Quadratic Forms in Convolutional Neural Networks

Peter Adema
14460165

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 900
1098 XH Amsterdam

Supervisor
Dr. ir. R. van den Boomgaard

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 900
1098 XH Amsterdam

Semester 2, 2024-2025

Abstract TODO

Acknowledgements TODO

Contents

Chapter 1

Introduction

In the field of computer vision, machine learning has been successfully applied for societal benefit in various ways, ranging from analysing medical imaging data [1, 2] to classifying agricultural produce [3, 4] and recognising license plate numbers [5]. One of the primary tools used in these applications is the Convolutional Neural Network (CNN) [6], a type of neural network that leverages existing theoretical knowledge of image processing to learn representations of images more efficiently.

Two components are often seen within a typical CNN: the titular convolutional layer, which analyses the image, and a pooling layer, which compresses the image representation (see [7] for an introduction). The latter pooling layer is often implemented as a max-pooling layer, which selects the highest value in a small area around every point in the image. Slightly more generally, a max-pooling layer can be seen as an operation that weighs neighbours around a point in the image and selects the neighbour with the highest weight, but with all neighbours being weighed equally. However, this general perspective suggests the possibility of a variation on a max-pooling layer where pixels are not weighed equally: perhaps pixels further away from the centre are considered less in the selection for the maximum.

This idea has been formalised in mathematical morphology as dilation [8] and in tropical geometry as a max-plus convolution [9]. Using this formalism, a separate function G (the structuring element or kernel) defines the weighting for neighbouring pixels. This function can be parameterised in various ways, but a concave function centred around the origin is typically used for G. One such function is the quadratic function $f(x) = x^T\Sigma^{-1}x$, and another is its isotropic form $f(x) = x^T(sI)^{-1}x$, with parameters Σ and s respectively [10].

Previous studies [11] and theses [12, 13] have investigated the possibility of using such a dilation (generalised max-pooling) with an isotropic quadratic structuring element as a layer within a CNN, with the parameter s being learned via gradient descent (a standard optimisation method within machine learning). These previous works showed that using an isotropic quadratic kernel (which is strictly more expressive than a standard max-pooling) resulted in higher performance on a small selection of datasets. This thesis aims to expand upon this by examining using an anisotropic quadratic structuring element within the dilation, specifically whether the anisotropic parameters Σ could also be learned via gradient descent. The expectation would be that since anisotropic quadratic kernels are again strictly more expressive than the isotropic versions, such a dilation layer would further improve performance.

1.1 Related work TODO

For now, I've just copied my literature survey from the project proposal.
I intend to rewrite this evenutally, though: no need to review this for now.

Modern Convolutional Neural Networks (CNNs) often use linear convolutional layers to process images and max-pooling layers to condense information and shrink the feature space [7]. However, both of these operations are equivalent to a semifield convolution: the first in the linear field (with a learned kernel) and the second in the tropical-max field (with a step-function-like kernel) [14]. In [14], Bellaard et al. provide an axiomatic foundation for using various semifields within the context of PDE-CNNs but do not discuss using semifields for conventional (discrete) CNNs.

The ideas underlying the usage of tropical fields are, however, older than [14]: the field of mathematical morphology researches the shapes and forms of objects and functions, and two of the core operators within mathematical morphology are dilation (equivalent to a tropical-max correlation) and erosion (equivalent to a tropical-min correlation) [9]. Heijmans [8] is an excellent treatment of many of the theoretical fundaments and generalised cases of morphology, with Chapter 11 describing morphology for grey-scale images (most similar to the convolutional operations relevant to this project). Furthermore, morphological operations with specifically a quadratic structuring element were researched by Boomgaard in [10] and other papers, showing that many aspects of the resulting calculation can be performed in closed form without first approximating the quadratic into a fixed-size kernel.

Another paper of note regarding the efficient calculation of the convolutional stencil in tropical semifields may be [15], in which Geusebroak and van de Weijer discuss how to perform an efficient calculation for the linear field with a Gaussian kernel. Whether such methods will be needed within this project is yet unclear, but the notes regarding approximate separability by reorienting the axes of a quadratic show a possible direction for further performance improvements.

Besides relevant theory, there are also some more recent pieces of literature somewhat close to this topic. Notably, [16, 17] show that a CNN that learns quadratic scale parameters for the kernels of its linear convolution can, in some cases, learn to perform tasks similar to those of a CNN that directly learns all kernel parameters. This adjustment significantly reduces the parameters required for the linear convolutions replaced in such a way. Furthermore, it is equivalent to the original Bachelor project proposal, where the task would have been to parameterise a linear convolution with the PDF of a Gaussian.

Finally, previous projects under Dr. Boomgaard have partly investigated discrete subfield convolutions. The isotropic case (where scales are uncorrelated and bound between dimensions) for tropical-max fields has been relatively well-researched by [12, 13], showing minor performance increases in basic vision models. However, a more general treatment of anisotropic kernels in tropical max semifields (and other fields) is not yet present within either the public domain or the UvA collection of theses.

Chapter 2

Background

First, mathematical formalisms must be covered to understand the concepts discussed in later sections. These include the convolutional operator, its generalisation using semifields, and relevant semifields from mathematical morphology. Subsequently, we will discuss variations on the convolutional operator, as well as quadratic distance functions and a method for learning the positive definite matrices parameterising them.

2.1 Convolutional operator

At the core of a convolutional neural network is the convolutional operation $f * g$. The general form of a continuous convolution of functions f and g can be written as:

$$(f * g)(x) = \int_{y \in \mathcal{D}} f(x - y)\ g(y), \tag{2.1}$$

where $\mathcal{D}$ is the (continuous) domain of f and g. However, save for a handful of functions whose convolutions can be calculated algebraically, we are typically required to approximate this convolution in the discrete domain. In this case, we approximate the functions f and g by sampling them at fixed intervals, the result of which can be represented as discrete arrays F and G. A discrete convolution could then be written as [18]:

$$(F * G)[x] = \sum_{y \in \mathcal{Y}} F[x - y]\ G[y], \tag{2.2}$$

where $\mathcal{Y}$ is the set of all indices in the domain of F and G (e.g. $\mathcal{Y} = \mathbb{Z}^2$ for infinitely large 2D images and kernels), and G is typically referred to as the (convolutional) kernel.

2.2 Fields and semifield convolutions

In the convolutional operator, a part of the image is repeatedly multiplied element-wise with a kernel, and the resulting values are summed to obtain an activation for each point. However, while we typically use scalar addition and multiplication in this calculation, it is also possible to use different operators in the reduction by defining a different field in which the reduction is done.

In this section, we will briefly look at the concept of fields insofar as they are relevant to implementing an alternative version of the convolutional operator.

In mathematics, a field is a set of values with a pair of operators: one operator corresponds to the concept of addition, and one operator corresponds to the concept of multiplication. Fields are, in effect, a generalisation of standard addition and multiplication on integers or reals that allow for describing a set of values other than typical scalars or an alternate method for combining typical numbers. Formally, a field can be described as a tuple $(\mathcal{F}, \oplus, \otimes)$, where the operators $\oplus$ and $\otimes$ are of the type $\mathcal{F} \times \mathcal{F} \to \mathcal{F}$. Furthermore, the operators $\oplus$ and $\otimes$ are both beholden to the field axioms: informally, a set of rules to ensure they act 'similarly' to standard scalar addition and multiplication.

These field axioms can be written as (adapted from [19] and [14]):

$$\oplus \text{ is associative: } \forall a, b, c \in \mathcal{F} \qquad a \oplus (b \oplus c) = (a \oplus b) \oplus c \tag{2.3}$$

$$\otimes \text{ is associative: } \forall a, b, c \in \mathcal{F} \qquad a \otimes (b \otimes c) = (a \otimes b) \otimes c \tag{2.4}$$

$$\oplus \text{ is commutative: } \forall a, b \in \mathcal{F} \qquad a \oplus b = a \oplus a \tag{2.5}$$

$$\otimes \text{ is commutative: } \forall a, b \in \mathcal{F} \qquad a \otimes b = a \otimes a \tag{2.6}$$

$$\oplus \text{ has an identity: } \exists \mathbb{0}\ \forall a \in \mathcal{F} \qquad a \oplus \mathbb{0} = a \tag{2.7}$$

$$\otimes \text{ has an identity: } \exists \mathbb{1}\ \forall a \in \mathcal{F} \qquad a \otimes \mathbb{1} = a \tag{2.8}$$

$$\oplus \text{ has inverse elements: } \forall a\ \exists b \in \mathcal{F} \qquad a \oplus b = \mathbb{0} \tag{2.9}$$

$$\otimes \text{ has inverse elements: } \forall a\ \exists b \in \mathcal{F} \qquad a \otimes b = \mathbb{1} \tag{2.10}$$

$$\mathbb{0} \text{ is absorbing for } \otimes\text{: } \forall a \in \mathcal{F} \qquad a \otimes \mathbb{0} = \mathbb{0} \tag{2.11}$$

$$\otimes \text{ distributes over } \oplus\text{: } \forall a, b, c \in \mathcal{F} \qquad a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \tag{2.12}$$

One use case for fields in machine learning is to describe a weighted reduction (as in a kernel-based convolution) more generally. To better understand this, let us return to the example of the convolutional operator. In this case, we must reduce the neighbourhood around a pixel x in the image F, weighted by the values in the kernel G. While this would typically be written as:

$$(F * G)[x] = \sum_{y \in \mathcal{Y}} F[x - y]\ G[y], \tag{2.13}$$

we could instead use a field $S = (\mathcal{F}, \oplus, \otimes)$ and write a similar operation:

$$(F \circledast_S G)[x] = \bigoplus_{y \in \mathcal{Y}} F[x - y] \otimes G[y] \tag{2.14}$$

Performing this operation does not, strictly speaking, require any of the above field axioms to hold for $\oplus$ or $\otimes$: the only relevant restriction would be that of the types being $\mathcal{F} \times \mathcal{F} \to \mathcal{F}$. Implementing this operation for a CNN would further require both operators to be differentiable, and for $\mathbb{0}$ to exist (to deal with border effects in a convolution). However, it is generally also useful for the reduction operator $\oplus$ to be both associative and commutative, as this makes the order of elements in the input irrelevant, promoting the stability of the result and allowing efficient parallel implementation of the reduction [20]. Furthermore, most of the field axioms hold regardless for most operators we wish to use for $\oplus$ and $\otimes$. As such, we will use a semifield [14]:

A semifield is a field, except an additive ($\oplus$) inverse (Eq. 2.9) does not necessarily exist.

We can then define the operator $\circledast_S$ for the following sections (identically to Eq. 2.14) as the semifield convolutional operator for any semifield L.

2.3 Semifields from mathematical morphology

Knowing that an operator similar to convolution can be used in any semifield, we can examine if there are relevant semifields in which we can perform a convolution other than the standard linear field. For this, we can take inspiration from mathematical morphology, the study of object and function shapes.

Two core classes of discrete operators from mathematical morphology are dilations and erosions, where dilations informally correspond with 'making a function larger' (scaling the umbra of a function), and erosions with 'making a function smaller'. The result of the common dilation is shown in Fig. 4.3. Examining the local effects of the dilation more closely, we can see that it is somewhat similar to taking a local maximum, which can also be seen in the formula for this dilation operator $\boxplus$ (from [8], using the Minkowski sum):

$$F \boxplus G, \text{ where } (F \boxplus G)[x] = \bigvee_{y \in \mathcal{Y}} (F[x-y] + G[y]) \tag{2.15}$$

Here, F is the image (or object or sampled function) to be dilated, G is a structuring element describing how the dilation will occur, and $\bigvee$ denotes the supremum. If we further restrict F and G to be of finite size, then $\mathcal{Y}$ will be of finite size, and the correspondence with the local maximum becomes exact:

$$\text{For finite-size } F \text{ and } G : (F \boxplus G)[x] = \max_{y \in \mathcal{Y}} (F[x-y] + G[y]) \tag{2.16}$$

An intuitive explanation would be to see the structuring element G as a (negated) distance function and the dilation $\boxplus$ as the operation that takes the highest value weighted by how 'close' it is to x. If G is a step function with value zero near its centre and $-\infty$ outside (Fig. 4.3, first row), we can see that this is precisely taking the maximum value in the area where G is zero. However, we may also wish to use a quadratic (Fig. 4.3, second row) or other function as G. Depending on G, we may still be able to perform the dilation exactly (using algebraic solutions and/or leveraging separability), but to compute the dilation in the general case, we may wish to clip G to be above $-\infty$ in only a constrained domain (Fig. 4.3, third row). The dilation with the clipped version of $F \boxplus G_{clipped}$ could then be seen as an approximation of the dilation $F \boxplus G$ with the full (unclipped) G while having the advantage that the set of relevant indices $\mathcal{Y}$ is bounded in size. It should be noted, however, that if F is K-Lipschitz and G is strictly decreasing with the second derivative $|\delta^2 G| > K$ outside a central area, then clipping G to this central area will not change the result of the dilation ($F \boxplus G_{clipped}$ is then exact, see Appendix 6.2).

Looking at the operation performed by dilation more closely, we can see that it is, in effect, a maximum operation weighted by a distance function. Similarities with the weighted reduction in the semifield convolution $\circledast_S$ may lead us to believe that this can also be viewed as a convolution in the appropriate semifield S, and this is indeed the case. By defining $\oplus = \max$ and $\otimes = +$, we obtain the tropical max semifield $T_+ = (\mathbb{R} \cup \{-\infty\}, \max, +)$ with neutral elements ($\mathbb{0} = -\infty, \mathbb{1} = 0$) [9, 14]. A convolution $F \circledast_{T_+} G$ in this tropical semifield T_+ (also known as a max-plus convolution [9]) would then be $\boxplus$:

$$\begin{aligned}
(F \circledast_{T_+} G)[x] &= \bigoplus_{y \in \mathcal{Y}} F[x-y] \otimes G[y] \\
&= \max_{y \in \mathcal{Y}} (F[x-y] + G[y]) \\
&= (F \boxplus G)[x]
\end{aligned}$$

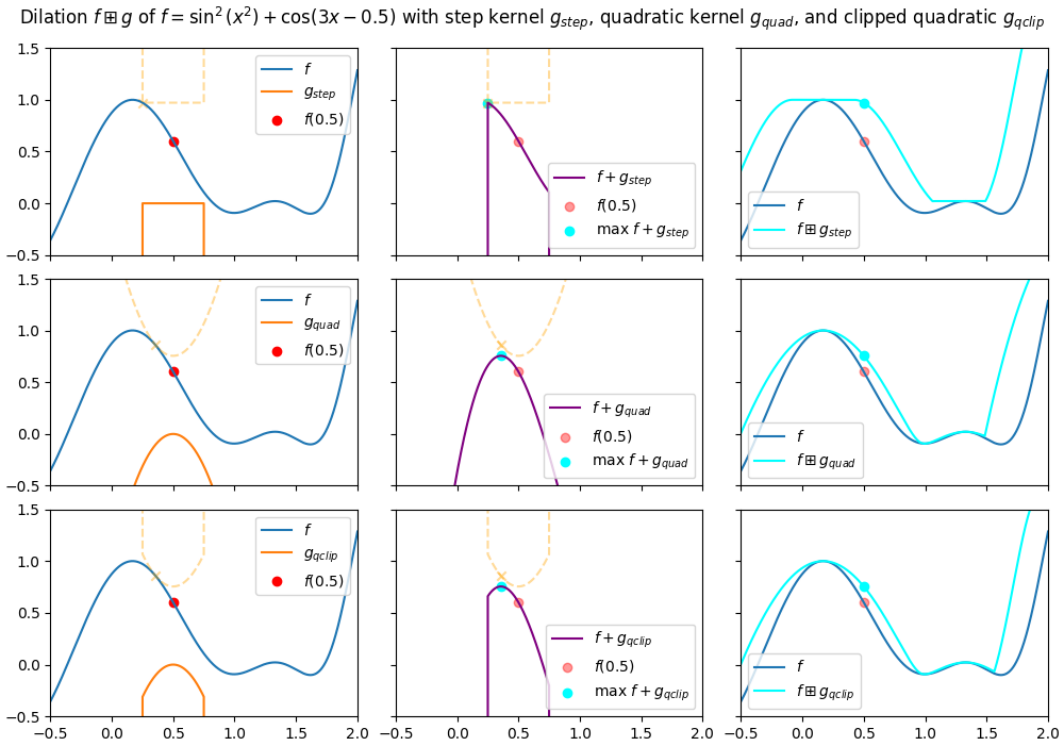

Figure 2.1: Illustration of the effects of the dilation $\boxplus$ with three kernels on a sinusoidal f. $g_{step} = 0$ in a region of size 0.5 and $-\infty$ outside, while $g_{quad} = -5x^2$ and $g_{qclip} = g_{quad} + g_{step}$. An alternative intuition for $\boxplus$ is also illustrated, corresponding with 'lowering' a negated version of g' down towards the point x until it intersects f and taking the value of the lowered and flipped $g'(x)$ as the result of $\boxplus$ at point x. Note the continuous, thus lowercase f and g.

This result is interesting because we can see the standard max pooling layer in a convolutional neural network as a dilation with a fixed, step-function-like G (a 2D version of G_{step} from Fig. 4.3). Generalising G to be a quadratic form is a logical next step, where this thesis focuses on the anisotropic case.

In mathematical morphology, dilations and erosions come in pairs named adjunctions. It can be shown that the erosion which adjoins $\boxplus$ (henceforth referred to as the operator $\boxminus$) is effectively a minimum (infimum in the infinite case) weighted by a (negated) distance function [8]:

$$(F \boxminus G)[x] = \min_{y \in \mathcal{Y}} (F[x + y] - G[y]) \tag{2.17}$$

Using similar logic as above, we can show that, in the corresponding tropical min semifield $T_- = (\mathbb{R} \cup \{\infty\}, \min, +)$[1] [9] with neutral elements $(\mathbb{0} = \infty, \mathbb{1} = 0)$, the convolution with $\check{G}^*(x) = -G(-x)$ is equivalent to the erosion $\boxminus$:

$$\begin{aligned}
(F \circledast \check{G}^*(x))[x] &= \bigoplus_{y \in \mathcal{Y}} F[x - y] \otimes \check{G}^*(x)[y] && (2.18)\\
&= \min y \in \mathcal{Y}(F[x - y] + \check{G}^*(x)[y]) && (2.19)\\
&= \min y \in \mathcal{Y}(F[x - y] - G[-y]) && (2.20)\\
&= \min y^* \in \mathcal{Y}(F[x + y^*] - G[y^*]) && (2.21)\\
&= (F \boxminus G)[x] && (2.22)
\end{aligned}$$

[1]For both T_+ and T_-, it should be noted that standard addition $+$ is undefined for $\pm\infty$. In accordance with [9], we define $\forall x : x + (-\infty) = (-\infty)$ in T_+, while $\forall x : x + \infty = \infty$ in T_-

2.4 Further relevant mathematical morphology

Besides the common adjoint $(\boxplus, \boxminus)$ using $+$ and $-$ as the operators weighing each point, one can also define an adjoint that uses multiplication and division. This alternate adjoint on $\mathbb{R}_+$ then has the dilation $\dot{\boxplus}$, defined as [8]:

$$(F \mathbin{\dot{\boxplus}} G)[x] = \bigvee_{y \in \mathcal{Y}} F[x - y]G[y] \tag{2.23}$$

However, it can be shown that this adjoint is equivalent (anamorphic) to the common adjoint $(\boxplus, \boxminus)$ using the transformation $t(x) = e^x$ and its inverse $t^{-1}(x) = \log(x)$. As such, using $\dot{\boxplus}$ would not result in a more expressive convolution while complicating implementation with the requirement of $\mathbb{R}_+$.

Two other concepts from mathematical morphology that do increase expressivity are the concepts of openings and closings [8, 21]. Here, an opening is an operator using an adjoint, where the input is first eroded and then dilated with the same structuring element G. Notably, an opening is anti-extensive (decreasing), meaning that for $(\boxplus, \boxminus)$ we can write the opening $\square$:

$$\forall x \in \mathcal{Y} \quad (F \mathbin{\square} G)[x] = ((F \mathbin{\boxminus} G) \mathbin{\boxplus} G)[x] \leqslant F[x] \tag{2.24}$$

The operator dual to an opening is the closing using the same adjoint: here, a closing refers to first performing a dilation, and then performing the adjoining erosion with the same structuring element G. Notably, a closing is extensive (increasing), meaning that for $(\boxplus, \boxminus)$ we can write the closing $\blacksquare$:

$$\forall x \in \mathcal{Y} \quad (F \mathbin{\blacksquare} G)[x] = ((F \mathbin{\boxplus} G) \mathbin{\boxminus} G)[x] \geqslant F[x] \tag{2.25}$$

One of the possible advantages of using closing and opening over only dilation or erosion is that opening and closing both produce results with contours similar to their inputs [21]. Illustrating the effects of opening and closing with a simple diagonal kernel in Figure 2.2, we can see that the results of the closing have a higher value (due to the extensivity of $\blacksquare$), but the shapes are less distorted by the shape of the kernel G when compared with the results of the dilation.

This property, when combined with the extensivity of $\blacksquare$, suggests it can also be used in the place of a dilation within the context of max-pooling for CNNs. As such, we will later apply the closing $\blacksquare$ in that context, implemented as not a single semifield convolution $\circledast$, but a pair in T_+ and T_- respectively.

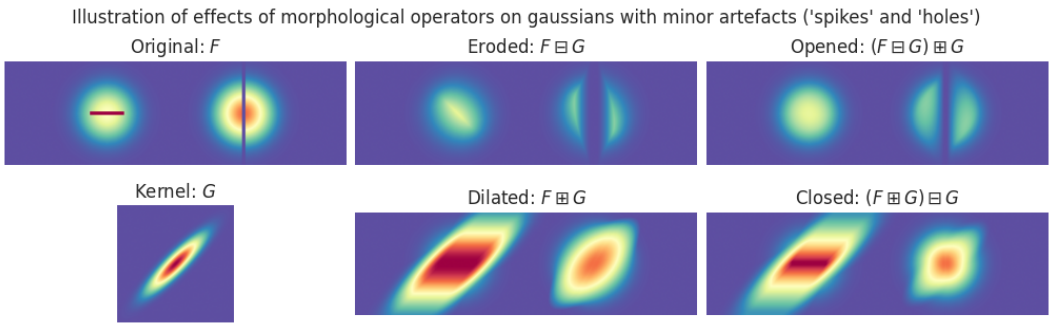

Figure 2.2: Illustration of the effects of opening and closing, where warmer (red) colors indicate higher values, and colder (blue) colors indicate lower values.

2.5 Variations on the convolution in CNNs

Within a Convolutional Neural Network, there are layers typically referred to as convolutional layers. These layers apply an operation very similar to a discrete mathematical convolution $*$ of the input activations (or image) and a parameterised kernel (here with an explicit domain $\mathcal{X}$ for valid values of x):

$$\forall x \in \mathcal{X} : \quad (F * G)[x] = \sum_{y \in \mathcal{Y}} F[x - y]\, G[y] \tag{2.26}$$

However, a CNN convolution has additional parameters that may change its behaviour. To better understand how we can apply semifield convolutions $\circledast$ in CNNs, we must therefore understand these parameters, usually named 'stride', 'dilation', 'padding' and 'groups' in modern deep learning frameworks [22, 23]. Additionally, images and activations in CNNs have 'channels': an additional axis in the input, which is treated differently from the spatial ones. We will first discuss stride, dilation and padding, which do not interact with channels, and then discuss the concept of channels and convolutional groups.

Stride controls the spacing of the sampling grid for F with respect to the output position x: where a regular convolution uses $F[x-y]$, a strided convolution would have $F[\text{STRIDE} \times x - y]$. Striding can also be seen as 'moving' the receptive field (accessed values of F) of the convolution with a step size equal to the stride: see Fig. 2.3. For finite images, this reduces the size of the convolution result by shrinking the set of valid indices $\mathcal{X}$. Therefore, a $\text{STRIDE} > 1$ can allow later convolutions to be influenced by more inputs without increasing kernel sizes. A standard convolutional layer typically has a stride of 1, but a pooling layer might have a stride of 2 or higher.

Dilation is similar, but instead adjusts the step size for y: the term $F[x-y]$ becomes $F[x - \text{DILATION} \times y]$, creating 'gaps' between the sampling points for F (e.g. for $x = 1$, sampling $F[-1]$, $F[1]$, $F[3]$) without changing the sampling for the kernel G. However, a $\text{DILATION} > 1$ results in a locally non-smooth operator, (which seems undesirable for smooth pooling). As such, none of the later experiments investigated using a $\text{DILATION} > 1$.

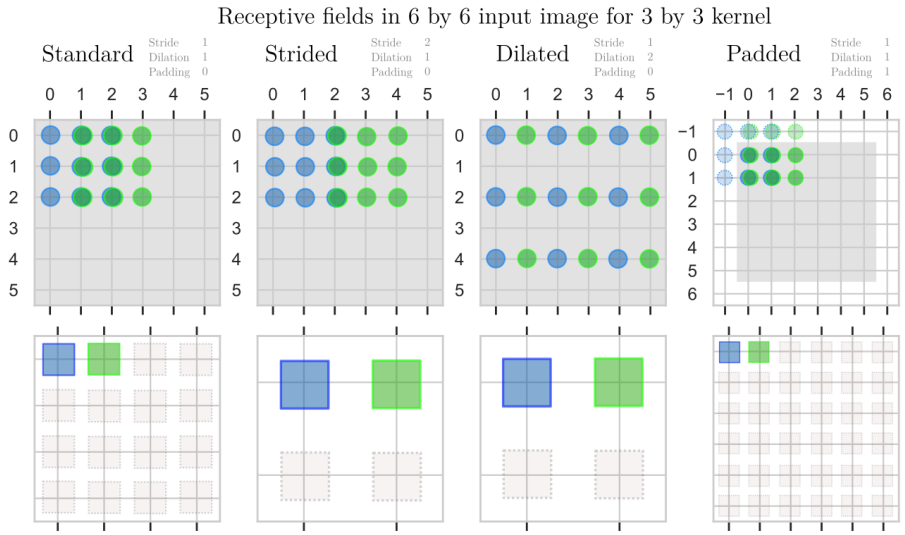

Figure 2.3: Illustration of the effects of stride, dilation and padding with a 3×3 kernel on a 6×6 input. The top row shows the receptive field of the reduction (accessed values of F), while the bottom row shows the output structure.

Padding refers to convolving with a widened F: if a certain F has domain $[0, 5]$, then a padding of 1 would correspond with convolving with an expanded F' with domain $[-1, 6]$ that is equal to F within $[0, 5]$. While there are many so-called border strategies [21] in computer vision for how to determine the value of the newly added $F'[-1]$ and $F'[6]$, the method typically used within CNNs is to set these new values to 0, thereby keeping the sum unchanged. Our use of padding lies in ensuring that the size of the output image does not shrink with kernel sizes larger than 1: this simplifies network structures by ensuring the output size decreases only if STRIDE > 1. For an odd-valued kernel dimensionality K_o (such as 3×3, see Fig. 2.3), this can be achieved by setting the padding to $\lfloor \frac{K_o}{2} \rfloor$. Even-valued kernel dimensionalities K_e depend on where we define the centre of the kernel to lie. Suppose we define it as towards the lower indices in the kernel (up and to the left in 2D images) and allow for different amounts of padding at the beginning and end of F: in that case, we can retain image sizes by padding with $\frac{K_e}{2} - 1$ at the low-index side of F (top/left) and padding with $\frac{K_e}{2}$ at the high-index side of F (bottom/right).

Besides modifying the receptive field of the convolution, another way in which CNN convolutions can diverge from mathematical ones is in their usage of a channel axis, distinct from the spatial axes. A small 2D RGB image might be stored as a $3 \times 50 \times 50$ array, but while the output index x in a CNN convolution would represent a position in the X- and Y- axes of the image, the channel (colour) axis is not indexed. Instead, every convolutional kernel is responsible for one channel in the output and reads from a fixed set of channels in the input: while the kernel is 'moved' through the spatial dimensions during the convolution, it stays fixed relative to the channel dimensions. In a standard CNN convolution, all kernels read from all input channels, meaning that a square convolution of size 5 in this RGB image would require a convolutional kernel of size $3 \times 5 \times 5$ (as the kernel reads from 3 channels). In contrast, a typical max-pooling works per channel: every convolution operation only reads from one input channel. However, both cases can be seen as an adjusted version of the standard convolution formula. For a linear convolution, we simply add a summation over the set of input channels $\mathcal{C}$, indexing both F and G [22]:

$$\forall x \in \mathcal{X} : \quad (F * G)[x] = \sum_{c \in \mathcal{C}} \sum_{y \in \mathcal{Y}} F_c[x - y] \, G_c[y] \tag{2.27}$$

Groups are then the parameter which allows us to precisely characterise this set of input channels $\mathcal{C}$. If the number of input- and output channels is a multiple of a certain number N_{groups}, then we can split both input and output into N_{groups} equally sized groups. Here, N_{groups} is the parameter named 'groups' in deep learning frameworks. A kernel in the i'th group of the output then reads from all the input channels in the i'th group of the input. For example, suppose we have an image with 6 input channels ($C_i = 6$): $N_{groups} = 1$ would result in every kernel reading from every input channel (a standard convolution), while $N_{groups} = C_i = 6$ would make kernels read from only one channel (like a pooling). Values of N_{groups} where $1 < N_{groups} < C_i$ interpolate between these extremes, where every kernel reads a fixed subset of the input channels. It should be noted that while the number of output channels C_o (equal to the number of kernels) is constrained to be a multiple of N_{groups}, the number of kernels per group in the output S_o can be freely chosen and does not need to equal the number of channels per input group S_i: see Fig. 2.4 for examples.

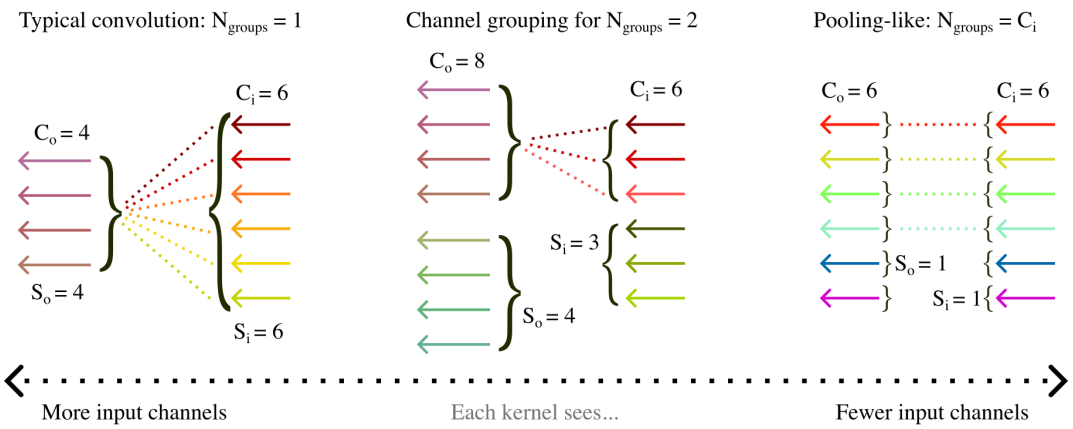

Figure 2.4: Illustration of channels in the input and output, and the effects of the group parameter on which input channels are used by the kernels.

2.6 Non-morphological semifields

In previous sections, we discussed two semifields related to mathematical morphology, namely T_+ and T_-, and showed their correspondence with the pooling layers within a CNN. Using these results, we could see that a max-pooling was equivalent to a semifield convolution (in T_+, with a flat kernel). However, the standard convolutional layers within a CNN can also be interpreted as a semifield convolution in the linear semifield $L = (\mathbb{R}, +, \times)$: filling in the equation for $\circledast_L$ (Eq. 2.14), we would again obtain a standard convolution $*$.

This raises the question of whether there are alternative semifields that act similarly to the linear semifield, such that they could be used in place of the standard convolutional layer in a CNN. In an investigation of some semifields, [14] identified sets of relevant semifields that were isomorphic (equivalent under a bijective mapping) to the linear semifield L. These can be written as:

$L_{\mu+}$ The positive log semifields: $(\mathbb{R} \cup \{-\infty\}, \oplus_\mu, +)$ for all $\mu > 0$ where $a \oplus_\mu b = \frac{1}{\mu} \ln(e^{\mu a} + e^{\mu b})$, $\mathbb{0} = -\infty$ and $\mathbb{1} = 0$ (using $e^{-\infty} = 0$)

$L_{\mu-}$ The negative log semifields: $(\mathbb{R} \cup \{+\infty\}, \oplus_\mu, +)$ for all $\mu < 0$ where $a \oplus_\mu b = \frac{1}{\mu} \ln(e^{\mu a} + e^{\mu b})$, $\mathbb{0} = +\infty$ and $\mathbb{1} = 0$

R_p The root semifields: $(\mathbb{R}_+, \oplus_p, \times)$ for all $p \neq 0$ where $a \oplus_p b = \sqrt[p]{a^p + b^p}$, $\mathbb{0} = 0$ and $\mathbb{1} = 1$

In [14] it was also noted that the positive log semifield $L_{\mu+}$ becomes equivalent to T_+ when μ approaches $+\infty$, while $L_{\mu-}$ becomes equivalent to T_- as μ approaches $-\infty$. Additionally, one can see that the root semifields R_p also become equivalent to T_+ when p approaches $+\infty$, as $\oplus_p$ then becomes the infinity-norm (which is equivalent to the maximum function). Informally, one can also see that as p approaches $-\infty$, the semifield addition $\oplus_p$ approximates the minimum function, making $R_{-\infty}$ equivalent to T_-.

These additional equivalences with T_+ suggest the possibility of using log or root semifields with high values for μ or p as another alternative for traditional max-pooling layers in CNNs. It should be noted, however, that neither log nor root semifields seem to be part of an adjoint, meaning the resulting pooling would likely not be easily modelled using principles of mathematical morphology, and many of the theoretical guarantees would be lost.

Chapter 3

Method

With a greater understanding of what semifields are, how convolutions work and some examples of relevant semifield convolutions, we can now examine how to apply semifield convolutions within the context of CNNs.

First, we will go through some implementation notes on semifield convolutions, and describe how to use quadratic forms to parameterise convolutional kernels. Afterwards, we will describe the experimental setup used to examine which kinds of quadratic forms work best in the context of CNNs, and which parameters of a semifield convolution have a positive impact on the performance of a CNN on various image classification tasks.

3.1 Semifield convolutions in a CNN TODO

I used a library that I wrote, please see other report for more details.
Important note: reduction over channels with $+$ or with $\oplus$

3.2 Quadratic forms for convolutional kernels TODO

We may have reason to believe that kernels should be smooth. A quadratic form is a simple yet useful smooth function to parameterise the kernel.

[10]

3.3 Learning positive definite matrices

While there are many methods for parameterising a 2×2 positive definite matrix, since the calculation of the distance as used in the quadratic distance function is equivalent to the Mahanalobis distance, we may wish to view the matrix as a covariance matrix $\Sigma \in \mathbb{R}^{2\times 2}$. Since Σ must be symmetric, we know by the spectral theorem that Σ is diagonalisable [24]:

$$\begin{aligned} &\text{For some orthogonal matrix } Q \in \mathbb{R}^{2\times 2}, \text{ and} \\ &\text{for some diagonal matrix } D \in \mathbb{R}^{2\times 2}, \\ &\Sigma = QDQ^T && (3.1) \\ &= Q \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} Q^T && (3.2) \end{aligned}$$

Here, Q (as an orthogonal matrix) can either be a rotation or a reflection. However, since Q occurs twice, its determinant cancels, and fixing Q to be a rotation does not reduce expressivity (see Appendix 6.1). As such, we can use:

$$\Sigma = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \tag{3.3}$$

This parameterisation can be efficiently inversed for the quadratic form, as

$$\begin{aligned} \Sigma^{-1} &= (QDQ^T)^{-1} && (3.4) \\ &= (Q^T)^{-1} D^{-1} Q^{-1} && (3.5) \\ &= Q \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix} Q^T && (3.6) \end{aligned}$$

In order for Σ to be positive-definite, σ_1^2 and σ_2^2 are required to be strictly positive, while there are no constraints on ϕ. As such, for any $\boldsymbol{\theta} \in \mathbb{R}^3$:

$$\text{Let } \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \log|\sigma_1| \\ \log|\sigma_2| \\ \phi \end{bmatrix}, \text{ then a valid } \Sigma^{-1} \text{ would be} \tag{3.7}$$

$$\Sigma^{-1} = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 \\ \sin\theta_3 & \cos\theta_3 \end{bmatrix} \begin{bmatrix} e^{-2\theta_1} & 0 \\ 0 & e^{-2\theta_2} \end{bmatrix} \begin{bmatrix} \cos\theta_3 & \sin\theta_3 \\ -\sin\theta_3 & \cos\theta_3 \end{bmatrix} \tag{3.8}$$

Since we placed no assumptions on $\boldsymbol{\theta}$, it is safe for a black-box or gradient-based optimiser to adjust in any direction, as the resulting Σ will always be a positive definite matrix.

This parameterisation has the advantage of being easily interpretable: e^{θ_1} and e^{θ_2} are the standard deviations of a hypothetical normal distribution in the first and second principal axis of the quadratic, while θ_3 is the counter-clockwise angle the first principal axis forms with the x-axis. An alternative parameterisation, based on the Pearson correlation coefficient, can be found in Appendix 6.3.

3.4 Experimental setup TODO

Hardware, software, hyperparameter tuning

3.5 Image classification datasets TODO

Which ones did I use?

Chapter 4

Experiments TODO

I only really want to write the accompanying text when I have a clear idea of what the experiments are going to be: for now, I'm just putting some graphs in from the test experiments I conducted.

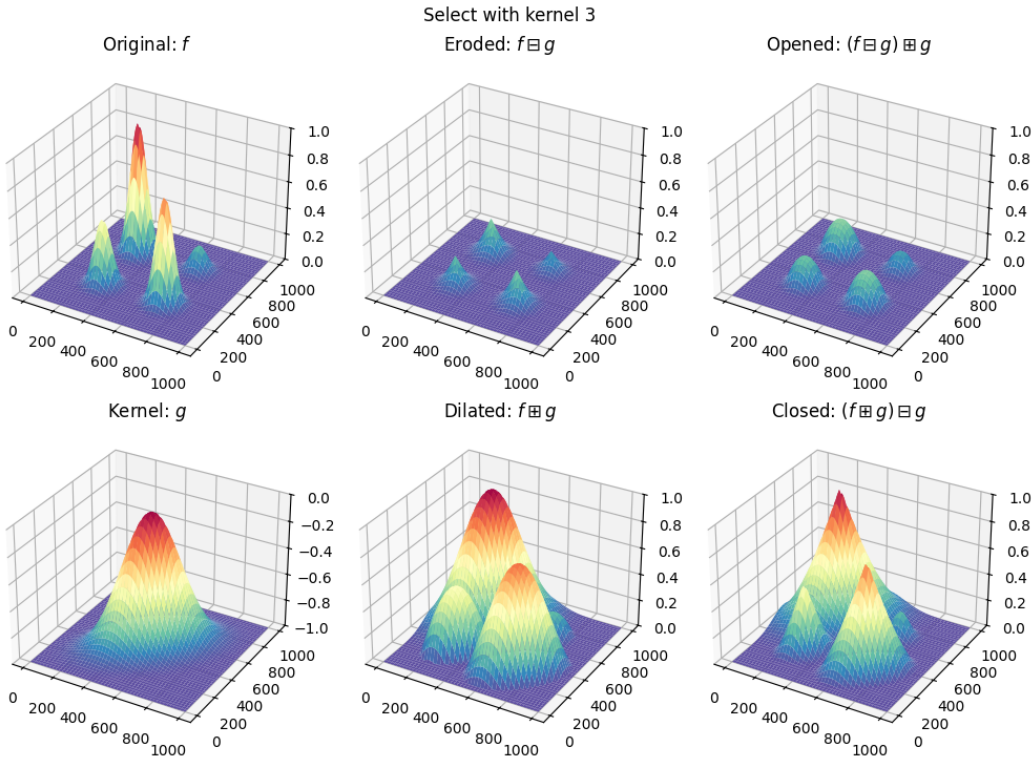

Figure 4.1: Illustration of the effects of dilation, erosion and their combination in two dimensions, with height representing the function value.

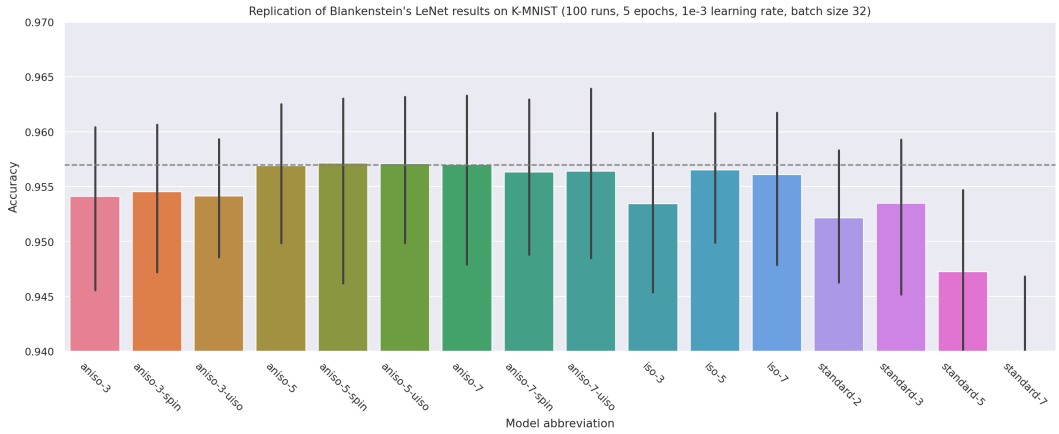

Figure 4.2: Replication of Blankenstein's LeNet results [12]

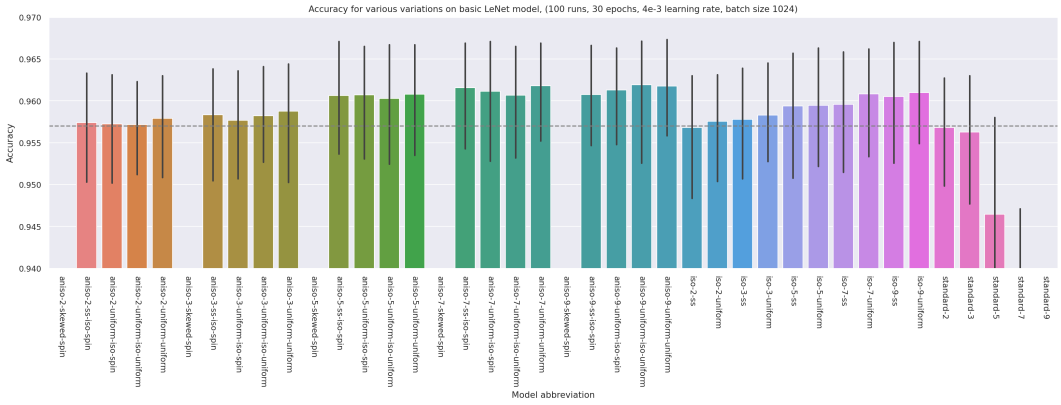

Figure 4.3: Extension of LeNet results using more (100) epochs, with the aniso results other than the base aniso-3/5/7 being obtained with higher LR

Chapter 5

Conclusion TODO

.....

Bibliography

[1] A. Esteva, K. Chou, S. Yeung, N. Naik, A. Madani, A. Mottaghi, Y. Liu, E. Topol, J. Dean, and R. Socher, "Deep learning-enabled medical computer vision," *NPJ digital medicine*, vol. 4, no. 1, p. 5, 2021.

[2] S. Jain, N. Pise, *et al.*, "Computer aided melanoma skin cancer detection using image processing," *Procedia Computer Science*, vol. 48, pp. 735–740, 2015.

[3] S. Wan and S. Goudos, "Faster r-cnn for multi-class fruit detection using a robotic vision system," *Computer Networks*, vol. 168, p. 107036, 2020.

[4] A. Sivaranjani, S. Senthilrani, B. Ashok Kumar, and A. Senthil Murugan, "An overview of various computer vision-based grading system for various agricultural products," *The Journal of Horticultural Science and Biotechnology*, vol. 97, no. 2, pp. 137–159, 2022.

[5] L. Xie, T. Ahmad, L. Jin, Y. Liu, and S. Zhang, "A new cnn-based method for multi-directional car license plate detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 507–517, 2018.

[6] Y. Le Cun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard, "Handwritten digit recognition: Applications of neural net chips and automatic learning," in *Neurocomputing: Algorithms, Architectures and Applications*, pp. 303–318, Springer, 1990.

[7] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *ArXiv e-prints*, 11 2015.

[8] H. J. Heijmans and J. Serra, *Morphological image operators*, vol. 38. Philadelphia, Society for Industrial and Applied Mathematics., 1996.

[9] P. Maragos, "Tropical geometry, mathematical morphology and weighted lattices," in *Mathematical Morphology and Its Applications to Signal and Image Processing* (B. Burgeth, A. Kleefeld, B. Naegel, N. Passat, and B. Perret, eds.), (Cham), pp. 3–15, Springer International Publishing, 2019.

[10] v. d. R. Boomgaard, "Numerical solution schemes for continuous-scale morphology," in *Scale-Space*, 1999.

[11] R. Groenendijk, L. Dorst, and T. Gevers, "Morphpool: efficient non-linear pooling & unpooling in cnns," *arXiv preprint arXiv:2211.14037*, 2022.

[12] T. Blankenstein, "Investigating the parabolic dilation as the max pooling operation in deep learning," 2022.

[13] K. Veldhorst, "Local operators in semifields: Parabolic pooling revisited," 2024.

[14] G. Bellaard, S. Sakata, B. M. N. Smets, and R. Duits, "Pde-cnns: Axiomatic derivations and applications," 2024.

[15] J.-M. Geusebroek, A. W. M. Smeulders, and J. van de Weijer, "Fast anisotropic gauss filtering," *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 12 8, pp. 938–43, 2002.

[16] P. Mantini and S. K. Shah, "Cqnn: Convolutional quadratic neural networks," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 9819–9826, 2021.

[17] Y. Jiang *et al.*, "Nonlinear cnn: improving cnns with quadratic convolutions," *Neural Computing and Applications*, vol. 32, pp. 8507 – 8516, 2019.

[18] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[19] J. A. Beachy, *Abstract Algebra (Third Edition)*. 2006.

[20] A. Paszke, M. J. Johnson, R. Frostig, and D. Maclaurin, "Parallelism-preserving automatic differentiation for second-order array languages," in *Proceedings of the 9th ACM SIGPLAN International Workshop on Functional High-Performance and Numerical Computing*, pp. 13–23, 2021.

[21] R. Gonzalez and R. Woods, *Digital Image Processing Global Edition*. Pearson Deutschland, 2017.

[22] "Conv2d — PyTorch 2.7 documentation."

[23] "XLA convolution operation semantics."

[24] D. Poole, "Linear algebra: A modern introduction," 2015.

Chapter 6

Appendix

6.1 Redundancy of mirroring in QDQ^T

Suppose we had some symmetric $\Sigma \in \mathbb{R}^{2\times 2}$; we could then use orthogonal diagonalisation to write $\Sigma = QDQ^T$ for some orthogonal Q and diagonal D.

In 3.3, the claim was made that requiring Q to be a rotation (and not a reflection) did not decrease the expressivity of the representation, i.e. all symmetric positive definite Σ are representable as RDR^T with R being a rotation. To show this, we can suppose some reflection $Q \in \mathbb{R}^{2\times 2}$, and see that Q can be written as a rotation with angle ϕ (R_ϕ) of a reflection in the x-axis [24]:

$$Q = \begin{bmatrix} \cos\phi & \sin\phi \\ \sin\phi & -\cos\phi \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = R_\phi \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{6.1}$$

Then, we can write out the orthogonal diagonalisation using 6.1:

$$\begin{aligned} \Sigma &= QDQ^T && (6.2)\\ &= \left(R_\phi \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\right) D \left(R_\phi \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\right)^T && (6.3)\\ &= R_\phi \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} R_\phi^T && (6.4)\\ &= R_\phi \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} R_\phi^T && (6.5)\\ &= R_\phi D R_\phi^T && (6.6) \end{aligned}$$

□

6.2 Clipping kernels with Lipschitz conditions TODO

Todo: proof, and better name for this section

6.3 Alternative parameterisation for $\Sigma \in \mathbb{R}^{2\times 2}$

A different way of parameterising a 2×2 covariance matrix would be to use the Pearson correlation coefficient ρ instead of the angle ϕ. We can then keep the covariance matrix in its Cholesky decomposed form, using a lower triangular L such that $\Sigma = LL^T$. Then, for any $\boldsymbol{\theta} \in \mathbb{R}^3$, we can find the corresponding L:

$$\text{Let } \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \log|\sigma_1| \\ \log|\sigma_2| \\ \tan\rho \end{bmatrix}, \text{ then a valid } \Sigma \text{ would be} \tag{6.7}$$

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho \\ \sigma_1\sigma_2\rho & \sigma_2^2 \end{bmatrix} = LL^T = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} \\ 0 & l_{22} \end{bmatrix} \tag{6.8}$$

$$= \begin{bmatrix} l_{11}^2 & l_{11}l_{21} \\ l_{11}l_{21} & l_{21}^2 + l_{22}^2 \end{bmatrix} \tag{6.9}$$

As such, we know that:

$$l_{11} = \sqrt{\sigma_1^2} = \sigma_1 = \exp(\theta_1) \tag{6.10}$$

$$l_{21} = \frac{\sigma_1\sigma_2\rho}{\sigma_1} = \sigma_2\rho = \exp(\theta_2)\tanh(\theta_3) \tag{6.11}$$

$$l_{22} = \sqrt{\sigma_2^2 - \sigma_2\rho} = \sqrt{\exp(2\theta_2) - \exp(\theta_2)\tanh(\theta_3)} \tag{6.12}$$

which is then a valid parameterisation for the Cholesky decomposed form for a 2×2 positive definite matrix[1]. If we keep the covariance matrix in this triangular form, we can see that the quadratic form can be calculated in an efficient manner:

(based on the PyTorch code for the multivariate normal PDF)

$$\mathbf{x}^T\Sigma^{-1}\mathbf{x} = \mathbf{x}^T(LL^T)^{-1}\mathbf{x} \tag{6.13}$$

$$= \mathbf{x}^T(L^T)^{-1}L^{-1}\mathbf{x} \tag{6.14}$$

$$= \mathbf{x}^T(L^{-1})^TL^{-1}\mathbf{x} \tag{6.15}$$

$$= ((L^{-1})\mathbf{x})^T(L^{-1}\mathbf{x}) \tag{6.16}$$

$$= (L^{-1}\mathbf{x}) \cdot (L^{-1}\mathbf{x}) \tag{6.17}$$

$$\text{Suppose } \mathbf{b} = L^{-1}\mathbf{x}, \text{ then}$$

$$L\mathbf{b} = \mathbf{x}, \text{ so}$$

$$\mathbf{b} = \textsc{solve-triangular}(L, \mathbf{x}) \tag{6.18}$$

$$\mathbf{x}^T\Sigma^{-1}\mathbf{x} = \mathbf{b} \cdot \mathbf{b} \tag{6.19}$$

where SOLVE-TRIANGULAR performs efficient backsubstitution to avoid computing the inverse. This method is significantly (>5x) faster on the CPU it was tested on while still showing modest performance improvements on the GPU it was tested on ($\sim 10\%$). However, interpretation of the Pearson correlation coefficient may be more challenging compared to interpreting the angular offset of the first primary axis, and the calculation of the quadratic forms is a negligible part of the model runtime on the GPU, so it was chosen to instead parameterise Σ with the angle ϕ.

[1] To extend this parameterisation for higher dimensions, see the Cholesky-Banachiewicz algorithm for the Cholesky decomposition