The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

# Learning Across Scales — Multiscale Methods for Convolution Neural Networks

**Eldad Haber,**[1,2] **Lars Ruthotto,**[2,3] **Elliot Holtham,**[2] **Seong-Hwan Jun**[4]
[1] Dept. of Earth and Ocean Science, University of British Columbia, Vancouver, Canada eldadhaber@gmail.com
[2] Xtract Technologies, Vancouver, BC, Canada, elliot@xtract.tech
[3] Dept. of Mathematics and Computer Science, Emory University, Atlanta, GA, USA, lruthotto@emory.edu
[4] Dept. of Statistics, University of British Columbia, Vancouver, Canada, seong.jun@stat.ubc.ca

**Abstract**

In this work, we establish the relation between optimal control and training deep Convolution Neural Networks (CNNs). We show that the forward propagation in CNNs can be interpreted as a time-dependent nonlinear differential equation and learning can be seen as controlling the parameters of the differential equation such that the network approximates the data-label relation for given training data. Using this continuous interpretation, we derive two new methods to scale CNNs with respect to two different dimensions. The first class of multiscale methods connects low-resolution and high-resolution data using prolongation and restriction of CNN parameters inspired by algebraic multigrid techniques. We demonstrate that our method enables classifying high-resolution images using CNNs trained with low-resolution images and vice versa and warm-starting the learning process. The second class of multiscale methods connects shallow and deep networks and leads to new training strategies that gradually increase the depths of the CNN while re-using parameters for initializations.

## 1 Introduction

We consider the problem of designing and training Convolution Neural Networks (CNNs). CNNs have been a major field of research over the last years, after showing remarkable success, e.g., in classifying images of hand writing, natural images, videos (see, e.g., LeCun and Bengio; Krizhevsky, Sutskever, and Hinton; LeCun, Kavukcuoglu, and Farabet (1995; 2012; 2010) and references therein). This success has resulted in thousands of research papers and a few celebrated software packages.

Despite their success, CNNs are not fully understood, and in fact, tuning network architecture and parameters can be very hard in practice. Often many trial-and-error experiments are required to find a CNN that is effective for a specific class of data. In addition to the computational costs associated with those experiments, in many cases, small changes to the network can yield substantial changes in the learning performance. To overcome the difficulty of learning, systematic approaches using Bayesian optimization have been proposed to infer the best architecture (see Hernández-Lobato et al. (2016)).



Crucial design parameters in CNN are the network architecture and, currently, the resolution of the image data. Changing any of those parameters in the training or prediction phase can severely affect the performance of the CNN. For example, CNNs are typically trained using images of a fixed resolution, and classifying images of a different resolution requires interpolation. This can be computationally expensive, particularly on resource-limited systems or if the data represents videos or high-resolution 3D images as is common in applications, e.g., in medical imaging and geosciences (see Jiang, Trundle, and Ren; Karpathy et al. (2010; 2014)).

In this paper, we derive a framework that allows scaling CNNs across image resolution and network depth and thus enables multiscale learning. As a backbone of our methods, we present an interpretation of deep CNNs as an optimal control problem involving a nonlinear time-dependent differential equation. This understanding leads structure that is very common in fields such as path planning, data assimilation, and nonlinear Kalman filtering (see, e.g., Biegler et al. (2009) and reference therein).

We present new methods for scaling CNNs from low- to high-resolution image data and vice versa. We propose an algebraic multigrid approach to adapt the coefficients of the convolution kernel across scales and demonstrate the importance of this step. Our method allows multiscale learning using image pyramids, where the network is trained at different resolutions. Such a process is known to be very effective in other fields, giving rise to computational savings and adding robustness to local minima (see, e.g., Modersitzki; Haber and Modersitzki; ans A. Ratcliffe et al. (2004; 2004; 2013)). Our new method also enables one to classify low-resolution images using networks that have been designed for and trained using high-resolution images *without* interpolation of the coarse scale image to finer scales.

We also present a method for scaling the number of layers in a deep CNN. Our method uses the interpretation of the forward propagation in CNN as a discretization of a time-dependent nonlinear differential equation. In our framework, the number of layers corresponds to the number time steps. This observation motivates the use of multilevel learning algorithms that accelerate the training of deep networks by solving a series of learning problems from shallow to deep architectures.

The rest of this paper is structured as follows. In the next section, we show the connection between optimal control of time-dependent differential equations and training CNNs. This connection allows us to introduce the continuous problem at the heart of our approach. In Sec. 3 we present multiscale methods connecting CNNs across image resolutions and depths. In Sec. 4 we demonstrate the potential of our methods using image classification benchmarks. Finally, in Sec. 5 we summarize the paper.

## 2 CNNs Meet Optimal Control

We demonstrate the similarity of training deep convolution neural networks and optimal control of a time-dependent nonlinear partial differential equation. First, we derive a continuous interpretation of the spatial convolution and the forward propagation. Second, we discuss the remaining components of CNNs and present the continuous optimal control problem and its discretization.

For brevity, we focus on image classification and assume we are given training data consisting of discrete $d$-dimensional images $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)} \in \mathbb{R}^n$ and corresponding labels $\mathbf{c}^{(1)}, \mathbf{c}^{(2)}, \ldots, \mathbf{c}^{(m)} \in \mathbb{R}^\ell$. As common in image processing, we interpret the image data as a discretization of a continuous function $x : \Omega \to \mathbb{R}$ at the cell-centers of a rectangular grid on the domain $\Omega \subset \mathbb{R}^d$ with $n$ equally sized pixels of edge length $h$.In this paper, we consider $d = 2$.

**Continuous Forward Propagation Model.** We show that the forward propagation in Residual Neural Networks (ResNN) proposed in (He et al. 2016b) can be seen as a nonlinear differential equation. A simple way to write the forward propagation of a discrete image $\mathbf{x} \in \mathbb{R}^n$ through a ResNet is

$$\begin{aligned} \mathbf{y}_{k+1} &= \mathbf{y}_k + \delta t F(\mathbf{y}_k, \boldsymbol{\theta}_k), \\ \mathbf{y}_0 &= \mathbf{L}\mathbf{x}, \quad \forall k = 0, 1, \ldots, N. \end{aligned} \tag{1}$$

Here, $N$ is the number of layers in the CNN, $\mathbf{y}_0 \in \mathbb{R}^{n_f}$ are the input features, $\mathbf{y}_1, \ldots, \mathbf{y}_N$ are the hidden layers, and $\mathbf{y}_{N+1}$ are the output layers. The matrix $\mathbf{L}$ maps the input image into the feature space $\mathbb{R}^{n_f}$. This matrix can be "learned" or fixed. The parameters $\boldsymbol{\theta}_k$ will be determined by the "learning" process. We generalize the original ResNet model by adding the parameter $\delta t > 0$, which helps derive the continuous interpretation below ( $\delta t = 1$ gives the original formulation).

In CNN, the function $F$ contains a convolution and $\boldsymbol{\theta}$ consists of the convolution weights and bias. This leads to the explicit expression

$$F(\mathbf{y}, \mathbf{s}, \mathbf{b}) = \sigma_\alpha \left( \mathbf{K}(\mathbf{s})\mathbf{y} + \mathbf{b} \right). \tag{2}$$

Here $\mathbf{K}(\mathbf{s})$ is a convolution matrix, which is a circulant matrix that represents the convolution and depends on the stencil or convolution kernel, $\mathbf{s} \in \mathbb{R}^{n_s}$, $\mathbf{b} \in \mathbb{R}^N$ is a bias vector and $\sigma_\alpha$ is an activation function. Next, we interpret the depth of the network in a continuous framework. We start by rewriting the forward propagation (1) as

$$\frac{\mathbf{y}_{k+1} - \mathbf{y}_k}{\delta t} = \sigma_\alpha(\mathbf{K}(\mathbf{s}_k)\mathbf{y}_k + \mathbf{b}_k). \tag{3}$$

The left hand side of the above equation is a finite difference approximation to the differential operator $\partial_t \mathbf{y}$ with step size $\delta t$. While the approximation used in the original ResNet (using $\delta t = 1$) is valid if features change sufficiently slowly, the obtained dynamical system can be chaotic if the features change quickly. Having a chaotic system as a forward problem implies that one can expect difficulties when considering the learning problem (see, e.g., discussion in Haber and Ruthotto (2017)).

To obtain a fully continuous formulation of the forward propagation, we note that the convolution weights $\mathbf{s}$ can be seen as a discretization of continuous functions $s : \Omega \to \mathbb{R}$ whose support is limited to a small region around the origin. This allows to interpret $\mathbf{K}(\mathbf{s})\mathbf{y}$ as a discretization of $s * y$. Upon taking the limit $\delta t \to 0$ in (3) we obtain the *continuous* forward propagation process

$$\dot{y}(t) = \sigma_\alpha \left( s(t) * y(t) + b(t) \right), y(0) = Lx, \tag{4}$$

for all $t \in [0, T]$, where $T$ is the final time corresponding with the output layer. General stability theory of ordinary differential equations (ODEs) applies for this process as discussed by Haber and Ruthotto (2017). The continuous interpretation also offers other computational benefits, e.g., reversible and memory free implementation (Chang et al. 2017a).

Given the continuous forward propagation in (4) we interpret (3) as a forward Euler discretization with a fixed time step size of $\delta t$. The discrete forward propagation is stable as long as the real parts of the eigenvalues of the convolution and the time steps are sufficiently small. We note that there are numerous methods for time integration, some of which provide superior stability of the forward propagation (see, e.g., Ascher (2010) for details).

**Optimal Control Formulation.** Having discussed forward propagation, we now briefly review the classification problem and summarize the continuous and discrete formulation of the learning problem.

The hypothesis or classification function, which predicts the label for each data using the values at the output layer, $\mathbf{y}_{N+1}$, can be written as

$$\mathbf{c}_{\text{pred}} = g(h^d \mathbf{W}^\top \mathbf{y}_{N+1} + \mu), \tag{5}$$

where the columns of $\mathbf{W} \in \mathbb{R}^{n_f \times \ell}$ are classification weights and $\mu \in \mathbb{R}^\ell$ are biases for the respective classes. Commonly used choices are softmax, least-squares, logistic regression, or support vector machines. We have generalized the common notation by adding the parameter $h^d$ that allows to interpret $\mathbf{W}^\top \mathbf{y}_{N+1}$ as a midpoint rule applied to the standard $L_2$ inner product $(w_j, y)_{L_2} = \int_\Omega w_j(r) y(r) dr$ for a sufficiently regular function $w_j : \Omega \to \mathbb{R}$. The $j$th column of $\mathbf{W}$ is the discretization of $w_j$ at the cell-centers of the grid. This generalization allows us to adjust the weights across image resolutions.

For training data consisting of continuous functions $x^{(1)}, \ldots, x^{(m)}$ and labels $\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(m)}$, learning corre-

sponds to solving the optimal control problem

$$\min_{w,\mu,s,b} \frac{1}{m}\sum_{j=1}^{m} S(g((w, y^{(j)}(T))_{L_2} + \mu), \mathbf{c}^{(j)}) + R(w,\mu,s,b) \quad \text{s.t.} \quad y^{(j)} \text{ satisfies (4).}$$

Here $S$ is a loss function measuring the mismatch between the predicted (e.g., cross entropy) and known label and $R$ is a regularization function that penalizes undesired features in the parameters and avoids overfitting. Typically, the problem is not solved to a high accuracy, and low accuracy solutions are sufficient. A validation set is often used to determine the stopping criteria for the optimization algorithm. For completeness we note that a discrete version of the optimal control problem is

$$\begin{aligned} \min_{\mathbf{W},\mu,\mathbf{s}_k,b_k} \quad & \frac{1}{m}\sum_{j=1}^{m} S(g(h^d \mathbf{W}^\top \mathbf{y}_{N+1}^{(j)} + \mu), \mathbf{c}^{(j)}) \\ & + R(\mathbf{W},\mu,\mathbf{s}_{1,2,\ldots,N}, b_{1,2,\ldots,N}) \qquad (6) \\ \text{s.t.} \quad & \mathbf{y}_{k+1}^{(j)} = \mathbf{y}_k^{(j)} + \delta t \sigma_a(\mathbf{K}(\mathbf{s}_k)\mathbf{y}_k^{(j)} + \mathbf{b}_k), \\ & \mathbf{y}_0^{(j)} = \mathbf{L}\mathbf{x}^{(j)}, \quad \forall j = 1,\ldots,m. \end{aligned}$$

Note that for simplicity we have ignored the pooling layer, although it can be added in general (see, e.g., Springenberg et al. (2014) for a discussion on the necessity of pooling).

## 3 Multiscale Methods

We present new methods for scaling deep CNNs along two dimensions: First, we discuss restriction and prolongation of convolution operators as a way to scale CNNs along image resolution. Second, we present a method for scaling the depth of the network, which simplifies initialization and accelerates training.

**From High-Resolution to Low-Resolution.** Assume first that we are given some image data, $\mathbf{y}_h$, on a mesh with pixel size $h$ and a stencil, $\mathbf{s}_h$ that operates on this image. Assume also that we would like to apply the fine mesh convolution to an image, $\mathbf{y}_H$, given on a coarser mesh with pixel size $H > h$. In other words, the goal is to find a stencil $\mathbf{s}_H$ for which the coarse mesh convolution is equivalent to refining the image data and applying fine mesh convolution with $\mathbf{s}_h$. This problem is well-studied in the multigrid literature (see Trottenberg, Oosterlee, and Schuller (2000) for details).

Our method for restricting the stencil follows the algebraic multigrid approach (see, e.g., Trottenberg, Oosterlee, and Schuller (2000) for details and alternative approaches using re-discretization). Assume the following connection between the fine mesh image, $\mathbf{y}_h$, and the coarse mesh image $\mathbf{y}_H$

$$\mathbf{y}_H = \mathbf{R}\mathbf{y}_h \quad \text{and} \quad \tilde{\mathbf{y}}_h = \mathbf{P}\mathbf{y}_H. \qquad (7)$$

Here, $\mathbf{P}$ is a prolongation matrix, $\mathbf{R}$ is a restriction matrix, and $\tilde{\mathbf{y}}_h$ is an interpolated coarse scale image on the fine mesh. A typical assumption is that $\mathbf{RP} = \gamma\mathbf{I}$ for some $\gamma$, which depends on the dimensionality of the problem. The interpretation is that the coarse scale image is obtained using some linear transformation from the fine scale image (e.g., by averaging). Conversely, an approximate fine scale image can be obtained from the coarse scale image by interpolation. This interpretation easily extends to 3D or spatio-temporal data, e.g., videos.

Let $\mathbf{K}_h(\mathbf{s}_h)$ be the sparse matrix that represents the convolution on the fine scale. This matrix operates on a vectorized image and is equivalent to convolving the vector $\mathbf{y}_h$ with the stencil, $\mathbf{s}_h$. The matrix is circulant and sparse with a few non-zero diagonals. Our goal is to build a coarse scale convolution, $\mathbf{K}_H$ that operates on a vector $\mathbf{y}_H$ and is consistent with the operation of $\mathbf{K}_h$ on a fine scale vector $\mathbf{y}_h$. Using the prolongation and restriction we obtain that

$$\mathbf{K}_H\mathbf{y}_H = \mathbf{R}\mathbf{K}_h\mathbf{P}\mathbf{y}_H. \qquad (8)$$

That is, given $\mathbf{y}_H$ we first prolong it to the mesh $h$, then operate on it with the matrix $\mathbf{K}_h$, which yields a vector on mesh with pixel size $h$. Finally, we restrict the result to the coarse mesh with pixel size $H$. This implies that the coarse scale convolution matrix can be expressed as $\mathbf{K}_H = \mathbf{R}\mathbf{K}_h\mathbf{P}$. This construction of the coarse mesh operator is independent of the specific choice of the interpolation/restriction operators. Furthermore, assuming that the stencil, $\mathbf{s}_H$ is constant on the coarse mesh (i.e., it is not changing on the mesh as commonly assumed in CNN), it is straightforward to evaluate it without generating the matrix $\mathbf{K}_H$, as commonly done in algebraic multigrid.

**Example 1.** *To demonstrate the concept of adapting the convolution operators across different resolutions using the following simple example illustrated in Fig. 1. We select an image from the MNIST data set (bottom left) and convolve it with the fine mesh convolution parameterized by the stencil*

$$\mathbf{s}_h = \begin{pmatrix} -0.89 & -2.03 & 4.30 \\ -2.07 & 0.00 & -2.07 \\ 4.39 & -2.03 & 1.28 \end{pmatrix}$$

*obtaining the image in the bottom right panel of Fig. 1. Now, by restricting the weights using the algebraic multigrid approach, we obtain that on a coarse mesh the weights are*

$$\mathbf{s}_H = \begin{pmatrix} -0.48 & -0.17 & 0.82 \\ -0.15 & -0.80 & 0.37 \\ 0.84 & 0.40 & 0.07 \end{pmatrix}.$$

*These weights are used to convolve the coarse scale image (top left panel of Fig. 1) resulting in the filtered image on the top right panel of Fig. 1. Looking at the weights obtained on the coarse mesh, it is evident that they are significantly different from the fine scale weights. It is also evident from the fine and coarse images that adjusting the weights on the coarse mesh is necessary to satisfy* (8).

The interpretation of images and convolution weights as continuous functions, allows us to work with different image resolutions. This has two significant consequences. First, assume that we have trained our network on some fine scale images and that we are given a coarse scale image. Rather than interpolating the image to a fine mesh (which can

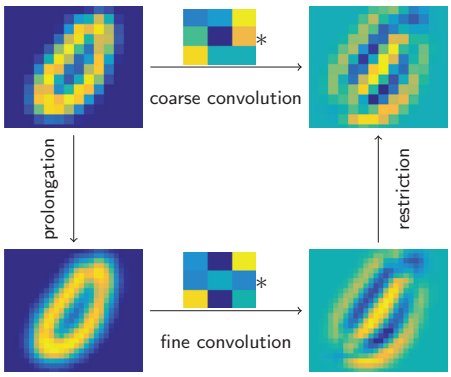


Figure 1: Fine mesh vs. coarse mesh convolution.

**Algorithm 1** Multigrid Prolongation

1: Restrict the images $n_c$ times
2: Initialize stencils, biases, and classifier
3: **for** $i = n_c : -1 : 1$ **do**
4: Solve (6) on mesh $i$ from its initial point
5: Prolong the stencils to level $i - 1$
6: Prolong the classifier weights to level $i - 1$
7: **end for**

be memory intensive and computationally expensive), *we transform the stencils to a coarse mesh and use the coarse mesh stencils to classify the coarse scale image*. Such a process can be particularly efficient when considering the classification of videos on mobile devices where expanding the video to high-resolution can be computationally prohibitive. A second consequence is that we can train the network on a coarse mesh and then interpolate the result to a fine mesh. As we see next, this allows us to use a process of image pyramid or multi-resolution for the solution of the optimization problem that is at the heart of the training process.

**From Low-Resolution to High-Resolution.** Understanding how to move between different scales allows us to construct efficient algorithms that use inexpensive coarse mesh representations of the problem to initialize the problem on finer scales. This is similar to the classical image pyramid process and multilevel methods that are used in applications that range from full waveform inversion to shape from shading (see Haber and Modersitzki (2004) and Zhang et al. (1999), resp.). The idea is to solve the optimization problem on a coarse mesh first in order to initialize fine grid parameters. The algorithm is summarized in Alg. 1.

Solving each optimization problem on coarser meshes is cheaper than solving the problem on finer meshes. In fact, when an image is coarsened by a factor of two, each convolution step is four times cheaper in 2D and eight times cheaper in 3D. Ideally, this leads to linear complexity of the problem (Trottenberg, Oosterlee, and Schuller 2000).

In order to apply such algorithms in our context, we need to address the transformation of the coarse scale operator to a fine scale one. This process is different from classical multigrid where the operator on a fine mesh is given, and a coarse scale representation is desired. As previously discussed, we use the classical multigrid result to transform a fine mesh operator to a coarse one

$$\mathbf{K}_H = \mathbf{R}\mathbf{K}_h\mathbf{P}. \tag{9}$$

In the standard multigrid implementation, one has a hold on the *fine scale* operator $\mathbf{K}_h$, and the goal is to compute the coarse scale operator $\mathbf{K}_H$. In our application, throughout the mesh continuation method, we are given the *coarse mesh* operator, and we aim at computing the fine mesh operator. In principle, there is no unique fine scale operator given a coarse scale one; however, assuming that the fine scale operator is a convolution with a fixed-sized stencil (as common in CNN), there is a unique solution. This is a classical result of Fourier analysis of multigrid methods (Trottenberg, Oosterlee, and Schuller 2000). Since (9) represents a linear connection between $\mathbf{K}_h$ and $\mathbf{K}_H$, we extract $n_K^2$ equations (where $n_K$ is the size of each convolution stencil) that connect the fine scale convolutions to the coarse scale ones. For a convolution stencil of size $3^2$ and linear prolongation/restriction, this is a $9\times9$ linear system that can be easily solved to obtain the fine scale convolution. A standard multigrid result is that this linear system is well-posed. Assuming that the coarse mesh is a $n_K \times n_K$ stencil and that the interpolation is linear, the fine mesh stencil is also a $n_K \times n_K$ stencil which is uniquely determined from the coarse mesh stencil.

**From Shallow to Deep Networks.** We now consider scaling the number of layers in the network as another way to use the continuous framework. In our case, we gradually increase the number of layers keeping the final time $T$ constant in order to accelerate learning by re-using parameters from shallow networks to initialize the learning problem for the deeper architecture. Note that the number of layers in the network corresponds to the number of discretization points in the discrete forward propagation. Similar ideas have been used in multigrid by Bornemann and Deuflhard (1996) and image processing by Modersitzki (2009).

To solve a learning task in practice, we first solve the learning problem (6) using a network with only a few layers. Subsequently, we prolongate the estimated parameters of the forward propagation to initialize the optimization problem for the next network that features, e.g., twice as many layers. To be specific, we interpolate the weights of the network parameters in time to obtain initial guesses at the newly added layers. In our experiments, we use linear interpolation. However, the method can be extended to higher-order schemes. We repeat this process until we reach the desired network depth.

Besides realizing some obvious computational savings on shallower networks, the main motivation behind our approach is to obtain good starting guesses for the next level. This is key since, while deeper architectures offer more flexibility to model complicated data-label relation, deeper networks are notoriously difficult to initialize. Also, good initialization results in faster convergence rates of second-order learning algorithms.

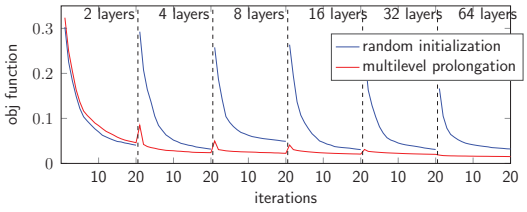


Figure 2: Multilevel results for MNIST problem.

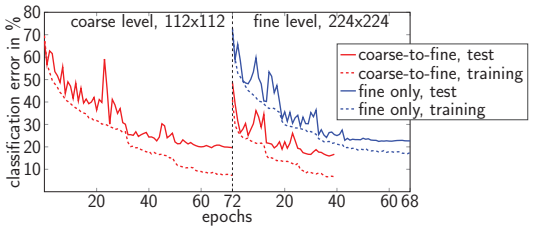


Figure 3: ImageNet-10 Results. Comparison of training and validation accuracy for fine-mesh training using $224 \times 224$ images (blue) and our coarse-to-fine approach trained also on $112 \times 112$ (red).

## 4 Experiments

**Classification of Across Resolutions.** We demonstrate that using the continuous formulation we can classify low-resolution images using CNNs trained on high-resolution images and vice versa. Here, no additional learning is performed and, e.g., classifying the low-resolution images does require neither interpolation nor high-resolution convolutions. This is important for efficient classification of high-resolution data on resource-limited devices.

We consider the MNIST dataset and independently train two networks with two layers each using the coarse and fine data, respectively. The MNIST dataset that consists of 60,000 labeled images each with $28 \times 28$ pixels. Since the images are rather coarse, we use only two levels. To obtain coarse scale images, the fine scale images are convolved with a Gaussian and restricted to a coarse mesh using the operator introduced above. This yields a coarse mesh data consisting of $14 \times 14$ images. We randomly divide the datasets into a training set consisting of $50,000$ images, and a validation set consisting of $10,000$ images. In all experiments, we choose a CNN with identical layers, $\tanh$ activation function, and a softmax classifier. For optimization, we use the following Block-Coordinate-Descent (BCD) method: Each iteration consists of one Gauss-Newton step with subsampled Hessian to update the forward propagation parameters and five inexact Newton steps to update the weights and biases of the classifier. To avoid overfitting and stabilize the process, we enforce spatial smoothness of the classification weights and smoothness across layers for the propagation parameters through derivative-based regularization as also suggested by Haber and Ruthotto (2017).

The validation accuracy on their native resolution is around 98.28% and 98.18% for the coarse and fine scale network, respectively. Next, we prolongate the classification weights and apply the multigrid prolongation to the convolution kernels from the coarse network to the fine resolution using the strategy outlined in Sec. 3. Using only the results from the coarse level and no training on the fine level, we get a validation accuracy of $91.0\%$. For comparison, using the original convolution kernels (i.e., without prolongation of the kernel weights) gives a validation accuracy of only $61.0\%$. Next, we restrict the classification weights and convolution kernel of the network trained on fine data as described in Sec. 3. This results in a validation accuracy of $94.9\%$ (compared to $84.1\%$ without restricting the kernels). We note that no coarse-scale training is performed.

**Shallow to Deep Training.** We solve a sequence of training problems using the MNIST example for CNNs whose depths increase in powers of two from two layers up to 64. For each CNN, we estimate the parameters using 20 iterations of the BCD. Except for the number of layers, all parameters are chosen as above. We compare the convergence properties of the learning algorithm using random and the multiscale initialization described in Sec. 3, which uses the prolongated network parameters from the previous level. The validation accuracy can be seen in Fig.2. Expectedly, the initial guesses provided by the multiscale process have a lower value of the loss function and higher validation accuracy. For the deeper networks, where training is most costly, the optimal accuracy is reached after only a few iterations using the multiscale method while for random initialization more iterations are needed to achieve a comparable accuracy.

**Multiscale CNN Training on ImageNet.** We compare multiscale training to training only the fine mesh CNN. We select ten categories from the ImageNet dataset (Russakovsky et al. 2015). ImageNet consists of images that are of varying dimensions and hence, as in He et al. (2016a), we pre-process the images to be of dimension $224 \times 224$. The image quality is sufficient to visually recognize the objects in the images, and the discrete images appear smooth, i.e., free of block artifacts. This yields a non-trivial training problem which we aim to solve using our multigrid approach. For each category, there are 1,300 images. We randomly divide the data into 10,000 training images and 3,000 test images. We use ResNet-34 architecture as described by He et al. (2016a) with two differences, first, the first CNN kernel is of dimension $3 \times 3 \times 64$ rather than $7 \times 7 \times 64$ and second, we did not use the fully connected layer. Instead, we directly connect the output of the average pooling to the classification layer with softmax activation. The average pooling renders the dimension of the penultimate layer independent of the dimension of the input layer.

For coarse-scale training, we restrict the $224 \times 224$ images to a $112 \times 112$ mesh and train ResNet-34 on the coarse data and use the results to initialize the fine-scale training (see Sec. 3). For comparison, we also train the CNN using the original image data. Fig. 3 and Tab. 1 show that the multi-

| | fine-scale only | coarse-to-fine |
|---|---|---|
| runtime [sec] | 59,122±7,540 | 43,882±3, 476 |
| validation acc. | 76.47±0.93% | 82.67±0.93% |

Table 1: Average runtime and validation accuracy for ImageNet-10 example for five different splits.

scale approach results in considerable reduction in the number of epochs and runtime. We use early stopping to stop training if the loss does not improve for 10 epochs. To ensure that this is not a phenomenon that is specific to the given train-test data split, we repeated the experiment five times using different splits.

## 5 Conclusions

In this work, we use the connection between optimal control and training CNNs to enable learning across scales. We show how this mathematical framework can be used to scale deep CNNs along two dimensions: Image resolution and depth. While the obtained multiscale approaches are new in deep learning, they are commonly used for the numerical solution of related optimal control problems, e.g., in image processing and parameter estimation.

Our method for connecting low- and high-resolution images is unique in that *it scales the parameter of the network rather than interpolating the image data to different resolutions*. To this end, we present an algebraic multigrid approach for computing convolution stencils that are consistent with coarse- and fine-scale images. We exemplified the benefit of our approach in two ways in Sec. 4. First, we show that CNNs trained on fine resolution images can be adapted and used to classify coarse-scale images and vice versa. Our method is advantageous when computational resources are limited, and the interpolation of images or videos is not practical. Second, we show that it is possible to use a coarse representation of the images to learn the convolution kernels and – after prolongation – use the weights to classify high-resolution images.

Our example in Sec. 4 shows that scaling the number of layers of the CNN can improve and accelerate the training of deep networks through initialization using results from shallow ones. In our experiment, this drastically reduces the number of iterations for the deep networks where cost-per-iteration is high. Similar results have also been observed in a follow-up study (Chang et al. 2017b)

## 6 Acknowledgements

This work is supported in part by the US National Science Foundation (NSF) award DMS 1522599.

## References

Ascher, U. 2010. *Numerical methods for Evolutionary Differential Equations*. Philadelphia: SIAM.

Biegler, L. T.; Ghattas, O.; Heinkenschloss, M.; Keyes, D.; and van Bloemen Waanders (Editors), B. 2009. *Real-Time PDE-Constrained Optimization*. SIAM, Philadelphia.

Bornemann, F. A., and Deuflhard, P. 1996. The cascadic multigrid method for elliptic problems. *Numerische Mathematik* 75(2):135–152.

Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. abs/1709.03698, 2017.

Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. Multi-level Residual Networks from Dynamical Systems View. abs/1710.10348, 2017.

Haber, E., and Modersitzki, J. 2004. Multilevel methods for image registration. *SIAM J. on Scientific Computing* 27:1594–1607.

Haber, E., and Ruthotto, L. 2017. Stable architectures for deep neural networks. *Inverse Problems* 10.1088/1361-6420/aa9a90.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 630–645. Springer.

Hernández-Lobato, J. M.; Gelbart, M. A.; Adams, R. P.; Hoffman, M. W.; and Ghahramani, Z. 2016. A general framework for constrained bayesian optimization using information-based search. volume 17, 1–53.

Jiang, J.; Trundle, P.; and Ren, J. 2010. Medical image analysis with artificial neural networks. *Computerized Medical Imaging and Graphics* 34:617631.

Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; and Fei-Fei, L. 2014. Large-scale video classification with convolutional neural networks. In *CVPR*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 61:10971105.

LeCun, Y., and Bengio, Y. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361:255258.

LeCun, Y.; Kavukcuoglu, K.; and Farabet, C. 2010. Convolutional networks and applications in vision. *IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems* 253256.

Modersitzki, J. 2004. *Numerical Methods for Image Registration*. Oxford.

Modersitzki, J. 2009. *FAIR: Flexible Algorithms for Image Registration*. Philadelphia: SIAM.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115(3):211–252.

Springenberg, J. T.; Dosovitskiy, A.; Brox, T.; and Ried-

miller, M. A. 2014. Striving for simplicity: The all convolutional net. *CoRR* abs/1412.6806.

Trottenberg, U.; Oosterlee, C. W.; and Schuller, A. 2000. *Multigrid*. Academic press.

Warner, M.; Ratcliffe, M. W.; Nangoo, T.; Morgan, J.; Umpleby, A.; Shah, N.; Vinje, V.; Štekl, I.; Guasch, L.; Win, C.; Conroy, G.; and Bertrand, A. 2013. Anisotropic 3d full-waveform inversion. *GEOPHYSICS* 78(20):59–80.

Zhang, R.; Tsai, P.-S.; Cryer, J.; and Shah, M. 1999. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21-8:690–706.