MGiaD: Multigrid in all dimensions. Efficiency and robustness by weight sharing and coarsening in resolution and channel dimensions*

Antonia van Betteray Matthias Rottmann Karsten Kahl
University of Wuppertal
{betteray, rottmann, kkahl}@uni-wuppertal.de

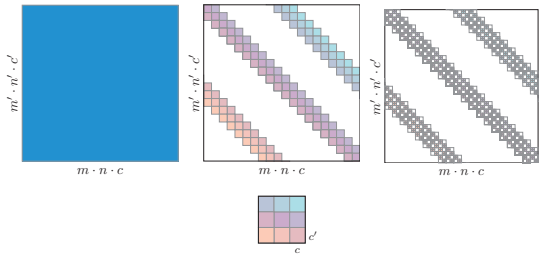
Abstract

Current state-of-the-art deep neural networks for image classification are made up of 10–100 *million learnable parameters, i.e. weights. Despite their high classification accuracy these networks are heavily overparameterized. The complexity of the weight count can be considered as a function of the number of channels, the spatial extent of the input and the number of layers of the network. Due to the use of convolutional layers the scaling of weight complexity is usually linear with regard to the resolution dimensions, but remains quadratic with respect to the number of channels. Active research in recent years in terms of using multigrid inspired ideas in deep neural networks have shown that on one hand a significant number of weights can be saved by appropriate weight sharing and on the other that a hierarchical structure in the channel dimension can improve the weight complexity to linear. Utilizing these findings, we introduce an architecture that establishes multigrid structures in all relevant dimensions, contributing a drastically improved accuracy-parameter trade-off. Our experiments show that this structured reduction in weight count reduces overparameterization and additionally improves performance over state-of-the-art ResNet architectures on typical image classification benchmarks.*

1. Introduction

Deep convolutional neural networks (CNNs) have proven to be among the most powerful methods for image recognition tasks [20, 27, 12].

In general, current state-of-the-art CNN architectures for computer vision easily comprise $\mathcal{O}(10^7)$–$\mathcal{O}(10^8)$ learnable weights. This large amount of weights entails the risk of overparameterization which can lead to poor generalization. Thus, weight count reduction is desirable, however it may induce an undesirable bias. This trade-off is referred to as "bias-complexity trade-off", which constitutes a fundamental problem of machine learning, see e.g. [29].



(a) fully connected (b) convolution (c) grouped

Figure 1: Sparsity patterns of NN layers in matrix representation. (1a) depicts the fully connected case. (1b) and (1c) depict the coupling in case of convolutions. The weights are colored according to the convolution kernel depicted in the second row.

In this work we address this problem by introducing a CNN architecture that achieves a more favorable bias-complexity trade-off, in terms of an accuracy-weight trade-off, by exploiting multigrid inspired ideas. Similar to state-of-the-art architectures, its weight count scales linear in the resolution dimensions and substantial reductions are achieved by appropriate weight sharing. A hierarchical structure w.r.t. the channel dimensions facilitates linear scaling of the weight count. In combination we obtain an architecture whose number of weights scales only linearly in all relevant dimensions, i.e., the input resolution and number of available channels. To motivate our approach and establish an appropriate context we make a short, abstract tour of the history of neural network (NN) development. From a theoretical point of view NNs are composed of a sequence of layers, consisting of linear mappings, combined with bi-

*This work is supported by the German Federal Ministry for Economic Affairs and Climate Action, within the project "KI Delta Learning", grant no. 19A19013Q.

ases and non-linear activation functions. The main bulk of the weights of a NN is found within the linear mappings

$$\theta : \mathbb{R}^{m\times n\times c} \longrightarrow \mathbb{R}^{m'\times n'\times c'}, \tag{1}$$

i.e., $\theta \in \mathbb{R}^{(m\cdot n\cdot c)\times(m'\cdot n'\cdot c')}$ where m, n and m', n' are the spatial dimensions and c, c' denote the channel dimension of the in- and output, respectively. Without further assumptions, these linear maps are given by dense matrices, corresponding to a fully connected layer. Such a layer then possesses $(m \cdot n \cdot c) \cdot (m' \cdot n' \cdot c')$ weights and becomes quickly intractable for growing m, m' and n, n' [29]. The assumption that learnable features are shift invariant enabled the introduction of CNNs [22, 7]. From the perspective of weight complexity and, in particular, the structure of linear weight maps, these convolutional layers can be viewed as blocked, banded matrices as illustrated in fig. 1. Denoting the stencil size by $s \times s'$, a convolutional layer has $\mathcal{O}(s \cdot s' \cdot c \cdot c')$ weights with the huge advantage that s, s' are fixed w.r.t. the resolution dimensions m, m' as well as n, n'. However, this comes at the price of slow information exchange which requires the use of many layers and incorporation of poolings to speed up the spatial exchange. Gating mechanisms, such as skip connections in residual networks, e.g. ResNets [13, 14], simplify information flow across many layers. By gradually restoring information from feature maps, the vanishing-gradient problem and accuracy saturation in CNNs is avoided.

To reduce the complexity of CNNs, [10] utilize the inherent similarity of multigrid (MG) and residual layers, which has already been pointed out in [13]. MG methods are hierarchical methods, typically used to solve large sparse linear systems of equations stemming from discretization of partial differential equations [31]. Inspired by MG, the architecture of [10], termed MgNet, finds justification to share weight tensors across multiple convolutional layers in ResNet-like structures, and thus reduces the overall weight count. Still, the weight count scales quadratically w.r.t. the number of channels.

Unfortunately, an assumption like shift-invariance in the spatial dimensions is amiss regarding the channel dimension and any attempt to manually sparsify its connectivity, i.e., by blocking or dropping connections, is typically met with significant performance loss. Attempts to automatically reduce CNN weight count while preserving most of the predictive performance include pruning [8, 23, 15, 32], neural architecture search [1, 6] as well as the development of resource-efficient architectural components [17, 28, 34, 35]. An MG perspective onto this sparsification problem is taken by [5], where the artificially limited exchange of information between channels is addressed by another hierarchical structure, termed multigrid in channels (MGIC). In this way, a linear scaling of the weight count can be determined. In this work, we pick up the recent developments in MG inspired architecture and present a new efficient CNN architecture. Our contribution can be summarized as follows:

1. We introduce an efficient ResNet-type architecture of MG inspired CNNs that exhibits improved scaling behavior w.r.t. all relevant dimensions, i.e., number of channels and layers, compared to recent architectures.

2. Our architecture combines elements introduced in [10] and [5] in a natural, albeit subtle, manner and can be fully explained using MG terminology.

3. The resulting architecture reduces overparameterization drastically, i.e. substantially cuts the weight count compared to similar residual architectures. Moreover, compared to ResNet, MgNet and MGIC, our approach achieves superior performance in terms of accuracy.

In our experiments, we compare our architecture to these architectures on various datasets. E.g., compared to ResNet18 on CIFAR-10, we reduce the weight count by a factor of 10 while sacrificing only up to 0.5 percent points (pp) in accuracy. Even a reduction of the weight count by a massive factor of 28, the loss in accuracy remains below 1 pp.

The remainder of this article is organized as follows: we discuss related works in section 2. In section 3 we elaborate on the similarities of residual networks and MG, including the development of our MG block. Ultimately, we present numerical results in section 4.

2. Related Works

Work related to our MG approach presented in this paper can be grouped into three categories, proceeding from remotely related to closely related.

Reducing the Number of Channels The reduction of weight count is often a byproduct in attempts to lower the computational footprint of an NN, e.g. to achieve real time capability. Though, pruning [8, 9, 23, 15] and sparsity-enhancing methods [3, 7] also reduce model complexity in terms of weight count while trading performance. Usually, pruning drops connections between channels after training, proving experimentally that there is redundancy in CNNs [25]. While the aforementioned approaches can be viewed as an automatic post-training treatment of NN, our scope is to find architectures with a reduced weight count pre-training that yields a favorable weight-accuracy trade-off, compared to post-training reductions.

Modified Layers Another line of research is concerned with the development of convolutional layers with improved

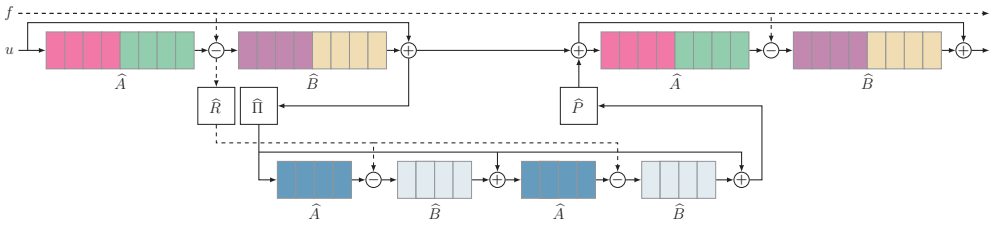


Figure 2: SiC-block on one resolution level with 2 channel levels. The convolutions $\widehat{A}$ and $\widehat{B}$ are in groups of size 4, indicated by the colors. On each channel level, $\widehat{A}$ and $\widehat{B}$ are shared between pre-smoothing (left) and post-smoothing (right), respectively. Operators on the second, coarser level, are different from the first level, but also shared. The transfer operators $\widehat{\Pi}$ and $\widehat{R}$ halve the number of channels and the prolongation mapping $\widehat{P}$ doubles the number of channels. Indices are omitted to simplify notation.

computational efficiency. Compared to the previous category, all techniques reviewed here stem from human intuition and classical methods for improving computational efficiency. One such idea is to use so-called depth-wise separable convolutions, that were introduced as a key feature of MobileNet architectures [17, 28, 16]. Serializing the spatial dimensions, the resulting convolution kernel can be viewed as a rank one matrix of dimensions given by the kernel's spatial extent s^2 times the kernel's channel extent c. While this allows to perform convolutions with less floating point operations, that approach also reduces the number of weights in the given layer from $s^2 \cdot c$ to $s^2 + c$. Another approach to reduce the computational effort of convolutions consists of grouping channels [20, 34]. While the convolution usually acts within each group, the groups themselves are decoupled. In the $s^2 \times c$-matrix representation of the convolution kernel, this approach amounts to a block diagonal matrix. Entirely decoupling of the groups hinders the distribution of information across channels. To circumvent this issues, e.g. in ShuffleNet [35] channel shuffling and grouping are combined. Our MGiaD architecture reduces the weight count complexity without decoupling effects.

Multigrid-Inspired Architectures In scientific computing, MG methods are known to be optimal methods for solving linear systems arising from partial differential equations (PDEs) [31, 30, 2]. These methods consist of two components that act complementary on the spectrum of the system matrix, namely the smoother and the coarse grid correction. While the former treats high frequency components, the latter treats low frequency ones. MG and deep learning have many computational components in common [13, 10]. The similarity of MG and CNNs also led to different architectural developments. [18] proposed an architecture wherein every layer is a pyramid of different scaled convolutions and every layer processes coarse and fine grid representations in parallel. [10] and [11] further exploited the close connection between CNNs and MG for the development of a framework called MgNet that formulates common CNN architectures as MG methods and yields a justification for sharing weight tensors across multiple layers within a given CNN architecture. MgNet utilizes MG in spatial dimensions and is capable of reducing weight counts considerably while maintaining the model's classification accuracy. Likewise, [5] achieve a reduction of the weight count by applying MG in the channels dimensions, which naturally extends grouped convolutions in an MG fashion. The resulting CNN building block is termed multigrid-in-channels (MGIC). It is built upon grouped convolutions and performs coarsening via channel pooling, thus utilizing MG in the channel dimension. As opposed to our work, neither of the mentioned works takes a unified MG perspective onto CNNs in all dimensions.

3. Residual Learning and Multigrid Methods

In this work, the focus is on image data, characterized by resolution, i.e., a grid of pixels. Furthermore, we consider the channel dimension itself as a grid. In this section we explicate both smoothing and coarsening, the MG main concepts, pointing out similarities of ResNet and MG.

Revisting ResNet and MgNet In [10] was proposed that data-feature relations $A(u) = f$ can be optimized, if A is learnable. The right-hand-side f represents the data, u denotes the feature space, s.t. relations between data $f \in \mathbb{R}^{m\times n\times c}$ and features $u \in \mathbb{R}^{m\times n\times h}$ are given by

$$A : \mathbb{R}^{m\times n\times h} \mapsto \mathbb{R}^{m\times n\times c}, \quad \text{s.t. } A(u) = f \tag{2}$$

$$B : \mathbb{R}^{m\times n\times c} \mapsto \mathbb{R}^{m\times n\times h}, \quad \text{s.t. } u \approx B(f), \tag{3}$$

with m and n spatial dimension and c and h number of input and output channels. In that sense, we can think of A as a feature-to-data map, whereas B is a map that extracts features from an element of the data space. In general A and B are weight matrices of convolutional layers, i.e., banded matrices. To explain the connection between MG and residual networks, we ignore for now the non-linear activation functions and focus on the weight matrices. The key to un-

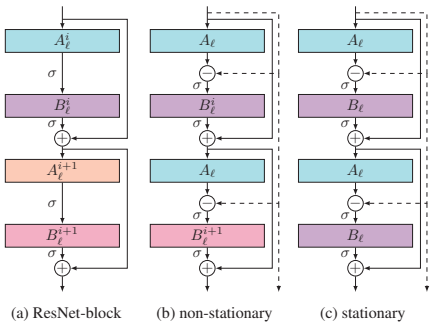


Figure 3: Weight sharing in ResNet and MgNet; (3a) ResNet-blocks, no weight sharing; (3b) MgNet-blocks, shared A_ℓ; (3c) MgNet-blocks, shared layers A_ℓ and B_ℓ.

derstanding the iterative solution of $A(u) = f$ is the observation that given any approximation $\widetilde{u}$ for the solution u the error $e = u - \widetilde{u}$ fulfills the residual equation

$$Ae = A(u - \widetilde{u}) = f - A\widetilde{u} = r. \tag{4}$$

Hence, an approximate solution $\widetilde{e} = Br$, defined by an appropriate feature extractor B, of the residual equation can be used to update the approximation $\widetilde{u} \leftarrow \widetilde{u} + \widetilde{e}$. By iterating this idea, we obtain the general structure for the solution of $A(u) = f$ by

$$u = u + B^i(f - A(u)) \quad \text{for} \quad i = 1, 2, \ldots, \tag{5}$$

starting with an initial guess $u = u_0$. Structurally, adding non-linear activation functions after any A or B^i, (5) resembles a ResNet block as has been examined in detail by [10].

Based on the interpretation of A and B^i as data-to-feature and feature-to-data maps, respectively, it makes sense to fix A (i.e., use weight sharing across multiple blocks) and to either choose a different B^i in every step, corresponding to a non-stationary iteration, or fixed as well, turning (5) into a stationary iteration. Fixing A in every block is the main difference of the MgNet architecture compared to standard ResNet. Sharing weights in A and B reduces the weight count. E.g. ResNet with k blocks without weight sharing has a weight complexity of $\mathcal{O}(2k \cdot (s^2 \cdot c \cdot h))$ whereas sharing both A and B, i.e., considering the stationary iteration case of MgNet, leads to $\mathcal{O}(2 \cdot (s^2 k \cdot c \cdot h))$, cf. fig. 3. We require the feature extractors B^i to have a convolutional structure, so it is clear, that even if chosen optimally, the iteration converges slowly as the pseudo-inverse of A is generally a dense matrix, i.e., requires a fully connected weight matrix for its representation. On the upside, convolutional B^i are cheap to apply and act locally on the data, i.e., they resolve the feature-data relation up to a certain scale, but do not encompass the whole domain. This local smoothing of features is the main observation of a MG construction, as the resulting error after a few iterations can be accurately represented on a coarser scale.

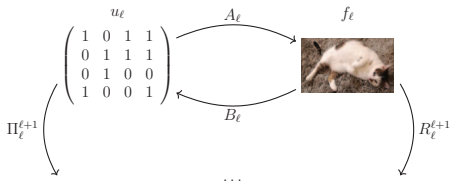


Figure 4: Data-feature relations on resolution level ℓ followed by transfer to coarser resolution $\ell + 1$.

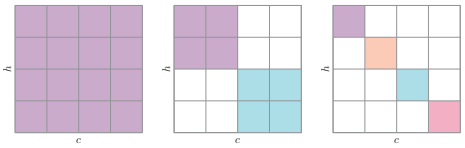


Figure 5: Channel relations in convolutions. Left: fully coupled, mid: grouped with $g_s = 2$, right: depthwise $g = c$.

Resolution Coarsening The transfer of (residual) data to coarser scales is facilitated by mappings

$$R_\ell^{\ell+1} : \mathbb{R}^{m_\ell \times n_\ell \times c_\ell} \mapsto \mathbb{R}^{m_{\ell+1} \times n_{\ell+1} \times c_{\ell+1}}, \tag{6}$$

resulting in a hierarchy of resolution levels $\ell = 1, \ldots, L$. On each level ℓ, smoothing iterations (5) are applied with resolution-wise mappings A_ℓ and B_ℓ^i. The building blocks of CNNs, equivalent to restrictions in MG, are pooling operations, which typically reduce the resolution dimensions by strides > 1, while increasing the channel dimensions. Combining smoothing iterations and restrictions we obtain algorithm 1, which can be understood as the coarsening leg (\) of a standard MG V-cycle [31].

Algorithm 1 \- MgNet(f_l)

1: Initialization: $u_\ell = 0$
2: **for** $\ell = 1, \ldots, L$ **do**
3: **for** $i = 1, \ldots, \nu$ **do**
4: $u_\ell = u_\ell + B_\ell^i(f_\ell - A_\ell(u_\ell))$
5: $u_{\ell+1} = 0$
6: $f_{\ell+1} = R_\ell^{\ell+1}(f_\ell - A_\ell(u_\ell))$

Full Approximation Scheme (FAS) for Resolution Coarsening So far, we ignored the non-linearity of the overall CNN structure due to activation functions, potentially non-linear pooling and normalization operations.

As usual in iterative methods for solving non-linear problems, the initial guess not only determines which solution is found, but also decisively influences the convergence rate. Thus transferring the current feature approximation u_ℓ to the coarser scale $u_{\ell+1}$ can make a significant difference

over choosing $u_{\ell+1} = 0$ as an initial guess. Consequently, to solve non-linear problems in MG a linear mapping

$$\Pi_\ell^{\ell+1} : \mathbb{R}^{n_\ell \times m_\ell \times c_\ell} \mapsto \mathbb{R}^{n_{\ell+1} \times m_{\ell+1} \times c_{\ell+1}}, \tag{7}$$

is introduced to initialize $u_{\ell+1} = \Pi_\ell^{\ell+1} u_\ell$. Now, that we start on resolution level $\ell + 1$ with a non-trivial initial solution, the restricted (residual) data input $f_{\ell+1}$ needs to be adjusted by adding $A_{\ell+1}(u_{\ell+1})$. This adjustment can be incorporated into algorithm 1 by changing lines 4 and 5 to

$$u_{\ell+1}^0 = \Pi_\ell^{\ell+1} u_\ell \tag{8}$$

$$f_{\ell+1} = R_l^{\ell+1}(f_\ell - A_\ell(u_\ell)) + A_{\ell+1}(u_{\ell+1}). \tag{9}$$

Clearly, $\Pi_\ell^{\ell+1}$ corresponds to yet another pooling when viewed in the CNN context, but does not have any counterpart in the general ResNet architecture. Figure 4 summarizes all mappings relevant on any resolution level ℓ.

Channel Coarsening While the convolutional and hierarchical structure of CNNs allows for an efficient treatment of the resolution dimensions, with a linear scaling of weights w.r.t. these dimensions, the situation is completely different w.r.t. the channel dimension. Here, typically no restriction is put on the connectivity structure, i.e., the convolutional maps are dense w.r.t. the coupling of input to output channels. Clearly, the full coupling of channels enables efficient exchange of information, but poses the problem of quadratic scaling in the weight count and could lead to a poor accuracy-weight trade-off due to redundancies. Assuming that the number of channels cannot be reduced, a decrease in the weight count is only possible by addressing their connectivity. Unfortunately, one cannot profit from an invariance assumption that allowed the introduction of convolutional connections in the resolution dimensions. Thus any reduction in the channel connectivity is of an ad-hoc nature. The most straightforward way to limit the connectivity is grouping channels, s.t. exchange remains only within each group, cf. fig. 5. Denoting the group size by g_s such a strategy reduces the weight count from $\mathcal{O}(s^2 \cdot c \cdot h)$ to $\mathcal{O}(s^2 \cdot \frac{c \cdot h}{g})$ [20].

While replacing A_ℓ and (or only) B_ℓ by grouped convolutions with $g_s < c$ cuts the weight count significantly, the lack of interaction between the channels also decreases the accuracy, c.f. fig. 7. To facilitate efficient channel interaction, [5] introduced a grouped restriction mapping

$$\widehat{R}_{\ell,\kappa}^{\kappa+1} : \mathbb{R}^{m_\ell \times n_\ell \times c_{\ell,\kappa}} \mapsto \mathbb{R}^{m_\ell \times n_\ell \times c_{\ell,\kappa+1}}, \tag{10}$$

with $c_{\ell,\kappa+1} = (\frac{c_{\ell,\kappa}}{2})$, halving the number of channels, e.g. as depicted for two in-channel levels in fig. 2. Corresponding to a MG V-cycle (cf. [31]) another grouped mapping

$$\widehat{P}_{\ell,\kappa+1}^{\kappa} : \mathbb{R}^{m_\ell \times n_\ell \times c_{\ell,\kappa+1}} \mapsto \mathbb{R}^{m_\ell \times n_\ell \times c_{\ell,\kappa}} \tag{11}$$

is defined to refine the number of channels. This prolongation map interpolates coarse level features $\widehat{u}_{\ell,\kappa+1}$ to the fine level, starting from the coarsest which uses a dense CNN-block, e.g. a ResNet-block [5]. As in MG such a coarse-level update on κ is given by

$$\widehat{u}_{\ell,\kappa} = \widehat{u}_{\ell,\kappa} + \widehat{P}_{\ell,\kappa+1}^{\kappa}(\widehat{u}_{\ell,\kappa+1}) \tag{12}$$

Using this MG strategy for the channel dimensions allows for a significant reduction in the number of weights without sacrificing much accuracy [5].

Algorithm 2 (MG) Smoothing in channels $\texttt{SiC}(f_\kappa, u_\kappa)$

1: **for** $i = 1, \dots, \eta_{\text{pre}}$ **do**
2: $\quad u_\kappa = u_\kappa + \widehat{B}_\kappa^i(f_\kappa - \widehat{A}_\kappa u_\kappa)$ ▷ pre-smoothing
3: **if** $\kappa \neq K$ **then**
4: $\quad u_{\kappa+1} = \widehat{\Pi}_\kappa^{\kappa+1}(u_\kappa)$
5: $\quad f_{\kappa+1} = \widehat{R}_\kappa^{\kappa+1}(f_\kappa - \widehat{A}_\kappa(u_\kappa)) + \widehat{A}_{\kappa+1}(u_{\kappa+1})$
6: $\quad \widehat{u}_{\kappa+1} = \texttt{SiC}(f_{\kappa+1}, u_{\kappa+1})$
7: $\quad u_\kappa = u_\kappa + \widehat{P}_{\kappa+1}^{\kappa}(\widehat{u}_{\kappa+1})$
8: **for** $i = 1, \dots, \eta_{\text{post}}$ **do** ▷ post-smoothing
9: $\quad u_\kappa = u_\kappa + \widehat{B}_\kappa^i(f_\kappa - \widehat{A}_\kappa(u_\kappa))$

Multigrid in all Dimensions: MGiaD To obtain an architecture that ultimately scales linearly in the weight count w.r.t. all problem dimensions (resolution and channels), connecting the ideas of MgNet and FAS, we introduce a (MG) smoothing in channels (SiC) block. SiC, depicted in fig. 2, uses (10) and (11) to build an in-channel hierarchy, incorporating smoothing iterations (5) with shared weights w.r.t. the in-channel level κ. To be more precise we replace the maps A_ℓ and B_ℓ^i in algorithm 1 by an in-channel V-cycle in the following way. The convolutions A_ℓ and B_ℓ^i, which are fully connected w.r.t. the channel dimensions are replaced by grouped convolutions $\widehat{A}_{\ell,1}$ and $\widehat{B}_{\ell,1}^i$, respectively. In addition we introduce grouped convolutions $\widehat{A}_{\ell,\kappa}, \widehat{B}_{\ell,\kappa}^i$ for $\kappa = 1, 2, \dots, K_\ell$ as well as restrictions $\widehat{R}_{\ell,\kappa}^{\kappa+1}$ and interpolations $\widehat{P}_{\ell,\kappa+1}^{\kappa}$ for $\kappa = 1, 2, \dots, K_\ell - 1$. In here, the number of levels K_ℓ is chosen such that $\widehat{A}_{\ell,K_\ell}$ and $\widehat{B}_{\ell,K_\ell}$ are again fully connected w.r.t. the channel dimension. The overall structure of the resulting method is sketched in fig. 6. On each channel level κ the grouped convolutions $\widehat{A}_{\ell,\kappa}$ and $\widehat{B}_{\ell,\kappa}^i$ are arranged as in (5), i.e., as an in-channel smoothing iteration. Analogous to an FAS-type restriction $\Pi_\ell^{\ell+1}$ of the current feature map in MgNet, we introduce an in-channel FAS restriction map

$$\widehat{\Pi}_{\ell,\kappa}^{\kappa+1} : \mathbb{R}^{m_\ell \times n_\ell \times c_{\ell,\kappa+1}} \mapsto \mathbb{R}^{m_\ell \times n_\ell \times c_{\ell,\kappa}}. \tag{13}$$

Clearly, both $\widehat{R}_{\ell,\kappa}^{\kappa+1}$ and $\widehat{\Pi}_{\ell,\kappa}^{\kappa+1}$ have to be grouped mappings as well in order to end up with a linear scaling of the

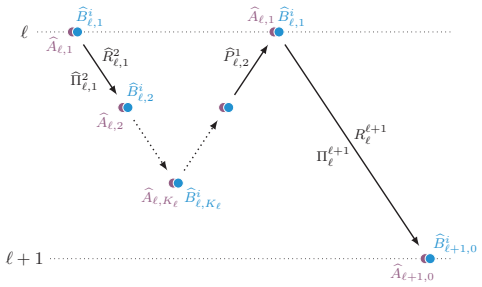


Figure 6: Structure of SiC on resolution levels ℓ and $\ell+1$.

weight count w.r.t. the channel dimension size. Our final MG smoothing in-channel algorithm (SiC) is summarized in algorithm 2.

The dependency of the number of weights is reduced from quadratic to linear scaling w.r.t. the channel dimension when compared to a fully connected structure as in ResNet or MgNet. Replacing the smoothing iteration in MgNet by in-channel-MG-blocks finally yields a MG-like architecture that achieves linear scaling of the number of weights in all dimensions. The resulting method termed multigrid in all dimensions (MGiaD) is given in algorithm 3.

Algorithm 3 MG in all dimensions $\texttt{MGiaD}(f_\ell)$

Initialization $u_1 = 0$
for $\ell = 1, \dots, L-1$ **do**
  $u_\ell = \texttt{SiC}(f_\ell, u_\ell)$
  $u_{\ell+1} = \Pi_\ell^{\ell+1}(u_\ell)$
  $f_{\ell+1} = R_\ell^{\ell+1}(f_\ell - A_\ell(u_\ell)) + A_{\ell+1}(u_{\ell+1})$

4. Experimental Setup & Evaluation

We evaluate our approach on improving the accuracy-weight trade-off for classification tasks on different popular datasets such as CIFAR-10, CIFAR-100 [19], Fashion-MNIST [33], Tiny ImageNet [21] and ImageNet [4]. We report the number of weights, train and test accuracy with standard deviation (std), obtained from three runs with random seed. Since std ≤ 0.01 holds for the train accuracy, it is omitted in the following.

Training Setup Our models are implemented in Pytorch [26]. Unless otherwise stated, we train the models with batch-size 128 for 400 epochs with an SGD-optimizer, a momentum of 0.9 and a weight decay of 10^{-4}. In accordance to ResNet [13] we use batch normalization followed by a ReLU activation function after every convolutional layer. The initial learning rate is set to 0.05 and we use a cosine-annealing learning rate schedule [24].

Evaluation of ResNet and MgNet on CIFAR-10 The CIFAR-10 dataset contains $60k$ color images of size 32×32 in 10 classes. We compare our MGiaD approach to ResNet18, ResNet20 and corresponding MgNet architectures. The ResNet18 architecture is composed of 4 resolution levels with $[64, 128, 256, 512]$ channels and 2 ResNet-blocks on each resolution level. According to [10] we reduce the number of channels on the last resolution level from 512 to 256 in MgNet. ResNet20, a ResNet version specifically designed for CIFAR-10, is made up of 3 resolution levels à 3 blocks and $[16, 32, 64]$ channels. The parameters of the MgNet architectures are chosen accordingly in either situation. We include tests of MgNet with sharing only A, referred to by MgNet$^{\mathrm{A}}$ and sharing both A and B, referred to by MgNet$^{\mathrm{A,B}}$. In our experiments we observe that sharing convolutions within the ResNet-blocks significantly reduces the weight count of the models at minor decrease in performance. E.g. ResNet18 has $11{,}174k$ weights and achieves 96.26% accuracy, while the corresponding MgNet$^{\mathrm{A,B}}$, i.e. two shared layers, achieves an accuracy of 96% with $2{,}751k$ weights. The smaller ResNet20 is built from $270k$ weights and achieves 92.44% accuracy, while the corresponding MgNet$^{\mathrm{A,B}}$ has $101k$ weights and achieves 90.58% accuracy. Based on these findings as well as those by He and Xu [11], we opt to default sharing A and B, when showing results for MGiaD. In all tests we chose learnable depthwise poolings Π and R in both MgNet and MGiaD. We include additional tests where channel grouping in MgNet architectures is used for further sparsification in A and B simultaneously. While a significant reduction in weight count is achieved, we observed in fig. 7 that the performance drops significantly. This finding can be attributed to the worse exchange of information across channels. More detailed results are provided in the supplementary material. These results indicate that a more sophisticated sparsification scheme in the channel dimension is in demand, which is provided by our MGiaD architecture.

Evaluation of MGiaD on CIFAR-10 To identify the parameters that play a role in the accuracy-weight trade-off, we performed a large-scale parameter study on CIFAR-10. We start by studying results with varying group sizes and size of coarsest level (c_K) within the channel MG subcycle. All results are summarized in fig. 7, a detailed discussion can be found in the supplementary material. The results clearly indicate that the size of the fully connected coarsest level of the MG subcycle has a significant influence on the performance of the resulting network, while the group size of the grouped convolutions $\widehat{A}$ and $\widehat{B}$ has a much smaller impact. At $g_s = 4$ and $c_K = 64$ we obtain an architecture with a weight count 30 times smaller compared to ResNet18 at a cost of only 1.5 pp of accuracy. Even compared to the slim MgNet$^{\mathrm{A,B}}$, this architecture yields compa-

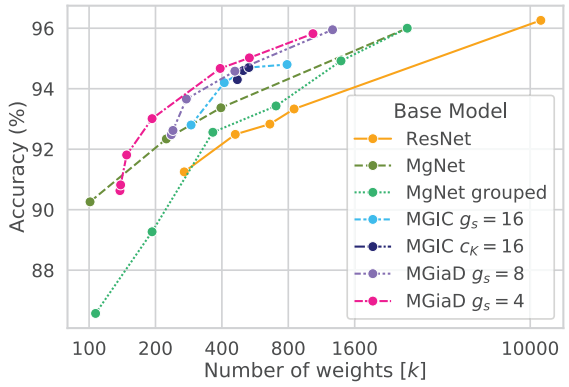


Figure 7: Accuracy-weight trade-off on CIFAR-10 for different ResNet, MgNet, MGIC and MGiaD models. For MGiaD the group size $g_s \in \{4, 8\}$ is fix and the number of channels c_K varies.

rable performance at a $7\times$ reduction in weight count. The more economical ResNet20 comes with only $270k$ weights. For MGiaD with $g_s = 8$ and $c_K = 32$ we obtain a model with similar weight count and substantially improved accuracy. For $g_s = 4$ and $c_K = 16$ we achieve roughly half the number of weights of ResNet20, while being on par accuracy-wise.

To utilize freed up capacity in terms of weights and pursue high accuracy, we introduce a channel scaling parameter λ. We use this parameter to multiplicatively scale the initial number of channels fixing $c_K = 64$ with $g_s = 4$ or $g_s = 8$. The higher number of overall channels leads to a deeper hierarchy, more parameters and improved performance, cf. fig. 7. Another way of performance improvement is the (re)-use of $\widehat{A}$ and $\widehat{B}$ in the channel MG subcycle in a fashion akin to post-smoothing in MG [31], where such a process is known to speed up the time and work to solve the problem. Correspondingly, we add multiple post-smoothing steps η_{post} to the channel hierarchy. As the weight tensors are shared on one channel level, only weights for new batch normalizations are added to the total. For most setups, multiple post-smoothing steps do not improve the accuracy, c.f. supplementary material. We compile all results for CIFAR-10 in terms of weight count vs. test accuracy in fig. 7, in which we also include results for the MGIC method by Eliasof et al. [5]. We obtain very good results using $g_s = 4$ and varying values of c_K and λ. However, there is some diminishing returns starting to set in around $500k$ weights and the best result we were able to achieve uses $g_s = 8$ combined with $\lambda = 3$. Due to reduced number of channels through the MG hierarchy and shared weight tensors, the amount of weights is cut drastically. Yet, considering the number of floating point operations (FLOPs), our MGiaD models require a similar number of FLOPs for the same performance as ResNet and MgNet, cf. table 1.

Model	λ	g_s	**#weights** (k)	**GFLOPs**	**accuracy** (%) (± **std**) test	train
ResNet18	-	-	11,170	1.12	96.26 (0.16)	98.14
MgNet^{A,B}	-	-	2,751	1.40	96.00 (0.27)	97.60
MGiaD	3	4	1,035	1.14	95.64 (0.09)	97.21
MGiaD	3	8	1,270	1.34	95.95 (0.12)	97.44

Table 1: Computational costs in terms of GFLOPs and weight count of ResNet18, MgNet and MGiaD models performing best on CIFAR-10. We report MGiaD models with group sizes $g_s = 4$ and 8, which are built with $c_K = 64$ fully coupled channels and a channel scaling of $\lambda = 3$.

Model	c_K	g_s	**#weights** (k)	**accuracy** (%) (± **std**) test	train
ResNet18	-	-	11,003	**93.84** (0.17)	100
MgNet^{A,B}	-	-	2,747	**93.84** (0.16)	100
MGiaD	64	4	389	93.45 (0.12)	100
		64	1,357	**93.84** (0.10)	100
	32	4	189	93.32 (0.06)	100
		32	437	**93.71** (0.49)	100
	16	4	420	**93.40** (0.21)	100
		16	144	93.28 (0.08)	100
ResNet20	-	-	270	93.02 (0.31)	100
MgNet^{A,B}	-	-	101	**93.29** (0.15)	100
MGiaD	16	4	28	92.85 (0.06)	97.07
		8	37	**93.35** (0.09)	97.63
		16	55	93.29 (0.15)	98.48

Table 2: Influence of the number of fully coupled channels c_K and g_s on accuracy and weight count for models trained on FashionMNIST. The best model w.r.t. accuracy of each comparison is marked in bold font and the overall best is highlighted purple.

Evaluation on FashionMNIST FashionMNIST contains $70k$ 32×32 grayscale images in 10 classes. The initial learning rate is 0.05 and multiplied by 0.1 every 25 epochs. In all our experiments we observe strong overfitting, results are presented in table 2. However, MGiaD still improves the accuracy-weight trade-off. In particular, in comparison to ResNet18, MGiaD with $c_K = g_s = 64$ achieves the same accuracy with $8\times$ less weights. Additionally, the weight count for MGiaD can be cut by another factor of 3 while sacrificing only 0.1 pp in performance. To reduce overfitting, we decrease the number of resolution levels from 4 to 3 for experiments with $c_K = 16$. The resulting architectures are compared to ResNet20, also consisting of 3 resolution levels. We indeed observe a mild decrease of overfitting along with a reduction in weight count by a factor of 2.7.

Evaluation on CIFAR-100 CIFAR-100 contains 100 classes with 600 images each (same specs as CIFAR-10). We observed for CIFAR-10 and FashionMNIST, that a high number of fully connected channels has a significant influence on the accuracy, thus we opt to choose $c_K = 64$. In table 3 we study the influence of the group size and the number of channels on the accuracy when varying the channel multiplier λ. Similar to CIFAR-10 we observe that

Dataset			CIFAR-100			Tiny ImageNet			ImageNet		
Model	λ	g_s	#weights (k)	accuracy (%) (± std) test	train	#weights (k)	accuracy (%) (± std) test	train	#weights (k)	accuracy (%) (± std) test	train
ResNet18	-	-	11,220	**75.42** (0.13)	99.98	11,271	59.67 (0.66)	91.29	11,690	**71.89** (0.04)	75.26
MgNet^{A,B}	-	-	2,774	74.42 (0.28)	99.98	2,799	**60.12** (0.25)	87.77	3,013	67.83 (0.10)	64.87
MGiaD	1	8	481	69.91 (0.37)	99.25	508	56.23 (0.50)	85.37	721	59.61 (0.14)	58.12
		64	1,384	**72.53** (0.45)	99.97	1,411	**60.36** (0.54)	70.99	1,625	**66.38** (0.36)	63.35
	2	8	745	71.48 (0.48)	99.91	801	58.33 (0.09)	64.35	1,226	64.82 (0.20)	60.40
		64	3,067	**75.12** (0.71)	99.98	3,123	**62.24** (0.99)	81.17	3,549	**72.12** (0.12)	69.67
	3	8	1,338	72.75 (0.62)	99.97	1,422	59.87 (0.13)	70.49	2,060	68.23 (0.13)	64.09
		64	4,822	**75.85** (0.14)	99.98	4,906	**62.68** (0.75)	84.71	5,544	**74.39** (0.24)	72.50

Table 3: Influence of the overall number of channels w.r.t. the resolution levels, scaled by λ on CIFAR-100 and (Tiny) ImageNet on accuracy and weight count. For MGiaD the number of fully coupled channels is set to $c_K = 64$ and we study group sizes 4 and 8. For each dataset the best model w.r.t. accuracy of each comparison is marked in boldface and the overall best is highlighted purple.

a higher weight count in general leads to better accuracy. But, compared to CIFAR-10 where the group size had a minor impact, for CIFAR-100 a big group size is essential for a high accuracy. E.g. a model with $g_s = 8$ and $481k$ weights achieves an accuracy of almost 70 pp, whereas its counterpart with $g_s = 64$ achieves over 72 pp with a way higher weight count, i.e., over $1{,}300k$. In accordance to our CIFAR-10 results, the channel scaling successfully utilizes the freed-up capacity. Multiplying the number of channels by 3 results in the best overall accuracy. The resulting model halves the weight count of ResNet18, but improves upon its accuracy by 0.4 pp.

Evaluation on Tiny ImageNet Tiny ImageNet consists of $100k$ images composed of 200 equally sized classes of 64×64 colored images. Based on previous findings we did not tune c_K to this dataset specifically, i.e., we keep $c_K = 64$. In table 3 we study accuracy and weight count both as a function of group size ($g_s = 64$ and $g_s = 8$) the channel scaling λ. We compare the resulting models with ResNet18 and corresponding MgNet. Consistent to observations on CIFAR-100 a higher weight count generally leads to a higher accuracy. Nevertheless MGiaD with $g_s = 32$ cuts the number of weights by a factor of 6 compared to MgNet, while achieving a slightly higher accuracy of 59.36%, cf. supplementary material. Increasing the group size to $g_s = 64$, MGiaD cuts the weight count by a factor of 9 compared to ResNet and a factor of 3 compared to MgNet, while improving the accuracy for both architectures by at least 0.5 pp to 60.36%. Increasing the initial number of channels by $\lambda = 3$ leads to an accuracy gain of 3 pp over ResNet18 while requiring 2.3 times fewer weights.

Evaluation on ImageNet The ImageNet database contains colored images with resolution 224×224 in 1.000 semantic classes. As for the Tiny ImageNet subset, we fix $c_K = 64$ for the number of fully coupled channels on the coarsest level and study the influence of $g_s \in \{8, 64\}$ in combination with $\lambda \in \{1, 2, 3\}$ and compare the resulting models with ResNet18 and MgNets. All models are trained with a batch size of 512, except for MGiaD with $\lambda = 3$, for which the batch size is reduced to 256 due to memory limitations. Note that these limitations could be remedied by an optimized implementation. Our results are reported in table 3. In accordance with previous results on the other datasets, an higher weight count results in higher accuracy. Again we find, that channel scaling is a powerful tool to improve accuracy by utilizing the capacity freed up by the MG hierarchy in the channel dimension. E.g. an MGiaD model with $g_s = 64$, combined with $\lambda = 2$ improves the accuracy by 0.23 pp while using 3 times less weights in comparison to ResNet18. Multiplying the number of channels by 3 with a group size of 8 in MGiaD, the weight count of MgNet is cut by one third, but achieves an improved performance by 0.54 pp. This demonstrates that our MGiaD architecture yields an improved accuracy-weight trade-off and reduces overparameterization on ImageNet.

5. Conclusion

In this work we introduced a NN architecture that utilizes the concept of MG in spatial and channel dimensions. Our experiments suggest that, although the reduction in weight count introduces an additional architectural bias, this bias does not seem to affect the overall performance of the network in most cases, in particular if the network is overparameterized. In problems requiring only limited capacity, e.g. CIFAR-10, the proposed architecture substantially reduces the weight count while almost maintain performance in terms of accuracy. These observations generalize to bigger problems such as Tiny ImageNet and ImageNet. For ImageNet we even find architectures with an accuracy superior to ResNet18 and MgNet while requiring less weights. Our approach offers another way to account for overparameterization of NNs and achieves an improved scaling behavior w.r.t. the depth and width hyperparameters. In future work, it will be of interest to study how the smoother B can be replaced by a polynomial in A to additionally reduce the weight count.

References

[1] Jose M. Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, 2016. 2

[2] William Briggs, Van Henson, and Steve McCormick. A Multigrid Tutorial, 2nd Edition. 01 2000. 3

[3] Soravit Changpinyo, Mark Sandler, and Andrey Zhmoginov. The power of sparsity in convolutional neural networks. ArXiv, abs/1702.06257, 2017. 2

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009. 6

[5] Moshe Eliasof, Jonathan Ephrath, Lars Ruthotto, and Eran Treister. MGIC: Multigrid-in-Channels Neural Network Architectures. 2020. 2, 3, 5, 7

[6] A. Gordon, E. Eban, O. Nachum, B. Chen, H. Wu, T. Yang, and E. Choi. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 1586–1595, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society. 2

[7] Song Han, Jeff Pool, Sharan Narang, Huizi Mao, E. Gong, Shijian Tang, Erich Elsen, Péter Vajda, Manohar Paluri, J. Tran, Bryan Catanzaro, and W. Dally. DSD: Dense-Sparse-Dense Training for Deep Neural Networks. ICLR, 2017. 2

[8] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS'15, page 1135–1143, Cambridge, MA, USA, 2015. MIT Press. 2

[9] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. In S. Hanson, J. Cowan, and C. Giles, editors, Advances in Neural Information Processing Systems, volume 5. Morgan-Kaufmann, 1992. 2

[10] Juncai He and Jinchao Xu. MgNet: A unified framework of multigrid and convolutional neural network. Science China Mathematics, (7):1331–1354, 2019. 2, 3, 4, 6

[11] Juncai He, Jinchao Xu, Lian Zhang, and Jianqing Zhu. An interpretive constrained linear model for resnet and mgnet. Neural Networks, 162:384–392, 2023. 3, 6

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 1026–1034, Santiago, Chile, 2015. IEEE. 1

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, June 2016. 2, 3, 6

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, Computer Vision – ECCV 2016, pages 630–645. Springer International Publishing, 2016. 2

[15] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In 2017 IEEE International Conference on Computer Vision (ICCV), pages 1398–1406, 2017. 2

[16] Andrew Howard, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, Yukun Zhu, Ruoming Pang, Hartwig Adam, and Quoc Le. Searching for MobileNetV3. In 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pages 1314–1324, Seoul, Korea (South), 2019. IEEE. 3

[17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, M. Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. ArXiv, 2017. 2, 3

[18] Tsung-Wei Ke, Michael Maire, and Stella Yu. Multigrid Neural Architectures. pages 4067–4075, July 2017. 3

[19] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. 6

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc., 2012. 1, 3 , 5

[21] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015. 6

[22] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86:2278–2324, Nov. 1998. 2

[23] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. ArXiv, abs/1608.08710, 2016. 2

[24] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. arXiv: Learning, 2016. 6

[25] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017. 2

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach 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. Curran Associates Inc., Red Hook, NY, USA, 2019. 6

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115(3):211–252, 2015. 1

[28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In 2018 IEEE/CVF Conference on Computer Vision

and Pattern Recognition (CVPR), pages 4510–4520. IEEE Computer Society, jun 2018. 2, 3

[29] Shai Shalev-Shwartz and Shai Ben-David. Understanding machine learning: from theory to algorithms. Cambridge University Press, New York, NY, USA, 2014. 1, 2

[30] Eran Treister and Irad Yavneh. On-the-Fly Adaptive Smoothed Aggregation Multigrid for Markov Chains. SIAM J. Scientific Computing, 33:2927–2949, 2011. 3

[31] U. Trottenberg, C. W. Oosterlee, and Anton Schüller. Multigrid. Academic Press, 2001. 2, 3, 4, 5, 7

[32] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, page 2082–2090, 2016. 2

[33] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. 6

[34] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5987–5995. IEEE, 2017. 2, 3

[35] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6848–6856, 2018. 2, 3