NeuraNIL

Model Agnostic Meta-Learning for Neural Decoding

"Outer loopers": Yishu Li, Jania Vandevoorde, Carlos Ramos

INTRODUCTION

Intracortical brain-computer interfaces (iBCIs) enable people with advanced ALS to communicate by decoding movement intentions from neural signal. However, current iBCI system require frequent decoder re-calibration due to the **non-stationary of neural signal** to maintain the performance.

Meta Learning is a branch of machine learning where models are trained to learn new tasks on their own. Therefore, in our data, we can treat each day as a new task and use meta-learning approach to let the model fast adapt to the new day's non-stationarity.

RELATED WORKS

Meta-learning:

Finn et al. (2017) and **Raghu et al. (2020)**

Model Agnostic Meta Learning (MAML) and Almost No Inner Loop (ANIL): train a model to solve different tasks than those it trained on. Divides the model into two parts:

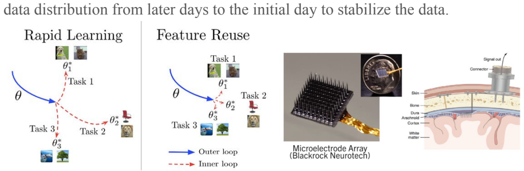
→ ***Outer loop:*** updates the meta-initialization of the NN parameters to enable fast adaptation to new tasks

→ ***Inner loop:*** perform task-specific adaptation over a few labeled samples.

Deep learning and iBCI decoders:

Hosman et al. (2018) showed that non-linear recurrent neural networks (RNNs) can provide higher performance iBCI control compared with linear methods.

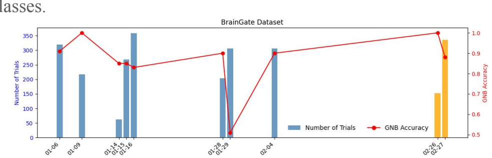
Degenhart et al. (2020) developed a manifold-based iBCI decoder that maps the data distribution from later days to the initial day to stabilize the data.



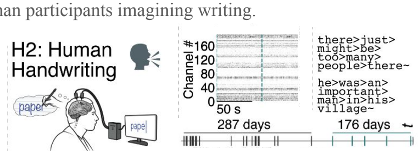
Previous meta-learning methods used by MAML and ANIL respectively (left) and Utah Array used for iBCI made by Blackrock Neurotech (right).

DATASETS

1) BrainGate2 clinical trial data: neural signals of advanced ALS patients in locked in stage trying to communicate by imagining a set of hand gestures. The dataset includes 11 days and 6 poorly labelled classes.



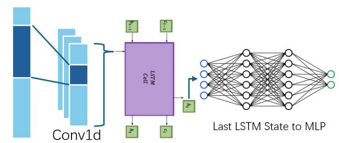
2) FALCON H2 data: from Karpowicz et al. (2024). Neural signals of human participants imagining writing.

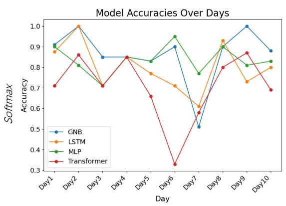


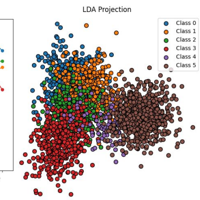
BASE MODEL & RESULTS

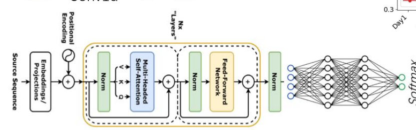
Normalization: Although deep neural networks are thought to be an architecture that can approach universal functions, we found normalization to be important to our model's performance. Since we're using temporal data, we normalized using `BatchNorm1d`.

Base Models: Both MAML and ANIL are model agnostic, therefore, we can select any models to be the outer and inner loops as long as they are trained using gradient descent. In this project, we implemented MLP, CRNN, Transformer Encoder.









Left: Model structure of CRNN (top left) and transformer encoder (bottom left).
Middle: Base models' performance across days. Models seem to overfit.
Right: LDA projection for all classes.

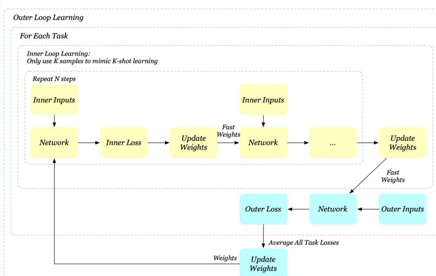
META-LEARNING MODEL & RESULTS

Model Structure & Training

The data is divided into episodes, each with a support set and query set.

→ **Outer loop ("learner")**: takes in the raw neural features and maps it to a lower dimensional latent space to avoid non-stationary. Will be an LSTM or transformer due to the data's sequential nature.

→ **Inner loop ("classifier")**: takes the latent space neural data and makes a prediction. Typically an MLP but any other model trained on gradient descent will work.



Algorithm 1 NeuraNIL training

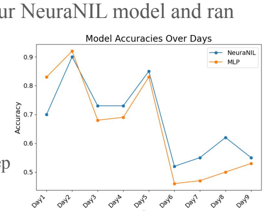
Input: Training set divided to n episodes $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. Each (x_i, y_i) is an episode that can be divided into support set (x_{Si}, y_{Si}) and query set (x_{Qi}, y_{Qi})

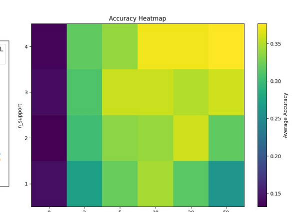
Initialize learner parameter θ_0 and classifier w_0
while not done **do**
  Sample *batch_size* episodes x_i, y_i from $\mathcal{D}$ as B_j
  $\mathcal{L}_j = 0$
  for each *episode* **do**
    $x_{Si}, y_{Si}, x_{Qi}, y_{Qi} \leftarrow x_i, y_i$
    Learner forward pass $x'_{Si} = \phi_{\theta_j}(x_{Si})$
    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

Results:

We successfully implemented our NeuraNIL model and ran several experiments:

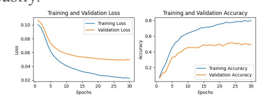
Figure left 1: Next-day prediction accuracy of MLP and NeuraNIL on the BG dataset.
Figure left 2: The n_support-k sweep of NeuraNIL on the FALCON dataset.





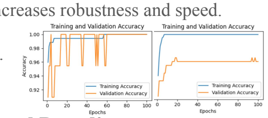
CONCLUSION

1) Base Model: We developed several models based on both machine learning and deep learning method. We found that fancier deep learning model not necessarily performs better, they tend to overfit more easily.



2) `Conv1d` for RNN: Adding a `Conv1d` layer to a RNN is like downsampling, which increases robustness and speed.

Left figure 1: CRNN training.
Left figure 2: RNN training.



3) Meta-learning for Neural Decoding: During the next-day predict tasks, and NeuraNIL showed better results than the base model MLP, which indicates that meta-learning has the potential to fast-adapt to the neural non-stationarity.

4) Generalization for New Tasks: We tried to fast adapt the model to the tasks it has never seen before, the result accuracy is higher than random chance, and increases as k value increases.

CHALLENGES & LIMITATIONS

1) Model K selection: The number of inner loop iterations, K, plays a crucial role in meta-learning, as it determines how close the learner gets to the minimum for each specific task. The larger your K value is, the slower the model adapts, while the smaller your K value is, the harder it is for the model to generalize.

2) Dataset: In the BrainGate dataset, the tasks are simple and there are only 6 different classes. Therefore, it remains unknown whether the learner can learn a general distribution from the dataset. This may lead to the model overfitting.

3) Overfitting: In our experiments, the model overfits badly at first, but by adding the normalization, the model trained on the same got better, but still overfits across days. The problem may come from preprocessing rather than the model structure.

REPOSITORY

github.com/Yishu-Li/NeuraNIL

ACKNOWLEDGEMENTS

We would like to thank the entire 1470 course staff, especially Professor Eric Ewing and our project mentor Yujin Chung.