# Structural Observations on Neural Networks for Hierarchically Derived Reproducing Kernels

Maia Fraser

November 7, 2011

# Contents

# Chapter 1

# Introduction

Artificial neural networks have occupied an important place in the field of artificial intelligence (AI) since its early days. As with many tools developed in AI, the motivation has been two-fold: both to *achieve capabilities* observed in living organisms by emulating (abstracting) the seemingly responsible underlying mechanisms and to *understand* or *validate* the role of these mechanisms by observing the success of the tool.

An artificial neural network (NN) – an abstraction of a network of biological neurons – is essentially a directed graph, where each node is a computational gate computing a particular function for each of its outgoing arcs (outputs), based on its incoming arcs (inputs). Additionally, however, there are assumed to be some arcs leading into the digraph (representing global input to the NN) and other arcs leading out of the digraph (representing global output of the NN). In general, if there are directed cycles in the graph, the order and tact of computation can be complex to specify. Such NN's are said to be *recurrent*. In this thesis, however, we restrict ourselves to *feedforward* NN's, where no directed cycles are present.

In these NN's it is understood that the global input values are fed into the feedforward NN at the start, each computational gate waits until all of its inputs have been computed before itself computing, and so on, until finally the global output is obtained. We will allow the values carried by arcs to be real numbers. We will, moreover, assume that each node computes a single output value and that there is only *one* global output value (i.e. a single top node). Such a feedforward NN may therefore equivalently be viewed as an *encoding* of a mapping from $\mathbb{R}^n$ to $\mathbb{R}$, where $n$ is the number of global input arcs, and the mapping is specified *in the form of a circuit* (as just described). In other contexts (e.g. the multiclass perceptron) multiple global outputs may be allowed; such feedforward NN's are equivalent to maps from $\mathbb{R}^n$ to $\mathbb{R}^m$.

More precisely, this thesis will be concerned with feedforward NN's used to compute so-called *reproducing kernels* (which are real-valued). Chapter 2 gives a brief overview of

the development of such *NN models*. By NN model is meant a class of NN's, e.g. those with gates and/or circuits of a prescribed type.

**Example 1.0.1.** [$\Sigma$ networks] (see [9]). Given a (Borel) measurable function $G : \mathbb{R} \to \mathbb{R}$ and a positive integer $r$, let $\Sigma^r(G)$ denote the class of all maps $f : \mathbb{R}^r \to \mathbb{R}$, defined (and computed) as

$$f(x) = \sum_{i=0}^{p} \alpha_i G(A_i(x)), \forall x \in \mathbb{R}^r,$$

where $p \in \mathbb{N}$ and for each $i$, $\alpha_i \in \mathbb{R}$ and $A_i : \mathbb{R}^r \to \mathbb{R}$ is an affine function. By viewing each of the affine functions $A_i$ and the summation procedure as single computational operations, we are also implicitly specifying above *how* each function in the class $\Sigma^r(G)$ is computed. This class may therefore also be seen as an NN model, i.e. class of NN's: the associated networks are laid out with $p$ gates in the lower layer, each of which has the same $r$ inputs (the coordinates of $x$), and which compute $A_1(x), \ldots, A_p(x)$ respectively, plus a single gate at the top layer which *pools* the outputs of the lower layer to produce the global output $f(x)$.

In this example, $\Sigma^r(G)$ represents an NN model, where the circuit topology is fixed and the types of gates is restricted to affine transformations on the lower layer, and summation on the top. Some NN models do not fix the circuit topology absolutely, though they may restrict it. In particular, in some models – for example those discussed in Chapter 2 – the depth of the circuit may be unbounded.

Because of the strict sequential order of computation (determined by distance from the global input), it is common to speak of *layers* in feedforward NN's; the bottom being the layer where global input arrives and the top being a single node computing the global output (in our case). In general all layers except for the top and bottom are referred to as *hidden layers*. Networks with few hidden layers (zero or one, typically) are said to be *shallow* and those with more layers, *deep(er)*.

The design issue of **shallow vs. deep** NN's has been considered in various cases, going back as far as 1969, when Minsky and Papert showed, in their book *Percetrons* [13], that a perceptron without hidden layers (the so-called *single layer perceptron*) cannot compute the XOR function. Several recent papers of Bengio and LeCun, as well as Hinton and some of his students, make the claim that deep NN's are superior to shallow NN's in a fundamental way and that shallow NN's are in this sense inadequate. This claim is sometimes promoted by appealing to Håstad's seminal result in circuit complexity which states that depth-$k$ circuits with only AND, OR or NOT gates must be of exponential size in order to compute the parity function [8]. These are not the gates used in typical NN's. And there is in fact no existing result in circuit complexity for such general gates and very little work on real circuits at all (what there is usually assumes global input/ouput that is Boolean, allowing only intermediate variables to be possibly real-valued). In order to

prove a meaningful result like Haastad's for the NN setting, one would first need to clearly define the class of real circuits to be considered (specifying the types of computational gates and their computational power) and *also* define the function to be computed. In this step, the definition of the function must be independent of the design of the circuit. Herein lies the central weakness in the mentioned claims: the function to be computed is defined by an NN and one wishes to consider only NN's very much like this one - so much like it, in fact, that it seems plausible it is the only one. Such a result would not be a fair analog of Haastad's. Moreover, Håstad's result is *asymptotic* while biological circuits have size bounded above by many other natural factors. This is not to say, there is no justification for deep NN's. Indeed there are likely compelling reasons for nature to form deep circuits, since they are so prevalent. This thesis seeks to understand the role of depth in some NN's, by showing - at least for a particular class of NN's - what it does not accomplish.

In particular, we show that for the very general model of NN considered here – an extension of the CBCL model – shallow networks produce the **same** class of mappings as deep networks (see Theorem 4.6.1).

Interestingly, our argument against a Håstad-like result for that model fails if one strongly restricts the type of transformations from one level to the next of the hierarchy (this will be defined in Chapter 3). Such a restriction is presumably at the core of the pro-deep claims mentioned above and illustrates how sensitive such mathematical results may be to changes in hypotheses.

One may in fact regard this thesis as a step towards enunciating and possibly proving the superiority of deep circuits within a certain framework. Indeed, in this thesis we show how to reduce any deep circuit to a shallow one, while making explicit the transformation sets (between levels) which this would imply. If one can rule out such transformation sets then the performed collapsing would be impossible.

One crucial issue regarding NN's that has not been mentioned so far is that of choosing, i.e. *learning*, a particular instance of an NN model based on known input/output pairs that are in some sense typical of what the NN should produce. This is generally posed as an optimization problem, where one seeks the values of parameters that minimize a cost function (resp. maximize an objective function). More specifically, it is a statistical estimation problem. In any case, the optimization is then solved by a *training* algorithm. In nature, however, it would necessarily be handled by some form of *feedback* - growing neurons or losing connections in response to information fed back to the structure. Because the NN's we have discussed so far have all been *feedforward*, they can only be tuned in this way by imposing feedback from the outside; backpropagation is a widespread *supervised* method of this kind.

It is worth noting that depth is not usually posed as one of the parameters to learn; typically one learns only the parameters that determine the operations of the gates. But this then forces the designer to make a priori heuristic choices of layers and depth, rather

than letting the data determine them. Clearly in nature the entire design - including choice of layers - evolves by a natural process. The resulting development of layers may well be closely related to naturally occurring transformations and the quantities which they leave invariant. Iterative explanations seem most natural: for example, a two layer unit (which will become a building block in larger structures) may only form to detect specific quantities if these quatities exhibit a great deal of invariance to common transformations. Once such sub-units exist, their output may then exhibit invariances which lead to the next level and so on. On the other hand, sub-units which are never used by higher structures may be less likely to persist.

Both of these hypothetical development mechanisms involve a form of feedback about topology, but on a different time scale from the feedforward operation of the NN as a mapping. While still remaining within the category of feedforward NN's this feedback may be incorporated at least into the *training* of the NN (as with backpropagation): by first learning the layers in an unsupervised way, so as to determine the topology of the NN, and then within that restricted model using labeled data to learn the parameters of gates, by standard supervised methods. In the case of vision, one has fairly strong prior knowledge about what the layers and sub-units should be (based on experimental data), but in more general applications this is no longer the case and flexibility concerning the layers/transformations could be an interesting option to pursue.

# Chapter 2

# Neural Networks for Visual Detection

## 2.1 Early milestones

Commonly regarded as a first step in the development of artificial neural networks, the **Threshold Logic Unit** (TLU) was proposed by McCulloch and Pitts in 1943 [14]. It models a neuron as a computational gate: it takes $r$ inputs (i.e. an $r$-vector), computes a linear real-valued function (i.e. weighted sum) of the input and then follows this with a thresholding operation (originally a Heaviside function). The inputs and output are assumed to be Boolean. By combining these artificial neurons into networks, it was observed that one may accomplish Boolean AND and OR operations and thus compute any Boolean function.

A subsequent milestone was the **Perceptron**, proposed in 1957 by Rosenblatt [18]. This was essentially a network of TLU's but with the specification of a particular rule for learning the weights from a training data set, consisting of binary labeled vectors in $\mathbb{R}^r$. Novikov [15] (see also Rosenblatt[19]) proved that this training algorithm will in a finite number of steps find weights for the Perceptron such that it outputs the correct classification for all training data - if such weights exist (i.e. the data is *linearly separable*).

Soon after this result, however, in 1969, came Minsky and Papert's book on Perceptrons [13]. Among other things, it showed that a Perceptron without hidden layers cannot implement XOR, a result which is widely credited with dampening enthusiasm for NN's for years afterwards.

In 1989, however, Cybenko [4] proved that a multilayer perceptron with one hidden layer can in fact approximate *all* continuous, real-valued functions to any desired degree of accuracy. The problem with perceptrons however is their fully-connectedness. This means that training for a 32x32 grid, for example, requires learning over 1,000 parameters for

*each* neuron of the hidden layer. Also, invariance to motion – a key feature of biological visual systems – is not built into the perceptron design.

In 1980, Fukushima proposed the **Neocognitron**. This model is very similar to those we will consider in this thesis; it is inspired - as they are - specifically by the pathways of the visual system and addresses the two shortcomings of perceptrons just identified. Its computational gates are of two kinds: S-cells and C-cells. These were intended as models respectively of the simple and complex cells of the visual cortex, which had been identified by Hubel and Wiesel in 1959 [10]. They had observed that the simple cells evolved to detect specific features, such as edges, in restricted parts of the visual field while the complex cells pooled signals from the simple cells to provide position invariant detection, and also additional features such as direction of motion. In the Neocognitron the layers consist alternately of S- and C-cells - a hierarchical architecture originally proposed by Hubel and Wiesel themselves. Only the parameters of the S-cells are learned. The C-cells are always OR gates, whose input is S-cells from various parts of the visual field. These operations - of filtering followed by pooling - are repeated in hierarchical fashion until the entire visual field has been sensed and processed.

## 2.2 Recent models for visual selection/detection:

One of the key properties of the visual system, which Neocognitron sought to reproduce/explain was translation invariance in object recognition. More general transformation-invariance was also subsequently studied in the neuroscience community [16, 22].

Several computational NN models, aimed at addressing this more general transformation invariance began to appear around 2000. The proposed architectures are similar to Neocognitron but more flexible. They differ slightly but one common ingredient is the idea, introduced in [16], that transformation-invariance could be obtained by pooling over various transformed versions of an object. These models – which are the focus of this thesis – can largely be grouped into two schools.

- Poggio and Riesenhuber [17] proposed in 1999 an NN for visual detection in which S-cells and C-cells alternate, but the C-cells perform a max operation (instead of the fixed OR of Neocognitron); this was modified and refined in various papers (for example [20] from 2007) and then Smale et al proposed a mathematical framework which underlies these NN's in [21] and does not prescribe which type of pooling is done at which layers. We will loosely call such NN's *CBCL Networks*.

- Bengio and LeCun [11] proposed in 1998 an NN, building on the ideas of LeCun's earlier NN models [12] from the late 80's and leading to many variations (in architecture and training) described in subsequent papers. These are generally referred to as *Convolution Neural Networks*. They alternate linear pooling ("convolution" plus

a bias), and sub-sampling (or max) operations - but in each case these operations are followed by a non-linear *activation* function (e.g sigmoid) which "squashes" the output before passing it on as input to the next layer. The layers at the end (fully connected without max operations) form essentially a multi-layer Perceptron. As with classic multi-layer Perceptrons (see [3] for a discussion), the learning of these weights is equivalent to logistic regression.

Both of these NN models will be considered in more detail in the next Chapter.

Similar networks also appeared in the statistical literature around the same time [1, 2]. Amit [2] proposes a depth-limited NN similar to the above but differing in how learning is done. Learning is done separately at the different scales: local features are learned from local image patches, object models are learned from larger image patches.

Another related body of work is that on *Auto-encoders* (Hinton). The underlying network is essentially the same as what we have considered so far; the focus is on training methods for dealing with NN's with many hidden layers.

## 2.3 Focus of this thesis

We will not concern ourselves with learning/training algorithms for NN's in this thesis. Rather we seek to understand purely what can be *computed* by NN's of the above two models as a function of different design choices, e.g. depth.

NN's of both models produce **similarity measures**, so that one obtains an output $K(x, y)$, where $y$ is hard-coded and $x$ is the input. This value should represent how similar the images $x$ and $y$ are. This is sometimes used for classification into $n$ classes $c_1, \ldots, c_n$ if one has objects $y_1, \ldots, y_n$ that are known to be "typical" of the respective classes: the class $c_i$ which maximizes $K(x, y_i)$ over all $i$ is chosen. Ideally, to support such a procedure there should be a statistical interpretation which allows one to relate $K(x, y_i)$ to a probability that $x$ is in class $c_i$. This statistical aspect is not addressed in the work on CBCL or Convolution Networks, but receives explicit treatment in some statistical approaches [2]. We will not discuss it in this thesis, although a possible conclusion of the present work is that such statistical and learning considerations may be more suited than computability to explain the role of depth.

As mentioned, the output of the NN's we will discuss is typically of the form $K(x, y)$, where $x$ and $y$ are images. In fact, in both Convolution NN's and CBCL NN's this output has the structure of a **reproducing kernel** on the space of images. This observation was first made by Smale et al [21] with regards to CBCL networks. This thesis makes essential use of reproducing kernels to decompose NN's of the above two models.

### 2.3.1 Reproducing Kernels

A reproducing kernel is a generalization of an inner product to sets which are not necessarily vector spaces.

**Definition 2.3.1.** Given an arbitrary set $X$, a map $K : X \times X \to \mathbb{R}$ is said to be a (real-valued) **positive definite kernel** on $X$ if for any $n \in \mathbb{N}$, $x_1, \ldots, x_n \in X$ and $a_1, \ldots, a_n \in \mathbb{R}$,

$$\sum_{i,j=1}^{n} a_i a_j K(x_i, x_j) \geq 0. \tag{2.3.1}$$

If in addition the kernel is symmetric, i.e.

$$K(x, x') = K(x', x),\ \forall x,\ x' \in X,$$

then we call it a **reproducing kernel** on $X$. The condition expressed in (2.3.1) is known as **Mercer's condition**.

**Remark 2.3.2.** If $X$ is a topological space and $K$ is continuous then $K$ is said to be a *Mercer kernel*. The nomenclature is somewhat misleading: it is not the Mercer condition which sets a Mercer kernel apart from a reproducing kernel but the added continuity hypothesis.

**Remark 2.3.3.** If $X$ is a real vector space, then a reproducing kernel $K(\cdot, \cdot)$ on $X$ is an inner product if and only if it is non-degenerate and linear in the first coordinate (hence bilinear), where non-degeneracy means that $K(x, y) = 0$, $\forall y \in X$ if and only if $x = 0$.

# Chapter 3

# CBCL Model

## 3.1 Background

The hierarchical model for deriving reproducing kernels proposed by Poggio and Smale in [21] is based on nested patches of the visual plane

$$v_1 \subset \cdots \subset v_{\ell-1} \subset v_\ell \subset \cdots \subset v_d.$$

It generalizes, as well providing a **mathematical framework** in which to understand, the basic feedforward archictecture for visual processing proposed by Poggio and Riesenhuber several years earlier (in [17]). The term CBCL model is loosely applied to either.

This thesis takes advantage of the mathematical structure introduced in [21] to answer certain design questions about the underlying NN's: specifically it addresses the issues of deep vs. shallow and linear vs. nonlinear neural responses using the geometry of Hilbert spaces and linear algebra.

## 3.2 The original framework

One assumes a space of images at each layer, denoted $\mathrm{Im}(v_\ell)$, and also spaces $H_{\ell-1}$ of transformations between successive layers, where $h \in H_{\ell-1}$ is a map $h\colon v_{\ell-1} \to v_\ell$.

Then, given a reproducing kernel $K = K_1$ at the *bottom layer* $\mathrm{Im}(v_1)$, kernels are produced in inductive fashion on layers all the way up, till a kernel is obtained on $\mathrm{Im}(v_d)$. The recursive step, which computes the kernel on $\mathrm{Im}(v_\ell)$ from the kernel on $\mathrm{Im}(v_{\ell-1})$, is determined by the *neural response map*. This is a map

$$N_\ell\colon \mathrm{Im}(v_\ell) \to \mathcal{L}^2(T_{\ell-1}, \nu_{\ell-1}),$$

which associates to each large-sized image $f \in \mathrm{Im}(v_\ell)$ a functional $N_\ell(f)$ on certain special smaller sized images $t \in T_{\ell-1} \subset \mathrm{Im}(v_{\ell-1})$, called the *templates*. Then the recurrence is given by:

$$K_\ell(f,g) = \langle N_\ell(f), N_\ell(g) \rangle_{\mathcal{L}^2(T_{\ell-1},\nu_{\ell-1})}, \; \forall f,g \in \mathrm{Im}(v_\ell) \tag{3.2.1}$$

where $\nu_\ell$ is a measure on $T_\ell$ that has been fixed in advance (for each $\ell = 1, \ldots, d-1$).

**Definition 3.2.1** (Derived kernel, induced kernel)**.** A reproducing kernel that is obtained by the recurrence (3.2.1) is said to be **derived** or **induced** by $N$.

### 3.2.1 Image patches and transformations

Certain axioms on the relationship of the $\mathrm{Im}(v_\ell)$ and $H_\ell$ are required as part of the construction:

$$f \circ h \in \mathrm{Im}(v_{\ell-1}) \quad \text{if } f \in \mathrm{Im}(v_\ell) \text{ and } h \in H_{\ell-1}. \tag{3.2.2}$$

Moreover, each $H_\ell$ is assumed to be finite and a measure $\mu_\ell^H$ is fixed on $H_\ell$.

### 3.2.2 Neural responses

Two examples of neural responses are considered:

$$N_{\max}(f)(u) = \max_{h \in H} K_{\ell-1}(f \circ h, u)$$

and

$$N_{\mathrm{avg}}(f)(u) = \sum_{h \in H} \mu_{\ell-1}^H(h) K_{\ell-1}(f \circ h, u)$$

where $u$ is an arbitrary element of $\mathrm{Im}(v_{\ell-1})$, or of a finite template set $T_{\ell-1} \subset \mathrm{Im}(v_{\ell-1})$. $N_{\mathrm{avg}}$ is referred to as a *weighted average*. Note that it makes use of the postulated measures $\mu_\ell^H$ on the sets $H_\ell$.

## 3.3 The CBCL framework

While the description of the mathematical framework given above and in [21] is tied to visual processing, the framework is applicable much more widely. In the original paper, besides visual processing, text processing is also considered (the model is used to produce reproducing kernels on character strings). Moreover, the work in this thesis began during a visit to Smale in Hong Kong in the summer of 2010, as preparation for an application in bioinformatics. The goal was to use the same framework to develop a reproducing kernel on strings of amino acids, which would predict pairs of peptides likely to provoke similar immune responses. In all these applications there are certain commonalities.

We therefore present here a slightly more abstract version of the framework of [21] which applies to all these settings and which we will use throughout the remainder of the thesis. We refer to this as the "CBCL framework" from now on. It is a mathematical framework, as in [21], for hierarchically deriving a reproducing kernel. This may be viewed as a computation performed by an NN. We will use the term "CBCL model" to refer to the underlying NN model.

**Definition 3.3.1** (CBCL Framework)**.** Instead of using visual patches and image spaces which consist of functions (denoted $f, g$ etc..) on these patches, we start with simpler and more general data:

- sets $X_d, \ldots, X_2, X_1$ (whose elements we denote for example $x$),
- for each $\ell = 2, \ldots, d$ a finite set $H_\ell$ of maps from $X_\ell$ to $X_{\ell-1}$, called *transformations* and a probability measure $\mu_{H_\ell}$ on $H_\ell$,
- for each $\ell = 1, \ldots, d-1$ a finite set $T_\ell \subset X_\ell$ of *templates* and a probability measure $\nu_\ell$ on $T_\ell$,
- a reproducing kernel $K_1$ on $X_1$.

We then use the same recursion as before, given by (3.2.1), and the same neural responses, $N_{\max}$ or $N_{\text{avg}}$, though from time to time it will also be useful to consider arbitrary neural responses $N_\ell : X_\ell \to \mathcal{L}^2_{\nu_{\ell-1}}(T_{\ell-1})$.

Using the new notation, the standard neural responses (from the $\ell$'th layer to the $(\ell-1)$'th) are as follows:

$$N_{\max}(x)(t) = \max_{h \in H} K_{\ell-1}(h(x), t)$$

and

$$N_{\text{avg}}(x)(t) = \sum_{h \in H} \mu^H_{\ell-1}(h) K_{\ell-1}(h(x), t),$$

for $x$ an arbitrary element of $X_\ell$ and $t$ an arbitrary element of $X_{\ell-1}$ (or of a finite template set $T_{\ell-1} \subset X_{\ell-1}$).

**Warning:** The sets $X_\ell$ correspond to the sets $\text{Im}(v_\ell)$ of the orginal framework. However, $H_\ell$ in the original framework was a subset of

$$\text{Maps}(\text{Im}(v_{\ell-1}) \to \text{Im}(v_\ell)).$$

These maps were used only to convert elements of $\text{Im}(v_\ell)$ to elements of $\text{Im}(v_{\ell-1})$. This was done by precomposition (i.e. pull-back):

$$x \in \text{Im}(v_\ell) \rightsquigarrow x \circ h \in \text{Im}(v_{\ell-1}),$$

which amounts to picking out a particular small image from within a large one. We are now retaining only this conversion aspect, and taking $H_\ell$ to be a set of maps from $X_\ell$ to $X_{\ell-1}$, i.e. now $H_\ell$ is a subset of

$$\text{Maps}(X_\ell \to X_{\ell-1}).$$

## 3.4 More detail on the mathematics of this framework

Before we go further, we look in more detail at the mathematical objects and constructions in this framework. First we look at the two types of Hilbert spaces used: reproducing kernel Hilbert spaces and $\mathcal{L}^2$ spaces. Then we look at how they are used.

### 3.4.1 Reproducing kernel Hilbert spaces

We refer to Cucker and Smale's paper [5] Chapter III, for a more detailed discussion of relevant results on reproducing kernel Hilbert spaces. The essential ingredients we will use are given in their Theorem 2 in that Chapter. Paraphrased, it is as follows.

Given a set $X$ with a reproducing (i.e. symmetric and positive definite) kernel $K\colon X \times X \to \mathbb{R}$ (see Definition 2.3.1), there is a unique Hilbert space $\mathcal{H}_K$ of functions on $X$ such that

- $\text{span}\{K_x \mid x \in X\}$ is a dense subset of $\mathcal{H}_K$, and
- $f(x) = \langle K_x, f\rangle_{\mathcal{H}_K}$ for all $f \in \mathcal{H}_K$,

where for all $x \in X$, $K_x$ is defined to be the function $K(x, \cdot)$. The space $\mathcal{H}_K$ is said to be a *reproducing kernel Hilbert space* (RKHS). Suppose we associate to each $x \in X$ the map $F_x(f) := f(x)$. Then $F_x(f) = \langle K_x, f\rangle_{\mathcal{H}_K}$ and so $F_x$ is necessarily a continuous linear functional on $\mathcal{H}_K$ (by the properties of the inner product and its induced metric).

Conversely, an arbitrary Hilbert space $\mathcal{H}$ of functions on a space $X$ is called a reproducing kernel Hilbert space if for each $x \in X$, the so-called evaluation map $F_x$ defined by $F_x(f) = f(x)$ is continuous (i.e. if for each $x \in X$ there exists $M_x > 0$ such that $|F_x(f)| = |f(x)| \leq M_x||f||_{\mathcal{H}}$ for all $f \in \mathcal{H}$). In this case, the Riesz representation theorem produces for each $x \in X$ the unique $K_x$ such that $F_x(f) = \langle K_x, f\rangle_{\mathcal{H}}$ for all $f \in \mathcal{H}$. One may then define $K(x, y)$ to be $\langle K_x, K_y\rangle_{\mathcal{H}}$ and verify that this is a reproducing kernel on $X$. Moreover, $\mathcal{H}$ is then exactly the $\mathcal{H}_K$ defined above.

We remark that the construction in [5] also deals with the continuity of the elements of $\mathcal{H}_K$ as *functions on* $X$ and it assumes both that $X$ is a compact topological space and that $K$ is a Mercer kernel (i.e. continuous, in addition to being symmetric and positive definite). However, the well-definedness of the Hilbert space $\mathcal{H}_K$ and the continuity of $F_x$ *as a linear functional on* $\mathcal{H}_K$ do not require either of these additional properties.

Indeed, if a Hilbert space of real-valued functions on $X$ satisfies the above conditions then it is the completion of the inner product space $\operatorname{span}\{K_x \mid x \in X\}$ whose inner product is defined by linearly extending $\langle K_x, K_y \rangle = K(x, y)$. We refer to $\mathcal{H}_K$ as the reproducing kernel Hilbert space associated to $K$ and denote by $\langle \cdot, \cdot \rangle_{\mathcal{H}_K}$ its inner product (obtained as just mentioned).

### The feature map $\Phi$

Given a space $X$ with reproducing kernel $K(\cdot, \cdot)$ and associated Hilbert space $\mathcal{H}_K$, define a map $\Phi \colon X \to \mathcal{H}_K$ by

$$\Phi \colon x \mapsto K_x.$$

This is called the **feature map** for the kernel $K$ (and Hilbert space $\mathcal{H}_K$). When we have several spaces $X_i$, each with a kernel $K_i(\cdot, \cdot)$, as is the case when using the CBCL framework, we will denote by $\Phi_i$ the feature map for $K_i(\cdot, \cdot)$.

### Note on positive definiteness vs. positive semidefiniteness

Recall the definition of a reproducing kernel as a symmetric, positive definite kernel (see Definition 2.3.1), where positive definiteness means: for all $n \in \mathbb{N}$, for all $x_1, \ldots, x_n \in X$ and for all $a_1, \ldots, a_n \in \mathbb{R}$,

$$\sum_{i,j=1}^{n} a_i a_j K(x_i, x_j) \geq 0. \tag{3.4.1}$$

While this condition is equivalent to the matrix $M = (K(x_i, x_j))$ being positive *semidefinite* for any choices of $m \in \mathbb{N}$ and $x_1, \ldots, x_m \in X$, it in fact results in a positive *definite* bilinear form on the appropriate vector space of functions on $X$ (see Lemma 3.4.1 below). Essentially this happens because the feature map $\Phi \colon x \mapsto K_x$ from $X$ to $\operatorname{span}\{K_x \mid x \in X\}$ sends all $x$ such that $K(x, x) = 0$ to the zero function. More precisely, if we define a bilinear form $\langle \cdot, \cdot \rangle$ on $\operatorname{span}\{K_x \mid x \in X\}$ by setting $\langle K_x, K_y \rangle := K(x, y)$ and extending bilinearly, the Lemma says this form is *positive definite*.[1] This makes $\langle \cdot, \cdot \rangle$ an inner product (as its symmetry and linearity are immediate).

[1] Note that we may view $\mathcal{H}_K$ as $\mathcal{R}/\mathcal{R}_0$ where $\mathcal{R} \subset \mathbb{R}^X$ consists of vectors which have only finitely many nonzero entries and $\mathcal{R}_0$ consists of such vectors which represent the zero function on $X$ via $K$ (i.e. via $\Phi$), namely $\mathcal{R}_0 = \{(\alpha_x)_{x \in X} \mid \sum_{x \in X} \alpha_x K_x = 0\}$. The Lemma is a general fact about bilinear forms $b(\cdot, \cdot)$ defined using a semi-definite matrix. A priori we have such a bilinear form on $\mathcal{R}$ (corresponding to the form $\langle \cdot, \cdot \rangle$ defined above). The Lemma says that all vectors $\widetilde{z} \in \mathcal{Z}_b = \{z \mid b(z, z) = 0\}$ are automatically in the kernel of $b$ (recall $\ker b$ consists of those vectors $z$ for which $b(z, y) = 0,\ \forall y \in X$) and thus these two sets are equal. This means that when we quotient by $\mathcal{R}_0$, which is $\ker b$, then we are in fact quotienting by $\mathcal{Z}_b$ and thus we obtain a nondegenerate bilinear form on the quotient space.

**Lemma 3.4.1.** *Given a reproducing kernel $K$ on the space $X$ and an arbitrary element $z = \sum_{i=1}^{m} \beta_i K_{x_i}$ in span$\{K_x \mid x \in X\}$ such that $\langle z, z \rangle = 0$, it follows that $z(y) = 0$ for all $y \in X$, i.e. $z$ is the zero function on $X$.*

*Proof.* Suppose $\langle \sum_{i=1}^{m} \beta_i K_{x_i}, \sum_{j=1}^{m} \beta_j K_{x_j} \rangle = 0$, i.e. that $\sum_{i,j=1}^{m} \beta_i \beta_j K(x_i, x_j) = 0$. If there exists $y \in X$ such that $z(y) = R \neq 0$, then write $x_{m+1} = y$, $\alpha_{m+1} = 1$, and $\alpha_i = \alpha\beta_i$ for $1 \leq i \leq m$, where

$$\begin{aligned} & \alpha < -\frac{K(y,y)}{2R} \quad \text{if } R > 0, \\ \text{or} \\ & \alpha > -\frac{K(y,y)}{2R} \quad \text{if } R < 0. \end{aligned}$$

Note that $R = z(y) = \sum_{i=1}^{m} \beta_i K(x_i, y)$. We have

$$\begin{aligned} \sum_{i,j=1}^{m+1} \alpha_i \alpha_j K(x_i, x_j) &= \alpha^2 \sum_{i,j=1}^{m} \beta_i \beta_j K(x_i, x_j) + 2\alpha \sum_{i=1}^{m} \beta_i K(x_i, y) + K(y, y) \\ &= 2\alpha \sum_{i=1}^{m} \beta_i K(x_i, y) + K(y, y) \\ &= 2\alpha R + K(y, y) \\ &< -K(y, y) + K(y, y) = 0, \end{aligned}$$

which contradicts (3.4.1). □

### 3.4.2 $\mathcal{L}^2$-spaces

Given a measure space $(T, \mu)$, we denote by

$$\mathcal{L}^2(T, \mu)$$

or $\mathcal{L}^2_\mu(T)$ the Hilbert space of "square integrable functions" on $T$. This is the completion of the standard inner product space obtained by defining

$$\langle f, g \rangle := \left( \int_T f g d\mu \right)^{\frac{1}{2}}$$

for all $f, g$ in the normed vector space of "square integrable functions" on $T$. The elements of this vector space are equivalence classes of functions on $T$ for the relation

$$f \sim g \Leftrightarrow ||f - g||_2 = 0,$$

where

$$||f||_2 := \left( \int_X |f|^2 d\mu \right)^{\frac{1}{2}}.$$

In this quotient space $|| \cdot ||_2$ then defines a norm.

We remark that there is a natural embedding of $\mathcal{H}_{K_\ell}$ in $\mathcal{L}^2(\mathrm{Im}(v_\ell))$. See Cucker and Smale [5] for a discussion of this. Actually, in our framework $T$ will be a finite set of templates, so we may replace the above integrals with weighted sums.

**Remark 3.4.2.** We may sometimes omit mention of the measure $\mu$ when this is clear from the context or irrelevant; then we simply write $\mathcal{L}^2(T)$.

### 3.4.3 Abstract derived kernels

We make the following abstract observation. Let $X$ and $Y$ be two sets and $\phi\colon X \to Y$. Suppose that $Y$ is equipped with a reproducing kernel $K$ (see Definition 2.3.1). For example, if $Y$ is a Hilbert space, then the inner product on $Y$ is a reproducing kernel. Then we can define a reproducing kernel on $X$ by pulling $K$ back:

**Proposition 3.4.3.** *The map $\phi^*K\colon X \times X \to \mathbb{R}$ defined by*

$$\phi^*K(x_1, x_2) = K(\phi(x_1), \phi(x_2))$$

*is a reproducing kernel on $X$.*

*Suppose $X$ and $Y$ are in fact vector spaces and $K$ is an inner product on $Y$. Then if the map $\phi$ is linear and injective, the form $\phi^*K$ will be an inner product on $X$. On the other hand, linearity of $\phi$ and non-degeneracy of $\phi^*K$ together will imply injectivity of $\phi$, while surjectivity of $\phi$ and bilinearity of $\phi^*K$ will imply linearity of $\phi$.*

*Proof of Proposition 3.4.3.* The symmetry and positive definiteness of $\phi^*K$ follow immediately from the symmetry and positive definiteness of $K$. In the case where $Y$ is a vector space with inner product $K$, then linearity of $\phi$ implies bilinearity of $\phi^*K$ and injectivity of $\phi$ implies nondegeneracy of $\phi^*K$ thus making it an inner product on $X$.

Suppose $\phi$ is linear but not injective and let $x_1, x_2$ be two distinct elements of $X$ such that $\phi(x_1) = \phi(x_2)$. Then $\phi^*K(x_1 - x_2, x_1 - x_2) = K(0,0) = 0$ and yet $x_1 - x_2 \neq 0$.

Suppose $\phi$ is surjective and $\phi^*K$ is bilinear. Then for any $x_1, x_2, z \in X$,

$$\begin{aligned}\phi^*K(x_1 + x_2, z) &= \phi^*K(x_1, z) + \phi^*K(x_2, z) \\ &= K(\phi(x_1), \phi(z)) + K(\phi(x_2), \phi(z)) = K(\phi(x_1) + \phi(x_2), \phi(z))\end{aligned}$$

while $\phi^*K(x_1 + x_2, z) = K(\phi(x_1 + x_2), \phi(z))$ by definition. Thus, $K(\phi(x_1 + x_2), \phi(z)) = K(\phi(x_1)+\phi(x_2), \phi(z))$. Since this holds for all $\phi(z)$ and $\phi$ is surjective, we have $\phi(x_1+x_2) = \phi(x_1) + \phi(x_2)$ by the non-degeneracy of $K$. □

The above construction of pullback kernel is exactly that used in the recurrence (3.2.1). Proposition 3.4.3 shows this indeed yields a reproducing kernel.

### 3.4.4 Normalization

In practise, one may wish to use *normalized neural responses*, so that the derived kernel has values between $-1$ and 1 and $K_2(x,x) = 1$ for all $x \in X_2$. We remark that the latter condition implies the former.

**Lemma 3.4.4.** *If $K$ is a reproducing kernel on $X$ such that $K(x,x) = 1$, $\forall x \in X$, then $-1 \leq K(x,y) \leq 1, \forall x, y \in X$.*

*Proof.* This is a consequence of the Cauchy-Schwarz inequality for positive definite kernels. To give a direct proof, take $x_1, x_2 \in X$ arbitrary and $\alpha_1 = \alpha_2 = 1$. Then the positive definite condition implies $\sum_{i,j=1}^{2} \alpha_i \alpha_j K(x_i, x_j) \geq 0$. So,

$$\begin{aligned} K(x_1,x_1) + K(x_2,x_2) + 2K(x_1,x_2) &\geq 0 \\ 2 + 2K(x_1,x_2) &\geq 0 \\ K(x_1,x_2) &\geq -1. \end{aligned}$$

On the other hand, by taking $\alpha_1 = 1, \alpha_2 = -1$ we obtain

$$2 - 2K(x_1,x_2) \geq 0$$

so $K(x_1,x_2) \leq 1$. □

In order to obtain a normalized version of a neural response $N : X_2 \to \mathcal{L}^2(T)$, it therefore suffices to multiply, for each $x \in X_2$, the value of $N(x)$ by a scalar $\alpha(x)$ that is chosen so that $\langle \alpha(x)N(x), \alpha(x)N(x) \rangle_{\mathcal{L}^2(T)} = 1$.

**Definition 3.4.5** (Normalized neural response)**.** Specifically, given a neural response $N : X_2 \to \mathcal{L}^2_\nu(T)$, the normalized neural response, denoted $\hat{N}$ is defined by

$$\hat{N}(x)(t) = \alpha(x)N(x)(t),\ \forall x \in X_2, \forall t \in T,$$

where

$$1/\alpha(x) = ||N(x)||_{\mathcal{L}^2_\nu(T)} = \left( \sum_{t \in T} \nu(t)[N(x)(t)]^2 \right)^{1/2}$$

(assuming finite $T$).

### 3.4.5 Non-degenerate kernels

Given a finite set $X$, consider the Hilbert space $\mathcal{L}^2(X)$ (defined using the empirical, i.e. uniform, measure on $X$). $\mathcal{L}^2(X)$ consists of all real-valued functions on $X$ and for any two such functions, $f$ and $g$, we have

$$\langle f, g\rangle_{\mathcal{L}^2(X)} := \sum_{x\in X} \frac{1}{N} f(x)g(x),$$

where $N = |X|$. Or alternatively, one could consider $\mathbb{R}^{|X|}$ with the standard dot product $u \cdot v := \sum_{x\in X} u_x v_x$; this Hilbert space is isomorphic to $\mathcal{L}^2(X)$. By Section 3.4.1, there exists a reproducing kernel $K(\cdot,\cdot)$ on $X$ such that $\mathcal{L}^2(X) = \mathcal{H}_K$. This is because $\mathcal{L}^2(X)$ is finite dimensional and so the evaluation map $F_x$ defined by $F_x(f) = f(x)$, being linear, is necessarily continuous.

**Lemma 3.4.6.** *The associated feature map $\Phi : X \to \mathcal{L}^2(X)$ is injective and in fact all $\Phi(x)$, $x \in X$ are linearly independent. The function $\Phi(x) = K_X$ takes on the value 1 on $x$, and 0 on all other $y \in X$.*

We give the proof for $\mathcal{L}^2(X)$ as stated, but the proof goes through for any isomorphic Hilbert space (for example $\mathbb{R}^{|X|}$).

*Proof.* If $\sum_{i=1}^{M} a_i K_{x_i} = 0$ for some $M \in \mathbb{N}, a_1, \ldots, a_M \in \mathbb{R}$, and distinct $x_1, \ldots, x_M \in X$ then $\langle \sum_{i=1}^{M} a_i K_{x_i}, f\rangle_{\mathcal{L}^2(X)} = 0$ for all functions $f$ on $X$, which, by the reproducing property, is equivalent to $\sum_{i=1}^{M} a_i f(x_i) = 0$ for all functions $f$ on $X$, and therefore implies $a_i = 0, \forall i$ because we can for example take $f$ such that $f(x_i) = a_i$.

Now, given $x, x' \in X$, let $f, f'$ be the functions which are 1 on $x$, respectively on $x'$, and 0 elsewhere, then $f = K_x, f' = K_{x'}$ and thus

$$K(x, x') = \langle f, f'\rangle_{\mathcal{L}^2(X)}.$$

Indeed $f(y) = 0$ for $y \neq x$ means $\langle f, K_y\rangle_{\mathcal{L}^2(X)} = 0$ by the reproducing property, so $f \in \operatorname{span}(K_x)$ (i.e. $f$ is a multiple of $K_x$) and therefore $f = K_x$ because $f(x) = \langle f, K_x\rangle_{\mathcal{L}^2(X)} = 1$. □

**Definition 3.4.7** (Non-degenerate kernel)**.** We call a kernel $K$ on a finite set $X$ **non-degenerate** if $\dim \mathcal{H}_K = |X|$.

We have given one example of a non-degenerate kernel $K$, with $\mathcal{L}^2(X)$ as its RKHS. All non-degenerate kernels on a finite $X$ have RKHS of same dimension, i.e. have isomorphic RKHS's. Lemma 3.4.6 may be applied to any of them.

Now given an arbitrary kernel $K'$ on $X$, its reproducing kernel Hilbert space $\mathcal{H}_{K'}$ is isomorphic to a subspace of $\mathcal{L}^2(X)$, since $\dim \mathcal{H}_{K'}$ is at most the dimension of $\mathcal{L}^2(X)$, namely $|X|$, and these are finite-dimensional Hilbert spaces. Let $\Theta : \mathcal{H}_{K'} \to \mathcal{L}^2(X)$ be such an injective isometry. Then we may define

$$\Psi := \Theta \circ \Phi_{K'}$$

which is a map $\Psi : X \to \mathcal{L}^2(X)$. We observe $K'(x, x') = \langle \Psi(x), \Psi(x') \rangle_{\mathcal{L}^2(X)}$ for all $x, x' \in X$. Indeed:

$$\begin{aligned} \langle \Psi(x), \Psi(x') \rangle_{\mathcal{L}^2(X)} &= \langle \Theta \circ \Phi_{K'}(x), \Theta \circ \Phi_{K'}(x') \rangle_{\mathcal{L}^2(X)} \\ &= \langle \Phi_{K'}(x), \Phi_{K'}(x') \rangle_{\mathcal{H}_{K'}} \\ &= K'(x, x'). \end{aligned}$$

So,

**Lemma 3.4.8.** *All reproducing kernels on $X$ are pullbacks of the inner product in $\mathcal{L}^2(X)$. Alternatively, any reproducing kernel $K$ satisfies:*

$$K(x, x') = \Theta(x) \cdot \Theta(x')$$

*where $\Theta$ is some map $\Theta : X \to \mathbb{R}^{|X|}$, and the operation on the right hand side is the Euclidean dot product.*

**Remark 3.4.9.** Another related observation is that any Hilbert space $\mathcal{H}$ of functions on a finite set $X$ is isomorphic to a subspace of $\mathcal{L}^2(X)$ since $\dim \mathcal{H} \leq \dim \mathcal{L}^2(X) < \infty$.

## 3.5 Universal Axiom for Neural Responses

The recurrence (3.2.1) and the neural responses of the CBCL framework are similar to the recurrence and neural responses of convolution networks. We formalize this similarity in terms of an axiom which both frameworks satisfy. We then define an extension of the CBCL framework (the "extended CBCL framework") – in terms of this Axiom. It encompasses convolution neural networks and CBCL networks. In Chapter 4, we show that any hierarchy of the extended CBCL framework can be decomposed into simple pieces and its depth thus reduced.

**Recursion** To avoid excess notation we specialize to layers 1 and 2, and drop subscripts. Then in both cases (CBCL and convolution networks - see Section 3.7 for a discussion of the latter) we have a recurrence of the form,

$$K_2(x, x') := \langle N(x), N(x') \rangle_{\mathcal{L}^2(T_1)} \tag{3.5.1}$$

for $x, x' \in X_2$ with neural response $N : X_2 \to \mathcal{L}^2(T_1)$. The induced $K_2(\cdot, \cdot)$ is indeed a reproducing kernel by Proposition 3.4.3. In fact it would still be one if we were to pull back a reproducing kernel defined on an arbitrary space of functions on $T_1$ (instead of the standard inner product on $\mathcal{L}^2(T_1)$); however, we will not need this generality. See also the comment in Section 3.5.3.

In both models, the neural responses $N$ and kernels $K_1$ satisfy the following axiom[2]:

**Axiom 3.5.1.** *All the functions in $N(X_2) \subset \mathcal{L}^2(T_1)$ are of a special form, namely for each $x_2 \in X_2$ there exists $N_{base}(x_2) \in Maps(X_1 \to \mathcal{H}_{K_1})$ such that,*

$$N(x_2)(t) = \langle N_{base}(x_2)(t), K_{1,t} \rangle_{K_1} = N_{base}(x_2)(t)(t), \, \forall t \in T_1. \tag{3.5.2}$$

This means that *evaluation* of the elements of $N(X_2) \subset \mathcal{L}^2(T_1)$ on points of $T_1$ is done **using the inner product of $\mathcal{H}_{K_1}$** and equation (3.5.2) specifies exactly how. This formula will be essential to what we do in the remainder.

**Remark 3.5.2.** In fact the axiom also provides a way to define the values of $N(x_2)$ on all elements of $X_1$ (not just the particular subset $T_1 \subset X_1$).

**Remark 3.5.3 (Generality of the Axiom and the space of functions on $T_1$).** The above Axiom is quite weak. Let $N$ be an *arbitrary* map $N \colon X_2 \to \mathcal{L}^2(T_1)$ and assume that *for each $t \in T_1$, $K_{1,t}$ is not the zero function on $X_1$*. Then we claim that $N$ satisfies Axiom 3.5.1.

In practice, most neural responses are defined directly in terms of elements of $\mathcal{H}_{K_1}$ so the Axiom is immediately verified. In this sense it serves mainly as a comment on the means of definition. Its use as an axiom is in allowing us to formalize the notion of linearity of neural responses. We do this is Section 3.5.1.

We now prove the claim. Let $N$ be given and assume the stated nondegeneracy hypothesis on $K_{1,t}$ for each $t \in T_1$. Define $N_{\text{base}} \in \text{Maps}(X_1 \to \mathcal{H}_{K_1})$ as follows. For each $t \in T_1$, let $y(t) \in X_1$ such that $K_1(t, y(t)) \neq 0$. For each $x_2 \in X_2$ and $t \in T_1$, set,

$$N_{\text{base}}(x_2)(t) := \frac{N(x_2)(t)}{K_1(t, y(t))} K_{1,y(t)} \in \mathcal{H}_{K_1}.$$

[2]The reader may easily verify this for the CBCL model; we show it for convolution neural networks in Section 3.7.

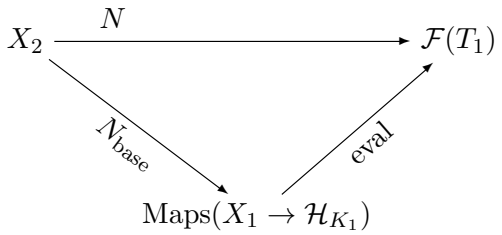
And for $x_1 \in X_1 \setminus T_1$ define $N'_{\text{base}}(x_2)(x_1) \in \mathcal{H}_{K_1}$ arbitrarily. Then, for all $x_2 \in X_2$ and $t \in T_1$ we have,

$$\begin{aligned} N_{\text{base}}(x_2)(t)(t) = \frac{N(x_2)(t)}{K_1(t, y(t))} K_{1,y(t)}(t) &= \frac{N(x_2)(t)}{K_1(t, y(t))} K_1(t, y(t)) \\ &= N(x_2)(t), \end{aligned}$$

so $N$ indeed satisfies Axiom 3.5.1.

A second consequence of the above construction is the following. Assume $T_1$ is finite. Had we postulated an arbitrary Hilbert space of functions $\mathcal{F}(T_1)$ on $T_1$ in the recursion and Axiom, we could replace $\mathcal{F}(T_1)$ by $\mathcal{L}^2(T_1)$ and replace $N\colon X_2 \to \mathcal{F}(T_1)$ by $N'\colon X_2 \to \mathcal{L}^2(T_1)$ where $N'(x_2) = \Theta(N(x_2))$ for all $x_2 \in X_2$ with $\Theta : \mathcal{F}(T_1) \to \mathcal{L}^2(T_1)$ being an isometry that embeds the Hilbert space $\mathcal{F}(T_1)$ in $\mathcal{L}^2(T_1)$ (such an isometry must exist since $\dim \mathcal{F}(T_1) \leq |T_1| = \dim \mathcal{L}^2(T_1) < \infty$). Then, using the above construction, one sees that the new $N'$ would still satisfy Axiom 3.5.1. Its derived kernel would be the same as that derived by $N$ since $\Theta$ is an isometry. Thus, we have lost no generality in restricting ourselves to $\mathcal{F}(T_1) = \mathcal{L}^2(T_1)$.

Axiom 3.5.1 can be re-expressed by saying, $N$ **factors through** $\text{Maps}(X_1 \to \mathcal{H}_{K_1})$:




Here the map "eval" is **evaluation using the inner product** of $\mathcal{H}_{K_1}$. Specifically given $f \in \text{Maps}(X_1 \to \mathcal{H}_{K_1})$ and $x_1 \in X_1$, it is defined by

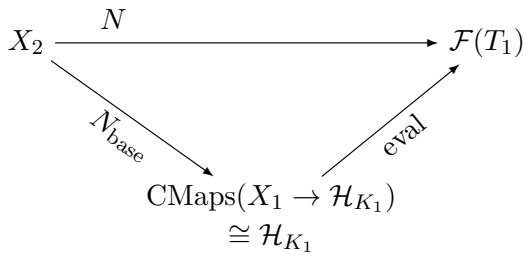
$$(\text{eval } f)(x_1) = \langle f(x_1), K_{1,x_1} \rangle_{K_1}.$$

### 3.5.1 Aside on Linearity

**Definition 3.5.4 (Linear Neural Responses).** When for each $x_2 \in X_2$, $N_{\text{base}}(x_2)(x_1) \in \mathcal{H}_{K_1}$ is constant as a function of $x_1 \in X_1$, then we say the neural response $N$ is **linear**.

In the alternate terminology given above, this corresponds to $N$ factoring through

*constant maps*, $\mathrm{CMaps}(X_1 \to \mathcal{H}_{K_1}) \cong \mathcal{H}_{K_1}$:




**Remark 3.5.5.** This definition is consistent with the common CBCL usage of *linear neural response*; the average neural response is linear by this definition and the max is not. Moreover, as we will see, the neural responses of convolution neural networks which use a non-linear activation function (i.e.squashing function) are in general non-linear.

The justification for this terminology is that upon applying "eval" to a constant map, the element of $\mathcal{L}^2(T_1, \nu_1)$ that one obtains is actually just the restriction to $T_1$ of an element of the Hilbert space $\mathcal{H}_{K_1}$ (i.e., it is a *linear* functional on $\Phi_1(T_1)$):

$$\begin{aligned} & f \in \mathrm{CMaps}(X_1 \to \mathcal{H}_{K_1}) \text{ s. t. } (\forall x_1 \in X_1) f(x_1) = F \in \mathcal{H}_{K_1} \\ & \Rightarrow \quad (\mathrm{eval}\ f)(x_1) = \langle F, K_{1,x_1} \rangle_{K_1} = F(x_1) \end{aligned}$$

where the last equality follows from the reproducing property of $\mathcal{H}_{K_1}$. Note that in fact the element obtained is *exactly* $F$ itself.

**Remark 3.5.6.** By abuse of notation, **in the case of linear neural responses** we will consider $N_{\text{base}}$ to be a map into $\mathcal{H}_{K_1}$ itself (not the isomorphic space $\mathrm{CMaps}(X_1 \to \mathcal{H}_{K_1})$): given $x_2 \in X_2$, $N_{\text{base}}(x_2) \in \mathcal{H}_{K_1}$ is just the element $F$ mentioned above (i. e. the constant value in $\mathcal{H}_{K_1}$ to which the true $N_{\text{base}}(x_2)$ is equal for all $x_1 \in X_1$).

**Example 3.5.1.** To give two concrete examples,

$$N^{\max}_{\text{base}}(x)(t) = \max_{h \in H} K_{1,h_t(x)},$$

where $h_t$ is the element of $H$ for which $K_1(h(x), t)$ is maximized, and

$$N^{\text{avg}}_{\text{base}}(x)(t) = \sum_{h \in H} \mu^H_{\ell-1}(h) K_{1,h(x)}.$$

In the first case, the element of $\mathcal{H}_{K_1}$ to which $N^{\max}_{\text{base}}(x)(t)$ corresponds depends on $t$. In the second case, it does not. $N^{\max}_{\text{base}}$ is therefore *not* linear, whereas $N^{\text{avg}}$ *is* linear.

**Example 3.5.2.** For a simple (toy) example of a linear neural response, not a priori expressed as a weighted average, consider the kernel $K$ on the set $X_1 = \{a, b\}$ given by

$$K(i,j) = \begin{cases} 1 \text{ if } i = j, \\ 0 \text{ otherwise} \end{cases}$$

Let $X_2 = [0,1]$, the closed unit interval, and define a neural response $N_{\text{base}}$ by $N_{\text{base}}(x) = xK_a + (1-x)K_b$ for each $x \in X_2$. Then $N_{\text{base}}$ defines a *linear* neural response since for each $x \in X_2$, $N_{\text{base}}(x)$ is an element of $\mathcal{H}_K$.

### 3.5.2 Base neural responses

We now look at Axiom 3.5.1 more closely.

**Proposition 3.5.7.** *Assume Axiom 3.5.1. $N_{base}$ and $T_1$ uniquely determine $N$. As a partial converse, if span of $\{K_{1,t} \colon t \in T\}$ is equal to all of $\mathcal{H}_{K_1}$ and $N$ is linear, then $N_{base}$ is uniquely determined.*

*Proof.* The first direction is trivial: suppose we are given $N_{\text{base}}$ and $T_1$, then Axiom 3.5.1 defines $N(x_2)(t) = \langle N_{\text{base}}(x_2)(t), K_{1,t} \rangle_{K_1}$.

Now suppose $N$ linear satisfies the axiom and there are $N_{\text{base}}$ and $N'_{\text{base}}$ such that

$$\langle N_{\text{base}}(x_2), K_{1,t} \rangle_{K_1} = N(x_2)(t) = \langle N'_{\text{base}}(x_2), K_{1,t} \rangle_{K_1}$$

for all $t \in T_1$. Then $N_{\text{base}}(x_2) = N'_{\text{base}}(x_2)$ by the nondegeneracy of the inner product and the fact that the $K_{1,t}$ span $\mathcal{H}_{K_1}$. □

**Definition 3.5.8.** Given $N$ and $N_{\text{base}}$ satisfying Axiom 3.5.1, we call $N_{\text{base}}$ a **base neural response** for $N$. Given a base neural response $N_{\text{base}}$ and a choice of template set $T_1$ we call $N$ the **neural response associated to** $N_{\text{base}}$ and $T_1$. and denote it $\rho_{T_1}(N_{\text{base}})$:

$$\rho_{T_1}(N_{\text{base}}) : X_2 \to \mathcal{L}^2(T_1).$$

We have

$$\rho_{T_1}(N_{\text{base}})(x_2)(t) = N_{\text{base}}(x_2)(t)(t)$$

for all $x_2 \in X_2, t \in T_1$.

Note that $\rho_{T_1}$ is thus a functional which takes a map $N_{\text{base}} : X_2 \to \text{Maps}(X_1 \to \mathcal{H}_{K_1})$ to a map $N = \rho_{T_1}(N_{\text{base}}) : X_2 \to \mathcal{L}^2(T_1)$. When the dependence of $\mathcal{L}^2(T_1) = \mathcal{L}^2(T_1, \nu_1)$ on $\nu_1$ needs to be made explicit, we will write $\rho_{T_1,\nu_1}$.

## 3.6 The extended CBCL Framework

**Definition 3.6.1** (Extended CBCL Framework)**.** The extended CBCL framework is exactly as the CBCL framework, with spaces $X_1, \ldots, X_d$ (see Definition 3.3.1), except that:

- a given layer may have *several* kernels, $K_\ell^1, \ldots, K_\ell^s$, (all on $X_\ell$) and corresponding neural responses $N_\ell^1, \ldots, N_\ell^s$ into Hilbert spaces $\mathcal{L}^2(T_{\ell-1}^1, \nu_{\ell-1}^1), \ldots, \mathcal{L}^2(T_{\ell-1}^s, \nu_{\ell-1}^s)$, and

- each $K_\ell^i$ satisfies a recursion of the form

$$K_\ell^i(x, x') = \langle N_\ell^{1i}(x), N_\ell^{1i}(x') \rangle_{\mathcal{L}^2(T_\ell^{1i}, \nu_\ell^{1i})} + \ldots$$
$$\ldots + \langle N_\ell^{ri}(x), N_\ell^{ri}(x') \rangle_{\mathcal{L}^2(T_\ell^{ri}, \nu_\ell^{ri})},$$

  for $x, x' \in X_\ell$, where $K_{\ell-1}^{1i}, \ldots, K_{\ell-1}^{ri}$ are the kernels on level $\ell - 1$ and each neural response $N_\ell^{ji}$ satisfies Axiom 3.5.1 for lower kernel $K_{\ell-1}^{ji}$.

**Notation:** The number $r = r_i$ of lower layer kernels may also depend on $i$, but we have reduced the notation. Moreover, when dealing with a single kernel $K_\ell^i$ with $i$ fixed, we will omit the superscript $i$ in the names of neural responses, and lower level templates and kernels which are used to construct $K_\ell^i$. Note that, given $s$ kernels $K_\ell^1, \ldots, K_\ell^s$ at the layer $\ell$, such that the $i$'th kernel $K_\ell^i$ depends on $r_i$ kernels at the layer $\ell - 1$, there will in total be $S = \sum_{i=1}^s r_i$ kernels at layer $\ell - 1$ (and this $S$ will play the role of $s$ when we recurse again).

We call the underlying model of NN the *extended CBCL model*. It is described, along with the CBCL model, in Section 3.12.

**Remark 3.6.2.** We note that $N_\ell^{ji}$ is not restricted in any other way in this framework (besides satisfying Axiom 3.5.1). One may for example define $N_\ell^{ji}$ as a weighted average using an arbitrary transformation set $H^{ji}$ and/or weights $\mu^{ji}(h) \in \mathbb{R}$. This is technically allowed in the CBCL model as well, though usually the transformations and weights there are viewed as having some physical meaning.

As mentioned in Section 3.5.1, for a *linear* neural response $N_2$ which satisfies Axiom 3.5.1, one may always assume that $N(x_2)$ – as a set of functions on $T_1$ – is given by *linear functionals* on $\mathcal{H}_{K_1}$. Their Riesz representatives then span a *vector subspace of* $\mathcal{H}_{K_1}$. It will be convenient to view this subspace as carrying a different inner product, namely, the one inherited from $\mathcal{L}^2(T_1)$, instead of the standard inner product on $\mathcal{H}_{K_1}$.

## 3.7 Convolution neural networks and the extended CBCL framework

We abbreviate these CNN's. As remarked in Chapter 2, their lower (not fully-connected) layers alternate so-called convolution layers and max or sub-sampling layers. In each of those layers, the named operation (convolution, max, sub-sampling) is typically followed by an additional function before the output of the layer is produced and may serve as input for the next layer. The additional function is either a "squashing" function (a.k.a. "activation function") - typically a sigmoid - or addition of a bias term (constant) followed by a squashing function. We will consider it to be the identity function if no such additional function is present.

We claim each of the named operations (convolution, max, sub-sampling) is in fact a weighted average or max as defined above for the CBCL model but that when it is followed by the additional function (squashing or addition of bias and then squashing) we may handle this with a neural response which still satisfies Axiom 3.5.1. Thus the lower (non fully-connected) part of a CNN may be viewed as an *extended CBCL NN*. The remainder of this Section explains this conversion.

**CNN's**

We look first at convolutions. Given an input image $x$ whose pixels are labeled by the square grid $[-m,m]\times[-m,m]$, a convolution (in LeNet, for example) produces as output several feature vectors $y^k, k\in K$ for each of which the $i$'th pixel, $i\in[-p,p]\times[-p,p]$, is

$$y_i^k = \sum_{h\in H} W_h^k \cdot [\mathrm{sub}_i(x)]_h.$$

Here the set $H=[-n,n]\times[-n,n]$ is specifically a grid, $\mathrm{sub}_i(x)$ denotes a subimage of $x$ centered at the $i$'th pixel, and $[\mathrm{sub}_i(x)]_h$ denotes the $h$'th pixel of $\mathrm{sub}_i(x)$. We are assuming that $m>n$ is sufficiently large that all the subimages $\mathrm{sub}_i(x)$ of $x$, for $i\in[-p,p]\times[-p,p]$, are well-defined[3]. Each of the feature vectors $y^k$ uses the same input $x$ (and same "receptive field') and each is followed by addition of a bias term and application of a sigmoid function. The pixels of each feature vector are said to form a "plane".

LeNet learns the weights $W_h$ and also the bias term $b$ which will be added before a sigmoid function is applied.

In the above, we have taken $x$ as the bottom-layer input of a CNN and so the $h$'th pixel of $\mathrm{sub}_i(x)$ is/has in fact a $1\times 1$ receptive region, $\mathrm{sub}_i(x)$ is a $[-n,n]\times[-n,n]$ receptive region, and $x$ is the full $[-m,m]\times[-m,m]$ receptive region. However, the above

[3] i.e. $p+n\le m$.

convolution formula may also be applied at higher layers. In that case, $x$ would be a feature map with receptive region *larger* than $[-m,m]\times[-m,m]$. This feature map would still have $(2m+1)^2$ pixels, labeled by $[-m,m]\times[-m,m]$, but each of these would correspond to more than a single pixel of the bottom-layer input of the CNN. For example, the $h$'th pixel of $\text{sub}_i(x)$ might have a $5\times 5$ receptive region.

We remark that when applying the above convolution formula at *higher levels* (where $h(x)$ is a pixel of a feature map and so corresponds to a receptive field larger than $1\times 1$), the convolution operation, may involve *several* lower-level feature maps for the same receptive field. The $i$'th pixel of the output feature map $y$ will then be given by

$$y_i^k = \sum_{h\in H, j\in J} W_h^{jk}\cdot[\text{sub}_i(x)]_{h,j}, \tag{3.7.1}$$

where $J$ indexes the feature maps (i.e. planes) of the lower level, and $[\text{sub}_i(x)]_{h,j}$ is the $h$'th pixel in the $j$'th plane for receptive field $\text{sub}_i(x)$. Note: we write $W_h^{jk}$ for the $h$'th entry of the weights matrix $W^{jk}$ which comes from the 4-tensor of weights $W$ (recall that $h=(a,b)\in[-n,n]\times[-n,n]$ is usually written $ab$ in the CNN literature, so one sees $W_{ab}^{jk}$).

**CBCL NN's**

Now, consider a CBCL model for vision where the over-all input is an image $z\in X_d$ (i.e. this is the full-sized image that feeds into the full $d$-layer architecture). Using the notation established so far for CBCL frameworks, suppose $T_1=\{s\}$ consists of a *single template*, such that $K_{1,s}$ is not the zero function. As long as $K_1$ is not trivial (constantly equal to 0), such an $s$ must exist.

Consider the following neural response $N\colon X_2\to\mathcal{L}^2(T_1)$, defined by $N(x)(s)=\psi(N^{\text{avg}}(x)(s))$ for each $x\in X_2$ and $s\in T_1$. It satisfies Axiom 3.5.1, by Remark 3.5.3 for any map $\psi$. We will let $\psi\colon\mathbb{R}\to\mathbb{R}$ be defined by $\psi(r)=A\sigma(r+B)$ where $B$ is the bias term we added after the convolution in the CNN above, $\sigma$ is the squashing function used there, and $A$ is a constant we will determine in a moment.

Using the simple version of the extended CBCL recursion (with only a single lower kernel), see (3.5.1), we have

$$K_2(x,x')=\langle N(x),N(x')\rangle_{\mathcal{L}^2(T_1)}.$$

We assume for simplicity the uniform measure (always 1) on $T_1$ when defining $\mathcal{L}^2(T_1)$ (there is only one element in $T_1$ anyway).

To make the CNN $\to$ extended CBCL correspondence, let $X_2$ consist of images $x'$ of the size of $\text{sub}_i(x)$ in the previous CNN example. We will denote by $\mathcal{H}$ the set of transformations - still to be determined - which are used by $N^{\text{avg}}$. For all $x'\in X_2$ and

$t \in T_2$:

$$\begin{aligned} K_2(x',t) &= \psi\left(\sum_{h\in\mathcal{H}} \mu(h) K_1(h(x'),s)\right)\ \psi\left(\sum_{h\in\mathcal{H}} \mu(h) K_1(h(t),s)\right) \\ &= A^2\sigma\left(\sum_{h\in\mathcal{H}} \mu(h) K_1(h(x'),s)+B\right)\ \sigma\left(\sum_{h\in\mathcal{H}} \mu(h) K_1(h(t),s)+B\right) \end{aligned}$$

For a fixed $t \in T_2$, this becomes,

$$K_2(x',t) = A^2 c(s,t)\,\sigma\left(\sum_{h\in\mathcal{H}} \mu(h) K_1(h(x'),s)+B\right),$$

where $c(s,t) = \sigma[\sum_{h\in\mathcal{H}} \mu(h) K_1(h(t),s) + B]$ is a constant depending on $s, t, B$. Assuming $\sigma > 0$ as is the case with the logistic function, we have $1/c(s,t) > 0$, so we may define $A = \sqrt{1/c(s,t)}$. For other squashing functions, such as the hyperbolic tangent or arctangent, which are odd, there are special circumstances where it is also easy to guarantee $c(s,t) > 0$. We consider only the simple case of the logistic for now. With the above value of $A$, we obtain,

$$K_2(x',t) = \sigma\left(\sum_{h\in\mathcal{H}} \mu(h) K_1(h(x'),s)+B\right).$$

This has the same general form as the output $y_i$ of the earlier CNN, keeping in mind that $x'$ stands for $\mathrm{sub}_i(x)$. There is however a problem.

**Problem** In general, we need to produce output which is a linear combination of pixels of several feature maps from the lower layer. To preserve the spatial significance of pixels in the CNN, we will want the $h$'th pixel of the $j$'th plane to represent the degree of presence of the $j$'th feature in the subimage centred at the location determined by $h$ (in the global input $x$). We try to have it represent $K_{\ell-1}(h(y),s)$ for $y \in X_\ell$ where $y \in X_\ell$ is the portion of $x$ handled by the $\ell$'th level gate that the current pixel feeds into. Such linear combinations of kernel values can be accomplished by the extended CBCL model in one of two ways:

- by pooling over a set $H$ of transformations in a weighted average, or,
- by summing distinct kernels $K^1_{\ell-1}, \ldots, K^r_{\ell-1}$ (see Section 3.6).

If we use the **first** method, then (assuming $r = 8$ features on the lower layer) we must define artificial image spaces of the type $\tilde{X}_{\ell-1} := X_{\ell-1} \times \{1, \ldots, 8\}$ where $X_{\ell-1}$ is a usual space of $2D$-images, so that a set of transformations $\mathcal{H}$ from higher- to lower-level images

can map variously into one of the lower planes, and the lower kernel $K_{\ell-1}$ can take on different values for the same $2D$-image in different planes. We could do this by defining $\mathcal{H} = \bigcup_{j\in J} H^j$ where $J$ indexes the planes, i.e. feature maps, at the lower layer of the CNN, $H^j = [-n,n]\times[-n,n], \forall j \in J$ and for each $j \in J$, $x' \in X_\ell$ and $h \in H^j$, $h(x')$ is an appropriately sized sub-image centred on the $h$'th pixel in the $j$'th plane of $x'$. In that case, we would define $\mu(h^j) := W_h^j$ so that pooling over $\mathcal{H}$ gives the desired linear combination, with the $h$'th pixel of the $j$'th plane in the lower layer representing $K_{\ell-1}(h^j(x'), s)$ for $x' \in X_\ell$. However, when we also have several features on the higher level this no longer works: we would need to handle different weight matrices $W^{jk}$ for each pair $jk$ where $k$ is the upper plane and $j$ the lower one. We are left with the **second** option: summing distinct kernels. Here, the problem is that we must use the activation function on the *sum* of the kernels, but there is no place in the extended CBCL model to squash *after* summing the kernels; it must be done by a neural response. Therefore, we are forced to introduce an *extra layer* to perform the squashing. This can easily be done - by using the method described above (with $\psi$ and $A$, assuming the logistic, possibly preceded by addition of a bias). We take $X_{\ell-1} = X_\ell$ and a set of transformations $\mathcal{H} : X_\ell \to X_{\ell-1}$ which contains only the identity transformation. We assume all weights are 1 and $B = 0$. Then, letting $s = t$ be a fixed element of $X_\ell$, we obtain, $K_\ell(x,t) = \sigma(K_{\ell-1}(x,t) + B)$, for all $x \in X_\ell$.

We now continue the discussion, assuming there is no squashing function and that we can *choose our bias $B$ arbitrarily* (since we will handle both of these at once with an extra layer as just described). Suppose we have $r$ feature maps on the lower layer and $p$ feature maps on the upper layer. *Fix one of these upper feature maps, the $k$'th.* To simplify notation, let us assume $X_1$ and $X_2$ are the sets of possible input to the lower and upper gates respectively[4]. We seek to define $K_2(\cdot,\cdot)$ appropriately. In fact, there should properly be an index on this kernel to indicate its dependence on $k$, but we will omit it to avoid cumbersome notation, and instead keep $k$ fixed from now on. We consider an extended CBCL recursion *with several component kernels*:

$$K_2(x,x') = K_2^1(x,x') + \ldots + K_2^r(x,x'). \tag{3.7.2}$$

For consistency with earlier notation, we write $J := \{1,\ldots,r\}$. Let $A_j, B_j, j \in J$ be integers, *to be determined*. We assume that for each $j \in J$ we have

$$K_2^j(x,x') = \langle N^j(x), N^j(x')\rangle_{\mathcal{L}^2(T_1^j)}$$

which makes use of a lower kernel $K_1^j$, with $T_1^j = \{s_j\}$ as before (fixed, i.e. chosen already at a previous step) and we define each $N^j = A_j(N^j_{\text{avg}} + B_j)$ (as in the earlier

[4]To make multiple planes possible at this lower level, one could imagine another layer $X_0$ further below.

disussion, but without sigmoid), where $N^j_{\text{avg}}$ is a weighted average with transformation set $H^j = [-n, n] \times [-n, n]$ and measure

$$\mu^j(h) = W^{jk}_h$$

(the $W^{jk}_h$ being those of the earlier CNN).

Fix the notation $[x']_{h,j} = K^j_1(h(x'), s_j)$ for any $x' \in X_2$. We will assume the $h$'th pixel of the $j$'th plane at the lower layer of our original CNN is the same as $[x']_{h,j} = K^j_1(h(x'), s_j)$, and then show that in the extended CBCL model we are setting up, this correspondence holds at the next higher layer as well. Namely, the $h$'th pixel of the $k$'th plane at the next layer up of the CNN will agree with $K^k_2(x', t)$ of that CBCL NN. This implies, by induction, that the two NN's produce the same global output given the same global input.

We now choose any $t \in X_2$ as the template at the higher level. This can depend on $k$ but we do not burden the notation, since $k$ is kept fixed. For each $j \in J$, let

$$c_j(t, s_j) := \sum_{h \in H^j} \mu^j(h) \cdot [t]_{h,j},$$

and let $B_j$ be such that $c_j(t, s_j) + B_j > 0$.

Then,

$$K^j_2(x', t) = \left(A^2_j(c_j(s_j, t) + B_j)\right) \left( \sum_{h \in H^j} W^{jk}_h \cdot [x']_{h,j} + B_j \right).$$

Since $c_j(t, s_j) + B_j > 0$, we let $A = \sqrt{1/c_j(t, s_j)}$ so that

$$K^j_2(x', t) = \sum_{h \in H^j} W^{jk}_h \cdot [x']_{h,j} + B_j.$$

Now, applying (3.7.2) we obtain

$$K_2(x', t) = \sum_{j \in J, h \in H^j} W^{jk}_h \cdot [x']_{h,j} + \sum_{j \in J} B_j.$$

By assuming the CNN bias was $B = \sum_{j \in J} B_j$ we have that $K_2(x', t)$ is in fact the $h$'th pixel of the $k$'th plane of the CNN (recall we could choose $B$ arbitrarily since it can be corrected at the next level, when the squashing function is also applied).

The convolution operation in a CNN (before activation) is therefore a weighted average (as in the CBCL model) which is then composed with the activation function. The bottom

layers of a CNN using convolution, thus fit the *extended CBCL framework* (see Section 3.6). This correspondence and the fact that particular pixels of a feature map correspond to values of a kernel $K_\ell$ on certain sub-images and templates will be addressed again on the level of NN's in Section 3.12, where the underlying neural networks are discussed.

The argument for max and sub-sampling[5] operations is similar.

**Summary** We have seen that templates are not explicit in CNN's, but are bundled into the definition of a particular feature map. More precisely, the $i$'th pixel of the $j$'th feature map evaluated on $x'$ encodes $K^j_{\ell-1}(h(\text{sub}_i x), s)$. Moreover, while the CBCL model uses multiple templates for a single transformation set, CNN's do not: they essentially use a single template per feature map. Different feature maps of a CNN meanwhile correspond to different kernels in the extended CBCL model. The activation function of a CNN amounts to an additional layer in a CBCL.

## 3.8 Example of convolution neural net to CBCL conversion

**Example 3.8.1.** As an example of the above conversion, suppose we are given a CNN which takes as input $13 \times 13$ grayscale images and outputs to its top fully connected layers 10 different (single pixel) features. We look at each feature separately and show that it may be computed by an extended CBCL NN. Note that CBCL NN's (extended or not) compute a single global (top gate) output, so we need[6] 10 of these NN's to produce these 10 features. We will do the conversion for the *first* feature.

Suppose there are 3 lower layers (before the fully connected part):

- each gate of the bottom layer computes 8 convolution-based feature maps (with weight matrices $W^1, \ldots, W^8$) on $3 \times 3$ patches and squashes each with a logistic function.
- each gate of the hidden layer computes a max over a $5 \times 5$ grid of the bottom layer output and thus handles a $7 \times 7$ receptive field.
- the single "top" gate (of the lower part of the CNN, for one feature) computes a convolution-based feature map (with weight tensor $U$) and squashes it with a logistic function; it thus handles a $13 \times 13$ receptive field.

Figure 3.1 illustrates how the size of these receptive fields was determined. Let the over-all input to the CNN be $x$, the output of the next layer (i.e. input to the hidden layer) be

[5] This is a special case of convolution with many weights 0 and others 1.

[6] Actually, one could use a portion of a *single* CBCL NN (with multiple outputs upwards) but technically this would not be a CBCL NN; extended or not, these require a single top gate.

$y$, the output of the hidden layer (i.e. input to the top layer) be $z$, and the output of the top layer $w$. Then, in the convolution + squashing operations we compute

$$y_i^k = \sigma\left(\sum_{h\in H} W_h^k \cdot [\mathrm{sub}_i(x)]_h + B\right), \text{ for } k = 1,\ldots,8 \text{ and } i \in [-5,5]\times[-5,5],$$

and

$$w = \sigma\left(\sum_{h\in H'', j\in J} U_h^{j1} \cdot [\mathrm{sub}_{(0,0)}(z^j)]_h + B''\right).$$

Here $H = [-1]\times[1]$ (a centred $3\times 3$ grid) and $H'' = [-3]\times[3]$ (a centred $7\times 7$ grid). In each case the transformation $h = (a,b) \in [-n]\times[n]$ picks out the pixel which lies $a$ units to the right of the center and $b$ units above the center ($a$ and $b$ may be negative) of whatever image it is acting on.

In the case of the top layer computation, there is only one $7\times 7$ sub-image of the $13\times 13$ global image which gets used. This is the one centred at $(0,0)$. There is also just one pixel of output (for the first feature, which we have chosen to work with). In the computation of $y_i^k$ we let $i$ range across a central $11\times 11$ subimage of $x$; thus, each of the planes, $y^1,\ldots,y^8$, is of size $11\times 11$.

For consistency with the standard notation (used for weight tensors of a CNN), $j1$ appears in superscript on $U$ in the above formula. The 1 indicates that we are computing the first feature and the $j$ indicates the plane to which weights will be applied (by the matrix $U^{j1}$).

The hidden layer, on the other hand, computes,

$$z_i^k = \sigma\left(\max_{h\in H'} [\mathrm{sub}_i(y^k)]_h + B'\right), \text{ for } i \in [-3,3]\times[-3,3],$$

where $H' = [-2,2]\times[-2,2]$ (a centred $5\times 5$ grid).

**Remark 3.8.1** (**Size of sub-images).** Note that $\mathrm{sub}_i$ is not the same in each context in which it is used. The *size* of the sub-image that it produces varies and is not noted explicitly. It should however be clear from the context. When we see $K_3(\mathrm{sub}_i x, t')$ we know that $\mathrm{sub}_i$ must produce $7\times 7$ images because $K_3$ is defined on $X_3$.

**Remark 3.8.2** (**Variable names).** As mentioned, the top layer convolution operation uses $z$ as input, and the hidden layer uses $y$ as input. This is a deviation from the variable names used at the beginning of this section, where $y$ named an intermediate output which still needed to be squashed before being sent on to serve as input for the next layer. Now (to avoid excess notation) these variables denote outputs *after squashing.*

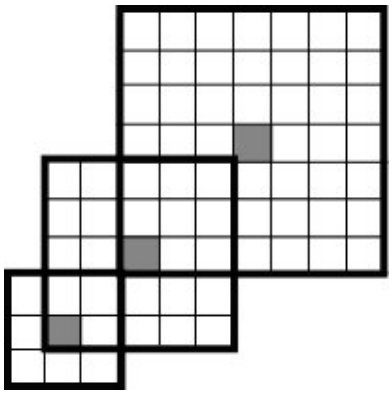


Figure 3.1: The three patches of sizes $3 \times 3$, $5 \times 5$ and $7 \times 7$, with their centers shaded. They correspond to the patches used by $y$, $z$ and $w$ respectively. They are placed with the center of each patch at the corner of the next one to illustrate the maximum receptive field that will be covered (in this example, patch size happens to increase with the layer of the CNN). By labeling the pixels of the patches respectively with coordinates $[-1, 1] \times [-1, 1]$, $[-2, 2] \times [-2, 2]$ and $[-3, 3] \times [-3, 3]$ it is clear that the maximum over-all receptive field can be labeled with $[-6, 6] \times [-6, -6]$ (because $1 + 2 + 3 = 6$), i.e. it is of size $13 \times 13$. Similarly, one works out that the lower layer gates handle receptive fields of size $7 \times 7$ (hidden layer) and $3 \times 3$ (bottom layer).

The bottom and top convolution layers are linear filters followed by squashing functions. In fact, this example is inspired by the lower layers of the NN in Amit [1] (where a discrete thresholding function is used instead of the sigmoid), but can equally well be cast as a CNN example, as we do here.

To implement the above CNN with a CBCL NN, we use the following spaces:

$$
\begin{aligned}
X_0 &= 1 \times 1 \text{ images} \\
X_1 &= 3 \times 3 \text{ images} \\
X_2 &= X_1 \\
X_3 &= 7 \times 7 \text{ images} \\
X_4 &= 13 \times 13 \text{ images} \\
X_5 &= X_4.
\end{aligned}
$$

We start with a simple inner product $K_0$ on $X_0$ (computing the product of single pixel values) and then build up kernels recursively until obtaining $K_5(x, t)$ defined for $x, t \in X_5$ with $t$ fixed and $x$ arbitrary. There are no bias terms in this example. The general scheme is as follows:

First, there will be 8 kernels derived on $X_2 = X_1$ corresponding to the 8 feature vectors on the second layer of the CNN. This is done in two steps: first deriving kernels $K_1^1, \ldots, K_1^8$ on $X_1$ from $K_0$ by weighted averages (with weights given by $W$) and then using the method explained in "Problems" to (respectively) derive kernels $K_2^1 = \sigma(K_1^1), \ldots, K_2^8 = \sigma(K_1^8)$ on $X_2 = X_1$ from these.

Then, we derive 8 kernels on $X_3$ from the previous ones, using a max neural response (with set of transformations $[-2, 2] \times [-2, 2]$). There is no cross-interaction: for each $j = 1, \ldots, 8$ the kernel $K_3^j$ is derived from $K_2^j$.

Next, we derive a single kernel $K_4$ on $X_4$ as a sum of 8 different kernels $K_4^1, \ldots, K_4^8$, each derived via a weighted average (with weights for the $j$'th one given by $U^{j1}$). Finally, we use the method in "Problems" to derive a kernel $K_5 = \sigma(K_4)$ on $X_5 = X_4$ from $K_4$.

## 3.9 Lifting an abstract neural map

Suppose one has a neural response (i.e. map, in general) $N_\ell : X_\ell \to \mathcal{F}_{\ell-1}$ which maps into some Hilbert space $\mathcal{F}_{\ell-1}$ of functions on $X_\ell$ and the inner product of $\mathcal{F}_{\ell-1}$ is then pulled back via $N_\ell$ to obtain a kernel $K_\ell$ on $X_\ell$:

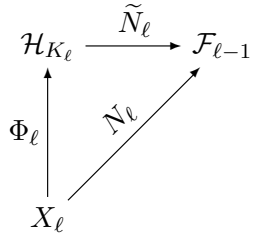
$$K_\ell(\cdot, \cdot) = \langle N_\ell(\cdot), N_\ell(\cdot) \rangle_{\mathcal{F}_{\ell-1}}.$$

In the CBCL model $\mathcal{F}_{\ell-1} = \mathcal{L}^2(T_{\ell-1})$. Then $N_\ell$ may in fact be lifted to a map

$$\widetilde{N}_\ell \colon \mathcal{H}_{K_\ell} \to \mathcal{F}_{\ell-1}$$

so the diagram below commutes. This observation was made in joint work with A. Wibisono for the case $\mathcal{F}_{\ell-1} = \mathcal{H}_{K_{\ell-1}}$. The argument for that case goes through in the general case and is reproduced below.

Here $\Phi_\ell \colon X_\ell \to \mathcal{H}_{K_\ell}$ is the feature map that sends $x \in X_\ell$ to $K_{\ell,x} \in \mathcal{H}_{K_\ell}$.




This is accomplished by setting $\widetilde{N}_\ell(K_{\ell,x}) = N_\ell(x)$ for each $x \in X_\ell$, then extending by linearity and continuity. The following lemma shows that this construction of $\widetilde{N}_\ell$ is well-defined. To see this construction more concretely, suppose $N_\ell$ is a max neural response:

$$N_\ell(x)(t) = \langle K_{\ell-1,h_t(x)}, K_{\ell-1,t} \rangle_{\mathcal{H}_{K_{\ell-1}}},$$

where $\forall t \in T_{\ell-1}$, $h_t$ is the element $h \in H_\ell$ which maximizes $\langle K_{\ell-1,h_t(x)}, K_{\ell-1,t}\rangle_{\mathcal{H}_{K_{\ell-1}}}$. Then,

$$\widetilde{N}_\ell(K_{\ell,x})(t) = \langle K_{\ell-1,h_t(x)}, K_{\ell-1,t}\rangle_{\mathcal{H}_{K_{\ell-1}}}, \forall t \in T_{\ell-1}.$$

**Lemma 3.9.1** (Fraser, Wibisono)**.** *Given $N_\ell$ as above, it follows that*

$$\sum_{i=1}^m a_i K_{\ell,x_i} = 0 \implies \sum_{i=1}^m a_i N_\ell(x_i) = 0$$

*and hence one may extend the basic definition of $\widetilde{N}_\ell$ on the $K_{\ell,x}$'s by linearity to a map*

$$\widetilde{N}'_\ell \colon span\{K_{\ell,x} \mid x \in X_\ell\} \to \mathcal{F}_{\ell-1}.$$

*Moreover, $\widetilde{N}'_\ell$ is in fact a linear isometry and hence may be extended by continuity to a map*

$$\widetilde{N}_\ell \colon \mathcal{H}_{K_\ell} \to \mathcal{F}_{\ell-1}.$$

*Proof.* We have

$$\begin{aligned}
\sum_{i=1}^m a_i K_{\ell,x_i} = 0 \;&\Rightarrow \sum_{i=1}^m a_i K_\ell(x_i, x) = 0, && \forall x \in X_\ell \\
&\Rightarrow \sum_{i=1}^m a_i \langle N_\ell(x_i), N_\ell(x)\rangle_{\mathcal{F}_{\ell-1}} = 0, && \forall x \in X_\ell \\
&\Rightarrow \Big\langle \sum_{i=1}^m a_i N_\ell(x_i), y \Big\rangle_{\mathcal{F}_{\ell-1}} = 0, && \forall y \in N_\ell(X_\ell) \\
&\Rightarrow \Big\langle \sum_{i=1}^m a_i N_\ell(x_i), y \Big\rangle_{\mathcal{F}_{\ell-1}} = 0, && \forall y \in \mathrm{span} N_\ell(X_\ell),
\end{aligned}$$

which implies that

$$\sum_{i=1}^m a_i N_\ell(x_i) \in \Big(\mathrm{span} N_\ell(X_\ell)\Big)^\perp.$$

Since clearly $\sum_{i=1}^m a_i N_\ell(x_i)$ lies in the span of $N_\ell(X_\ell)$, we conclude that $\sum_{i=1}^m a_i N_\ell(x_i) = 0$, as desired. This allows us to extend $\widetilde{N}'_\ell$ by linearity to a well-defined map

$$\widetilde{N}'_\ell \colon \mathrm{span}\{K_{\ell,x} \mid x \in X_\ell\} \to \mathcal{F}_{\ell-1}.$$

By definition, $\widetilde{N}'_\ell$ is linear and the inner product defined on the pre-Hilbertian space $\mathrm{span}\{K_{\ell,x} \mid x \in X_\ell\}$ within $\mathcal{H}_{K_\ell}$ is the same as that obtained by pullback via $\widetilde{N}'_\ell$ of $\langle\cdot,\cdot\rangle_{\mathcal{F}_{\ell-1}}$. $\widetilde{N}'_\ell$ is therefore an isometry. Indeed, given $\alpha, \beta \in \mathrm{span}\{K_{\ell,x} \mid x \in X_\ell\}$, we have

$$\begin{aligned}
\|\alpha - \beta\|^2_{\mathcal{H}_{K_\ell}} &= \langle \alpha - \beta, \alpha - \beta\rangle_{\mathcal{H}_{K_\ell}} \\
&= \langle \widetilde{N}'_\ell(\alpha - \beta), \widetilde{N}'_\ell(\alpha - \beta)\rangle_{\mathcal{F}_{\ell-1}} \\
&= \langle \widetilde{N}'_\ell(\alpha) - \widetilde{N}'_\ell(\beta), \widetilde{N}'_\ell(\alpha) - \widetilde{N}'_\ell(\beta)\rangle_{\mathcal{F}_{\ell-1}} \\
&= \|\widetilde{N}'_\ell(\alpha) - \widetilde{N}'_\ell(\beta)\|^2_{\mathcal{F}_{\ell-1}},
\end{aligned}$$

so $\widetilde{N}'_\ell$ takes Cauchy sequences to Cauchy sequences. In particular, if two sequences converge to the same limit in $\mathcal{H}_{K_\ell}$ then their images under $\widetilde{N}'_\ell$ will also converge and share a limit. Therefore, given $\alpha \in \mathcal{H}_{K_\ell}$ we may take any $(\alpha_n)_{n=1}^\infty \subset \mathrm{span}\{K_{\ell,x} \mid x \in X_\ell\}$ such that $\alpha = \lim_{n\to\infty} \alpha_n$ in $\mathcal{H}_{K_\ell}$, and define $\widetilde{N}_\ell(\alpha) = \lim_{n\to\infty} \widetilde{N}'_\ell(n)$. □

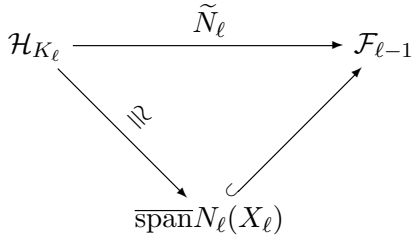
In the above proof we also proved the following.

**Lemma 3.9.2.** *Such a definition of $\widetilde{N}_\ell$ implies that $N_\ell = \widetilde{N}_\ell \circ \Phi_\ell$, and hence that the inner product on $\mathcal{H}_{K_\ell}$ is a pullback of that on $\mathcal{F}_{\ell-1}$ via $\widetilde{N}_\ell$.*

### 3.9.1 Injectivity of $\widetilde{N}_\ell$

We remark that this in fact implies $\widetilde{N}_\ell$ is injective (by the last statement in Proposition 3.4.3). This occurs once we can define $\widetilde{N}_\ell$ as above, irrespective of whether $N_\ell$ is injective. It is a consequence of the way in which the feature map kills off the degenerate part of a kernel in lifting to the Hilbert space (see the discussion in Section 3.4.1).

Therefore, $\widetilde{N}_\ell$ is an isomorphism of Hilbert spaces between $\mathcal{H}_{K_\ell}$ and $\widetilde{N}_\ell(\mathcal{H}_{K_\ell}) = \overline{\mathrm{span}} N_\ell(X_\ell)$, and we have the following commutative diagram.




## 3.10 Templates of the first and second kind

Poggio et al sometimes refer to **templates of the second kind**. We define this notion below. To distinguish it from the notion of templates we have discussed up to now, they use the term **templates of the first kind** for the latter.

**Definition 3.10.1** (Templates of the Second Kind)**.** A set of templates of the second kind is a set $\mathcal{T}_\ell \subset \mathcal{H}_{K_\ell}$. It implies some modifications to the basic framework described so far. Namely, for $\ell = 2, \ldots, d$ the neural responses are taken to be of the form

$$N_\ell \colon X_\ell \to \mathcal{L}^2(\mathcal{T}_{\ell-1}, \nu_{\ell-1})$$

and the recurrence is

$$K_\ell(x, x') = \langle N_\ell(x), N_\ell(x') \rangle_{\mathcal{L}^2(\mathcal{T}_{\ell-1}, \nu_{\ell-1})},$$

where $\nu_{\ell-1}$ is a measure on $\mathcal{T}_{\ell-1}$.

**Remark 3.10.2.** To distinguish the two kinds of templates, we will always use calligraphic letters, such as $\mathcal{T}$, for templates of the second kind, and roman letters, such as $T$ for templates of the first kind.

Note that both $N_{\max}$ and $N_{\text{avg}}$ can be viewed as neural responses of this type as follows:

$$N_{\max}(x)(\tau) = \max_{h \in H} \langle K_{\ell-1,h(x)}, \tau \rangle_{K_{\ell-1}}$$

and

$$N_{\text{avg}}(x)(\tau) = \sum_{h \in H} \mu^H_{\ell-1}(h) \langle K_{\ell-1,h(x)}, \tau \rangle_{K_{\ell-1}}$$

where $\tau \in \mathcal{T} \subset \mathcal{H}_{K_{\ell-1}}$ is a template of the second kind and $x \in X_\ell$. This is compatible with the usual definition of these neural responses since in general we have $K(y,z) = \langle K_y, K_z \rangle_K$ (see Section 3.4.1).

### 3.10.1 Using templates of the second kind with linear neural responses

Consider now an arbitrary neural response $N : X_2 \to \mathcal{L}^2(T, \nu)$ which satisfies Axiom 3.5.1 and is *linear* (so the linear base neural response $N_{\text{base}} : X_2 \to \mathcal{H}_{K_1}$ is well-defined) we can then define $N' : X_2 \to \mathcal{L}^2(\mathcal{T}', \nu')$, for any set of templates of the second kind, $\mathcal{T}' \subset \mathcal{H}_{K_1}$ with measure $\nu'$, as follows:

$$N'(x)(\tau) = \langle N_{\text{base}}(x), \tau \rangle_{K_1}, \, \forall \tau \in \mathcal{T}'.$$

This is an extension - to templates of the *second kind* - of the formula in Axiom 3.5.1; that formula says how to evaluate neural responses in terms of base neural responses, in the case of templates of the *first kind*.

In particular, if $\mathcal{T}' = \{\Phi_1(t) : t \in T\}$ and we let $\nu'$ denote the pushforward of $\nu$ via $\Phi$ (defined on $\mathcal{T}'$ by $\nu'(\Phi_1(t)) := \nu(t)$, $\forall t \in T$), then:

$$\langle N(\cdot), N(\cdot) \rangle_{\mathcal{L}^2(T,\nu)} = \langle N'(\cdot), N'(\cdot) \rangle_{\mathcal{L}^2(\mathcal{T}',\nu')} \tag{3.10.1}$$

### 3.10.2 Caution regarding templates of the second kind for non-linear neural responses

Suppose $N : X_2 \to \mathcal{L}^2(T, \nu)$ satisfying Axiom 3.5.1 is possibly *non-linear*. We would like to define evaluation of $N$ for templates of the second kind in a *compatible* manner, i.e., by a formula which reduces to (3.10.1) when $N$ is linear. Specifically, $\rho_{\mathcal{T}}(N_{\text{base}})(x_2)$ (as a functional on $\mathcal{T}$) should map $\tau = \sum_{i=1}^m a_i K_{1,x_i} \in \mathcal{T}$ to

$$\sum_{i=1}^m a_i \langle N_{\text{base}}(x_2)(x_i), K_{1,x_i} \rangle_{K_1},$$

where $x_1, \ldots, x_m \in X_1$ are chosen so $K_{1,x_1}, \ldots, K_{m,x_m}$ form a basis for $\mathcal{H}_{K_1}$. However this in general will depend on the choice of $x_1, \ldots, x_m \in X_1$ if $N$ is nonlinear and so is not well-defined.

Therefore, either one must restrict oneself to using templates of the second kind **only with linear neural responses** or one must specify a **particular maximally independent subset** of $\Phi_1(X_1)$, i.e. specify $x_1, \ldots, x_m \in X_1$ such that $K_{1,x_1}, \ldots, K_{m,x_m}$ form a basis for $\mathcal{H}_{K_1}$ (then the above calculation will be well-defined with respect to that subset). Alternatively, if one considers $\mathcal{T} = \Phi_1(T)$ for some template set $T$ of the first kind then this issue does not arise; one defines

$$\rho_{\mathcal{T}}(N_{\text{base}})(x_2)(\Phi_1(t)) := \rho_T(N_{\text{base}})(x_2)(t) = \langle N_{\text{base}}(x_2)(t), K_{1,t} \rangle_{K_1}.$$

## 3.11 Closer look at the average neural response

We now take a closer look at the CBCL model in the special case of the average neural response.

### 3.11.1 Skipping a layer

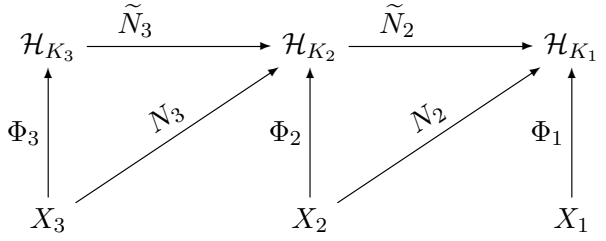
Let $N_\ell$ be an average neural response. It turns out that the composed map

$$N_\ell \circ \widetilde{N}_{\ell-1} \circ \cdots \circ \widetilde{N}_1$$

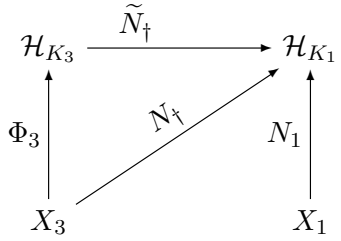
is again an average neural response, with the natural transformation set, and that it produces the same kernel on the top layer *if one allows templates of the second kind*. This is therefore *not* a true collapsing of the layers within the original CBCL model. It was first observed in joint work of the author with A. Wibisono, and the exposition given here follows that work.

One of the main results of this thesis (Proposition 4.5.1) is a strengthening of that initial result (allowing us to choose the templates), which *does* produce true collapsing within the CBCL model – in fact for arbitrary linear neural responses.

We perform the computation for a 3-layer architecture; the general case will then follow by induction. Assume the standard setting and notation, so we have the following diagram.



Let $H = \{h \circ h' \mid h \in H_2, h' \in H_3\}$ (which are transformations from $X_3$ to $X_1$) and define the measure $\mu^H$ on $H$ to be the pushforward measure of $\mu_2^H \times \mu_3^H$ on $H_2 \times H_3$ under the map $(h, h') \mapsto h \circ h'$. Consider the following diagram




where $N_\dagger$ is the average neural response defined by the transformations in $H$. Given $x \in X_3$, we can compute

$$\begin{aligned}
(\widetilde{N}_2 \circ N_3)(x) &= \widetilde{N}_2\Big(\sum_{h' \in H_3} K_{2,h'(x)} \mu_3^H(h')\Big) \\
&= \sum_{h' \in H_3} \widetilde{N}_2(K_{2,h'(x)}) \mu_3^H(h') \\
&= \sum_{h' \in H_3} N_2(h'(x)) \mu_3^H(h') \\
&= \sum_{h' \in H_3} \sum_{h \in H_2} K_{1,h \circ h'(x)} \mu_2^H(h) \mu_3^H(h') \\
&= \sum_{g \in H} K_{1,g(x)} \mu^H(g) \\
&= N_\dagger(x).
\end{aligned}$$

We now wish to compare the kernel

$$K_3 = N_3^* \langle \cdot, \cdot \rangle_{\mathcal{L}^2(T_2)}$$

with the kernel one would obtain via $N_\dagger$,

$$K_3' := N_\dagger^* \langle \cdot, \cdot \rangle_{\mathcal{L}^2(T_1)}.$$

However, we will at this point need to allow template sets of the second kind: $\mathcal{T} \subset \mathcal{H}_K$. (see Section 3.10). Specifically, we will be comparing kernels derived by the following two architectures:

**Architecture 1.** Three-layers: i.e. spaces $X_1, X_2, X_3$, transformation sets $(H_1, \mu_1)$ and $(H_2, \mu_2)$, initial kernel $K_1$, template sets of the second kind $(\mathcal{T}_1, \nu_1)$, $(\mathcal{T}_2, \nu_2)$, average neural responses $N_2$, $N_3$, and derived kernels $K_2$, $K_3$.

**Architecture 2.** Two-layers: space $X_1$ as the bottom layer and $X_3$ as the top layer (no hidden layer). The transformation set is given by

$$H = \{h_2 \circ h_3 \mid h_2 \in H_2, h_3 \in H_3\},$$

and the measure $\mu$ on $H$ is given by the pushforward measure from $H_2 \times H_3$ via the map $(h_2, h_3) \mapsto h_2 \circ h_3$. The initial kernel on $X_1$ is $K_1$. The template set is $\mathcal{T} = F(\mathcal{T}_2) \subset \mathcal{H}_{K_1}$, where $F\colon \mathcal{H}_{K_2} \to \mathcal{H}_{K_1}$ is given by

$$F(\tau) = \sum_{\tau' \in \mathcal{T}_1} \widetilde{N}_2(\tau)(\tau')\, \tau'\, \nu_1(\tau'), \tag{3.11.1}$$

with the measure $\nu = F_*\nu_2$. The neural response $N_\dagger\colon X_3 \to \mathcal{L}^2_\nu(\mathcal{T})$ is the average neural response

$$N_\dagger(x)(\tau) = \sum_{g \in H} \langle K_{1,g(x)},\, \tau \rangle_{\mathcal{H}_{K_1}}\, \mu(g),$$

for $x \in X_3, \tau \in \mathcal{T}$and the derived kernel $K_\dagger\colon X_3 \times X_3 \to \mathbb{R}$ is given by

$$K_\dagger(x, x') = \langle N_\dagger(x),\, N_\dagger(x') \rangle_{\mathcal{L}^2_\nu(\mathcal{T})},$$

for $x, x' \in X_3$.

**Remark 3.11.1.** Note that $\mathcal{T} = F(\mathcal{T}_2) \subset \text{span}\mathcal{T}_1$.

We now prove:

**Proposition 3.11.2** (Fraser and Wibisono)**.** *For any $x \in X_3$ and $\tau \in \mathcal{T}_2$, we have $N_3(x)(\tau) = N_\dagger(x)(F(\tau))$, and therefore,*

$$K_3(x, x') = K_\dagger(x, x') \quad \textit{for all } x, x \in X_3.$$

*Proof.* Let $x \in X_3$ and $\tau \in \mathcal{T}_2$. We compute

$$\begin{aligned}
N_3(x)(\tau) &= \sum_{h_3 \in H_3} \langle K_{2,h_3(x)},\, \tau \rangle_{\mathcal{H}_{K_2}}\, \mu_3(h_3) && (3.11.1)\\
&= \sum_{h_3 \in H_3} \langle N_2(h_3(x)),\, \widetilde{N}_2(\tau) \rangle_{\mathcal{L}^2_{\nu_1}(\mathcal{T}_1)}\, \mu_3(h_3) && (3.11.2)\\
&= \sum_{h_3 \in H_3} \sum_{\tau' \in \mathcal{T}_1} N_2(h_3(x))(\tau')\, \widetilde{N}_2(\tau)(\tau')\, \nu_1(\tau')\, \mu_3(h_3) && (3.11.3)\\
&= \sum_{h_3 \in H_3} \sum_{\tau' \in \mathcal{T}_1} \sum_{h_2 \in H_2} \langle K_{1,h_2 \circ h_3(x)},\, \tau' \rangle_{\mathcal{H}_{K_1}}\, \widetilde{N}_2(\tau)(\tau')\, \mu_2(h_2)\, \nu_1(\tau')\, \mu_3(h_3) &&\\
&= \sum_{g \in H} \sum_{\tau' \in \mathcal{T}_1} \langle K_{1,g(x)},\, \tau' \rangle_{\mathcal{H}_{K_1}}\, \widetilde{N}_2(\tau)(\tau')\, \mu(g)\, \nu_1(\tau') && (3.11.4)\\
&= \sum_{g \in H} \Big\langle K_{1,g(x)},\, \underbrace{\sum_{\tau' \in \mathcal{T}_1} \widetilde{N}_2(\tau)(\tau')\, \tau'\, \nu_1(\tau')}_{=F(\tau)} \Big\rangle_{\mathcal{H}_{K_1}}\, \mu(g) && (3.11.5)\\
&= N_\dagger(x)(F(\tau)). && (3.11.6)
\end{aligned}$$

Therefore, for any $x, x' \in X_3$,

$$\begin{aligned}
K_3(x,x') &= \langle N_3(x),\, N_3(x') \rangle_{\mathcal{L}^2_{\nu_2}(\mathcal{T}_2)}\\
&= \sum_{\tau \in \mathcal{T}_2} N_3(x)(\tau)\, N_3(x')(\tau)\, \nu_2(\tau)\\
&= \sum_{\tau \in \mathcal{T}_2} N_\dagger(x)(F(\tau))\, N_\dagger(x')(F(\tau))\, \nu_2(\tau)\\
&= \sum_{\widehat{\tau} \in \mathcal{T}} N_\dagger(x)(\widehat{\tau})\, N_\dagger(x')(\widehat{\tau})\, \nu(\widehat{\tau})\\
&= \langle N_\dagger(x),\, N_\dagger(x') \rangle_{\mathcal{L}^2_\nu(\mathcal{T})}\\
&= K_\dagger(x,x').
\end{aligned}$$

□

## 3.12 Underlying neural network

The framework just described can be computed by the neural network defined in the Figures below. We refer to this NN model as the **CBCL model** or **extended CBCL model** - the distinction will be made clear below.

Together, Figure 3.2 and Figure 3.3 show the *recursive step* of the computation, namely, how $K_\ell$ is obtained from $K_{\ell-1}$. Figure 3.3 is a close-up of the workings of a $t$-filter, which

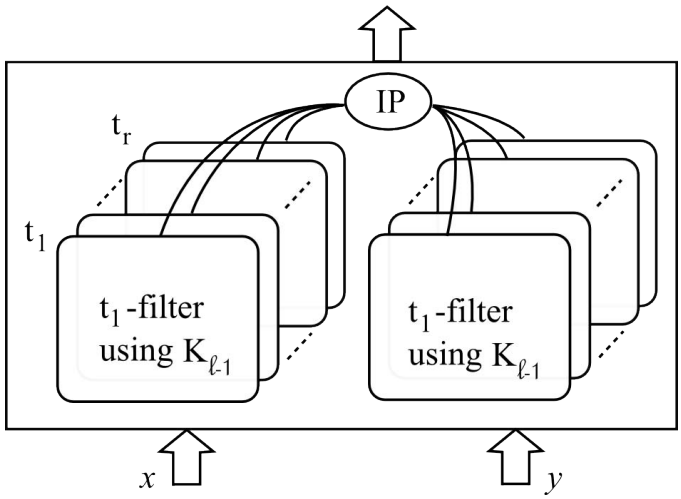



Figure 3.2: The neural network computing $K_\ell(x, y)$, for $x, y \in X_\ell$. Each rounded rectangle represents a $t_i$-filter of the kind depicted in the Figure 3.3. More details are given in Remark 3.12.1.

is a subnetwork of the neural network in Figure 3.2. The *entire neural network* is obtained by substituting a network analogous to that of Figure 3.2 (which computed $K_\ell$) to perform each of the calculations of $K_{\ell-1}$ appearing in Figure 3.3 and so on... until reaching $K_1$ which is assumed to be computed atomically (either by a single computational gate, or a fixed network with all known components).

**Remark 3.12.1** (**on the Figures).** We have used rounded rectangles to represent $t$-filters, sharp-cornered rectangles, i.e. boxes, to represent nodes computing a kernel ($K_\ell$ or $K_{\ell-1}$) and finally, ovals/circles for all other computational nodes. In Figure 3.2, the gate marked **IP** computes the inner product in $\mathcal{L}^2(T_{\ell-1}, \nu_{\ell-1})$:

$$((z_1, \ldots, z_r), (w_1, \ldots, w_r)) \mapsto \sum_{i=1}^{r} z_i w_i \nu(t_i),$$

where $T_{\ell-1} = \{t_1, \ldots, t_r)$. In Figure 3.3, the gate marked $\mu$**-sum over all h** is a weighted average. Supposing $H_{\ell-1} = \{h_1, \ldots, h_N\}$, this gate computes

$$(z_1, \ldots, z_N) \mapsto \sum_{i=1}^{N} \mu_{\ell-1}^{H}(h_i)(z_i).$$

If the $t$-filter were based on a different pooling, namely the max operation, then we would

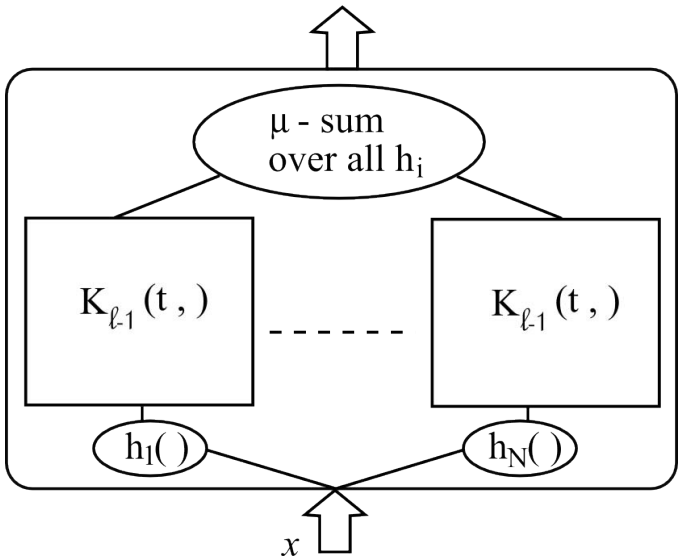



Figure 3.3: A $t$-filter, as used in the above neural network; this one pools by weighted average, pooling by max would be analogous, or alternatively any other neural response could be used. More details are given in Remark 3.12.1. Here we assume $t \in X_{\ell-1}$ is a template and $\mu$ is the measure on $H = \{h_1, \ldots, h_N\} \subset \text{Maps}(X_2, X_1)$. For each $i \in [N]$ the input $x \in X_\ell$ is converted into $h_i(x) \in X_{\ell-1}$ before being input into a box computing $K_{\ell-1}(t, )$. This box is of the same type as that of Figure 3.2 which computed $K_\ell(\cdot, \cdot)$. **Note:** a $t$-filter is thus defined by the pooling function (with $H$) and the kernel $K_{\ell-1}$.

have, instead of this gate, one marked **max over all h**, which computes

$$(z_1, \ldots, z_N) \mapsto \max_{i \in [N]} z_i.$$

In order to implement an arbitrary neural response, the subnetworks which compute $K_{\ell-1}(h(x), \cdot)$ in Figure 3.3 would be possibly indexed by some set other than $H$. See Section 3.12.2.

**Definition 3.12.2** (Extended CBCL model vs. CBCL model)**.** These are distinguished by two aspects.

1. The extended CBCL model allows $t_i$**- and** $t_j$**-filters to be different**, i.e. to use different kernels $K_{\ell-1}$ and pooling functions (i.e. neural responses) for $i \neq j$. The CBCL model requires that all filters on a given level use the same kernel. In either model, a *given* $t_i$-filter (which appears twice) must be the same in both its occurrences.

2. The extended CBCL model allows arbitrary **neural responses satisfying Axiom 3.5.1**. The CBCL model restricts the neural responses to be either $N^{\max}$ or $N^{\text{avg}}$.

### 3.12.1 Convolution Neural Networks

The lower layers of a convolution neural network alternate pooling by weighted average and sub-sampling (which amounts to pooling with some weights zero) or sometimes max'ing. The final, upper layers are fully connected, using linear neural responses. The discussion in Section 3.7 shows that a convolution neural network may be viewed as a special case of an extended CBCL network.

### 3.12.2 General $t$-filters

Suppose $t \in X_{\ell-1}$ is a template. Then a $t$-filter of the CBCL or extended CBCL model (there is no difference as far as individual $t$-filters are concerned) computes, for any $x \in X_\ell$,

$$\langle N_{\text{base}}(x)(t), K_{\ell-1,t}\rangle_{\mathcal{H}_{K_{\ell-1}}}$$

where

$$N_{\text{base}} : X_\ell \to \text{Maps}(X_{\ell-1} \to \mathcal{H}_{K_{\ell-1}})$$

is a particular base neural response and $K_{\ell-1}$ a kernel on $X_{\ell-1}$, both of which have been fixed in advance to define that $t$-filter.

Standard $t$-filters – those with $N^{\text{avg}}$ or $N^{\max}$ – make use of a transformation set $H$ to index computations of the type $K_{\ell-1}(h(x), t)$ which are then pooled. A similar principle can be used to construct a general $t$-filter, since $\mathcal{H}_{K_{\ell-1}}$ is finite-dimensional ($X_{\ell-1}$ is finite in all NN applications). Let $x_1, \ldots, x_n \in X_{\ell-1}$ be such that $K_{\ell-1,x_1}, \ldots, K_{\ell-1,x_n}$ form a basis for $\mathcal{H}_{K_{\ell-1}}$. Then to each $x \in X_\ell$ we may associate $a_1(x), \ldots, a_n(x)$ such that

$$N_{\text{base}}(x)(t) = \sum_{i\in[n]} a_i(x) K_{\ell-1,x_i}.$$

The $a_i(x)$ of course depend on $t$ as well, but this is fixed for the given filter. Define $n$ computational gates, where the $i$'th one computes $a_i(x)K_{\ell-1}(x_i, t)$. Pool the outputs of these $n$ gates at a higher level gate by summing them. This produces

$$\begin{aligned}\sum_{i\in[n]} a_i(x) K_{\ell-1}(x_i, t) &= \sum_{i\in[n]} a_i(x)\langle K_{\ell-1,x_i}, K_{\ell-1,t}\rangle_{\mathcal{H}_{K_{\ell-1}}} \\ &= \langle N_{\text{base}}(x)(t), K_{\ell-1,t}\rangle_{\mathcal{H}_{K_{\ell-1}}}\end{aligned}$$

which is exactly what this $t$-filter should compute. Thus every $N_{\text{base}}$ – not just $N^{\text{avg}}$ or $N^{\max}$ – is in fact a pooling.

**Caveat.** This is not meant as a practical solution. Rather, its purpose is to allow us to formally define a class of NN's which correspond to the architectures of the (extended) CBCL model. The algebraic complexity of computing the functions $a_i(\cdot)$ (and also $h_i(\cdot)$ in the definition below) must be addressed before comparing instances of this class. Note that computing $h(x)$ is usually free in nature, as neurons are hardwired to collect data from the appropriate sub-patch of the visual field.

**Definition 3.12.3** (General $t$-filter)**.** We define a **general $t$-filter** as an NN whose top (global output) gate pools by sum or by max the outputs of $n$ lower gates, each of which computes $a_i(x)K_{\ell-1}(h_i(x), t)$ for an $i \in [n]$, where $\forall x \in X_\ell, \forall i \in [n]$ $a_i(x) \in \mathbb{R}$ and $h_i(x) \in X_{\ell-1}$.

The computation above shows that by allowing Figure 3.3 to be a general $t$-filter, the resulting class of extended CBCL NN's is broad enough to include an instance for each extended CBCL architecture. In other words, for each mathematicval hierarchy in the framework there is an NN which computes it.

**Definition 3.12.4.** Given an NN of the CBCL or extended CBCL model, we refer to the gates which appear explicitly in Figure 3.2 together with those in Figure 3.3 as forming a **stage**.

One stage thus corresponds to a transition between spaces $X_\ell$ and $X_{\ell-1}$ in the mathematical framework. Therefore a $d-1$-stage NN corresponds to a $d$-layer architecture of the framework.

# Chapter 4

# Decomposing and Simplifying the Hierarchy

We begin, in Section 4.1, by stating four Propositions which show the inter-relation between design aspects of neural networks used for deriving reproducing kernels. Specifically we look at how choice of neural response, choice of transformations and choice of templates are related. These results also reveal on a Hilbert space level the mechanics of this kernel construction and show that there is an inherent formal linearity in the kernels thus obtained. This observation allows us in Section 4.5 to collapse hierarchies where we have sufficient freedom of choice in the mentioned design aspects. For such hierarchies, and purely in terms of the class of kernels which the underlying NN's compute, shallow networks are equivalent to deep ones. In nature such freedom of choice is most likely not present. Even for abstract neural networks, *learning* considerations may well restrict this freedom.

## 4.1 Initial structural results

The four propositions below all assume the same two-layer CBCL architecture, with some additional assumptions. We will however, allow neural responses other than $N^{\text{avg}}$ and $N^{\max}$ when indicated. We now describe the assumptions and fix the notation.

### 4.1.1 Assumed architecture

Start with a two-layer CBCL architecture (see Definition 3.3.1): spaces $X_1$ and $X_2$, a reproducing kernel $K_1$ on $X_1$, a finite set of templates $T_1 = \{t_1, \ldots, t_m\} \subset X_1$ equipped with measure $\nu_1$ and a neural response

$$S : X_2 \to \mathcal{L}^2(T_1, \nu_1),$$

such that the reproducing kernel $K_2$ on $X_2$ is derived via $S$ (as given by Equation 3.2.1):

$$K_2(\cdot,\cdot) := \langle S(\cdot), S(\cdot)\rangle_{\mathcal{L}^2(T_1,\nu_1)}.$$

We assume that $S$ **satisfies Axiom 3.5.1** but for the time being, do not restrict $S$ further.

Note that convolution networks are special cases of extended CBCL networks and we will see how these may be handled in Section 4.6.

We assume further that $\mathcal{T}_1 = \{\Phi_1(t_i) : i \in [m]\}$ is **linearly independent** and for each $i \in [m]$, let $\tau_i = \Phi_1(t_i)$. We let $n = \dim \mathcal{H}_{K_2}$, so $n \leq m \leq \dim \mathcal{H}_{K_1}$[1].

**Additional Notation:** We use $\pi_Y$ to denote projection onto the linear span of $Y$, when $Y$ is a subset of a Hilbert space.

### 4.1.2 Propositions

First, we address replacing an arbitrary neural response by a linear one (or several linear ones - see the Corollary).

**Proposition 4.1.1.** *Assume the architecture specified in Section 4.1.1. There exists a* **linear neural response** $S' : X_2 \to \mathcal{L}^2(T_1,\nu_1)$ *such that* $K_2$ *can be derived via* $S'$*:*

$$K_2(\cdot,\cdot) := \langle S'(\cdot), S'(\cdot)\rangle_{\mathcal{L}^2(T_1,\nu_1)}.$$

*In particular,* $\langle S_{base}(x), t\rangle_{\mathcal{H}_{K_1}}$ *and* $\langle S'_{base}(x), t\rangle_{\mathcal{H}_{K_1}}$ *agree for all* $x \in X_2$ *and* $t \in T_1$*.*

**Corollary 4.1.2.** *Given an arbitrary neural response* $S : X_2 \to \mathcal{L}^2(T)$ *satisfying Axiom 3.5.1, and a partitioning of* $T$ *as* $T = T_1 \cup \ldots \cup T_r$ *where each* $\mathcal{T}_i = \Phi_1(T_i)$ *is a linearly independent set in* $\mathcal{H}_{K_1}$*, the derived kernel* $K_2$ *induced by* $S$ *on* $X_2$ *can be written as*

$$K_2(\cdot,\cdot) = K^1(\cdot,\cdot) + \ldots + K^r(\cdot,\cdot)$$

*where each* $K^i$ *is a kernel on* $X_2$ *induced by a linear base neural response*

$$N^i_{base} : X_2 \to \mathcal{H}_{K_1}.$$

*Proof of the Corollary.* This is a consequence of the fact that $\mathcal{L}^2(T,\nu)$ can be expressed as a direct sum

$$\mathcal{L}^2(T,\nu) = \mathcal{L}^2(T_1,\nu_1) \oplus \cdots \oplus \mathcal{L}^2(T_r,\nu_r),$$

where $\nu_i$ is the restriction of the measure $\nu$ to the subset $T_i \subset T$. Thus,

$$\begin{aligned} K_2(x,x') &= \langle S(x), S(x')\rangle_{\mathcal{L}^2(T)} \\ &= \langle \pi_{Y_1} \circ S(x), \pi_{Y_1} \circ S(x')\rangle_{\mathcal{L}^2(T_1)} + \cdots + \langle \pi_{Y_r} \circ S(x), \pi_{Y_r} \circ S(x')\rangle_{\mathcal{L}^2(T_r)} \end{aligned}$$

[1] If $m = \dim \mathcal{H}_{K_1}$ then $\pi_{\mathcal{T}_1}$ is the identity, so $\pi_{\mathcal{T}_1} \circ S_{\text{base}} = S_{\text{base}}$.

for any $x, x' \in X_2$, where $Y_i$ denotes the elements of $\mathcal{L}^2(T,\nu)$ which are zero on $t \in T \setminus T_i$. We then obtain the desired fact by applying Proposition 4.1.1 to each of the neural responses $\pi_{\mathcal{T}_i} \circ S : X_2 \to \mathcal{L}^2(T_i)$, $i = 1, \ldots, r$. □

Before proceeding with the next Proposition, we clarify one of the hypotheses we will be using.

**Lemma 4.1.3.** *Suppose $S : X \to \mathcal{F}(T)$, a Hilbert space, and $K_2(\cdot,\cdot) = S^*\langle\cdot,\cdot\rangle_{\mathcal{F}(T)}$. Let $\Phi_2 : X_2 \to \mathcal{H}_{K_2}$ be the associated feature map. Then*

$$\sum_{i=1}^{k} a_i \Phi_2(x_i) = 0 \Leftrightarrow \sum_{i=1}^{k} a_i S(x_i) = 0$$

*holds for any $k \in \mathbb{N}, a_1, \ldots, a_k \in \mathbb{R}, x_1, \ldots, x_k \in X_2$.*

*Proof.*

$$\begin{aligned}
\sum a_i K_{2,x_i} = 0 &\Rightarrow (\forall y \in X_2)\langle \sum a_i K_{2,x_i}, K_{2,y}\rangle_{H_{K_2}} = 0 \\
&\Rightarrow (\forall y \in X_2) \sum a_i K_2(x_i, y) = 0 \\
&\Rightarrow (\forall y \in X_2)\langle \sum a_i S(x_i), S(y)\rangle_{\mathcal{F}(T)} = 0 \\
&\Rightarrow (\forall w \in \overline{\text{span}} S(X_2))\langle \sum a_i S(x_i), w\rangle = 0 \\
&\Rightarrow \sum a_i S(x_i) = 0
\end{aligned}$$

since $\overline{\text{span}} S(X_2)$ is a Hilbert space to which $\sum a_i S(x_i)$ belongs. The reverse implications are mostly trivial; the top one uses the fact that $H_{K_2} = \overline{\text{span}}\{K_{2,y} : y \in X_2\}$. □

We now show that a neural response can be changed on points $x_1, \ldots, x_k \in X_2$ for which $\{\Phi_2(x_1), \ldots, \Phi_2(x_k)\}$ are linearly independent, while still retaining the same derived kernel $K_2$ (on those points) provided we have the freedom to change the templates to arbitrary templates of the second kind.

**Proposition 4.1.4.** *Assume the architecture specified in Section 4.1.1. By the pevious proposition, assume without loss of generality that $S_{base}$ is linear. Let $N_{base} : X_2 \to \mathcal{H}_{K_1}$ be any* **specific linear base neural response**. *Suppose $x_1, \ldots, x_k \in X_2$ such that*

$$S(x_1), \ldots, S(x_k) \in \mathcal{L}^2_\nu(T_1)$$

*are linearly independent. Then*

$$N_{base}(x_1), \ldots, N_{base}(x_k) \in H_{K_1}$$

*are linearly independent if and only if there exists a set of templates of the second kind, $\mathcal{T} \subset H_{K_1}$, contained in the span of the $N_{base}(x_i)$, and a measure $\nu$ on $\mathcal{T}$ such that:*

$$K_2(x_i, x_j) = \langle N(x_i), N(x_j) \rangle_{\mathcal{L}^2_\nu(\mathcal{T})}, \forall i, j \in [k] \tag{4.1.1}$$

*where $N = \rho_{\mathcal{T}}(N_{base})$ is the neural response into $\mathcal{L}^2_\nu(\mathcal{T})$ induced by $N_{base}$. Moreover, if $k$ is maximal, i.e. $k = \dim H_{K_2}$, then such $\mathcal{T}$, when it exists, is unique.*

That (4.1.2) implies the $N(x_i)$ are linearly independent is an immediate consequence of Lemma 4.1.3. Using $N(x) = \pi_{\mathcal{T}} \circ N_{\text{base}}(x), \forall x \in X_2$, this then implies the $N_{\text{base}}(x_i)$ are linearly independent. The other direction, i.e. the existence of suitable $\mathcal{T}, \nu$, remains to be proved, as does uniqueness.

**Remark 4.1.5.** More generally, by Lemma 4.1.3, having

$$K_2(x, x') = \langle N(x), N(x') \rangle_{\mathcal{L}^2_\nu(T_1)}$$

for all $x, x' \in U$ for some subset $U \subset X_2$, implies the *linear dependencies* induced by $S$ within $U$ coincide with those induced by $N$. We will return to this discussion in Section 4.2.3.

**Remark 4.1.6.** The non-existence or uniqueness of templates of the second kind making (4.1.2) hold applies to templates of either first or second kind, since $\langle \cdot, \cdot \rangle_{\mathcal{L}^2(T,\nu)} = \langle \cdot, \cdot \rangle_{\mathcal{L}^2(\mathcal{T}',\nu')}$ for $\mathcal{T}' = \{\Phi_1(t) : t \in T\}$ and $\nu' = (\Phi_1)_* \nu$; see Section 3.10 and in particular equation (3.10.1) for details.

We emphasize, as a Corollary, a particular case of the final statement of the Proposition.

**Corollary 4.1.7.** *In particular, in the previous Proposition, if we are interested in obtaining a* **given** *weighted average, i.e.*

$$N_{base}(x) = N^{avg}_{base}(x) = \sum_{h \in H} \mu(h) K_{1,h(x)}, \forall x \in \{x_1, \ldots, x_k\} \subset X_2$$

*(with fixed choice of transformation set $H$ and measure $\mu$ on $H$) then (4.1.2) is possible if and only if $x_1, \ldots, x_k \in X_2$ are such that,*

$$\sum_{h \in H} \mu(h) K_{1,h(x_1)}, \ldots, \sum_{h \in H} \mu(h) K_{1,h(x_k)} \in \mathcal{H}_{K_1}$$

*are linearly independent. See Section 4.2.3 regarding extension of (4.1.2) to all $x, x' \in X_2$ (not just $x, x' \in \{x_1, \ldots, x_k\} \subset X_2$).*

Somewhat of a converse to the previous Proposition also holds, allowing us to choose the templates but then forcing the neural response.

**Proposition 4.1.8.** *Assume the architecture specified in Section 4.1.1. By the Proposition 4.1.1, assume without loss of generality that $S_{base}$ is linear. Let $T = \{s_i : i \in [m]\} \subset X_1$ be any specific template set such that $\Phi_1(s_1), \ldots, \Phi_1(s_m)$ are linearly independent. Suppose $x_1, \ldots, x_n \in X_2$ such that*

$$S(x_1), \ldots, S(x_n) \in \mathcal{L}^2_\nu(T_1)$$

*are linearly independent. Then there exists a linear base neural response $N_{base}$ into the span of the $\Phi_1(s_i)$ such that*

$$K_2(x_i, x_j) = \langle N(x_i), N(x_j) \rangle_{\mathcal{L}^2_\nu(\mathcal{T})}, \forall i, j \in [k] \tag{4.1.2}$$

*where $N = \rho_{\mathcal{T}}(N_{base})$ is the neural response into $\mathcal{L}^2_\nu(\mathcal{T})$ induced by $N_{base}$. Moreover, if $k$ is maximal, i.e. $k = n = \dim H_{K_2}$, then this $N_{base}$ is unique.*

Finally, we show that linear neural responses can always be replaced by weighted averages when $H, \mu$ are not fixed in advance. We do this by constructing $H, \mu$ which make the conditions of the previous Corollary hold.

**Proposition 4.1.9.** *Assume the architecture specified in Section 4.1.1. Suppose $m < \dim \mathcal{H}_{K_1}$ or there exists $y \in X_1$ such that $K_{1,y} = 0$. Also suppose $X_2$ is finite (as it would be in all practical applications)[2]. Then there exists a finite transformation set $H \subset Maps(X_2 \to X_1)$ and measure $\mu$ on $H$ such that the weighted average $N^{avg}$ that $H, \mu$ define into $\mathcal{L}^2_\nu(T_1)$ induces a derived kernel that coincides exactly with $K_2$, the kernel induced by $S$. In other words, $S$* **can be replaced by a weighted average.**

## 4.2 Preliminaries

Before proving the Propositions, we develop some tools. We first observe a general fact about linear independent vectors and inner products.

### 4.2.1 Change of bases

**Lemma 4.2.1.** *Suppose we are given two sets of $k$ linearly independent vectors:*

$$\sigma_1, \ldots, \sigma_k$$

*and*

$$\eta_1, \ldots, \eta_k$$

*in an $n$-dimensional real Hilbert space with inner product $\langle \cdot, \cdot \rangle$ and a basis $\tau_1, \ldots, \tau_m$. Then there exists a new basis $\tilde{\tau}_1, \ldots, \tilde{\tau}_m$ such that $\langle \sigma_i, \tau_j \rangle = \langle \eta_i, \tilde{\tau}_j \rangle$ for all $i, j$.*

[2] In fact, as long as there are only finitely many vectors $N(x_2)$ such that $x_2 \in X_2$ the same construction can be made to work.

*Proof.* First, assume that $k = m$, by completing the linearly independent sets to bases (in arbitrary fashion) . Fix $j \in [m]$, and for each $i \in [m]$ let $c_i = \langle \sigma_i, \tau_j \rangle_{K_1}$. Then

$$H_i := \{v \ : \ \langle \eta_i, v \rangle = c_i\}$$

is an affine hyperplane orthogonal to $N(x_i)$. Specifically,

$$H_i = \frac{c_i}{||\eta_i||^2_{K_1}} \eta_i + \text{span}[\eta_i]^{\perp}.$$

It can then be shown by induction that $\bigcap_{i=1}^{n} H_i$ is nonempty. Indeed if the intersection $\bigcap_{i=1}^{k} H_i$ is nonempty for some $k < m$ then it is of the form

$$w + \text{span}[\eta_1, \ldots, \eta_k]^{\perp}$$

and so cannot be parallel to $H_{k+1}$ since this would imply

$$\eta_{k+1} \in \text{span}[\eta_1, \ldots, \eta_k].$$

Therefore $\bigcap_{i=1}^{k+1} H_i \neq \emptyset$ and by induction, $\bigcap_{i=1}^{m} H_i \neq \emptyset$. Let $\tilde{\tau}_j$ be an element of this intersection. By construction, $\forall i \in [m]$,

$$\langle \sigma_i, \tau_j \rangle = \langle \eta_i, \tilde{\tau}_j \rangle.$$

This may now be repeated to construct a $\tilde{\tau}_j$ for each $j \in [m]$. Any linear dependence $\sum_j a_j \tilde{\tau}_j = 0$ would imply $\langle \sigma_i, \sum_j a_j \tau_j \rangle = 0$ for all $i$, and hence $\sum_j a_j \tau_j = 0$ because the $\sigma_i$ are a basis for the span of $\tau_1, \ldots, \tau_m$. Thus the linear independence of the $\tau_j$ forces that of the $\tilde{\tau}_j$. □

### 4.2.2 Rank

**Definition 4.2.2.** We define the **rank** of a neural response as

$$\text{rank } (N_{\text{base}}) := \max \dim \overline{\text{span}}(\rho_{\mathcal{T}}(N_{\text{base}})(X_2))$$

where the maximum is taken over all possible template sets $\mathcal{T} \subset \mathcal{H}_{K_1}$.

Because the elements of $\rho_{\mathcal{T}}(N_{\text{base}})(X_2)$ are functions on $\mathcal{T}$, rank is bounded above by $|X_1|$ and may in general achieve this. For linear neural responses, however, the functions in $\rho_{\mathcal{T}}(N_{\text{base}})(X_2)$ are restrictions of elements of $\mathcal{H}_{K_1}$ so rank cannot exceed $\dim \mathcal{H}_{K_1}$.

Given finite subsets $Y_1 \subset X_1$ and $Y_2 \subset X_2$, the values of $N_{\text{base}}(y_2)(y_1)$ for $y_1 \in Y_1, y_2 \in Y_2$ are elements of $\mathcal{H}_{K_1}$ and so give rise to functions on $Y_1$. Let $M_{Y_1,Y_2}$ be the matrix of these values.

$$(M_{Y_1,Y_2})_{ij} = N_{\text{base}}(y_2)_j(y_1)_i(y_1)_i, \forall i \in [|Y_1|], j \in [|Y_2|],$$

where $(y_1)_i$ is the $i$'th element of $Y_1$ and $(y_2)_j$ is the $j$'th element of $Y_2$.

Then the rank of $N_{\text{base}}$ is just the maximal rank of $M_{Y_1,Y_2}$ over all possible choices of finite subsets $Y_1, Y_2$.

### 4.2.3 Formal linear dependencies

Rank is part of a more general phenomemon. We observe that any neural response, as a map from a set $X_2$ to a Hilbert space $\mathcal{F}(T_1)$ induces linear dependencies[3] in the set $X_2$ in a formal way. Suppose $K_2 = N^*\langle\cdot,\cdot\rangle$. Then whenever there are $r \in \mathbb{N}$, $a_1, \ldots, a_r \in \mathbb{R}$, $x_1, \ldots, x_r \in X_2$ such that $\sum_{i=0}^{r} a_i N(x_i) = 0$ we declare the formal linear dependency $\sum_{i=0}^{r} a_i x_i = 0$ in the set $X_2$. Rank is then the maximal cardinality of a set $\{x_1, \ldots, x_k\} \subset X_2$ with no nontrivial formal linear dependencies.

Note that $\sum_{i=0}^{r} a_i N(x_i) = 0$ if and only if $\sum_{i=0}^{r} a_i K_{2,x_i} = 0$, by Lemma 4.1.3. Thus, as observed in Remark 4.1.5, a *necessary* condition for a kernel $K_2$ to be derived by the neural response $N$ is that the $K_{2,x}$ satisfy the same linear dependencies as those formally induced among the $x \in X_2$ by $N$. Moroever, if we let $x_1, \ldots, x_n \in X_2$ be such that the $K_{2,x_i}$ form a *basis* for $\mathcal{H}_{K_2}$ then the Gram matrix of the $K_{2,x_i}$ in the derived Hilbert space $\mathcal{H}_{K_2}$ must coincide with the Gram matrix of the $N(x_i)$ w.r.t the inner product in $\mathcal{F}(T_1)$ (this is just a statement of the definition of derived kernel). Conversely, both of these conditions together imply the kernel is derived by $N$. We have,

**Lemma 4.2.3.** *Suppose that two neural responses, $N$ and $S$, map $X_2$ into $\mathcal{F}(T)$ and $\mathcal{F}(T')$ respectively. Then, the kernels derived via $N$ and $S$ coincide, i.e.,*

$$\langle N(x), N(x')\rangle_{\mathcal{F}(T)} = \langle S(x), S(x')\rangle_{\mathcal{F}(T')}, \forall x, x' \in X_2$$

*if and only if $N$ and $S$ induce the* **same formal linear dependencies** *in $X_2$ and*

$$N_{base}(x_i) = S_{base}(x_i), \forall i \in [n]$$

*for some (hence any) $x_1, \ldots, x_n \in X_2$ such that $K_{2,x_i}$ form a basis for $\mathcal{H}_{K_2}$.*

[3] A motivation of the so-called "kernel trick" is to establish a weaker version of a vector space structure on the space of objects of interest. In fact, it establishes formal linear dependencies as just described, and moreover, a weaker form of an inner product: the derived reproducing kernel.

This means that when formal linear dependencies coincide we are able to concern ourselves only with the values of neural responses at elements $x_1, \ldots, x_n$ such that $K_{2,x_i}$ form a *basis* for $\mathcal{H}_{K_2}$ (the formal linear dependencies will take care of all other $x \in X_2$). On the other hand, when formal linear dependencies do not coincide there is no hope of deriving the same kernel via the two neural responses.

### 4.2.4 Convenient view of $\mathcal{L}^2(T)$ for finite $T$

Recall that when a neural response $N : X_2 \to \mathcal{L}^2(T)$ satisfying Axiom 3.5.1 is linear, we have $N(x_2)(t) = \langle N_{\text{base}}(x_2), K_{1,t}\rangle_{H_{K_1}}$ for all $x_2 \in X_2$ and $t \in T \subset X_1$ (assuming the usual two-layer notation). Thus, while we **pull back the inner product from** $\mathcal{L}^2(T)$ (to obtain $K_2$), the elements of $N(x_2)$ are functions on $T$ and are actually obtained by restriction of **functions in** $\mathcal{H}_{K_1}$ to $T$. They thus correspond to functions in $\mathcal{H}_{K_1}$ which are zero on the orthogonal complement of $\mathcal{T} = \Phi_1(T)$. Specifically,

$$N(x_2)(t) = \langle \pi_{\mathcal{T}} \circ N_{\text{base}}(x_2), K_{1,t}\rangle_{K_1}.$$

To make the proofs of the Propositions simpler, we will therefore work within $\mathcal{H}_{K_1}$, using its vector space structure (as a space of functions), and define a non-standard inner product in this vector space which we can pull-back to achieve the same effect obtained by that of $\mathcal{L}^2(T)$.

**Remark 4.2.4. Warning:** We will now have two inner products on the vector space $\mathcal{H}_{K_1}$: one standard, the other not. We will use the notation $\pi_{\mathcal{T}}$ and $\perp$ *exclusively* to mean projection and orthogonal complement according to the *standard* inner product of $\mathcal{H}_{K_1}$.

Suppose we have a **finite** template set (of second kind) $\mathcal{T} = \{\tau_1, \ldots, \tau_m\} \subset \mathcal{H}_{K_1}$, whose elements are **linearly independent**. Or if we start with a set of usual templates $T \subset X_1$, then define $\mathcal{T} = \{K_{1,t} \colon t \in T\} \subset \mathcal{H}_{K_1}$. Let $\nu$ be a total measure on $\mathcal{T}$. Define a non-standard inner product on $\mathcal{H}_{K_1}$ by:

$$\langle u, v\rangle_{(\mathcal{T},\nu)} := \sum_{i=1}^{m} \langle u, \tau_i\rangle_{K_1}\langle v, \tau_i\rangle_{K_1}\nu(\tau_i) + \langle u', v'\rangle_{K_1},$$

where $u', v'$ are the projections of $u, v$ respectively in $\text{span}(\mathcal{T})^{\perp}$. This gives the usual inner product on $\text{span}(\mathcal{T})^{\perp}$ but a non-standard one on $\text{span}(\mathcal{T})$. Clearly $\langle \cdot, \cdot\rangle_{(\mathcal{T},\nu)}$ is a symmetric bilinear form and it is nondegenerate because only the zero vector $v = 0$ can belong to $\text{span}(\mathcal{T})$ and at the same time have $\langle u, \tau_i\rangle_{K_1} = 0$ for all $\tau_i \in \mathcal{T}$. The key property of this inner product is the following.

**Lemma 4.2.5.** *Let $N_{base}$ be a linear base neural response and $N = \rho_{\mathcal{T}}(N_{base})$ the associated neural response into $\mathcal{L}^2_\nu(\mathcal{T})$. Then we have*

$$\langle N(\cdot), N(\cdot)\rangle_{\mathcal{L}^2_\nu(\mathcal{T})} = \langle \pi_{\mathcal{T}} \circ N_{base}(\cdot), \pi_{\mathcal{T}} \circ N_{base}(\cdot)\rangle_{\mathcal{T},\nu} \tag{4.2.1}$$

*where $\pi_{\mathcal{T}}$ denotes projection onto the span of $\mathcal{T}$.*

*Proof.* Indeed, for $x_2, x_2' \in X_2$ :

$$\begin{aligned}\langle N(x_2), N(x_2')\rangle_{\mathcal{L}^2_\nu(\mathcal{T})} &= \sum_{i=0}^{m} N(x_2)(\tau_i)\cdot N(x_2')(\tau_i)\nu(\tau_i)\\ &= \sum_{i=0}^{m}\langle N_{\text{base}}(x_2),\tau_i\rangle_{K_1}\cdot\langle N_{\text{base}}(x_2'),\tau_i\rangle_{K_1}\nu(\tau_i)\\ &= \langle \pi_{\mathcal{T}}\circ N_{\text{base}}(x_2), \pi_{\mathcal{T}}\circ N_{\text{base}}(x_2')\rangle_{\mathcal{T},\nu}\end{aligned}$$

because $\pi_{\mathcal{T}}\circ N_{\text{base}}(x_2)$ as a linear functional takes on the same values as $N_{\text{base}}(x_2)$ on the elements of span$(\mathcal{T})$ but is zero on their orthogonal complement (i.e. as a vector it belongs to span$(\mathcal{T})$ ). □

Note that to each $N(x_2) \in \mathcal{L}^2_\nu(\mathcal{T})$ we have associated $\pi_{\mathcal{T}}\circ N_{\text{base}}(x_2) \in \mathcal{H}_{K_1}$.

## 4.3 Proof of the Propositions

**Remark 4.3.1.** Let $\dim \mathcal{H}_{K_2} = n$. Recall that if $K_2$ is obtained as a pullback via $N$ of an inner product $\langle\cdot,\cdot\rangle_{\mathcal{F}}$ in a Hilbert space $\mathcal{F}$, then we may lift $N$ to an injective linear isometry $\widetilde{N} : \mathcal{H}_{K_2} \to \mathcal{F}$ (see Section 3.9.1) whose image is the closure of the span of $N(X_2)$. This allows us to conclude that $N(X_2)$ spans an $n$-dimensional subspace of $\mathcal{F}$ (it is closed because finite-dimensional). In our setting, using (4.2.1), this implies that both $N(X_2) \subset \mathcal{L}^2_{\nu_1}(T_1)$ and $\pi_{\mathcal{T}}\circ N_{\text{base}}(X_2) \subset \mathcal{H}_{K_1}$ span subspaces of dimension $n$.

*Proof of Proposition 4.1.1.* Let $x_1, \ldots, x_n \in X_2$ such that

$$\Phi_2(x_1), \ldots, \Phi_2(x_n)$$

are linearly independent in $\mathcal{H}_{K_2}$. Now, for each $i$, consider $S_{\text{base}}(x_i) \in \text{Maps}(X_1 \to \mathcal{H}_{K_1})$. As described in Section 3.10.2, it can be evaluated at each $\tau_j \in \mathcal{T}_1$. Since the $\tau_j$ are linearly independent, there exists a linear functional on $\mathcal{H}_{K_1}$ which agrees with $S(x_i)$ on $\mathcal{T}$. Let $F_i \in \mathcal{H}_{K_1}$ be the Riesz representative of this functional.

Now let $S'_{\text{base}}$ be the map

$$S'_{\text{base}} : X_2 \to \mathcal{H}_{K_1}$$

defined by

$$S'_{\text{base}}(x_i) := F_i, \text{ for each } i = 1, \ldots, n,$$

and extended to all of $X_2$ using the formal linear dependencies already induced in $X_2$ (by $S$). This is a linear base neural response. And, its values on the $\tau_j \in \mathcal{T}_1$ agree with those of $S$, therefore, the induced neural response

$$S' : X_2 \to \mathcal{L}^2(\mathcal{T}_1, \nu_1)$$

defines the same derived kernel on $X_2$ as that defined by $S$. □

We now prove the Proposition regarding a specific linear neural response $N_{\text{base}}$.

*Proof of Proposition 4.1.4.* By the preceding argument we may, without loss of generality, assume that $S_{\text{base}}$ is linear. Suppose we are given elements $x_1, \ldots, x_n \in X_2$ such that

$$\pi_{\mathcal{T}_1} \circ S_{\text{base}}(x_1), \ldots, \pi_{\mathcal{T}_1} \circ S_{\text{base}}(x_k)$$

and

$$\pi_{\mathcal{T}_1} \circ N_{\text{base}}(x_1), \ldots, \pi_{\mathcal{T}_1} \circ N_{\text{base}}(x_k)$$

are both are linearly independent in $\mathcal{H}_{K_1}$. See Remark 4.2.4 regarding the notation $\pi_{\mathcal{T}_1}$.

By Lemma 4.2.1 there exist linearly independent $\tilde{\tau}_1, \ldots, \tilde{\tau}_m \in \mathcal{H}_{K_1}$ such that $(\forall i \in [n])(\forall j \in [m])$,

$$\langle \pi_{\mathcal{T}_1} \circ S_{\text{base}}(x_i), \tau_j \rangle_{K_1} = \langle \pi_{\mathcal{T}_1} \circ N_{\text{base}}(x_i), \tilde{\tau}_j \rangle_{K_1}.$$

This implies $(\forall i \in [n])(\forall j \in [m])$,

$$\langle \pi_{\mathcal{T}_1} \circ S_{\text{base}}(x_i), \pi_{\mathcal{T}_1} \circ S_{\text{base}}(x_j) \rangle_{\mathcal{T}_1,\nu_1} = \langle \pi_{\mathcal{T}_1} \circ N_{\text{base}}(x_i), \pi_{\mathcal{T}_1} \circ N_{\text{base}}(x_j) \rangle_{\mathcal{T},\nu} \tag{4.3.1}$$

where $\mathcal{T} = \tilde{\tau}_1, \ldots, \tilde{\tau}_m$ and $\nu(\tilde{\tau}_i) = \nu_1(\tau_i)$.

Given $\mathcal{T}$ and $\nu$ as above define

$$N = \rho_{\mathcal{T}}(N_{\text{base}}) : X_2 \to \mathcal{L}^2_\nu(\mathcal{T}),$$

the neural response into $\mathcal{L}^2_\nu(\mathcal{T})$ induced by $N_{\text{base}}$. We have:

$$\begin{aligned}
K_2(x_i, x_j) &= S^* \langle x_i, x_j \rangle_{\mathcal{L}^2_{\nu_1}(\mathcal{T}_1)} \\
&= \langle S(x_i), S(x_j) \rangle_{\mathcal{L}^2_{\nu_1}(\mathcal{T}_1)} \\
&= \langle \pi_{\mathcal{T}_1} \circ S_{\text{base}}(x_i), \pi_{\mathcal{T}_1} \circ S_{\text{base}}(x_j) \rangle_{\mathcal{T}_1,\nu_1} \text{ in } \mathcal{H}_{K_1} \\
&= \langle \pi_{\mathcal{T}_1} \circ N_{\text{base}}(x_i), \pi_{\mathcal{T}_1} \circ N_{\text{base}}(x_j) \rangle_{\mathcal{T},\nu} \text{ in } \mathcal{H}_{K_1} \text{ by (4.3.1)} \\
&= \langle N(x_i), N(x_j) \rangle_{\mathcal{L}^2_\nu(\mathcal{T})} \\
&= N^* \langle x_i, x_j \rangle_{\mathcal{L}^2_\nu(\mathcal{T})}.
\end{aligned}$$

The final statement of the Proposition follows from Lemma 4.2.3. □

**Remark 4.3.2.** In the above proof, if $N_{\text{base}}$ and $S_{\text{base}}$ induce the *same formal linear dependencies* in $X_2$ then so do $\pi_{\mathcal{T}_1} \circ N_{\text{base}}$ and $\pi_{\mathcal{T}_1} \circ S_{\text{base}}$ so $K_2(x, x')$ and $N^* \langle x, x' \rangle_{\mathcal{L}^2_\nu(\mathcal{T})}$ coincide for all $x, x' \in X_2$ (extending by linearity the statement proven above for the $x_i$).

*Proof of Proposition 4.1.8.* The argument is essentially the same as in the previous proof, except that we use Lemma 4.2.1 instead to get linearly independent $\tilde{\tau}_1, \ldots, \tilde{\tau}_n \in \mathcal{H}_{K_1}$ such that $(\forall i \in [n])(\forall j \in [m])$,

$$\langle \pi_{\mathcal{T}_1} \circ S_{\text{base}}(x_i), \tau_j \rangle_{K_1} = \langle \Phi_1(s_j), \tilde{\tau}_i \rangle_{K_1}$$

and then set $N_{\text{base}}(x_i) = \tilde{\tau}_i$ for each $i \in [n]$. □

We now prove the final Proposition.

*Proof of Proposition 4.1.9.* We do so by exhibiting a suitable set of transformations $H$ with measure $\mu$. There will be $|X_2|d^2$ transformations, assuming $X_2$ is finite [4].

As before, let $\mathcal{T}_1 = \Phi_1(T_1)$ and for each $i \in [m]$ let $\tau_i = \Phi_1(t_i)$.

Let $y_1, \ldots, y_d \subset X_1$ be such that $K_{1,y_1}, \ldots, K_{1,y_d}$ form a basis for $\mathcal{H}_{K_1}$ and suppose there is $y \in X_1$ such that $K_{1,y}$ is orthogonal to all $\tau_i, i \in [m]$. Such $y$ trivially exists if there is some $y$ with $K_{1,y} = 0$ and may exist if $m < d = \dim \mathcal{H}_{K_1}$.

If no such $y$ exists, then at least $m < d$ implies we can find a nontrivial linear combination $\theta = \sum_{i=1}^{d} b_i K_{1,y_i}$ which is orthogonal to the span of the $\tau_i$ (because they span an $m$-dimensional subspace of the $d$-dimensional space $\mathcal{H}_{K_1}$). We will return to this case later, and first deal with the simpler case where a single $y$ exists as described. From now on, **write** $K$ **for** $K_1$ to simplify the reading.

Let $x_1, \ldots, x_n \in X_2$ be such that

$$\pi_{\mathcal{T}_1} \circ S_{\text{base}}(x_1), \ldots, \pi_{\mathcal{T}_1} \circ S_{\text{base}}(x_n)$$

are linearly independent. For each $j \in [d]$ and for each $x, x' \in X_2$, let

$$h_j^x(x') = \begin{cases} y_j & \text{if } x = x', \\ y & \text{otherwise.} \end{cases}$$

This makes each $h_j^x$ into an element of $\text{Maps}(X_2 \to X_1)$. Now let $H = \{h_j^x \ : \ j \in [d], x \in X_2\}$. It remains to define a measure $\mu$ on $H$. Because the $y_j$ were chosen so that $K_{y_j}$ form a basis for $\mathcal{H}_{K_1}$, it follows that for each $x \in X_2$, there exist unique coefficients $a_j(x)$ such that $S_{\text{base}}(x) = \sum_j a_j(x) K_{y_j}$. For each $x \in X_2, j \in [d]$ set

$$\mu(h_j^x) := a_j(x).$$

[4] Or, if $X_2$ is infinite but there are only finitely many different vectors $N(x_2)$ for $x_2 \in X_2$, say $k$ of them, then one could alter the proof so that $kd$ or $kd^2$ transformations would suffice.

Now let $N^{\text{avg}}_{\text{base}}$ be the (base) weighted average neural response into $\mathcal{H}_{K_1}$. For arbitrary $x \in X_2$, we have

$$\begin{aligned} N^{\text{avg}}_{\text{base}}(x) &= \sum_{j\in[d]} \sum_{z\in X_2} a_j(z) K_{h^z_j(x)} \\ &= \sum_{j\in[d]} a_j(x) K_{h^x_j(x)} + \sum_{j\in[d]} \sum_{\substack{z\in X_2 \\ z\neq x}} a_j(z) K_{h^z_j(x)} \\ &= \sum_{j\in[d]} a_j(x) K_{y_j} + R(x) K_y \\ &= S_{\text{base}}(x) + R(x) K_y, \end{aligned}$$

where $R(x) = \sum_{j\in[d]} \sum_{\substack{z\in X_2 \\ z\neq x}} a_j(z)$ depends only on $x$.

Note that $\pi_{\mathcal{T}_1} \circ N^{\text{avg}}_{\text{base}}(x) = \pi_{\mathcal{T}_1} \circ S_{\text{base}}(x)$ for each $x \in X_2$ so the base neural responses $S_{\text{base}}$ and $N^{\text{avg}}_{\text{base}}$ have the same associated neural response into $T_1$. Here we are exploiting the fact that $m < d = \dim \mathcal{H}_{K_1}$ so base neural responses are not unique (see Proposition 3.5.7). The derived kernel for the given weighted average therefore coincides with $K_2$.

We now address the more complicated case where no single $y$ existed as described, but where one does have some nontrivial $\theta = \sum_{\ell=1}^{d} b_\ell K_{y_\ell}$ orthogonal to all $\tau_i$ for $i \in [m]$. We also require that

$$\sum_{\ell=1}^{d} b_\ell = 1.$$

Then construct a set of transformations as above for each $\ell = 1, \ldots, d$:

$$H_\ell := \{h^x_j[\ell] : j \in [d], x \in X_2\},$$

that are exactly the same as the $h^x_j$ except that $y_\ell$ should be used instead of $y$ in the case $x \neq x'$:

$$h^x_j[\ell](x') = y_\ell.$$

Let $H' = \bigcup_{\ell=1}^{d} H_\ell$ and define a measure $\nu$ on $H'$ by

$$\nu(h^x_j[\ell]) = b_\ell \cdot \mu(h^x_j) = b_\ell a_j(x)$$

Then for arbitrary $x \in X_2$, we have

$$\begin{aligned} N^{\text{avg}}_{\text{base}}(x_i) &= \sum_{\ell=1}^{d} b_\ell \sum_{j\in[d]} \sum_{z\in X_2} a_j(z) K_{h_j^z(x)} \\ &= \sum_{\ell=1}^{d} b_\ell \left[S_{\text{base}}(x) + R(x) K_{y_\ell}\right] \\ &= S_{\text{base}}(x) + R(x)\theta \end{aligned}$$

and the rest of the argument is the same. □

## 4.4 Examples

**Example 4.4.1.** Consider an alphabet $\Sigma = \{a, b, c\}$ of 3 letters. Let

$$X_1 = T_1 = \Sigma \tag{4.4.1}$$
$$X_2 = \Sigma^2. \tag{4.4.2}$$

Let $K_1$ be a kernel on $X_1$ given by the matrix:

$$M = \begin{pmatrix} 1 & 0 & \sqrt{\frac{3}{4}} \\ 0 & 1 & \frac{1}{2} \\ \sqrt{\frac{3}{4}} & \frac{1}{2} & 1 \end{pmatrix}.$$

Let $S$ be the **max** neural response

$$S : X_2 \to \mathcal{L}^2(\mathcal{T}_1)$$

where $\mathcal{T}_1 = \{K_a, K_b, K_c\}$ and $\nu_1$ is the empirical, i.e. uniform, measure on $\mathcal{T}_1$ - from now on when no measure is specified it will be assumed to be uniform. Let $K_2$ be the derived kernel,

$$K_2(\cdot,\cdot) = S^* \langle \cdot, \cdot \rangle_{\mathcal{L}^2_{\nu_1}(\mathcal{T}_1)}.$$

The following table gives the values of $S(x_2)(\tau)$ for all $x_2 \in X_2, \tau \in \mathcal{T}_1$:

| $X_2 \backslash \mathcal{T}_1$ | $K_a$ | $K_b$ | $K_c$ |
|---|---|---|---|
| $aa$ | 1 | 0 | $\sqrt{\frac{3}{4}}$ |
| $ab$ | 1 | 1 | $\sqrt{\frac{3}{4}}$ |
| $ac$ | 1 | $\frac{1}{2}$ | 1 |
| $ba$ | 1 | 1 | $\sqrt{\frac{3}{4}}$ |
| $bb$ | 0 | 1 | $\frac{1}{2}$ |
| $bc$ | $\sqrt{\frac{3}{4}}$ | 1 | 1 |
| $ca$ | 1 | $\frac{1}{2}$ | 1 |
| $cb$ | $\sqrt{\frac{3}{4}}$ | 1 | 1 |
| $cc$ | $\sqrt{\frac{3}{4}}$ | $\frac{1}{2}$ | 1 |

This is clearly not a linear neural response since the first three rows, for example, are linearly independent making $\dim \mathcal{H}_{K_2} = 3$ (i.e. $S(aa), S(ab), S(ac)$ are L.I. in $\mathcal{L}^2(\mathcal{T}_1)$) while $\dim \mathcal{H}_{K_1} = 2$. Or to see it more directly, note that $K_c$ is a linear combination of $K_a$ and $K_b$ and yet the third column is not a linear combination of the first two.

Let $S_1'$ and $S_2'$ be the neural responses defined respectively (into $\mathcal{L}^2(\mathcal{T}_1^1)$ and $\mathcal{L}^2(\mathcal{T}_1^2)$) by the tables,

$S_1'$ :

| $X_2 \backslash \mathcal{T}_1^1$ | $K_a$ | $K_b$ |
|---|---|---|
| $aa$ | 1 | 0 |
| $ab$ | 1 | 1 |
| $ac$ | 1 | $\frac{1}{2}$ |
| $ba$ | 1 | 1 |
| $bb$ | 0 | 1 |
| $bc$ | $\sqrt{\frac{3}{4}}$ | 1 |
| $ca$ | 1 | $\frac{1}{2}$ |
| $cb$ | $\sqrt{\frac{3}{4}}$ | 1 |
| $cc$ | $\sqrt{\frac{3}{4}}$ | $\frac{1}{2}$ |

$S_2'$ :

| $X_2 \backslash \mathcal{T}_1^2$ | $K_c$ |
|---|---|
| $aa$ | $\sqrt{\frac{3}{4}}$ |
| $ab$ | $\sqrt{\frac{3}{4}}$ |
| $ac$ | 1 |
| $ba$ | $\sqrt{\frac{3}{4}}$ |
| $bb$ | $\frac{1}{2}$ |
| $bc$ | 1 |
| $ca$ | 1 |
| $cb$ | 1 |
| $cc$ | 1 |

where $\mathcal{T}_1^1 = \{K_a, K_b\}$ and $\mathcal{T}_1^2 = \{K_c\}$. These correspond to base linear neural responses if we extend by linearity to all of $\mathcal{H}_{K_1}$ (since $K_a, K_b$ are linearly independent, as is $K_c$), and the associated neural responses into $\mathcal{L}^2(\mathcal{T}_1^1)$ and $\mathcal{L}^2(\mathcal{T}_1^2)$ respectively are just given by the same tables.

Because these smaller template sets are linearly independent, we can apply Corollary 4.1.2 to obtain that

$$K_2(\cdot, \cdot) = K_2^1(\cdot, \cdot) + K_2^2(\cdot, \cdot)$$

where $K_2^i(\cdot,\cdot) = (S_i')^* \langle \cdot, \cdot \rangle_{\mathcal{L}^i(\mathcal{T}_1^1)}$ for $i = 1, 2$. To see this concretely, let the matrices $N_1$ and $N_2$ correspond to the entries of the two tables above and and let $N$ correspond to the table for $S$, then one computes that $N_1 N_1^T + N_2 N_2^T = N N^T$. The summands are the $9 \times 9$ matrices giving the kernels $K_2^i(\cdot,\cdot)$ respectively; $N^T N$ is the $9 \times 9$ matrix giving the kernel $K_2$.

This is a general fact about inner product matrices. If $N_1$ and $N_2$ are submatrices of $N$ corresponding to a decomposition of $N$ into two vertical blocks ($N_1$ is the first $r$ columns, and $N_2$ the remaining ones), then the following relation holds between their inner product matrices: $N N^T = N_1 N_1^T + N_2 N_2^T$.

Now suppose one wants to derive the same kernel $K_2^1$ from a specific linear neural response $N_{\text{base}}$, such that $N_{\text{base}}(aa)$ is 1 on $K_a$ and 0 on $K_b$, while $N_{\text{base}}(ab)$ is 0 on $K_a$ and 5 on $K_b$, with the same linear dependencies as $S$ on the rest of $X_2$. Let us moreover assume that $N_{\text{base}}(X_2)$ is already contained in $\mathcal{T}_1^1$ so we do not need to compose with $\pi_{\mathcal{T}_1^1}$ and the notation remains simpler.

Note that because $K(a,a) = 1 = K(b,b)$ and $K(a,b) = 0$, the values that $N_{\text{base}}$ takes on $K_a$ and $K_b$ are exactly the coefficients of $N_{\text{base}}$ when expressed as a linear combination of $K_a$ and $K_b$ respectively. Thus $N_{\text{base}}(aa) = K_a$ and $N_{\text{base}}(ab) = 5K_b$.

We now need to compute a new template set $\mathcal{T} = \{\tilde{\tau}_1, \tilde{\tau}_2\}$ such that

$$\begin{aligned}
\langle N_{\text{base}}(aa), \tilde{\tau}_1 \rangle &= \langle S(aa), K_a \rangle = 1 \\
\langle N_{\text{base}}(ab), \tilde{\tau}_1 \rangle &= \langle S(ab), K_a \rangle = 1 \\
\langle N_{\text{base}}(aa), \tilde{\tau}_2 \rangle &= \langle S(aa), K_b \rangle = 0 \\
\langle N_{\text{base}}(ab), \tilde{\tau}_2 \rangle &= \langle S(ab), K_b \rangle = 1.
\end{aligned}$$

This is accomplished by $\tilde{\tau}_1 = K_a + .2K_b$ and $\tilde{\tau}_2 = .2K_b$. Indeed

$$\begin{aligned}
\langle N_{\text{base}}(aa), \tilde{\tau}_1 \rangle &= \langle K_a, K_a + .2K_b \rangle_{K_1} = 1 \\
\langle N_{\text{base}}(ab), \tilde{\tau}_1 \rangle &= \langle 5K_b, K_a + .2K_b \rangle_{K_1} = 1 \\
\langle N_{\text{base}}(aa), \tilde{\tau}_2 \rangle &= \langle K_a, .2K_b \rangle_{K_1} = 0 \\
\langle N_{\text{base}}(ab), \tilde{\tau}_2 \rangle &= \langle 5K_b, .2K_b \rangle_{K_1} = 1
\end{aligned}$$

Because $N_{\text{base}}$ and $S_1'$ have the same linear dependencies, this implies that the kernel on $X_2$ derived by $N_{\text{base}}$ into $\mathcal{L}^2(\mathcal{T})$ coincides exactly with $K_2^1$, the kernel on $X_2$ that is derived by $S_1'$ into $\mathcal{L}^2(\mathcal{T}_1^1)$.

**Example 4.4.2.** Now consider $X_1 = \{a, b, c, d\}$ and $X_2 = X_1^2$ with the kernel $K_1$ on $X_1$ given by,

$$M = \begin{pmatrix} 1 & 0 & \sqrt{\frac{3}{4}} & 0 \\ 0 & 1 & \frac{1}{2} & 0 \\ \sqrt{\frac{3}{4}} & \frac{1}{2} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

This gives $\dim \mathcal{H}_{K_1} = 3$. Let $S$ be the max neural response as before, but with $\mathcal{T} = \{K_a, K_b, K_c, K_d\}$. We have the following values for $S$:

| $X_2 \backslash \mathcal{T}_1$ | $K_a$ | $K_b$ | $K_c$ | $K_d$ |
|---|---|---|---|---|
| $aa$ | 1 | 0 | $\sqrt{\frac{3}{4}}$ | 0 |
| $ab$ | 1 | 1 | $\sqrt{\frac{3}{4}}$ | 0 |
| $ac$ | 1 | $\frac{1}{2}$ | 1 | 0 |
| $ad$ | 1 | 0 | $\sqrt{\frac{3}{4}}$ | 1 |
| $ba$ | 1 | 1 | $\sqrt{\frac{3}{4}}$ | 0 |
| $bb$ | 0 | 1 | $\frac{1}{2}$ | 0 |
| $bc$ | $\sqrt{\frac{3}{4}}$ | 1 | 1 | 0 |
| $bd$ | 0 | 1 | $\frac{1}{2}$ | 1 |
| $ca$ | 1 | $\frac{1}{2}$ | 1 | 0 |
| $cb$ | $\sqrt{\frac{3}{4}}$ | 1 | 1 | 0 |
| $cc$ | $\sqrt{\frac{3}{4}}$ | $\frac{1}{2}$ | 1 | 0 |
| $cd$ | $\sqrt{\frac{3}{4}}$ | $\frac{1}{2}$ | 1 | 1 |
| $da$ | 1 | 0 | $\sqrt{\frac{3}{4}}$ | 1 |
| $db$ | 0 | 1 | $\frac{1}{2}$ | 1 |
| $dc$ | $\sqrt{\frac{3}{4}}$ | $\frac{1}{2}$ | 1 | 1 |
| $dd$ | 0 | 0 | 0 | 1 |

Note that $S(aa), S(ab), S(ac)$ are linearly independent in $\mathcal{L}^2(\mathcal{T}_1)$, as in the example before, and they are linearly independent from $S(dd)$, so $\dim \mathcal{H}_{K_2} = 4 > \dim \mathcal{H}_{K_1}$, thus $S$ is not linear.

Now decompose using the two template sets $\{K_a, K_b\}$ and $\{K_c, K_d\}$; each of these is linearly independent (since 1st and 2nd rows of $M$ are L.I. as are 3rd and 4th). We obtain the linear $S_1'$ given by the table below:

| $X_2 \backslash \mathcal{T}_1^1$ | $K_a$ | $K_b$ |
|---|---|---|
| $aa$ | 1 | 0 |
| $ab$ | 1 | 1 |
| $ac$ | 1 | $\frac{1}{2}$ |
| $ad$ | 1 | 0 |
| $ba$ | 1 | 1 |
| $bb$ | 0 | 1 |
| $bc$ | $\sqrt{\frac{3}{4}}$ | 1 |
| $bd$ | 0 | 1 |
| $ca$ | 1 | $\frac{1}{2}$ |
| $cb$ | $\sqrt{\frac{3}{4}}$ | 1 |
| $cc$ | $\sqrt{\frac{3}{4}}$ | $\frac{1}{2}$ |
| $cd$ | $\sqrt{\frac{3}{4}}$ | $\frac{1}{2}$ |
| $da$ | 1 | 0 |
| $db$ | 0 | 1 |
| $dc$ | $\sqrt{\frac{3}{4}}$ | $\frac{1}{2}$ |
| $dd$ | 0 | 0 |

This $S_1'$ maps into the same $\mathcal{L}^2(\mathcal{T}_1^1)$ as in the previous example (of dimension 2), but now $\dim \mathcal{H}_{K_1} = 3$ so we can apply Proposition 4.1.9 (since $m = 2 < 3 = \dim \mathcal{H}_{K_1}$) to show that $S_1'$ can be "replaced" by a weighted average neural response (into a different template set) - in the sense that the normalized derived kernels will coincide.

We have two templates of the second kind, $K_a$ and $K_b$ (these would be labeled $\tau_1$ and $\tau_2$ in the Proposition). Take $aa$ and $ab$ as the elements $x_1, x_2$ of the Proposition (so $n = 2$), and take $y_1 = a$, $y_2 = b$, $y_3 = d$, so the $K_{y_i}$ form a basis for $\mathcal{H}_{K_1}$. Now we are lucky, and see that $K_d$ is orthogonal to both $K_a$ and $K_b$ ($\tau_1$ and $\tau_2$). We therefore let $y = d$ and use the simple version of the construction in the Proposition. We start by setting

$$H \quad = \quad \{h_j^x \colon 1 \leq j \leq 3, x \in X_2\},$$

where,

$$h_j^x(x) = y_j$$

and for $x \neq x'$,

$$h_j^x(x') = y.$$

In fact, in this simple example, we could manage with even fewer transformations (since the span of $S_1'(X_2)$ happens to be the span of $K_a, K_b$). This will become apparent in the next paragraph but we stick with the original procedure of the Proposition for clarity.

We now determine the weights $\mu(h_j^x)$. These are the coefficients of each $(S_1')_{\text{base}}(x)$ in terms of the basis elements $K_a, K_b, K_d$. Recall, however, that $(S_1')_{\text{base}}$ is not uniquely defined in this setting (since the rank of $S_1'$ is strictly less than the dimension of $\mathcal{H}_{K_1}$). This gives us additional freedom. One simply seeks, for each row in the table for $S_1'$, any linear combination of first, second and fourth rows of the matrix $M$ which coincides at least in the first two coordinates with the given row of $S_1'$. By achieving this we will be ensuring that this linear combination takes the same values on $K_a, K_b$ as do $(S_1')_{\text{base}}(x)$ and $S_1'(x)$. One easily verifies the following linear combinations work:

| row$\backslash$L.C. | $K_a$ | $K_b$ | $K_d$ |
|---|---|---|---|
| $S'_{\text{base}}(aa)$ | 1 | 0 | 0 |
| $S'_{\text{base}}(ab)$ | 1 | 1 | 0 |
| $S'_{\text{base}}(ac)$ | 1 | $\frac{1}{2}$ | 0 |
| $S'_{\text{base}}(ad)$ | 1 | 0 | 0 |
| $S'_{\text{base}}(ba)$ | 1 | 1 | 0 |
| $S'_{\text{base}}(bb)$ | 0 | 1 | 0 |
| ... | ... | ... | |

(we give only the first part of the table to illustrate the pattern...).

For each $x \in X_2$, the row for $S(x)$ gives the coefficients $a_j(x), j = 1, 2$ in the Proposition, and $\mu(h_j^x) = a_j(x)$ for all these $x$ and $j$. For the elements shown above we thus obtain:

$$\begin{aligned}
N^{\text{avg}}(aa) &= 1K_a + 0K_b + 0K_d + R(aa)K_d = K_a + R(aa)K_d \\
N^{\text{avg}}(ab) &= 1K_a + 1K_b + 0K_d + R(ab)K_d = K_a + K_b + R(ab)K_d \\
N^{\text{avg}}(ac) &= 1K_a + \frac{1}{2}K_b + 0K_d + R(ac)K_d = K_a + \frac{1}{2}K_b + R(ac)K_d \\
N^{\text{avg}}(ad) &= 1K_a + 0K_b + 0K_d + R(ad)K_d = K_a + R(ad)K_d \\
N^{\text{avg}}(ba) &= 1K_a + 1K_b + 0K_d + R(ba)K_d = K_a + K_b + R(ba)K_d \\
N^{\text{avg}}(bb) &= 0K_a + 1K_b + 0K_d + R(bb)K_d = K_b + R(bb)K_d
\end{aligned}$$

After projection into the span of $K_a, K_b$, this gives us the same table for $N^{\text{avg}}$ as we had for $S_1'$ (on all of $X_2$).

### 4.4.1 Vision example

The above example can also be re-interpreted to provide a vision example:

**Example 4.4.3.** Suppose that $a, b, c, d$ are single pixels of color yellow, blue, green, red respectively. Let $X_1$ be the set of such $1 \times 1$ images and suppose we have a heuristic kernel $K_1$ on $X_1$ as given in Example 4.4.2. It says that $K_a$ (the similarity function for yellow) and $K_b$ (blue) are linearly independent, but $K_c$ (green) is a linear combination of these, and all three are orthogonal to $K_d$ (red). This could capture, for example, a situation where pixels classified as green are similar in a certain way to those classified as yellow and blue, but those classified as red are very distinct.

Suppose we are interested in the set $X_2$ consisting of all possible $2 \times 1$ images whose pixels are of colors yellow, blue, green, or red, and that we want to derive a kernel on the set $X_2$ - *using the max neural response* $S$ and the full set $a, b, c, d$ of templates (with uniform measure). This means that for each pair $x, x'$ of $2 \times 1$ images we will take for $K_2(x, x)$ the scalar product of their responses to the four colors:

$$K_2(x, x') = S(x)(a)S(x')(a) + S(x)(b)S(x')(b) + S(x)(c)S(x')(c) + S(x)(d)S(x')(d)$$

The results of this Chapter say that this derived kernel $K_2$ can be expressed as a sum $K_2(x, x') = K_2^1(x, x') + K_2^2(x, x')$ where each $K_2^i$ is derived (as above) via a linear neural response $S_i'$ that uses a smaller template set. As in Example 4.4.2, we may suppose $S_1'$ takes into account only the colors yellow and blue, as templates.

Moreover, $S_1'$ can be expressed as a weighted average, where there are three transformations $h_1^x, h_2^x, h_3^x$ for each element $x$ of $X_2$. $|X_2| = 16$, so using the method of the earlier example, we would associate to each possible $2 \times 1$ image $x$, 48 small images (single pixel size). To give an example, let $Y, B, G, R$ denote the possible single pixels (yellow, blue, green, red) and consider the image $x = YB$. We would have:

$$h_1^x(x) = Y, h_2^x(x) = B, h_3^x(x) = R$$
$$h_j^z(x) = R \text{ for all } z \neq x, \text{ and for } j = 1, 2, 3.$$

For a different image, say $y = BR$, we would have

$$h_1^y(y) = Y, h_2^y(y) = B, h_3^y(y) = R$$
$$h_j^z(x) = R \text{ for all } z \neq y, \text{ and for } j = 1, 2, 3,$$

in other words, the same (universal) set of small images, $\{Y, B, R\}$, would be obtained via $H$ from of each of the elements of $x_2 \in X_2$, but the actual transformation(s) producing each one would change.

Not only does this make many small images, but they are *not sub-images* of $x$. This is in contrast to the standard weighted average of vision examples, where there would likely be just 2 single pixel subimages for a $2 \times 1$ image. Let $c$ be the number of values available for each pixel and $d$ the dimension of the RKHS on the layer below. For $N \times N$ images we would be looking at $dc^{(N^2)}$ small images (transformations) in this nonstandard

weighted average that replaces a max, compared with, in general, $O(N^2)$ sub-images for the standard weighted average.

As in Example 4.4.2, the weight $\mu(h_j^x)$ associated to each $h_j^x \in H$ is the $j$'th coefficient of $(S_1')_{\text{base}}(x)$ expressed as a linear combination of $K_a$, $K_b$, and $K_d$. In this example, $\mu(h_3^x) = 0$ for all $x \in X_2$ and $\mu(h_1^x)$ and $\mu(h_2^x)$ are the first and second entries of the row for $S(x)$ in the table for $S$. This means that in practice if one wishes to replace a max neural response for a given, standard set of transformations $H_{\text{std}}$ by a weighted average as done in the Proposition, then not only would a larger set of transformations be needed but one would have to pre-compute the values that the max would have had, and use these in defining the measure $\mu_H$ for the new transformation set $H$. One is essentially encoding this data in the measure $\mu_H$. The over-all computation required would be the same, if one assumes that the new transformations are hard-wired after being determined (just as the determination of $h$'th pixel is hard-wired in a standard vision NN) and assuming that sums with zero are not performed (one may write a token null image instead of $y$ for every case $h_j^x(x')$ such that $x \neq x'$, and simply not process these further as they will not contribute to the weighted sum).

## 4.5 Collapsing of the mathematical architecture: general case

We saw in Proposition 3.11.2 of Section 3.11.1 how a three layer architecture in the CBCL framework may be replaced by a two layer architecture, with the natural transformation set – *in the case that all neural responses were weighted averages*; this also involved using templates of the second kind. We now show that the same thing holds *in general*, for any linear neural responses, and without having to resort to templates of the second kind (in fact, one may specify the templates in advance - to specific templates of the first kind, for example).

**Proposition 4.5.1.** *Consider a three-layer CBCL architecture:*

- *spaces* $X_3, X_2, X_1$,
- *templates sets* $T_2 \subset X_2$ *and* $T_1 \subset X_1$,
- *neural responses* $N_3, N_2$ *respectively into* $\mathcal{L}^2(T_2)$ *and* $\mathcal{L}^2(T_1)$.

*Assume* $N_3$ *and* $N_2$ *satisfy Axiom 3.5.1. Let* $K_2$ *and* $K_3$ *be the derived kernels on* $X_2$ *and* $X_3$ *which result. Suppose* $N_3$ *is linear (but* $N_2$ *may be nonlinear) and* $T \subset X_1$ *is any chosen template set in* $X_1$ *such that* $\{K_{1,t} \ : \ t \in T\}$ *is linearly independent and its span has at least the dimension of the span of* $\{K_{1,t} \ : \ t \in T_1\}$. *Then there exists a measure* $\nu$ *on* $T$ *and a* **linear** *neural response*

$$N_\ddagger : X_3 \to \mathcal{L}^2_\nu(T)$$

*such that* $K_3(\cdot, \cdot) = N^*_\ddagger \langle \cdot, \cdot \rangle_{\mathcal{L}^2_\nu(T)}$. *This result also holds in the extended CBCL architecture if*

$$K_2(\cdot, \cdot) = K_2^1(\cdot, \cdot) + \cdots + K_2^s(\cdot, \cdot),$$

*for some* $s$ *where each of the* $K_2^i$ *is derived by a different neural response (and templates), albeit with the same kernel* $K_1$ *on* $X_1$.

This implies:

**Corollary 4.5.2.** Assume an architecture as above, but this time, allow $N_3$ to be nonlinear as well. Let $T^1, \ldots, T^r$ be any chosen sets of templates in $X_1$, such that for each $i$, $\{K_{1,t} \ : \ t \in T^i\}$ is linearly independent and its span has at least the dimension of the span of $\{K_{1,t} \ : \ t \in T_1\}$. Then,

1. the derived kernel $K_3(\cdot,\cdot)$ on $X_3$ can be written as a sum of kernels,

$$K_3(\cdot,\cdot) = K_3^1(\cdot,\cdot) + \cdots + K_3^r(\cdot,\cdot),$$

where each $K_3^i$ is derived by a linear neural response $N_3^i : X_3 \to \mathcal{L}^2_{\nu_i}(T^i)$ with suitable measure $\nu_i$.

2. there exits a measure $\nu$ and a (generally nonlinear) neural response

$$N_\dagger : X_3 \to \mathcal{L}^2_\nu(T^1 \sqcup \cdots \sqcup T^r),$$

such that $K_3$ is derived by $N_\dagger$ from the inner product on $\mathcal{L}^2_\nu(T^1 \sqcup \cdots \sqcup T^r)$; in particular if there are enough elements in $X_1$ then this disjoint union can be chosen to be a usual template set of the first kind, i.e. without repetition: $N_\dagger : X_3 \to \mathcal{L}^2_\nu(T)$ with $T = T^1 \cup \cdots \cup T^r$.

This means that an arbitrary 3-layer NN of the CBCL model whose neural responses satisfy Axiom 3.5.1 can be replaced by a 2-layer NN of the extended CBCL model. Indeed, by Corollary 4.5.2 part 1, the top output can be expressed as:

$$K_3(x,y) = \sum_{t\in T^1} \nu_1(t) \left[\sum_{i\in[n]} a_i^1(x) K_1(x_i,t)\right]\left[\sum_{i\in[n]} a_i^1(y) K_1(x_i,t)\right] + \cdots$$
$$\cdots + \sum_{t\in T^r} \nu_r(t) \left[\sum_{i\in[n]} a_i^r(x) K_1(x_i,t)\right]\left[\sum_{i\in[n]} a_i^r(y) K_1(x_i,t)\right],$$

where $x_1,\ldots,x_n \in X_1$ are such that $K_{1,x_1},\ldots,K_{1,x_n}$ form a basis for $\mathcal{H}_{K_1}$ and each linear neural base response $N^k_{\text{base}}$ is given by its coefficients $a_i^k$ in this basis; more specifically, for each $z \in X_3$,

$$N^k_{\text{base}}(z) = \sum_{i\in[n]} a_i^k(z) K_{1,x_i}.$$

This corresponds to a 2-layer extended CBCL NN, where the first group of $t$-filters (those with $t \in T^1$) are defined by the neural response $N_3^1$, ..., and the last group of $t$-filters (with $t \in T^r$) are defined by $N_3^r$. In the next section we will strengthen this claim and show that it also holds for NN's of the *extended* CBCL model: they too may be replaced by shallow NN's of the extended CBCL model.

*Proof of Proposition 4.5.1.* Let $N_{3,\text{base}}$ be the linear base neural response underlying $N_3$,

$$N_{3,\text{base}} : X_3 \to \mathcal{H}_{K_2},$$

and let $\mathcal{T} = \{K_{2,t_1}, \ldots, K_{2,t_k}\}$ where $t_1, \ldots, t_k \in T_2$ for some $k \leq n$ such that

$$\text{span}\{K_{2,t_1}, \ldots, K_{2,t_k}\} = \text{span}(\Phi_2(T_2)).$$

Since $N_3$ is linear and maps into $\mathcal{L}^2(T_2)$, we may without loss of generality assume[5]

$$N_{3,\text{base}}(x) = \sum_{i=1}^{k} b_i(x) K_{2,t_i},$$

for some real coefficients $b_i(x)$. Assume $N_2$ decomposes into linear pieces $N_2'$ and $N_2''$ which map into $\mathcal{L}^2_{\nu_1'}(\mathcal{T}_1')$ and $\mathcal{L}^2_{\nu_1''}(\mathcal{T}_1'')$ respectively; the general case of more linear pieces is analogous. The final statement of the Proposition is also handled by this mechanism. Here, $\mathcal{T}_1' = \{K_{1,t} \ : \ t \in T_1'\}$ and likewise for $\mathcal{T}_1''$ with $T_1 = T_1' \cup T_1''$. To save space, let us denote the corresponding base neural responses into $\mathcal{H}_{K_1}$ by $N_{2,b}'$ and $N_{2,b}''$.

Fix $j \in [k]$ and let $\tau = K_{2,t_j}$. Let $x \in X_3$. We have:

$$\begin{aligned}
N_{3,\text{base}}(x)(\tau) &= \sum_{i\in[k]} b_i(x) \langle K_{2,t_i},\ K_{2,t_j} \rangle_{\mathcal{H}_{K_2}} && (4.5.1)\\
&= \sum_{i\in[k]} b_i(x) \langle N_2(t_i),\ N_2(t_j) \rangle_{\mathcal{L}^2_{\nu_1}(T_1)} \\
&= \sum_{i\in[k]} b_i(x) \langle N_2'(t_i),\ N_2'(t_j) \rangle_{\mathcal{L}^2_{\nu_1'}(\mathcal{T}_1')} \\
&\quad + \sum_{i\in[k]} b_i(x) \langle N_2''(t_i),\ N_2''(t_j) \rangle_{\mathcal{L}^2_{\nu_1''}(\mathcal{T}_1'')} && (4.5.2)\\
&= \sum_{i\in[k]} \sum_{\tau'\in\mathcal{T}'_1} b_i(x)\ N_2'(t_i)(\tau')\ N_2'(t_j)(\tau')\ \nu_1'(\tau') \\
&\quad + \sum_{i\in[k]} \sum_{\tau''\in\mathcal{T}''_1} b_i(x)\ N_2''(t_i)(\tau'')\ N_2''(t_j)(\tau'')\ \nu_1''(\tau'') && (4.5.3)
\end{aligned}$$

[5]The component of $N_{3,\text{base}}(x)$ in the orthogonal complement of $\text{span}(\Phi_2(T_2))$ may without loss of generality be assumed to be zero since it will not contribute to the value of $K_3$. Indeed, $N_{3,\text{base}}$ is only evaluated on elements of $T_2$ in determining $K_3$.

$$
\begin{aligned}
&= \sum_{i\in[k]} \sum_{\tau'\in\mathcal{T}'_1} b_i(x)\, \langle N'_{2,b}(t_i),\, \tau'\rangle_{K_1}\, N'_2(t_j)(\tau')\, \nu'_1(\tau') \\
&\quad + \sum_{i\in[k]} \sum_{\tau''\in\mathcal{T}''_1} b_i(x)\, \langle N''_{2,b}(t_i),\, \tau''\rangle_{K_1}\, N''_2(t_j)(\tau'')\, \nu''_1(\tau'') \qquad (4.5.4)\\
&= \sum_{i\in[k]} b_i(x)\, \langle N'_{2,b}(t_i),\, \underbrace{\sum_{\tau'\in\mathcal{T}'_1} N'_2(t_j)(\tau')\, \nu'_1(\tau')\, \tau'}_{=F'(\tau)}\rangle_{K_1} \\
&\quad + \sum_{i\in[k]} b_i(x)\, \langle N''_{2,b}(t_i),\, \underbrace{\sum_{\tau''\in\mathcal{T}''_1} N''_2(t_j)(\tau'')\, \nu''_1(\tau'')\, \tau''}_{=F''(\tau)}\rangle_{K_1} \qquad (4.5.5)\\
&= \sum_{i\in[k]} b_i(x)\, \langle \bar{N}'_{2,b}(t_i),\, K_{1,s_j}\rangle_{K_1} \\
&\quad + \sum_{i\in[k]} b_i(x)\, \langle \bar{N}''_{2,b}(t_i),\, K_{1,s_j}\rangle_{K_1} \\
&= N_\ddagger(x)(K_{1,s_j}). \qquad (4.5.6)
\end{aligned}
$$

where $N_{\ddagger,\mathrm{base}}(x) := \sum_{i\in[k]} b_i(x)\,[\bar{N}'_{2,b}(t_i) + \bar{N}''_{2,b}(t_i)] \in \mathcal{H}_{K_1}$ is a linear base neural response, and $s_j \in T$. The lines above are numbered for the purpose of comparing with the calculation of Proposition 3.11.2. The line between lines (4.5.5) and (4.5.6) holds due to the change of basis lemma, Lemma 4.2.1. The argument is as follows (we give it for $N'_{2,b}$; it is the same for $N''_{2,b}$).

Let $I \subseteq [k]$ such that $\{N'_{2,b}(t_i) \;:\; i \in I\}$ is a maximal linearly independent subset of the $N'_{2,b}(t_i), i \in [k]$ and let $J \subseteq [k]$ such that $\{F'(K_{2,t_j}) \;:\; j \in J\}$ is a maximal linearly independent subset of the $F'(K_{2,t_j}), j \in [k]$. Complete $\{N'_{2,b}(t_i) \;:\; i \in I\}$ to a basis for $\mathcal{H}_{K_1}$ by adding the elements $v_1, \ldots, v_d$.

Note that all images of $F'$ are linear combinations of the $\tau' \in \mathcal{T}'_1 = \Phi_1(T'_1)$ so,

$$|J| \leq \dim \mathrm{span}(\Phi_1(T'_1)) \leq \dim \mathrm{span}(\Phi_1(T_1))$$

which is at most $\dim \mathrm{span}(\Phi_1(T)) = |T|$ by hypothesis. So $|J| \leq |T|$. Pick $|J|$ distinct elements of $T$ and denote them $s_1, \ldots, s_{|J|}$. Since these are all elements of $T$, the set $\{K_{s_j} \;:\; j \in J\}$ is linearly independent.

We thus have two sets of $|J|$ linearly independent vectors, the set $\{K_{s_j} \;:\; j \in J\}$ and the set $\{F'(K_{2,t_j}) \;:\; j \in J\}$, each contained in $\mathcal{H}_{K_1}$. On the other hand, $\mathcal{H}_{K_1}$ is spanned by the linearly independent set $\{N'_{2,b}(t_i) \;:\; i \in I\} \cup \{v_1, \ldots, v_d\}$. We may therefore apply Lemma 4.2.1 to obtain new elements, denoted $\bar{N}'_{2,b}(t_i), i \in I$ (and $\bar{v}_1, \ldots, \bar{v}_d$) such that for all $i \in I, j \in J$:

$$\langle N'_{2,b}(t_i), F'(K_{2,t_j})\rangle_{K_1} = \langle \bar{N}'_{2,b}(t_i), s_j\rangle_{K_1} \qquad (4.5.7)$$

(and additional conditions on $v_1, \ldots, v_d$ which we do not need).

Now, for $p \notin I$, $N'_{2,b}(t_p)$ is a unique linear combination of the $N'_{2,b}(t_i), i \in I$ so define $\bar{N}'_{2,b}(t_i)$ to be the corresponding combination of the $\bar{N}'_{2,b}(t_i), i \in I$. It follows by linearity of the inner product that the equality (4.5.7) then holds for all $i \in [k]$ (and for all $j \in J$).

Finally, for each $p \notin J$, $F'(K_{2,t_p})$ is a linear combination of the $F'(K_{2,t_j}), j \in J$ and so the equality (4.5.7) now holds for all $i \in [k]$ and all $j \in [k]$. □

**Remark 4.5.3.** The reason we are able to get an arbitrary template set - in particular of the first kind - is that we only require that the resulting neural response be linear, not of any more special form, and so we can define it using the $\bar{N}'_2(t_i)$'s which are given to us by the Lemma.

## 4.6 Collapsing of the NN's

We will prove:

**Theorem 4.6.1.** *Any multilayer NN of the extended CBCL model can be replaced by an NN of the extended CBCL model* without hidden layers *(i.e. by a single-stage extended CBCL NN) which computes the same class of mappings. The templates may be retained.*

**Remark 4.6.2.** The over-all computation involved is the same. This is illustrated in Example 4.4.3. We do not give a formal proof, as this would require formalizing a class of circuits to be considered and is beyond the scope of the present thesis. We merely provide the example as a warning against naive extrapolation of Haastad's result.

Before proving the Theorem, we will prove a standardizing result for 2-layer extended CBCL NN's.

### 4.6.1 Changing kernels vs. changing neural responses

In the extended CBCL model, we allow different $t_i$-filters on a same level to differ. They may differ by type of neural response and/or choice of lower level kernel ($K_{\ell-1}$ in Figure 3.3). In fact, as long as we are dealing with finite sets $X_i$ (as we are when the neural network is viewed as a computational tool), all kernels can be "obtained" from a non-degenerate kernel as seen in Section 3.4.5. We show that this allows one to convert any instance of an extended CBCL NN into another one where the lowest level kernels coincide.

#### Making lower kernels coincide

**Proposition 4.6.3.** *Suppose $K_1$ is the kernel and $S_{base}$ the neural response used in all $t$-filters for $t \in T_1$ on the lowest level of an extended CBCL NN. Let $K$ be the non-degenerate*

*kernel constructed as above on $X_1$. Then we may replace each t-filter for $t \in T_1$ by a t-filter that uses $K$ instead of $K_1$. If $T_1$ is partitioned into subsets such that the $K_{1,t}$ are linearly independent in each subset then the new t-filters may be assumed to use a common base neural response for $t$ in each subset.*

*Proof.* Without loss of generality assume that the $K_{1,t}, t \in T_1$ are linearly independent (if not, then decompose $\mathcal{L}^2_\nu(T_1)$ as a direct sum as done in Corollary 4.5.2 and apply the current discussion to each of the parts - for which we do indeed have linear independence of the kind just hypothesized). Write $T_1 = t_1, \ldots, t_n$. Let $\Theta : \mathcal{H}_{K_1} \to \mathcal{H}_K$ be an injective isometry such that $\Theta : K_{1,t_i} \mapsto K_{t_i}$ for all $i \in [n]$. Now let $S'_{\text{base}}(x)(t) = \Theta[S_{\text{base}}(x)(t)], \forall x \in X_2, t \in T_1$, which is a base neural response into $\text{Maps}(X_1 \to \mathcal{H}_K)$. We have,

$$\begin{aligned}\langle S'_{\text{base}}(x)(t), K_{t_i}\rangle_{\mathcal{H}_K} &= \langle \Theta[S_{\text{base}}(x)(t)], \Theta[K_{1,t_i}]\rangle_{\mathcal{H}_K} \\ &= \langle S_{\text{base}}(x)(t), K_{1,t_i}\rangle_{\mathcal{H}_{K_1}}\end{aligned}$$

so for all $t \in T_1$, a $t$-filter defined by $S'_{\text{base}}$ and $K$ will compute the same ouput as one defined by $S_{\text{base}}$ and $K_1$, for any input $x \in X_2$. We may therefore replace the original $t$ filters by these new ones, all of which use $K$. □

This allows us to replace the various kernels used in the input layer $t$-filters of an extended CBCL NN by a common kernel $K$, by changing the base neural responses of these filters - albeit keeping groups of common neural responses common as described above.

### 4.6.2 Proof of the Theorem

Recall that a three-layer mathematical architecture in the CBCL framework corresponds to a two-stage extended CBCL network (see Section 3.12).

*Proof of Theorem 4.6.1.* Given a general 2-stage extended CBCL NN, the top stage computes a kernel $K_3$ as a sum of kernels $K_3^1 + \cdots + K_3^r$, each of which may be obtained by a different 1-stage extended CBCL NN. We consider one of these kernels and its subnetwork.

This corresponds to a 3-layer architecture which is almost like that in Corollary 4.5.2 except that the kernels at the bottom layer do not necessarily coincide. This can be arranged however by applying Proposition 4.6.3. We then use the Corollary to replace the 3-layer architecture with a 2-layer one. This corresponds to a single stage CBCL NN and so we may replace the 2-stage subnetwork under consideration by a 1-stage subnetwork.

We assume this has been done for each of the kernels $K_3^i$ and its subnetwork. This gives over-all a 1-stage extended CBCL NN which computes the same top kernel as the original 2-stage NN.

Thus, given an $n$-stage extended CBCL network, we may perform the above operation on each of its 2-stage lower subnetworks, thereby reducing the over-all depth to $(n-1)$-stages. By induction, we can therefore reduce to a single-stage extended CBCL NN.

□

# Chapter 5

# Appendix: Distinguishing Ability

We now look more closely at the ability of a derived kernel $K_\ell$ in the CBCL framework to *distinguish* between elements of $X_\ell$. This property is more commonly referred to as *selectivity* or sometimes *discrimination* and stands in contrast to *invariance*. We first define it and then show that in the CBCL model it may be conveniently characterized using linear algebra – in the case of **average neural responses** and finite spaces $X_i$.

**Definition 5.0.4** (**Basic Distinguishing).** Assuming a normalized kernel, so that

$$(\forall x)\ K_\ell(x, x) = 1,$$

we say $K_\ell$ *distinguishes* $x$ and $x'$, if

$$K_\ell(x, x') \neq 1.$$

Non-normalized kernels may be dealt with by first normalizing them so the above definition may be applied.

## 5.1 Assumed architecture and matrix notation

We assume a CBCL architecture in which $X_\ell$ and $X_{\ell-1}$ are subsequent object spaces in the hierarchy. Let,

$$\begin{aligned} X_\ell &= \{x_1, \ldots, x_n\} \\ X_{\ell-1} &= \{y_1, \ldots, y_m\} \end{aligned}$$

Then there is a natural **matrix associated to each kernel**, given by

$$\begin{aligned} (M_\ell)_{ij} &= K_\ell(x_i, x_j) \\ (M_{\ell-1})_{ij} &= K_{\ell-1}(y_i, y_j). \end{aligned}$$

These are respectively $n \times n$ and $m \times m$ symmetric positive semidefinite matrices. Each $K_\ell$ is uniquely determined by $M_\ell$. Let

$$\begin{aligned} u_i &\quad \text{be the} \quad i\text{'th column of } M_\ell, \text{ and} \\ v_i &\quad \text{be the} \quad i\text{'th column of } M_{\ell-1}. \end{aligned}$$

These are vertical (column) vectors. Their transposes are the rows of the respective matrices (since symmetric).

### 5.1.1 Average neural responses

In the case of **average neural responses**, we assume a finite transformation set $H$ and a measure $\mu$ on $H$ with

$$N_\ell^{\text{avg}}(f)(g) = \sum_{h \in H} \mu(h) K_{\ell-1}(f \circ h, g).$$

We have omitted the subscript on $H$ since we only go between these two layers.

**Remark 5.1.1.** The average neural response $N_\ell^{\text{avg}}(x)(y)$ is actually defined *for all* $x \in X_\ell$ and *for all* $y \in X_{\ell-1}$. This general fact about base neural responses was commented on in Remark 3.5.2. This wider definition is also important to bring out the structure of the hierarchy as regards distinguishing power.

Given the above definition of $N_\ell^{\text{avg}}$ we define $W$, the **matrix of weights**, by

$$W_{ij} = \sum_{\substack{h \in H \text{ s.t.} \\ x_i \circ h = y_j}} \mu(h).$$

$W$ captures the joint effect of $H$ and the weights used in averaging. Rows of this matrix will be denoted $w_i$ (horizontal vector). Using this notation, the following fact is immediate:

$$N_\ell^{\text{avg}}(x_i)(y_k) = w_i \cdot v_k. \tag{5.1.1}$$

This implies:

**Lemma 5.1.2.** *If we view $N_\ell^{avg}(x_i)$ as a row vector of length $m = |X_{\ell-1}|$, its $k$'th coordinate is:*

$$W v_k$$

*i.e. its coordinates are the images under a* **linear transformation** *of the columns of $M_{\ell-1}$.*

## 5.2 Derived kernels in matrix notation

To define the kernel $K_\ell$ we use the template set $T \subseteq X_{\ell-1}$ and retain only those coordinates of $N_\ell(x_i)$ which correspond to the elements of $T_{\ell-1}$. Let $N_{\ell,T}(x_i)$ denote this new (typically shorter) vector and $R_{\ell,T}$ the matrix whose rows are the $N_{\ell,T}(x_i)$. Then,

**Lemma 5.2.1.** *For any neural response (not necessarily the average), we have*

$$M_\ell = (R_{\ell,T})(R_{\ell,T})^T.$$

**Proposition 5.2.2.** *Suppose $K_\ell$ is normalized, so $(\forall x)\ K_\ell(x,x) = 1$, and that it is defined by the template set $T$, then the following are equivalent (and all correspond to "$K_\ell$ does* **not** *distinguish $x_i$ and $x_j$"):*

1. $K_\ell(x_i, x_j) = 1$
2. $N_\ell(x_i)(y) = N_\ell(x_j)(y)$ *for all* $y \in T \subseteq Im(v_{\ell-1})$
3. *the* $2\times 2$ *submatrix of* $M_\ell$ *corresponding the* $i,j$*'th rows/columns has all entries equal to* 1
4. *the* $2 \times 2$ *submatrix of* $M_\ell$ *corresponding the* $i,j$*'th rows/columns is singular*
5. *(in the case of the average neural response)* $w_i \cdot v = w_j \cdot v$ *for all vectors* $v$ *in the span of the columns of* $M_{\ell-1}$ *corresponding to the elements of* $T$*.*

*Proof.* The first two points are equivalent since $M_\ell$ is an inner product matrix of the vectors $N_{\ell,T}(x_k)$ for $k = 1, \ldots, n$ (by Lemma 5.2.1) with ones on the diagonal (by normalization) and thus $K_\ell(x_i, x_j) = 1$ implies $N_{\ell,T}(x_i) = N_{\ell,T}(x_j)$ by Cauchy-Schwarz.

This viewpoint also shows equivalence with point 3 and 4.

The 2nd and 5th point are equivalent by Lemma 5.1.2.

□

## 5.3 Distinguishing ability of the average neural response

The second and fifth points of Lemma 5.2.2 immediately imply

**Corollary 5.3.1.** *Assuming a weight matrix $W$, and assuming a fixed kernel $K_{\ell-1}$, the truth value of*

$$N_\ell^{avg}(x_i) = N_{\ell,T}^{avg}(x_j)$$

*depends only on*

$$span\{v_i \mid y_i \in T\}.$$

*Therefore, the* **maximum distinguishing ability** *of $K_\ell$ is achieved by choosing a template set $T \subseteq Im(v_{\ell-1})$ such that* $\text{span}\{K_{\ell-1,t} \mid t \in T\} = \mathcal{H}_{K_{\ell-1}}$. *Choosing more vectors does not help.*

One may then ask what can be accomplished by changing $W$.

### 5.3.1 Relationship between W and distinguishing ability

Given our notation and the fact the sets $X_i$ are finite, each set $T$ of $r$ elements in $X_{\ell-1}$ corresponds uniquely to a subset $I_T$ of $[m]$ consisting of the indices of the elements $x_1, \ldots, x_m$ of $X_{\ell-1}$ which occur in $T$. And the vectors $K_{\ell-1,t}, t \in T$ are linearly independent in $\mathcal{H}_{K_{\ell-1}}$ if and only if the columns $v_i, i \in I_T$ are linearly independent.

**Proposition 5.3.2.** *Assume $T$ corresponds to a maximal linearly independent set of columns of $M_{\ell-1}$. Then $x_i$ and $x_j$ are indistinguishable by $K_\ell$ iff $(w_{T,i} - w_{T,j})^T$ is not in the span of columns of $M_{\ell-1}$. Note this condition does not depend on which maximal linearly independent $T$ we take.*

*Proof.*

$$\begin{aligned}
& w_i(v) = w_j(v)\ \forall v \text{ columns of } M_{\ell-1} \text{ corresponding to elements of } T \\
\Leftrightarrow\ & w_i(v) = w_j(v)\ \forall v \text{ columns of } M_{\ell-1} \\
\Leftrightarrow\ & M_{\ell-1}(w_i)^T = M_{\ell-1}(w_j)^T \\
\Leftrightarrow\ & M_{\ell-1}((w_i - w_j)^T = 0 \\
\Leftrightarrow\ & (w_i - w_j)^T \notin \text{ span of columns of } M_{\ell-1}
\end{aligned}$$

where the last line follows from the fact that a symmetric matrix is diagonalizable and so its nullspace and range can only intersect in $\{0\}$. □

# Bibliography


[1] Y. Amit, *A neural network architecture for visual selection.* Neural Computation (2000).

[2] E. Bernstein, Y. Amit, *Statistical Models for Object Classification and Detection.* Computer Vision and Pattern Recognition (CVPR), 734-740, (2005).

[3] C. Bishop, *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[4] G. Cybenko, *Approximation by superpositions of a sigmoidal function.* In Mathematics of Control, Signals, and Systems, 2(4):303314, (1989).

[5] F. Cucker and S. Smale, *On the Mathematical Foundations of Learning.* Bulletin of the American Mathematical Society, 39:1-49, (2002).

[6] K. Fukushima, *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.* Biological Cybernetics, 36(4):193-202, (1980).

[7] K. Fukushima, S. Miyake, and T. Ito. *Neocognitron: a neural network model for a mechanism of visual pattern recognition.* IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-13, 3:826834, (1983).

[8] J. Håstad, *Computational limitations of small depth circuits*, PhD thesis, MIT, 1987.

[9] K. Hornik, M. Stinchcombe and H. White, *Multilayer Feedforward Networks are Universal Approximators.* Neural Networks, 2:359-366, (1989).

[10] D. H. Hubel and T. N. Wiesel, *Receptive Fields Of Single Neurones In The Cat's Striate Cortex.* In Journal of Physiology, 48:574-59I, (1959).

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 86(11), 22782324, (1998).

[12] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, *Backpropagation applied to handwritten zip code recognition*. In Neural Computation, 1(4):541551, (1989).

[13] M.L. Minsky and S.A. Papert, Perceptrons. Cambridge, MA, MIT Press, 1969.

[14] W. McCulloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity*. In Bulletin of Mathematical Biophysics, 7:115 - 133, (1943).

[15] A. B. Novikoff, *On convergence proofs on perceptrons*. Symposium on the Mathematical Theory of Automata, 12:615-622, Polytechnic Institute of Brooklyn, (1962).

[16] D. Perrett, M. Oram, *Neurophysiology of shape processing*. Imaging Vis. Comput. 11:317333, (1993).

[17] M. Riesenhuber and T. Poggio, *Hierarchical models of object recognition in cortex*. Nature Neuroscience, 2(11):1019-1025, (1999).

[18] F. Rosenblatt, *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*. Cornell Aeronautical Laboratory, Psychological Review, 65(6):386408, (1958).

[19] F. Rosenblatt, *Principles of Neurodynamics*. Washington, DC, Spartan Books, 1962.

[20] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, T. Poggio, *Robust object recog- nition with cortex-like mechanisms*. IEEE Trans. Pattern Anal. Mach. Intell., 29(3):411426, (2007).

[21] S. Smale, L. Rosasco, J. Bouvrie, A. Caponnetto, and T. Poggio, *Mathematics of the Neural Response*. Foundations of Computational Mathematics, 10(1):67-91, (2010).

[22] G. Wallis, E. Rolls, *A model of invariant object recognition in the visual system*. Prog. Neurobiol. 51:167194, (1997).