Multiscale Deep Equilibrium Models

Shaojie Bai
Carnegie Mellon University

Vladlen Koltun
Intel Labs

J. Zico Kolter
Carnegie Mellon University
Bosch Center for AI

Abstract

We propose a new class of implicit networks, the multiscale deep equilibrium model (MDEQ), suited to large-scale and highly hierarchical pattern recognition domains. An MDEQ directly solves for and backpropagates through the equilibrium points of multiple feature resolutions *simultaneously*, using implicit differentiation to avoid storing intermediate states (and thus requiring only $O(1)$ memory consumption). These simultaneously-learned multi-resolution features allow us to train a *single* model on a diverse set of tasks and loss functions, such as using a single MDEQ to perform both image classification and semantic segmentation. We illustrate the effectiveness of this approach on two large-scale vision tasks: ImageNet classification and semantic segmentation on high-resolution images from the Cityscapes dataset. In both settings, MDEQs are able to match or exceed the performance of recent competitive computer vision models: the first time such performance and scale have been achieved by an implicit deep learning approach. The code and pre-trained models are at `https://github.com/locuslab/mdeq`.

1 Introduction

State-of-the-art pattern recognition systems in domains such as computer vision and audio processing are almost universally based on multi-layer hierarchical feature extractors [32, 34, 35]. These models are structured in stages: the input is processed via a number of consecutive blocks, each operating at a different resolution [31, 52, 49, 25]. The architectures explicitly express hierarchical structure, with up- and downsampling layers that transition between consecutive blocks operating at different scales. An important motivation for such designs is the prominent multiscale structure and extremely high signal dimensionalities in these domains. A typical image, for instance, contains millions of pixels, which must be processed coherently by the model.

An alternative approach to differentiable modeling is exemplified by recent progress on *implicit* deep networks, such as Neural ODEs (NODEs) [12] and deep equilibrium models (DEQs) [5]. These constructions replace explicit, deeply stacked layers with analytical conditions that the model must satisfy, and are able to simulate models with "infinite" depth within a constant memory footprint. A notable achievement for implicit modeling is its successful application to large-scale sequences in natural language processing [5].

Is implicit deep learning relevant for general pattern recognition tasks? One clear challenge here is that implicit networks do away with flexible "layers" and "stages". It is therefore not clear whether they can appropriately model multiscale structure, which appears essential to high discriminative power in some domains. This is the challenge that motivates our work. Can implicit models that forego deep sequences of layers and stages attain competitive accuracy in domains characterized by rich multiscale structure, such as computer vision?

To address this challenge, we introduce a new class of implicit networks: the multiscale deep equilibrium model (MDEQ). It is inspired by DEQs, which attained high accuracy in sequence modeling [5]. We expand upon the DEQ construction substantially to introduce simultaneous equilibrium modeling

Preprint. Under review.

of multiple signal resolutions. MDEQ solves for equilibria of multiple resolution streams *simultaneously* by directly optimizing for stable representations on *all* feature scales at the same time. Unlike standard explicit deep networks, MDEQ does not process different resolutions in succession, with higher resolutions flowing into lower ones or vice versa. Rather, the different feature scales are maintained side by side in a single "shallow" model that is driven to equilibrium.

This design brings two major advantages. First, like the basic DEQ, our model does not require backpropagation through an explicit stack of layers and has an $O(1)$ memory footprint during training. This is especially important as pattern recognition systems are memory-intensive. Second, MDEQ rectifies one of the drawbacks of DEQ by exposing multiple feature scales at equilibrium, thereby providing natural interfaces for auxiliary losses and for compound training procedures such as pretraining (e.g., on ImageNet) and fine-tuning (e.g., on segmentation or detection tasks). Multiscale modeling enables a *single* MDEQ to simultaneously train for multiple losses defined on potentially very different scales, whose equilibrium features can serve as "heads" for a variety of tasks.

We demonstrate the effectiveness of MDEQ via extensive experiments on large-scale image classification and semantic segmentation datasets. Remarkably, this shallow implicit model attains comparable accuracy levels to state-of-the-art deeply-stacked explicit ones. On ImageNet classification, MDEQs outperform baseline ResNets (e.g., ResNet-101) with similar parameter counts, reaching 77.5% top-1 accuracy. On Cityscapes semantic segmentation (dense labeling of 2-megapixel images), identical MDEQs to the ones used for ImageNet experiments match the performance of recent explicit models while consuming much less memory. Our largest MDEQ surpasses 80% mIoU on the Cityscapes validation set, outperforming strong convolutional networks and coming tantalizingly close to the state of the art. This is by far the largest-scale application of implicit deep learning to date and a remarkable result for a class of models that until recently were applied largely to "toy" domains.

2 Background

Implicit Deep Learning. Virtually all modern deep learning approaches use *explicit* models, which provide explicit *computation graphs* for forward propagation. Backward passes proceed in reverse order through the same graph. This approach is the core of popular deep learning frameworks [1] and is associated with the very concept of "architecture". In contrast, *implicit* models do not have prescribed computation graphs. They instead posit a specific criterion that the model must satisfy (e.g., the endpoint of an ODE flow, or the root of an equation). Importantly, the algorithm that drives the model to fulfill this criterion is not prescribed. Therefore, implicit models can leverage black-box solvers in their forward passes and enjoy analytical backward passes that are independent of the forward pass trajectories.

Implicit modeling of hidden states has been explored by the deep learning community for decades. Pineda [42] and Almeida [2] studied implicit differentiation techniques for training recurrent dynamics, also known as recurrent back-propagation (RBP) [36]. Implicit approaches to network design have recently attracted renewed interest [19, 23]. For example, Neural ODEs (NODEs) [12, 18] model a recursive residual block using implicit ODE solvers, equivalent to a continuous ResNet taking infintesimal steps. Deep equilibrium models (DEQs) [5] solve for the fixed point of a sequence model with black-box root-finding methods, equivalent to finding the limit state of an infinite-layer network. Other instantiations of implicit modeling include optimization layers [17, 3], differentiable physics engines [14, 43], logical structure learning [56], and continuous generative models [24].

Our work takes the deep equilibrium approach [5] into signal domains characterized by rich multiscale structure. We develop the first one-layer implicit deep model that is able to scale to realistic visual tasks (e.g., megapixel-level images), and achieve competitive results in these regimes. In comparison, ODE-based models have so far only been applied to relatively low-dimensional signals due to numerical instability. For example, Chen et al. [12] downsampled 28×28 MNIST images to 7×7 before feeding them to Neural ODEs.

More broadly, our work can be seen as a new perspective on implicit models, wherein the models define and optimize simultaneous criteria over multiple data streams that can have different dimensionalities. While DEQs and NODEs have so far been defined on a single stream of features, a single MDEQ can jointly optimize features for different tasks, such as image segmentation and classification.

Multiscale Modeling in Computer Vision. Computer vision is a canonical application domain for hierarchical multiscale modeling. The field has come to be dominated by deep convolutional networks [32, 31]. Computer vision problems can be viewed in terms of the granularity of the desired output: from low-resolution, such as a label for a whole image [16], to high-resolution output that assigns a label to each pixel, as in semantic segmentation [47, 11, 59, 62]. State-of-the-art models for these problems are explicitly structured into sequential stages of processing that operate at different resolutions [31, 52, 49, 25]. For example, a ResNet [25] typically consists of 4-6 sequential stages, each operating at half the resolution of the preceding one. A dilated ResNet [60] uses a different schedule for the progression of resolutions. A DenseNet [26] uses different connectivity patterns to carry information between layers, but shares the overarching structure: a sequence of stages. Other designs progressively decrease feature resolution and then increase it step by step [44]. Downsampling and upsampling can also be repeated, again in an explicitly choreographed sequence [41, 51].

Multiscale modeling has been a central motif in computer vision. The Laplacian pyramid is an influential early example of multiscale modeling [7]. Multiscale processing has been built into convolutional networks for scene parsing by Farabet et al. [20] and has been explicitly addressed in many subsequent architectures [47, 11, 59, 8, 37, 62, 27, 10, 55].

Our work brings multiscale modeling to *implicit* deep networks. MDEQ has in essence only *one* stage, in which the different resolutions coexist side by side. The input is injected at the highest resolution and then propagated implicitly to the other scales, which are optimized simultaneously by a (black-box) solver that drives them to satisfy a joint equilibrium condition. Just like DEQs, an MDEQ is able to represent an "infinitely" deep network with only a constant memory cost.

3 Multiscale Deep Equilibrium Models

We begin by briefly summarizing the basic DEQ construction and some major challenges that arise when extending it to computer vision.

3.1 Deep Equilibrium (DEQ): Generic Formulation

One of the core ideas that motivated the DEQ approach was weight-tying: the same set of parameters can be shared across the layers of a deep network. Formally, Bai et al. [5] formulated an L-layer weight-tied transformation with parameter θ on hidden state $\mathbf{z}$ as

$$\mathbf{z}^{[i+1]} = f_\theta(\mathbf{z}^{[i]}; \mathbf{x}), \quad i = 0, \dots, L-1 \tag{1}$$

where the input representation $\mathbf{x}$ was injected into each layer. When sufficient stability conditions were ensured, stacking such layers infinitely (i.e., $L \to \infty$) was shown to essentially perform fixed-point iterations and thus tend to an equilibrium $\mathbf{z}^\star = f_\theta(\mathbf{z}^\star; \mathbf{x})$. Intuitively, as we iterate the transformation f_θ, the hidden representation tends to converge to a stable state, $\mathbf{z}^\star$. Such construction has a number of appealing properties. First, we can directly solve for the fixed point, which can be done faster than explicitly iterating through the layers. We formulate this as a root-finding problem:

$$g_\theta(\mathbf{z}; \mathbf{x}) \coloneqq f_\theta(\mathbf{z}; \mathbf{x}) - \mathbf{z} \quad \Longrightarrow \quad \mathbf{z}^\star = \mathsf{Rootfind}(g_\theta; \mathbf{x}) \tag{2}$$

For example, one can leverage Newton or quasi-Newton methods to achieve quadratic or superlinear convergence to the root. Second, one can directly backpropagate through the equilibrium state using the Jacobian of g_θ at $\mathbf{z}^\star$, without tracing through the forward root-finding process. Formally, given a loss $\ell = \mathcal{L}(\mathbf{z}^\star, \mathbf{y})$ (where $\mathbf{y}$ is the target), the gradients can be written as

$$\frac{\partial \ell}{\partial \theta} = \frac{\partial \ell}{\partial \mathbf{z}^\star} \left(-J_{g_\theta}^{-1}|_{\mathbf{z}^\star}\right) \frac{\partial f_\theta(\mathbf{z}^\star; \mathbf{x})}{\partial \theta} \qquad \frac{\partial \ell}{\partial \mathbf{x}} = \frac{\partial \ell}{\partial \mathbf{z}^\star} \left(-J_{g_\theta}^{-1}|_{\mathbf{z}^\star}\right) \frac{\partial f_\theta(\mathbf{z}^\star; \mathbf{x})}{\partial \mathbf{x}}. \tag{3}$$

See Bai et al. [5] for the proof, which is based on the implicit function theorem [29]. This means that the forward pass of a DEQ can rely on any black-box root solver, while the backward pass is based independently on differentiating through only one layer at the equilibrium (i.e., $\frac{\partial f_\theta(\mathbf{z}^\star; \mathbf{x})}{\partial(\cdot)}$). The memory consumption of the entire training process is equivalent to that of just one layer rather than $L \to \infty$ layers. Since the Jacobian of g_θ can be expensive to compute, DEQs solve for a linear equation involving a vector-Jacobian product, which is a lot cheaper:

$$\mathbf{x}(J_{g_\theta}|_{\mathbf{z}^\star}) + \frac{\partial \ell}{\partial \mathbf{z}^\star} = \mathbf{0}. \tag{4}$$

The DEQ model therefore solves for the network output at its *infinite depth*, with each step of the model now implicitly defined to reach an analytical objective (the equilibrium).

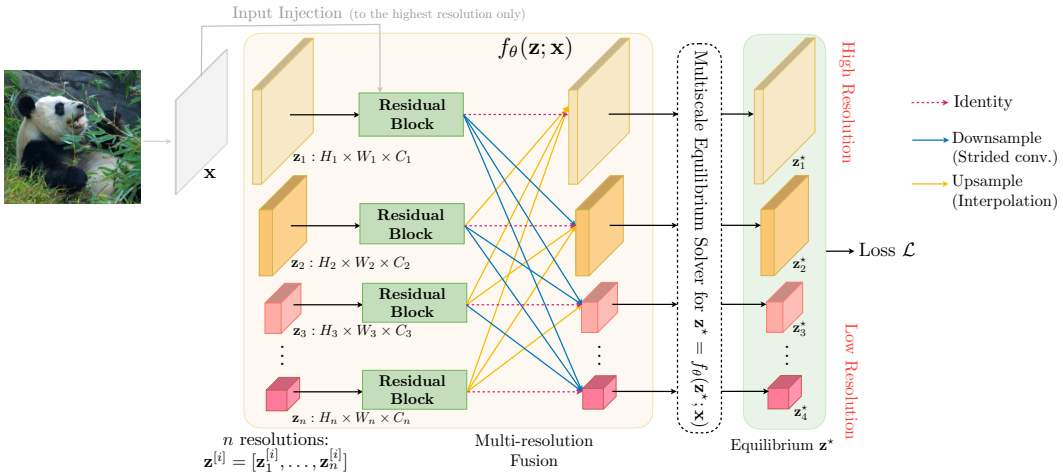


Figure 1: The structure of a multiscale deep equilibrium model (MDEQ). *All components* of the model are shown in this figure. MDEQ consists of a transformation f_θ that is driven to equilibrium. Features at different scales coexist side by side and are driven to equilibrium simultaneously.

Challenges. The construction of Bai et al. [5], which we have just summarized, was primarily aimed at processing *sequences*. As we transition from sequences to high-resolution images, we note important differences between these domains. First, unlike typical autoregressive sequence learning problems (e.g., language modeling), where input and output have identical length and dimensionality, general pattern recognition systems (such as those in vision) entail multi-stage modeling via a combination of up- and downsampling in the architecture. The basic DEQ construction does not exhibit such structure. Second, the output of a computer vision task such as image classification (a label) or object localization (a region) may have very different dimensionality from the input (a full image): again a feature that the basic DEQ does not support. Third, state-of-the-art models for tasks such as semantic segmentation are commonly based on "backbones" that are pretrained for image classification, even though the tasks are structurally different and their outputs have very different dimensionalities (e.g., one label for the whole image versus a label for each pixel). It's not clear how a DEQ construction can support such transfer. Fourth, whereas past work on DEQs could leverage state-of-the-art weight-tied architectures for sequence modeling as the basis for the transformation f_θ [4, 15], no such models exist in computer vision.

3.2 The MDEQ Model

Notation. Figure 1 illustrates the *entire* structure of MDEQ. As before, f_θ denotes the transformation that is (implicitly) iterated to a fixed point, $\mathbf{x}$ is the (precomputed) input representation provided to f_θ, and $\mathbf{z}$ is the model's internal state. We omit the batch dimension for clarity.

Transformation f_θ. The central part of MDEQ is the transformation f_θ that is driven to equilibrium. We use a simple design in which features at each resolution are first taken through a residual block. The blocks are shallow and are identical in structure. At resolution i, the residual block receives the internal state $\mathbf{z}_i$ and outputs a transformed feature tensor $\mathbf{z}_i^+$ at the same resolution. Notably, the highest resolution stream (i.e., $i = 1$) also receives an input injection $\mathbf{x}$ that is precomputed directly from the source image and injected to the highest-resolution residual block. (See Eq. (5) and the discussion below.)

The internal structure of the residual block is shown in Figure 2. We largely adopt the design of He et al. [25], but use group normalization [57] rather than batch normalization [28], for stability reasons that are discussed in Section 3.3. The residual block at resolution i can be formally expressed as

$$\begin{aligned} \tilde{\mathbf{z}}_i &= \text{GroupNorm}\big(\text{Conv2d}(\mathbf{z}_i)\big) \\ \hat{\mathbf{z}}_i &= \text{GroupNorm}\big(\text{Conv2d}(\text{ReLU}(\tilde{\mathbf{z}}_i)) + 1_{\{i=1\}} \cdot \mathbf{x}\big) \\ \mathbf{z}_i^+ &= \text{GroupNorm}\big(\text{ReLU}(\hat{\mathbf{z}}_i + \mathbf{z}_i)\big) \end{aligned} \tag{5}$$

Following these blocks, the second part of f_θ is a multi-resolution fusion step that mixes the feature maps across different scales (see Figure 1). The transformed features $\mathbf{z}_i^+$ undergo either upsampling

or downsampling from the current scale i to each other scale $j \neq i$. In our construction, downsampling is performed by $j - i$ consecutive 2-strided 3×3 Conv2d, whereas upsampling is performed by direct bilinear interpolation. The final output at scale j is formed by summing over the transformed feature maps provided from all incoming scales i (along with $\mathbf{z}_j^+$); i.e., the output feature tensor at each scale is a mixture of transformed features from all scales. This forces the features at all scales to be consistent and drives the whole system to a coordinated equilibrium that harmonizes the representations across scales.

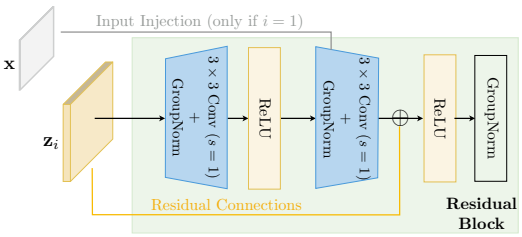


Figure 2: The residual block used in MDEQ. An MDEQ contains only *one* such layer.

Input Representation. The raw input first goes through a transformation (e.g., a linear layer that aligns the feature channels) to form $\mathbf{x}$, which will be provided to f_θ. The existence of such input injection is vital to implicit models as it (along with θ) correlates the flow of the dynamical system with the input. However, unlike multiscale input representations used by some explicit vision architectures [20, 11], we only inject $\mathbf{x}$ to the highest-resolution feature stream (see Eq. (5)). The input is provided to MDEQ at a single (full) resolution. The lower resolutions hence start with no knowledge at all about the input; this information will only *implicitly* propagate through them as all scales are gradually driven to coordinated equilibria $\mathbf{z}^\star$ by the (black-box) solver.

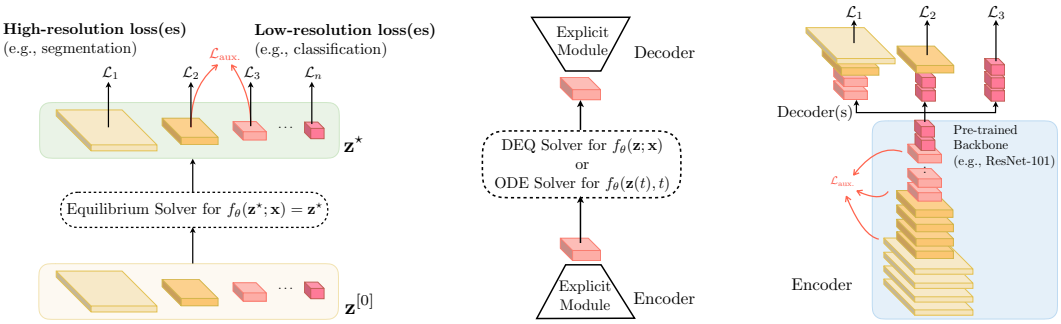
(Limited-memory) Multiscale Equilibrium Solver. In the DEQ, the internal state is a single tensor $\mathbf{z}$ [5]. The MDEQ state, however, is a *collection* of tensors at n resolutions: $\mathbf{z} = [\mathbf{z}_1, \dots, \mathbf{z}_n]$. Note that this is not a concatenation, as the different $\mathbf{z}_i$ have different dimensionalities, feature resolutions, and semantics.

With this in mind, our equilibrium solver leverages Broyden's method. We initialize the internal states by setting $\mathbf{z}_i^{[0]} = \mathbf{0}$ for all scales i. $\mathbf{z} = [\mathbf{z}_1, \dots, \mathbf{z}_n]$ is maintained as a collection of n tensors whose respective equilibrium states (i.e., roots) are solved for and backpropagated through simultaneously (with each resolution inducing its own loss).

The original Broyden solver was not efficient enough when applied to computer vision datasets, which have very high dimensionality. For example, in the Cityscapes segmentation task (see Section 4), the Jacobian of a 4-resolution MDEQ at $\mathbf{z}^\star$ is well over 2,000 times larger than its single-scale counterpart in word-level language modeling [5]. Note that even with low-rank approximations of the Jacobian in quasi-Newton methods, the high dimensionality of images can make storing these updates extremely expensive. To address this, we improve the memory efficiency of the forward and backward passes by optimizing Broyden's method. We implemented a new solver that is inspired by Limited-memory BFGS (L-BFGS) [38], where we only keep the latest m low-rank updates at any step and discard the earlier ones (see Appendix B.1).

Pretraining and Auxiliary Losses. Figure 3 provides a comparison of MDEQ with single-stream implicit models such as the DEQ, and with explicit deep networks in computer vision. These different models expose different "interfaces" that can be used to define losses for different tasks. Prior implicit models such as neural ODEs and DEQs typically assume that a loss is defined on a single stream of implicit hidden states, which has a uniform input and output shape (Figure 3b). It is therefore not clear how such a model can be flexibly transferred across structurally different tasks (e.g., pretraining on image classification and fine-tuning on semantic segmentation). Furthermore, there is no natural way to define auxiliary losses [33], because there are no "layers" and the forward and backward computation trajectories are decoupled.

In comparison, MDEQ exposes convenient "interfaces" to its states at multiple resolutions. One resolution (the highest) can be the same as the resolution of the input, and can be used to define losses for dense prediction tasks such as semantic segmentation. Another resolution (the lowest) can be a vector in which the spatial dimensions are collapsed, and can be used to define losses for image-level labeling tasks such as image classification. This suggests clean protocols for training the same model for different tasks, either jointly (e.g., multi-task learning in which structurally different supervision flows through multiple heads) or in sequence (e.g., pretraining for image classification through one head and fine-tuning for semantic segmentation through another).



(a) MDEQ exposes multiple interfaces at equilibrium

(b) Single-stream implicit models (e.g., DEQs and NODEs)

(c) Explicit deep models in vision

Figure 3: A visual comparison of MDEQ with prior implicit models and with standard explicit models in computer vision. Equilibrium states at multiple resolutions enable MDEQ to incorporate supervision in different forms.

3.3 Integrating Common DL Techniques with MDEQs

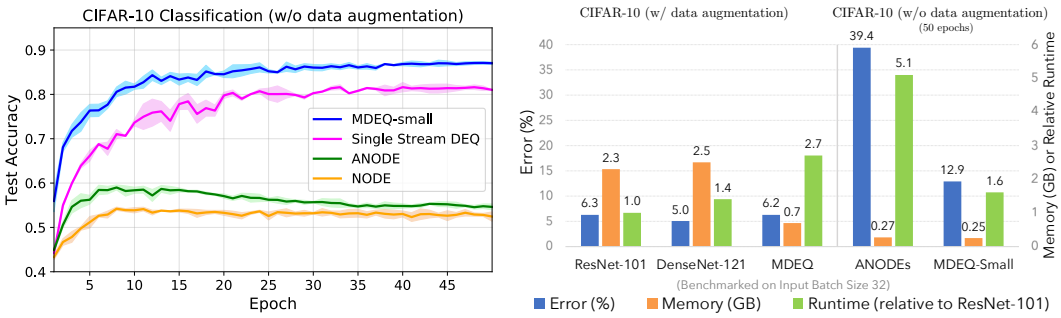
MDEQ simulates an "infinitely" deep network by implicitly modeling one layer. Such implicitness calls for care when adapting common deep learning practices. We provide an exploration of such adaptations and their impact on the training dynamics of MDEQ. We believe these observations will also be valuable for future research on implicit models.

Normalization. Layer normalization of hidden activations in f_θ played an important role in constraining the output and stabilizing DEQs on sequences [5]. A natural counterpart in vision is batch normalization (BN) [28]. However, BN is not directly suitable for implicit models, since it estimates population statistics based on layers, which are implicit in our setting, and the Jacobian matrix of the transformation f_θ will scale badly to make the fixed point significantly harder to solve for. We therefore use group normalization (GN) [57], which groups the input channels and performs normalization within each group. GN is independent of batch size and offers more natural support for transfer learning (e.g., pretraining and fine-tuning on structurally different tasks). Unlike in DEQs, we keep the learnable affine parameters of GN.

Dropout. The conventional spatial dropout used by explicit vision models applies a random mask to given layers in the network [50]. A new mask is generated whenever dropout is invoked. Such layer-based stochasticity can significantly hurt the stability of convergence to the equilibrium. In fact, as two adjacent calls to f_θ most probably will have different Bernoulli dropout masks, it is almost impossible to reach a fixed point where $f_\theta(\mathbf{z}^\star;\mathbf{x})=\mathbf{z}^\star$. We therefore adopt variational dropout [21] and apply the exact same mask at all invocations of f_θ in a given training iteration. The mask is reset at each iteration.

Nonlinearities. The multiscale features are initialized to $\mathbf{z}_i^{[0]}=\mathbf{0}$ for all resolutions i. However, we found that this could induce certain instabilities when training MDEQ (especially in the starting phase of it), most likely due to the drastic change of slope of the ReLU non-linearity at the origin, where the derivative is undefined [22]. To combat this, we replace the last ReLU in both the residual block and the multiscale fusion by a softplus [22] in the initial phase of training. These are later switched back to ReLU. The softplus provides a smooth approximation to the ReLU, but has slope $1-\frac{1}{1+\exp(\beta\mathbf{z})}\to\frac{1}{2}$ around $\mathbf{z}=0$ (where β controls the curvature).

Convolution and Convergence to Equilibrium. Whereas the original DEQ model focused primarily on self-attention transformations [54], where all hidden units communicate globally, MDEQ models face additional challenges due to the nature of typical vision models. Specifically, our MDEQ models employ convolutions with small receptive fields (e.g., the two 3×3 convolutional filters in f_θ's residual block) on potentially very large images: for instance, we eventually evaluate our semantic segmentation model on megapixel-scale images. In consequence, we typically need a higher number of root-finding iterations to converge to an exact equilibrium. While this does pose a challenge, we find that using the aforementioned strategies of 1) multiscale simultaneous up- and downsampling and 2) quasi-Newton root-finding, drives the model close to equilibrium within a reasonable number of iterations. We further analyze convergence behavior in Appendix B.



(a) Training dynamics of implicit models

(b) Runtime and memory consumption on CIFAR-10

Figure 4: Left: test accuracy as a function of training epochs. Right: MDEQ-Small and ANODEs correspond to the settings and results reported in Table 1. For all metrics, lower is better.

4 Experiments

In this section, we investigate the empirical performance of MDEQs from two aspects. First, as prior implicit approaches such as NODEs have mostly evaluated on smaller-scale benchmarks such as MNIST [32] and CIFAR-10 (32×32 images) [30], we compare MDEQs with these baselines on the same benchmarks. We evaluate both training-time stability and inference-time performance. Second, we evaluate MDEQs on large-scale computer vision tasks: ImageNet classification [16] and semantic segmentation on the Cityscapes dataset [13]. These tasks have extremely high-dimensional inputs (e.g., 2048×1024 images for Cityscapes) and are dominated by explicit models. We provide more detailed descriptions of the tasks, hyperparameters, and training settings in Appendix A.

Table 1: Evaluation on CIFAR-10. Standard deviations are calculated on 5 runs.

	Model Size	Accuracy
CIFAR-10 (*without* data augmentation)		
Neural ODEs [18]	172K	53.7% $\pm$ 0.2%
Aug. Neural ODEs [18]	172K	60.6% $\pm$ 0.4%
Single-stream DEQ [5]	170K	82.2% $\pm$ 0.3%
ResNet-18 [25] **[Explicit]**	170K	81.6% $\pm$ 0.3%
MDEQ-small (ours)	**170K**	**87.1%** $\pm$ 0.4%
CIFAR-10 (*with* data augmentation)		
ResNet-18 [25] **[Explicit]**	10M	**92.9%** $\pm$ 0.2%
MDEQ (ours)	**10M**	**93.8%** $\pm$ 0.3%

Our focus is on the behavior of MDEQs and their competitiveness with prior implicit or explicit models. We are not aiming to set a new state of the art on ImageNet classification or Cityscapes segmentation, as this typically involves substantial additional investment [58]. We will release our full implementation and pretrained models. A copy of the code is provided in the supplement.

All experiments with MDEQs use the limited-memory version of Broyden's method in both forward and backward passes, and the root solvers are stopped whenever 1) the objective value reaches some predetermined threshold ε or 2) the solver's iteration count reaches a limit T. On large-scale vision benchmarks (ImageNet and Cityscapes), we downsample the input twice with 2-strided convolutions before feeding it into MDEQs, following the common practice in explicit models [62, 55]. We use the cosine learning rate schedule for all tasks [40].

4.1 Comparing with Prior Implicit Models on CIFAR-10

Following the setting of Dupont et al. [18], we run the experiments on CIFAR-10 classification (without data augmentation) for 50 epochs and compare models with approximately the same number of parameters. However, unlike the ODE-based approaches, we do not perform downsamplings on the raw images before passing the inputs to the MDEQ solver (so the highest-resolution stream stays at 32×32). When training the MDEQ model, *all* resolutions are used for the final prediction: higher-resolution streams go through additional downsampling layers and are added to the lowest-resolution output to make a prediction (i.e., a form of auxiliary loss).

The results of MDEQ models on CIFAR-10 image classification are shown in Table 1. Compared to NODEs [12] and Augmented NODEs [18], a small MDEQ with a similar parameter count improves accuracy by more than 20 percentage points: an error reduction by *more than a factor of 2*. MDEQ also improves over the single-stream DEQ (applied at the highest resolution). The training dynamics of the different models are visualized in Figure 4a. Finally, a larger MDEQ matches and even

Table 2: Evaluation on ImageNet classification with top-1 and top-5 accuracies reported. MDEQs were trained for 100 epochs.

	Model Size	top1 Acc.	top5 Acc.
AlexNet [31]	238M	57.0%	80.3%
ResNet-18 [25]	13M	70.2%	89.9%
ResNet-34 [25]	21M	74.8%	91.1%
Inception-V2 [28]	12M	74.8%	92.2%
ResNet-50 [25]	26M	75.1%	92.5%
HRNet-W18-C [55]	21M	76.8%	93.4%
Single-stream DEQ + global pool [5]	18M	72.9%	91.0%
MDEQ-small (ours) [Implicit]	18M	75.5%	92.7%
ResNet-101 [25]	52M	77.1%	93.5%
W-ResNet-50 [61]	69M	78.1%	93.9%
DenseNet-264 [26]	74M	79.7%	94.8%
MDEQ-large (ours) [Implicit]	63M	77.5%	93.6%
MDEQ-XL (ours) [Implicit]	81M	79.2%	94.5%

Table 3: Evaluation on Cityscapes `val` semantic segmentation. "*" marks the current SOTA. Higher mIoU (mean Intersection over Union) is better.

	Backbone	Model Size	mIoU
ResNet-18-A [39]	ResNet-18	3.8M	55.4
ResNet-18-B [39]	ResNet-18	15.24M	69.1
MobileNetV2Plus [46]	MobileNetV2	8.3M	74.5
GSCNN [53]	ResNet-50	-	73.0
HRNetV2-W18-Small-v2* [55]	HRNet	4.0M	76.0
MDEQ-small (ours) [Implicit]	MDEQ	7.8M	75.1
U-Net++ [64]	ResNet-101	59.5M	75.5
Dilated-ResNet [60]	D-ResNet-101	52.1M	75.7
PSPNet [62]	D-ResNet-101	65.9M	78.4
DeepLabv3 [9]	D-ResNet-101	58.0M	78.5
PSANet [63]	ResNet-101	-	78.6
HRNetV2-W48* [55]	HRNet	65.9M	81.1
MDEQ-large (ours) [Implicit]	MDEQ	53.0M	77.8
MDEQ-XL (ours) [Implicit]	MDEQ	70.9M	80.3

exceeds the accuracy of a ResNet-18 with the same capacity: the first time such performance has been demonstrated by an implicit model.

4.2 ImageNet Classification

We now test the ability of MDEQ to scale to a much larger dataset with higher-resolution images: ImageNet [16]. As with CIFAR-10 classification, we add a shallow classification layer after the MDEQ module to fuse the equilibrium outputs from different scales, and train on a combined loss.

We benchmark both a small MDEQ model and a large MDEQ to provide appropriate comparisons with a number of reference models, such as ResNet-18, -34, -50, and -101 [25]. Note that MDEQ has only *one layer* of residual blocks followed by multi-resolution fusion. Therefore, to match the capacity of standard explicit models, we need to increase the feature dimensionality within MDEQ. This is accomplished by adjusting the width of the convolutional filter within the residual block (see Figure 2).

Table 2 shows the accuracy of two MDEQs (of different sizes) in comparison to well-known reference models in computer vision. MDEQs are remarkably competitive with strong explicit models. For example, a small MDEQ with 18M parameters outperforms ResNet-18 (13M parameters), ResNet-34 (21M parameters), and even ResNet-50 (26M parameters). A larger MDEQ (64M parameters) reaches the same level of performance as ResNet-101 (52M parameters). This is far beyond the scale and accuracy levels of prior applications of implicit modeling.

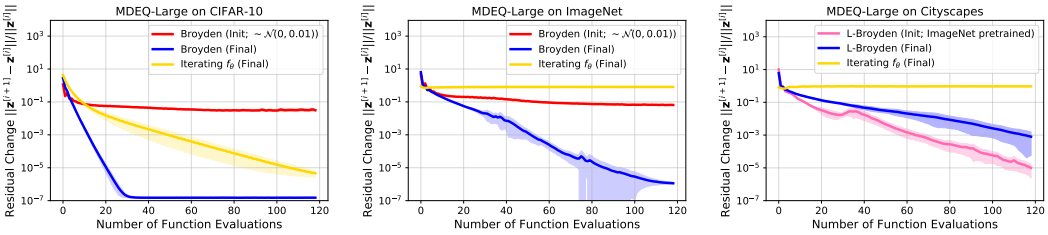
4.3 Cityscapes Semantic Segmentation

After training on ImageNet, we train *the same MDEQs* for semantic segmentation on the Cityscapes dataset [13]. When transferring the models from ImageNet to Cityscapes, we directly use the highest-resolution equilibrium output $\mathbf{z}_1^\star$ to train on the highest-resolution loss. Thus *MDEQ is its own "backbone"*. We train on the Cityscapes `train` set and evaluate on the `val` set. Following the evaluation protocol of Zhao et al. [63] and Wang et al. [55], we test on a single scale with no flipping.

MDEQs attain remarkably high levels of accuracy. They come close to the current state of the art, and match or outperform well-known and carefully architected explicit models that were released in the past two years. A small MDEQ (7.8M parameters) achieves a mean IoU of 75.1. This improves upon a MobileNetV2Plus [46] of the same size and is close to the SOTA for models on this scale. A large MDEQ (53.5M parameters) reaches 77.8 mIoU, which is within 1 percentage point of highly regarded recent semantic segmentation models such as DeepLabv3 [9] and PSPNet [62], whereas a larger version (70.9M parameters) surpasses them. It is surprising that such levels of accuracy can be achieved by a "shallow" implicit model, based on principles that have not been applied to this domain before. Examples of semantic segmentation results are shown in Appendix C.

4.4 Runtime and Memory Consumption

We provide a runtime and memory analysis of MDEQs using CIFAR-10 data, with input batch size 32. Since prior implicit models such as ANODEs [18] are relatively small, we provide results for both MDEQ and MDEQ-small for a fair comparison. All computation speeds are benchmarked relative



(a) CIFAR-10 classification (b) ImageNet classification (c) Cityscapes segmentation

Figure 5: Plots of MDEQ's convergence to equilibrium (measured by $\frac{\|\mathbf{z}^{[i+1]} - \mathbf{z}^{[i]}\|}{\|\mathbf{z}^{[i]}\|}$) as a function of the number of times we evaluate f_θ. As input image resolution grows (from CIFAR-10 to Cityscapes), MDEQ takes more steps to converge with (L-)Broyden's method. Standard deviation is calculated on 5 randomly selected batches from each dataset.

to the ResNet-101 model (about 150ms per batch) on a single RTX 2080 Ti GPU. The results are summarized in Figure 4b.

MDEQ saves more than 60% of the GPU memory at training time compared to explicit models such as ResNets and DenseNets, while maintaining competitive accuracy. Training a large MDEQ on ImageNet consumes about 6GB of memory, which is mostly used by Broyden's method. This low memory footprint is a direct result of the analytical backward pass. Meanwhile, MDEQs are generally slower than explicit networks. We observe a $2.7\times$ slowdown for MDEQ compared to ResNet-101, a tendency similar to that observed in the sequence domain [5]. A major factor contributing to the slowdown is that MDEQs maintain features at all resolutions throughout, whereas explicit models such as ResNets gradually downsample their activations and thus reduce computation (e.g., 70% of ResNet-101 layers operate on features that are downsampled by 8×8 or more). However, when compared to ANODEs with 172K parameters, an MDEQ of similar size is $3\times$ *faster while achieving a* $3\times$ *error reduction*. Additional discussion of runtime and convergence is provided in Appendix B.

4.5 Equilibrium Convergence on High-resolution Inputs

As we scale MDEQ to higher-resolution inputs, the equilibrium solving process becomes more challenging. This is illustrated in Figure 5, where we show the equilibrium convergence of MDEQ on CIFAR-10 (low-resolution), ImageNet (medium-resolution) and Cityscapes (high-resolution) images by measuring the change of residual with respect to the number of function evaluations. We empirically find that (limited-memory) Broyden's method and multiscale fusion both help stabilize the convergence on high-resolution data. For example, in all three cases, Broyden's method (blue lines in Figure 5) converges to the fixed point in a more stable and efficient manner than simply iterating f_θ (yellow lines). Further analysis of convergence behavior is provided in Appendix B.

5 Conclusion

We introduced multiscale deep equilibrium models (MDEQs): a new class of implicit architectures for domains characterized by high dimensionality and multiscale structure. Unlike prior implicit models, such as DEQs and Neural ODEs, an MDEQ solves for and backpropagates through synchronized equilibria of multiple feature representations at different resolutions. We show that a single MDEQ can be used for different tasks, such as image classification and semantic segmentation. Our experiments demonstrate for the first time that "shallow" implicit models can scale to practical computer vision tasks and achieve competitive performance that matches explicit architectures characterized by sequential processing through deeply stacked layers.

The remarkable performance of implicit models in this work brings up core questions in machine learning. Are complex stage-wise hierarchical architectures, which have dominated deep learning to date, necessary? MDEQ exemplifies a different approach to differentiable modeling. The most significant message of our work is that this approach may be much more relevant in practice than previously appeared. We hope that this will contribute to the development of implicit deep learning and will further broaden the agenda in differentiable modeling.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. TensorFlow: A system for large-scale machine learning. In *OSDI*, 2016.

[2] L. B. Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Artificial Neural Networks*. 1990.

[3] B. Amos and J. Z. Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

[4] S. Bai, J. Z. Kolter, and V. Koltun. Trellis networks for sequence modeling. In *International Conference on Learning Representations (ICLR)*, 2019.

[5] S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, 2019.

[6] C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 1965.

[7] P. Burt and E. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4), 1983.

[8] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017.

[10] L.-C. Chen, M. D. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *Advances in Neural Information Processing Systems*, 2018.

[11] 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. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4), 2018.

[12] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018.

[13] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes dataset for semantic urban scene understanding. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter. End-to-end differentiable physics for learning and control. In *Advances in Neural Information Processing Systems*, 2018.

[15] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and Ł. Kaiser. Universal transformers. In *International Conference on Learning Representations (ICLR)*, 2019.

[16] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[17] J. Djolonga and A. Krause. Differentiable learning of submodular models. In *Advances in Neural Information Processing Systems*, 2017.

[18] E. Dupont, A. Doucet, and Y. W. Teh. Augmented neural ODEs. In *Advances in Neural Information Processing Systems*, 2019.

[19] L. El Ghaoui, F. Gu, B. Travacca, and A. Askari. Implicit deep learning. *arXiv:1908.06315*, 2019.

[20] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 2013.

[21] Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2016.

[22] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *AISTATS*, 2011.

[23] S. Gould, R. Hartley, and D. Campbell. Deep declarative networks: A new hope. *arXiv:1909.04866*, 2019.

[24] W. Grathwohl, R. T. Chen, J. Betterncourt, I. Sutskever, and D. Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations (ICLR)*, 2019.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[26] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[27] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations (ICLR)*, 2018.

[28] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

[29] S. G. Krantz and H. R. Parks. *The implicit function theorem: History, theory, and applications*. Springer, 2012.

[30] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[32] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4), 1989.

[33] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, 2015.

[34] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International Conference on Machine Learning (ICML)*, 2009.

[35] H. Lee, P. Pham, Y. Largman, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems*, 2009.

[36] R. Liao, Y. Xiong, E. Fetaya, L. Zhang, K. Yoon, X. Pitkow, R. Urtasun, and R. Zemel. Reviving and improving recurrent back-propagation. In *International Conference on Machine Learning (ICML)*, 2018.

[37] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[38] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3), 1989.

[39] Y. Liu, C. Shu, J. Wang, and C. Shen. Structured knowledge distillation for dense prediction. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[40] I. Loshchilov and F. Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

[41] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision (ECCV)*, 2016.

[42] F. J. Pineda. Generalization of back propagation to recurrent and higher order neural networks. In *Advances in Neural Information Processing Systems*, 1988.

[43] Y.-L. Qiao, J. Liang, V. Koltun, and M. C. Lin. Scalable differentiable physics for learning and control. In *International Conference on Machine Learning (ICML)*, 2020.

[44] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[45] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, 2016.

[46] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[47] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4), 2017.

[48] J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 1950.

[49] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[50] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15, 2014.

[51] S. Sun, J. Pang, J. Shi, S. Yi, and W. Ouyang. FishNet: A versatile backbone for image, region, and pixel level prediction. In *Advances in Neural Information Processing Systems*, 2018.

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[53] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler. Gated-scnn: Gated shape cnns for semantic segmentation. In *International Conference on Computer Vision (ICCV)*, 2019.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[55] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao. Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[56] P.-W. Wang, P. Donti, B. Wilder, and Z. Kolter. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning (ICML)*, 2019.

[57] Y. Wu and K. He. Group normalization. In *European Conference on Computer Vision (ECCV)*, 2018.

[58] Q. Xie, E. Hovy, M.-T. Luong, and Q. V. Le. Self-training with noisy student improves ImageNet classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[59] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016.

[60] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[61] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv:1605.07146*, 2016.

[62] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[63] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. Change Loy, D. Lin, and J. Jia. PSANet: Point-wise spatial attention network for scene parsing. In *European Conference on Computer Vision (ECCV)*, 2018.

[64] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang. UNet++: A nested U-Net architecture for medical image segmentation. In *MICCAI*, 2018.

A Task Descriptions and Training Settings

We provide a detailed description of all tasks and some additional details on the training of MDEQ.

Image Classification on CIFAR-10. CIFAR-10 is a well-known computer vision dataset that consists of 60,000 color images, each of size 32×32 [30]. There are 10 object classes and 6,000 images per class. The entire dataset is divided into training (50K images) and testing (10K) sets.

We use two different training settings for evaluating the MDEQ model on CIFAR-10. Following Dupont et al. [18], we compare MDEQ-small with other implicit models on CIFAR-10 images *without data augmentation* (i.e., the original, raw images), using approximately 170K learnable parameters in the model. In the second setting, we apply data augmentation to the input images (i.e., random cropping, horizontal flipping, etc.), a setting that most competitive vision baselines (e.g., ResNets) use by default.

Image Classification on ImageNet. The dataset we use contains 1.2 million labeled training images from ImageNet [31] distributed over 1,000 classes, and a test set of 150,000 images. The original ImageNet consists of variable-resolution images, and we follow the standard setting [25] to use the 224×224 crops as inputs to the model.

ImageNet is frequently used for pretraining general-purpose image feature extractors that are used on downstream tasks [25, 61, 60, 55]. We train a small and large MDEQ model, which will act as their own "backbone" when later fine-tuned on the Cityscapes segmentation task. We train MDEQ on ImageNet for 100 epochs. Following the practice of Bai et al. [5] with DEQ models for sequences, we start the training (the first few epochs) of MDEQ with a shallow (5-layer) weight-tied stacking of f_θ to warm up the weights, and then switch to the implicit equilibrium (root) solver for the rest of the training epochs.

Semantic Segmentation on Cityscapes. Cityscapes is a large-scale urban scene understanding dataset containing high-quality, pixel-level annotated street scene images from 50 cities [13]. The dataset consists of 5,000 images, which are divided into 2,975 (`train`), 500 (`val`) and 1,525 (`test`) sets. Each pixel is classified in a 19-way fashion for evaluation.

We follow the training protocol of prior works [62, 55] to train the MDEQ models on the Cityscapes `train`, and perform random cropping (to 1024×512) and random horizontal flipping on the training inputs. The models are evaluated on the Cityscapes `val` (single scale and no random flipping) with the original resolution 2048×1024. We use the identical MDEQ model(s) as used in ImageNet training, but now predict with the high-resolution head.

Hyperparameters. We provide the hyperparameters of the models we used in each of these tasks in Table 4. Note that we use a *single* model for both ImageNet classification and Cityscapes segmentation, so the models share the same configuration (highlighted in red in Table 4 for clarity). For all tasks, the MDEQ features in resolution $i = 1, \dots, n$ take the shape $\left(\frac{H}{2^{i-1}}, \frac{W}{2^{i-1}}\right)_{i=1,\dots,n}$, where H, W are the dimensions of the original input. In other words, each resolution uses half the feature size of its next higher resolution stream. We apply weight normalization [45] to all of the learnable weights in f_θ.

Hardware. Experiments were conducted on RTX 2080 Ti GPUs. Both ImageNet and Cityscapes experiments used 4 GPUs, while CIFAR-10 classification models were trained on 1 GPU (including the baselines).

Initialization of MDEQ Models. For CIFAR-10 and ImageNet, we initialize the parameters of f_θ randomly from $\mathcal{N}(0, 0.01)$ (Cityscapes MDEQs use pretrained ImageNet MDEQs). Generally, we observe that the final performance of MDEQ is not sensitive to the choice of initialization distribution. However, such random initialization could occasionally induce instabilities in the starting phase of the training (see red lines in Figure 5). We solve this problem by either 1) temporarily replacing ReLU with softplus in the first few epochs of training; or 2) warming up the weights by training a shallow (e.g., 5-layer) weight-tied stacking of f_θ, then switching to MDEQ's equilibrium solver for the rest of the training.

Table 4: Settings & hyperparameters of each task. "cls." means classification task, and "seg." means segmentation task. These models coorespond to the ones reported in Tables 1, 2, and 3.

	CIFAR-10 (cls.)		**ImageNet** (cls.)		**Cityscapes** (seg.)	
	MDEQ-Small	MDEQ	MDEQ-Small	MDEQ-Large	MDEQ-Small	MDEQ-Large
Input Image Size	32×32		224×224		1024×512 (train) 2048×1024 (test)	
Number of Epochs	50	200	100	100	480	480
Batch Size	128	128	128	128	12	12
Optimizer	Adam	Adam	SGD	SGD	SGD	SGD
(Start) Learning Rate	0.001	0.001	0.05	0.05	0.01	0.01
Nesterov Momentum	-	-	0.9	0.9	-	-
Weight Decay	0	0	5e-5	1e-4	2e-4	3e-4
Use Pre-trained Weights	-	-	-	-	Yes, from ImageNet	Yes, from ImageNet
Number of Scales	3	4	4	4	(Exact same model as in ImageNet)	
# of Channels for Each Scale	[8,16,32]	[28,56,112,224]	[32,64,128,256]	[80,160,320,640]		
Width Expansion (in the residual block)	5×	5×	5×	5×		
Normalization (# of groups)	GroupNorm(4)	GroupNorm(4)	GroupNorm(4)	GroupNorm(4)		
Weight Normalization	✓	✓	✓	✓		
# of Downsamplings Before Equilirbium Solver	0	0	2	2		
Forward Quasi-Newton Threshold T_f	15	15	22	22	27	27
Backward Quasi-Newton Threshold T_b	18	18	25	25	30	30
Limited-Mem. Broyden's Method Storage Size m	12	12	18	18	18	18
Variational Dropout Rate	0.2	0.25	0.0	0.0	0.03	0.05

B Equilibrium Solving and Convergence Analysis

We extend our discussion on the convergence to equilibrium in Section 3.3 here. First, we briefly introduce the (limited-memory) Broyden's method that we use to perform the root-solving.

B.1 (Limited-memory) Broyden's Method

As our goal is to solve the equation $g_\theta(\mathbf{z}^\star; \mathbf{x}) = f_\theta(\mathbf{z}^\star; \mathbf{x}) - \mathbf{z}^\star = 0$ for the (root) equilibrium point $\mathbf{z}^\star$ as efficiently as possible, an ideal choice would be Newton's method:

$$\mathbf{z}^{[i+1]} = \mathbf{z}^{[i]} - (J_{g_\theta}^{-1}\big|_{\mathbf{z}^{[i]}}) g_\theta(\mathbf{z}^{[i]}; \mathbf{x}); \quad \mathbf{z}^{[0]} = \mathbf{0} \tag{6}$$

However, in practice this involves two major difficulties. First, for a deep network with realistic size, the Jacobians are typically prohibitively large to compute and store. For instance, for a layer converting an input tensor of dimension $32 \times 32 \times 80$ (e.g., height $\times$ width $\times$ channels) to an output of the same shape, the resulting Jacobian will have dimension 81920×81920, which needs 25GB of memory to store. Second, even if we can store this Jacobian, inverting it would be an extremely expensive (cubic complexity) operation.

We therefore use a variant of Broyden's method [6, 5]:

$$\mathbf{z}^{[i+1]} = \mathbf{z}^{[i]} - \alpha \cdot B^{[i]} g_\theta(\mathbf{z}^{[i]}; \mathbf{x}); \quad \mathbf{z}^{[0]} = \mathbf{0} \tag{7}$$

where α is an adjustable step size and $B^{[i]}$ is a *low-rank approximation* to $J_{g_\theta}^{-1}\big|_{\mathbf{z}^{[i]}}$. Notably, we do not need to form the Broyden matrix $B^{[i]}$ explicitly, as we can write it as a sum of low-rank updates:

$$B^{[i+1]} = B^{[0]} + \sum_{k=1}^{i} \mathbf{u}^{[k]} {\mathbf{v}^{[k]}}^\top = B^{[0]} + UV^\top \tag{8}$$

where $\mathbf{u}, \mathbf{v}$ comes from the Sherman-Morrison formula [48]. We initialize the Broyden matrix to $B^{[0]} = -I$. As described in Section 3.2, we further extended Broyden's method with a limited-memory version that stores no more than m low-rank updates $\mathbf{u}$, $\mathbf{v}$ each. Specifically, when the maximum storage memory m is used, we free up memory by discarding the oldest update in U and V (other schemes are also possible).

B.2 Discussions

Runtime. The rate of convergence of MDEQ is directly related to the runtime of MDEQ. Because an MDEQ does not have "layers", a good indicator of computational complexity of MDEQ is the number of root-finding iterations (e.g., each Broyden iteration evalute f_θ exactly once). In practice, we stop the Broyden iterations at some threshold limit (e.g., 22 iterations), which usually does not yield the *exact equilibrium* (see Figure 5 and the discussion below). However, we find these estimates of the equilibria are usually good enough and sufficient for very competitive training of the MDEQ models. Similar observations have also been made in sequence-level DEQs [5].

Convergence on High-resolution Inputs. As we scale MDEQ to higher-resolution inputs, the equilibrium solving process also becomes increasingly challenging. We identify at least two major reasons behind this phenomenon.

1. As the input resolution gets higher, so does the size of the Jacobian of f_θ which we try to approximate via Broyden's method. Therefore, more low-rank updates are expected for the Broyden matrix approximate the Jacobian and solve for the high-dimensional root.
2. Due to the nature of typical vision models, MDEQ employs convolutions with small receptive fields (e.g., the two 3×3 convolutions in f_θ's residual block) on very large inputs. To see how this complicates the equilibrium solving, consider a case where we simply iterate $f_\theta(\cdot; \mathbf{x})$ on $\mathbf{z}$ to reach the equilibrium point (i.e., not using Broyden's method; assuming f_θ is stable). Then we need *at least* as many iterations as required for the stacked f_θ to have a receptive field large enough to cover the entire image. Otherwise, new pixels covered by the larger receptive field will be available for each additional stack of f_θ (which disrupts the equilibrium).

This phenomenon is visualized in Figure 5, where we show equilibrium convergence of MDEQ models on CIFAR-10 (low resolution), ImageNet (medium resolution), and Cityscapes (high resolution) images by measuring the change of residual $\frac{\|\mathbf{z}^{[i+1]} - \mathbf{z}^{[i]}\|}{\|\mathbf{z}^{[i]}\|}$ with respect to calls to f_θ. As with our experimental setting in Section 4, we initialize the Cityscapes MDEQ with the weights pretrained on ImageNet classification (pink line in Figure 5c). In particular, we observe that more Broyden iterations were required to reach the fixed point as the images get larger. For example, whereas MDEQ typically finds the equilibria with a good level of accuracy within 30 steps on CIFAR-10 images, over 100 steps are used on Cityscapes images.

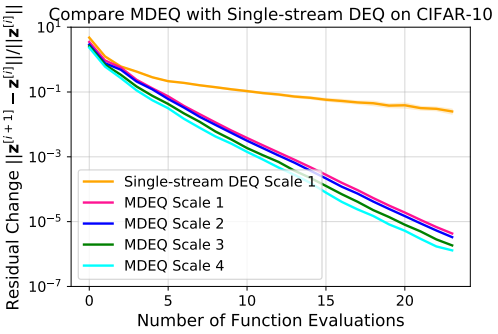


Figure 6: Comparing MDEQ with single-stream DEQ on CIFAR-10. All resolutions of MDEQ converge *simultaneously* and in a much stabler way than the single-scale DEQ model. Larger scale index means higher resolution (e.g., "scale 1" is the highest scale).

Moreover, in all three cases, Broyden's method (blue lines in Figure 5) converges to the fixed point in a more stable and efficient manner than simply iterating f_θ (yellow lines), which often converges poorly or does not converge at all.

We find that the simultaneous multiscale fusion also effectively stabilizes the equilibrium convergence of an MDEQ. Figure 6 visualizes the convergence of all equilibrium streams (i.e., $\frac{\|\mathbf{z}_k^{[i+1]} - \mathbf{z}_k^{[i]}\|}{\|\mathbf{z}_k^{[i]}\|}$ for resolution k) in an MDEQ that is applied on CIFAR-10. For comparison, we also visualize the convergence of a single-stream DEQ [5] that maintains only the highest-resolution stream (i.e., 32×32). Specifically, from Figure 6 one can observe that: 1) all MDEQ resolution streams indeed converge to their equilibria in parallel; 2) lower-resolution streams converge faster than higher-resolution streams; and 3) high-resolution convergence is much faster in multiscale setting (pink line) than in the single-stream setting (orange line).

We hypothesize that Broyden's method and the multiscale fusion help with the equilibrium convergence because both techniques provide a faster way to expand the receptive field of f_θ (than simply stacking it). For Broyden's method (see Eq. (7)), the Broyden matrix $B^{[i]}$ is a full matrix that mixes all locations of the feature map (which is represented by $g_\theta(\mathbf{z}^{[i]}; x)$); whereas typical convolutional filters only mix the signals locally. On the other hand, multiscale up- and downsamplings broaden the effective receptive field on the high-resolution stream by direct interpolation from lower-resolution feature maps.

C Qualitative Segmentation Results on Cityscapes

We demonstrate in Figure 7 some examples of the segmentation results of the MDEQ-large model (see Table 3) on Cityscapes (`val`) images (of resolution 2048×1024).

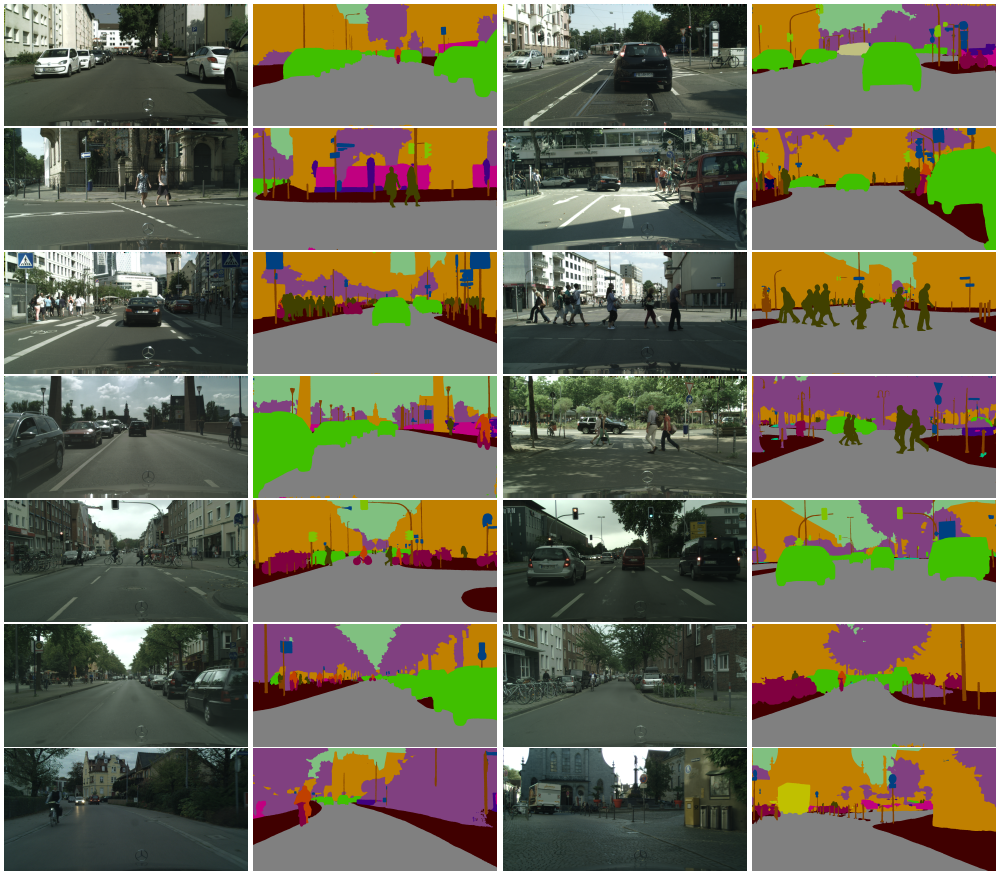


Figure 7: Examples of MDEQ-large segmentation results on the Cityscapes dataset.