NeuraNIL

Model Agnostic Meta Learning for Neural Signal Prediction

"Outer Loopers": Yishu Li, Jania Vandevoorde, Carlos

GitHub: github.com/Yishu-Li/NeuraNIL
Slides: NeuraNIL Presentation Slides

Introduction & Motivation

Intracortical brain-computer interfaces (iBCIs) are devices that directly interact with the brain's cortex to enable communication between the brain and external devices by recording neural activity. They hold significant promise for restoring communication and motor functions in patients with conditions such as ALS, stroke, spinal cord injury, and many others. However, the instability of neural signals across days remains one of the greatest challenges to achieving long-term stable neural decoding. The non-stationarity of neural signals is a significant hurdle in the development of reliable and durable iBCIs.

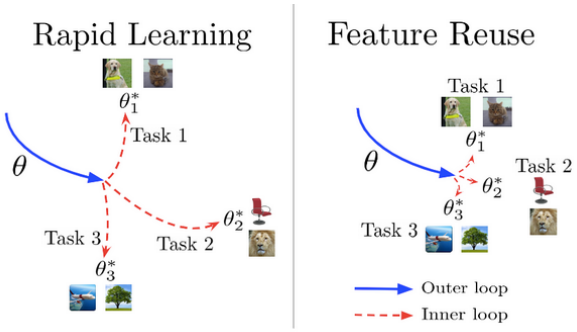
This project proposes the use of the meta-learning approach, ANIL (Almost No Inner Loop), to enable rapid adaptation to the daily variations in neural signals, enhancing the reliability of neural decoding systems across longer time periods. Furthermore, considering the few-shot learning capabilities of meta-learning, this approach is well-suited for adapting to novel motor

imagery types that are not present during the initial training phase. We will adapt this method from Raghu et al. (2020) for application on neural data. This is a structured prediction problem.

Model Agnostic Meta-Learning

In 2017, Finn et al. proposed a model-agnostic meta-learning method (MAML) to train a model that can solve different tasks than those it was trained on. They achieved this and made the models easy to generalize with only a small number of gradient steps and training data needed to solve the new tasks. As a model-agnostic method, this approach is compatible with any model trained using gradient descent.

The training process is divided into two types of parameter updates: the outer loop and inner loop. The outer loop, or the "learner," updates the meta-initialization of the neural network parameters to a setting that enables fast adaptation to new tasks. The inner loop, or the "classifier," takes the outer loop initialization and performs task-specific adaptation over a few labeled samples. In 2019, Raghu et al. conjectured that we can obtain the same rapid learning performance of MAML solely through feature reuse by only updating the last layer of the model during the inner loop. To test this hypothesis, they introduced ANIL (almost no inner loop), a simplified algorithm of MAML that is equally effective but computationally faster. The figures below clarify the difference between the MAML and ANIL methods.



	MAML	ANIL
Outer loop training	$\theta_i \leftarrow \theta_i - \beta \nabla_{\theta_i} \sum_{\tau_i \sim p(\tau)} L_{\tau_i}(f_{\theta_i'})$	$\theta_{i+1} = \theta_i - \alpha \nabla_{\theta_i} \sum_{\tau_i \sim p(\tau)} L_T(w_i'^T \phi_{\theta_i}(x), y)$ $w_{i+1} = w_{i+1} - \nabla_{w_i} \sum_{\tau_i \sim p(\tau)} L_T(w_i'^T \phi_{\theta_i}(x), y)$
Inner loop Adaptation	$\theta_i' = \theta_i - \alpha \nabla_{\theta_i} L_{\tau_i}(f_{\theta_i})$	$w_i' = w_i - \beta \nabla_{w_i} L_T(w_i'^T \phi_{\theta_i}(x), y)$
Model	Any Model	Learner + Task-specific heads

Figure 1. Illustration and equations of Rapid Learning in MAML and Feature Reuse in ANIL

Deep Learning, Non-Linearity, and Neural Decoding

Linear decoders such as the Kalman filter have been widely used for neural decoding to enable individuals with paralysis to control computer mice. However, more work is needed to improve this. Previous research shows that deep learning methods such as multi-layer perceptrons (MLPs) and recurrent neural networks (RNNs) can provide higher performance for the task by inducing

non-linearity into the decoders. However, the problem with these approaches is that training the models requires a lot more data than simple non-linear decoders. Deep learning models are also harder to tune and adapt to new data.

Neural Non-Stationary and Manifold-Based iBCI Decoders:

Previous research has shown the non-stationarity of neural signals: models trained on signals from previous days may suffer from performance degradation or even break after several days or weeks. Many methods have been proposed to address this problem, such as manifold-based decoders. These begin by mapping the neural data to a lower-dimensional latent space. Then, the latent space neural data is aligned to the initial distribution where the decoder was first trained, eliminating the non-stationary and maintaining the decoder performance across days.

Methodology

We want to implement both standard deep learning models ("base models") and our NeuraNIL model based on the ANIL approach from Raghu et al. (2020).

Data

This project uses two different datasets to train and test the base models and our NeuraNIL model:

1) **BrainGate2 Clinical Trial Data:** The BrainGate2 clinical trial data we are going to use is from an ongoing clinical trial where a BrainGate2 participant with advanced ALS is trying to communicate by imagining a set of gestures. The raw signal was recorded as 30000 Hz voltage data, then preprocessing, including re-reference, filtering, threshold crossing, and power calculation will was performed to get binned spiking rate and spiking power features. There are 11 days worth of data with two different gesture sets, for a total of 6 classes. We will treat each day as a new task. For example, we could try to determine which gesture was imagined for day X.
2) **FALCON H2 Dataset:** The 5 FALCON datasets include neural data recorded from humans, non-human primates, and birds. The H2 dataset is a classification task where a human participant imagines writing. The decoder will need to decode the character the participant is trying to write, for a total of 26 classes (one per English letter). The dataset is divided into three parts: Held-in, Held-out, and Minival. The Held-in and Minival sets are from the training period, where the model should be trained and validated, and the Held-out set is from the first several blocks in each day from a testing period, where the model can be fast recalculated for that day.

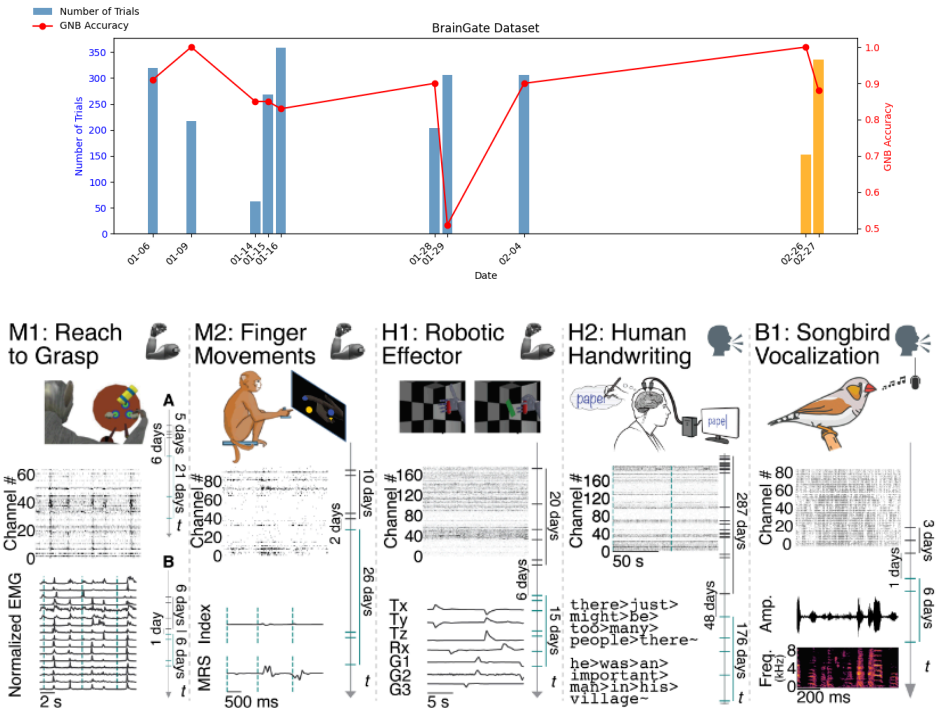


Figure 2. Datasets used in the project. We used H2 in FALCON

Base Models

The base models are standard neural network-style models that leverage normalization. This required using BatchNorm1d layers since our data is time series neural signals. Since we're still in a model-agnostic setting, we can use any model trained using gradient descent. We chose to test an MLP, a convolutional RNN (CRNN/LSTM), a transformer, and Gaussian naive bayes (GNB). Examples of the model structures for CRNN and transformer encoder are shown in Figure 3.

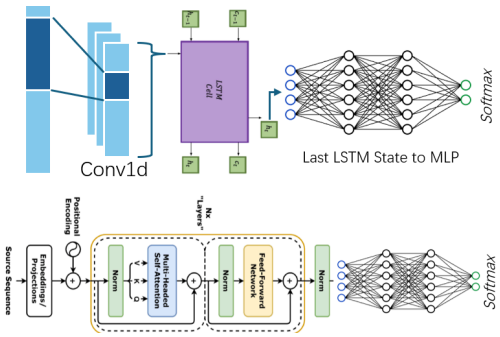
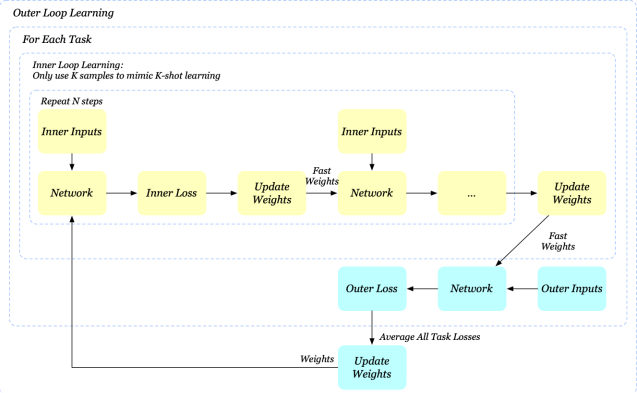


Figure 3. Structures of some of the base models

NeuraNIL

Just like Raghu et al. (2020)'s ANIL, our NeuraNIL model can be broken down into two key parts: the outer loop and the inner loop. The outer loop updates the meta-initialization of the neural network parameters to enable fast adaptation. It takes in the raw neural signal features and maps them to a lower-dimensional latent space to avoid non-stationarity. In our case, we will use an LSTM or a transformer due to the data's sequential nature. The inner loop performs task-specific adaptation over a few labeled examples. It takes the data in the low-dimensional latent space and makes a prediction. It's almost always an MLP, but another gradient descent model could be used instead. Figure 4 shows this pipeline in a diagram and how it's implemented in pseudocode.



Algorithm 1 NeuraNIL training

Input: Training set divided to n episodes $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. Each (x_i, y_i) is an episode that can be divided into support set (x_{Si}, y_{Si}) and query set (x_{Qi}, y_{Qi})

Initialize learner parameter θ_0 and classifier w_0
while not done **do**
  Sample *batch_size* episodes x_i, y_i from $\mathcal{D}$ as B_j
  $\mathcal{L}_j = 0$
  for each *episode* **do**
    $x_{Si}, y_{Si}, x_{Qi}, y_{Qi} \leftarrow x_i, y_i$
    Learner forward pass $x'_{Si} = \phi_{\theta_j}(x_{Si})$
    Classifier coefficient $w'_i = w_i - \beta \nabla_{w_i} L_T(w_i'^T \phi_{\theta_i}(x_{Si}), y_{Si})'$
    $\hat{y}_{Qi} = w_i'^T(\phi_{\theta_j}(x_{Qi}))$
    $\mathcal{L}_j += CE(y_{Qi}, \hat{y}_{Qi})$
  end for
  $\theta_{i+1} = \theta_i - \alpha \nabla_{\theta_i} \sum_{\tau_i \sim p(\tau)} L_T(w_i'^T \phi_{\theta_i}(x), y)$
  $w_{i+1} = w_{i+1} - \nabla_{w_i} \sum_{\tau_i \sim p(\tau)} L_T(w_i'^T \phi_{\theta_i}(x), y)$
end while

Figure 4. Diagram of meta-learning pipeline and proposed pseudocode for the approach

Results

Base Models

Of all the base models tested on the BrainGate dataset, GNB performed the best on most days. This is shown below in Figure 5. LSTM and MLP also performed well, with the transformer being by far the worst model. This is interesting—it seems like the transformer architecture may be overkill for this small amount of data. All four of the models perform significantly better than random chance (16.67%), with the best performance being the GNB on day 9 at 99% and the worst performance being the transformer on day 6 at 32%. At worst, we're still 2 times better than random chance.

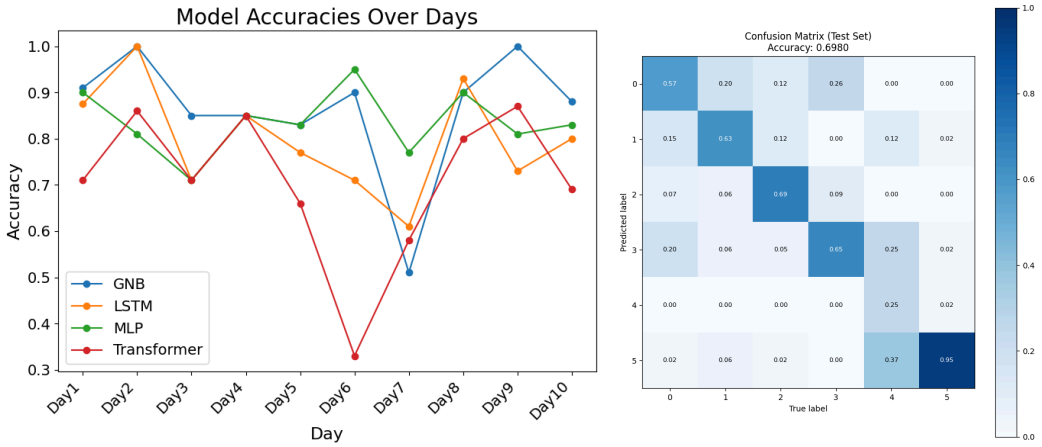


Figure 5. Base model performance on BrainGate data

On the FALCON data, the base models all significantly overfit. This is shown in figure 6 for the MLP. Training accuracy and loss are both significantly better than validation accuracy and loss. Considering this is a 26-class classification task, a validation accuracy of 40% isn't horrible—it is over 10 times better than random chance (3.8%).

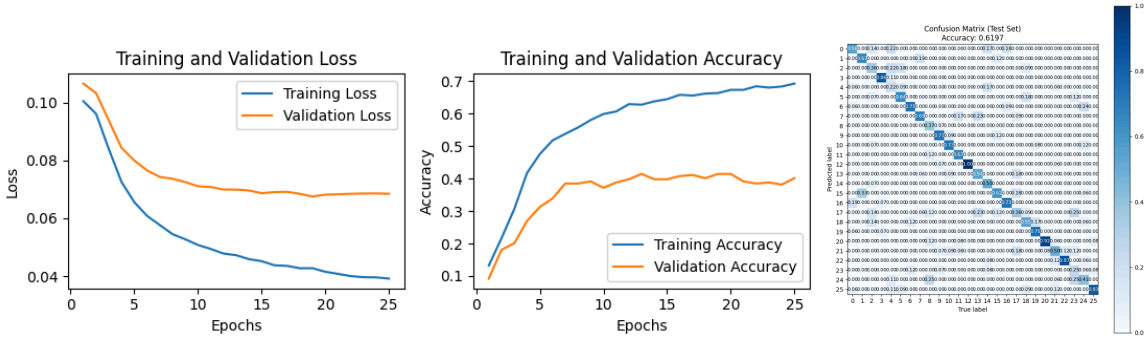


Figure 6. Base model performance on FALCON data

NeuraNIL

After implementing our NeuraNIL model, we ran several experiments on both datasets to determine the model's quality. Some of these are shown in Figure 7. On the BrainGate data, NeuraNIL performs better than the MLP base model on 7 out of 9 days. Its accuracy ranges from around 50% to 89%. We consider this implementation a success, but there's a lot of room for improvement. For the FALCON data, the highest accuracy is around 35%, which is not great. The base models seemed to perform slightly better, even despite their overfitting.

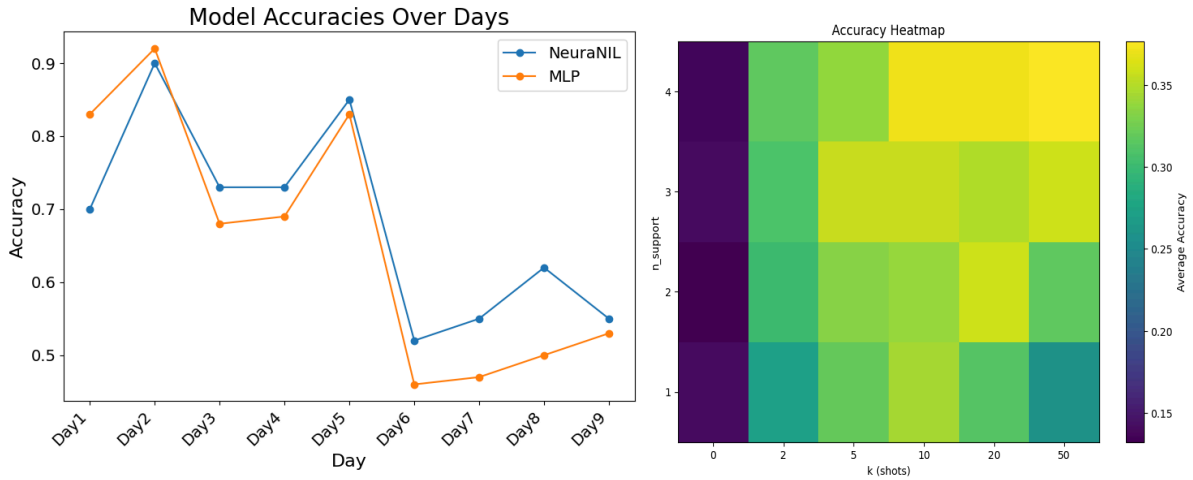


Figure 7. Next-day prediction of NeuraNIL on BrainGate data and KN-sweep of NeuraNIL on FALCON data

Challenges

The main challenges for this project lie in performing the meta-learning method on real-world, poorly labelled data. The BrainGate data in this project was obtained from an advanced ALS patient who has been in a locked-in stage. The patient was trying to use the BrainGate system to communicate by imagining different gestures and would optimize his strategy using this closed-loop communicator. In addition, the non-stationary properties of the neural signals were also a challenge. From day to day, or even within the same day, neural signals can significantly vary, which can lead to the failure of a decoder. Accuracy on this task is essential, as it's the only way for these patients to communicate.

Regarding the use of the H2 dataset from the FALCON paper, we experienced difficulties in making the format adaptable to our deep learning pipeline. The dataset was stored in `.mat` files, and the labeling was not very clear. We needed to adapt it for NumPy/Python workflows. The data was nested in deeply layered MATLAB-specific objects, and the keys were hard to identify. Sometimes the structures were not consistent across sessions or trials. We needed to get the original paper from which the dataset came to fully understand the data. The data's formatting was also not uniform. In addition to that, the reshaping of tensors and normalization processes were difficult. There were many type mismatches, which made the preprocessing process take longer. It also made it harder to run the models using the dataset.

Another challenge was that the data was collected with different trial lengths—1 and 2 seconds. Creating batches and performing LSTM was tricky. We handled this using the `pad` and `pack` functions in PyTorch.

Reflection

Overall, this project was both a challenging and research-intensive process. We faced a lot of obstacles along the way, with the most significant being the lack of data to train our model to a satisfactory level of accuracy. This exposed us to the real-world challenge of finding enough high-quality data to train deep learning models, especially for tasks that have significant real-world implications. While our previous coursework introduced us to model-building and data handling, we hadn't worked with limited data until this project, which made it a great learning experience.

In terms of our original base and target goals, we approached the base goal for the BrainGate data with NeuraNIL performing similarly well to GNB. However, for the FALCON dataset, our model performs significantly worse than the paper's accuracy (~80%). We are disappointed with these results, but we believe that, with more time, we could have taken several steps to enhance our model's performance. By acquiring additional data, tuning our model's hyperparameters, and conducting more experiments with different architectures, we could have made substantial improvements to the accuracy and robustness of our results.

Most of the implementations went according to plan, with the biggest struggle being the different sequence lengths within the data and handling that across models. Overall, we are happy with the results and takeaways from this project, but we wish we had more time to improve our model even further.

References

[1] Raghu et al. (2020), *Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML*. https://arxiv.org/abs/1909.09157.
[2] https://www.rnel.pitt.edu/research/neuroprosthetics/intracortical-brain-computer-interfaces
[3] Degenhart, A. D. et al. *Stabilization of a brain–computer interface via the alignment of low-dimensional spaces of neural activity*. Nat Biomed Eng 4, 672–685 (2020).
[4] Finn et al. (2017), *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*. https://arxiv.org/abs/1703.03400.
[5] Karpowicz et al. (2024), *Few-shot Algorithms for Consistent Neural Decoding (FALCON) Benchmark*. ttps://www.biorxiv.org/content/10.1101/2024.09.15.613126v1.
[6] https://pub.towardsai.net/how-to-train-maml-model-agnostic-meta-learning-90aa093f8e46.