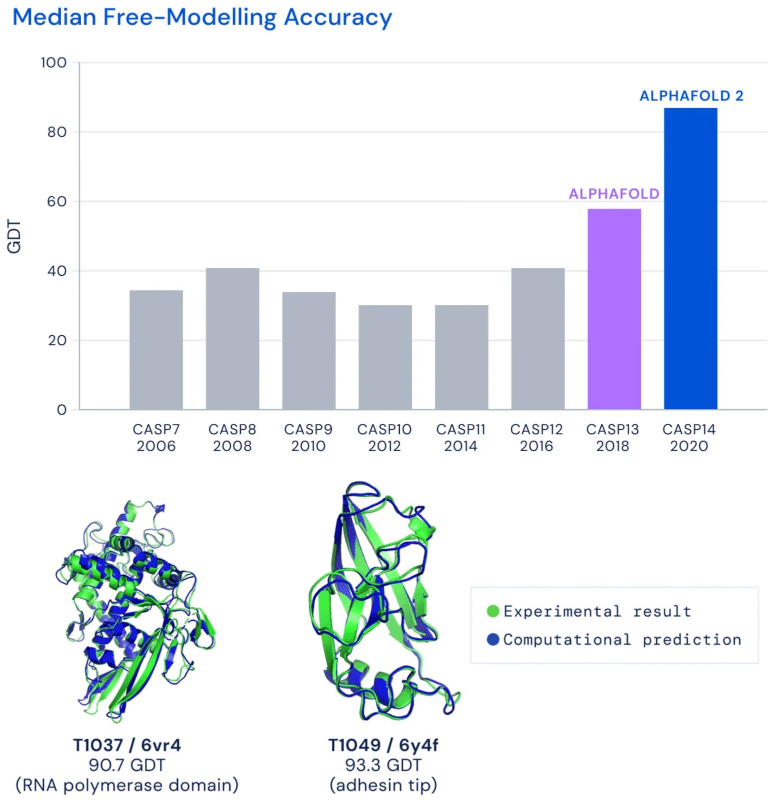
Deep neural networks via monotone operators

J. Zico Kolter
Carnegie Mellon University and Bosch Center for AI

Work with Shaojie Bai, Ezra Winston, Vladlen Koltun (Apple)



The deep learning revolution (recent examples)



AlphaFold: Jumper et al., 2021

Explaining a joke

Prompt

```
Explain this joke:

Joke: Did you see that Google just hired an eloquent whale for
their TPU team? It showed them how to communicate between two
different pods!
```

Model Response

```
Prediction: TPUs are a type of computer chip that Google uses
for deep learning. A "pod" is a group of TPUs. A "pod" is also a
group of whales. The joke is that the whale is able to
communicate between two groups of whales, but the speaker is
pretending that the whale is able to communicate between two
groups of TPUs.
```

PaLM explains an original joke with two-shot prompts.

PaLM: Chowdhery et al., 2022

vibrant portrait painting of Salvador Dalí with a robotic half face

a shiba inu wearing a beret and black turtleneck

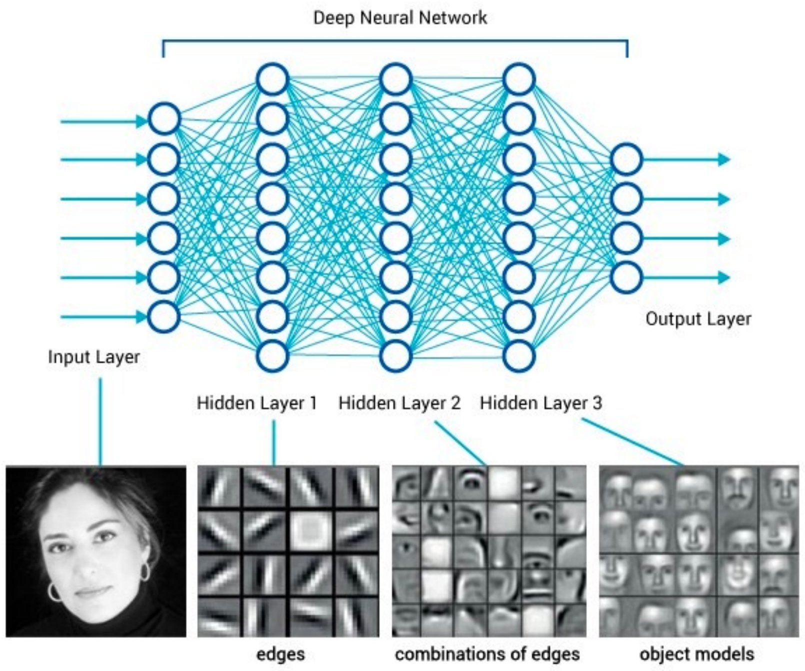
an espresso machine that makes coffee from human souls, artstation

panda mad scientist mixing sparkling chemicals, artstation

DALL-E 2: Ramesh et al., 2022

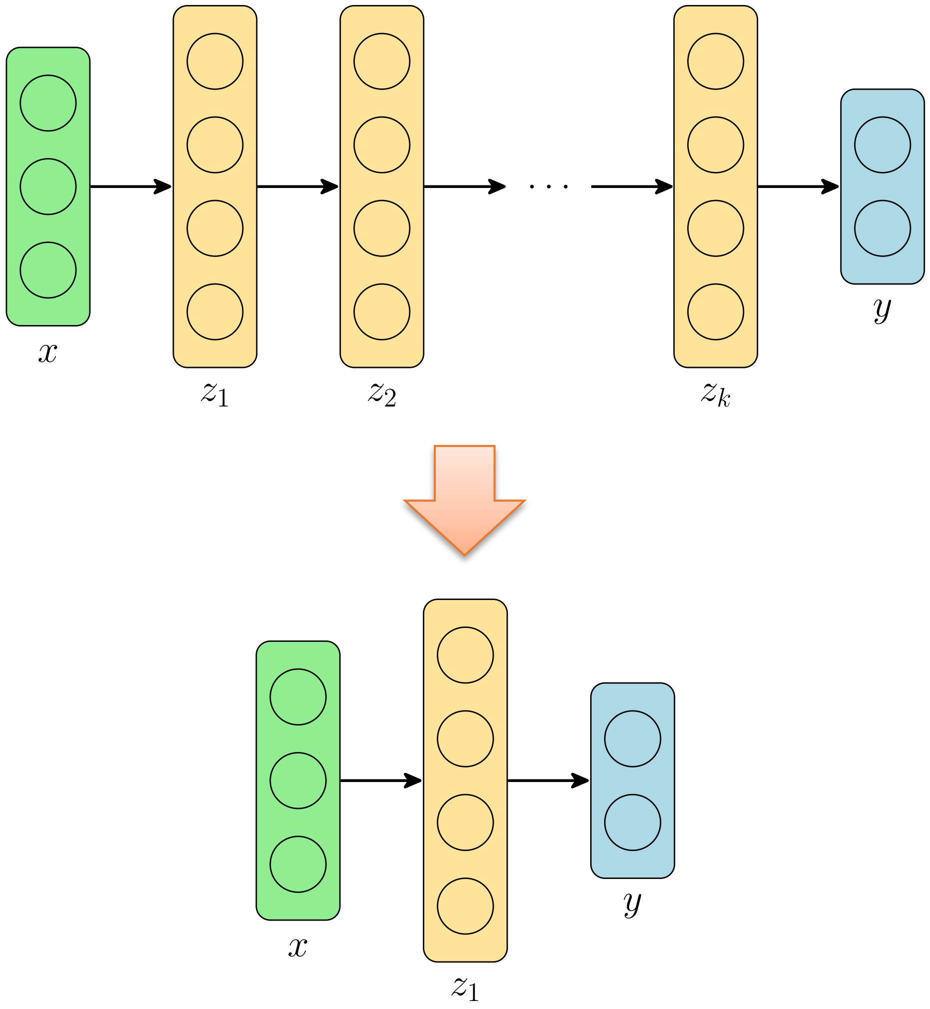
Deep Learning

The story we all tell: deep learning algorithms build hierarchical models of input data, where the earlier layers create "simple" features and layer layers create high-level abstractions of the data



[Lee, Grosse, Ranganath and Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations", ICML 2009]

This talk



We can replace traditional depth in deep networks with a single (implicit) layer

- Simpler architectural design
- Vastly reduced memory requirements
- Matches or exceeds accuracy of comparable fixed-depth networks

Offers a new perspective on what deep networks are “really” computing

Outline

Deep equilibrium models

Monotone equilibrium networks

Final thoughts

Outline

Deep equilibrium models

Monotone equilibrium networks

Final thoughts

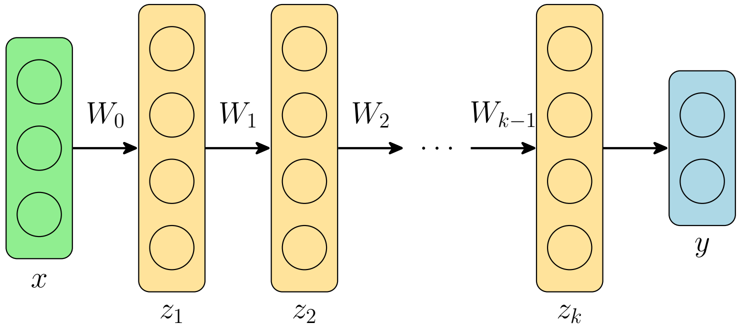
[Bai, Koltun, Kolter "Deep Equilibrium Models", NeurIPS 2019]

[Bai, Koltun, Kolter "Multiscale Deep Equilibrium Models", NeurIPS 2020]

From deep networks to DEQs

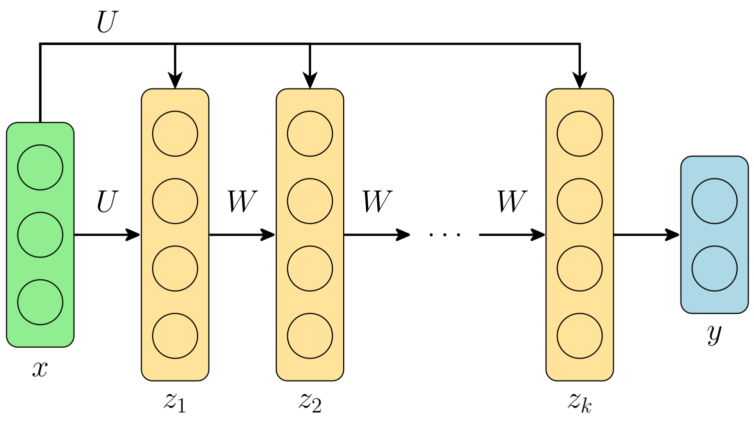
Traditional network

$$z_{i+1} = \sigma(W_i z_i + b_i)$$



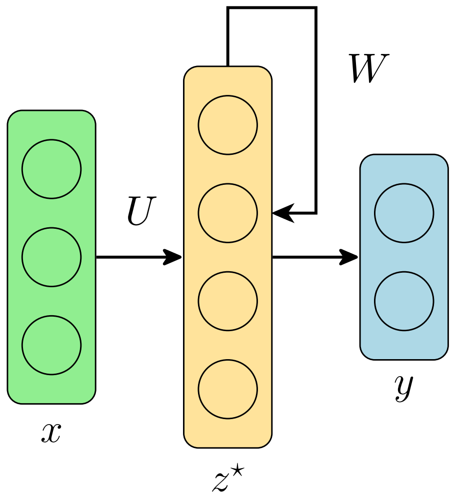
Weight-tied, input-injected network

$$z_{i+1} = \sigma(W z_i + Ux + b)$$



Deep Equilibrium (DEQ) model

$$z^\star = \sigma(W z^\star + Ux + b)$$

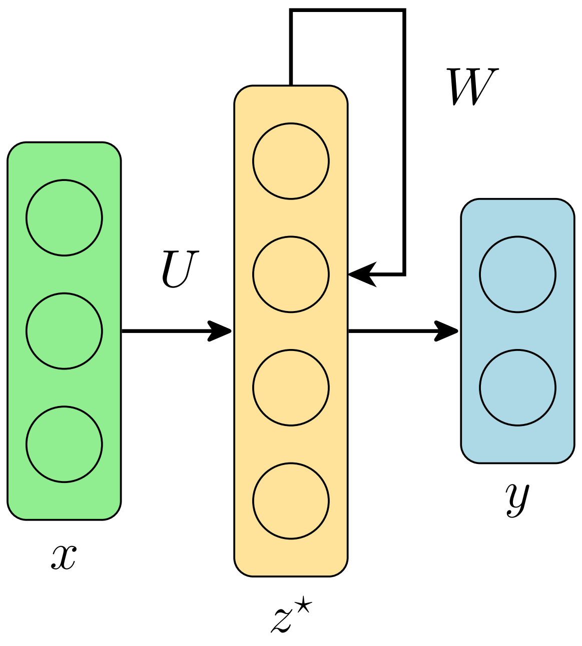


Long history of related work

Fixed point iterations and other implicit layers have a long history within deep learning

- Roots in recurrent backpropagation [Almaida, 1987; Pineda 1990]
- Recent advances in recurrent backprop [Liao, Xiong, Fetaya, Zhang, Yoon, Pitkow, Urtasun, Zemel 2017]
- Similarities to Neural ODEs [Chen, Rubanova, Bettencourt, Duvenaud, 2018]
- Modern usage in Trellis Networks [Bai, Kolter, Koltun, 2019], Universal Transformers [Dehghani, Gouws, Vinyals, Uszkoreit, Kaiser, 2018]

$$z^\star = \sigma(W z^\star + Ux + b)$$



DEQs in theory: “One layer is all you need”

Theorem 1: A single-layer DEQ can represent any feedforward deep network

Proof: “Stack” all hidden layers together, and let f be “shifted” application of all layers (**important note:** just for theory, *not* what is done in practice)

$$\begin{aligned} z_1 &= \sigma(W_0 x + b_0) \\ z_2 &= \sigma(W_1 z_1 + b_1) \\ z_3 &= \sigma(W_2 z_2 + b_2) \end{aligned} \iff \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \sigma\left(\begin{bmatrix} 0 & 0 & 0 \\ W_1 & 0 & 0 \\ 0 & W_2 & 0 \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} + \begin{bmatrix} W_0 \\ 0 \\ 0 \end{bmatrix} x + \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}\right)$$

$$\bar{z}^\star = \sigma(\bar{W}\bar{z}^\star + \bar{U}x + \bar{b}) \qquad \blacksquare$$

Note: in practice, use $z^\star = \sigma(Wz^\star + Ux + b)$ with *dense* W, U, b (of larger size, to make # of parameters equal across networks)

DEQs in Theory: "One layer is all you need"

Theorem 2: A single-layer DEQ can represent any multi-layer DEQ

Proof: Two equilibrium models can again be represented as a single equilibrium model with layer again "stacked" together

$$\begin{aligned} z_1^\star &= \sigma(W_1 z_1^\star + U_1 x + b_1) \\ z_2^\star &= \sigma(W_2 z_2^\star + U_2 z_1^\star + b_2) \end{aligned} \quad \Longleftrightarrow \quad \begin{bmatrix} z_1^\star \\ z_2^\star \end{bmatrix} = \sigma\left(\begin{bmatrix} W_1 & 0 \\ U_2 & W_2 \end{bmatrix} \begin{bmatrix} z_1^\star \\ z_2^\star \end{bmatrix} + \begin{bmatrix} U_1 \\ 0 \end{bmatrix} x + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \right)$$

$\blacksquare$

But doesn't address … existence of equilibrium point? uniqueness? stability?

Implementing DEQs

In practice, use a single "cell" rather than a literal single layer

$$z^\star = f(z^\star, x)$$

E.g., a transformer block, residual block, etc

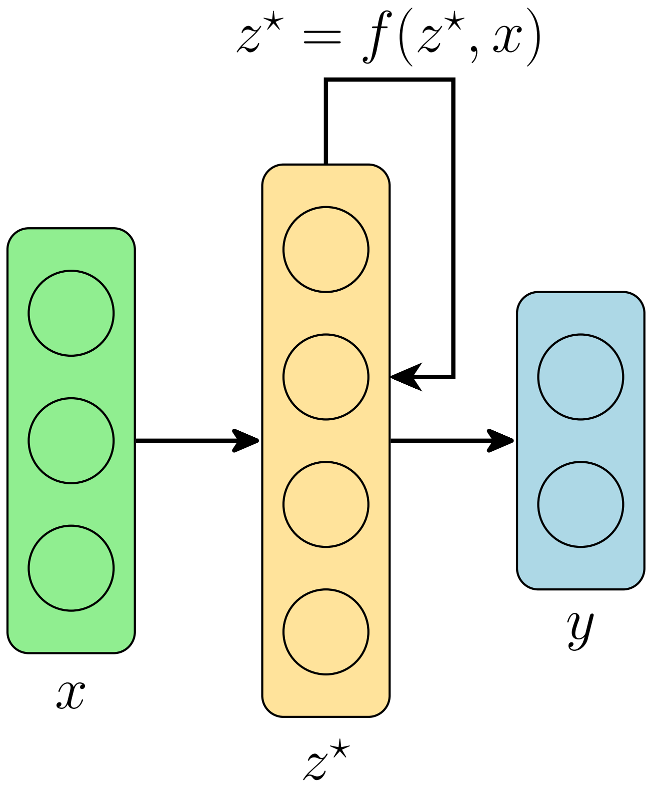
Forward pass: Given x, compute equilibrium point $z^\star$

How?

Backward pass: Compute the gradient through the equilibrium point

$$\frac{\partial \ell}{\partial \theta} = \frac{\partial \ell}{\partial z^\star} \cdot \frac{\partial z^\star}{\partial \theta}$$

How?



The DEQ forward pass

How do we compute the equilibrium point $z^\star = f(z^\star, x)$?

We *could* simply perform the forward iteration: initialize $z_0 = 0$ and repeat

$$z_{t+1} = f(z_t, x)$$

… but this is quite slow to converge in practice

Instead, can use (any) accelerated root-finding technique, e.g. Anderson Acceleration [Anderson, 1965]

Anderson(z_0, f, m, T):
For $t = 1, \ldots, T$
1. Solve the optimization:
$$\min_{1^T\alpha=1} \|\textstyle\sum r_{t-i}\alpha_i\|^2$$
where $r_t = f(z_t, x) - z_t$
2. Update:
$$z_{t+1} = \sum_i \alpha_i f(z_{t-i}, x)$$

Next iterate is a *linear combination* of previous function evals

How to train your DEQ

Compute gradients analytically via *implicit function theorem*

$$z^\star(\theta) = f(z^\star(\theta), x)$$

$$\Longrightarrow \frac{\partial z^\star(\theta)}{\partial \theta} = \frac{\partial f(z^\star(\theta), x)}{\partial \theta}$$

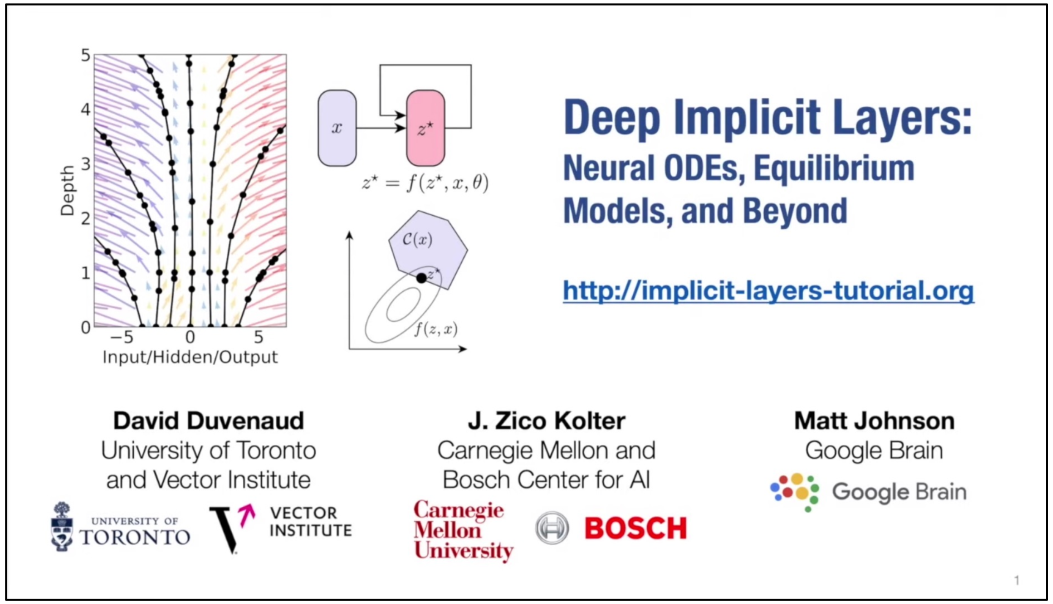
$$\Longrightarrow \frac{\partial z^\star(\theta)}{\partial \theta} = \underbrace{\frac{\partial f(z^\star, x)}{\partial \theta}} + \underbrace{\frac{\partial f(z^\star, x)}{\partial z^\star}} \cdot \frac{\partial z^\star(\theta)}{\partial \theta}$$

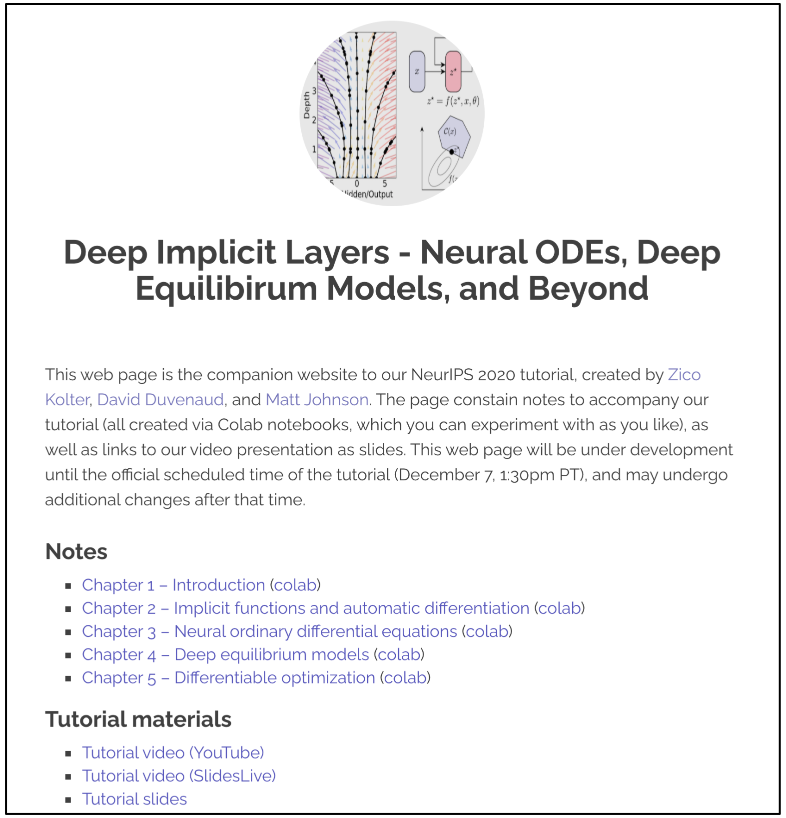
Known via "ordinary" automatic differentiation

$$\Longrightarrow \frac{\partial z^\star(\theta)}{\partial \theta} = \left(I - \frac{\partial f(z^\star, x)}{\partial z^\star}\right)^{-1} \frac{\partial f(z^\star, x)}{\partial \theta}$$

Use Anderson acceleration to also solve the backward pass indirectly, without directly computing/storing the inverse

More information on implicit layers




Deep Implicit Layers - Neural ODEs, Deep Equilibirum Models, and Beyond

This web page is the companion website to our NeurIPS 2020 tutorial, created by Zico Kolter, David Duvenaud, and Matt Johnson. The page constain notes to accompany our tutorial (all created via Colab notebooks, which you can experiment with as you like), as well as links to our video presentation as slides. This web page will be under development until the official scheduled time of the tutorial (December 7, 1:30pm PT), and may undergo additional changes after that time.

Notes

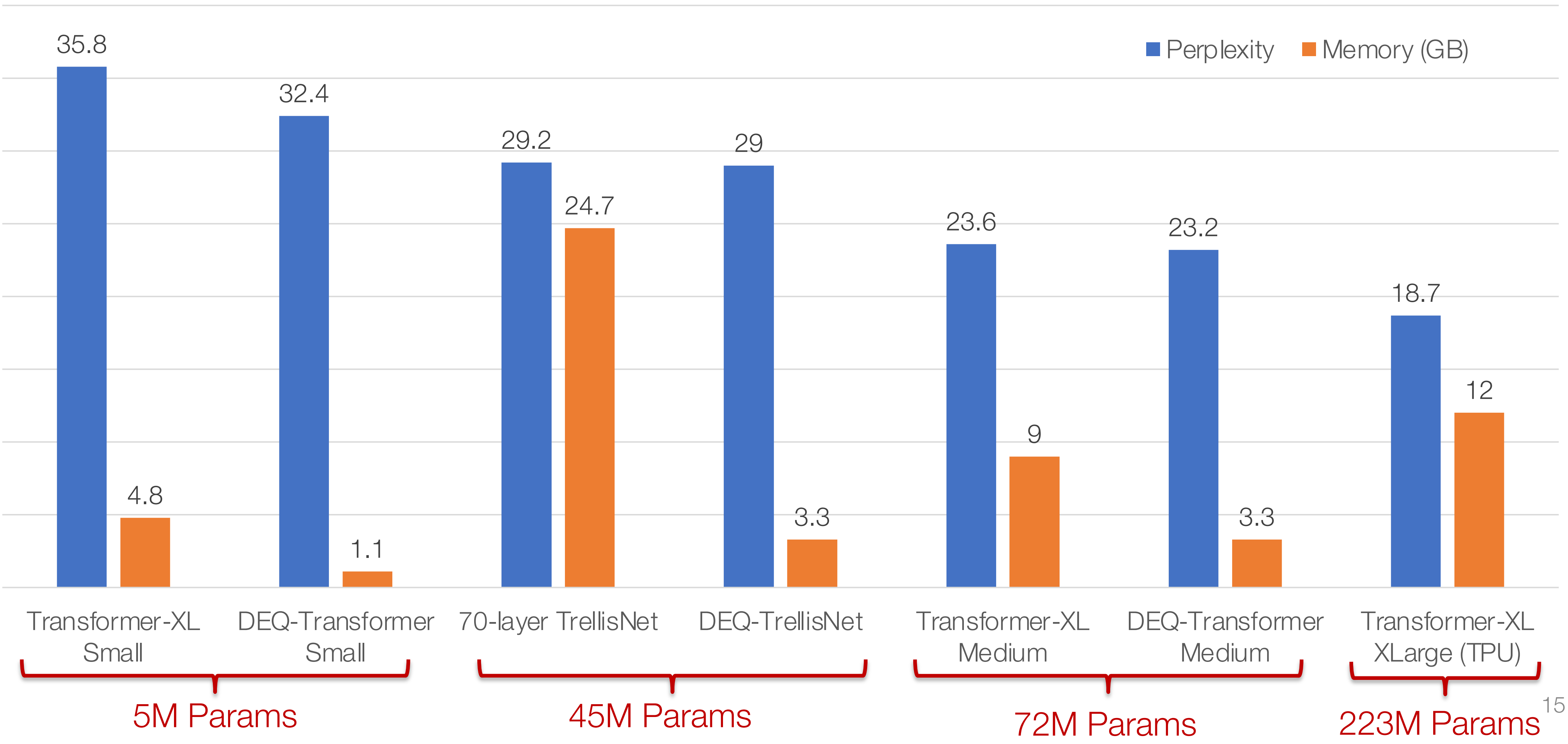
- Chapter 1 – Introduction (colab)
- Chapter 2 – Implicit functions and automatic differentiation (colab)
- Chapter 3 – Neural ordinary differential equations (colab)
- Chapter 4 – Deep equilibrium models (colab)
- Chapter 5 – Differentiable optimization (colab)

Tutorial materials

- Tutorial video (YouTube)
- Tutorial video (SlidesLive)
- Tutorial slides

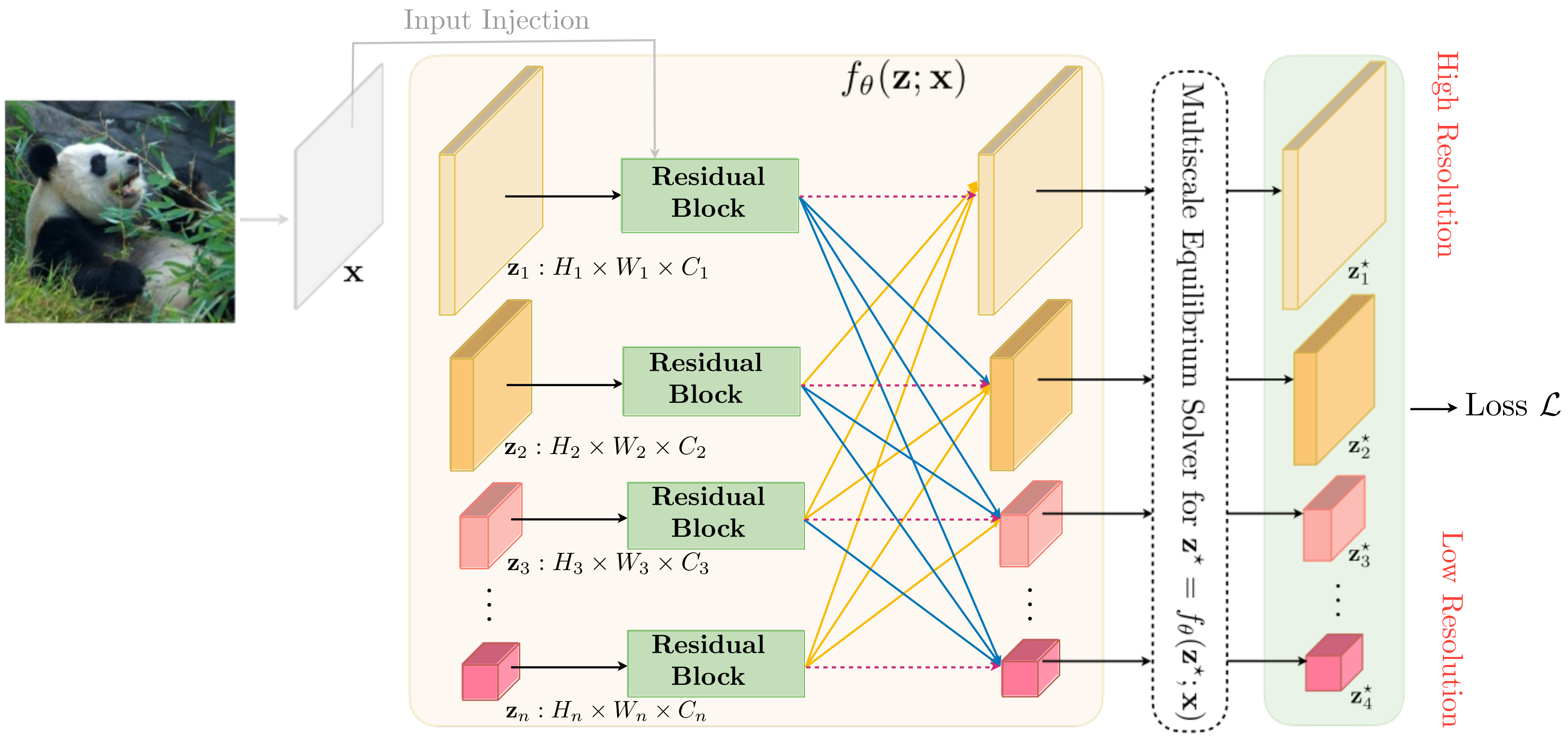
http://implicit-layers-tutorial.org

Language modeling: WikiText-103



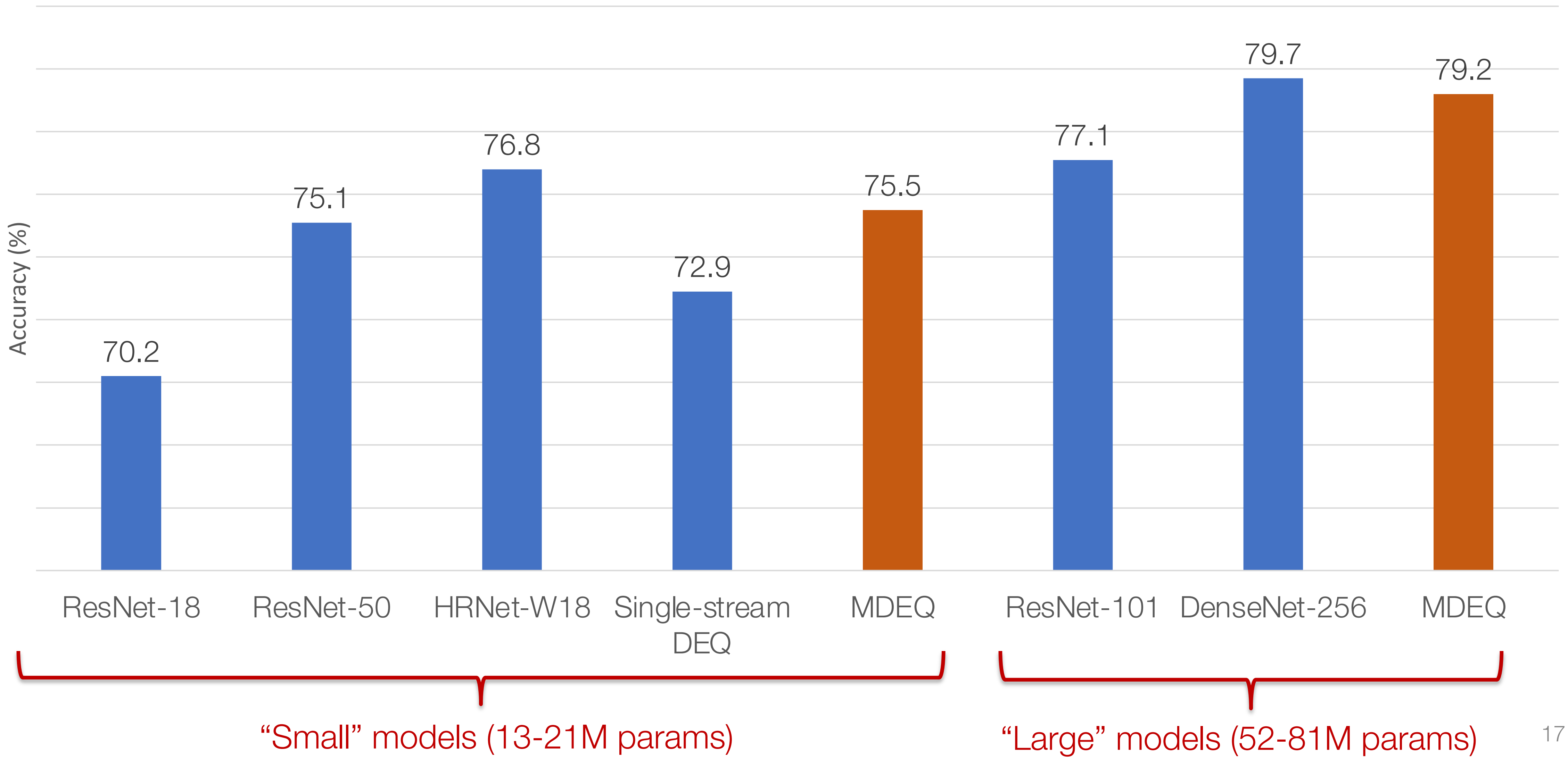
Multiscale deep equilibrium models

For visual networks, layers also serve as multi-resolution representations $\Longrightarrow$ *multiscale DEQ* (MDEQ) maintains multiple spatial resolutions simultaneously

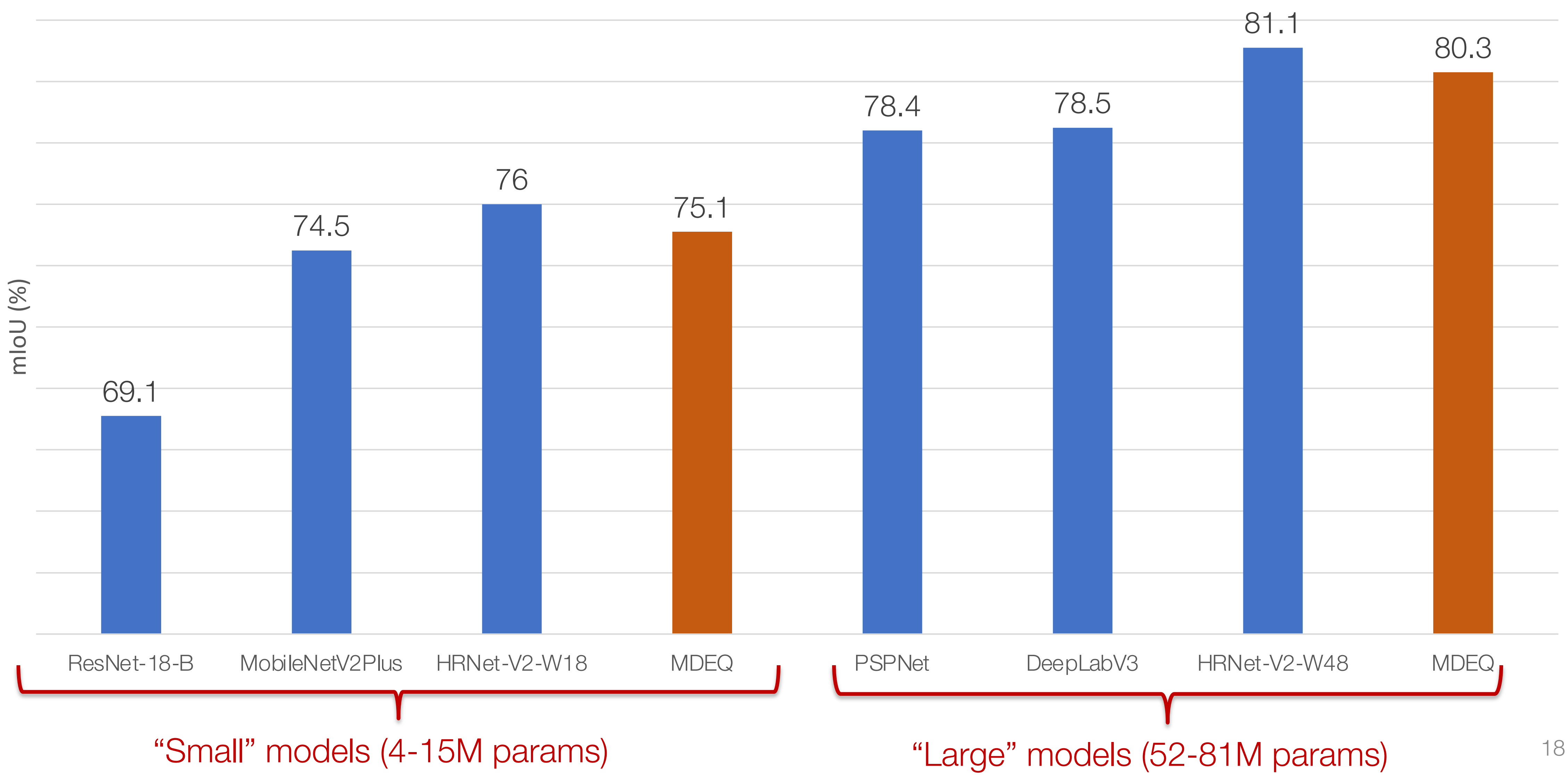


[Bai, Koltun, Kolter "Multiscale Deep Equilibrium Models", NeurIPS 2020]

ImageNet Top-1 Accuracy



Citiscapes mIoU



Visualization of Segmentation

Outline

Deep equilibrium models

Monotone equilibrium networks

Final thoughts

[Winston, Kolter, “Monotone operator equilibrium networks”, NeurIPS 2020]

Theoretical/algorithmic challenges for DEQs

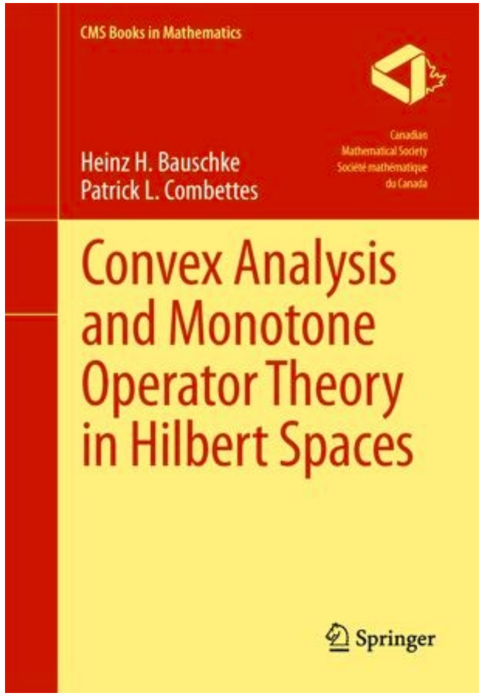
What can we say about the fixed point?

$$z^\star = \sigma(Wz^\star + Ux + b)$$

In general, a non-linear dynamical system, can be difficult to establish existence, uniqueness, and stability of the fixed point iteration

- Unlike e.g., Neural ODEs where existence/uniqueness is guaranteed by Picard's theorem

Monotone operator theory provides a useful framework for analyzing these questions



Appl. Comput. Math., V.15, N.1, 2016, pp.3-43

A PRIMER ON MONOTONE OPERATOR METHODS
SURVEY

ERNEST K. RYU[1], STEPHEN BOYD[2]

Abstract. This tutorial paper presents the basic notation and results of monotone operators and operator splitting methods, with a focus on convex optimization. A very wide variety of algorithms, ranging from classical to recently developed, can be derived in a uniform way. The approach is to pose the original problem to be solved as one of finding a zero of an appropriate monotone operator; this problem in turn is then posed as one of finding a fixed point of a related operator, which is done using the fixed point iteration. A few basic convergence results then tell us conditions under which the method converges, and, in some cases, how fast. This approach can be traced back to the 1960s and 1970s, and is still an active area of research. This primer is a self-contained gentle introduction to the topic.

Keywords: Monotone Operators, Convex Optimization, Splitting Methods, Fixed Point Iteration.

AMS Subject Classification: 47H05, 47H09, 47H10, 47N10, 65K05, 90C25.

1. Introduction

In the field of convex optimization, there are a myriad of seemingly disparate algorithms each with its specific setting and convergence properties. It is possible to understand, derive, and analyze many of these methods in a unified manner, using the abstraction of monotone operators and a single approach. First, the problem at hand is expressed as finding a zero of a monotone operator. This problem is in turn transformed into finding a fixed point of a related function. The fixed point is then found by the fixed point iteration, yielding an algorithm for the original problem. This single approach yields many different algorithms, with different convergence conditions, depending on how the first and second steps are done (*i.e.*, the selection of the monotone operator and fixed point function). It recovers many classical and modern algorithms along with conditions under which they converge.

The idea of this basic approach is not new, and several surveys based on it have already been written, e.g., by [7, 33, 34, 36, 44]. Several surveys that rigorously develop the theory behind monotone operators also have been written, e.g., by [3, 7, 17, 103].

In fact, the ideas can be further traced back to the 1960s and 1970s. In the 1960s, the notion of monotone operators was first formulated and studied [71,87,88]. Much of the initial work was done in the context of functional analysis and partial differential equations [24–26], but it was soon noticed that the theory is relevant to convex functions and convex optimization [71,90,110]. In the 1970s, iterative algorithms constructed from monotone operators and fixed point functions were itroduced [81,85,86,117]. Since then, this field has grown considerably and is still an active area of research

Key result

Theorem: Consider single-layer DEQ

$$z^\star = \sigma(Wz^\star + Ux + b)$$

Then there exists a unique fixed point of the system provided that

1. We have $2(1-m)I \succcurlyeq W + W^T$ (in positive definite sense)
2. The nonlinearity is given by $\sigma = \mathrm{prox}_f$ for some convex f

And resulting network has Lipschitz constant $\|U\|_2/m$ (no dependence on $\|W\|_2$)

The result comes from the fact that the fixed point can be viewed as the solution to a monotone operator splitting problem

Proof sketch for simpler case

Proof: Consider the iteration

$$z^\star = \sigma(W z^\star + Ux + b)$$

with 1) $(1-m)I \succ W$ (W symmetric), and 2) $\sigma(z) = [z]_+ \equiv \mathrm{ReLU}(z)$

And consider the (strictly convex) optimization problem

$$\min_{z \geq 0} \; \frac{1}{2} z^T (I - W) z - z^T (Ux + b)$$

A projected gradient descent step for the above problem is given by

$$z^{t+1} = [z^t - \alpha(I - W) z^t + \alpha(Ux + b)]_+$$

which is equal to above update for $\alpha = 1$ (and fixed point at optimum). ■

Monotone operator equilibrium networks

Are prox operators good nonlinearities?

- Yes [Combettes and Pesquet, 2018; Bibi, Ghanem, Koltun, Ranftl, 2019]; e.g. $\mathrm{ReLU} = \mathrm{prox}_{I\{x>0\}}$, $\tanh \approx \mathrm{prox}_{\log(1+x^2)}$, many others

Is it realistic to assume/require that $2I \succ W + W^T$?

- No: will be true at most commonly-used random initializations, but training will quickly cause this condition to be violated
- The trick is to parameterize the network in a way that enforces this condition
$$W = (1-m)I \ - F^T F + G^T - G$$
for linear operators F, G (which can be e.g., convolutions)
- In practice, performs worse than “normal” DEQs, but provides a useful theoretical setting for analyzing equilibrium models

Advantages of monotone operator formulation

1. (What I've stated already) By parameterizing the DEQ in the manner, we can guarantee that there exists a unique fixed point $z^\star = \sigma(Wz^\star + Ux + b)$

2. We can use more involved operator splitting approaches (e.g., Forward-Backward-Forward, Peaceman-Rachford) to find the fixed point $z^\star$, which will be guaranteed to converge (linearly)
   - Note: more complex operator splitting methods like Peaceman-Rachford require that we invert the operator $(I - W)$, which is challenging for e.g. convolutional networks, but can be accomplished via the FFT

3. Backward pass, via implicit function theorem *also* has operator splitting formulation, can use this to also derive efficient backward pass methods

Initial study: CIFAR10

"Simple" multiscale network:

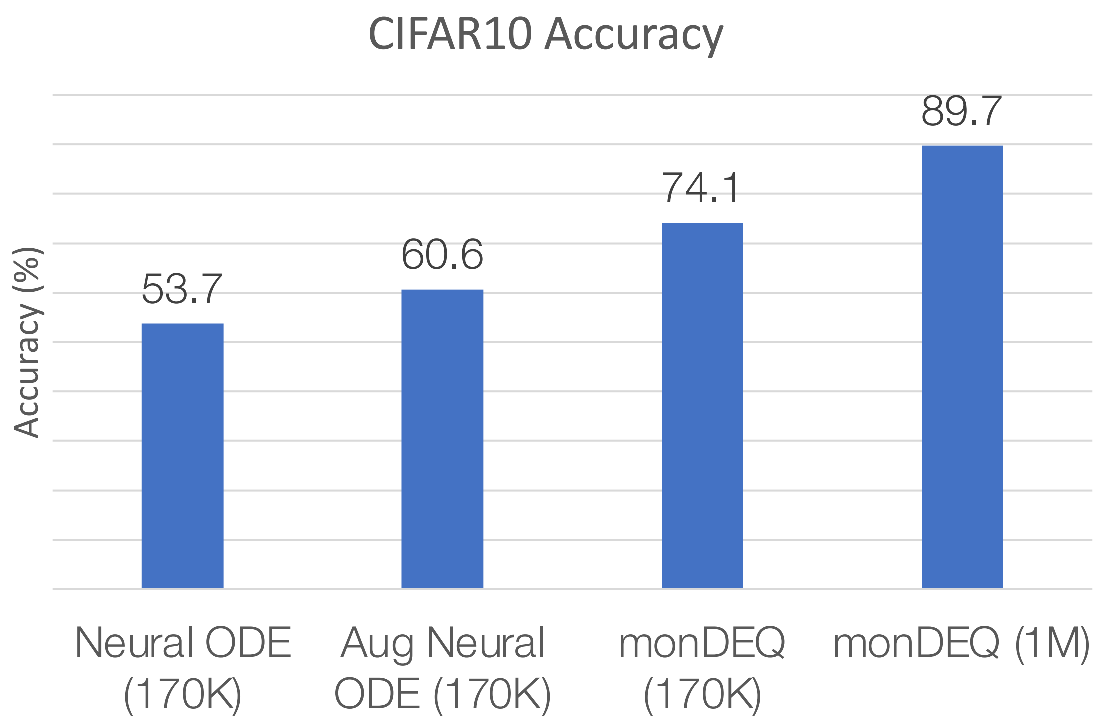
$$W = \begin{bmatrix} W_{11} & 0 & 0 \\ W_{21} & W_{22} & 0 \\ 0 & W_{32} & W_{33} \end{bmatrix}$$

Downsampling convolutions

Square convolutions

Similar tricks to enforce PSD form of $I - W$, and compute inverse via FFT

Fixed point computation using Peaceman-Rachford algorithm



Outperforms all implicit methods with guaranteed fixed points

Additional points on monotone DEQs

We can bound the Lipschitz constant of monDEQ models by $\|U\|_2/m$, which allows us to, e.g., build networks that are more robust to adversarial pertrubations

Can use similar techniques to bound Lipchitz constant w.r.t to *weights* of the network W, use this to obtain generalization bounds on network

- [Pabbaraju, Winston, Kolter, "Estimating Lipschitz constants of monotone deep equilibrium models", ICLR 2021]

Can also relate fixed point iteration to mean-field inference in *Boltzmann machines* (multi-layer Markov random fields), provide sufficient conditions for *global* conference of mean-field inference

- [Feng, Winston, Kolter, "Monotone deep Boltzmann machines", under review 2022]

Outline

Deep equilibrium models

Monotone equilibrium networks

Final thoughts

Final thoughts

The output of deep networks is often (rightly?) viewed as a tangle of arbitrary operations, with millions of parameters

We can alternatively formulate them as solutions to (simpler) conditions, i.e. equilibrium states or solutions to monotone operator splitting problems

Offers insight into the nature of deep networks, and works as well as traditional deep networks (for non-monotone DEQ case)

A return of shallow (implicit, structured) learning?

Papers and code at:
http://zicokolter.com

Also see:
http://implicit-layers-tutorial.org