Visual Receptive Field Development

CS-479 Mini Project 1

Spring Semester 2025

Grading and submission

Grading The mini-project is **mandatory** and counts towards 30% of the final grade. You will be graded on the results you obtained and your answers to the questions, and a fraction of the grading is also dedicated to the quality and clarity of the report.

Teams You will have to work in teams of two students. To find a project partner if you don't have one yet, you can use the EdStem forum (`https://edstem.org/eu/courses/2085/discussion`).

Final report Your submission should be the interactive notebook .ipynb. Running it should reproduce all your results and plots (but submit it with the results and plots already displayed). The code itself will not be graded, only your answers to the questions. Clearly label your answers to the questions and the plots that form part of the answer. Write **concisely** in complete English sentences and present figures carefully (axis labels, legends etc.). Write **concisely**: go straight to the point in answering the question.

Submission You will have to submit your .ipynb notebook on the Moodle page. The file name should have the structure `MPX_NameMember1_NameMember2.ipynb`, where `X` is the miniproject number (either 1 or 2). If you work in teams both of you should submit the same file. On Moodle, make sure you press the submit button before the deadline.

Deadline The deadlines to submit notebook are:

- May 26th 2025, at 11.55 pm (fraud-detection slots: May 27th and 28th, 2025);
- June 2nd 2025, at 11.55 pm (fraud-detection slots: June 3rd and 4th, 2025).

The early deadline allows you to do the fraud detection before the exam preparation period, which you might find convenient.

Regulations and fraud detection After the submission, we will perform a fraud detection interview which consists of a few questions about your report and your code. You will need to attend **in-person** (both members of the team). Choose your deadline accordingly.

The mini-project is graded and the same rules as for exams apply ("Ordonnance sur le contrôle des études à l'EPFL"). In short: EPFL takes any form of cheating and plagiarism very serious. The mini-project has to be your own work. You are not allowed to share code or answers across teams. However you can discuss ideas between teams and ask questions on the Ed forum. If you split the work between the two team members, both team members have to understand all parts of the mini-project. Note: The goal of this meeting is fraud detection; it is neither an exam nor a presentation.

Introduction

In sensory processes, a receptive field (RF) describe the subset of inputs a given neuron will respond to. For example, inner hair cells in the cochlea respond to a specific band of frequency, and transduce their signal to neurons in the auditory cortex. Thus neurons in the (primary) auditory cortex have a specific frequency response profile[2], which is a receptive field for auditory signal. In this miniproject, we will explore the receptive fields of visual signals. We will see how Hebbian learning can explain the formation of specific and localized receptive fields in the primary visual cortex V1[1].

We will consider a single layer of N_{out} neurons $y \in \mathbb{R}^{N_{out}}$ receiving inputs from N_{in} pre-synaptic neurons $x \in \mathbb{R}^{N_{in}}$ who encode pixel intensity for a 16×16 patch from an image. The response, i.e., spiking frequency, of the output neurons y is given for all $1 \leq i \leq N_{out}$ by $y_i = (w_i^T x)_+$ where $(-)_+$ is the linear rectifier and the weights $w_i \in \mathbb{R}^{N_{in}}$ are the receptive field of (output) neuron y_i. If we arrange the weights as rows of a matrix $W \in R^{N_{out} \times N_{in}}$, we can write,

$$y = (Wx)_+. \tag{1}$$

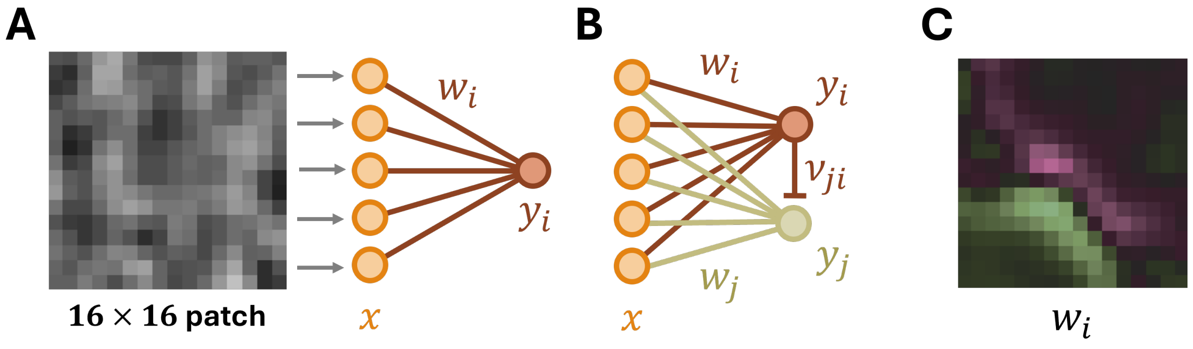

Figure 1: **Visual Network & Receptive Field**
(A) Single neuron visual network. (B) Competitive multi-neuron visual network with lateral inhibition ν. (C) Receptive Field w_i associated with neuron y_i.

As you saw in the lectures, Hebbian learning consist in updating the weights using only local information:

$$\Delta w_i \propto f(w_i^T x) \cdot x. \tag{2}$$

The function f is sometimes called the effective Hebbian non-linearity and describe the contribution of the post-synaptic potential $w_i^T x$ to the weight change. We will consider various effective non-linearities, one of which you may already be familiar with, $f(w_i^T x; \theta) = y_i(y_i - \theta)$ where $y_i = (w_i^T x)_+$ and $\theta_i := \langle y_i^2 \rangle$, the BCM rule.

Getting Started

1. Download the dataset of ten bitmap (.bmp) images and load them using Image.open(*path*) from the PIL library (Pillow).

2. Standardize each image such that the pixel have zero mean and unit variance

3. Extract 5000 patches of size 16×16 pixels per image, flatten and organize them into a dataset.

1 Single Neuron RF

In this first part we will focus on a single neuron ($N_{out} = 1$), see Fig. 1A, and explore the development of its receptive field (Fig. 1C) using the BCM rule.

1. Setup a *visual_network* class with the following functions:
   - *__init__*(N_{in}, N_{out}, *seed*): initialize the network with $W \in \mathbb{R}^{N_{out} \times N_{in}}$ (here $N_{out} = 1$) with i.i.d. $\mathcal{N}(0, 1)$ weights. Also allow to specify a seed for **reproducibility**!
   - *forward*(x): implements the neuron model of Eq. 1.
   - $f(u; \Theta)$: the effective non-linearity with current $u := w_i^T x$, by default implement the BCM rule where $\theta_i := \langle y_i^2 \rangle$ ($\Theta := \{\theta\}$). In general Θ will be a set of additional parameters.
   - *update*(x, γ, Θ): updates the weights following the rule of Eq. 2 using a learning rate γ, i.e., $\Delta w_i = \gamma f(w_i^T x; \Theta) \cdot x$. Importantly, the weights w_i are **re-normalized**, i.e. $||w_i|| = 1$, after each update for stability!

2. Train your network using the BCM rule with a fixed $\theta = 1$ for $n_{iter} = 150 \cdot 10^3$ iterations (random patch presentation) and $\gamma = 10^{-4}$.

3. Display the RF (i.e., w_1) using the *show_RF* function. Is the RF a random pattern? If not what kind of pattern is neuron y_1 responding to?

4. Compare the activation of your neuron y_1 before and after training. In particular, plot the difference of spiking frequency Δy before and after training in a histogram. Display some of the patches which had their spiking frequency increased. What about the patches with decreased response, are they meaningful?

5. How could you evaluate if the response to your trained RF is significantly different to a random RF? (You don't need to do it, explain a statistical test you could conduct)

6. Repeat 1.2. for $K = 20$ trials, each with a different seed, and display the resulting RFs (using the *show_RFs* function). What do you observe?

7. We will now allow θ to vary, in particular it will estimate $\langle y_1^2 \rangle$ using a running-average,

$$\Delta\theta = \frac{1}{\tau}(y_1^2 - \theta). \tag{3}$$

   Train your network over $K = 20$ trials (use a different seed for each trial) with the BCM rule for $n_{iter} = 150 \cdot 10^3$ and $\gamma = 10^{-5}$. Initialize $\theta = 0$ and update θ after each patch is presented using the above rule with $\tau = 10^2$.

8. Display the resulting RFs. Are the patterns the same as in 1.6.? What happens if you reduce γ? What happens if you increase γ?

9. Explain the role of a variable θ in developing (or hindering) the development of diverse RFs.

10. Try out at least two other Hebbian non-linearities and repeat 1.6. You could consider $f(u) = (u - \theta)_+$ (θ is fixed) or any function of your choice! What happens if your "non-linearities" is linear?

11. How could you adjust the update rule to avoid explicit re-normalization of the weights?

12. (**Bonus**) Train your network as in 1.6. or 1.7. on patches taken from another set of images of your choice. Are the RF patterns different to the ones you obtained previously? If so why?

2 Extracting RF

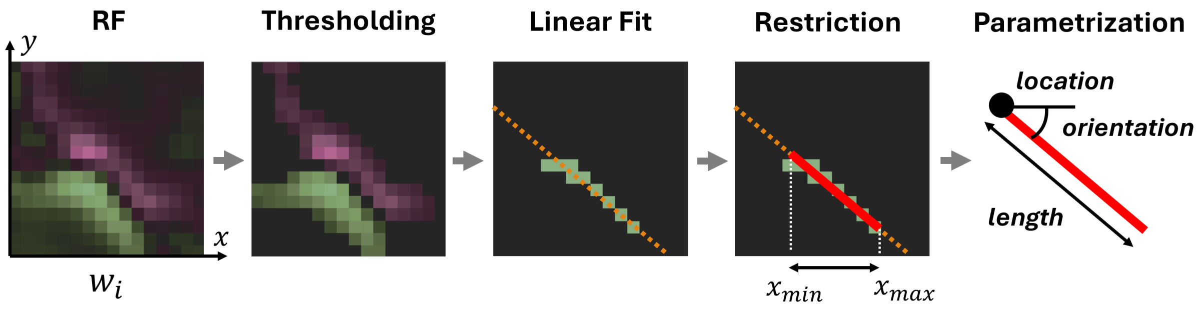

Figure 2: **RF extraction process**
From left to right: the RF w_i is thresholded; a linear regression is fitted to the boundary points; the regression line is restricted to a range of x-coordinates such that the corresponding segment fits the boundary point cloud; length, orientation and location are computed.

In this part, we will try to extract parameters for the RF patterns in order to quantify their diversity. We will parametrize RF in form of stripes by the length, orientation and location of the 'main stripe'. In the following, you will be tasked with creating a function which can extract those parameters (see Fig. 2).

1. Given a RF w, reshape it into a 16×16 matrix W. Threshold W by only keeping values which are greater in absolute value than $\lambda \geq 0$. We will refer to the thresholded W as W_λ

2. Create a function *is_boundary*(W_λ, i, j) which return true if the set of pixel $W_{\lambda,ij}$ and its direct neighbours contains **both** strictly positive and negative values, and false otherwise (see Fig. 2 Linear Fit for a visualization).

3. Create a function *find_boundary*(W_λ) which return the coordinates of boundary points in W_λ using *is_boundary*.

4. Show the resulting boundary for a RF of your choice from the previous section for various λ. What happens when $\lambda = 0$?

5. Create a function *linear_regression*(X_b) which fit a simple linear regression $\hat{y} = \alpha x + \beta$ to the set of boundary points $X_b := \{(x, y) : \textit{is\_boudary}(W_\lambda, x, y)\}$, and return its intersect $\hat{\alpha}$ and slope $\hat{\beta}$.

6. Create a function *find_parameters*(X_b, α, β). The function needs to find an appropriate range of x-coordinates from the set of boundary points X_b so that the linear regression

restricted to this range match the size of the point cloud (see Fig. 2 Restriction). It then computes the length of the segment, its orientation, i.e., angle with the x-axis and its location (start of the segment).

7. Using your functions, compute and display a parametrized RF (i.e. segment) of your choice using its length, orientation and location for various λ. Using RFs of Section 1, show that it can capture RFs of various lengths, orientation and locations.
8. Overlay the parametrized RFs of 1.7. on a single plot representing a patch for various λ. Compare with the same plot but computed from the RFs of 1.6. How does the spread of length, orientation and location change?
9. Is this parametrization appropriate? Could you suggest an alternative parametrization or representation of the RF?

3 Multi-Neuron RF

3.1 Non-Competitive RF

We will now consider a network made up of $N_{out} > 1$ output units. First, we consider the neurons without interactions between each other.

1. Adjust *visual_network* such that $y \in \mathbb{R}^{N_{out}}$ and in particular $W \in \mathbb{R}^{N_{out} \times N_{in}}$, where each row w_i of W corresponds to the RF of output neuron y_i.
2. Train your network for $N_{out} = 20$ using the BCM rule with fixed $\theta = 1$, $n_{iter} = 150 \cdot 10^3$, $\gamma = 10^{-4}$. Display the RFs (using *show_RFs*) for at least two seeds.
3. Contrast your RFs to the RFs from 1.6. What is different in this setup to 1.6.? What can you conclude about diversity of RF formation in a non-competing context for fixed θ (think about the source of randomness)?
4. Implement the variable θ from 1.7., in particular $\theta \in \mathbb{R}^{N_{out}}$ where θ_i is an estimate of $\langle y_i^2 \rangle$. The update rule is,
$$\Delta\theta_i = \frac{1}{\tau}(y_i^2 - \theta_i) \tag{4}$$
Train your network with the same parameters as 3.1.2. at the exception of $\gamma = 10^{-5}$, with initial $\theta_i = 0$ for all i and $\tau = 10^2$. Display the resulting RFs for at least two seeds.
5. What is the effect of θ on RF development in a non-competing context (think about the source of randomness)?

3.2 Competitive RF

Now, we will add competition between output neurons through lateral inhibitory projections between them. The activity of a given neuron y_i will now depend on all neurons y_j with $j \neq i$, mediated by lateral inhibitory weights $V \in \mathbb{R}_{\geq 0}^{N_{out} \times N_{out}}$ (see Fig. 1B). The model can be formulated as:
$$y = (Wx - Vy)_+. \tag{5}$$

Notice that this is a fixed point equation, we cannot simply evaluate y as some explicit function of x. To solve it we will use a differential equation formulation for the current u:

$$\dot{u}(t) = -u(t) + Wx - V(u(t))_+. \tag{6}$$

The steady-state u^* of this equation is such that $y := (u^*)_+$ is a solution to Eq. 5.

1. Create a new *competitive_visual_network* class with the following functions:
   - *__init__*$(N_{in}, N_{out}, seed)$: initialize the network with $W \in \mathbb{R}^{N_{out} \times N_{in}}$ with i.i.d. $W_{ij} \sim \mathcal{N}(0,1)$, and $V \in \mathbb{R}_{\geq 0}^{N_{out} \times N_{out}}$ with $V_{ii} = 0$ and i.i.d. $V_{ij} \sim |\mathcal{N}(0,1)|$ for $i \neq j$. Also allow to specify a seed for **reproducibility**!
   - *forward*(x): run the differential Eq. 6 using a forward Euler method with $dt = 0.1$ for $n_{step} = 10$ to obtain an approximation of u^* and then return $y := (u^*)_+$ and current u^*.
   - $f_W(u;\theta)$: the effective non-linearity for W with current u, by default implement the BCM rule where $\theta_i := \langle y_i^2 \rangle$.
   - $f_V(u;\phi)$: the effective non-linearity for V with current u. Here we will use the simple rule $f_V(u;\phi) = (u)_+ - \phi$.
   - *update*$(x, \gamma, \theta, \phi)$: computes the steady-state current u^* and activity y using *forward*. Then updates the weights with learning rate γ according to the rules,

     $$\Delta w_i = \gamma f_W(u_i^*; \theta_i) \cdot x,$$
     $$\Delta v_i = \gamma f_V(u_i^*; \phi_i) \cdot y,$$

     where w_i and v_i are the rows of W and V. Importantly, the weights w_i are **re-normalized** ($||w_i|| = 1$), the weights v_i are enforced to be **positive** (discard the negative part) and set $v_{ii} = 0$ after each update.
2. Train your network with $N_{out} = 20$ units using the BCM rule with a fixed $\theta = 1$ and $\phi = 0$ for $n_{iter} = 150 \cdot 10^3$ and $\gamma = 10^{-4}$. Display the resulting RFs (weights W) as well as the lateral inhibition V. How does it compare to the previous networks' RFs?
3. Compare the activation y of your competitive network against the non-competitive *visual_network* for a patch of your choice. Which one is sparser?
4. For a neuron of your choice (choose a non-random RF), compare the activation y before and after training in an histogram. Display a few patches with increased response.
5. Implement a variable ϕ with rule,

   $$\Delta \phi_i = \frac{1}{\tau}(y_i - \phi_i). \tag{7}$$

   What is ϕ approximating with this rule?
6. Train your network as in 3.2.2. with variable ϕ initialized with $\phi_i = 0$ and $\tau = 10^2$. Display the resulting RFs and V. What do you observe?
7. (**Bonus**) Is the variable θ from the BCM rule necessary? Try the variable θ with both fixed and variable ϕ.
8. Using your parametric extraction of RFs from part 2., overlay the RFs as in 2.8.
9. Discuss the difference(s) in diversity and convergence of RFs for a variable θ for BCM in a non-competitive setting versus a lateral-inhibition competitive setting. Do you see advantage and disadvantage from the perspective of brain development?

References

[1] Carlos S. N. Brito and Wulfram Gerstner. Nonlinear hebbian learning as a unifying principle in receptive field formation. *PLOS Computational Biology*, 12(9):1–24, 09 2016.

[2] Dale Purves, George J Augustine, David Fitzpatrick, C Katz Lawrence, Anthony-Samuel Lamantia, James O McNamara, and S Mark Williams. *Neuroscience 2nd edition*, chapter The Auditory Cortex. Sinauer Associates, 2001.