

Multigrid Neural Architectures

Tsung-Wei Ke
 UC Berkeley / ICSI

twke@icsi.berkeley.edu

Michael Maire
 TTI Chicago

mmaire@ttic.edu

Stella X. Yu
 UC Berkeley / ICSI

stellayu@berkeley.edu

Abstract

We propose a multigrid extension of convolutional neural networks (CNNs). Rather than manipulating representations living on a single spatial grid, our network layers operate across scale space, on a pyramid of grids. They consume multigrid inputs and produce multigrid outputs; convolutional filters themselves have both within-scale and cross-scale extent. This aspect is distinct from simple multiscale designs, which only process the input at different scales. Viewed in terms of information flow, a multigrid network passes messages across a spatial pyramid. As a consequence, receptive field size grows exponentially with depth, facilitating rapid integration of context. Most critically, multigrid structure enables networks to learn internal attention and dynamic routing mechanisms, and use them to accomplish tasks on which modern CNNs fail.

Experiments demonstrate wide-ranging performance advantages of multigrid. On CIFAR and ImageNet classification tasks, flipping from a single grid to multigrid within the standard CNN paradigm improves accuracy, while being compute and parameter efficient. Multigrid is independent of other architectural choices; we show synergy in combination with residual connections. Multigrid yields dramatic improvement on a synthetic semantic segmentation dataset. Most strikingly, relatively shallow multigrid networks can learn to directly perform spatial transformation tasks, where, in contrast, current CNNs fail. Together, our results suggest that continuous evolution of features on a multigrid pyramid is a more powerful alternative to existing CNN designs on a flat grid.

1. Introduction

Since Fukushima’s neocognitron [9], the basic architectural design of convolutional neural networks has persisted in form similar to that shown in the top of Figure 1. Processing begins on a high resolution input, of which filters examine small local pieces. Through stacking many layers, in combination with occasional pooling and subsampling, receptive fields slowly grow with depth, eventually

encompassing the entire input. Work following this mold includes LeNet [23], the breakthrough AlexNet [21], and the many architectural enhancements that followed, such as VGG [32], GoogLeNet [34], residual networks [11, 12, 38] and like [22, 15]. Coupled with large datasets and compute power, this pipeline drives state-of-the-art vision systems.

However, sufficiency of this design does not speak to its optimality. A revolution in performance may have blinded the community to investigating whether or not unfolding computation in this standard manner is the best choice. In fact, there are shortcomings to the typical CNN pipeline:

- It conflates abstraction and scale. Early layers cannot see coarser scales, while later layers only see them. For tasks requiring fine-scale output, such as semantic segmentation, this necessitates specialized designs for reintegrating spatial information [24, 10, 3, 1].
- The fine-to-coarse processing within a standard CNN is in opposition to a near universal principle for efficient algorithm design: coarse-to-fine processing. The first layer in a standard CNN consists of many filters independently looking at tiny, almost meaningless regions of the image. Would it not be more reasonable for the system to observe some coarse-scale context before deciding how to probe the details?
- Communication is inefficient. A neuron’s receptive field is determined by the units in the input layer that could propagate a signal to it. Standard CNNs implement a slow propagation scheme, diffusing information across a single grid at rate proportional to convolutional filter size. This may be a reason extremely deep networks [34, 11, 22] appear necessary; many layers are needed to counteract inefficient signal propagation.

These points can be summarized as inherent deficiencies in representation, computation, and communication. Our multigrid architecture (Figure 1, bottom) endows CNNs with additional structural capacity in order to dissolve these deficiencies. It is explicitly multiscale, pushing choices about scale-space representation into the training process.

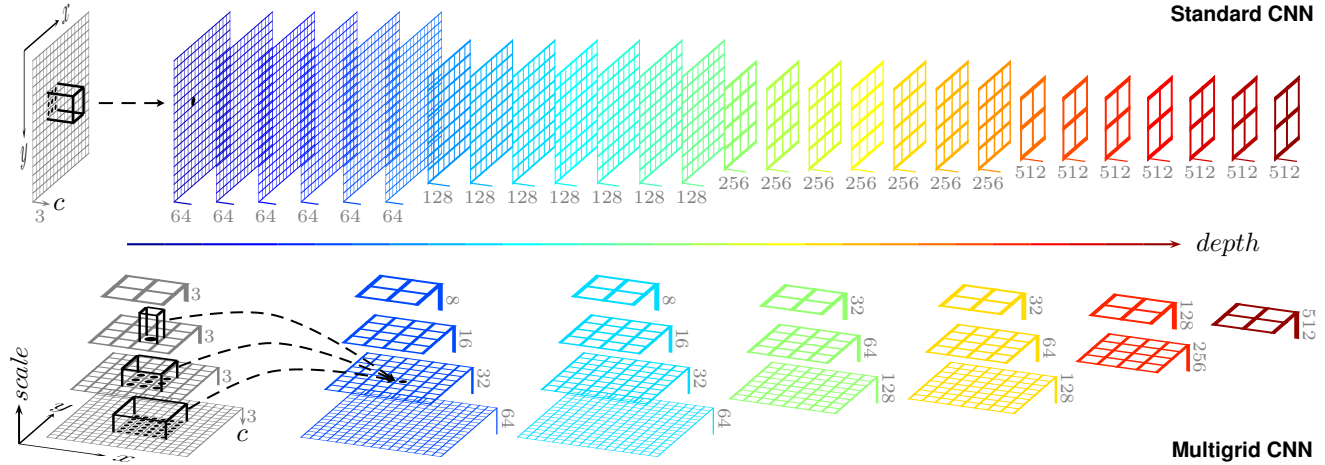


Figure 1. **Multigrid networks.** *Top:* Standard CNN architectures conflate scale with abstraction (depth). Filters with limited receptive field propagate information slowly across the spatial grid, necessitating the use of very deep networks to fully integrate contextual cues. *Bottom:* In multigrid networks, convolutional filters act across scale space (x, y, c, s) , thereby providing a communication mechanism between coarse and fine grids. This reduces the required depth for mixing distant contextual cues to be logarithmic in spatial separation. Additionally, the network is free to disentangle scale and depth: every layer may learn several scale-specific filter sets, choosing what to represent on each pyramid level. Traditional pooling and subsampling are now similarly multigrid, reducing the size of an entire pyramid.

Computation occurs in parallel at all scales; every layer process both coarse and fine representations. Section 3.2 also explores coarse-to-fine variants that transition from processing on a coarse pyramid to processing on a full pyramid as the network deepens. Pyramids provide not only an efficient computational model, but a unified one. Viewing the network as evolving a representation living on the pyramid, we can combine previous task-specific architectures. For classification, attach an output to the coarsest pyramid level; for segmentation, attach an output to the finest.

Multigrid structure facilitates cross-scale information exchange, thereby destroying the long-established notion of receptive field. Most neurons have receptive field equivalent to the entire input; field size grows exponentially with depth, or, in progressive multigrid networks, begins with the full (coarse) input. Quick communication pathways exist throughout the network, and enable new capabilities.

We specifically demonstrate that multigrid CNNs, trained in a pure end-to-end fashion, can learn to attend and route information. Their emergent behavior may dynamically emulate the routing circuits articulated by Olshausen *et al.* [28]. We construct a synthetic task that standard CNNs completely fail to learn, but multigrid CNNs accomplish with ease. Here, attentional capacity is key.

As Section 2 reviews, recent CNN architectural innovations ignore scale-space routing capacity, focusing instead on aspects like depth. Multigrid, as Section 3 details, complements such work. Section 4 measures performance improvements due to multigrid on classification tasks (CIFAR and ImageNet) and synthetic semantic segmentation tasks.

Multigrid boosts both baseline and residual CNNs. On a synthetic spatial transformation task, multigrid is more than a boost; it is required, as residual networks alone do not possess attentional capacity. Section 5 discusses implications.

2. Related Work

In wake of AlexNet [21], exploration of CNNs across computer vision has distilled some rules of thumb for their design. Small (*e.g.* 3×3) spatial filters, in many successive layers, make for efficient parameter allocation [32, 34, 11]. Feature channels should increase with spatial resolution reduction [21, 32] (*e.g.* doubling as in Figure 1). Deeper networks are better, so long as a means of overcoming vanishing gradients is engineered into the training process [34, 16] or the network itself [33, 11, 22, 15]. Width matters [38].

The desire to adapt image classification CNNs to more complex output tasks, such as semantic segmentation, has catalyzed development of ad-hoc architectural additions for restoring spatial resolution. These include skip connections and upsampling [24, 3], hypercolumns [10], and, autoencoder-like, hourglass or U-shaped networks that reduce and re-expand spatial grids [1, 13, 27, 30].

This latter group of methods reflects the classic intuition of connecting bottom-up and top-down signals. Our work differs from these and earlier models [2] by virtual of decoupling pyramid level from feature abstraction. Representations at all scales evolve over the depth of our network. Such dynamics also separates us from past multiscale CNN work [8, 7, 35], which does not consider embedded and continual cross-scale communication.

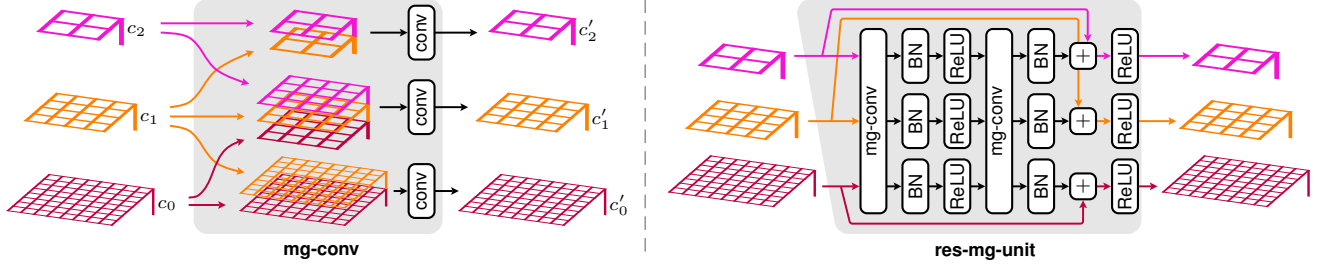


Figure 2. **Multigrid layers.** *Left:* We implement a multigrid convolutional layer (mg-conv) using readily available components. For each grid in the input pyramid, we rescale its neighboring grids to the same spatial resolution and concatenate their feature channels. Convolution over the resulting single grid approximates scale-space convolution on the original pyramid. Downscaling is max-pooling, while upscaling is nearest-neighbor interpolation. *Right:* A building block (res-mg-unit) for residual connectivity [11] in multigrid networks.

While prior CNN designs have varied filter size (e.g. [21, 34]), our choice to instead vary grid resolution is crucial. Applying multiple filter sets, of different spatial size, to a single grid could emulate a multigrid computation. However, we use exponential (power of 2) grid size scaling. Emulating this via larger filters quickly becomes cost prohibitive in terms of both parameters and computation; it is impractical to implement in the style of Inception modules [34]. Dilated [37] and atrous convolution [3], while related, do not fully capture the pooling and interpolation aspects (detailed in Section 3) of our multigrid operator.

Recent efforts to improve the computational efficiency of CNNs, though too numerous to list in full, often attack via parameter [36, 4, 17] or precision [29, 5] reduction. Concurrent work extends such exploration to include cascades [14, 26], in which explicit routing decisions allow for partial evaluation of a network. Unlike this work, our cross-scale connections are bi-directional and we explore routing in a different sense: internal transport of information (e.g. attention [28]) for solving a particular task. We focus on a coarse-to-fine aspect of efficiency, borrowing from the use of multigrid concepts in image segmentation [31, 25].

3. Multigrid Architectures

Figure 1 conveys our intention to wire cross-scale connections into network structure at the lowest level. We can think of a multigrid CNN as a standard CNN in which every grid is transformed into a pyramid. Every convolutional filter extends spatially within grids (x, y) , across grids of multiple scales (s) within a pyramid, and over corresponding feature channels (c) . A pyramidal slice, across scale-space, of the preceding layer contributes to the response at a particular corresponding neuron in the next.

This scheme enables any neuron to transmit a signal up the pyramid, from fine to coarse grids, and back down again. Even with signals jumping just one pyramid level per consecutive layer, the network can exploit this structure to quickly route information between two spatial locations on the finest grid. Communication time is logarithmic in

spatial separation, rather than linear as in standard CNNs.

This difference in information routing capability is particularly dramatic in light of the recent trend towards stacking many layers of small 3×3 convolutional filters [32, 11]. In standard CNNs, this virtually guarantees that either very deep networks, or manually-added pooling and unpooling stages [1, 13, 27, 30], will be needed to propagate information across the pixel grid. Multigrid allows for faster propagation with minimal additional design complexity. Moreover, unlike fixed pooling/unpooling stages, multigrid allows the network to learn data-dependent routing strategies.

Instead of directly implementing multigrid convolution as depicted in Figure 1, we implement a close alternative that can be easily built out of standard components in existing deep learning frameworks. Figure 2 illustrates the multigrid convolutional layer (mg-conv) we use as a drop-in replacement for standard convolutional (conv) layers when converting a network from grids to pyramids.

With multigrid convolution, we may choose to learn independent filter sets for each scale within a layer, or alternatively, tie parameters between scales and learn shared filters. We learn independent sets, assuming that doing so affords the network the chance to squeeze maximum performance out of scale-specific representations. Channel counts between grids need not match in any particular manner. Each of $c_0, c_1, c_2, c'_0, c'_1, c'_2$ in Figure 2 is independent. For comparisons, we either set channel count on the finest grid to match baseline models, or calibrate channel counts so that total parameters are similar to the baselines. Section 4 reveals quite good performance is achievable by halving channel count with each coarser grid (e.g. $c_1 = \frac{1}{2}c_0$, $c_2 = \frac{1}{4}c_0$). Thus, multigrid adds minimal computational overhead; the coarse grids are cheap.

Given recent widespread use of residual networks [11], we consider a multigrid extension of them. The right side of Figure 2 shows a multigrid analogue of the original residual unit [11]. Convolution acts jointly across the multigrid pyramid, while batch normalization (BN) [18] and ReLU apply separately to each grid. Extensions are also possible

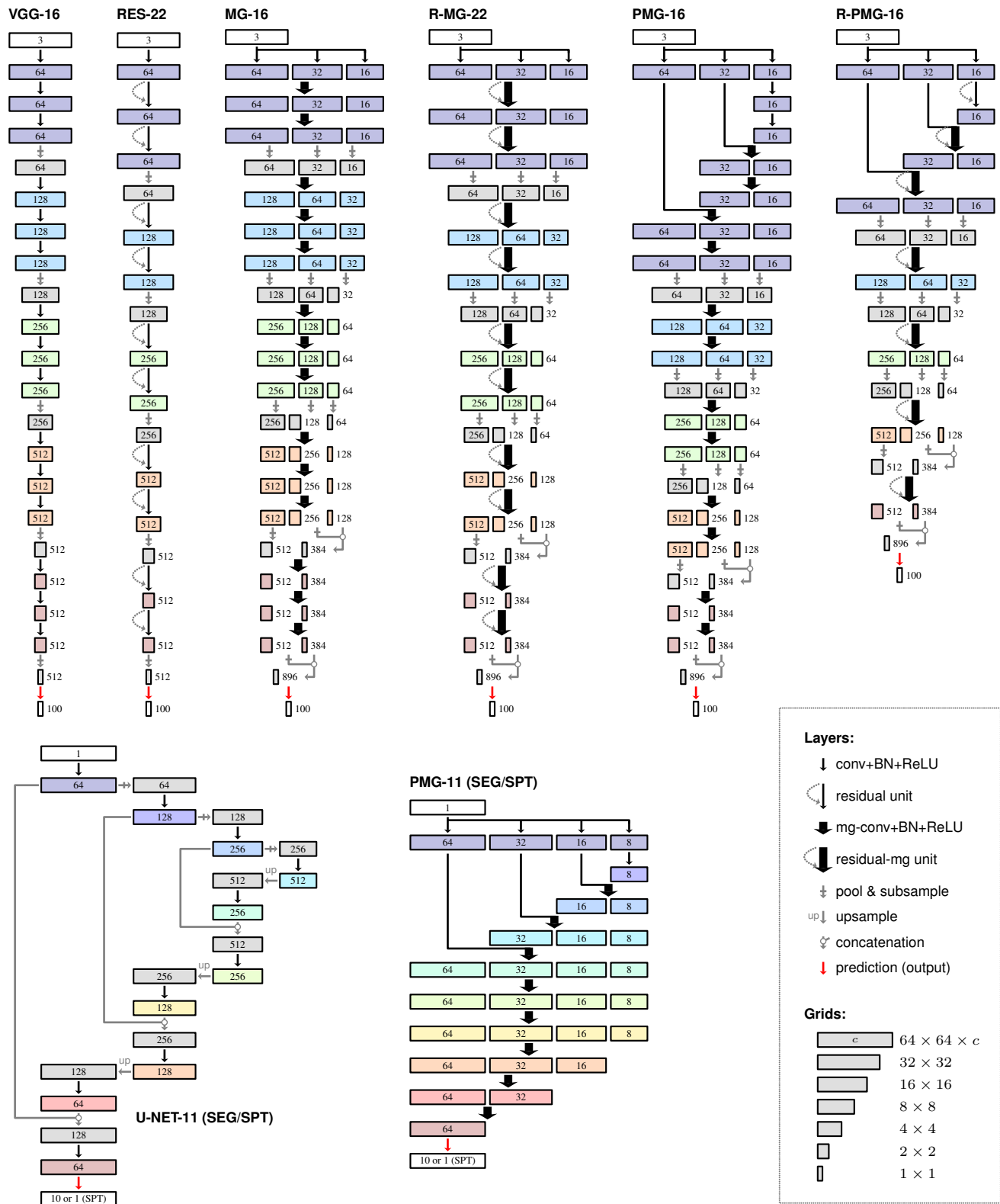


Figure 3. **Network diagrams.** *Top:* Baseline (VGG) and multigrid (MG) CNNs for CIFAR-100 classification, as well as residual (RES, R-) and progressive (PMG) versions. We vary depth by duplicating layers within each colored block. *Bottom:* Semantic segmentation (SEG) and spatial transformation (SPT) architectures: U-NET [30], and our progressive multigrid alternative (PMG). We also consider U-MG (not shown), a baseline that replaces U-NET grids with pyramids (as in VGG \rightarrow MG), yet still shrinks and re-expands pyramids.

with alternative residual units [12], but we leave such exploration to future work. Our focus is on comparison of baseline, residual, multigrid, and residual-multigrid networks.

3.1. Pooling

Within an mg-conv layer, max-pooling and subsampling acts as a lateral communication mechanism from fine to coarse grids. Similarly, upsampling facilitates lateral communication from coarse to fine grids. Rather than locating these operations at a few fixed stages in the pipeline, we are actually inserting them everywhere. However, their action is now lateral, combining different-scale grids in the same layer, rather than between grids of different scales in two layers consecutive in depth.

While Figure 1 shows additional max-pooling and subsampling stages acting depth-wise on entire pyramids, we also consider pipelines that do not shrink pyramids. Rather, they simply attach an output to a particular grid (and possibly prune a few grids outside of its communication range). Success of this strategy motivates rethinking the role of pooling in CNNs. Instead of explicit summarization employed at select stages, multigrid yields a view of pooling as implicit communication that pervades the network.

3.2. Progressive Multigrid

The computation pattern underlying a multigrid CNN’s forward pass has analogy with the multiscale multigrid scheme of Maire and Yu [25]. One can view their eigensolver as a linear diffusion process on a multiscale pyramid. We view a multigrid CNN as a nonlinear process, on a similar pyramid, with the same communication structure.

Extending the analogy, we port their progressive multigrid computation scheme to our setting. Rather than starting directly on the full pyramid, a progressive multigrid CNN can spend several layers processing only a coarse grid. Following this are additional layers processing a small pyramid (e.g. coarse and medium grids together), before the remainder of the network commits to work with the full pyramid.

In all multigrid and progressive multigrid experiments, we set the fine-scale input grid to be the original image and simply feed in downsampled (lower resolution) versions to independent initial convolution layers. The outputs of these initial layers then form a multigrid pyramid which is processed coherently by the rest of the network.

3.3. Model Zoo

Figure 3 diagrams the variety of network architectures we evaluate. For classification, we take a network with minor differences from that of Simonyan and Zisserman [32] as our baseline. In abuse of notation, we reuse their VGG name. Our 16-layer version, VGG-16, consists of 5 sections of 3 convolutional layers each, with pooling and subsampling between. A final softmax layer produces class predic-

tions (on CIFAR-100). Convolutional layers use 3×3 filters. We instantiate variants with different depth by changing the number layers per section (2 for VGG-11, 4 for VGG-21, etc.). Residual baselines (RES) follow the same layout, but use residual units within each section. Recall that each residual unit contains two convolutional layers.

Multigrid (MG) and residual multigrid (R-MG) networks, starting respectively from VGG or RES, simply flip from grids to pyramids and convolution to multigrid convolution. Progressive variants (PMG, R-PMG) expand the first section in order to gradually work up to computation on the complete pyramid. Even as their depth increases, a significant portion of layers in progressive networks avoid processing the full pyramid. As diagrammed, multigrid networks match their fine-grid feature count with baselines, and hence add some capacity and parameters. For classification, we also consider smaller (denoted -sm) multigrid variants with fewer channels per grid in order to be closer in parameter count with baseline VGG and RES networks.

For the semantic segmentation and spatial transformation tasks detailed in the next section, we use networks that produce per-pixel output. As a baseline, we employ the U-NET design of Ronneberger *et al.* [30]. Again, convolutional filters are 3×3 , except for the layers immediately following upsampling; here we use 2×2 filters, following [30]. We examine progressive multigrid alternatives (PMG, R-PMG) that continue to operate on the multiscale pyramid. Unlike the classification setting, we do not reduce pyramid resolution. These networks drop some coarse grids towards the end of their pipelines for the sole reason that such grids do not communicate with the output.

4. Experiments

Our experiments focus on systematic exploration and evaluation of the architectural design space, with the goal of quantifying the relative benefits of multigrid and its synergistic combination with residual connections.

4.1. Classification: CIFAR-100 [20]

We evaluate the array of network architectures listed in Table 1 on the task of CIFAR-100 image classification.

Data. We whiten and then enlarge images to 36×36 . We use random 32×32 patches from the training set (with random horizontal flipping) and the center 32×32 patch from test examples. There are 50K training and 10K test images.

Training. We use SGD with batch size 128, weight decay rate 0.0005, and 300 iterations per epoch for 200 epochs. For VGG, MG, and PMG, learning rate exponentially decays from 0.1 to 0.0001; whereas for RES, R-MG, and R-PMG, it decays at rate 0.2 every 60 epochs.

Results. From Table 1, we see that adding multigrid capac-

Method	Params ($\times 10^6$)	FLOPs ($\times 10^6$)	Error (%)
VGG-6	3.96	67.88	32.15
VGG-11	9.46	228.31	30.26
VGG-16	14.96	388.75	30.47
VGG-21	20.46	549.18	31.89
MG-sm-6	3.28	46.10	33.83
MG-sm-11	8.02	154.60	31.55
MG-sm-16	12.75	263.10	32.59
MG-sm-21	17.49	371.60	34.21
MG-6	8.34	116.63	32.08
MG-11	20.46	391.88	28.39
MG-16	32.58	667.13	29.91
MG-21	44.69	942.38	30.03
PMG-sm-9	3.33	73.23	32.38
PMG-sm-16	8.07	183.34	30.47
PMG-sm-30	12.82	293.46	31.06
PMG-9	8.46	186.21	30.61
PMG-16	20.60	468.09	28.11
PMG-30	32.74	749.98	29.89
RES-12	9.50	266.02	28.64
RES-22	20.49	586.93	28.05
RES-32	31.49	907.79	27.12
RES-42	42.48	1228.65	27.60
R-MG-sm-12	8.06	180.12	30.24
R-MG-sm-22	17.52	397.11	28.65
R-MG-sm-32	26.99	614.11	29.27
R-MG-sm-42	36.46	831.11	26.85
R-MG-12	20.56	457.20	27.84
R-MG-22	44.79	1007.70	26.79
R-MG-32	69.02	1558.20	25.29
R-MG-42	93.26	2108.71	26.32
R-PMG-sm-16	8.07	183.34	29.41
R-PMG-sm-30	17.56	403.57	27.68
R-PMG-sm-44	27.05	623.79	27.00
R-PMG-16	20.60	468.09	27.35
R-PMG-30	44.88	1031.87	26.44
R-PMG-44	69.16	1595.64	26.68

Table 1. CIFAR-100 classification performance.

ity consistently improves over both basic (compare green entries), as well as residual CNNs (blue entries). Progressive multigrid nets of similar depth achieve better or comparable accuracy at reduced expense, as quantified in terms of both parameters and floating point operations (red entries).

4.2. Semantic Segmentation

We generate synthetic data (10K training, 1K test images) for semantic segmentation by randomly scaling, rotating, translating, and pasting MNIST [23] digits onto a 64×64 canvas, limiting digit overlap to 30%. The task is to decompose the result into per-pixel digit class labels.

In addition to networks already mentioned, we consider a single grid (SG) baseline that mirrors PMG but removes all grids but the finest. We also consider U-MG, which changes resolution like U-NET, but works on pyramids.

Training. We use batch size 64, weight decay rate 0.0005, and 150 iterations per epoch for 200 epochs. We set learning rate schedules for non-residual and residual networks in the same manner as done for CIFAR-100.

Results. Table 2 and Figure 4 show dramatic performance

Method	Params ($\times 10^6$)	mean IoU (%)	mean Error (%)
U-NET-11	3.79	58.46	22.14
U-MG-11	5.90	58.02	21.45
SG-11	0.23	32.21	22.33
R-SG-20	0.45	69.86	14.43
PMG-11	0.61	61.88	14.75
R-PMG-20	1.20	81.91	8.89

Table 2. MNIST semantic segmentation performance.

Method	Params ($\times 10^6$)	mean IoU (%)	mean Error (%)
U-NET-11	3.79	12.20	37.55
SG-11	0.23	0	100.00
R-SG-20	0.45	0	82.98
PMG-11	0.61	50.64	29.20
R-PMG-20	1.20	55.61	25.59

Table 3. MNIST spatial transformation performance.

Method	Params ($\times 10^6$)	FLOPs ($\times 10^9$)	val, 10-crop		val, single-crop	
			Top-1	Top-5	Top-1	Top-5
VGG-16 [32]	138.0	15.47	28.07	9.33	-	-
ResNet-34 C [11]	21.8	3.66	24.19	7.40	-	-
ResNet-50 [11]	25.6	4.46	22.85	6.71	-	-
WRN-34 (2.0) [38]	48.6	14.09	-	-	24.50	7.58
R-MG-34	32.9	5.76	22.42	6.12	24.51	7.46
R-PMG-30-SEG	31.9	2.77	23.60	6.89	26.50	8.63

Table 4. ImageNet classification error (%) on validation set.

advantages of the progressive multigrid network. We report scores of mean intersection over union (IoU) with the true output regions, as well as mean per-pixel multiclass labeling accuracy. These results are a strong indicator that continuous operation on a multigrid pyramid should replace networks that pass through a low resolution bottleneck. See the supplementary material for more examples like Figure 4.

4.3. Spatial Transformers

Jaderberg *et al.* [19] engineer a system for inverting spatial distortions by combining a neural network for estimating the parameters of the transformation with a dedicated unit for applying the inverse. Such a split of the problem into two components appears necessary when using standard CNNs; they cannot learn the task end-to-end. But, progressive multigrid networks are able to learn such tasks.

We generate a synthetic spatial transformation dataset (60K training, 10K test images) in the same manner as the one for semantic segmentation, but with only one digit per image and an additional affine transformation with a uniformly random sheering angle in $(-60^\circ, 60^\circ)$. Training is the same as segmentation, except 800 iterations per epoch.

Results. Table 3 and Figure 5 show that all networks except PMG and R-PMG fail to learn the task. Figure 5 (right side) reveals the reason: U-NET (and others not shown) cannot learn translation. This makes sense for single grid methods, as propagating information across the fine grid would require them to be deeper than tested. U-NET’s failure seems to stem from confusion due to pooling/upsampling. It appears to paste subregions of the target digit into output, but

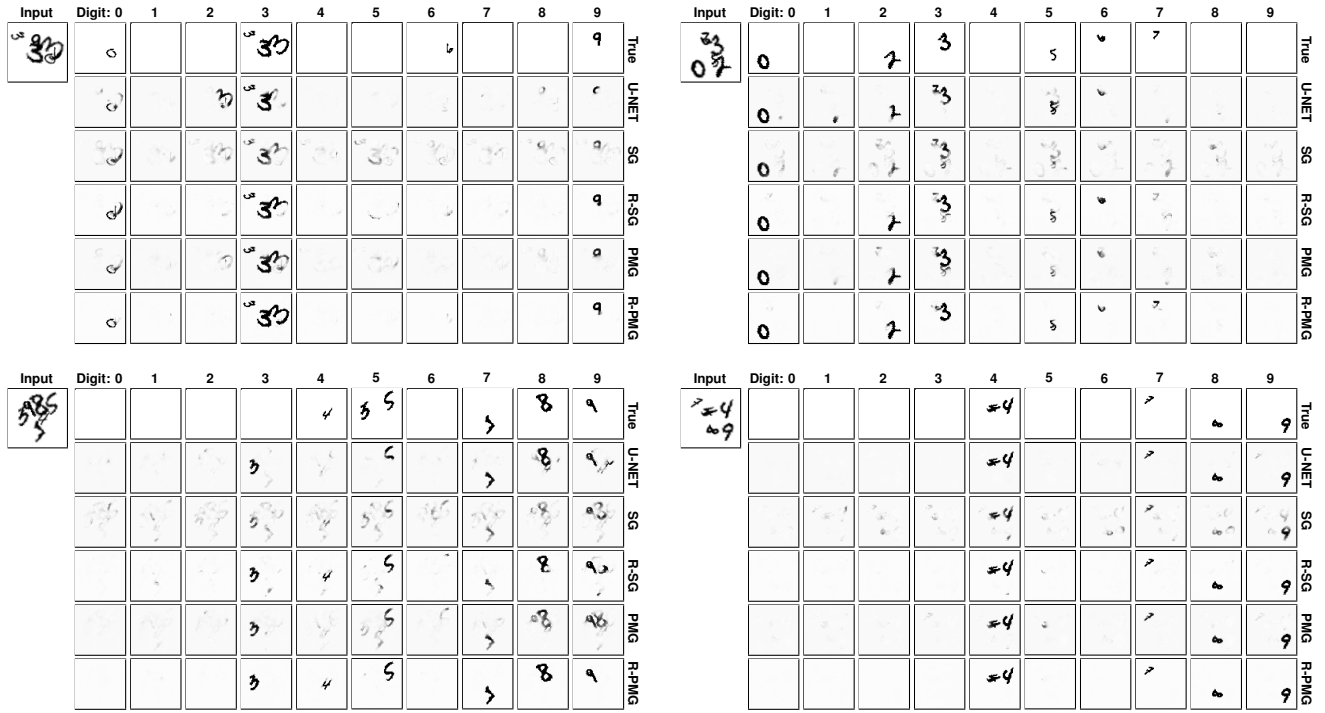


Figure 4. **Semantic segmentation on synthetic MNIST.** We compare U-NET, residual (R-), and progressive multigrid (PMG) networks on a synthetic task of disentangling randomly distorted and overlapped MNIST digits. Figure 3 specifies the 11-layer U-NET and PMG variants used here. An 11-layer single-grid (SG) baseline operates on only the finest scale. Analogous residual versions (R-SG, R-PMG) are 20 layers deep. PMG thus performs similarly to a deeper single-grid residual network (R-SG), while R-PMG bests all.

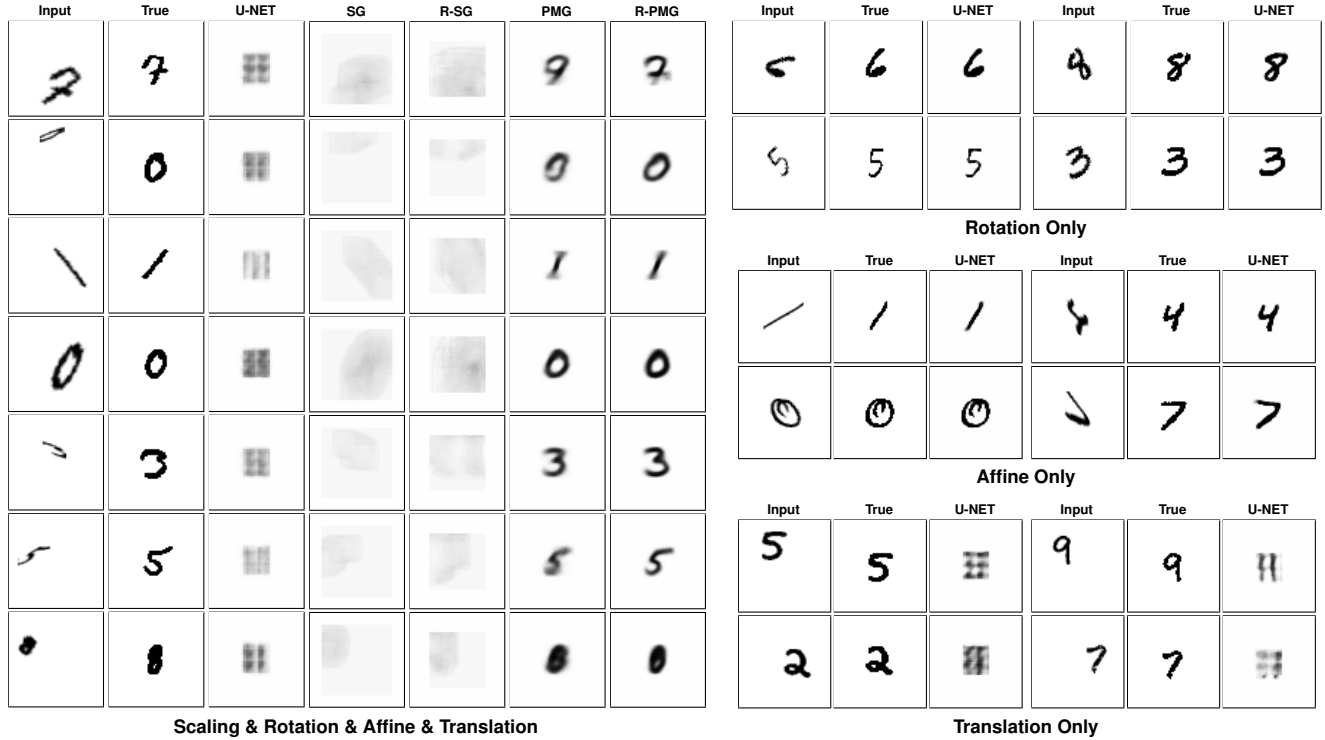


Figure 5. **Spatial transformations on synthetic MNIST.** *Left:* Only multigrid CNNs (PMG, R-PMG) are able to learn to take a distorted and translated digit as input and produce the centered original. *Right:* U-NET can learn to undo rotation and affine transformations, but not translation. U-NET’s failure stems from inability to properly attend and route information from a sub-region of the input.

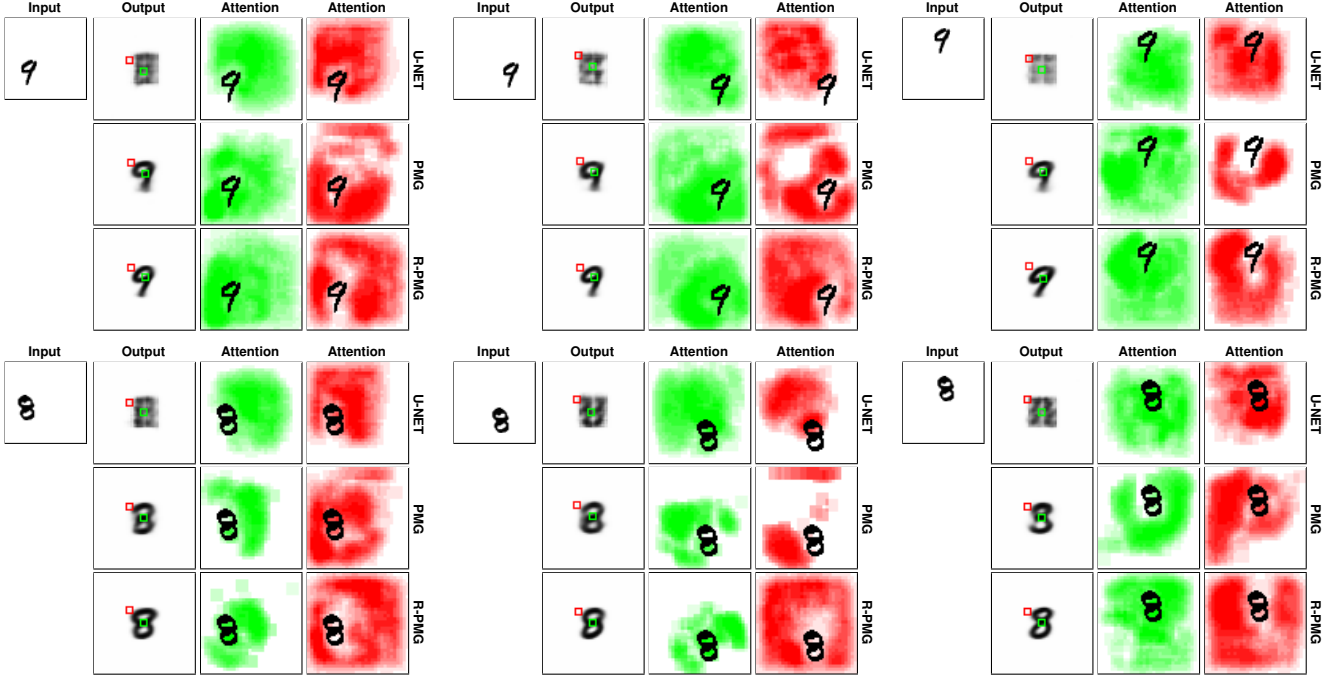


Figure 6. **Attention maps for MNIST spatial transformers.** To produce attention maps, we sweep an occluder (8×8 square of uniform random noise) over the 64×64 input image and measure sensitivity of the output at the red and green locations. U-NET, which fails to learn translation, attends to approximately the same regions regardless of input. PMG and R-PMG instead exhibit attention that translates with the input. Please see the supplementary material for a more detailed visualization of our attention map generation process.

not in the correct arrangement. Figure 6 reveals that, unlike U-NET, PMG and R-PMG exhibit attentional behavior.

4.4. Classification: ImageNet [6]

We select two designs to train from scratch on ImageNet:

- R-MG-34: A multigrid analogue of ResNet-34 [11], with finest grid matching ResNet, and two additional grids at half and quarter spatial resolution (and channel count). Pooling and resolution reduction mirror ResNet-34 and the R-MG diagram in Figure 3.
- R-PMG-30-SEG: Similar to our semantic segmentation network, but adapted to ImageNet, it maps 224×224 input onto 4 grids: 56×56 , 28×28 , 14×14 , 7×7 , that are maintained throughout the network without pooling stages. We increase feature count with depth and attach a classifier to the final coarsest grid.

Training uses SGD with batch size 256, weight decay 0.0001, 10K iterations per epoch for 100 epochs, starting learning rate 0.1 with decay rate 0.1 every 30 epochs. Table 4 compares our performance to that of ResNets and wide residual networks (WRN) [38]. We observe:

- Multigrid substantially improves performance. R-MG-34 even outperforms the deeper ResNet-50.
- Multigrid is a more efficient use of parameters than simply increasing feature channel count on a single

grid. Compare R-MG-34 with the double-wide residual network WRN-34: we match its performance with fewer parameters and require drastically fewer FLOPs.

- R-PMG-30-SEG outperforms ResNet-34. The same design we used for segmentation is great for classification, lending support to the idea of a single unifying architecture which evolves a multigrid representation.

5. Conclusion

Our proposed multigrid extension to CNNs yields improved accuracy on classification and semantic segmentation tasks. Progressive multigrid variants open a new pathway towards optimizing CNNs for efficiency. Multigrid appears unique in extending the range of tasks CNNs can accomplish, by integrating into the network structure the capacity to learn routing and attentional mechanisms. These new abilities suggest that multigrid could replace some ad-hoc designs in the current zoo of CNN architectures.

On a speculative note, multigrid neural networks might also have broader implications in neuroscience. Feedforward computation on a sequence of multigrid pyramids looks similar to combined bottom-up/top-down processing across a single larger structure if neurons are embedded in some computational substrate according to their spatial grid, rather than their depth in the processing chain.

References

- [1] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv:1511.00561*, 2015.
- [2] S. Behnke and R. Rojas. Neural abstraction pyramid: A hierarchical image understanding architecture. *International Joint Conference on Neural Networks*, 1998.
- [3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *arXiv:1606.00915*, 2016.
- [4] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. *arXiv:1504.04788*, 2015.
- [5] M. Courbariaux, Y. Bengio, and J.-P. David. BinaryConnect: Training deep neural networks with binary weights during propagations. *NIPS*, 2015.
- [6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. *CVPR*, 2009.
- [7] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. *NIPS*, 2014.
- [8] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *PAMI*, 2013.
- [9] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 1980.
- [10] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. *CVPR*, 2015.
- [11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016.
- [12] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *ECCV*, 2016.
- [13] P. Hu and D. Ramanan. Bottom-up and top-down reasoning with hierarchical rectified gaussians. *CVPR*, 2016.
- [14] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger. Multi-scale dense convolutional networks for efficient prediction. *arXiv:1703.09844*, 2017.
- [15] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *arXiv:1608.06993*, 2016.
- [16] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger. Deep networks with stochastic depth. *ECCV*, 2016.
- [17] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *arXiv:1602.07360*, 2016.
- [18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015.
- [19] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. *NIPS*, 2015.
- [20] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.
- [21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *NIPS*, 2012.
- [22] G. Larsson, M. Maire, and G. Shakhnarovich. FractalNet: Ultra-deep neural networks without residuals. *ICLR*, 2017.
- [23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.
- [24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
- [25] M. Maire and S. X. Yu. Progressive multigrid eigensolvers for multiscale spectral segmentation. *ICCV*, 2013.
- [26] M. McGill and P. Perona. Deciding how to decide: Dynamic routing in artificial neural networks. *arXiv:1703.06217*, 2017.
- [27] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. *ECCV*, 2016.
- [28] B. A. Olshausen, C. H. Anderson, and D. C. V. Essen. A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *Journal of Neuroscience*, 1993.
- [29] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. XNOR-Net: Imagenet classification using binary convolutional neural networks. *ECCV*, 2016.
- [30] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. *arXiv:1505.04597*, 2015.
- [31] E. Sharon, M. Galun, D. Sharon, R. Basri, and A. Brandt. Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442:810–813, 2006.
- [32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.
- [33] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *ICML*, 2015.
- [34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CVPR*, 2015.
- [35] S. Xie and Z. Tu. Holistically-nested edge detection. *ICCV*, 2015.
- [36] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. J. Smola, L. Song, and Z. Wang. Deep fried convnets. *ICCV*, 2015.
- [37] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *ICLR*, 2016.
- [38] S. Zagoruyko and N. Komodakis. Wide residual networks. *Bmvc*, 2016.