 6, and 9 random subjects across 3 random seeds. At seemingly deterministic intervals, the prediction accuracy drops across all classes. Solid lines denote mean prediction accuracy across runs with shaded regions denoting 95% bootstrap confidence intervals.

We noticed that when taking full batch updates, DN no longer exhibited such behaviour (we did not further investigate this issue for WP). Figure 6 shows the class-wise accuracy of DN when trained



(a) 1 Subject (b) 3 Subjects

Figure 6: Per-class prediction accuracy for the baseline dense network using full batch gradient updates on training data generated from **(a)** a single random subject over 3 random seeds and **(b)** 3 random subjects over 3 random seeds. Solid lines denote mean accuracy for each class with shaded regions denoting 95% bootstrap confidence intervals with 10,000 resamples. Data is smoothed over 7 epochs for readability.

using full-batch updates on the data of 1 or 3 randomly sampled subjects over 3 random seeds. Due to its significant memory footprint, we did not analyze the performance of DN when using full-batch updates for datasets composed of 6 or 9 randomly sampled subjects. Compared to using mini-batch updates, DN did not exhibit severe dips in prediction accuracy when using full-batch updates.

Figure 7 outlines the performance of each algorithm on the two held-out subjects' data from experiment 2 using mini-batch updating. Considering these plots, we see that DN and WP exhibit low variance performance for subject 11, but that for subject 12 these algorithms exhibit higher variation in performance with small dips in prediction accuracy through time. We speculate that there is some interplay between the data of specific subjects and the batch size which is causing these dips in prediction accuracy. We leave further investigation of this phenomenon to future work.

C Further Experimental Discussion

In this section, we provide supplementary details and analysis on our experiments and results.

C.1 Experiment 1

Tables 2 and 3 outline the 95% bootstrap confidence intervals for precision and recall of the random predictor, DN, SET, and WP over 20 independent runs with different random seeds for experiment 1. The random predictor predicts classes randomly according to their relative frequency in the dataset. All four predictors saw the same 20 random test sets. Overall, SET attained the highest precision/recall, exhibiting precision/recall that was statistically significantly higher than that attained by both DN and the random predictor baseline for datasets consisting of 3, 6, or 9 randomly selected subjects.

Table 4 outlines the accuracy attained by the random predictor, DN, SET, and WP over the 20 independent runs for experiment 1. As more subjects were added to the pool of training data, overall mean test accuracy tended to decrease. This was most noticeable for SET. The accuracy of SET decreased statistically significantly for each additional subject introduced into the dataset. The overall trend is similar for DN and WP, but with a reduced levels of statistical significance. Nevertheless, SET still attained the highest prediction accuracy for each additional subject added to the training dataset, and exhibited statistically significantly higher accuracy than either DN or WP in many cases.

For this experiment, the tuned hyperparameters for each algorithm were as follows. All algorithms had a tuned stepsize of 0.001. WP had a tuned sparsity level of 50% except for datasets of 3 subjects, for which the tuned sparsity level was 25%. For WP, weights were pruned at epoch 167. For SET, weights were pruned every 5 epochs, except for datasets of 3 subjects, for which weights were pruned every epoch. At these intervals, 15% of SET's weights were pruned except for datasets of 1 subject, for which 30% of weights were pruned. SET had a tuned sparsity level of 25%.

Algorithm	Dense				SET			
Subjects \ Class	0	1	2	3	0	1	2	3
1	(0.960, 0.992)	(0.942, 0.996)	(0.910, 0.997)	(0.914, 0.997)	(0.990, 0.996)	(0.988, 0.995)	(0.984, 0.992)	(0.985, 0.992)
3	(0.862, 0.922)	(0.826, 0.901)	(0.740, 0.856)	(0.770, 0.876)	**(0.979, 0.985)**	**(0.975, 0.981)**	**(0.967, 0.975)**	**(0.967, 0.976)**
6	(0.845, 0.896)	(0.806, 0.876)	(0.716, 0.820)	(0.736, 0.833)	**(0.964, 0.969)**	**(0.956, 0.963)**	**(0.947, 0.955)**	**(0.948, 0.956)**
9	(0.871, 0.904)	(0.849, 0.888)	(0.786, 0.849)	(0.808, 0.860)	**(0.946, 0.951)**	**(0.938, 0.943)**	**(0.919, 0.925)**	**(0.927, 0.932)**
	Weight Pruning				Random			
1	(0.981, 0.992)	(0.979, 0.990)	(0.970, 0.986)	(0.971, 0.987)	(0.300, 0.312)	(0.253, 0.271)	(0.207, 0.221)	(0.205, 0.219)
3	(0.894, 0.940)	(0.876, 0.932)	(0.817, 0.907)	(0.831, 0.911)	(0.309, 0.321)	(0.252, 0.265)	(0.210, 0.217)	(0.210, 0.216)
6	(0.866, 0.919)	(0.840, 0.906)	(0.775, 0.870)	(0.794, 0.874)	(0.309, 0.317)	(0.255, 0.261)	(0.213, 0.217)	(0.213, 0.217)
9	**(0.909, 0.959)**	**(0.890, 0.954)**	**(0.853, 0.946)**	**(0.866, 0.946)**	(0.314, 0.318)	(0.254, 0.258)	(0.214, 0.216)	(0.213, 0.215)

Table 2: 95% bootstrap confidence intervals for **precision** over 20 independent runs for the final learned models from experiment 1. Bold indicates statistical significance from the corresponding performance of the dense and random baselines.

Algorithm	Dense				SET			
Subjects \ Class	0	1	2	3	0	1	2	3
1	(0.952, 0.998)	(0.948, 0.992)	(0.907, 0.996)	(0.917, 0.999)	(0.991, 0.996)	(0.988, 0.995)	(0.981, 0.991)	(0.985, 0.993)
3	(0.819, 0.899)	(0.829, 0.906)	(0.778, 0.876)	(0.779, 0.880)	**(0.978, 0.984)**	**(0.974, 0.981)**	**(0.966, 0.975)**	**(0.971, 0.977)**
6	(0.791, 0.872)	(0.795, 0.870)	(0.756, 0.840)	(0.766, 0.849)	**(0.964, 0.970)**	**(0.957, 0.963)**	**(0.942, 0.952)**	**(0.951, 0.958)**
9	(0.845, 0.889)	(0.841, 0.884)	(0.811, 0.859)	(0.823, 0.875)	**(0.947, 0.952)**	**(0.934, 0.939)**	**(0.922, 0.928)**	**(0.927, 0.931)**
	Weight Pruning				Random			
1	(0.983, 0.992)	(0.976, 0.990)	(0.968, 0.985)	(0.974, 0.988)	(0.300, 0.314)	(0.254, 0.275)	(0.207, 0.223)	(0.206, 0.220)
3	(0.870, 0.936)	(0.873, 0.932)	(0.836, 0.908)	(0.842, 0.916)	(0.308, 0.321)	(0.251, 0.265)	(0.211, 0.216)	(0.211, 0.217)
6	(0.835, 0.906)	(0.830, 0.904)	(0.804, 0.877)	(0.811, 0.888)	(0.310, 0.318)	(0.255, 0.261)	(0.213, 0.218)	(0.212, 0.217)
9	(0.893, 0.967)	(0.882, 0.955)	**(0.867, 0.937)**	(0.873, 0.948)	(0.314, 0.317)	(0.255, 0.257)	(0.212, 0.215)	(0.214, 0.217)

Table 3: 95% bootstrap confidence intervals for **recall** over 20 independent runs for the final learned models from experiment 1. Bold indicates statistical significance from the corresponding performance of the dense and random baselines.

Subjects \ Algorithm	Dense	SET	Weight Pruning	Random
1	0.957, (0.933, 0.999)	0.990, (0.987, 0.993)	0.982, (0.976, 0.989)	0.253, (0.235, 0.271)
3	0.847, (0.806, 0.885)	**0.976, (0.973, 0.980)**	0.890, (0.858, 0.918)	0.270, (0.253, 0.287)
6	0.819, (0.780, 0.857)	**0.958, (0.955, 0.961)**	0.853, (0.822, 0.897)	0.267, (0.248, 0.287)
9	0.854, (0.832, 0.876)	**0.937, (0.935, 0.938)**	**0.908, (0.880, 0.949)**	0.267, (0.248, 0.286)

Table 4: Mean with 95% bootstrap confidence intervals for **accuracy** over 20 independent runs for the final learned models from experiment 1. Bold indicates statistical significance from the corresponding performance of the dense and random baselines.

C.2 Experiment 2

In this section we provide additional results for experiment 2. One important point with respect to the results of our second experiment is the quality of data from the two held-out test subjects, subjects 11 and 12. If these subjects' data is of sufficiently poor quality, this may explain the significant drop in prediction accuracy between testing on subjects whose data was trained on (experiment 1) and testing on held-out subjects data (experiment 2), shown in Figure 3. In other words, if learning on these subjects' data is not possible, then we should not expect our trained models to generalize well to their data in experiment 2.

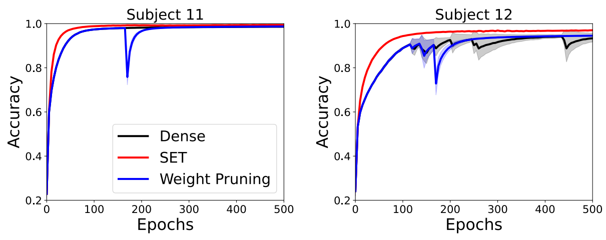


Figure 7: Mean prediction accuracy with 95% bootstrap confidence intervals when training and testing on the data from subjects 11 (left) and 12 (right) separately. Each algorithm can attain high prediction accuracy when trained on the data from each of these subjects separately.

To verify that learning was possible on the data from subjects 11 and 12, we trained DN, SET, and WP on the data from subjects 11 and 12 separately. We used the tuned hyperparameters from experiment 1 and stratified splitting to create train, test, and validation datasets (80%, 10%, and 10%, respectively) from each of these subject's data. Figure 7 shows the mean prediction accuracy with 95% bootstrap confidence intervals over 15 random seeds for each (algorithm, subject) pair. As can be seen, learning is possible on these subjects' data.

Tables 5 and 6 outline the 95% bootstrap confidence intervals for precision and recall over 20 independent runs for the random predictor, DN, WP, and SET for experiment 2. Although SET and WP often performed better than random in terms of precision and recall, overall performance is low. Both SET and WP often did not reach levels of precision and recall which where statistically significantly higher than those attained by DN.

Table 7 outlines the mean accuracy with 95% bootstrap confidence intervals for DN, SET, WP, and the random prediction baseline over 20 independent runs for experiment 2. In terms of accuracy, all algorithms performed statistically significantly better than random. Yet, the test accuracy attained by all three algorithms was only marginally higher than the test accuracy attained by the random predictor.

Algorithm	Dense				SET			
Class / Subjects	0	1	2	3	0	1	2	3
1	(0.406, 0.456)	(0.296, 0.323)	(0.223, 0.250)	(0.304, 0.327)	(0.410, 0.459)	(0.299, 0.327)	(0.224, 0.253)	(0.301, 0.322)
3	(0.452, 0.492)	(0.317, 0.336)	(0.245, 0.259)	(0.311, 0.323)	(0.454, 0.491)	(0.320, 0.339)	(0.243, 0.255)	(0.312, 0.321)
6	(0.461, 0.482)	(0.313, 0.331)	(0.246, 0.255)	(0.311, 0.319)	(0.460, 0.478)	(0.314, 0.329)	(0.243, 0.252)	(0.317, 0.323)
9	(0.466, 0.475)	(0.312, 0.318)	(0.239, 0.246)	(0.313, 0.318)	(0.467, 0.472)	(0.312, 0.317)	(0.238, 0.243)	**(0.319, 0.324)**
	Weight Pruning				Random			
1	(0.406, 0.455)	(0.297, 0.324)	(0.225, 0.253)	(0.300, 0.323)	(0.297, 0.297)	(0.274, 0.275)	(0.215, 0.216)	(0.214, 0.215)
3	(0.443, 0.484)	(0.317, 0.334)	(0.245, 0.259)	(0.310, 0.323)	(0.297, 0.297)	(0.274, 0.275)	(0.215, 0.216)	(0.214, 0.215)
6	(0.457, 0.476)	(0.313, 0.332)	(0.240, 0.251)	(0.309, 0.322)	(0.297, 0.297)	(0.274, 0.275)	(0.215, 0.216)	(0.214, 0.215)
9	(0.468, 0.478)	(0.310, 0.321)	(0.240, 0.246)	(0.316, 0.327)	(0.297, 0.297)	(0.274, 0.275)	(0.215, 0.216)	(0.214, 0.215)

Table 5: 95% bootstrap confidence intervals for **precision** over 20 independent runs for the final learned models from experiment 2. Random prediction baselines are computed over 30 runs. Bold indicates statistical significance from the corresponding performance of the dense and random baselines.

Algorithm	Dense				SET			
Class / Subjects	0	1	2	3	0	1	2	3
1	(0.402, 0.443)	(0.278, 0.315)	(0.240, 0.306)	(0.257, 0.327)	(0.400, 0.447)	(0.279, 0.312)	(0.242, 0.305)	(0.261, 0.332)
3	(0.491, 0.523)	(0.268, 0.305)	(0.233, 0.266)	(0.313, 0.358)	(0.453, 0.480)	(0.264, 0.297)	**(0.269, 0.300)**	(0.315, 0.354)
6	(0.524, 0.547)	(0.263, 0.285)	(0.220, 0.235)	(0.331, 0.353)	(0.495, 0.512)	(0.266, 0.293)	(0.236, 0.254)	(0.331, 0.358)
9	(0.529, 0.544)	(0.268, 0.277)	(0.212, 0.221)	(0.337, 0.347)	(0.520, 0.533)	(0.270, 0.277)	(0.223, 0.232)	(0.334, 0.342)
	Weight Pruning				Random			
1	(0.399, 0.441)	(0.275, 0.311)	(0.250, 0.311)	(0.257, 0.325)	(0.296, 0.297)	(0.274, 0.275)	(0.215, 0.216)	(0.214, 0.215)
3	(0.493, 0.519)	(0.268, 0.292)	(0.236, 0.267)	(0.309, 0.348)	(0.296, 0.297)	(0.274, 0.275)	(0.215, 0.216)	(0.214, 0.215)
6	(0.508, 0.535)	(0.264, 0.288)	(0.221, 0.237)	(0.327, 0.358)	(0.296, 0.297)	(0.274, 0.275)	(0.215, 0.216)	(0.214, 0.215)
9	(0.521, 0.541)	(0.271, 0.280)	(0.220, 0.229)	(0.331, 0.352)	(0.296, 0.297)	(0.274, 0.275)	(0.215, 0.216)	(0.214, 0.215)

Table 6: 95% bootstrap confidence intervals for **recall** over 20 independent runs for the final learned models from experiment 2. Random prediction baselines are computed over 30 runs. Bold indicates statistical significance from the corresponding performance of the dense and random baselines.

Algorithm / Subjects	Dense	SET	Weight Pruning	Random
1	0.328 (0.317, 0.338)	0.329 (0.318, 0.340)	0.327 (0.317, 0.337)	0.258 (0.255, 0.261)
3	0.354 (0.347, 0.362)	0.348 (0.342, 0.354)	0.351 (0.345, 0.358)	0.257 (0.255, 0.258)
6	0.356 (0.351, 0.362)	0.352 (0.348, 0.357)	0.354 (0.346, 0.360)	0.256 (0.255, 0.256)
9	0.354 (0.351, 0.356)	0.353 (0.351, 0.354)	0.356 (0.350, 0.359)	0.256 (0.255, 0.256)

Table 7: Mean with 95% bootstrap confidence intervals for **accuracy** over 20 independent runs for the final learned models in experiment 2. Random prediction baselines are computed over 30 runs.

C.3 Experiment 3

In this section, we provide additional analysis for experiment 3. Figure 8 outlines the average per-class accuracy (i.e. recall) throughout training for DN, SET, and WP for differing amounts of training data over 15 random seeds. As the algorithms were trained on more data, each attained higher recall.

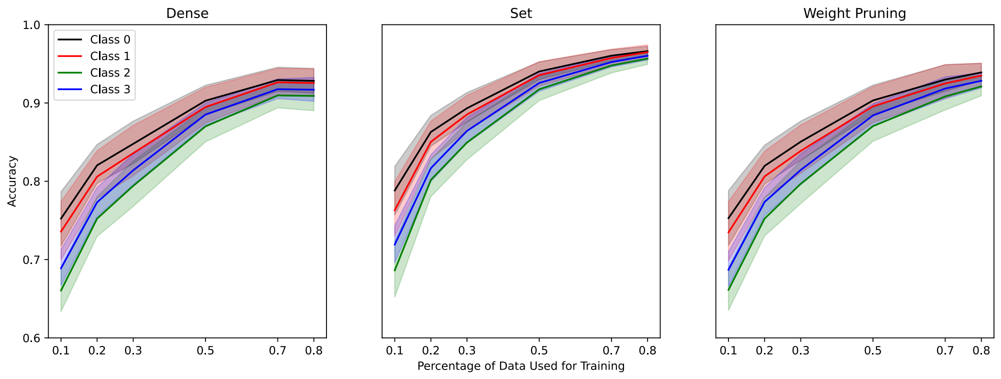


Figure 8: Sensitivity to training set size. Solid lines denote the mean class-wise accuracy with shaded regions denoting 95% bootstrap confidence intervals over 15 runs. See Figure 4a for further analysis.

When using more than 70% of the data for training, SET was able to achieve the highest recall out of all three algorithms.

Figure 9 outlines the precision and recall of the final learned model for DN, SET, and WP for differing levels of training data over 15 random seeds. We used a significance level of 0.1 when plotting these curves. For all levels of training data, WP did not attain statistically significantly higher precision or recall than DN. On the other hand, SET was able to attain statistically significantly higher precision and recall than DN for the largest training datasets. For any datasets composed of less than 80% of the subject's training data, the precision and recall of SET was not statistically significantly higher than that of DN.

For this experiment, the tuned hyperparameters for each algorithm were as follows. All algorithms had a tuned stepsize of 0.001. SET had a tuned sparsity level of 25%. For datasets composed of 10% or 20% of a subject's data, SET pruned and re-grew 15% of its weights, while for the remaining dataset sizes it pruned and re-grew 30% of its weights. The tuned pruning interval was 5 epochs for all dataset sizes. For WP, a tuned sparsity level of 25% was used for all dataset sizes, except when using 80% of a subject's data, for which the tuned sparsity level was 50%. For datasets composed of 10% or 20% of a subject's data, WP pruned weights at epoch 333 and at epoch 167 otherwise.

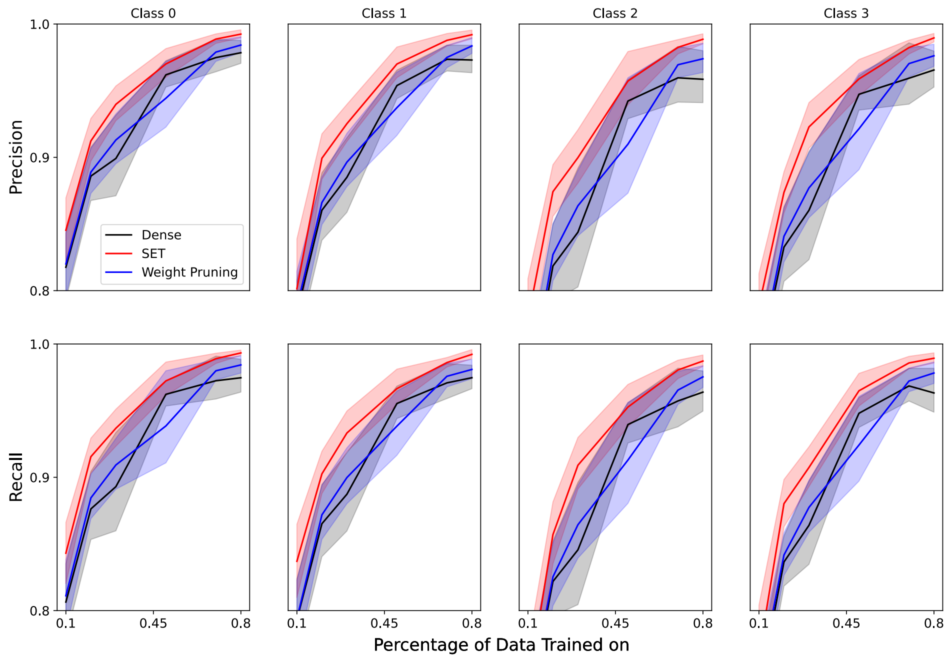


Figure 9: The per-class precision/recall of the final learned model for SET, DN, and WP for differing levels of training data from a single subject over 15 random seeds. Solid lines denote mean performance with shaded regions denoting 90% bootstrap confidence intervals.