MorphPool: Efficient Non-linear Pooling & Unpooling in CNNs

Rick Groenendijk
r.w.groenendijk@uva.nl
Leo Dorst
l.dorst@uva.nl
Theo Gevers
th.gevers@uva.nl

Computer Vision Group
University of Amsterdam
Amsterdam, the Netherlands

Abstract

Pooling is essentially an operation from the field of Mathematical Morphology, with max pooling as a limited special case. The more general setting of MorphPooling greatly extends the tool set for building neural networks. In addition to pooling operations, encoder-decoder networks used for pixel-level predictions also require *unpooling*. It is common to combine unpooling with convolution or deconvolution for up-sampling. However, using its morphological properties, unpooling can be generalised and improved. Extensive experimentation on two tasks and three large-scale datasets shows that morphological pooling and unpooling lead to improved predictive performance at much reduced parameter counts.

1 Introduction

Contemporary deep learning architectures exploit pooling operations for two reasons: to filter impactful activation values from feature maps, and to reduce spatial feature size [28]. The most used pooling operation is the **max pool**, which is used in nearly all common network architectures such as ResNet [14], VGGNet [32], and DenseNet [16]. These network architectures can be applied to pixel-level prediction tasks, such as semantic segmentation. To do so, inputs are down-sampled to a set of latent features of small spatial size, after which they are up-sampled to full resolution again. Up-sampling from pooled feature sets most often happens with a combination of unpooling and deconvolution [40, 41] and is used in seminal works such as [3, 22, 26].

As will be shown in this paper, down-sampling using max pooling can be formalised and improved using *mathematical morphology*, the mathematics of contact. Ever since the works of Serra [29], the underlying algebraic structure of data that is acquired using probing contact (*e.g.* LiDAR and radar) has been known to the computer vision community [5, 11, 25, 33]. It is different from the algebra of linear diffusion that is used to build convolutional neural networks (CNNs). Rather than just considering pooling, it would sensible to formalise pooling *and* unpooling in CNNs as fully morphological operations: in any encoding layer, pooling is a form of down-sampling and a decoder layer will ultimately need to up-sample. The task of down-sampling is one of morphological sampling and the task of up-sampling is in fact one of morphological interpolation.

The connection between morphology and pooling has been noted in the literature: concurrently with this paper, [1] remarks that max pooling and morphological operations indeed share commonalities, although they do not further explore the idea. [10] proposes a morphological *alternative* to pooling as a weighted combination of morphological dilation and erosion, and calls it morphological pooling. Both these papers do not formalise the pooling as morphological, nor do they treat up-sampling as the morphological semi-inverse of pooling. In this article, it is shown how pooling is of an essentially morphological nature (dilation or erosion), and that there is then a naturally associated form of up-sampling (unpooling), dictated by the tropical algebra of morphology.

The contributions of this paper are:

- A formalisation of max pooling showing that it is an unparameterised non-overlapping special case of the morphological dilation.
- A fully morphological up-sampling procedure that can replace the semi-morphological unpooling-deconvolution scheme that is commonly used in neural networks.
- Extensive experimentation to show morphological pooling and unpooling consistently outperform other down and up-sampling schemes at highly reduced parameter counts, including efficient CUDA implementations. This effect is most pronounced on data that is of an inherently geometric nature, such as depth images.

2 Background

Pooling Convolutional networks employ sequential convolution layers to obtain an expressive set of features for regression or classification. Pooling layers are used to force the network to regress high-quality features, and there exist many pooling schemes (for a review, refer to [12]). The most common pooling scheme is the max pool. Max pooling is used to obtain maximum-amplitude coefficients at intervals while reducing spatial feature sizes [6, 37], and to achieve invariance to small distortions from noise [17]. It is shown to be a very efficient procedure to encode meaningful features, often outperforming mean pooling (*e.g.* [17, 39]). In [4], it is argued that max pooling is more robust to high clutter noise in class separation than mean pooling.

Unpooling While pooling is used to down-sample and refine features, up-sampling may be necessary to obtain predictions at original resolution. Up-sampling can naturally happen in a variety of ways, but of interest is the inverse of pooling. [40, 41] are the first to define an inverted pooling operation using a combination of deconvolution and 3D reverse max pooling. Here a set of switches records the location of max values in the pooling step, that are then used to place back latent features at higher resolution; remaining elements are set to zero. The unpooling operation is interleaved with deconvolution, or rather transposed convolution, to obtain dense feature maps (*i.e.* infilling). The main advantage of deconvolution is that it does not employ any predefined interpolation scheme, but can learn the interpolation parameters [1].

[1] It is however prone to forming checkerboard artefacts [23].

There exist many examples in literature of this unpooling-infilling scheme for up-sampling, most notably [3, 22]. First, [22] introduce Deconvolutional Neural Networks. Their network interleaves 2D unpooling layers with deconvolutions, since it is argued that low-level visual features capture shape details and max pooling has been reported to recover shape well [19]. Second, [3] introduce SegNet, which interleaves 2D unpooling and convolution. The main motivation is that unpooling results in more sharply delineated semantic boundary predictions. This strategy is also employed in [20], who solve a semantic segmentation task as well.

While [3, 22] employ a deconvolution and a convolution for infilling respectively, it is worth noting that the term deconvolution is misleading: it is not the deconvolution in the mathematical sense, but rather the convolution with flipped kernel (correlation). Since parameters are freely learned in networks (and setting the stride similarly for either operation) they end up being equivalent.

Variations [9] and improvements [18, 38] of the unpooling-infilling scheme exist, but the (de)convolution operation remains necessary to deal with sparsity. Whereas pooling is parameterless, the infilling stage is heavily parameterised by (de)convolution to overcome this issue. It will now be shown that this is not necessary.

3 Method

In this section, max pooling will be shown to be a special case of the morphological dilation. Note that the same could be done for min pooling through erosion, by morphological duality [30]. First, a common notation is introduced for neural networks and individual (linear or morphological) layers within the network. A neural network f is a composite function $f(\mathbf{x}) = f_L \circ \cdots \circ f_l \circ \cdots \circ f_1(\mathbf{x})$ of L layers. Let $\mathbf{y}_n = f(\mathbf{x}_n)$ be the output of the network function. The network is tasked with fitting a (sample, target) dataset $\mathbf{D} = \{(\mathbf{x}_n, \mathbf{t}_n)\}_{n=0}^{N}$. In this case, training the network is the minimisation of the energy function $E(\mathbf{t}_n, \mathbf{y}_n)\,\forall n$, where $\mathbf{t}_n$ is the target output. The input to a specific layer is denoted f_-, where the subscript l is dropped for readability. The output to that layer is denoted f_+, and the parameters of the layer are h.

3.1 Morphological Pooling

Max pooling is a discrete operator that takes a non-overlapping patch-based maximum for any given discrete input map $f(\mathbf{x})$. It is shown below that max pooling is a non-overlapping, non-parameterised special case of the morphological dilation. The morphological dilation is defined on the semi-ring $\{\mathbb{R}_{-\infty}, \bigvee, +\}$ where $\bigvee$ denotes the supremum operation and $+$ is addition. This algebraic system extends the set of real numbers $\mathbb{R}$ with minus infinity: $\mathbb{R}_{-\infty} \equiv \mathbb{R} \cup -\infty$ [25].

A layer input signal $f_- : \mathbb{R}^D \to \mathbb{R}_{-\infty}$ indexed by indicator variable $\mathbf{x}$, and a structuring element $h \colon \mathbb{R}^D \to \mathbb{R}_{-\infty}$ indexed by indicator variable $\mathbf{z}$ are combined to produce the morphological dilation as the layer output signal f_+:

$$f_+(\mathbf{x}) = \bigvee_{\mathbf{z}} f_-(\mathbf{x} - \mathbf{z}) + h(\mathbf{z})\,. \tag{1}$$

On the other hand, max pooling can be written as

$$f_{\vee}(\mathbf{x}) = \max_{\mathbf{z} \in \mathbf{Z}} \left(f_{\wedge}(s\mathbf{x} + \mathbf{z}) \right), \tag{2}$$

where $f_{\vee}$ is the down-sampled layer output (similar to f_{+} in Equation 1), $f_{\wedge}$ is the layer input at high resolution, s is a predefined stride (usually 2), and $\mathbf{z}$ is an indicator set to capture the patch over which the maximum is taken (also usually of size 2). The spatial dimensions are thus reduced by stride s.

Let $\mathbf{x}_{\vee} \in \mathbf{x}$ be the output locations on the down-sampled output signal $f_{\vee}$. In that case, $\mathbf{x}_{\wedge}$ is the set of input locations scaled by the stride s, that is $\{\mathbf{x}_{\wedge} | (\exists \mathbf{x}_{\vee} \in \mathbf{x}) [\mathbf{x}_{\wedge} = s\mathbf{x}_{\vee}]\}$. Also, drop the assumption of discreteness to adopt the continuous supremum operator $\bigvee$. Then, we can rewrite Equation 2 to

$$f_{\vee}(\mathbf{x}_{\vee}) = \bigvee_{\mathbf{z}} f_{\wedge}(\mathbf{x}_{\wedge} + \mathbf{z}). \tag{3}$$

In mathematical morphology, the structuring element h is regarded as a shape that dilates or erodes a signal. The simplest structuring element is the *flat* structuring element that can be thought of as a flat disc $\mathbf{Z}$ centred around the origin:

$$h_{\text{flat}}[\mathbf{z}] = \begin{cases} 0 & \text{if} \quad \mathbf{z} \subseteq \mathbf{Z} \\ -\infty & \text{otherwise.} \end{cases} \tag{4}$$

Since h_{flat} is symmetric, the $\mathbf{z}$ indices could be substituted by $-\mathbf{z}$. Thus, Equation 3 becomes

$$f_{\vee}(\mathbf{x}_{\vee}) = \bigvee_{\mathbf{z}} f_{\wedge}(\mathbf{x}_{\wedge} - \mathbf{z}) + h_{\text{flat}}[\mathbf{z}]. \tag{5}$$

From Equation 5, it is clear that the max pool is in fact a dilation with a flat structuring element, and the $\vee, \wedge$-notation for the signals $f_{\vee}, f_{\wedge}$ and indicator sets $\mathbf{x}_{\vee}, \mathbf{x}_{\wedge}$ is to account for striding the input by stride s. A schematic overview of pooling by dilation with flat structuring element is given in Figure 1.

While flat structuring elements are often used in grey-value morphology, there are more interesting structuring elements. One example is the parabolic structuring element [35]:

$$h_{\text{parabolic}}[\mathbf{z}, \sigma] = \begin{cases} -\frac{||\mathbf{z}||^2}{2\sigma^2} & \text{if} \quad \mathbf{z} \subseteq \mathbf{Z} \\ -\infty & \text{otherwise.} \end{cases} \tag{6}$$

Using such a structuring element is a morphological weighting, where pixels closer to the centre affect the maximum more depending on the scale σ (and algebraically it is the natural counterpart of Gaussian weighting in convolution [35]). Using parabolic h, a morphological scale space can be constructed [15, 34]. Scale space is the natural way of dissecting object structure at different scales, and it provides clear intuition for why re-sampling signals can be done by dilation. These parabolic structuring elements can for example also be used to build scale equivariant networks [27, 36]. The flat and parabolic structuring elements are members of a larger set of parameterised morphological operations; the set of parameterised operations generalise max pooling.

Lastly, note that convolution and dilation are logarithmically related in the sense that convolution uses a multiplication-addition scheme where dilation uses an addition-supremum

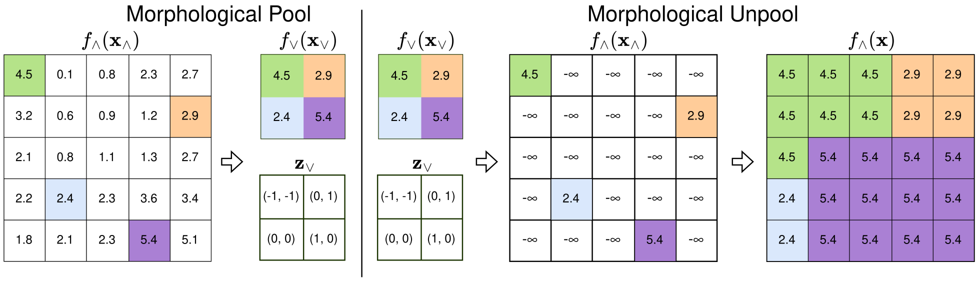

Figure 1: Unparameterised (*i.e.* flat structuring element) morphological pooling & unpooling. **(left)** denotes pooling, which is a dilation with a flat structuring element of size 3x3 combined with sub-sampling resulting in $f_\vee(\mathbf{x}_\vee)$ and provenance $\mathbf{z}_\vee$. The image is divided in four 3x3 patches; **(right)** shows the two-step process of unpooling, which uses the provenance to place the maximum elements at the correct locations resulting in $f_\wedge(\mathbf{x}_\wedge)$; then a 5x5 flat structuring element should be used to morphologically interpolate the maxima yielding the full resolution map $f_\wedge(\mathbf{x})$.

scheme. However, convolution on images acts on spatial as well as channel dimensions; grey-value dilation could as well, but pooling is a purely spatial operation. Therefore, the discussion is limited to purely spatial h, omitting channels dimensions from the dilation operation.

3.2 Morphological Unpooling

Morphological unpooling should invert, as much as possible, the pooling operation and up-sample back to original scale. Full inversion is not possible, because of non-maximum suppression of the dilation operation. Morphological unpooling is a two-stage process. The first stage is morphological *provenance mapping* [34] (syntactically similar to the morphological derivative introduced in [13]). For purposes of unpooling, $\mathbf{z}_\vee$ from the pooling stage can be thought of as provenance: recorded locations at which to place back values during up-sampling. That is:

$$f_\wedge(\mathbf{x}_\wedge - \mathbf{z}_\vee) = \begin{cases} f_\vee(\mathbf{x}_\vee) & \text{for all} \quad \mathbf{x}_\vee \,|\, (\exists \mathbf{x}_\wedge \in \mathbf{x})\,[\mathbf{x}_\wedge = s\mathbf{x}_\vee] \\ -\infty & \text{otherwise.} \end{cases} \tag{7}$$

Or, the locations $\mathbf{x}_\vee$ on the down-sampled signal $f_\vee$ are transferred to the regularly sampled locations $\mathbf{x}_\wedge$ compensated by the provenance of the maxima that were stored in $\mathbf{z}_\vee$ – see Figure 1. The values $-\infty$ denote values that are unknown. This is logarithmically analogous to setting these values to 0 when supplementing this operation with a (de)convolution, because that element would not contribute to the outcome of addition-multiplication. On the semi-ring $\{\mathbb{R}_{-\infty}, \bigvee, +\}$, the value $-\infty$ is the 0-element equivalent.

The values set to $-\infty$ at the other locations $\mathbf{x}$, however, cannot be used in training a neural network. Since the values $f_\vee(\mathbf{x}_\vee)$ at locations $\mathbf{x}_\vee$ were maxima in the pooling step, the undefined values at all $\mathbf{x}$ must be upper bounded by $f_\vee(\mathbf{x}_\vee)$ and h when unpooling. Consequently, the second stage is the application of a morphological dilation that retains the same spatial dimensions to morphologically interpolate between the relocated maxima at $\mathbf{x}_\wedge - \mathbf{z}_\vee$:

$$f_\wedge(\mathbf{x}) = \bigvee_{\mathbf{w}} f_\vee(\mathbf{x} - \mathbf{z}_\vee - \mathbf{w}) + h(\mathbf{w}), \tag{8}$$

where $\mathbf{w}$ is a scaled indicator set of s times the size of $\mathbf{z}$ to perform gap filling in $f_\wedge(\mathbf{x}_\wedge - \mathbf{z}_\vee)$. As a result, the output $f_\wedge(\mathbf{x})$ is guaranteed not to contain values at $-\infty$; in stead the values have been upper bounded by the maximum $\bigvee_{\mathbf{w}} f_\vee(\mathbf{x} - \mathbf{z}_\vee - \mathbf{w}) + h(\mathbf{w})$. See Figure 1 for an example with a flat structuring element h.

In summary, morphological unpooling is a combination of **(1)** a strided morphological derivative to up-sample the spatial features and **(2)** a dilation with increased kernel size to suppress unknown values. Again, it is possible to parameterise h of the dilation in the second stage (Equation 8) to make full use of concepts from morphological scale space and morphological interpolation. The full procedure of generalised morphological pooling & unpooling is called **MorphPool**.

4 Experiments

MorphPool is evaluated on semantic segmentation and depth auto-encoding on NYUv2 [31], SUN-RGBD [42], and Stanford 2D-3D-Semantics Dataset (2D-3D-S) [2]. Morphological unpooling is compared to the standard unpooling-infilling scheme, and to linear sampling and interpolation such as combinations of (de)convolutions. To this end, DownUpNet is introduced: the simplest encoder-decoder architecture that has variable down and up-sampling blocks. The aim of the paper is a proof-of-principle, for which a simple architecture like DownUpNet is sufficient; its goal is not to compete with specialised segmentation networks to achieve state-of-the-art results. It is hypothesised that morphological operations perform better at depth data, since depth data is inherently non-linear at geometric edges. In contrast, (de)convolution does not allow for processing 3D spatial information between pixel neighbours, and is sensitive to occlusion. Implementation details are given below, but more details are found in Supplementary Material **A**. Additionally, results in this paper are further supported by Supplementary Material **B**.

Down & Up-sampling MorphPool is compared to a regular pooling & unpooling baseline like *e.g.* in [3, 22]. In those networks, unpooling is followed by (de)convolution to deal with the sparsity that unpooling introduces. Another baseline is a linear sampling method (common in ResNet [14] architectures), where down-sampling is a strided convolution and up-sampling is bilinear interpolation followed by a convolution layer. Both methods introduce high numbers of parameters due to the (de)convolutions. More specifically, they introduce $C^2 \times K^2$ per layer, where C are the channels and K is the kernel size. This makes the sampling procedures potentially expressive at the cost of memory. The morphological pooling & unpooling with **flat** structuring elements however, introduce *no* additional parameters. When the morphological operations are parameterised **parabolically**, C parameters per layer are introduced. Finally, for structuring elements that are parameterised at each spatial location –the **general** setting– $C \times K^2$ parameters per layer are introduced. It will be shown that additional parameters of (de)convolution are redundant.

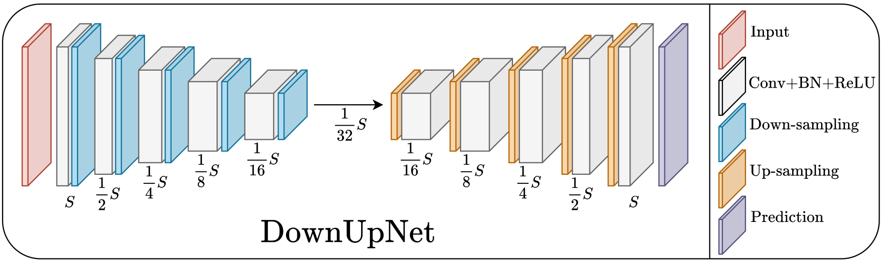

Figure 2: The DownUpNet architecture, which is a simple encoder-decoder. Each down-sampling operation reduces spatial dimensions S by 2, for a total of 2^5. The down-sampling operations are preceded by convolution including batch normalisation and ReLU non-linearity. The up-sampling operations are followed by the same type of block.

DownUpNet To fairly compare between down and up-sampling schemes, DownUpNet is introduced. DownUpNet down-samples 5 times, for a total reduction of spatial resolution of 2^5. It then up-samples back to the original spatial resolution. Each down-sampling operation is preceded by a convolution, with batch normalisation and ReLU non-linearity. Each up-sampling operation is followed by a convolution, again with normalisation and non-linearity. For an overview, see Figure 2. Specifically for the larger 2D-3D-S dataset, two convolutions, normalisation, and non-linearities are coupled with sampling operations.

Implementation Generalised morphological operations are not implemented by deep learning frameworks such as PyTorch [24]. Implementing the operations in PyTorch as a composed function of additions and maxima is not tractable: unlike convolution, in which multiplication-addition of the patch and the kernel happens jointly, addition-maximum is done in sequence. The addition of the kernel to each patch is either memory intensive or time intensive, depending on implementation. Instead, all morphological operations (*i.e.* forward operations and backward derivatives) in this article are implemented directly in C++/CUDA [21]. More high level functions such as losses and networks are implemented using PyTorch. All code, including the C++/CUDA code, is made available at `https://github.com/rickgroen/morphpool`.

Dataset & Training All experiments are performed on NYUv2 ($N_{\text{train}} = 795, N_{\text{test}} = 654$), SUN-RGBD ($N_{\text{train}} = 5285, N_{\text{test}} = 5050$), and 2D-3D-S ($N_{\text{train}} = 52903, N_{\text{test}} = 17593$). Experiments are run for both RGB input and for depth input. RGB images are normalised with training set statistics to follow a zero-mean Gaussian; depth images are not normalised. Because of this (and because depth data has more sensor noise and infilling artefacts) networks trained with depth input start with a learning rate $\lambda = 5\text{e}{-4}$ whereas RGB networks start with $\lambda = 5\text{e}{-3}$. Both learning rates exponentially decay to end at 2% of the initial λ. For training, random crops of size 384×384 are used for NYUv2 and SUN-RGBD, crops of 512×512 are used for 2D-3D-S. During testing, centre crops are used as close to full resolution as possible, while retaining a resolution that is divisible by 2^5.

Table 1: **2D-3D-S Segmentation on Depth.** The first column denotes the sampling method, the second and third column denote the kernel size for Pooling and Unpooling. To deal with issues of sparsity, (de)convolution post-processing of features is necessary; this is denoted at the top as None, Conv and Deconv. For each method, the number of *additional* parameters, mean Intersection over Union, pixel accuracy and Boundary F1-score is given. The bold-faced results indicate best performance per column. Clearly, MorphPool outperforms the alternatives at reduced parameter count.

Sampling			None				Conv				Deconv			
	P	U	#params	mIoU	acc	bf	#params	mIoU	acc	bf	#params	mIoU	acc	bf
Linear	3	-	12.6M	0.403	0.693	0.426	25.1M	0.413	0.699	0.434	25.1M	0.410	0.692	0.430
Standard Pool	2	3	0	0.352	0.657	0.375	12.6M	0.377	0.675	0.393	12.6M	0.397	0.693	0.407
MorphPool	2	3	0	0.385	0.682	0.441	12.6M	0.399	0.688	0.449	12.6M	0.410	0.694	0.454
MorphPool	3	5	0	**0.425**	**0.707**	**0.476**	12.6M	**0.428**	**0.714**	**0.479**	12.6M	**0.442**	**0.720**	**0.487**

Table 2: **SUN&NYU Segmentation on Depth.** Additional results on two more datasets, NYUv2 and SUN-RGBD using Depth input. MorphPool outperforms the alternatives.

	SUN-RGBD						NYU					
	None			Conv			None			Conv		
	mIoU	acc	bf	mIoU	acc	bf	mIoU	acc	bf	mIoU	acc	bf
Linear	0.349	0.711	0.308	0.398	0.741	0.339	0.305	0.597	0.175	0.320	0.609	0.176
Standard Pool	0.208	0.643	0.255	0.297	0.687	0.294	0.121	0.453	0.177	0.159	0.495	0.173
MorphPool	**0.382**	**0.735**	**0.346**	**0.412**	**0.748**	**0.364**	**0.323**	**0.607**	**0.200**	**0.357**	**0.627**	**0.208**

4.1 Semantic Segmentation on Depth

First, semantic segmentation experiments are performed on depth input. Morphological operations are expected to perform well, because depth maps are inherently non-linear in a morphological manner. That is, morphology allows data to be probed by structuring elements in space; depth is a geometric modality, not a visual one. Results are shown for 2D-3D-S in Table 1, and for SUN-RGBD and NYUv2 in Table 2. Performance is measured in mean Intersection over Union (mIoU), pixel accuracy, and a boundary F1-score [8] to measure performance at semantic edges. From the results, note:

- Standard unpooling is outperformed significantly by flat morphological unpooling; introduction of sparse features before (de)convolution hurts performance significantly.
- It is expected that for a pooling stride of 2, the minimum unpooling with which to fill the features completely is 3. Moreover, with a pooling stride of 3, an unpooling stride of 5 is required. Both are confirmed.
- Performing strided convolution as down-sampling and bilinear interpolation in combination with convolution for up-sampling does not match performance of morphological sampling on depth data. The addition of two full convolution layers, one for down-sampling and one for up-sampling introduces many parameters to the network. When flat morphological operations are used, there are no additional parameters.
- For these runs, there is a numerical difference between using convolution and deconvolution. It is expected that these differences fall within a margin due to initialisation; for other experiments, the differences are not so obvious. Convolution and deconvolution should in principle be equivalent. Therefore, only convolution will be used in reporting results from now on. The complete results can be verified in Supp. **B**.
- Morphological pooling & pooling perform best at semantic boundaries, denoted by bf-score. This is in line with the conclusions from [3, 22], although a fully morphological sampling outperforms those methods.

Table 3: **Parameterisation Results.** Three different parameterisations for both the morphological pooling & unpooling are compared, showing that there is a small benefit to using non-flat SEs. The SEs introduce several thousands (K) additional parameters.

		SUN-RGBD						NYU					
		None			Conv			None			Conv		
	#params	mIoU	acc	bf	mIoU	acc	bf	mIoU	acc	bf	mIoU	acc	bf
Flat	0	0.382	0.735	0.346	0.412	0.748	0.364	0.323	0.607	0.200	0.357	0.627	0.208
Parabolic	4.0K	0.396	0.741	0.351	0.412	**0.751**	0.362	0.348	0.627	0.205	**0.360**	**0.630**	**0.212**
General	67.5K	**0.398**	**0.745**	**0.352**	**0.416**	0.748	**0.366**	**0.353**	**0.632**	**0.207**	0.357	**0.630**	0.207

Table 4: **Depth-wise Convolution Results**. It is possible to use depth-wise convolutions to equalise the number of parameters available during up-sampling. Gray cells indicate networks that have the same number of available parameters to learn interpolation for up-sampling, either morphologically or linearly. MorphPool outperforms the linear setting.

Up	None				Depth-wise Conv				Conv			
Down	#params	mIoU	acc	bf	#params	mIoU	acc	bf	#params	mIoU	acc	bf
Conv	12576576	0.349	0.711	0.308	12626176	0.400	0.744	0.333	47494976	0.411	0.746	0.341
Depth-wise Conv	17856	0.356	0.716	0.313	67456	0.370	0.730	0.319	34936256	0.389	0.738	0.329
Standard Pool	0	0.208	0.643	0.255	49600	0.285	0.693	0.285	34918400	0.333	0.720	0.313
MorphPool	0	0.382	0.735	0.346	49600	0.407	0.753	0.351	34918400	0.427	**0.757**	**0.371**
Para. MorphPool	3968	0.396	0.741	0.351	53568	0.412	0.754	0.354	34922368	0.427	**0.757**	0.366
General MorphPool	67456	**0.398**	**0.745**	**0.352**	117056	**0.415**	**0.755**	**0.357**	34985856	**0.432**	**0.757**	**0.371**

Besides the results listed above, initial experiments also indicated that mixing linear and non-linear sampling (*e.g.* down-sampling by standard pooling and up-sampling by convolution) had no practical benefits. With the natural non-linearities of morphology available, mixing linear and non-linear operations does not seem worthwhile.

Parameterisation of the Structuring Element Besides using a purely flat structuring element (SE) from Equation 4, it also also possible to parameterise MorphPool. This could be done using parabolic SEs from Equation 6 or by a general parameterised SE: each value on the discrete kernel then has its own parameter that can be freely adjusted. Results are listed in Table 3 and indicate parameterising the structuring elements is helpful. General parameterisation of the SEs performs best, although it also introduces most parameters. Note that compared to introducing an additional convolution layer (*i.e.* millions of parameters), introducing thousands of parameters for the structuring elements is much more computationally reasonable. General parameterisation of the SEs is used for further experiments.

It is possible to compare networks that have the same number of parameters as well. For this, depth-wise convolutions [7] are used instead of full convolutions; depth-wise convolutions only operate along spatial dimensions. Results for depth input on the SUN-RGBD dataset are listed in Table 4. Note especially the gray cells indicating a morphological and linear setting with the same number of parameters. MorphPool still outperforms the linear setting.

4.2 Semantic Segmentation on RGB

Semantic segmentation is also performed on RGB input (see Table 5). It can be concluded that, again, regular pooling and unpooling yields worse performance than generalised morphological pooling and unpooling on semantic segmentation. Now, convolution and interpolation perform on par with morphology. Note, however, that the comparison is not completely fair due to the increased number of parameters convolution has available. Interpolation networks have 50% more parameters than morphological methods. Like for semantic

Table 5: **Segmentation on RGB.** Similar to depth, morphological pooling & unpooling outperforms standard pooling & unpooling. Unlike depth, morphological operations are now on par with linear methods. This is not surprising given that RGB partly encodes illumination, which is described by linear methods more easily than by morphological methods. In contrast, morphological pooling & unpooling clearly performs best at semantic boundaries as measured by the Boundary F1-score .

	2D-3D-S			SUN-RGBD			NYU		
	mIoU	acc	bf	mIoU	acc	bf	mIoU	acc	bf
Linear	**0.355**	0.637	0.289	0.381	0.694	0.314	0.321	0.578	0.191
Standard Pool	0.325	0.607	0.257	0.298	0.646	0.270	0.193	0.514	0.198
MorphPool	0.351	**0.638**	**0.310**	**0.388**	**0.698**	**0.337**	**0.327**	**0.586**	**0.211**

Table 6: **Depth Auto-encoding Results.** Similar to segmentation on depth, morphological operations are well suited to sampling features from a depth modality on an auto-encoding task. MorphPool outperforms linear sampling and standard pooling on this task.

	None			Conv		
	ARD (↓)	RMS (↓)	$\delta_t < 1.25$ (↑)	ARD (↓)	RMS (↓)	$\delta_t < 1.25$ (↑)
Linear	0.193	0.563	0.715	0.190	0.566	0.716
Standard Pool	0.239	0.751	0.605	0.224	0.687	0.634
MorphPool	**0.171**	**0.505**	**0.747**	**0.161**	**0.462**	**0.774**

segmentation using depth, morphological pooling & pooling for RGB input perform best at semantic boundaries. Performance of morphological pooling and unpooling on depth-based segmentation outperforms RGB-based segmentation. Compared to depth, it is understandable that RGB signals are less easily encoded using mathematical morphology: RGB does not just express geometry but also illumination, which has aspects that are well described linearly.

4.3 Depth Auto-encoding

As a final experiment, a depth auto-encoding task is solved; results are shown in Table 6. Performance is measured in Absolute Relative Distance (ARD), Root Mean Squared Error (RMS), and an accuracy δ_t of predictions within a threshold $t = 1.25$. The results show substantial improvement, confirming depth is a modality that is very well suited to the use of morphological operations.

5 Conclusions

In this paper, max pooling has been formalised as a non-parameterised non-overlapping special case of the more general morphological dilation. In view of this, the unpooling-infilling scheme can be replaced by a generalised morphological operation. The full procedure is called **MorphPool**. Experiments on two tasks and three datasets show that the proposed method outperforms other sampling schemes at much reduced parameter counts. The effect is most pronounced on depth data, which confirms the expectation that *mathematical morphology* may be the natural language to process such non-linear geometry. Given the success of morphological operations for processing depth, future research could investigate whether fully morphological encoders may be used in multi-modal networks. In that case, parallel treatment of linear and morphological features has to explored.

Acknowledgements This publication is part of the project FlexCRAFT (with project number P17-01) which is (partly) financed by the Dutch Research Council (NWO).

References

[1] Jesus Angulo. Some open questions on morphological operators and representations in the deep learning era. In *International Conference on Discrete Geometry and Mathematical Morphology*, pages 3–19. Springer, 2021.

[2] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[4] Y-Lan Boureau, Francis Bach, Yann LeCun, and Jean Ponce. Learning mid-level features for recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 2559–2566. IEEE, 2010.

[5] Vasileios Charisopoulos and Petros Maragos. Morphological perceptrons: geometry and training algorithms. In *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 3–15. Springer, 2017.

[6] Bo Chen, Gungor Polatkan, Guillermo Sapiro, David Blei, David Dunson, and Lawrence Carin. Deep learning with hierarchical convolutional factor analysis. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1887–1901, 2013.

[7] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[8] Gabriela Csurka, Diane Larlus, Florent Perronnin, and France Meylan. What is a good evaluation measure for semantic segmentation? *IEEE PAMI*, 26(1), 2004.

[9] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1538–1546, 2015.

[10] Gianni Franchi, Amin Fehri, and Angela Yao. Deep morphological networks. *Pattern Recognition*, 102:107246, 2020.

[11] Bernd Gärtner and Martin Jaggi. Tropical support vector machines. Technical report, Citeseer, 2008.

[12] Hossein Gholamalinezhad and Hossein Khosravi. Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*, 2020.

[13] Rick Groenendijk, Leo Dorst, and Theo Gevers. Geometric back-propagation in morphological neural networks. *TechRxiv preprint*, 2022. doi: https://doi.org/10.36227/techrxiv.20330667.v1.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Henk Heijmans and Rein van den Boomgaard. Algebraic framework for linear and morphological scale-spaces. *Journal of Visual Communication and Image Representation*, 13(1-2):269–301, 2002.

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[17] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE, 2009.

[18] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pages 239–248. IEEE, 2016.

[19] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616, 2009.

[20] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[21] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda: Is cuda the parallel programming model that application developers have been waiting for? *Queue*, 6(2):40–53, 2008.

[22] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[23] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. doi: 10.23915/distill.00003. URL http://distill.pub/2016/deconv-checkerboard.

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[25] Gerhard X Ritter and Peter Sussner. An introduction to morphological neural networks. In *Proceedings of 13th International Conference on Pattern Recognition*, volume 4, pages 709–717. IEEE, 1996.

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[27] Mateus Sangalli, Samy Blusseau, Santiago Velasco-Forero, and Jesus Angulo. Scale equivariant neural networks with morphological scale-spaces. In *International Conference on Discrete Geometry and Mathematical Morphology*, pages 483–495. Springer, 2021.

[28] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, pages 92–101. Springer, 2010.

[29] Jean Serra. Image analysis and mathematical morphology. 1983.

[30] Jean Serra and Luc Vincent. An overview of morphological filtering. *Circuits, Systems and Signal Processing*, 11(1):47–108, 1992.

[31] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[33] Peter Sussner. Morphological perceptron learning. In *Proceedings of the 1998 ISIC held jointly with CIRA*, pages 477–482. IEEE, 1998.

[34] Rein van den Boomgaard and Arnold Smeulders. The morphological structure of images: The differential equations of morphological scale-space. *IEEE transactions on pattern analysis and machine intelligence*, 16(11):1101–1113, 1994.

[35] Rein van den Boomgaard, Leo Dorst, Sherif Makram-Ebeid, and John Schavemaker. Quadratic structuring functions in mathematical morphology. In *Mathematical morphology and its applications to image and signal processing*, pages 147–154. Springer, 1996.

[36] Daniel Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. *Advances in Neural Information Processing Systems*, 32, 2019.

[37] Lingxi Xie, Qi Tian, Meng Wang, and Bo Zhang. Spatial pooling of heterogeneous features for image classification. *IEEE Transactions on Image Processing*, 23(5):1994–2008, 2014.

[38] Chunyan Xu, Jian Yang, Hanjiang Lai, Junbin Gao, Linlin Shen, and Shuicheng Yan. Up-cnn: Un-pooling augmented convolutional neural network. *Pattern Recognition Letters*, 119:34–40, 2019.

[39] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *2009 IEEE Conference on computer vision and pattern recognition*, pages 1794–1801. IEEE, 2009.

[40] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[41] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 international conference on computer vision*, pages 2018–2025. IEEE, 2011.

[42] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. *Advances in neural information processing systems*, 27, 2014.