Generalized Dilation Structures in Convolutional Neural Networks

Gavneet Singh Chadha[a], Jan Niclas Reimann and Andreas Schwung
Department of Automation Technology, South Westphalia University of Applied Sciences, Soest, Germany

Keywords: Deep Convolutional Neural Networks, Dilated Convolution, Constrained Learning.

Abstract: Convolutional neural networks are known to provide superior performance in various application fields such as image recognition, natural language processing and time series analysis owing to their strong ability to learn spatial and temporal features in the input domain. One of the most profound types of convolution kernels presented in literature is the dilated convolution kernel used primarily for aggregating information from a larger perspective or receptive field. However, the dilation rate and thereby the structure of the kernel has to be fixed a priori, which limits the flexibility of these convolution kernels. In this study, we propose a generalized dilation network where arbitrary dilation structures within a specific dilation rate can be learned. To this end, we derive an end-to-end learnable architecture for dilation layers using the constrained log-barrier method. We test the proposed architecture on various image recognition tasks by investigating and comparing with the SimpleNet architecture. The results illustrate the applicability of the generalized dilation layers and their superior performance.

1 INTRODUCTION

Convolutional Neural Networks (CNN) (Fukushima and Miyake, 1982; LeCun et al., 1990) are known to provide superior performance not only in image recognition tasks but also in speech recognition, natural language processing and time series analysis (Bai et al., 2018). In image recognition, convolution kernels in the initial layers extract low level features about the spatial relationships in the input domain, which are hierarchically combined to form more complex features in the deeper layers. However, depending on the provided task at hand, considering a larger field of view in the initial layers of the model can be helpful in reducing the total number of parameters required since the effective receptive field only occupies a fraction of the theoretical receptive field (Luo et al., 2016). Therefore, we conjecture that considering larger receptive fields can improve image recognition performance of CNNs and simultaneously reducing the number of convolution layers and active parameters required. Larger receptive fields with atrous or dilation kernels have been proposed in (Fisher Yu and Vladlen Koltun, 2016) where dilation refers to the consideration of greater spatial resolutions of the convolution kernels as illustrated in Fig. 1. The dilation rate of 1 is the standard convolution operation whereas increasing the dilation rate allows for a larger receptive field without increase in the number of learnable parameters. This is achieved by adding zeroes between the weights of the convolution kernel.

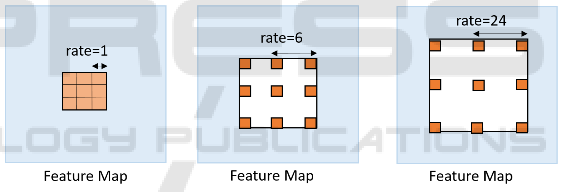

Figure 1: Dilation kernels with different dilation rates (where rate=1 is the original convolution operation) (Chen et al.,).

In this paper, we improve the previously proposed dilation neural networks in two aspects. First, we relax the original dilation structure for 2-dimensional convolution which assumes equidistant sampling in height and width of the input image. This relaxation enables varying sampling in both input dimensions as well as arbitrary patterns in the dilation kernels allowing for a more general selection of relevant input values (see Fig. 2). Second, this relaxation for varying sampling is made learnable through standard gradient descent techniques with the variant of the log-barrier methods for optimization. Through the proposed method, we aim to solve a continuous and discrete optimization problem simultaneously wherein

[a] https://orcid.org/0000-0002-9374-9074

the dilation parameters are encouraged to be binary variables, in contrast to the other parameters in the network.

The organization of the paper is as follows. Section 2 details the related work. In section 3, the dilation layer together with the proposed generalized dilation layer for CNNs is explained. Additionally, the end-to-end training technique with the log-barrier function along with its variants are detailed here. Experimental results on MNIST and CIFAR-10 image datasets along with a detailed analysis of generalized dilation layer are illustrated in section 4, with the conclusion detailed in section 5.

2 RELATED WORK

Convolutions with dilated filters were first introduced in (Holschneider et al., 1990) and (Shensa, 1992) in the context of wavelet decomposition. Dilated convolutions for deep neural networks have been initially presented in (Fisher Yu and Vladlen Koltun, 2016) to allow for multi-scale context aggregation without downsampling of the resolution. Since then, dilation networks have been used in semantic segmentation methods due to their ability to capture large context while preserving fine details. In (L. Chen et al., 2018), large dilation factor are used in the Deeplab model to provide large context, which results in improved performance. This is further enhanced in (Chen et al., 2016) by using atrous spatial pyramid pooling (ASPP), i.e. multi-level dilated convolutions, improving the results by leveraging local and wide context information. In (Sercu and Goel,), time dilated convolutions are used for dense prediction on sequences for speech recognition, while modeling long-distance genomic dependencies with dilated convolutions are reported in (Gupta and Rush, 2017). Iterated dilated convolutions are applied in (Strubell et al., 2017) for entity recognition in text modelling for improving the performance of variational autoencoders. Dilated residual networks are introduced in (Yu et al., 2017), considerably improving vanilla residual networks (He et al., 2016) in image classification and segmentation. An application of dilation to time series analysis is provided in WaveNets (van den Oord et al.,). However, in these works the dilation structure is fixed during training. In contrast, we keep the dilation structure flexible and end-to-end trainable for each channel within bounds defined by the hyperparameters.

Such training of flexible dilation structure has been presented in (He et al., 2017) where a dilation factor is trained for each channel of the convolution layer. Furthermore, the dilation factor is defined in $\mathbb{R}$ instead of $\mathbb{Z}_+$ as in the original derivation. This comes at the cost of calculating the output map by means of a bilinear transformation in the regular case of fractional dilation factors. It must be noted, that although fractional dilation factors are possible, the structure of the dilation is still fixed in this case. More general structures are allowed by active convolutions (Yunho Jeon and Junmo Kim, 2017), however with low deviations from the original convolution kernel. Deformable convolutional networks as introduced in (Dai et al., 2017) and further improved in (Zhu et al., 2019) allowing for similar convolution structures as proposed by our approach. There, each point in the convolution grid is augmented with a learnable real valued offset. As in (He et al., 2017), a bilinear transformation is needed before applying the convolution. Contrary to the other approaches, we hold on to an upper limit of integer dilation factors, while allowing for arbitrary structures of the convolution kernel within a receptive field using an end-to-end training methodology. Due to that, we end up in the exact grid position without the need for a bilinear transformation. Finally, (Carl Lemaire et al., 2018) propose a budget aware pruning methodology using the log-barrier method. On the contrary, we propose the selection of relevant structure in a receptive field for efficient feature extraction.

3 GENERALIZED DILATION NEURAL NETWORKS

In this section, we introduce generalized dilation layers starting with the description of standard dilation networks proposed in (Fisher Yu and Vladlen Koltun, 2016).

3.1 Dilation Layer

We start with the standard discrete convolution operator defined as

$$(F * k)(p) = \sum_{\mathbf{s}+\mathbf{t}=\mathbf{p}} F(\mathbf{s}) \cdot k(\mathbf{t}). \tag{1}$$

where F and k denote the receptive field and the convolution kernel respectively. In dilated convolution, we have instead

$$(F *_l k)(p) = \sum_{\mathbf{s}+l\mathbf{t}=\mathbf{p}} F(\mathbf{s}) \cdot k(\mathbf{t}), \tag{2}$$

where the term $*_l$ denotes the dilated convolution and l is the dilation factor. Note that the standard convolution operation is obtained for $l = 1$. By changing the dilation factors, the dilated convolution operator applies the same filter (with the same parameters) at a different range of the receptive field and hence, allows for better multiscale context aggregation (Fisher Yu and Vladlen Koltun, 2016). The effect of the dilation operation is illustrated in Figure 1 which shows the increasing size of the filter with increasing dilation factor.

3.2 Generalized Dilation Layers

The structure of the dilated convolution operation is kept fixed during training of a neural network. We aim to allow for more flexibility in that we want to make the dilation structure flexible and trainable. To this end, we will first derive an alternative representation of the dilation operation as follows. Basically, the dilation operation can be seen as a conventional convolution operation with receptive field size increased from $p \times p$ to $y \times y$ with $p < y$. Correspondingly, a larger weight matrix $W \in \mathbb{R}^{y \times y}$ is applied in which a certain number of weights are fixed to zero a priori such that the active weights (weights unequal zero) sum to $p \times p$. Hence, we define vectors $\psi_l \in \{0,1\}^y$ and $\psi_r \in \{0,1\}^y$ and matrices Ψ_l and Ψ_r where

$$\operatorname{diag}(\Psi_l) = \psi_l, \quad \operatorname{diag}(\Psi_r) = \psi_r. \tag{3}$$

Consequently, defining a new weight matrix as

$$\tilde{W} = \Psi_l \cdot W \cdot \Psi_r \tag{4}$$

and using this weight matrix for convolution provides a generalization to the dilation layer described previously. As an example, the dilation with active weight of size 3×3 and dilation factor of 2 can be obtained by defining $\psi_l = \psi_r = [1\ 0\ 1\ 0\ 1]^T$. Note that by arbitrarily setting the components of ψ_l and ψ_r to zero and one while at the same time forcing the total number of ones per ψ_l and ψ_r to p, arbitrary dilation structures in the weight matrix $\tilde{W}$ can be generated. Formally, we impose the constraints

$$\psi_l^T \cdot \mathbf{1} \leq p, \quad \psi_r^T \cdot \mathbf{1} \leq p, \tag{5}$$

where $\mathbf{1}$ denotes the all-one vector, yielding to arbitrary dilation-like patterns. Moreover, the dilation operation can be generalized by defining a matrix $\Psi \in \{0,1\}^{y \times y}$ and defining the new weight matrix as

$$\tilde{W} = W \odot \Psi \tag{6}$$

where $\odot$ denotes element-wise multiplication. As in the previous case, we again impose constraints on the matrix Ψ as

$$\mathbf{1}^T \Psi \cdot \mathbf{1} \leq p^2, \tag{7}$$

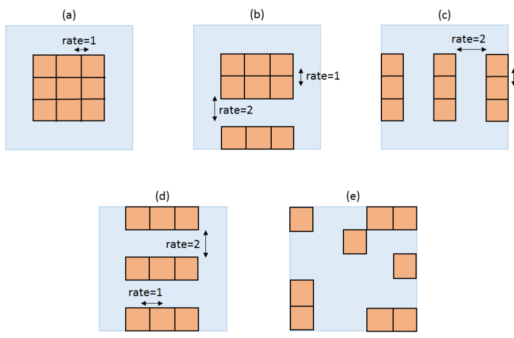

Figure 2: Different configurations of the parameters: (a) Original convolution, (b) Dilation with varying dilation rates; (c) Dilation in horizontal dimension only; (c) Dilation in vertical dimension only; (e) Arbitrary dilation kernel.

which assures, that the number of weights unequal to zero is less or equal to the kernel size. This reparameterization of the weight matrix $\tilde{W}$ allows for generalized dilation structures. An illustration of some examples for the different possible patterns is given in Figure 2. Particularly, some choices of the parameters are useful in practice. The original convolution operation (Figure 2 (a)) is obtained by setting $\psi_l = \psi_r = [0\ 1\ 1\ 1\ 0]$ while a dilation with dilation rate 2 yields $\psi_l = \psi_r = [1\ 0\ 1\ 0\ 1]$. However, this equidistant choice of the dilation might not be best to obtain representative features in some cases. This is relaxed by setting e.g. $\psi_l = [0\ 1\ 1\ 0\ 1]$ and $\psi_r = [0\ 1\ 1\ 1\ 0]$ resulting in non-equidistant patterns (see Figure 2 (b)) by incorporating the constraint in Eq. (5). Additionally, other arbitrary dilation patterns can be realized as shown in Figure 2 (e) by setting Ψ with the constraints shown in Eq. (7). Higher dilation rates within the layer can be obtained by setting $p << y$ which results in applying larger receptive fields. Additionally, considering multidimensional sequence modelling tasks where longer time horizon modelling is required, dilation patterns as shown in Figure 2 (c) using setting $\psi_l = [0\ 1\ 1\ 1\ 0]$ and $\psi_r = [1\ 0\ 1\ 0\ 1]$ can be beneficial. Similarly, a vertical dilation can be trained (Figure 2 (d)). Ultimately, arbitrary dilation patterns as illustrated in Figure 2 (e) can be obtained by setting

$$\Psi = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{8}$$

which is just restricted by the number of active parameters within the dilation kernel as presented in Eq. (7). Higher dilation rates within the layer can be obtained by setting $p << y$ which results in applying larger receptive fields.

3.3 End-to-End Training of Dilation Layers

So far we have introduced general dilation layers as an extension of the vanilla dilation layers proposed in (Fisher Yu and Vladlen Koltun, 2016) by defining suitable masking vectors and the relevant constraint that is required. However, optimizing the masking matrix and vectors results in optimizing binary variables which makes the learning problem combinatorical and does not directly allow for gradient based end-to-end training. To circumvent this, we follow a similar approach as in typical gating mechanisms (Hochreiter and Schmidhuber, 1997; Trask et al., 2018) in that we define continuous vectors $\tilde{\psi}_l, \tilde{\psi}_r \in \mathbb{R}^y$ and matrix $\tilde{\Psi} \in \mathbb{R}^{y \times y}$ which are passed through a sigmoidal activation function, i.e. we define

$$\psi_l = \sigma(\tilde{\psi}_l), \quad \psi_r = \sigma(\tilde{\psi}_r), \quad \Psi = \sigma(\tilde{\Psi}). \tag{9}$$

Using this reparameterization, the masking vector and matrix are bounded to the interval $[0,\ 1]$ and due to the characteristics of the sigmoid function will tend to its boundaries as training proceeds. The above reparameterization helps in approximating a discrete optimisation problem into a continuous one. Therefore, we end up with a trainable structure with the standard set of network parameters ω consisting of the kernel weights of the convolution and fully connected layers, and the proposed parameter vectors or matrices Ψ. Since we now have a continuous approximation of the binary variables, these can be trained by backpropagation using off the shelf gradient descent optimizers. However, we have to consider the additional constraints on the parameters Ψ as given in Eq. (5) and (7). These constraint have to be fulfilled during training and hence, should be imposed as hard constraints.

In general, various approaches exist to incorporate hard constraints in stochastic gradient descent algorithms, namely barrier functions, projection methods and active set methods (Boyd and Vandenberghe, 2004). In this work we propose to use barrier functions which contribute directly to the gradients and can approximate a hard constraint if the barrier penalty is high enough. To this end, we include the constraints by implementing them as a direct gradient $\nabla_C(\Psi)$, based only on the layer-inherent masking matrix Ψ of the form

$$\nabla_{\Psi_{ij}} L_b = b_r(\sigma(\Psi_{ij})) + b_c(\sigma(\Psi_{ij})) + b_a(\sigma(\Psi_{ij})), \tag{10}$$

where Ψ_{ij} denotes the elements of Ψ, L_b denotes the barrier loss and b_r, b_c and b_a represents the differentiable barrier function calculated over the row, column and all entries respectively. The barrier function chosen is a combination of exponential and linear as in Eq.(11) or quadratic function as in Eq.(12) where the penalty for exceeding the constraint is higher to keep the number of total active parameters in check.

$$b_r(x) = b_c(x) = \max(e^{\alpha_1 \cdot (x-p)} \cdot \alpha_2 \cdot (x-p), \alpha_3), \tag{11}$$

$$b_a(x) = \max(e^{\alpha_1 \cdot (x^2-p^2)} \cdot \alpha_2 \cdot (x^2-p^2), \alpha_3), \tag{12}$$

Different barrier functions with the different parameters α_1, α_2 and α_3 are illustrated in Fig. 3. Finally,

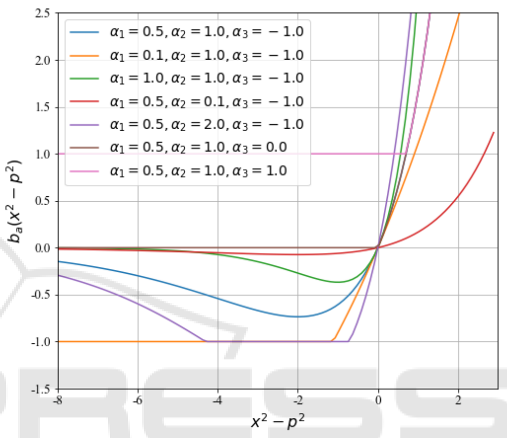

Figure 3: Barrier functions with different parameters.

the gradient for end-to-end training of the masking parameters yields

$$\nabla_{\Psi_{ij}} L_g = \frac{\partial L_s(\omega, \Psi)}{\partial \Psi_{ij}} + \nabla_{\Psi_{ij}} L_b, \tag{13}$$

where L_s is the classification loss function and L_g is the combined loss function. We do not scale the barrier loss L_b with an additional hyper-parameter, and use the barrier loss as it is. It can be useful to use a scaled version of the barrier loss, but this can also be accomplished with a change in one of the parameters α_1, α_2 and α_3. Therefore, we test with different barrier parameters and not a regularization hyperparameter. Since the value of the barrier function is used directly as the gradient for the masking parameters, negative constraint values (negative in the barrier function) result in a parameter increase, while positive constraint values result in a parameter decrease. The extent to which the masking parameter deviates from its saturation depends on the both the classification loss $L_s(\omega, \Psi)$ and barrier loss L_b .

3.4 Initialization of Masking Parameters

The barrier functions are designed such that the gradient increases exponentially if the masking matrices are over-parameterized. Thus, the boundary conditions cannot be neglected while initializing the elements in Ψ as large values of b_r, b_c and b_a yield unstable gradients. Furthermore, large gradients steer the parameter rapidly into saturating regions, which hurts performance. This problem is solved by directly using the information about kernel and receptive field size during initialization. The idea is to initialize the masking parameters in a basin where the gradients are not huge. Therefore, the parameters can be initialized such that the total sigmoidal sum of all the elements in Ψ does not exceed the constrained size of the masking matrix. Notice, that all barrier values return 0 for $\sigma(\Psi) = x = p$, thus the elements in the masking matrix are initialized by

$$\Psi = -\ln(\frac{y^2}{p^2} - 1) \pm 1. \tag{14}$$

This assures, that on average, the total sum of all sigmoidal parameters does not exceed the field size and thus does not break any boundary condition during initialization, since

$$b_a(\sum \frac{1}{1+e^{\ln(\frac{y^2}{p^2}-1)}}) = b_a(y^2 \cdot \frac{1}{1+e^{\ln(\frac{y^2}{p^2}-1)}}) \tag{15}$$

$$= b_a(p^2) = 0 \tag{16}$$

and

$$b_{r,c}(\sum \frac{1}{1+e^{\ln(\frac{y^2}{p^2}-1)}}) = b_{r,c}(y \cdot \frac{1}{1+e^{\ln(\frac{y^2}{p^2}-1)}}) \tag{17}$$

$$= b_{r,c}(\frac{p^2}{y}) < 0. \tag{18}$$

4 EXPERIMENTS

We test the proposed generalized dilation neural networks (*GDNN*) on two different benchmark image recognition tasks. We use the MNIST (LeCun et al., 2010) and CIFAR-10 (Krizhevsky and Hinton, 2010) data sets and compare the results with baseline networks. We analyse the effect of the different elements in the Generalized Dilation layer and report the results of the learnt masking and convolution filters.

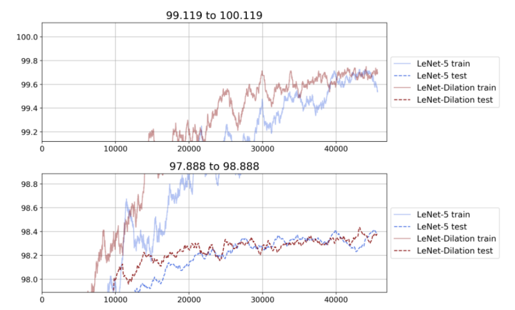

Figure 4: Accuracy graphs during training and testing over mini-batches for LeNet and the best performing generalized dilation network.

4.1 Experimental Results on MNIST

The initial experiments on MNIST were performed as a sanity check for the proposed architecture. The architecture chosen for the experiments was the same as the LeNet (LeCun et al., 1998) architecture with the first convolution layer being the generalized dilation (*GD*) layer with the receptive field size of $y = 9$. Both the architectures were trained for 400 epochs with a mean test accuracy of 98.41 % and 98.43% for the *GDNN* and the LeNet architecture respectively. However, the accuracy of the proposed method is consistently better during training as illustrated in Fig. 4 where a zoomed in training and testing progress over mini-batches is shown. Only during the fag end of the training the standard LeNet architecture improves its accuracy. After the initial test with MNIST, the CIFAR-10 dataset was used for a more comprehensive investigation.

4.2 Experimental Results on CIFAR-10

In order to analyze the performance of the *GDNN* we chose the SimpleNet network (Hasanpour et al., 2016) as a baseline architecture to investigate the performance on the CIFAR-10 dataset. The Adam (Kingma and Ba, 2014) optimizer with default parameters ($\mu = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, $\lambda = 0$) was used for 400 training epochs. If not stated otherwise, we use the same experimental settings and augmentation methods as in (Hasanpour et al., 2016). An initial set of experiments were performed to determine the optimal location of the *GD* layer in the SimpleNet architecture. The performance of the model was analysed based on the performance on a held out validation set. From this analysis, the *GD* layer was found to perform best as the second (b2) and fifth (b5) layer in the SimpleNet architecture for the CIFAR-10 dataset. This points to an observation that in simple image recognition tasks, the context aggregation fea-

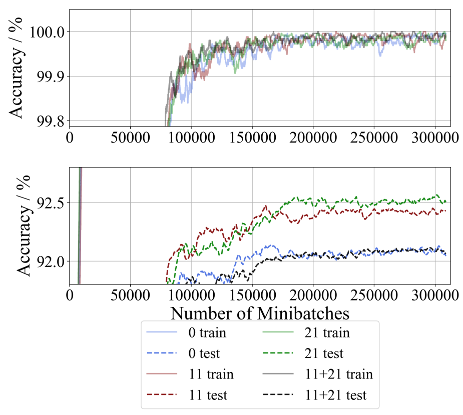

Figure 5: Training (top) and testing (bottom) accuracy graphs for Simplenet for the two best performing generalized dilation networks and a combination net with 2 *GD* layers.

ture of dilation kernels is useful in the earlier layers where low level features are captured rather than the deeper layers. Therefore, all the subsequent experiments were performed with these position of the *GD* layer respectively which are represented in the top and bottom half of the Table 1. We investigate the effect of: i. Changes in the size of receptive fields and convolution kernel, ii. Changes in the type of constraint namely row, column and all constraints, iii. Changes in the barrier function parameters namely the α, α_2 and α_3 and, and iv. Change in the number of channels of the GD layer. The final accuracy and loss reported in Table 1 are the best obtained values from the network configuration with the highlighted ones being the best for the respective position of the *GD* layer. Each distinct architecture tested is assigned a number denoted in the first column with 0 being the vanilla SimpleNet architecture. We analyze the effect of individual hyper-parameters which are highlighted blue in each row. All the individual hyper-parameters are explained in the caption of Table 1.

The foremost hyperparameter change we tested, was the size of the receptive field in the *GD* layer. A receptive field size of 7 performed generally better than the its counterparts for both the GD layer positions. Alternatively, increasing the channels for $P = 2$ yielded a better test accuracy while an increase in number of channels for $P = 5$ resulted in a decrease in accuracy. This can be attributed to the bottleneck effect where an increase in the number of output channels at the crucial bottleneck layer deteriorates performance. A clear evidence for which type of constraint to be used is not available since the change in performance is not significant enough. The effect of different barrier function parameters α_1, α_2 and α_3 is covered in the next section from the initial and learnt

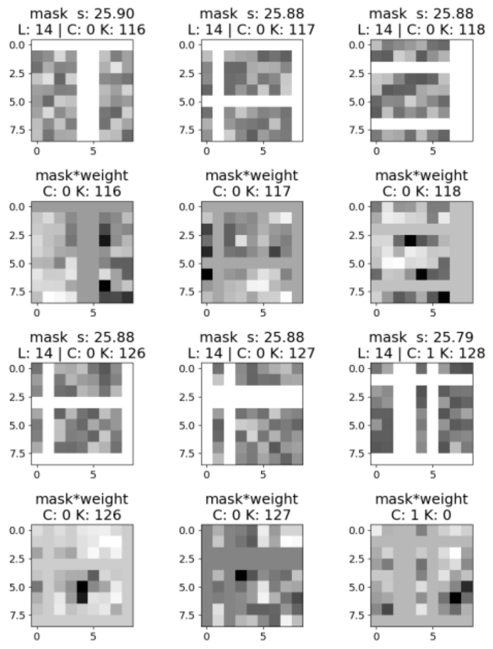

Figure 6: Trained masking matrices ($\sigma(\Psi)$) and resulting convolution kernel matrix for architecture number 3. White pixels in the masking matrix represent $\sigma(\Psi) = 0$, while black pixels represent $\sigma(\Psi) = 1$. Conversely, for the convolution kernel matrix ($\sigma(\Psi) \cdot k$), white and black pixels represent the corresponding minimum and maximum value for each kernel separately. Notice, that a zero in the masking matrix results in a constant region in the convolution matrix. s represents $\sum \Psi_k$, where k indicates the kernel number.

masking matrix distribution, but a stark change in the final performance of the network is not evident. Tests were also performed with two GD layers in the same network resulting in satisfactory performance but not better than the individual networks. It is evident from Table 1, that most permutations have a slight effect on the final test accuracy from the trained network. Apart from one *GDNN* network, all other network outperform the baseline SimpleNet (denoted by Nr. 0) architecture with a higher average test accuracy.

Figure 5 illustrates the training progress of the best performing architectures (Architecture Number 0, 11, 21 and 11+21) for each case. The figures show a zoomed in version to better distinguish among the architectures. The training progress shows both *GDNN* consistently and considerably outperforms the SimpleNet architecture in terms of testing accuracy.

4.2.1 Dilation Kernels and Masking Parameters

A sample of the trained masking matrices and the effective convolution filters are shown in Fig. 6 as greyscale maps where the headings on each of the masking maps represent the sum of all the sigmoidal parame-

Table 1: Different permutations of network architectures and their final training and testing results. Highlighted entries represent hyperparameters deviating from the baseline nets and the highest accuracy or lowest loss respectively. Exp. Nr.= Distinct Architecture Number, P = Position of Generalized Dilation layer, L = Number of GD Layers, C = Number of GD channels, p = Convolution kernel size, F = Size of Receptive Field, Constraints are defined as r = only rows, c = only columns and a = all.

Arch. Nr.	P	L	C	p	F	α_1	α_2	α_3	Constraints	Train		Test	
										Loss $\cdot 10^{-2}$	Acc. %	Loss $\cdot 10^{-2}$	Acc. %
0	2	-	128	3	-	-	-	-	-	0.042	99.98	66.64	91.99
b2	2	1	128	3	7	0.5	1	-1	r,c,a	0.040	99.98	64.74	92.32
1	2	1	128	3	*9*	0.5	1	-1	r,c,a	0.048	**100.0**	68.47	91.82
2	2	1	128	*5*	7	0.5	1	-1	r,c,a	0.088	99.97	66.62	92.38
3	2	1	128	*5*	*9*	0.5	1	-1	r,c,a	**0.010**	**100.0**	67.84	91.53
4	2	1	128	3	7	0.5	1	-1	*r,c*	0.045	99.84	**61.79**	92.15
5	2	1	128	3	7	0.5	1	-1	*r*	0.053	99.98	64.02	92.23
6	2	1	128	3	7	0.5	1	-1	*a*	0.068	99.97	64.78	92.15
7	2	1	128	3	7	*0.1*	1	-1	r,c,a	0.015	**100.0**	64.67	92.22
8	2	1	128	3	7	0.5	*0.1*	-1	r,c,a	0.068	99.98	67.06	92.00
9	2	1	128	3	7	0.5	1	*0*	r, c, a	0.056	99.98	67.37	92.01
10	2	1	*64*	3	7	0.5	1	-1	r,c,a	0.086	99.97	66.95	91.58
11	2	1	*256*	3	7	0.5	1	-1	r,c,a	0.018	**100.0**	66.09	**92.48**
b5	5	1	256	3	7	0.5	1	-1	r,c,a	0.055	99.97	62.64	91.94
12	5	1	256	3	*9*	0.5	1	-1	r,c,a	0.026	99.98	62.64	91.56
13	5	1	256	*5*	7	0.5	1	-1	r,c,a	0.019	**100.0**	62.28	91.84
14	5	1	256	*5*	*9*	0.5	1	-1	r,c,a	0.038	99.97	62.31	92.08
15	5	1	256	3	7	0.5	1	-1	*r,c*	0.169	99.97	62.47	92.17
16	5	1	256	3	7	0.5	1	-1	*r*	0.055	99.98	62.91	92.15
17	5	1	256	3	7	0.5	1	-1	*a*	0.008	**100.0**	**59.03**	92.13
18	5	1	256	3	7	*0.1*	1	-1	r,c,a	0.049	99.98	60.61	91.91
19	5	1	256	3	7	0.5	*0.1*	-1	r,c,a	0.042	99.98	58.62	92.23
20	5	1	256	3	7	0.5	1	*0*	r, c, a	**0.007**	**100.0**	59.18	92.20
21	5	1	*128*	3	7	0.5	1	-1	r,c,a	0.206	99.92	59.69	**92.44**
22	5	1	*512*	3	7	0.5	1	-1	r,c,a	0.065	99.97	62.99	91.96
11+21	2,5	1,1	128,256	3	7	0.5	1	-1	r,c,a	0.024	99.98	64.19	92.08

ters, layer number, channel number and kernel number. It is evident here that the optimizer separates out specific columns and rows, based on the constraints i.e. the difference between the receptive field size and the kernel size is big enough. From Fig. 6, a generalized dilation kernel is clear where only the most useful parts from the receptive fields are considered for training the network. This behaviour can be suppressed or emphasized depending on the α and other barrier parameters. For similar sizes between receptive fields and kernel size, the final masking matrices tend to be the same value for each entry, since the constraints are not forcing enough.

To better visualise the training of the proposed masking matrices, a distribution plot of the initial and learned masking parameters for solitary *GD* layer networks, specifically architecture number 11 and 21 (top and middle) and a network with two *GD* layers (bottom) is shown in Figure 7.The initialization was consistent in all the cases because of the initialization methodology of Ψ mentioned in Eq. (14). Surprisingly, the layers of the network with two GD layers are trained to almost the same distribution. This can be attributed to the specific configuration of barrier functions where the learned distribution seems to be the optimal distribution. In the case of two *GD* layer network, the barrier function from the last generalized dilation layer do not have a direct impact on the gradients backpropagated to the previous layers, but the varying output of the first generalized dilation layer should result in a different input images for the latter layers. The fact that the distributions are still the same can only be interpreted as an optimal solution for these chosen hyperparameters.

Changing the barrier function hyperparameter α_1, results in a small but noticeable difference in classification accuracy, but the masking distributions change drastically. The masking distributions from

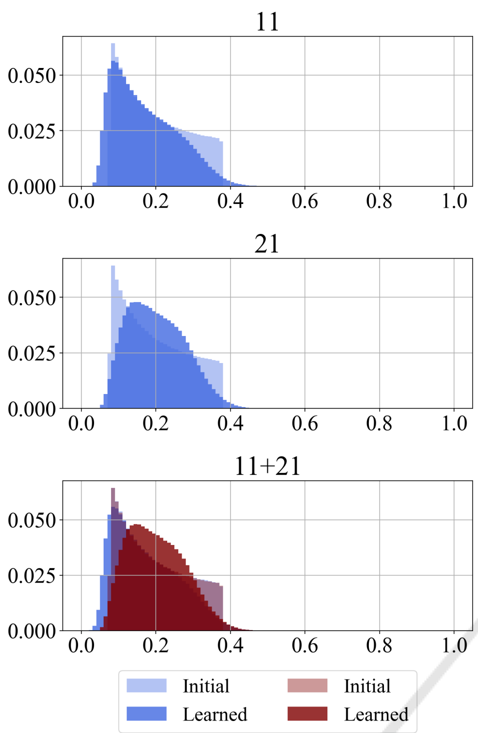

Figure 7: Distribution of masking parameters $\sigma(\Psi)$ for architecture number 11 (top) , 21 (middle) and $11+21$ (bottom).

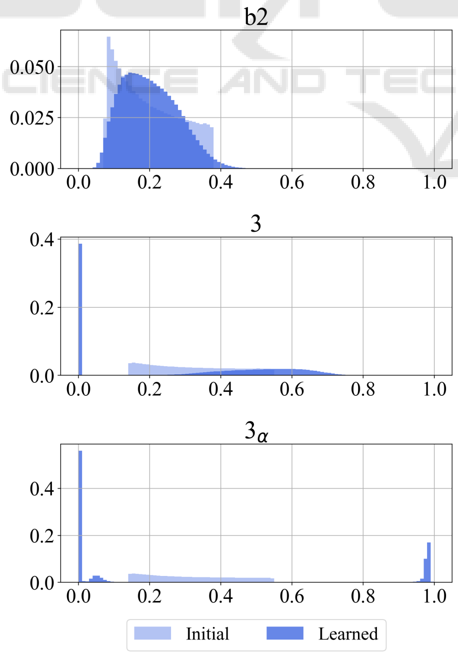

Figure 8: Distribution of masking parameters $\sigma(\Psi)$ for the architecture number $b2$ (top), 3 (middle) and 3_α (bottom).

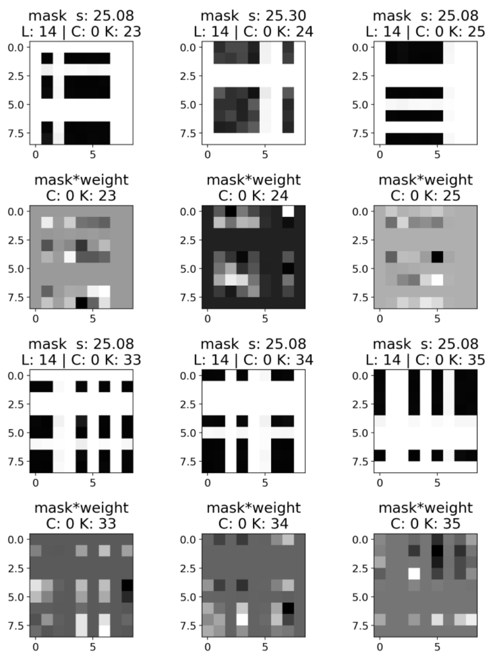

Figure 9: Trained masking matrices ($\sigma(\Psi)$) and resulting convolution matrix for architecture number 3. White pixels in the masking matrix represent $\sigma(\Psi) = 0$, while black pixels represent $\sigma(\Psi) = 1$. For the convolution matrix $(\sigma(\Psi) \cdot k)$, white and black pixels represent the corresponding minimum and maximum for each kernel separately. Notice, that a zero in the masking matrix results in a constant region in the convolution matrix. s represents $\sum \sigma(\Psi_k)$, whereas k indicates the kernel number.

architecutre numbers $b2$, 3 and 3 with an additional increase in α_1, from hereon called 3_α, are shown in Figure 8. Network $b2$ has a very similar masking distribution compared to 11, 21 and $11+21$ since all parameters regarding the constraint function are the same, they only differ in the convolution kernel and receptive field size. However, the networks 3 and 3_a show a very different distribution of masking parameters: $\sigma(\Psi)$ tends to be close to zero or one in most cases, pointing to a binary learnt maskings that have converged in choosing specific pixels completely and ignoring other pixel values. This behaviour is also visible in Figure 9 wherein the masks from architecture 3 have blocked out certain areas of the feautre maps that are not relevant for the classification task. Therefore, with an appropriate choice of the barrier function parameters, a continuous approximation for a discrete optimization problem has been demonstrated.

5 CONCLUSION

We present generalized dilation neural networks, a sub-module within the CNN architecture augmented with dilated filters. The generalization to the existing framework is provided by two extensions. First, the fixed dilation filters are made learnable by introducing an alternative continuous representation of the dilation operation using masking vectors or matrices which can be trained by standard gradient descent optimizers. To this end, we introduce a novel barrier function approach together with a suitable initialization scheme to account for the constraints imposed on the masking parameters. According to the authors, this is the first study which makes use of such techniques for constraining the dilation operation in a CNN. Second, we generalize the fixed structure of dilation kernels to arbitrary structures, allowing for an arbitrary coverage of the input space. We provide experimental evidence by testing the proposed architecture on two benchmark image recognition data-sets. The learned masking maps and distributions point to a discrete optimization of parameters using continuous gradient descent methods.

Although not presented in the results, the generalized dilations can also be applied on the whole input image rather than a receptive field, leading to a form of barrier function based attention. Using this architecture as the very first layer of a CNN with an receptive field size equal to the input size, the layer is forced to select only certain discriminative pixels from the actual input. Also part of the future work is the use of barrier function for selecting a certain number of channels in the convolution layers where complete channels will be masked out from training leading to a sparse network. On the application side, we will experiment with *GDNN* on domains like image segmentation, object detection and sequential modelling. Such tasks are suitable as the networks need to produce even denser predictions, for e.g. a prediction for each pixel.

REFERENCES

Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Boyd, S. P. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press, Cambridge.

Carl Lemaire, Andrew Achkar, and Pierre-Marc Jodoin (2018). Structured pruning of neural networks with budget-aware regularization. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9100–9108.

Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. Rethinking atrous convolution for semantic image segmentation.

Chen, L.-C., Yang, Y., Wang, J., Xu, W., and Yuille, A. L. (2016). Attention to scale: Scale-aware semantic image segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3640–3649.

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. (2017). Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773.

Fisher Yu and Vladlen Koltun (2016). Multi-scale context aggregation by dilated convolutions. In *ICLR*.

Fukushima, K. and Miyake, S. (1982). Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer.

Gupta, A. and Rush, A. M. (2017). Dilated convolutions for modeling long-distance genomic dependencies. *bioRxiv*.

Hasanpour, S. H., Rouhani, M., Fayyaz, M., and Sabokrou, M. (2016). Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. *arXiv preprint arXiv:1608.06037*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, Y., Keuper, M., Schiele, B., and Fritz, M. (2017). Learning dilation factors for semantic segmentation of street scenes. In *German Conference on Pattern Recognition*, pages 41–51.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Holschneider, M., Kronland-Martinet, R., Morlet, J., and Tchamitchian, P. (1990). A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pages 286–297. Springer.

Kingma, D. P. and Ba, J. L. (2014). Adam: Amethod for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Krizhevsky, A. and Hinton, G. (2010). Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7).

L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Cortes, C., and Burges, C. J. (2010). MNIST handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2.

Luo, W., Li, Y., Urtasun, R., and Zemel, R. (2016). Understanding the effective receptive field in deep convolutional neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4898–4906. Curran Associates, Inc.

Sercu, T. and Goel, V. Dense prediction on sequences with time-dilated convolutions for speech recognition.

Shensa, M. J. (1992). The discrete wavelet transform: wedding the a trous and mallat algorithms. *IEEE Transactions on Signal Processing*, 40(10):2464–2482.

Strubell, E., Verga, P., Belanger, D., and McCallum, A. (2017). Fast and accurate entity recognition with iterated dilated convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2670–2680, Copenhagen, Denmark. Association for Computational Linguistics.

Trask, A., Hill, F., Reed, S. E., Rae, J., Dyer, C., and Blunsom, P. (2018). Neural arithmetic logic units. In *Advances in Neural Information Processing Systems*, pages 8035–8044.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio.

Yu, F., Koltun, V., and Funkhouser, T. (2017). Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 472–480.

Yunho Jeon and Junmo Kim (2017). Active convolution: Learning the shape of convolution for image classification. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1846–1854.

Zhu, X., Hu, H., Lin, S., and Dai, J. (2019). Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9308–9316.