CSIC30100: Brain Computer Interface **(Due: 12/15/2023)**

HW3: Deep Learning for BCI

Instructor: Chun-Shu Wei
TA: Student Id

Introduction

In this homework, you are asked to utilize Convolutional Neural Network (CNN) for motor imagery EEG classification tasks, with experiments on 2 training schemes × 3 curriculums and 2 model architectures. Please follow the provided code template for CNN framework implementation, model training, and dataset loading to fulfill the requirements. Try to make observations through the progress and discuss the results.

Kaggle link
Code template

**Make sure that you download the sample code before starting your implementation.[†]

Submission Policy

Read all the instructions below carefully before you start working on the assignment, and before you make a submission.

- **PLAGIARISM IS STRICTLY PROHIBITED. (0 point for Plagiarism)**
- The code will be graded on code completeness, model structure and algorithmic correctness.
- Your report will not be checked with any AI content detector, but we would like to remind you that contents on EEG or EEG ML/DL generated with GPTs tend to be extremely erroneous.
- Submission deadline: **2023.12.15 10:00:00 AM**.
- Late submission penalty formula:

$$\text{original score} \times (0.7)^{\#(\text{days late})}$$

Submission Format

- Each student submits 1 zip file including **1 report** (.pdf file) **in English** and your **code** (.ipynb file). Paper submission or programming languages other than Python/deep learning framework other than PyTorch is not allowed.
- Remember to upload the results for unlabeled test data to Kaggle.
- The source code must contain **comments**.
- The report must contain **observations, results, and explanations**. Please name your zip file as hw3_studentID_Name.zip.
- Illegal format penalty: **−5 points** for violating each rule of submission format.

Materials

- PyTorch tutorials:
  Official: link
  Basics: link
- As "Basic Python programming" is a prerequisite of this course, you'll have to deal with basic programming problems by yourself.
- Other material for your own reference: EEGNet official implementation

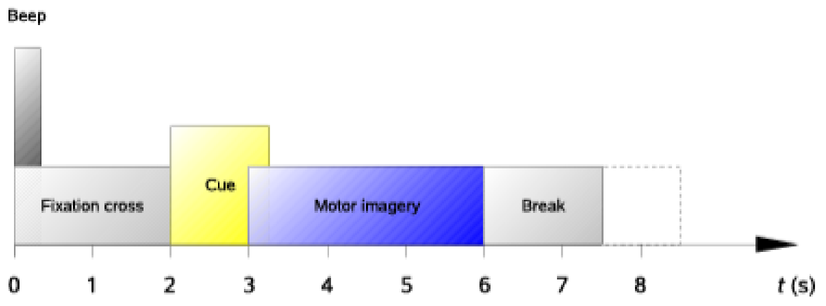
---

1. Problem

1.1. EEG Dataset

Dataset description: (Official document)

BCI competition IV 2a is a well-studied motor imagery dataset composed of **22** channel EEG data from **9** subjects. Four motor imagery tasks are involved, namely the imagination of movement of the **left hand** (class 1), **right hand** (class 2), **both feet** (class 3), and **tongue** (class 4). The experiment paradigm is shown as follows.



In a trial, the subject is presented with a visual fixation cross. The instruction cue to imagine specific body part movement is given at 2s and lasts for 1.25s, the subject is prompted to perform the desired motor imagery task until 6s. Each subject has one training and one evaluation session, each session includes **72** trials for the 4 classes, yielding a total of 72*4 trials per session. The given dataset is filtered by FIR with band [4, 38] Hz and resampled to 125.0 Hz. The event trials are epoched using the same procedure described in referenced literature[2] .

Dataset is provided through Kaggle. You should follow the instructions in the code template to download this dataset. The dataset will be organized according to the following structure.

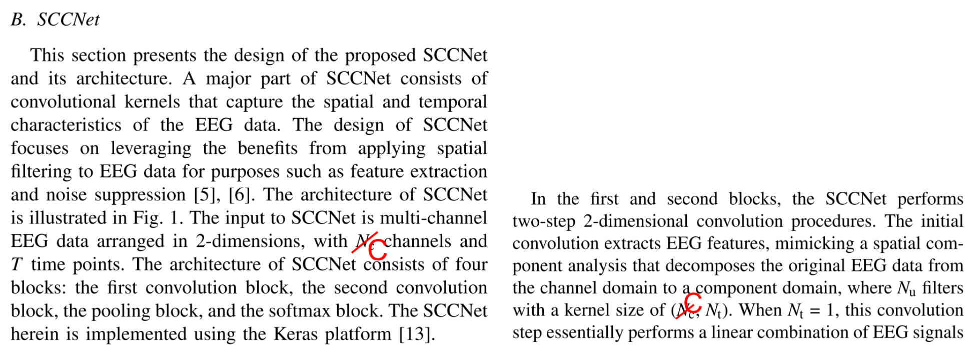
Kaggle data structure
- BCI_hw3_dataset/   - train/   - BCIC_S0x_T.mat (x from 1~9, keys=x_train, y_train)   - labeled_test/   - BCIC_S0x_E.mat (x from 1~4, keys=x_test, y_test)   - unlabeled_test/   - BCIC_S0x_E.mat (x: 5, 6, A, B, C, keys=x_test)

The dataset filename is under the format "BCIC_S{subject_ID}_{T(train)/E(test)}.mat", e.g., "BCIC_S01_T.mat" is the training session for subject one and "BCIC_S03_E.mat" is the test session for subject three; as the "training" and "evaluation" sessions was in context of the dataset publication, in this document we refer to the training session as "first session" and the evaluation session as the "second session". For the unlabeled test data, three of the subject IDs are hidden and the trials were shuffled.

1.2. CNNs

Two baseline EEG models: **EEGNet**[1] and Spatial Component-wise Convolutional Network (**SCCNet**)[2] are included in this homework. The EEGNet implementations are provided in the sample code, for SCCNet you will have to finish the implementation on your own. The model design ideas and more details could be found in the BCI_ML2 course material and referenced literature[1] [2].

Important: In the SCCNet paper description, notation *Nc* was accidentally used for both "Number of EEG channels" and "Number of kernels in the second convolution block". In the sample code "Number of EEG channels" is notated as *C*. You can read the description as:



B. SCCNet

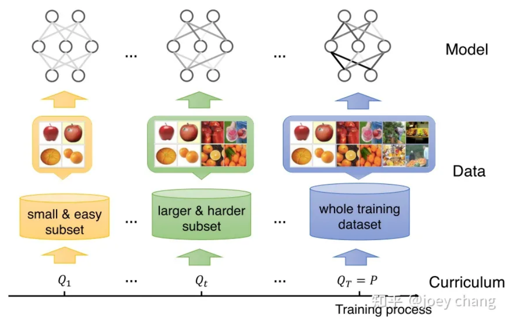
This section presents the design of the proposed SCCNet and its architecture. A major part of SCCNet consists of convolutional kernels that capture the spatial and temporal characteristics of the EEG data. The design of SCCNet focuses on leveraging the benefits from applying spatial filtering to EEG data for purposes such as feature extraction and noise suppression [5], [6]. The architecture of SCCNet is illustrated in Fig. 1. The input to SCCNet is multi-channel EEG data arranged in 2-dimensions, with ~~N~~ C channels and T time points. The architecture of SCCNet consists of four blocks: the first convolution block, the second convolution block, the pooling block, and the softmax block. The SCCNet herein is implemented using the Keras platform [13].

In the first and second blocks, the SCCNet performs two-step 2-dimensional convolution procedures. The initial convolution extracts EEG features, mimicking a spatial component analysis that decomposes the original EEG data from the channel domain to a component domain, where N_u filters with a kernel size of (~~N_c~~ C, N_t). When N_t = 1, this convolution step essentially performs a linear combination of EEG signals

1.3. Training Schemes

The visualized explanation can be found in the BCI_ML2 course material and the text explanation can be found in the referenced literature[2]. The training schemes refer to different training data / test data combinations to simulate different kinds of real life BCI application scenarios.

Difficulty based training schemes: The concept of **curriculum learning** involves initially training on an easier task and subsequently progressing to a more challenging task. For instance, in computer vision, the training process begins with an easy dataset subset, followed by fine-tuning on a more difficult subset. You are required to implement a pipeline similar to curriculum learning following the instructions provided below.



Train and test the model on subject 1-4 using following training schemes:

Subject Independent (**SI**) with difficulty based curriculums
Please refer to Table 1 In literature[2] as each subject's "difficulty" ranking. Based on the targeted test subject, the data from the other 8 subjects would be your training set. Within the training set, identify 4 "easy" subjects. The 4 easy subjects would be the "easy" data, and all 8 subjects would be the "all" data. Each subject with label (id 1-4) should be taken as the targeted test subject once.

1) Curriculum 1: Easy to all
   Phase 1: Train with the first session of the 4 (easy) subjects. Phase 2: Train with the first session of the 8 (all, excluding test subject) subjects. Evaluate on the second session of the test subject.

2) Curriculum 2: All
Train with the first session of the 8 (all, excluding test subject) subjects. Evaluate on the second session of the test subject.
3) Curriculum 3: All to easy
Phase 1: Train with the first session of the 8 (all, excluding test subject) subjects. Phase 2: Train with the first session of the 4 (easy) subjects. Evaluate on the second session of the test subject.

Subject Independent + Fine Tuning (**SI+FT**): After each curriculum under SI training scheme, fine tune on the first session of the targeted test subject and evaluate on the second session of the targeted test subject.

1.4. **Requirements** (60%)

1.4.1. (45%) Please go through the referenced literature[2] carefully and complete the "#TODO" parts. For SCCNet implementation, you have to use the predefined arguments to construct each layer. If your SCCNet structure is constructed with fixed constants instead of the provided arguments, the penalty is HW3 score -5. For training scheme implementation, make sure your dataset contents meet the scheme definitions.
Plot the confusion matrix of your best performed model-scheme-curriculum combination.
※ confusion matrix wikipedia
※ matplotlib.cm (optional)

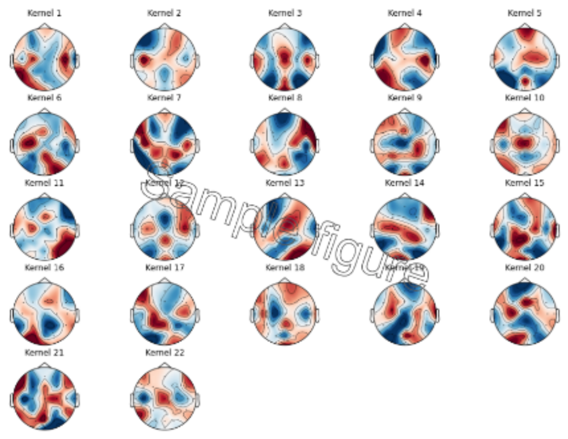
	EEGNet	SCCNet (or SCCNet_v2)
SI	3 curriculums x 4 subjects = 12 performances	3 curriculums x 4 subjects = 12 performances
SI+FT	3 curriculums x 4 subjects = 12 performances	3 curriculums x 4 subjects = 12 performances

1.4.2. (5%) Fill in your hyper-parameter settings (Batch size, learning rate, epochs, optimizer, etc) of your best performed model-scheme-curriculum combination.
※ if you didn't modify the sample code, fill in the hyper-parameters specified in the sample code.

1.4.3. (10%) Obtain spatial kernel weights from the first convolutional layer of your best performed SCCNet model, visualize the weights as topographic maps using the MNE package.
※ PyTorch Conv2D attributes
※ mne.channels.make_standard_montage (Hint: electrode position)
or mne.info (Hint: EEG recording metadata structure)

※ mne.viz.plot_topomap

(Hint: There should be 22 topo maps for one model in your figure, each corresponding to a channel. Model weights and their meanings article1, article2)



1.4.4. (0%) Make predictions to the 5 test subjects without true labels (unlabeled_test) with the one with best performance from your trained model, export the result as a .csv file using the sample code (section `Generate Submission csv File`) and **submit it to kaggle**. This part is to make sure your code is executable and your models are trainable, which should beat the 0.6 accuracy baseline (the baseline will be adjusted dynamically according to your submissions). **Otherwise you will only get half of the score in your implementation (1.4.1)**.
(Hint1: You may need to tune your hyper-parameter setting to get a higher accuracy)

1.5. **Discussion** (40%)

1.5.1. Strengths and weaknesses of the 2 CNN models and your observations regarding the models (structures, parameter size .etc)

1.5.2. Strengths and weaknesses of the training schemes and curriculums

1.5.3. (Optional) For models trained with different subject sets, what are the possible reasons for the difference in model performance.

1.5.4. Other topics you find worthy to discuss.

1.6. **Bonus Challenge**

1.6.1. **SCCNet_v2**

In the SCCNet architecture there is a “permutation layer”, whose functionality can be achieved simply through some modification of the kernels in the convolutional layer(s). To get bonus points, try to implement SCCNet without the permutation layer. (Hint: go through the referenced literature[2] and try to understand the kernel direction, their corresponding data dimensions, and the kernels’ operation objectives.)
Correct implementation: HW3 +5 points

1.6.2. Kaggle leaderboard

The top **3** submissions in the Kaggle leaderboard will get bonus points.
Top 1: HW3 +30 points
Top 2: HW3 +20 points
Top 3: HW3 +10 points

1.7. References

[1] Lawhern, Vernon J., et al. "EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces." Journal of neural engineering 15.5 (2018): 056013.
[2] Wei, Chun-Shu, Toshiaki Koike-Akino, and Ye Wang. "Spatial component-wise convolutional network(SCCNet) for motor-imagery EEG classification." 2019 9th International IEEE/EMBS Conference on Neural Engineering (NER). IEEE, 2019.

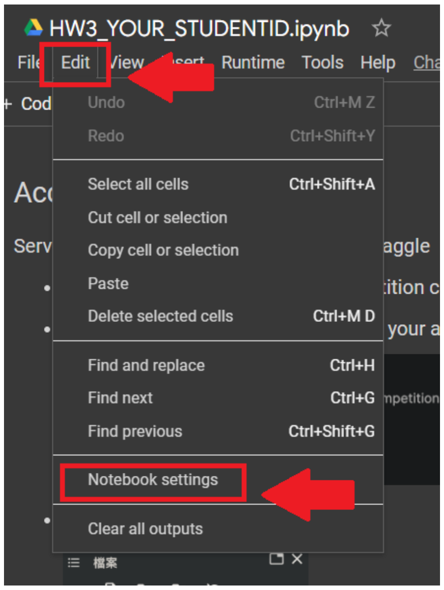
Pytorch Basics

Local machine installation: installation link
Select your local machine configuration and copy the command
(If you decide to use google colab, you don't have to install locally)

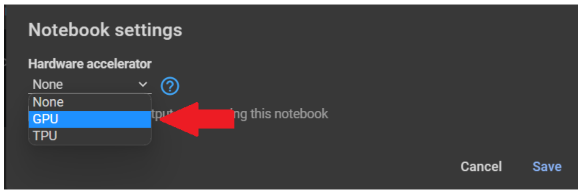
- Numerical data structure
    - tensor
    - (numpy array ~= pytorch tensor)
    - (np.function() ~= torch.function())
- Neural Network Official documentation
    - built by modules
- Dataset /Dataloader Official documentation
    - datasamples and the corresponding labels / iterator of Dataset
- Pytorch sample code

Google Colab GPU setting

1. Find "notebook setting" in "Edit" from top menubar



2. Select "GPU" and restart runtime session to apply the change


Notebook settings
Hardware accelerator
None
None
GPU
TPU
Cancel
Save