Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Corso di Laurea in Matematica

A Metric Model of Brain Architectures and Brain-Inspired Artificial Neural Networks

Tesi di Laurea in Analisi Matematica

Relatore:
Chiar.ma Prof.ssa
Giovanna Citti

Presentata da:
Eleonora Fabbri

Anno Accademico 2023/2024

Introduction

The aim of this thesis is to use notions from analysis on metric measure spaces to develop an effective and realistic model for the geometry of connections occurring in the primary visual cortex (V1). Building on this foundation, this work is inspired by the PhD thesis of Montobbio [2], which provides a metric framework to describe the functional architecture of V1. In particular, we present a reformulation of some key ideas using the mathematical tools at our disposal, aiming to synthesize the main results in a concise and accessible manner.

V1 is particularly interesting because it is the first area of the visual cortex responsible for processing visual information collected by the retina and the lateral geniculate nucleus (LGN).
In the 1960s, D.H. Hubel and T.N. Wiesel described the geometry of V1 [1], basing their studies on two main types of neurons: simple cells and complex cells. They also introduced the concept of receptive profiles (RPs) of these cells to describe how they respond to light stimuli. In facts, Huber and Wiesel discovered that the cortical neurons showed distinct selectivity to a larger number of different features, including orientation, colour, shapes and movement. The action of RPs is approximately linear, so that they are generally modelled as a family $\{\psi_p\}_{p\in\mathcal{G}} \subseteq \mathcal{L}^2(R^2)$ of linear filters, where $\mathcal{G}$ represents the feature space, which encompasses the specific characteristics of the set we are considering. The response of a RP to an image I is then described by the integral:

$$O_p(I) = \int_D I(x,y)\,\psi_p(x,y)\,dx\,dy,\ p \in \mathcal{G}.$$

Simple cells RPs are well approximated by a specific set of filters known as Gabor filters, whose feature space is $\mathcal{G} = \mathbb{R}^2 \times S^1$, highlighting the sensibility of the cells to their spatial position and orientation. Gabor filters' feature space is a well-known group and we can use its characteristics to underline some invariance in the structure we will define.

We will define a distance function $d : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$, such that

$$d(p, p_0) := \|\psi_p - \psi_{p_0}\|_{\mathcal{L}^2(\mathbb{R}^2)}$$

and a kernel $K : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$, defined as

$$K(p, p_0) := \mathrm{Re}\langle \psi_p, \psi_{p_0} \rangle_{\mathcal{L}^2(\mathbb{R}^2)}.$$

With the assumption $\|\psi_p\|^2_{\mathcal{L}^2(\mathbb{R}^2)} = \|\psi_{p_0}\|^2_{\mathcal{L}^2(\mathbb{R}^2)} = t$, $\forall\, \psi_p \in \{\psi_p\}_{p\in\mathcal{G}}$, with $t \in \mathbb{R}$, we obtain that $d^2(p, p_0) = 2(t - K(p, p_0))$.
Since the kernel $K(p, p_0)$ is defined as the inner product of the RPs ψ_p and ψ_{p_0}, it quantifies the similarity between their responses and it can be interpreted as a measure of correlation between the neurons indexed by $p, p_0 \in \mathcal{G}$. In this way, this model accounts for horizontal connections, which involve interactions of neurons within the same layer.

In the final part of our discussion, we will focus on neural networks, in particular on convolutional neural networks (CNNs), to outline the differences and similarities between these networks and the architecture of the human brain. We will also try to bridge the gap between the two, incorporating the biologically inspired structures we have defined in the first chapters into models.

Contents

List of Figures

Chapter 1

Neurophysiology and Visual Information Processing

In this first chapter, we provide a brief overview of the functionality of the visual cortex. These concepts, and in particular the structure of receptive fields and the connectivity feature, will be essential to develop a metric for the primary visual cortex.

1.1 The visual pathway

We start by outlining how the visual information is spread to the brain.
The visual pathway is a complex network responsible for processing visual information from outside [3]. It begins in the retina, which captures light with its photoreceptors and converts it into electrical signals. Then these signals are transmitted through a pathway of neurons from the eye to the brain. A schematic representation of this pathway is shown in Figure 1.1, which illustrates its main components and how they are connected.

Specifically, the visual pathway is composed by:

- the *retina*: the innermost layer of the eye, contains neurons that are sensitive to light;
- *optic nerve*: it is a paired cranial nerves, made of retinal ganglion cell axons and

glial cells;

- *optic chiasma*: the point where the optic nerves from both eyes partially cross, allowing signals from the inner halves of each retina to switch sides and reach the opposite hemisphere of the brain;
- *optic tracts*: contains retinal information from both eyes;
- *lateral geniculate nucleus* (LGN): it is the link between the retina and the cortex, works like a processing station of the visual information, mediating vision and visual perception;
- *primary visual cortex* (V1): the first area of the visual cortex to receive the sensory input from the LGN

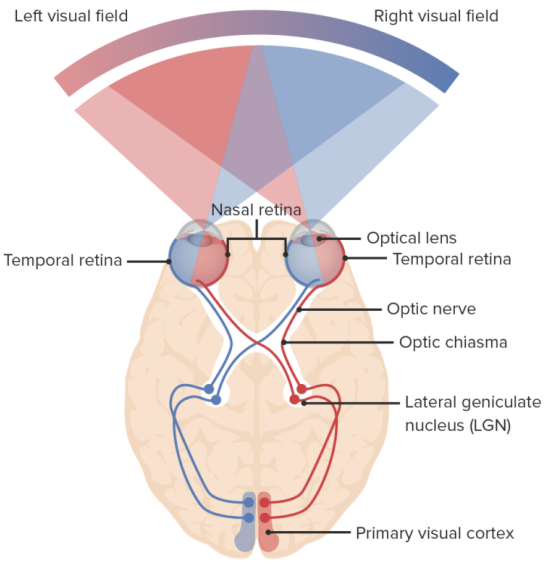

Figure 1.1: The visual pathway

The primary visual cortex

The visual cortex is the part of the cortical mantle of the brain that receives, integrates and processes visual information relayed from the retina and LGN. It is categorized into several functionally distinct areas, typically labeled as V1, V2, V3, V4, V5,

V6. Among these, V1 is the first cortical region to receive and process visual information, and it is also the best-understood one.
There are two main types of neurons in V1, which were first discovered by D.H. Hubel and T.N. Wiesel [1]: *simple cells* and *complex cells*. Simple cells show orientation selectivity, i.e. they respond to specific stimuli, such as edges and lines with a particular orientation. Complex cells, responsive to these and other features, integrate inputs from many simple cells.
In our discussion we mainly focus on simple cells, with complex ones treated as a generalization.

1.2 Receptive fields and profiles

We call *receptive field* (RF) of a neuron the region of the retina that, when stimulated, affects its activity [4]. The RF of a cell is a specific area of the visual field, and since each point in this field is projected onto the retina R , we can consider $D \subseteq R$ and refer to retinal coordinates.

We introduce the *receptive profile* (RP) of a cell as a function $\psi : D \to \mathbb{R}$, which describes the response of the neuron to light stimuli (in an excitatory or in an inhibitory way). We can assume that simple cells respond linearly to visual stimuli, so that, in general, to simplify our approach, we assume that visual stimuli satisfy $I \in \mathcal{L}^2(R)$ where R denotes the retina (the metric on R will be defined later). We also assume by simplicity that I is real-valued, and represents grayscale stimuli.

Since we assumed that the response of visual neurons to a light signal is close to a linear function, we can model the local response of a single cell like:

$$O_p(I) = \int_D I(x, y)\, \psi_p(x, y)\, dx\, dy,\ p \in \mathcal{G},$$

where D is the RF, ψ_p is the RP of a specific cell and I is the visual stimulus.

Physiological studies indicate that each point $(x, y) \in R$ on the retina transmits information to multiple groups of cells, each with a distinct functional role. For this reason, we consider sets of RPs of groups of cells $\{\psi_p\}_{p\in\mathcal{G}}$, where the *feature space* $\mathcal{G}$ is a set of indexes representing different features. Each $p \in \mathcal{G}$ corresponds to a specific feature to which the receptive profile ψ_p has the strongest response. Consequently, $p \in \mathcal{G}$ can be defined like $p = (x, y, f)$, where $(x, y) \in \mathbb{R}^2$ is a couple of coordinates on the retina R and f is the encoded feature.

The shape of a cell's RF gives valuable insight on the functionality of the cell and neurons in different areas can act quite differently. For example, RFs of retinal ganglion cells and LGN neurons, sensible to scalar features, are concentric and compact, whereas neurons in V1, sensible to a vector feature, have elongated RF and directional RP.

Indeed, RFs become progressively more complex as information moves up the hierarchy of visual areas [4]. This hierarchical organization is achieved through a process known as *pooling*, where information from smaller receptive fields is combined to form more sophisticated and abstract representations of visual stimuli. As a result, different areas of the visual cortex specialize in processing distinct aspects of the visual input: V1 is sensible for orientation, V2 for complex contours and patterns, V3 detects the shapes of moving objects, V4 is specific for colors and V5 for movements and depth.

We can model the simultaneous response of each RP as the convolution between the stimulus and the filter:

$$O_p(I) = \int_R I(x, y)\, \psi_p(x, y)\, dx\, dy = (I * \psi_p)(x, y),\ p \in \mathcal{G}.$$

Lateral connections

Visual signals are propagated through the visual pathway, following a specific hierarchy of regions, each involving progressively more complex features. Actually, neural responses are influenced by *feedback* signals from higher-level areas and intra-area connections, referred to as *horizontal connections*, which are commonly found in V1 [2]. Horizontal (or lateral) connections link neurons across different *hypercolumns*, which are groups of neurons sensitive to approximately the same retinal position but responsive to all possible orientations; thus, these relations associate neurons with different RFs and orientation specificity. In the last chapter of this discussion, we will explore how this neural hierarchy play a fundamental role in building biological inspired neural networks.

1.3 Retinotopy

The retina captures a two-dimensional representation of the visual field. When visual information is transmitted in V1, a point-by-point correspondence is maintained between adjacent regions of the visual field, creating a projection of the retinal structures. This phenomenon of mapping visual input from the retina to the cortex is known as *retinotopy* [4]. Clearly, during this process, the signal encounters a deformation d, which, near the center of the visual field can be well-approximated by the complex logarithmic function:

$$d_{a,k}(z) = k \cdot ln(z + a)$$

where a, k are parameters that depends on the species being studied, and $z \in \mathbb{C}$.

As a result, the visual field is represented topographically.

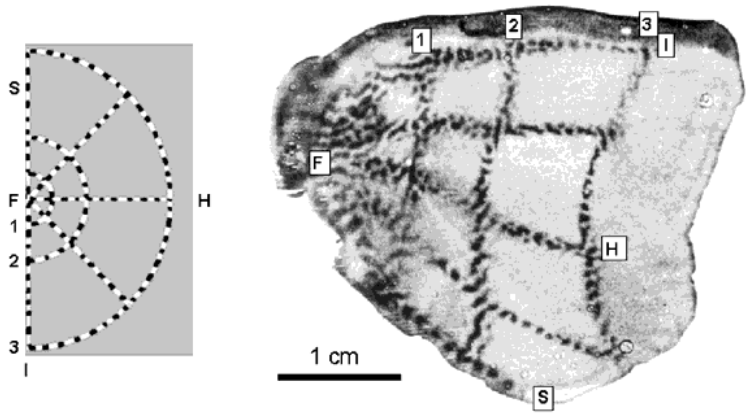

Figure 1.2: The visual stimulus used to estimate the retino-cortical mapping (on the left) and the flattened visual cortex of a macaque with the corresponding activated regions (on the right).

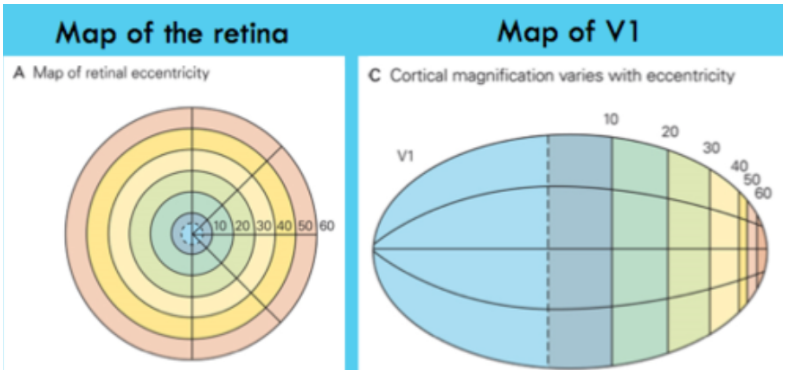

Figure 1.3: Model of retinotopy.

Chapter 2

From receptive profiles to cortical distances

2.1 Metric measure spaces

In this section, we recall some key concepts related to distances, length spaces, and measures [2]. We will discuss the fundamental notions underlying these topics, presenting the necessary definitions, propositions and theorems to understand the discussion that follows.

Definition 2.1 *Given an arbitrary set X, the function $d : X \times X \to \mathbb{R} \cup \{+\infty\}$ is a* distance *on X if:*

(i) $d(p, q) > 0 \ \ \forall\, p, q \in X$, *with* $p \neq q$, *and* $d(p, p) = 0$

(ii) $d(p, q) = d(q, p) \ \ \forall\, p, q \in X$

(iii) $d(p, q) \leq d(p, s) + d(s, q) \ \ \forall\, p, s, q \in X$

The space (X, d) is called a metric space.

Definition 2.2 *A continuous map $\gamma : [a, b] \to X$ is called a* path.

Definition 2.3 *Given a topological space X, $(A(X), L)$ is a* length structure, *if $A(X)$ is a set of paths on X, which is closed for restrictions, concatenations and linear reparameterizations of paths, and $L : A(X) \to \mathbb{R}$ is a function such that:*

(i) $L(\gamma) \geq 0$ for all $\gamma \in A(X)$

(ii) $L(\gamma_{|[a,b]}) = L(\gamma_{|[a,c]}) + L(\gamma_{|[c,b]})$ for all $c \in [a, b]$

(iii) *The function $t \longmapsto L(\gamma_{|[a,t]})$ is continuous on $[a, b]$*

(iv) $L(\gamma \circ \varphi) = L(\gamma)$ for all φ *linear homeomorphism*

(v) $\inf\{L(\gamma) \mid \gamma(a) = p, \gamma(b) \in X \setminus U_p\} > 0 \ \ \forall\ p \in X$ and for all neighborhoods $U_p \subseteq X$ of p

A path $\gamma \in A(X)$ is called admissible, *and $L(\gamma)$ is the length of γ.*

Definition 2.4 *Given $(A(X), L)$, for all $p, q \in X$ we define:*

$$d_L(p, q) := \inf\{L(\gamma) \mid \gamma : [a, b] \to X, \gamma \in A(X), \gamma(a) = p, \gamma(b) = q\}$$

Definition 2.5 *A length structure $(A(X), L)$ is* complete *if $\ \forall\ p, q \in X,\ \exists\ \gamma \in A(X)$ joining p and q such that $L(\gamma) = d_L(p, q)$. In this case, the distance d_L is called strictly intrinsic.*

The metric space (X, d) is called a length space *if there exists a length structure $(A(X), L)$ such that d coincides with the distance d_L.*

The metric space (X, d) is a geodesic space *if d_L is strictly intrinsic.*

Definition 2.6 *Given a set X, $\mathcal{A}$ is a σ-*algebra *on X, where $\mathcal{A} = \{A_i\}_{i \in I}, A_i \subseteq X$, if:*

(i) $\emptyset, X \in \mathcal{A}$

(ii) *if $A, B \in \mathcal{A} \Rightarrow A \setminus B \in \mathcal{A}$*

(iii) *if $\{A_i\}_{i \in I} \subseteq \mathcal{A}$ is a finite or countable collection $\Rightarrow \bigcup_{i \in I} A_i \subseteq \mathcal{A}$*

(iv) *if $\{A_i\}_{i \in I} \subseteq \mathcal{A} \Rightarrow \bigcap_{i=1}^{+\infty} A_i \subseteq \mathcal{A}$*

Given $\mathcal{G} = \{X_i\}_{i\in I}, X_i \subseteq X$, there exists a unique minimal $\mathcal{A}$ σ-algebra such that $\mathcal{G} \subseteq \mathcal{A}$, which is called the σ-algebra generated *by $\mathcal{G}$.*

Definition 2.7 *The function $\mu : \mathcal{A} \to \mathbb{R}_+ \cup \{+\infty\}$ is a* measure *on the σ-algebra $\mathcal{A}$ on X if:*

(i) $\mu(\emptyset) = 0$

(ii) *if $\{A_i\}_{i\in I} \subseteq \mathcal{A}$ is a finite or countable collection of subsets of X, where $A_i \cap A_j = \emptyset \; \forall i, j \in I$ such that $i \neq j \Rightarrow \mu\left(\bigcup_{i\in I} A_i\right) = \sum_{i\in I} \mu(A_i)$.*

Properties of a measure μ on $\mathcal{A}$:

1. if $A, B \subseteq \mathcal{A}$ such that $A \subseteq B \Rightarrow \mu(A) \leq \mu(B)$
2. if $A, B \subseteq \mathcal{A}$ such that $A \subseteq B \Rightarrow \mu(B \setminus A) = \mu(B) - \mu(A)$

$A \subseteq X$ is measurable *if $A \in \mathcal{A}$.*

Definition 2.8 *Given a topological space X, the σ-algebra generated by the set of all its open sets is called the* Borel σ-algebra *of X. A measure defined on the Borel σ-algebra is called a* Borel measure *over X.*

(X, d, μ) is a metric space, where X is a set, d is a distance on X and μ is a measure on the Borel σ-algebra of (X, d).

Definition 2.9 *Given a Hausdorff (T2) topological space X, a measure μ on the Borel σ-algebra of X is called a* Radon-measure *if:*

(i) $\forall A \subseteq X$ *open set,* $\mu(A) = \sup\{\mu(K) \mid K \subseteq A,\ K$ is compact$\}$ *(inner-regularity of μ)*

(ii) $\forall B \subseteq \mathcal{B}$, *where $\mathcal{B}$ is a Borel set,* $\mu(B) = \inf\{\mu(A) \mid B \subseteq A,\ A$ is open$\}$ *(outer-regularity of μ)*

(iii) $\forall\, x \in X, \exists U \subseteq X$ *a neighborhood of x, such that $\mu(U) < +\infty$ (μ is locally finite)*

2.2 RPs generated metric

In this section, we start describing the model presented in [2] by defining a metric space based on the receptive profiles (RPs) of simple cells. Drawing on biological structures from the retina to V1 we use filters (RPs) to link points within the visual pathway. However, to establish connections between points within V1 itself and to extend this construction iteratively across subsequent brain layers, we introduce a distance function to serve as a kernel on V1. To construct this kernel on the group $\mathbb{R}^2 \times S^1$, we employ a specific family of filters known as Gabor filters.

Definition 2.15 *Given a family $\{\psi_p\}_{p\in\mathcal{G}}$, where $\psi_p : \mathbb{R}^2 \to \mathbb{R}$ (or $\mathbb{C}$), $\psi_p \in \mathcal{L}^2(\mathbb{R}^2)$ $\forall\, p \in \mathcal{G}$, we call $\mathcal{G}$ the* feature space *associated with the family $\{\psi_p\}$.*

We define $d : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ as a distance function such that:

$$d(p, p_0) := \|\psi_p - \psi_{p_0}\|_{\mathcal{L}^2(\mathbb{R}^2)},$$

and a generating kernel *$K : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ such that:*

$$K(p, p_0) := \mathrm{Re}\langle\psi_p, \psi_{p_0}\rangle_{\mathcal{L}^2(\mathbb{R}^2)}.$$

Since d is defined as a restriction of the $\mathcal{L}^2$ distance function, it is itself a distance on $\mathcal{G}$.

Observe that:

$$d^2(p, p_0) = \|\psi_p\|^2_{\mathcal{L}^2(\mathbb{R}^2)} + \|\psi_{p_0}\|^2_{\mathcal{L}^2(\mathbb{R}^2)} - 2\,\mathrm{Re}\langle\psi_p, \psi_{p_0}\rangle_{\mathcal{L}^2(\mathbb{R}^2)}.$$

In facts, just making some basic calculations, we obtain:

$$\begin{aligned} d^2(p, p_0) &= \|\psi_p - \psi_{p_0}\|^2_{\mathcal{L}^2(\mathbb{R}^2)} \\ &= \langle\psi_p - \psi_{p_0}, \psi_p - \psi_{p_0}\rangle_{\mathcal{L}^2(\mathbb{R}^2)} \\ &= \langle\psi_p, \psi_p\rangle_{\mathcal{L}^2(\mathbb{R}^2)} - \langle\psi_p, \psi_{p_0}\rangle_{\mathcal{L}^2(\mathbb{R}^2)} - \langle\psi_{p_0}, \psi_p\rangle_{\mathcal{L}^2(\mathbb{R}^2)} + \langle\psi_{p_0}, \psi_{p_0}\rangle_{\mathcal{L}^2(\mathbb{R}^2)} \\ &= \|\psi_p\|^2_{\mathcal{L}^2(\mathbb{R}^2)} + \|\psi_{p_0}\|^2_{\mathcal{L}^2(\mathbb{R}^2)} - 2\,\mathrm{Re}\langle\psi_p, \psi_{p_0}\rangle_{\mathcal{L}^2(\mathbb{R}^2)}. \end{aligned}$$

With the assumption $\|\psi_p\|^2_{\mathcal{L}^2(\mathbb{R}^2)} = \|\psi_{p_0}\|^2_{\mathcal{L}^2(\mathbb{R}^2)} = t, \forall\, \psi_p \in \{\psi_p\}_{p\in\mathcal{G}}$, with $t \in \mathbb{R}$, we get that:

$$d^2(p, p_0) = 2t - 2\,\mathrm{Re}\langle\psi_p, \psi_{p_0}\rangle_{\mathcal{L}^2(\mathbb{R}^2)} = 2(t - K(p, p_0)).$$

This means that the kernel K can be thought as a measure of correlation between $p, p_0 \in \mathcal{G}$: $K(p, p_0)$ increases as they get "closer" (according to the distance d).

Now we want to add details on how the filters interact with each other in determining the geometry of the space. We need to define, around each $p_0 \in \mathcal{G}$, a *local patch* $\mathcal{P}(p_0) \subseteq \mathcal{G}$ and to restrict the definition of d to these elements. Then, the focus will be to see if it will be possible to "glue" all these distances together to obtain a global distance on the feature space $\mathcal{G}$.

Definition 2.16 *$\forall\, p, p_0 \in \mathcal{G}$, if $\exists\, \{q_j\}_{j=1,...,n}$ a sequence such that $q_0 = p_0, q_n = p$ and $q_j \in \mathcal{P}(q_{j-1})\, \forall\, j = 1, ..., n$, then we can define:*

$$\tilde{d}(p, p_0) : \mathcal{G} \times \mathcal{G} \to \mathbb{R},$$

$$\tilde{d}(p, p_0) := \inf\{\sum_{j=1}^{n} d(q_{j-1}, q_j) \mid n \in \mathbb{N}, q_0 = p_0, q_n = p \text{ and } q_j \in \mathcal{P}(q_{j-1}) \forall j\}$$

Otherwise, $\tilde{d}(p, p_0) := +\infty$.

Notice that, in general, the existence of such a sequence $\{q_j\}_{j=1,...,n}$ it is not guaranteed $\forall\, p, p_0 \in \mathcal{G}$, but this case could be not considered because this would mean having isolated points or regions of $\mathcal{G}$, corresponding to neurons whose activations are independent.

Proposition 2.1 *Given a set $\mathcal{G}$, if we can define a patch $\mathcal{P}(p_0) \subseteq \mathcal{G}$ such that $\forall\, p_0 \in \mathcal{G}$, $\exists\, \epsilon > 0$* s.t. *$B_\epsilon(p_0) := \{p \in \mathcal{G} \mid d(p, p_0) < \epsilon\} \subseteq \mathcal{P}(p_0)$ then the function $\tilde{d}(p, p_0) : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ already defined satisfies:*

(i) $\tilde{d}(p,q) \geq 0, \forall\, p,q \in \mathcal{G}$

(ii) $\tilde{d}(p,q) \leq \tilde{d}(p,s) + \tilde{d}(s,q), \forall\, p,s,q \in \mathcal{G}$

(iii) $\forall\, p,q \in \mathcal{G},\ \ \tilde{d}(p,q) = 0 \Leftrightarrow p = q$

Proof. Firstly, we want to prove that the function $\tilde{d}$ is well-defined, which means verifying that local distance functions coincide on overlapping patches, but this occurs by construction, since $d(p,p_0) := \|\psi_p - \psi_{p_0}\|_{\mathcal{L}^2(\mathbb{R}^2)}$.

Now we can prove the other properties:

(i) : $\tilde{d}$ is obviously ≥ 0 (since it is defined as a sum of positive elements).

(ii) :

$$\begin{aligned}
\tilde{d}(p,s) + \tilde{d}(s,q) &= \inf\left\{ \sum_{j=1}^{n} \tilde{d}(q_{j-1},q_j) \;\middle|\; n \in \mathbb{N},\, q_0 = s,\, q_n = p,\, q_j \in \mathcal{P}(q_{j-1})\, \forall j \right\} \\
&\quad + \inf\left\{ \sum_{j=1}^{n} \tilde{d}(q_{j-1},q_j) \;\middle|\; n \in \mathbb{N},\, q_0 = q,\, q_n = s,\, q_j \in \mathcal{P}(q_{j-1})\, \forall j \right\} \\
&= \inf\left\{ \sum_{j=1}^{n} \tilde{d}(q_{j-1},q_j) \;\middle|\; n \in \mathbb{N},\, q_0 = q,\, q_n = p,\, q_j \in \mathcal{P}(q_{j-1})\, \forall j,\, \exists\, j \text{ s.t. } q_j = s \right\} \\
&\geq \inf\left\{ \sum_{j=1}^{n} \tilde{d}(q_{j-1},q_j) \;\middle|\; n \in \mathbb{N},\, q_0 = p_0,\, q_n = p,\, q_j \in \mathcal{P}(q_{j-1})\, \forall j \right\} \\
&= \tilde{d}(p,q)
\end{aligned}$$

(iii) : We suppose $p \neq p_0$, for hypothesis $\exists\, \epsilon > 0$ s.t. $B_\epsilon(p_0) \subseteq \mathcal{P}(p_0)$, so:

- if $p \notin \mathcal{P}(p_0) \Rightarrow p \notin B_\epsilon(p_0) \Rightarrow \tilde{d}(p,p_0) \neq 0$
- if $p \in \mathcal{P}(p_0) \Rightarrow \tilde{d}(p,p_0) \neq 0$ because d is a distance on $\mathcal{P}(p_0)$.

□

Note that the condition $q_j \in \mathcal{P}(q_{j-1})$ does not mean that $q_{j-1} \in \mathcal{P}(q_j)$. In facts, in general, the definition of $\tilde{d}$ makes it a *quasimetric* distance (similar concept to asymmetric). This means that getting from p to p_0 may be "harder" than following the opposite path (we are saying that $\tilde{d}(p, p_0) \geq \tilde{d}(p_0, p)$), for example we can see it like walking between two points on a mountain: in this vision p_0 in uphill with respect to p. In this particular case, we are defining a distance which should model the lateral connectivity in V1, and we know that horizontal connections are reciprocal, so it is logical to model it through a symmetric distance. For this reason we are requiring that: $p \in \mathcal{P}(q) \Leftrightarrow q \in \mathcal{P}(p)$.

In other words, we can think at the kernel distance we have defined as a local object by restricting it to suitable patches around every $p \in \mathcal{G}$. With the Proposition already proved, we are stating that, under reasonable conditions on the choice of the patches, we obtain a well-defined global distance on $\mathcal{G}$.

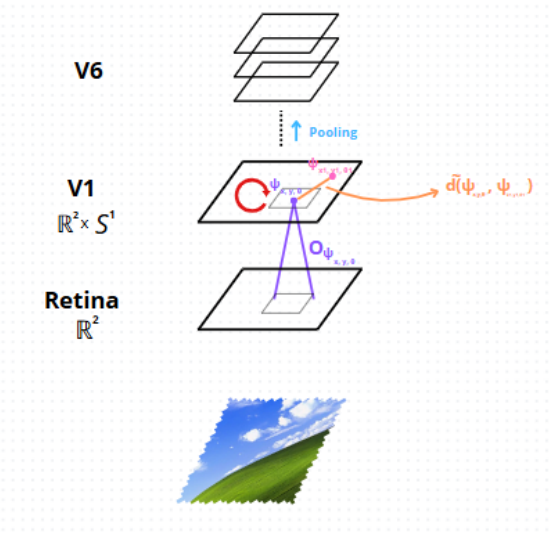

Figure 2.1: Schematic representation of the cortical metric

2.3 Gabor filters

As previously discussed, simple cells in the visual cortex respond strongly to lines or edges with specific orientations. To model their receptive profiles (RPs) while pre-

serving this orientation sensitivity, *Gabor filters* are commonly used [3]. These linear filters are effective for analyzing specific frequency content and directional features within an image.

A bank of Gabor filters $\{\psi_{x,y,\theta}\}_{x,y,\theta}$ is indexed by $\mathcal{G} = \mathbb{R}^2 \times S^1$, where $(x, y) \in \mathbb{R}^2$ give information on the position of the center of the filter, and $\theta \in S^1$ express its preferred orientation.

In particular, $\mathbb{R}^2 \times S^1$ is a group representing the translations and rotations of a given $p = (x, y, \theta)$: each $A_{x,y,\theta} \in \mathbb{R}^2 \times S^1$ can be represented like:

$$A_{x,y,\theta} \begin{pmatrix} u \\ v \end{pmatrix} = T_{x,y} \circ R_\theta \begin{pmatrix} u \\ v \end{pmatrix} = R_\theta \begin{pmatrix} u + x \\ v + y \end{pmatrix},$$

where $R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$.

We can easily define the inverse of a generic $A_{x,y,\theta} = T_{x,y} \circ R_\theta$ as:

$$A^{-1}_{x,y,\theta} \begin{pmatrix} u \\ v \end{pmatrix} = T^{-1}_{x,y} \circ R^{-1}_\theta \begin{pmatrix} u \\ v \end{pmatrix} = T_{-x,-y} \circ R_{-\theta} \begin{pmatrix} u \\ v \end{pmatrix}.$$

The composition law is:

$$A_{x_1,y_1,\theta_1} \circ A_{x_2,y_2,\theta_2} \begin{pmatrix} u \\ v \end{pmatrix} = A_{x_1,y_1,\theta_1} \left[\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} + R_{\theta_2} \begin{pmatrix} u \\ v \end{pmatrix} \right]$$

$$= \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + R_{\theta_1} \left[\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} + R_{\theta_2} \begin{pmatrix} u \\ v \end{pmatrix} \right]$$

$$= \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + R_{\theta_1} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} + R_{\theta_1} R_{\theta_2} \begin{pmatrix} u \\ v \end{pmatrix}$$

$$= \left(\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + R_{\theta_1} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \right) + R_{\theta_1+\theta_2} \begin{pmatrix} u \\ v \end{pmatrix}$$

$$= A_{x_3,y_3,\theta_3},$$

$$\text{where} \quad \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + R_{\theta_1} \left(\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \right),$$

$$\text{and} \quad \theta_3 = \theta_1 + \theta_2.$$

So each bank of Gabor filters $\{\psi_{x,y,\theta}\}_{x,y,\theta}$, is obtained from a mother filter

$$\psi_{0,0,0}(u,v) = \exp\left(\frac{2\pi i u}{\lambda}\right) \exp\left(-\frac{u^2+v^2}{2\sigma^2}\right)$$

as:

$$\psi_{x,y,\theta}(u,v) = \psi_{0,0,0}\left(T^{-1}_{(x,y)} R^{-1}_{\theta}(u,v)\right).$$

In other words, each filter $\psi_{x,y,\theta}(u,v) = \psi_{0,0,\theta}((u,v)-(x,y))$.

According to the notations used for describing general filters, the RP $\psi_{x,y,\theta}$ acts on the stimulus I (taken in V1) for convolution:

$$O_{x,y,\theta}(I) = \int I(u,v)\, \psi_{x,y,\theta}(u,v)\, du\, dv = \int I(u,v)\, \psi_{\theta}((u,v)-(x,y))\, du\, dv = (I * \psi)(u,v).$$

The case of Gabor filters is effective because, by defining $\mathcal{G} = \mathbb{R}^2 \times S^1$, we explicitly frame the problem within a group structure rather than just a metric space. To get an idea, on the retina $\mathcal{G} = \mathbb{R}^2$ and we know how to define a distance (and consequently a convolution) on the group $\mathbb{R}^2$; in V1, where $\mathcal{G} = \mathbb{R}^2 \times S^1$, we have demonstrated that this structure forms a group. We now show that the kernel associated with the distance d defined above exhibits certain invariances on $\mathbb{R}^2 \times S^1$.

Given $\psi_p, \psi_{p_0} \in \{\psi_{x,y,\theta}\}_{x,y,\theta}$, where $p = (x,y,\theta), p_0 = (x_0,y_0,\theta_0)$ and $p, p_0 \in \mathcal{G} = \mathbb{R}^2 \times S^1$, expressing $\psi_p = \psi_{1p} + i\psi_{2p}, \psi_{p_0} = \psi_{1p_0} + i\psi_{2p_0}$ we can explicitly compute

the kernel (using a change of variable):

$$\begin{aligned}
K(p,p_0) &= \mathrm{Re}\,\langle \psi_p, \psi_{p_0}\rangle_{\mathcal{L}^2(\mathbb{R}^2)} \\
&= \int_{\mathbb{R}^2} \psi_{1p}(u,v)\,\psi_{1p_0}(u,v)\,du\,dv \\
&= \int_{\mathbb{R}^2} \psi_{(0,0,0)}\left(T^{-1}_{(x,y)}R^{-1}_{\theta}(u,v)\right)\,\psi_{(0,0,0)}\left(T^{-1}_{(x_0,y_0)}R^{-1}_{\theta_0}(u,v)\right)\,du\,dv \\
&= \int_{\mathbb{R}^2} \psi_{(0,0,0)}\left(T^{-1}_{(x,y)}R^{-1}_{\theta}R_{\theta_0}T_{(x_0,y_0)}(s,t)\right)\,\psi_{(0,0,0)}(s,t)\,ds\,dt \\
&= \mathrm{Re}\,\langle \psi_{p^{-1}p_0}, \psi_e\rangle_{\mathcal{L}^2(\mathbb{R}^2)} = K(p^{-1}p_0, e).
\end{aligned}$$

In this way, it is possible to estimate the output similarly to the method in Section 1.2: given $O_{p_0}(I)$ the output (in R) of a $p_0 \in \mathcal{G}$

$$O'(O_{p_0}) := \int K(p,p_0)O_{p_0}\,dp_0 = \int K(p^{-1}p_0,e)O_{p_0}\,dp_0.$$

Now that we understand how the kernel K in $\mathbb{R}^2 \times S^1$ operates between two points in V1, we can extend this approach to the next layer by applying filters according to the group law of $\mathbb{R}^2 \times S^1$. The construction on subsequent layers then follows as an iterative application of this process.

Chapter 3

Bridging Biological and Artificial Neural Network

In this final chapter, we will focus on the distinctions and similarities between convolutional neural networks (CNNs) and the biological structures discussed in Chapter 1. Specifically, our aim is to report on results which introduce improvements and optimizations in CNNs by applying the properties of neurons in the visual system. We will understand how to bridge the gap between artificial intelligence and biological processes, however, there will still be differences that need to be addressed.
In particular, we will exhibit the similarities between the structure defined in Section 2 and neural networks, showing how CNN operate like filters.

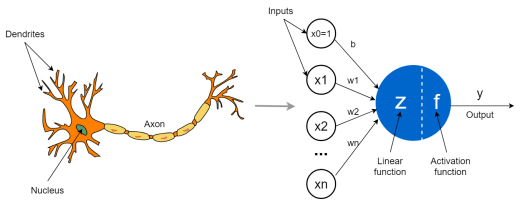

Figure 3.1: Parallel between a neuron and a neural network inspired by it

3.1 General background

In this section, we aim to introduce essential notions associated with deep neural networks (DNNs), with a particular focus on convolutional neural networks (CNNs). Neural networks (NNs) are computational models inspired by biological neural systems, designed to process information in a way that mimics the human brain. They are widely used for tasks such as predicting information, recognizing patterns, and making decisions, leveraging their ability to learn from data. Artificial neural networks consist of graphs where the vertices, called *nodes* (or neurons), are connected by edges representing the synaptic links. These nodes can be grouped into *layers*, which are hierarchically interconnected. Generally, a NN is composed by an input layer, some hidden layers and an output layer [5]. During a phase called *training*, each node receives input data from connected nodes, processes it by applying *weights* and the *activation function* (a non-linear function) and then transmits the output to the next connected node; this process is called *forward*.

3.2 DNNs

We examine a specific type of NN, constitute of multiple hidden layers: *deep neural networks* (DNNs). The simpler DNN structure is *feedforward*, where nodes in each layer pass information only to the ones in the next layer. The network is called *fully connected* if all the nodes of a layer are connected to all the nodes of the next one.
Now, we give a description of how DNNs are structured and how do they work.

A feedforward DNN defines an operator F, which is learned through a minimization process to approximate a given operator:

$$Z : H_0 \to H_L.$$

The function F is constructed as a composition of mappings, $F_0 \circ \cdots \circ F_L$, where each

map $F_l : H_l \to H_{l+1}$ acts between appropriate functional spaces, representing the layers of the network.

The operator F takes an initial function $h_0 : \mathcal{G}_0 \to \mathbb{R}$ (first layer) and generates an output $h_L : \mathcal{G}_L \to \mathbb{R}$ (last layer). For each $l \in \{0, ..., L\}$, the transformation $F_l : H_l \to H_{l+1}$ defines the operations occurring between consecutive layers, specifically between the l-th and the $(l+1)$-th layers.

In particular, the *activation* of the $(l+1)$-th layer, $h_{l+1} : \mathcal{G}_{l+1} \to \mathbb{R}$, is computed from the activation of the previous layer. The transformation F_l first applies a linear operation $A_l : H_l \to H_{l+1}$ to the input $h_l \in H_l$, followed by a nonlinear *activation function* $s_{l+1} : H_{l+1} \to H_{l+1}$. Explicitly:

$$h_{l+1} := F_l(h_l) = s_{l+1}(A_l h_l + b_l),$$

where $b_l \in H_{l+1}$ is an additional term known as *bias*.

We outline just two activation functions from the most commonly used:

- the *sigmoid* function: $s(z) = \frac{1}{1+e^{-z}}$, where $s : \mathbb{R} \to (0, 1)$;
- the *Rectified Linear Unit* (ReLU) function: $s(z) = \max(0, z)$, where $s : \mathbb{R} \to [0, \infty)$.

In general, the $\mathcal{G}_l$ are considered as discrete sets and we usually write $h_l = \{h_l(i)\}_{i \in \mathcal{G}_l}$. In this case, at each layer the linear operator A_l can be represented as a matrix whose elements are the weights $W := \{w_l(j, i)\}_{j \in \mathcal{G}_{l+1}, i \in \mathcal{G}_l}$, while the bias terms are vectors $b_l = \{b_l(j)\}_{j \in \mathcal{G}_{l+1}}$.

In a supervised learning framework, we are given a *dataset*, which is a finite subset $D \subseteq H_0$ along with corresponding target values defined by the operator Z. This dataset consists of input-output pairs that guide the learning process.

There are two main tasks that neural networks can perform:

- *regression*, where, from a set of inputs, the target is to learn a function that best fits the relationship between inputs and outputs
- *classification*, where the target is to learn a function that assigns each input to one of the possible classes.

To optimize the functionals F_l and ensure that Faccurately approximates the target operator Z, we define a *loss function* L, which quantifies the difference between the network's output or prediction h_L and the expected result g_L. The training process aims to adjust the weights W and biases b to minimize this loss.

Common choices for $L(h_L, g_L)$ are:

- *mean-squared error*, typically used for regression problems, is defined as:

$$L_{MSE} = \frac{1}{|\mathcal{G}_L|} \sum_{i \in \mathcal{G}_L} (h_L(i) - g_L(i))^2$$

- *cross-entropy loss*, used for classification problems, quantifies the difference between two probability distribution (the distribution from the model h_L and the true distribution g_L), is defined as:

$$L_{CE} = -\sum_{i \in \mathcal{G}_L} g_L(i) ln(h_L(i))$$

When training a DNN, an effective method to minimize the loss L is *gradient descent* (GD), which works iteratively adjusting the parameters. In general, the gradient of L with respect to the weights is a vector $\frac{\partial L}{\partial w_l(j,i)}$ that points in the direction of the steepest increase of L, so it gives information on how much the loss would change if we change the weights. In GD, in order to minimize the loss, we want to move in the opposite direction of the gradient. In each *epoch* (iteration), the weights are updated using:

$$w_l(j,i) \leftarrow w_l(j,i) - \alpha \frac{\partial L}{\partial w_l(j,i)},$$

where α is the *learning rate*, a small positive value that controls the step size of the update, so how big a step is taken in the direction of the GD. If α is too big, it is possible to miss the minimum, if is too small, it could take too long to converge or get stuck at the wrong minimum.
The same updates are applied to the biases:

$$b_l(j) \leftarrow b_l(j) - \alpha \frac{\partial L}{\partial b_l(j)}.$$

These updates are applied iteratively over many epochs until the L converges to a minimum, corresponding to an optimal solution for the given task.

Since the NN could have many layers, computing the gradients could be extremely slow and inefficient, for this reason it is common to use an algorithm, called *backpropagation*. We can summarize how it works like:

- it computes the error at the output layer and then calculate the derivative of the loss with respect to the output
- it propagates the error backward through the network, to calculate how much each weight in the previous layers contributed to the overall error
- it updates the weights with GD

Actually, GD is not the most effective method that can be used, because the gradient is calculated at each update step using the entire dataset. Obviously, for large datasets and complex NNs, this could become computationally expensive and slow. *Stochastic Gradient Descent* (SGD), on the other hand, introduces a practical solution by approximating the gradient using only a single randomly selected sample or a small batch of samples at each iteration.

A priori, the dataset D is split into three parts, each containing a subset of samples:

- D_{train}, the *training set*, used to determine A and b in the optimization process.

- $D_{\text{validation}}$, the *validation set*, used to validate the model's performance during training
- D_{test}, the *test set*, containing data that was not used in training and is reserved for evaluating the model's performance on unseen examples.

In this way, $D = D_{\text{train}} \cup D_{\text{validation}} \cup D_{\text{test}}$.

In particular, this splitting procedure is necessary to avoid *overfitting*, a common issue where the model memorizes training data instead of learning general patterns. When overfitting occurs, the function F performs well on D_{train} but generalizes poorly to new data in D_{test}. In deep learning, optimization landscapes are often highly non-convex, meaning they contain multiple local minima, making training more challenging. Even simple networks can exhibit an exponentially large number of such minima.

To mitigate overfitting, a common technique called early stopping is applied. The idea is to stop training when the validation loss stops improving, ensuring that the model does not continue learning noise from the training data.

3.3 CNNs

Image data

From this point forward, we will focus on a particular case of input data: image datasets. According to the notions used above, an image is a structure data that can be represented as a function, where each point corresponds to a pixel's intensity or color information. In a discrete setting, a grayscale image can be described as:

$$I : \{1, ..., h\} \times \{1, ..., w\} \to \mathbb{R},$$

where h is the height of the input image, w is its width, and $I(x, y)$ is an intensity value representing the brightness level (ranging from 0 to 255, from black to white).

In general, if we are considering n_c color channels, the domain of I is

$$\{1, ..., h\} \times \{1, ..., w\} \times \{1, ..., n_c\},$$

for example for RGB images:

$$I : \{1, ..., h\} \times \{1, ..., w\} \times \{1, 2, 3\} \to \mathbb{R}.$$

Images are frequently represented as matrices, to enable efficient storage and processing. Thus a grayscale image can be seen as a matrix $I \in \mathbb{R}^{h \times w}$, where each element $I_{i,j}$ holds the intensity value at pixel location (i, j).

We state also the definition of I in the continuous setting:

$$I : [a_1, a_2] \times [b_1, b_2] \to \mathbb{R}.$$

Sometimes these complex structures are ignored and images are treated as vectors of numbers by flattening the matrices, not respecting the spatial relation between pixels. For instance, an image $I \in \mathbb{R}^{h \times w}$ could be converted into a vector $I \in \mathbb{R}^{(h+w) \times 1}$ by concatenating the rows and columns, leading to a structure that does not retain any spatial information. This method is convenient for fully connected networks, because they are invariant to the order of features, thus they can give similar outcomes regardless if the spatial arrangement is maintained or not.
Although, this procedure is very limiting, because it ignores the local correlation between adjacent pixels, which often share similar intensity values or colors. For example, in a natural image, neighboring pixels could represent edges, textures, ... and treating them as independent feature lead to lose the ability to capture these patterns.

In general, this type of data is essential for image classification, the primary task of interest in our discussion.

Convolutional Neural Networks

The properties that we have observed above are exploited fully by a particular class of DNNs: *Convolutional Neural Newtworks* (CNNs). These networks are designed to capture spatial hierarchies and patterns and are richly inspired by biological processes as we will note later. The peculiarity of CNNs is that there are some layers that are not fully connected and the application of local convolutional filters that slide over the spatial domain. This design allows the network to maintain the relationships among nearby pixels, but also enforces translation invariance, enabling the shared weights of the filters to recognize features regardless of their position in the image.

In detail, given a set of filters $\{\psi_k^1\}_{k=1,...,n_1}$ with a localized support, and an input image $I(u, v, c)$ where u and v are spatial coordinates and $c \in \{1, ..., n_1\}$ indexes the color channels, the first *convolutional layer* produces an output feature map by computing the convolution:

$$\begin{aligned} A_0 I(u, v, k) &:= \psi_k^0 * I(u, v) \\ &= \int_{\mathcal{G}_0} \psi_k^1(u', v', c) I(u - u', v - v', c)\, du'\, dv'\, dc \\ &= \sum_{c \in \{1,2,3\}} \int_{b_1}^{b_2} \int_{a_1}^{a_2} \psi_k^1(u', v', c) I(u - u', v - v', c)\, du'\, dv' \end{aligned},$$

where A_0 is a linear operator from the first layer to the second one, and

$$\mathcal{G}_0 := \{1, ..., h\} \times \{1, ..., w\} \times \{1, ..., n_0\}.$$

Subsequent layers perform similar convolutions on the outputs of the previous layers:

$$\forall\, l \in \{0, ..., L - 1\},\ A_l h_l(u, v, k) := \psi_k^{l+1} * h_l(u, v)$$

where h_l is the feature map from the l-th layer and $\{\psi_k^{l+1}\}_{k=1,...,n_{l+1}}$ is the bank of filters associated to the $(l + 1)$-th layer.

In the discrete setting:

$$h_{l+1}(i,j,k) = s_{l+1} \sum_{i',j',k'} \psi_k^{l+1}(i',j',k') \cdot h_l(i-i', j-j', k') + b_l(k).$$

The activation of each convolutional layer is obtained by applying a non-linear function s_{l+1} to the result of the convolution and adding a bias term b_{l+1}:

$$h_{l+1}(u,v,k) := s_{l+1}(A_l h_l(u,v,k) + b_{l+1}(k))$$

The final layers of the network are usually not convolutional but fully connected, meaning every neuron in the last feature map layer h_L connects to every neuron in the subsequent layers. This is achieved by flattening h_L into a vector and applying a linear transformation, followed by a non-linear activation (such as ReLU). The last layer typically produces outputs of class probabilities using the softmax function.

Summarizing, convolutional layers use filters that scan over the input image. Each of them is smaller than the image and is applied repeatedly across spatial dimensions. This process involves:

- local connectivity: each filter connects only to a small region of the input (called receptive field, like the biological features we have analyzed in Section 1), reducing the number of parameters;
- weight sharing: each filter has a set of weights that are shared across different locations in the image, and the same weights are applied to all corresponding regions.

In this way, CNNs are more efficient than standard DNNs for image classification, because require fewer parameters.

Frequently, to simplify the output between convolutional layers, a *pooling* operation can be added, similar to the process described in Chapter 1. It is designed to reduce dimensionality, to retain key features and to introduce invariance, i.e. providing a degree of invariance to slight translations or distortions in the input, helping the model to generalize better.

The most commonly used pooling approaches are:

- *max pooling*, which takes the maximum value in each region
- *average pooling*, which takes the average of all values in each region.

For example, given a 4×4 input feature map: $\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$ by applying a 2×2 max pooling filter we obtain $\begin{bmatrix} 4 \end{bmatrix}$, while applying an average pooling filter of the same dimension we obtain $\begin{bmatrix} 2.5 \end{bmatrix}$.

Pooling layers have a biologically inspired role in CNNs. In the visual system, neurons that process visual information have small RFs, which are likely to grow towards cortical layers. In a similar way, subsampling the feature map in a CNN broadens the area each next filter covers. This sampling lets higher layers in the NN capture wider patterns from the image, like the brain's higher visual areas respond to larger structures.

3.4 Recurrent CNNs and Kernel CNNs

Since now, all the NN that we have examined are build respecting the hierarchical transmission between layers, but they lack the lateral connectivity structures that we have outlined when discussing the biological spreading of visual information in the primary visual cortex.

Recurrent CNNs (RecCNNs) are a modified version of CNNs where lateral connections are added. This means that these NN include both feed-forward connections between layers and recurrent connections within the same layer.

Introducing a time parameter t, the activation of the l-th layer at t can be expressed like:

$$h_l^t = s_l(\varphi^l * h_l^t + \psi^l * h_{l-1}^t + b_l),\ \forall\, t, l > 0,$$

where h_{l-1}^t is the output of the previous layer at the same t and $\{\varphi_k^l\}_{k=1,\dots,n_l}$ is a set on convolutional filters defining the lateral connections.

This combination of lateral and feed-forward connections is particularly used for tasks that require both spatial and temporal processing, such as video analysis, sequential image processing or, in general, situations where information across multiple frames is needed.

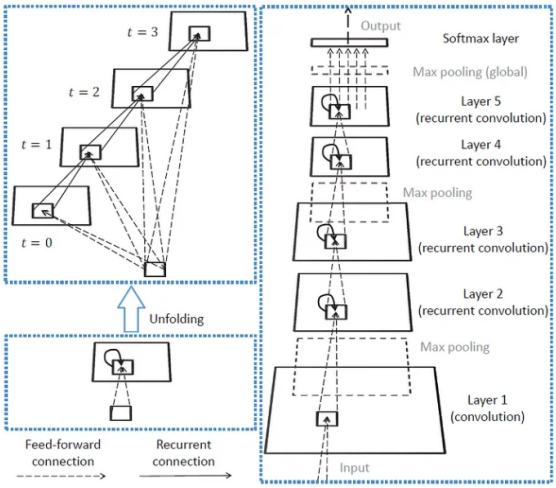

Figure 3.2: Overview of a RecCNN

It is important to note that, with this construction, lateral connections are added separately from the feedforward ones. This leads to an increase in the number of parameters and to the fact that the two types of connections are completely independent from one another, seeming unrelated.

To address this issue, a new type of CNN has been introduced: *Kernel CNNs* (KerCNNs), which can take care of this important relationship between lateral and feedforward connections. Essentially, the goal is to reinforce our model with the biologically inspired concepts that we discussed in Chapter 2, thereby inducing that metric structure

on the layers of the CNNs.

In summary, this approach involves adapting the loss function by applying a regularization term and then using the kernel $K(p, p_0)$ to introduce lateral connections into the network. In such manner, the recurrent kernels are obtained as functions of the feed-forward connections, eliminating the need for additional parameters (see [2] for more details on their structure).

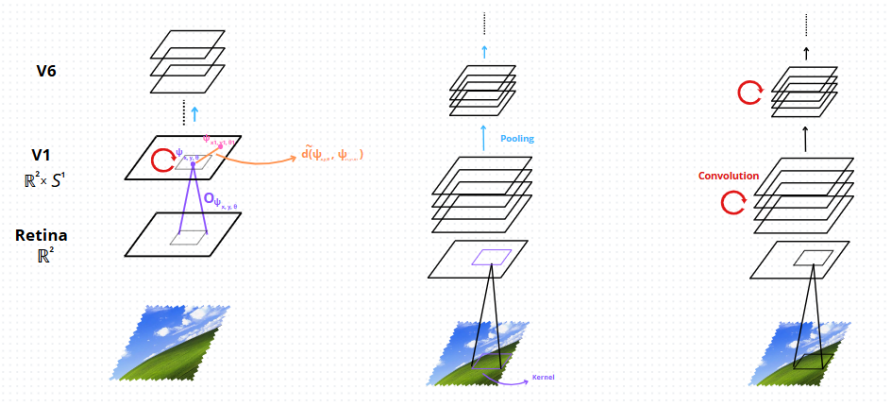

Figure 3.3: Parallel between the biological construction of Chapter 2 and KerCNNs

Conclusions

In this thesis, we presented a metric model to interpret the connections between neurons in the primary visual cortex (V1). We explored how neurons interact with each other, starting from their hierarchical organization, where neurons in one layer transmit information to those in the next. We then examined lateral (or horizontal) connections, which occur between neurons within the same layer and play a fundamental role in processing visual information.

To formalize these interactions, we introduced a distance function and a kernel that directly depend on the spatial position of neurons, explicitly expressed as a scalar product of two RPs. These functions allowed us to analyze the degree of correlation between neurons and model how they influence each other based on their relative positions. In this way, we were able to effectively describe how visual information propagates and is transmitted in V1.

In addition, we draw a parallel between the biological brain architecture and the architecture of artificial neural networks. In particular, we have described how these concepts of correlation and spatial relationships among neurons could be integrated to reinforce the CNNs structures.

Bibliography

[1] D. H. Hubel and T. N. Wiesel, *Receptive fields, binocular interaction and functional architecture in the cat visual cortex*, J. Physiol. (London), vol. 160, pp. 106â154, 1962.

[2] N. Montobbio, *A Metric Model of the Visual Cortex*, PhD thesis, Alma Mater Studiorum University of Bologna, in collaboration with Sorbonne University, 2019.

[3] D. Purves, G. J. Augustine, D. Fitzpatrick, W. C. Hall, A.-S. Lamantia, L. E. White, *Neuroscienze*, 4th Italian edition, Zanichelli, 2013.

[4] M. F. Bear, B. W. Connors, M. A. Paradiso, *Neuroscience: Exploring the Brain*, 4th ed., Wolters Kluwer, 2020.

[5] A. Zhang, Z. C. Lipton, M. Li, A. J. Smola, *Dive into Deep Learning*, Available online: `https://d2l.ai`, 2021.