Deep Equilibrium Models

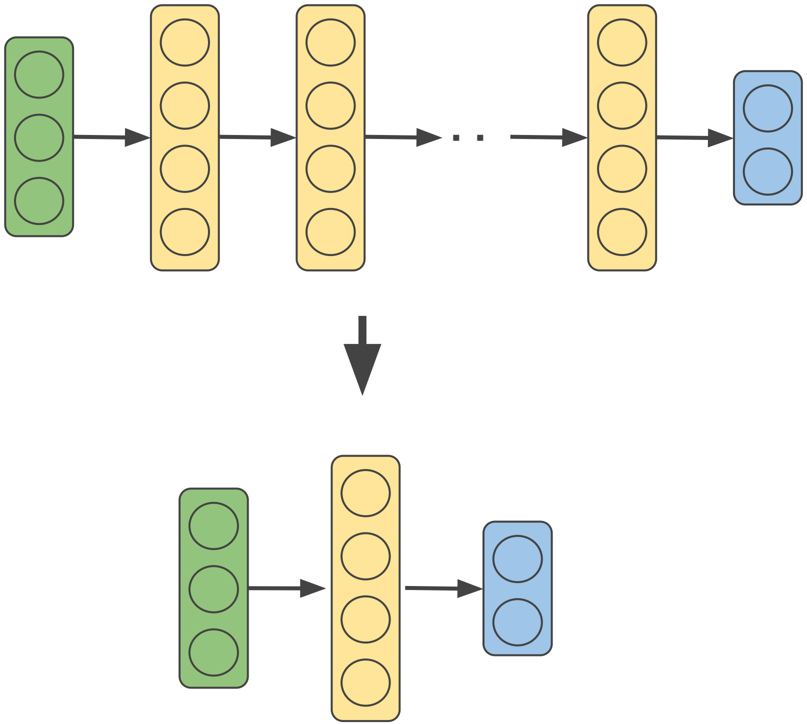
•••

authors:
Shaojie Bai, J. Zico Kolter, Vladlen Koltun

presented by:
Matthias Otth

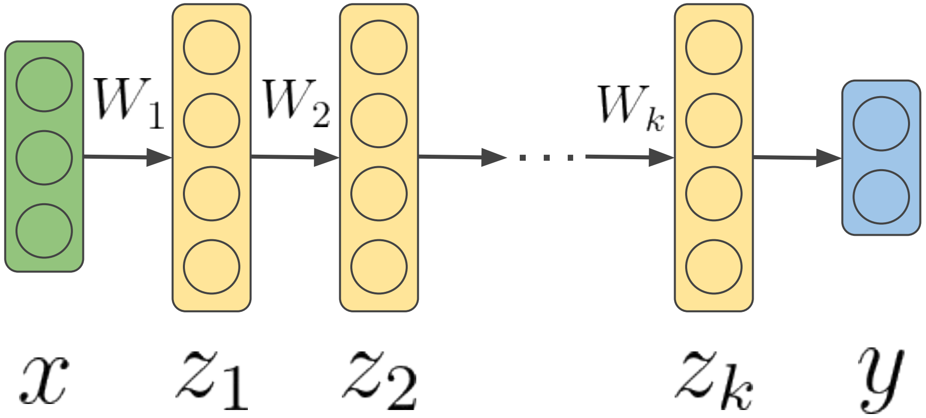
Overview

- Reinterpret deep NN
- Same performance with less memory consumption



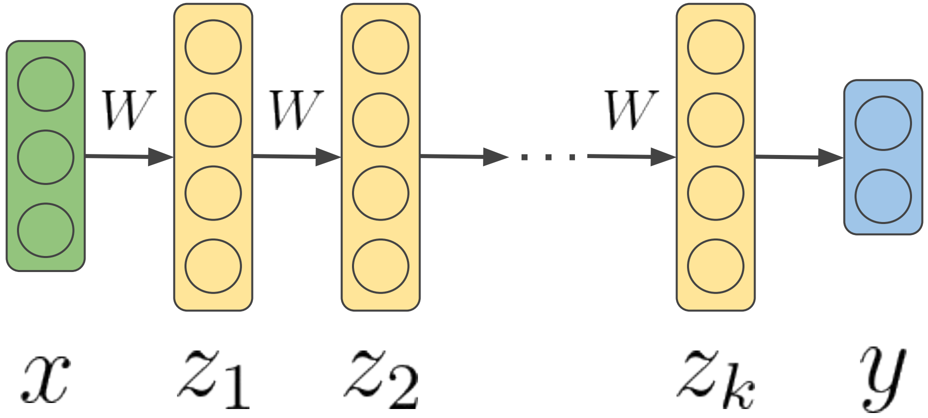
Graphics inspired by [2]: http://implicit-layers-tutorial.org/

Classical deep feedforward NN



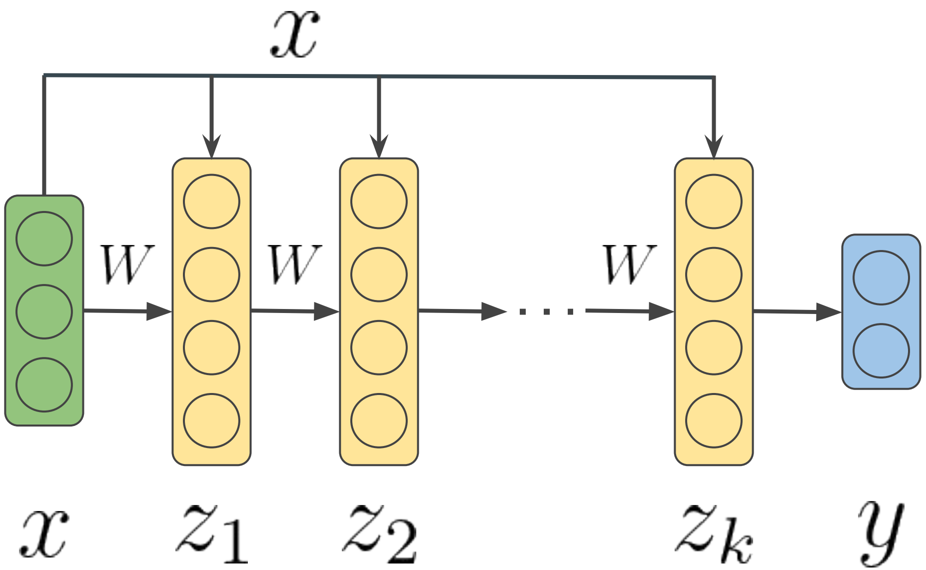
Graphics inspired by [2]: http://implicit-layers-tutorial.org/

Weight-tied Network



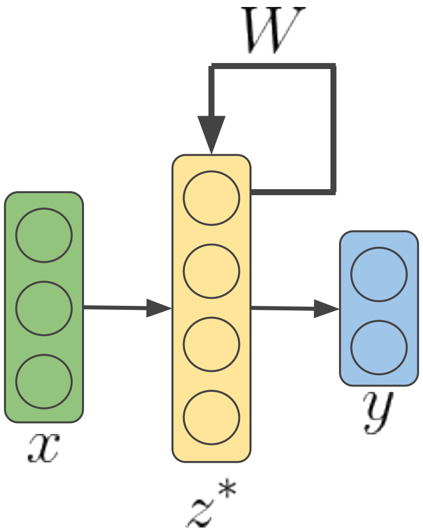
Graphics inspired by [2]: http://implicit-layers-tutorial.org/

Weight-tied, input-injected Network



Graphics inspired by [2]: http://implicit-layers-tutorial.org/

Infinite depth network



Graphics inspired by [2]: http://implicit-layers-tutorial.org/

Equilibrium formulation

- Almost any non-linear function:

$$z^{[i+1]} = f_\theta(z^{[i]}; x)$$

Equilibrium formulation

- Almost any non-linear function:

$$z^{[i+1]} = f_\theta(z^{[i]}; x)$$

- Assume equilibrium point exists:

$$\lim_{i \to \infty} z^{[i]} = z^* = f(z^*; x)$$

Equilibrium formulation

- Almost any non-linear function:

$$z^{[i+1]} = f_\theta(z^{[i]}; x)$$

- Assume equilibrium point exists:

$$\lim_{i \to \infty} z^{[i]} = z^* = f(z^*; x)$$

- Reformulate as root finding problem:

$$g_\theta(z^*; x) = f_\theta(z^*; x) - z^* = 0$$

Equilibrium formulation

- Almost any non-linear function:

$$z^{[i+1]} = f_\theta(z^{[i]}; x)$$

- Assume equilibrium point exists:

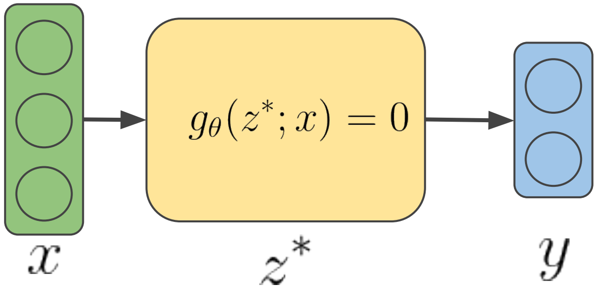
$$\lim_{i \to \infty} z^{[i]} = z^* = f(z^*; x)$$

- Reformulate as root finding problem:

$$g_\theta(z^*; x) = f_\theta(z^*; x) - z^* = 0$$

- Gives rise to a single (implicit) layer model

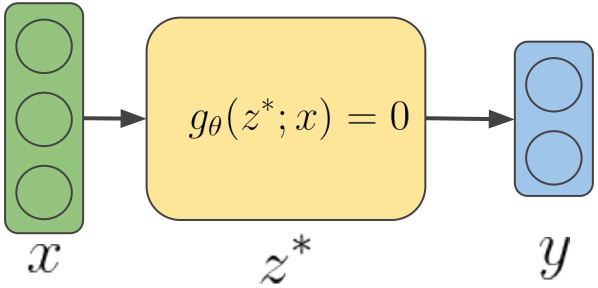
Implicit layer formulation



Graphics inspired by [2]: http://implicit-layers-tutorial.org/

DEQ

- Equivalent to infinite-depth network!
- Different interpretation of deep networks
- We can backpropagate through equilibrium point: O(1) memory



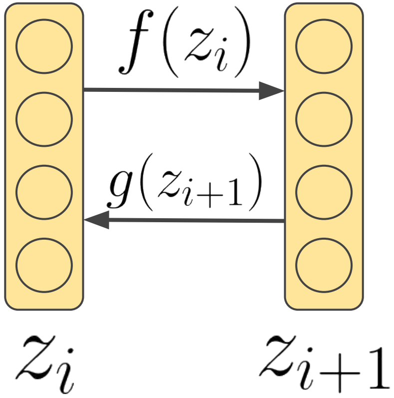
Graphics inspired by [2]: http://implicit-layers-tutorial.org/

Previous Work

Previous work: Implicit Layers

- Applied to small scales
- Very specific models and tasks

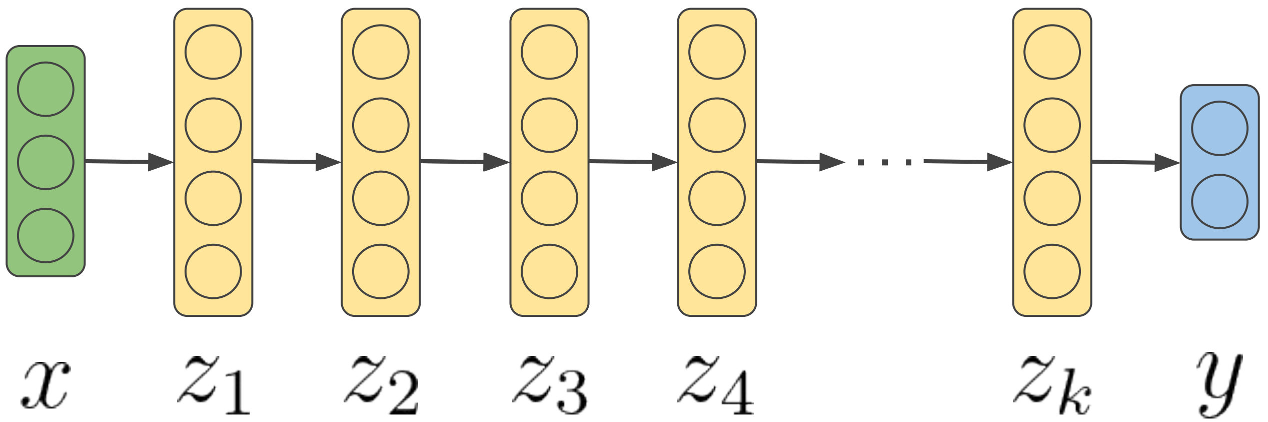
Previous work: Reversible Networks



- O(1) memory consumption
- Strong restriction in model architecture

Papers: Gomez et al. [2], MacKay et. al [3]

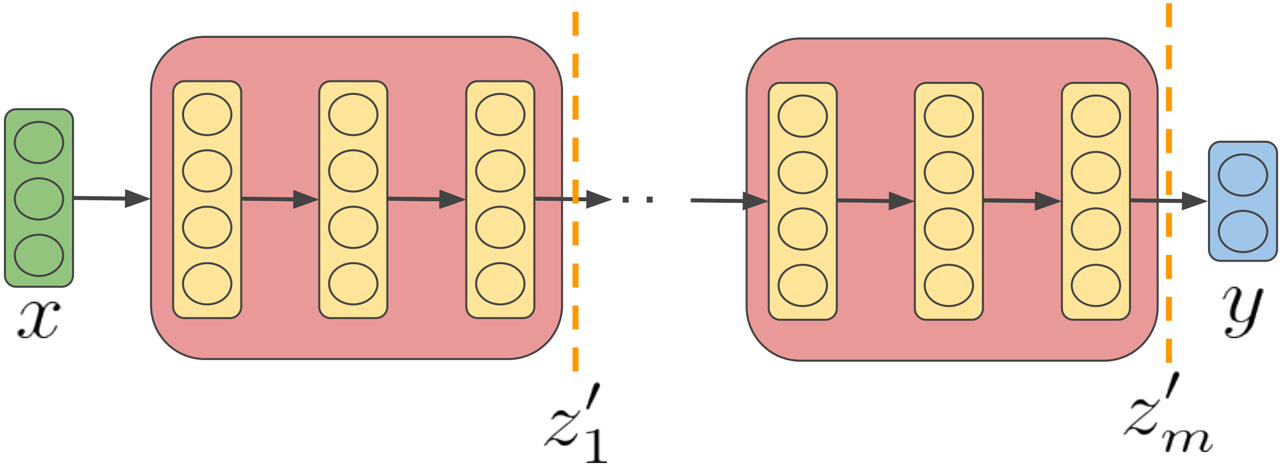
Previous work: Gradient Checkpointing



Paper: Chen et al. [5]

Previous work: Gradient Checkpointing

Step 1: High-level backpropagation

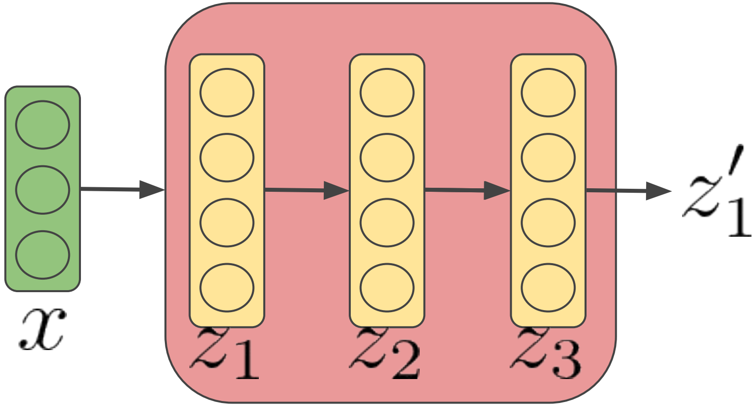


➔ Can calculate $\frac{\partial \mathcal{L}(y, y')}{\partial z_i'}$ in O(m) memory

Paper: Chen et al. [5]

Previous work: Gradient Checkpointing

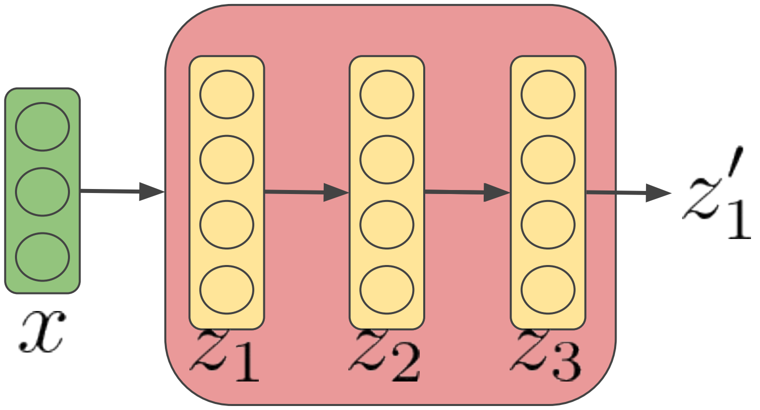
Step 2: Low-level backpropagation



Paper: Chen et al. [5]

Previous work: Gradient Checkpointing

Step 2: Low-level backpropagation

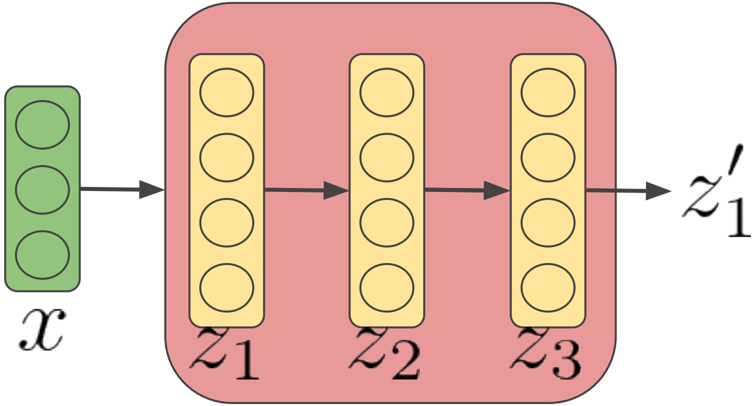


$$\frac{\partial \mathcal{L}(y,y')}{\partial z_i} = \overbrace{\frac{\partial \mathcal{L}(y,y')}{\partial z'_1}}^{\checkmark} \frac{\partial z'_1}{\partial z_i}$$

Paper: Chen et al. [5]

Previous work: Gradient Checkpointing

Step 2: Low-level backpropagation



$$\frac{\partial \mathcal{L}(y, y')}{\partial z_i} = \frac{\partial \mathcal{L}(y, y')}{\partial z'_1} \frac{\partial z'_1}{\partial z_i}$$

➔ Can calculate $\frac{\partial \mathcal{L}(y, y')}{\partial z_i}$ in O(S) memory

Paper: Chen et al. [5]

Previous work: Gradient Checkpointing

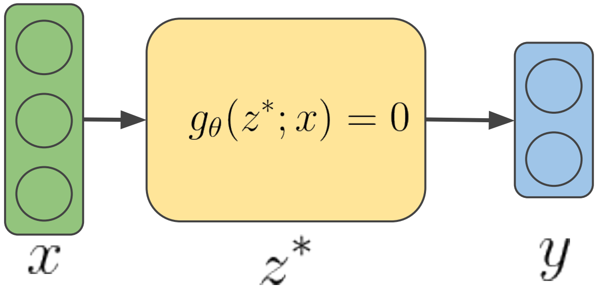
Summary:

- Cost: $O(S + m)$
- $L = S \times m$
- Can achieve $O(\sqrt{L})$ memory usage for 2x training time
- Can theoretically achieve O(log L) memory usage, if applied recursively

Paper: Chen et al. [5]

DEQ

- Equivalent to infinite-depth network!
- Different interpretation of deep networks
- We can backpropagate through equilibrium point: O(1) memory



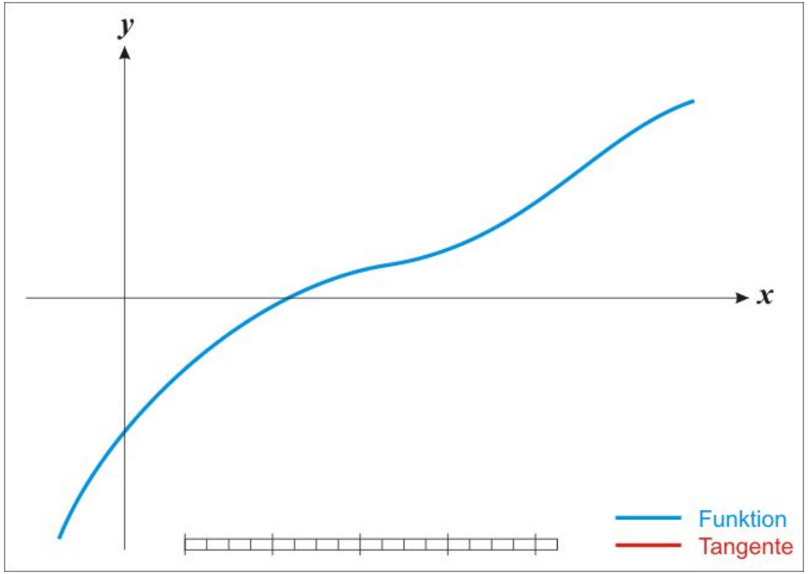
Forward pass

Find fixpoint: $g_\theta(z^*; x) = f_\theta(z^*; x) - z^* = 0$

Forward pass

For example with Newton's method:

$$z^{[i+1]} = z^{[i]} - \alpha \left(J_{g_\theta}^{-1} \big|_{z^{[i]}} \right) g_\theta(z^{[i]}, x)$$



Animation: wikipedia.org [6]

Forward pass

Find fixpoint: $g_\theta(z^*; x) = f_\theta(z^*; x) - z^* = 0$

For example with Newton's method:

$$z^{[i+1]} = z^{[i]} - \alpha(J_{g_\theta}^{-1} \mid_{z^{[i]}})g_\theta(z^{[i]}, x)$$

Can use any black-box root-finding algorithm $z* = \mathrm{RootFind}(g_\theta; x)$

Backward pass: 1st Approach

Procedure:

1. Fix a RootFind algorithm (e.g. Newton's method)
2. Unroll the Newton iterations
3. Do backpropagation through all iterations

Backward pass: 1st Approach

Procedure:

1. Fix a RootFind algorithm (e.g. Newton's method)
2. Unroll the Newton iterations
3. Do backpropagation through all iterations

Problems:

- Need knowledge of RootFind algorithm (Not a blackbox)
- Need to store intermediate results (Not O(1))

Backward pass: 2nd Approach

Procedure:

- Find root: $z^* = \mathrm{RootFind}(g_\theta; x)$
- Calculate loss: $\mathcal{L}(z^*, y)$
- Theorem 1: $\dfrac{\partial \mathcal{L}}{\partial \theta} = -\dfrac{\partial \mathcal{L}}{\partial z^*}\left(J_{g_\theta}^{-1}\mid_{z^*}\right)\dfrac{\partial f_\theta(z^*; x)}{\partial \theta}$

Backward pass: 2nd Approach

Procedure:

- Find root: $z^* = \mathrm{RootFind}(g_\theta; x)$
- Calculate loss: $\mathcal{L}(z^*, y)$
- Theorem 1: $\frac{\partial \mathcal{L}}{\partial \theta} = -\frac{\partial \mathcal{L}}{\partial z^*}\left(J_{g_\theta}^{-1}\big|_{z^*}\right)\frac{\partial f_\theta(z^*; x)}{\partial \theta}$

Advantages

- Independent of RootFind!
- Single step to backpropagate through 'infinite depth' network.

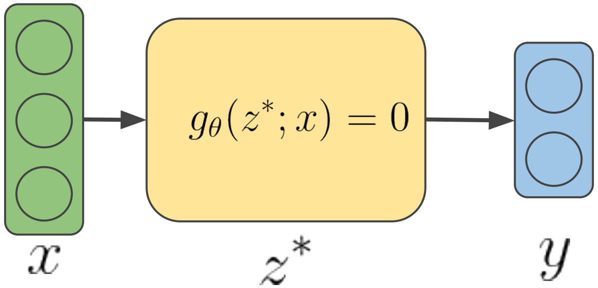
Broyden's Method

Problem: Calculating Jacobian Inverse is expensive

Solution: Use quasi-Newton methods.

$$\boxed{J_{g_\theta}^{-1}\Big|_{\mathbf{z}_{1:T}^{[i+1]}}} \approx B_{g_\theta}^{[i+1]} = B_{g_\theta}^{[i]} + \frac{\Delta \mathbf{z}^{[i+1]} - B_{g_\theta}^{[i]} \Delta g_\theta^{[i+1]}}{{\Delta \mathbf{z}^{[i+1]}}^\top B_{g_\theta}^{[i]} \Delta g_\theta^{[i+1]}} {\Delta \mathbf{z}^{[i+1]}}^\top B_{g_\theta}^{[i]}$$

DEQ



Guarantees:

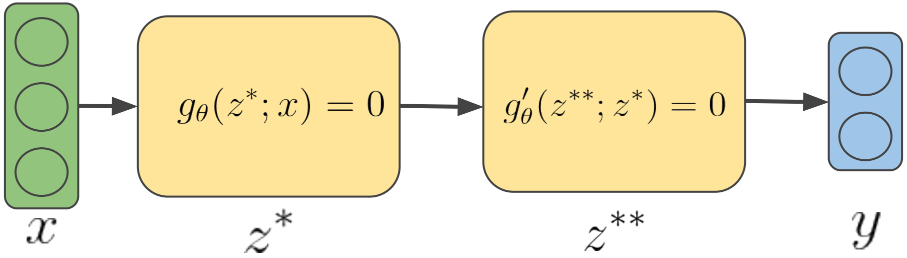
Memory consumption independent of depth

- O(1) memory consumption for backpropagation

Guarantees:

Sufficiency of a Single DEQ “Layer”

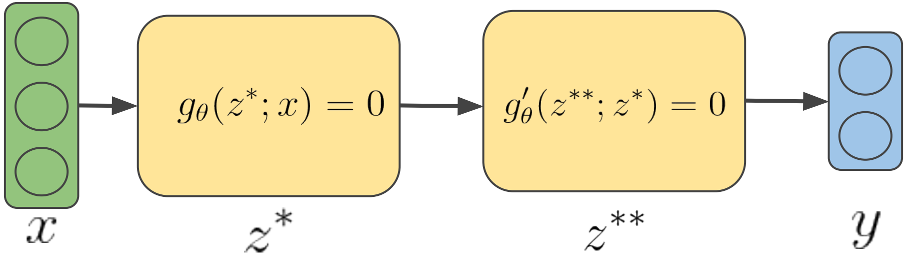
Idea: Stack multiple DEQs together, to get more representational power.



Guarantees:

Sufficiency of a Single DEQ “Layer”

Idea: Stack multiple DEQs together, to get more representational power.



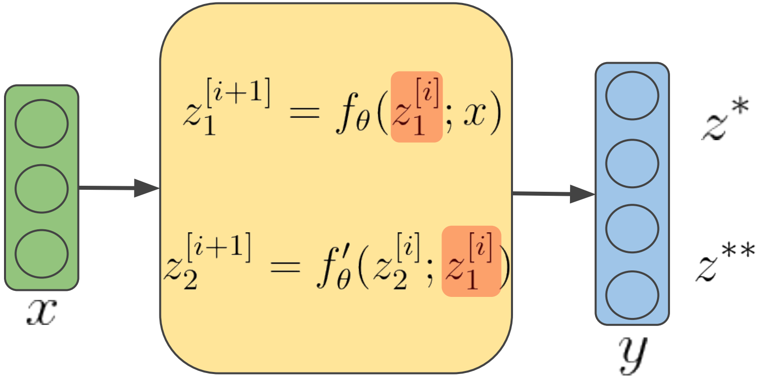
Theorem: A single DEQ “layer” is enough.

Guarantees:

Sufficiency of a Single DEQ “Layer”

Proof sketch:

- Stack the two layers
- Use output of first layer as input to second layer

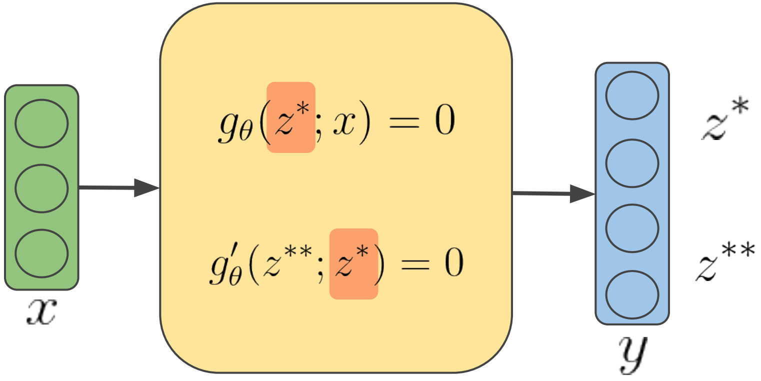


Guarantees:

Sufficiency of a Single DEQ "Layer"

Proof sketch:

- Stack the two layers
- Use output of first layer as input to second layer



Guarantees:

Universality of Weight-tied, Input-injected Networks

Theorem: Any traditional L-layer deep network can be represented by an L-layer deep weight tied, input-injected network with linear increase in width.

Guarantees:

Universality of Weight-tied, Input-injected Networks

Theorem: Any traditional L-layer deep network can be represented by an L-layer deep weight tied, input-injected network with linear increase in width.

Setting:

We have: $\mathbf{z}^{[i+1]} = \sigma^{[i]}(W^{[i]}\mathbf{z}^{[i]} + \mathbf{b}^{[i]}), \quad i = 0, \ldots, L-1, \quad \mathbf{z}^{[0]} = \mathbf{x}$

Guarantees:

Universality of Weight-tied, Input-injected Networks

Theorem: Any traditional L-layer deep network can be represented by an L-layer deep weight tied, input-injected network with linear increase in width.

Setting:

We have: $\mathbf{z}^{[i+1]} = \sigma^{[i]}(W^{[i]}\mathbf{z}^{[i]} + \mathbf{b}^{[i]}), \quad i = 0, \ldots, L-1, \quad \mathbf{z}^{[0]} = \mathbf{x}$

We want: $\tilde{\mathbf{z}}^{[i+1]} = \sigma(W_z\tilde{\mathbf{z}}^{[i]} + W_x\mathbf{x} + \tilde{\mathbf{b}}), \ \ i = 0, \ldots, L-1.$

Guarantees:

Universality of Weight-tied, Input-injected Networks

Proof:

We have: $\mathbf{z}^{[i+1]} = \sigma^{[i]}(W^{[i]}\mathbf{z}^{[i]} + \mathbf{b}^{[i]}), \quad i = 0, \ldots, L-1, \quad \mathbf{z}^{[0]} = \mathbf{x}$

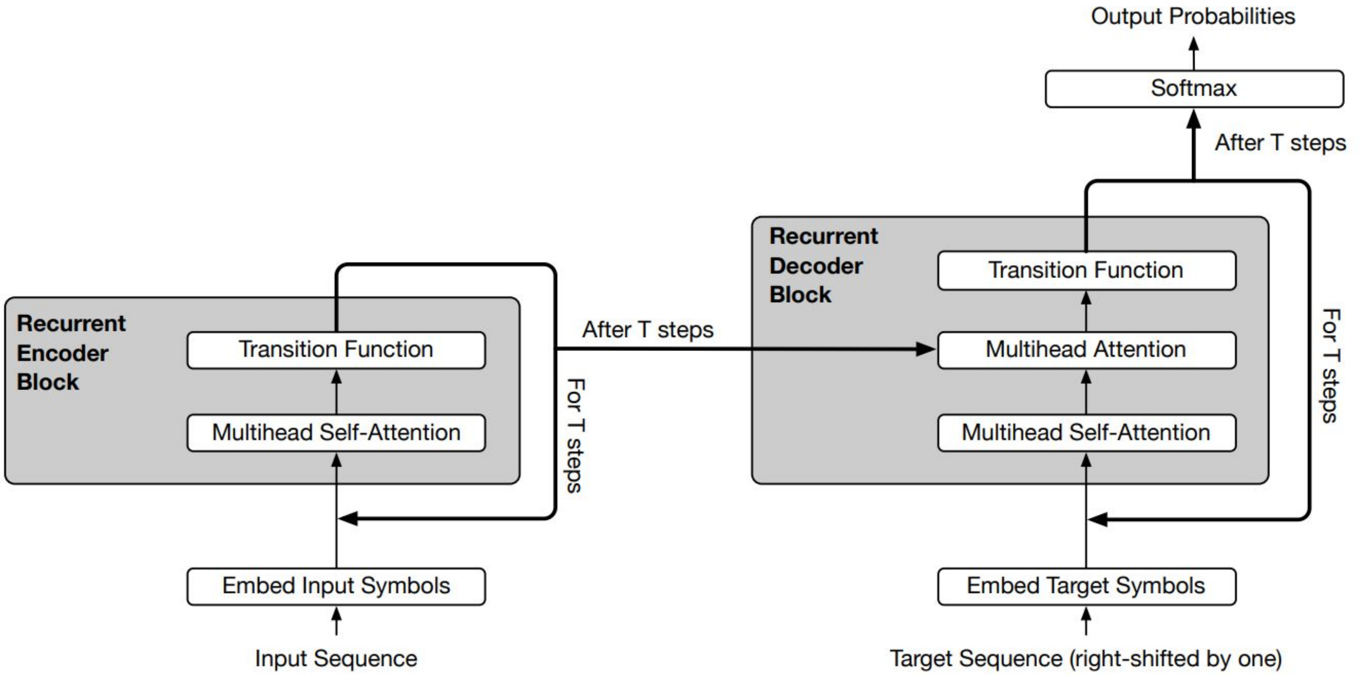
We want: $\tilde{\mathbf{z}}^{[i+1]} = \sigma(W_z\tilde{\mathbf{z}}^{[i]} + W_x\mathbf{x} + \tilde{\mathbf{b}}), \; i = 0, \ldots, L-1.$

$$W_z = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ W^{[1]} & 0 & \cdots & 0 & 0 \\ 0 & W^{[2]} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & W^{[L-1]} & 0 \end{bmatrix}, W_x = \begin{bmatrix} W^{[0]} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \; \tilde{\mathbf{b}} = \begin{bmatrix} \mathbf{b}^{[0]} \\ \mathbf{b}^{[1]} \\ \vdots \\ \mathbf{b}^{[L-1]} \end{bmatrix}, \; \sigma = \begin{bmatrix} \sigma^{[0]} \\ \sigma^{[1]} \\ \vdots \\ \sigma^{[L-1]} \end{bmatrix}$$

This is not done in practice!

Evaluation

Universal Transformer



Dehghani et al. [9]

TrellisNet: Atomic Level

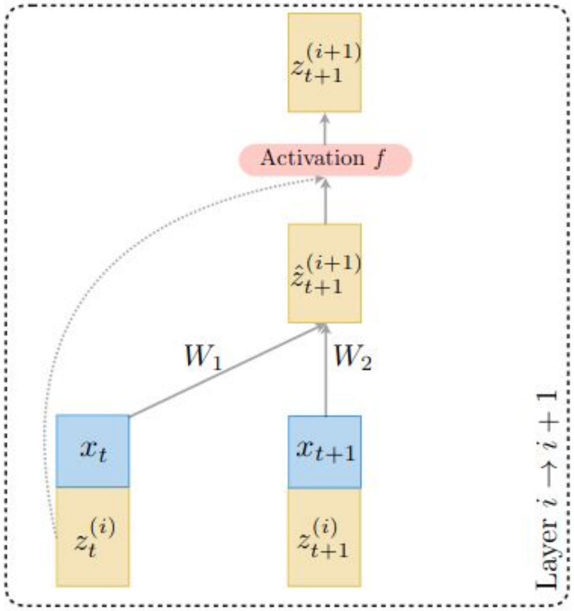


Image: Bai et al. [10]

TrellisNet

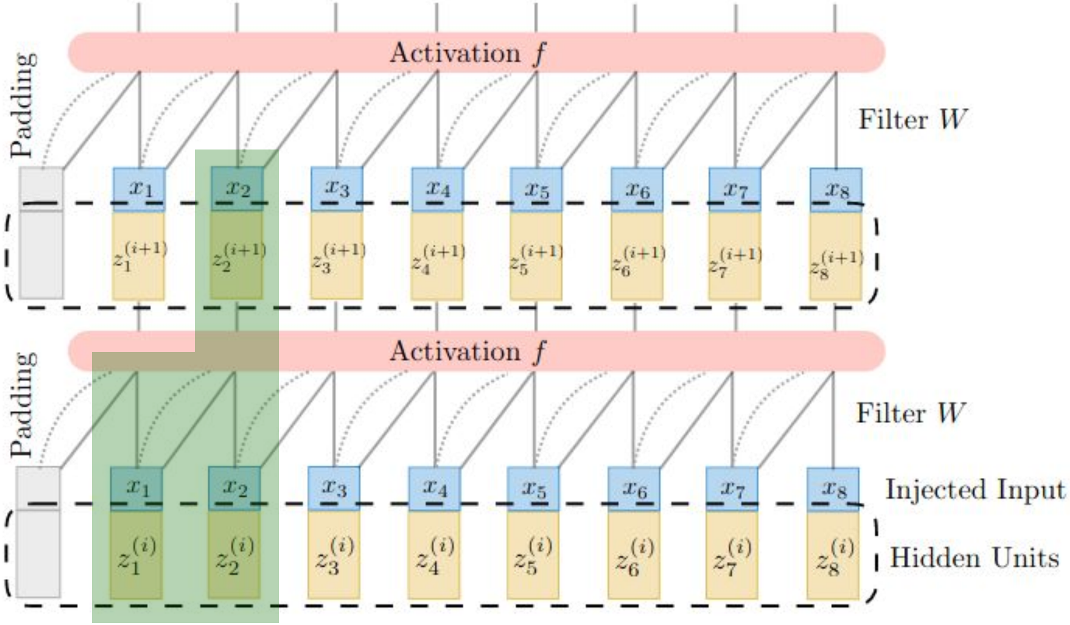


Image: Bai et al. [10]

Results: Penn Treebank

Word-level Language Modeling w/ Penn Treebank (PTB)				
Model	# Params	Non-embedding model size	Test perplexity	Memory†
Variational LSTM [22]	66M	-	73.4	-
NAS Cell [55]	54M	-	62.4	-
NAS (w/ black-box hyperparameter tuner) [32]	24M	20M	59.7	-
AWD-LSTM [34]	24M	20M	58.8	-
DARTS architecture search (second order) [29]	23M	20M	**55.7**	-
60-layer TrellisNet (w/ auxiliary loss, w/o MoS) [8]	24M	20M	57.0	8.5GB
DEQ-TrellisNet (ours)	24M	20M	57.1	**1.2GB**

Bai et al. [1]

Results: WikiText-103

Word-level Language Modeling w/ WikiText-103 (WT103)				
Model	# Params	Non-Embedding Model Size	Test perplexity	Memory†
Generic TCN [7]	150M	34M	45.2	-
Gated Linear ConvNet [17]	230M	-	37.2	-
AWD-QRNN [33]	159M	51M	33.0	7.1GB
Relational Memory Core [40]	195M	60M	31.6	-
Transformer-XL (X-large, adaptive embed., on TPU) [16]	257M	224M	**18.7**	12.0GB
70-layer TrellisNet (+ auxiliary loss, etc.) [8]	180M	45M	29.2	24.7GB
70-layer TrellisNet with *gradient checkpointing*	180M	45M	29.2	5.2GB
DEQ-TrellisNet (ours)	180M	45M	**29.0**	**3.3GB**
Transformer-XL (medium, 16 layers)	165M	44M	24.3	8.5GB
DEQ-Transformer (medium, ours).	172M	43M	24.2	**2.7GB**
Transformer-XL (medium, 18 layers, adaptive embed.)	110M	72M	23.6	9.0GB
DEQ-Transformer (medium, adaptive embed., ours)	110M	70M	**23.2**	3.7GB
Transformer-XL (small, 4 layers)	139M	4.9M	35.8	4.8GB
Transformer-XL (small, weight-tied 16 layers)	138M	4.5M	34.9	6.8GB
DEQ-Transformer (small, ours).	138M	4.5M	**32.4**	**1.1GB**

Bai et al. [1]

Results: WikiText-103

Word-level Language Modeling w/ WikiText-103 (WT103)				
Model	# Params	Non-Embedding Model Size	Test perplexity	Memory[†]
Generic TCN [7]	150M	34M	45.2	-
Gated Linear ConvNet [17]	230M	-	37.2	-
AWD-QRNN [33]	159M	51M	33.0	7.1GB
Relational Memory Core [40]	195M	60M	31.6	-
Transformer-XL (X-large, adaptive embed., on TPU) [16]	257M	224M	**18.7**	12.0GB
70-layer TrellisNet (+ auxiliary loss, etc.) [8]	180M	45M	29.2	24.7GB
70-layer TrellisNet with *gradient checkpointing*	180M	45M	29.2	5.2GB
DEQ-TrellisNet (ours)	180M	45M	**29.0**	**3.3GB**
Transformer-XL (medium, 16 layers)	165M	44M	24.3	8.5GB
DEQ-Transformer (medium, ours).	172M	43M	24.2	**2.7GB**
Transformer-XL (medium, 18 layers, adaptive embed.)	110M	72M	23.6	9.0GB
DEQ-Transformer (medium, adaptive embed., ours)	110M	70M	**23.2**	3.7GB
Transformer-XL (small, 4 layers)	139M	4.9M	35.8	4.8GB
Transformer-XL (small, weight-tied 16 layers)	138M	4.5M	34.9	6.8GB
DEQ-Transformer (small, ours).	138M	4.5M	**32.4**	**1.1GB**

Bai et al. [1]

Results: WikiText-103

Model	# Params	Non-Embedding Model Size	Test perplexity	Memory†
Word-level Language Modeling w/ WikiText-103 (WT103)				
Generic TCN [7]	150M	34M	45.2	-
Gated Linear ConvNet [17]	230M	-	37.2	-
AWD-QRNN [33]	159M	51M	33.0	7.1GB
Relational Memory Core [40]	195M	60M	31.6	-
Transformer-XL (X-large, adaptive embed., on TPU) [16]	257M	224M	**18.7**	12.0GB
70-layer TrellisNet (+ auxiliary loss, etc.) [8]	180M	45M	29.2	24.7GB
70-layer TrellisNet with *gradient checkpointing*	180M	45M	29.2	5.2GB
DEQ-TrellisNet (ours)	180M	45M	**29.0**	**3.3GB**
Transformer-XL (medium, 16 layers)	165M	44M	24.3	8.5GB
DEQ-Transformer (medium, ours).	172M	43M	24.2	**2.7GB**
Transformer-XL (medium, 18 layers, adaptive embed.)	110M	72M	23.6	9.0GB
DEQ-Transformer (medium, adaptive embed., ours)	110M	70M	**23.2**	3.7GB
Transformer-XL (small, 4 layers)	139M	4.9M	35.8	4.8GB
Transformer-XL (small, weight-tied 16 layers)	138M	4.5M	34.9	6.8GB
DEQ-Transformer (small, ours).	138M	4.5M	**32.4**	**1.1GB**

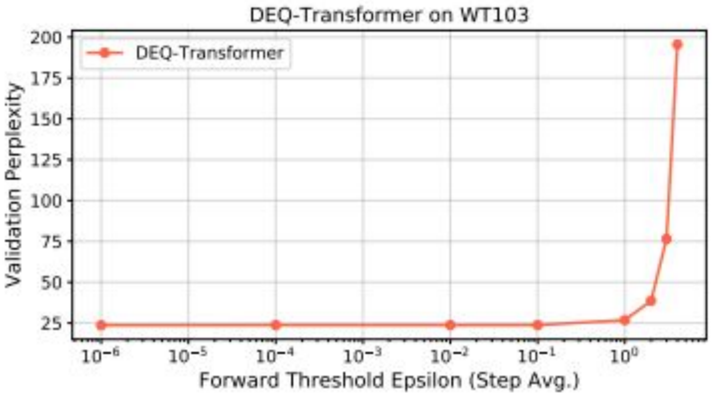
Bai et al. [1]

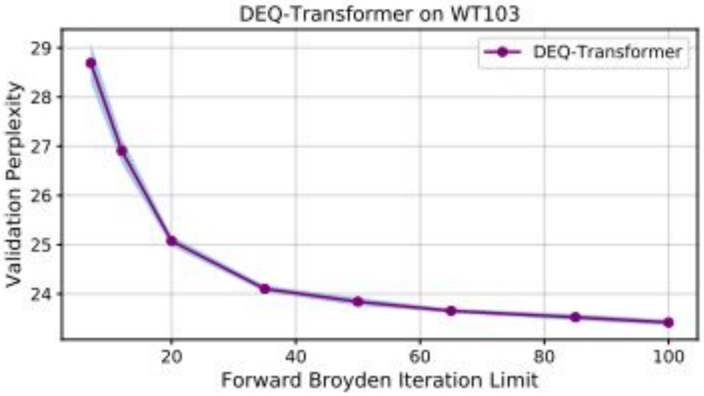
Results: WikiText-103

Word-level Language Modeling w/ WikiText-103 (WT103)				
Model	# Params	Non-Embedding Model Size	Test perplexity	Memory[†]
Generic TCN [7]	150M	34M	45.2	-
Gated Linear ConvNet [17]	230M	-	37.2	-
AWD-QRNN [33]	159M	51M	33.0	7.1GB
Relational Memory Core [40]	195M	60M	31.6	-
Transformer-XL (X-large, adaptive embed., on TPU) [16]	257M	224M	**18.7**	12.0GB
70-layer TrellisNet (+ auxiliary loss, etc.) [8]	180M	45M	29.2	24.7GB
70-layer TrellisNet with *gradient checkpointing*	180M	45M	29.2	5.2GB
DEQ-TrellisNet (ours)	180M	45M	**29.0**	**3.3GB**
Transformer-XL (medium, 16 layers)	165M	44M	24.3	8.5GB
DEQ-Transformer (medium, ours).	172M	43M	24.2	**2.7GB**
Transformer-XL (medium, 18 layers, adaptive embed.)	110M	72M	23.6	9.0GB
DEQ-Transformer (medium, adaptive embed., ours)	110M	70M	**23.2**	3.7GB
Transformer-XL (small, 4 layers)	139M	4.9M	35.8	4.8GB
Transformer-XL (small, weight-tied 16 layers)	138M	4.5M	34.9	6.8GB
DEQ-Transformer (small, ours).	138M	4.5M	**32.4**	**1.1GB**

Bai et al. [1]

Results: Broyden's Method





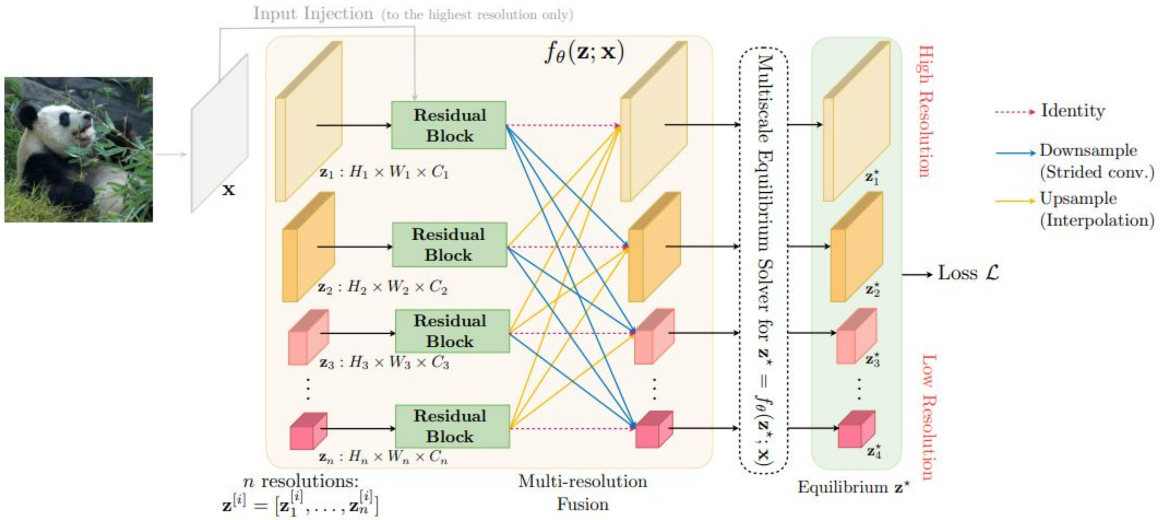
Bai et al. [1]

Results: Runtime

DEQ / 18-layer Transformer		DEQ / 70-layer TrellisNet	
Training	Inference	Training	Inference
2.82×	1.76×	2.40×	1.64×

Bai et al. [1]

DEQs Today



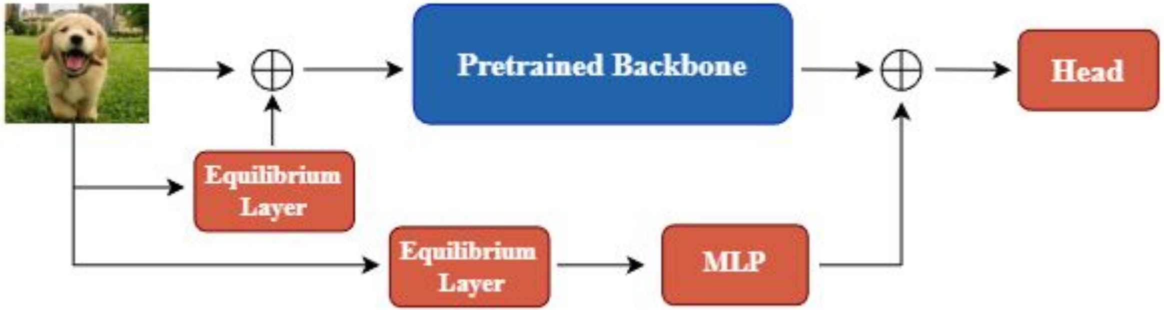
- Close to state of the art
- Very versatile (segmentation and classification)

Paper: Bai et al. [7]

DEQs Today

LION: Implicit Vision Prompt Tuning

Haixin Wang[1] **Jianlong Chang**[2] **Xiao Luo**[1] **Jinan Sun**[1] **Zhouchen Lin**[1] **Qi Tian**[2*]
[1]Peking University, Beijing, China [2]Huawei Cloud & AI, Beijing, China
wang.hx@stu.pku.edu.cn, {xiaoluo, sjn, zlin}@pku.edu.cn, {jianlong.chang, tian.qi1}@huawei.com



Paper: Wang et al. [8]

Conclusion

- Constant memory consumption
- New perspective on deep feed-forward NNs
- Slower to train
- Convergence to fix-point not guaranteed
- Theoretically equivalent to general network with linear width increase
- Every layer must have the same structure
- More restrictive than gradient checkpointing

Sources

[1]: Shaojie Bai, J. Zico Kolter, Vladlen Koltun. Deep Equilibrium Models
[2]: http://implicit-layers-tutorial.org/deep_equilibrium_models/
[3]: Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations.
[4]: Matthew MacKay, Paul Vicol, Jimmy Ba, and Roger B. Grosse. Reversible recurrent neural networks.
[5]: Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost.
[6]: https://en.wikipedia.org/wiki/Newton%27s_method
[7]: Shaojie Bai, J. Zico Kolter, Vladlen Koltun. Multiscale Deep Equilibrium Models
[8]: Wang, Haixin, Jianlong Chang, Xiao Luo, Jinan Sun, Zhouchen Lin and Qi Tian. LION: Implicit Vision Prompt Tuning.
[9]: Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, Łukasz Kaiser. Universal Transformers.
[10]: Shaojie Bai, J. Zico Kolter, Vladlen Koltun. Trellis Networks For Sequence Modeling

Results: Fixpoint