Other Notes

Huu Duc Nguyen M.Sc.

02 September 2022

Contents

Abbreviations

a.k.a.	also known as
FFT	Fast Fourier Transform
CNS	central nervous system
PNS	peripheral nervous system
EEG	Electroencephalography
fMRI	Functional Magnetic Resonance Imaging
ECoG	Electrocorticography
LFP	Local Field Potentials
BCI	Brain-Computer Interface
BMI	Brain Machine Interface
NMP	neuromotor prostheses
PSD	Power Spectral Density

1 Neuroscience

1.1 Nervous System

The nervous system has two main parts:

- The central nervous system (CNS) is made up of the brain and spinal cord.
- The peripheral nervous system (PNS) is made up of nerves that branch off from the spinal cord and extend to all parts of the body.
  - Allows the CNS to send and receive signal to other parts or our bodies
  - Regulate involuntary functions, e.g., heartbeat, breathing, releasing hormones like adrenaline, opening the pupil in response to light, and regulating the digestive system.

1.1.1 Neuron

The fundamental unit of the nervous system is nerve cell, also known as (a.k.a.) neuron.

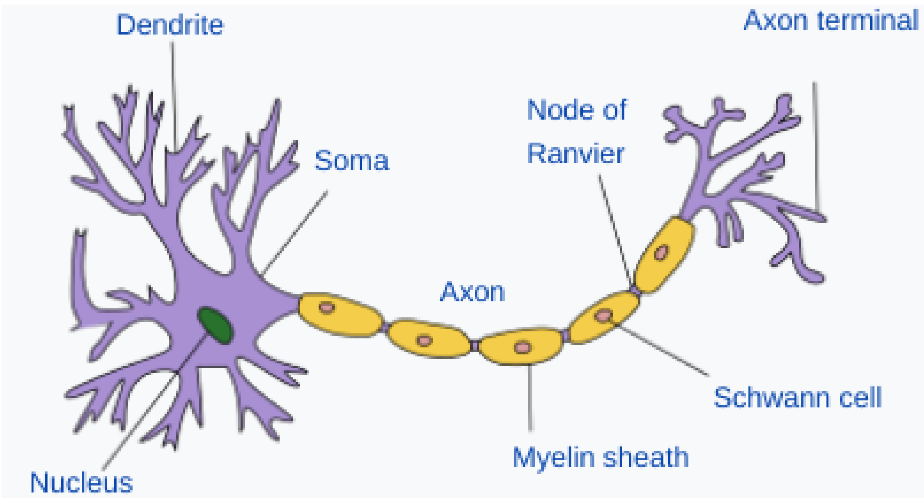

Figure 1.1: Neuron structure

- A *neuron* has a cell body, which includes the cell nucleus, and extensions called *axons* (/AK-sonz/) and *dendrites* (/DEN-drahytz/).
- *Synapse* (/SIN-aps/) is the space between the end of an axon and the tip of a dendrite from another neuron.
- Axons and dendrites allow neurons to communicate, even across long distances.
- Bundles of axons, called nerves, are found throughout the body.

- Neuron send a message to another neuron by
  - Send an electrical signal down the length of its axon
  - At the end of the axon, the electrical signal changes to a chemical signal, which is released with *neurotransmitters* (/noor-oh-TRANS-mit-erz/) into the synapse
  - The neighboring dendrite converts the chemical signal back into an electrical signal.
- *Glia* (/GLEE-uh/), non-neuron cells, perform many important functions that keep the nervous system working properly
  - Support and hold neurons in place
  - Protect neurons
  - Create myelin insulation called, which helps move nerve impulses
  - Repair neurons and help restore neuron function
  - Trim out dead neurons
  - Regulate neurotransmitters

1.1.2 References

- Wikipedia - Nervous System
- NIH - What are the parts of the nervous system?

1.2 The Brain

- *Brain anatomy* is also called brain structure.
  Animals' brains have common structure:
  - Forebrain
  - Midbrain
  - Hindbrain
  - Spinal cord
- *Brain physiology* is also called as brain function.
  The closer we are to the spinal cord, the more basic the functions are, e.g., breathing, digest food.

Human's brain structure:

- *Brainstem*
  - Includes: *midbrain, pons, medulla oblangata*
  - Functions:
    - Basic functions: breathing, circulation, digestion
    - Sort in/out sensory information
- *Cerebellum*: body control, motion memory

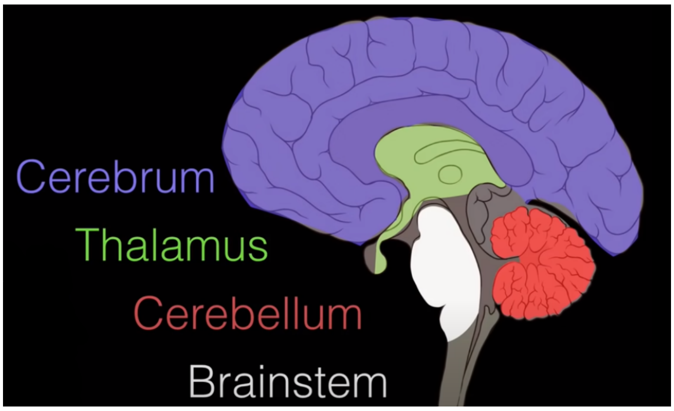

Figure 1.2: Human's brain structure.

- *Thalamus*: is like a router, sort data to the cerebrum
  - Includes: *hypothalamus*, *posterior pituitary*
  - Homeostasis for maintaining body temperature, circadian rhythm, etc.
- *Cerebrum*: 80% of the brain
  - Integrates different data
  - Has two hemisphere, connected via *corpus collosum*
  - Divides into four lobes:
    - Frontal lobe: personality, emotion, high-level thinking, controlling movements
    - Parietal lobe (/puh.**rai**.uhtl/): sensation, deal and react with environment
    - Occipital lobe (/uhk.**si**.puhtl/): vision, recognition of shapes, colors
    - Temporal lobe: language, hearing, reading, memory, other senses
  - The parts on two side of the edge between frontal and parietal lobes are called:
    - Somatosensory cortex: takes in sensation, makes sense of it
    - Motor cortex: sends information out

1.2.1 References

- Wikipedia - Brain
- medXclusive Learning - Human Brain: Major Structures and their Functions
- Bozeman Science - The Brain

1.3 Brainwaves

- Clusters of neurons, which getting message/feedback from each other, start synchronize their firing patterns with others.
- This results in *neural oscillations*, or *brainwaves*, which are rhythmic or repetitive patterns of neural activity in the CNS

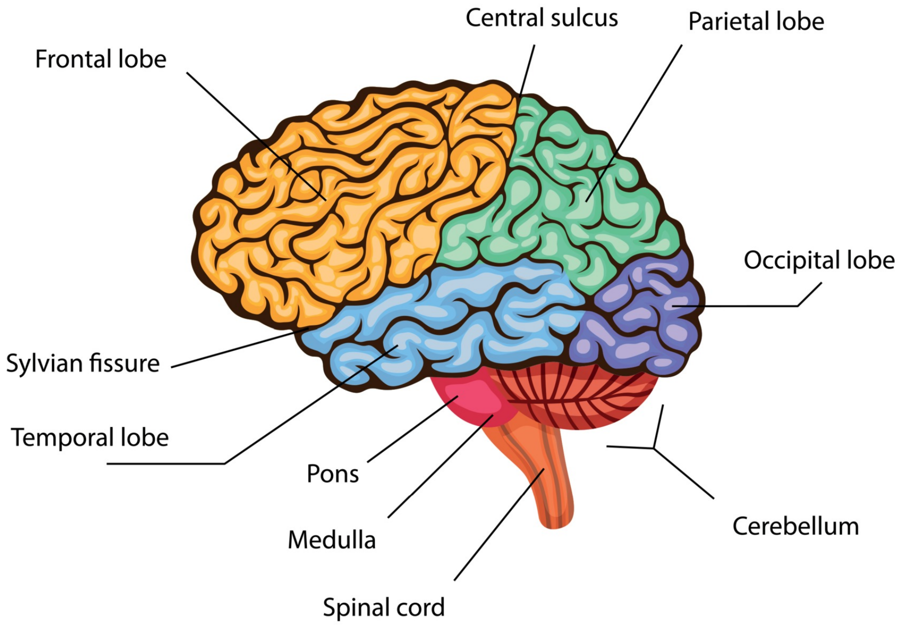

Figure 1.3: Human's brain structure.

1.3.1 Categorization

A non-inclusive list of types of oscillatory activity found in the central nervous system:

- Delta waves (0.1-4Hz): technically links with deep sleep
- Theta waves (4Hz-7.5Hz): daydreaming, or meditation
- Mu waves (8Hz-12Hz): ??
- Alpha waves (7.5Hz-15Hz): awake, but relax
- Beta waves (12Hz-30Hz): awake, thinking about something
- Gamma waves (30Hz-100Hz): deeply focus
- At one, all waves are somewhat active, it's just that some waves are more dominant.

NOTE:

- There are more than 5 types, check Wikipedia - Neural oscillation
- Each with different frequency and probably different functions in our brain.
- Generally, the higher the frequency, the more alert you are

1.3.2 Comparison

[**TODO: How to compare brain waves?**]

- Fast Fourier Transform (FFT)
- Cross-correlation

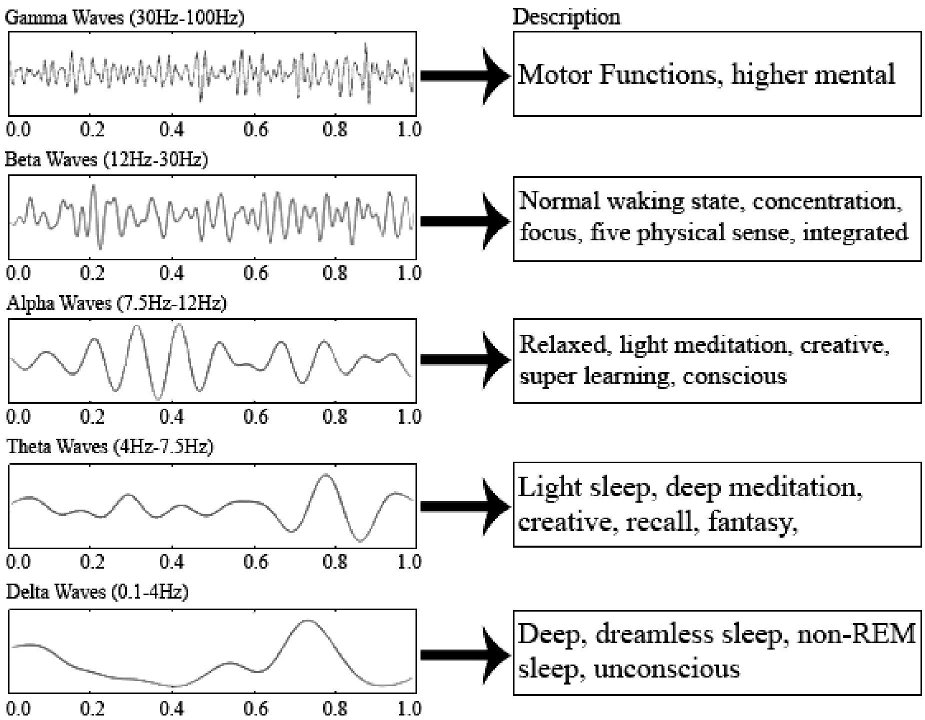

Figure 1.4: Some of the common brainwaves.

1.3.3 Brain Imaging Techniques

Brain imaging techniques are methods to measure these brainwaves. Some common methods with their pros and cons are (Fig. 1.5):

- Electroencephalography (EEG):

  +Inexpensive, portable, high temporal resolution

  −Low spatial resolution
- Functional Magnetic Resonance Imaging (fMRI)

  +Higher spatial resolution

  −Low temporal resolution, high cost
- Electrocorticography (ECoG)

  +Higher spatial and temporal resolution

  −Requires surgery, signals can degrade
- Local Field Potentials (LFP) and Spikes

  +Very high spatial and temporal resolution

  −Requires surgery, signals can degrade, high power consumption and latency

1.3.4 References

- Microsoft Research | Developing a BCI based on Visual Imagery

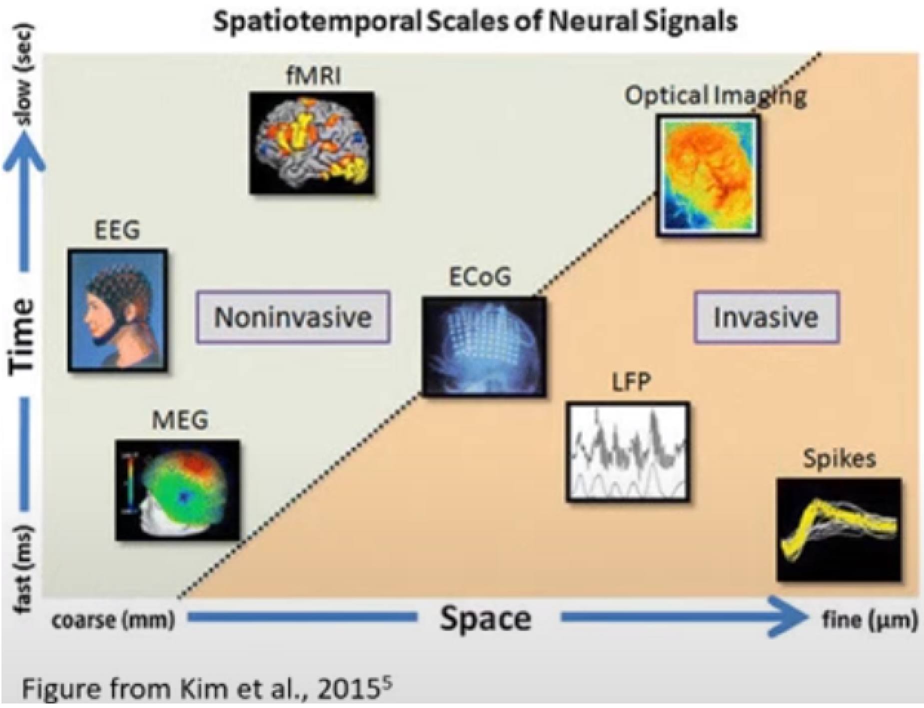

Figure 1.5: Different brain imaging techniques techniques vary in spatial and temporal resolution.

1.4 Brain-Computer Interface

Brain-Computer Interface (BCI), a.k.a. Brain Machine Interface (BMI), neuromotor prostheses (NMP), are devices that transform neural signals into command signal to operate other physical systems.

1.4.1 Paradigms

There are two BCI paradigms:

- Evoked potentials: providing participants with external stimulus, then measure their responses.
  - Visual P300
  - Steady state visual evoked potential (SSVEP)
  - Auditory stimuli: less studied
  - Somatosensory (tactile) stimuli: less studied
- Non-evoked potentials: participants engage in imagery and measure their brain activities.
  - Motor imagery
    * Sensorimotor rhythms (SMR): imagined movements of large body parts
    * Imagined body kinematics (IBK): imagined movements of a single body part in a multi-dimensional space

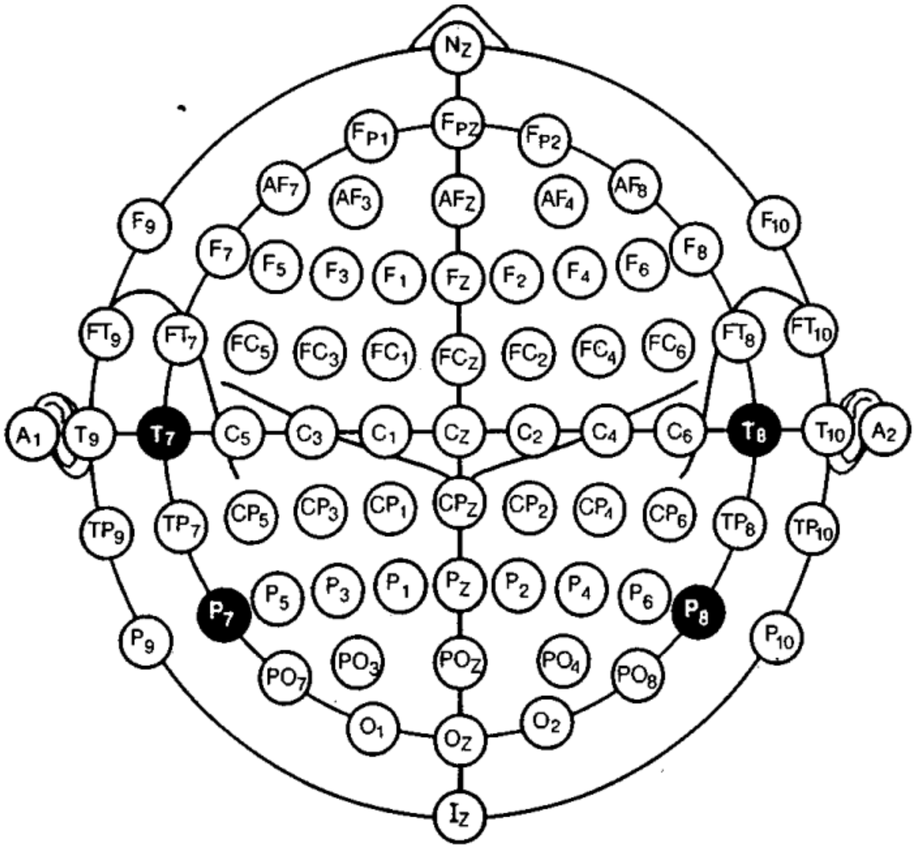

Figure 1.6: The standard 64 scalp electrodes of BCI system (view from above, subject's nose is at the top). Abbreviations: frontotemporal (FT), frontocentral (FC), temporal-posterior temporal (TP), centroparietal (CP), posterior temporo-occipital (PO), parieto-occipital (PO), anterior frontal (AF). [Sha91]

- Less studied non-evoked potentials
  * Visual imagery: conceptual reconstruction of a perceptual experience
  * Speed imagery: imagery of speed production
  * Mental effort: mental object rotation, math calculations, etc.

1.4.2 EEG Processing

There are 3 types of feature extraction methods for EEG processing:

- Spectral
  - Frequency - domain analysis: Estimation of spectral power (stationary signals)
    * Power Spectral Density (PSD) (Welch's Periodogram)
    * FFT
  - Time - frequency domain analysis
    Mother wavelets and wavelet transform

- Temporal
- Spatial

Microsoft Research:

1. Collect raw EEG Data
2. Channel selection (different electrodes in different brain areas)
3. Apply filter, remove noise (power line noise)
4. Artifact removal (eye, muscle movements)
5. Epoch data
6. Feature extraction

1.4.3 Motor Imagery

- Wolpaw et al. (2000) [WMV00] used μ and β waves from midpoint of the central sulcus (Fig. 1.3) bilaterally to control computer cursor. It is probably using signal from C3 and C4 electrodes (Fig. 1.6)
- Pfurtscheller et al. (2001) [PN01]
- Hoshino et al. (2021) [HTY+21] used 14 EEG signals are measured at AF3, AF4, F3, F4, F7, F8, FC5, FC6, T7, T8, P7, P8, O1, and O2, which lie on all 4 lobes. Filter from 10Hz-50Hz ($\alpha.\beta, \gamma$ waves)

1.4.4 Visual Imagery

1.4.5 Audio Imagery

1.4.6 References

- Wikipedia - Brain–computer interface
- Microsoft Research | Developing a BCI based on Visual Imagery
- Sharbrough (1991) [Sha91]. *“American Electroencephalographic Society guidelines for standard electrode position nomenclature”*.
- Wolpaw et al. (2000) [WMV00]. *“Brain-computer interface research at the Wadsworth Center”*.

1.5 TODO List

- How different is the brain waves when we concentrate on different things?
- How can we ensure, measure human concentration?
- How do the brain waves differ in varied context (quiet room, on the street, party, relax, focused)?
- How noise is the brain waves in the above context?
- How was the data collected? Can we augmented the brainwave dataset? If yes, how, augment irrelevant waves?
- EFP?

1.6 References

- Wikipedia - Nervous System
- Wikipedia - Neural oscillation
- Wikipedia - Brain–computer interface

2 Computer Science

2.1 Binary Search

Is a search algorithm to find a target value in a sorted array. With each iteration, half the values are eliminated.

```
def binary_search(array, target):
  left = 0
  right = len(array) - 1
  while (left <= right):
    mid = (right + left) // 2
    if array[mid] == target:
      return mid
    elif array[mid] < target:
      left = mid + 1
    else:
      right = mid - 1
  return -1
```

2.2 Fenwick Tree

Fenwick Tree, a.k.a., Binary indexed tree, is a data structure for fast sum querying over a range and updating of an given array [Fen94]. Check videos: Fenwick Tree range queries, point updates, tree construction.

E.g.: Given array $A = \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline 5 & -3 & 6 & 1 & 0 & -4 & 11 & 6 & 2 & 7 \\ \hline\end{array}$, we want to compute the sum of elements between index $[i, j)$, but also updating the array time after time. There are two naive approaches that would not scale up very well for a large array and great number of updates and queries:

- Simply keep the original array for update. This way, we can do update in $\mathcal{O}(1)$, but do query in $\mathcal{O}(N)$
- Pre-compute the sum and store values in an array $P = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|}\hline 0 & 5 & 2 & 8 & 9 & 12 & 8 & 19 & 25 & 27 & 34 \\ \hline\end{array}$, such that $P[I] = \Sigma_{i=0}^{I} A[i]$. This way, we can do query in $\mathcal{O}(1)$, but do update in $\mathcal{O}(N)$

Fenwick tree's properties and implementation:

- Complexity of Fenwick tree:
  - Construction: $\mathcal{O}(N)$

  - Point update: $\mathcal{O}(\log N)$
  - Range sum: $\mathcal{O}(\log N)$
  - Range update: $\mathcal{O}(\log N)$
  - Adding index: N/A
  - Removing index: N/A
- Implementation LSB (least significant bit)

```cpp
inline int LSB(int id) { return id & ~(id-1); }
inline int LSB(int id) { return id & -id; }
```

- Tree struct with an array/vector:

```cpp
template <typename T>
struct FenwickTree
{
    vector<T> Array, Tree;              // Original array and the tree
    FenwickTree(int N) : Array(N + 1), Tree(N + 1) {}

    T query(int id) const {
        T sum = 0;
        for (; id > 0; id -= LSB(id))
            sum += Tree[id];
        return sum;
    }
    T query(int low, int high) const { return query(high) - query(low - 1); }

    void update(int id, LL val)    {
        Array[id] += val;
        int N = T.size();
        for (; id < N; id += LSB(id))
            Tree[id] += val;
    }
    // Update by setting value
    void set(int id, T val) { update(id, val - Array[id]); }

    T at(int id) const { return Array[id]; } // Get array value at id
};
```

2.3 Segment Tree

Segment Tree, a.k.a. statistic tree, is also a data structure for fast range querying and updating of an array. It's more complicate than Fenwick tree but can do more than taking the sum. Check blogs: geeksforgeeks.org, cp-algorithms.com.

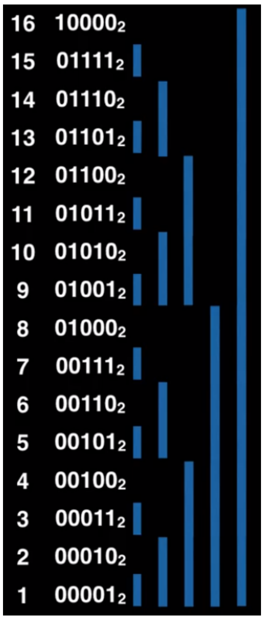

Figure 2.1: In a Fenwick tree, each cell is responsible for a range.

- It is also a binary tree
- Memory, construction, query complexity are: $\mathcal{O}(\log N)$

Two possible implementations:

- With array $1 \times (4N)$: slightly more complicate, but shorter

```
template <typename T>
class segment_tree {
  static T merge(const T &a, const T &b) { return a + b; }
  int length;
  vector<T> values;

  void build(int id, int low, int high, const vector<T>& array) {
    if (low == high) {
      values[id] = array[low];
      return;
    }
    int mid = low + (high - low) / 2;
    build(id * 2 + 1, low, mid, array);
    build(id * 2 + 2, mid + 1, high, array);
    values[id] = merge(values[id * 2 + 1], values[id * 2 + 2]);
  }

  T query(int id, int low, int high, int tgt_low, int tgt_high) const {
    if (low == tgt_low && high == tgt_high)
      return values[id];
    int mid = low + (high - low) / 2;
    if (tgt_high <= mid)
```

```
            return query(id * 2 + 1, low, mid, tgt_low, tgt_high);
        if (tgt_low > mid)
            return query(id * 2 + 2, mid + 1, high, tgt_low, tgt_high);
        return merge(
            query(id * 2 + 1, low, mid, tgt_low, mid),
            query(id * 2 + 2, mid + 1, high, mid + 1, tgt_high));
    }

    void update(int id, int low, int high, int target, const T &val) {
        if (target < low || target > high)
            return;
        if (low == high) {
            values[id] = val;
            return;
        }
        int mid = low + (high - low) / 2;
        update(id * 2 + 1, low, mid, target, val);
        update(id * 2 + 2, mid + 1, high, target, val);
        values[id] = merge(values[id * 2 + 1], values[id * 2 + 2]);
    }

public:
    segment_tree(const vector<T>& array)
    : length(array.size()), values(4 * length) {
        build(0, 0, length - 1, array);
    }
    T query(int low, int high) const { return query(0, 0, length - 1, low, high); }
    void update(int id, const T &val) { update(0, 0, length - 1, id, val); }
};
```

- With *Node**: slightly cleaner, but longer

```
template <typename T>
struct Node {
    T value_;
    int low_, high_;
    Node<T> *left_, *right_;

    Node() : value_(0), low_(0), high_(0), left_(nullptr), right_(nullptr) {};

    Node(int low, int high, const vector<T>& array)
    : low_(low), high_(high), left_(nullptr), right_(nullptr) {
        if (low_ == high_) {
            value_ = array[low_];
            return;
        }
        int mid_ = low_ + (high_ - low_) / 2;
```

```
      left_ = new Node(low_, mid_, array);
      right_ = new Node(mid_ + 1, high_, array);
      value_ = add(left_->value_, right_->value_);
  }

  T query(int low, int high) {
      if (low == low_ && high == high_)
         return value_;
      int mid_ = low_ + (high_ - low_) / 2;
      if (high <= mid_)
         return left_->query(low, high);
      if (low > mid_)
         return right_->query(low, high);
      return add(left_->query(low, mid_), right_->query(mid_ + 1, high));
  }

  void update(int id, T value) {
      if (id < low_ || id > high_)
         return;
      if (low_ == high_) {
         value_ = value;
         return;
      }
      left_->update(id, value);
      right_->update(id, value);
      value_ = add(left_->value_, right_->value_);
  }
};

template <typename T>
class ST {
  Node<T> *root;

  static void clean_up(Node<T>* node) {
      if (node->left_ != nullptr)
         clean_up(node->left_);
      if (node->right_ != nullptr)
         clean_up(node->right_);
      delete node;
  };
  public:
  ST(const vector<T>& array) { root = new Node(0, array.size()-1, array); }
  ~ST() { clean_up(root); }
  T query(int low, int high) const { return root->query(low, high); }
  void update(int id, const T& new_value) { root->update(id, new_value); }
};
```

2.4 Dijkstra Algorithm

3 Academic Writing

Simple and clear ideas are always better and have high number of citing. The ones that explain their approach in a simple and clear manner make it easier and more appealing for others to read their works. Thus it's easy for others to understand, apply their ideas and extend it.

3.1 Examples

Learning from examples is, without doubt, always great

3.1.1 Good examples

- A well structured paper, let alone the fact that every part is also well written [SQA+15]
  - Introduction: explains WHAT problem they tried to address, and WHY
  - Background
  - Main approach
    - A motivating example
    - Explain each components and changes in their approach, also with the problems that they solve.
- [AWR+17]
- Reading a long but well-explained paper is less tiring than reading a short but complicated one [SHM+16; SSS+17]

3.1.2 Bad examples

- makes large claims / comments without supporting evidences
- examples, images are not helpful / representative / illustrative
- not enough examples / explanation
- positioning of examples / explanation

The content is great, really, I read through it, but the way it was presented is not

- Learning to Sequence and Blend Robot Skills via Differentiable Optimization:

Bibliography

[AWR+17] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. "Hindsight Experience Replay". In: *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)* 30 (2017).

[Fen94] P. M. Fenwick. "A new data structure for cumulative frequency tables". In: *Software: Practice and experience* 24.3 (1994), pp. 327–336.

[HTY+21] S. Hoshino, T. Tagami, H. Yagi, and K. Kanda. "Brain-Mobility-Interface based on Deep Learning Techniques for Classifying EEG Signals into Control Commands". In: *2021 IEEE/SICE International Symposium on System Integration (SII)*. IEEE. 2021, pp. 150–156.

[PN01] G. Pfurtscheller and C. Neuper. "Motor imagery and direct brain-computer communication". In: *Proceedings of the IEEE* 89.7 (2001), pp. 1123–1134.

[Sha91] F. Sharbrough. "American Electroencephalographic Society guidelines for standard electrode position nomenclature". In: *J clin Neurophysiol* 8 (1991), pp. 200–202.

[SHM+16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *nature* 529.7587 (2016), pp. 484–489.

[SQA+15] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. "Prioritized Experience Replay". In: *arXiv preprint arXiv:1511.05952* (2015).

[SSS+17] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. "Mastering the Game of Go without Human Knowledge". In: *nature* 550.7676 (2017), pp. 354–359.

[WMV00] J. R. Wolpaw, D. J. McFarland, and T. M. Vaughan. "Brain-computer interface research at the Wadsworth Center". In: *IEEE transactions on rehabilitation engineering* 8.2 (2000), pp. 222–226.