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

Classical deep feedforward NN

\[ x \xrightarrow{W_1} z_1 \xrightarrow{W_2} z_2 \xrightarrow{\ldots} z_k \xrightarrow{W_k} y \]

Weight-tied Network

$\mathbf{x} \xrightarrow{W} \mathbf{z}_1 \xrightarrow{W} \mathbf{z}_2 \xrightarrow{\cdots W} \mathbf{z}_k \xrightarrow{W} \mathbf{y}$

Weight-tied, input-injected Network

Infinite depth network

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

\[ g_\theta(z^*; x) = 0 \]

DEQ

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

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

$x \xrightarrow{} z_1 \xrightarrow{} z_2 \xrightarrow{} z_3 \xrightarrow{} z_4 \xrightarrow{} \ldots \xrightarrow{} z_k \xrightarrow{} y$

Paper: Chen et al. [5]
Previous work: Gradient Checkpointing

Step 1: High-level backpropagation

→ Can calculate $\frac{\partial L(y, y')}{\partial z'_i}$ 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

\[
\frac{\partial L(y, y')}{\partial z_i} = \left\{ \frac{\partial L(y, y')}{\partial z'_1} \right\} \frac{\partial z'_1}{\partial z_i}
\]

Paper: Chen et al. [5]
Previous work: Gradient Checkpointing

Step 2: Low-level backpropagation

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

\[
\frac{\partial L(y, y')}{\partial z_i} = \frac{\partial L(y, y')}{\partial z'_1} \frac{\partial z'_1}{\partial z_i}
\]
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: \[ g_\theta(z^*; x) = f_\theta(z^*; x) - z^* = 0 \]
For example with Newton’s method:

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

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

For example with Newton’s method:

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

Can use any black-box root-finding algorithm \( z^* = \text{RootFind}(g_\theta; x) \)
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: \[ z^* = \text{RootFind}(g_\theta; x) \]
- Calculate loss: \[ \mathcal{L}(z^*, y) \]
- Theorem 1: \[ \frac{\partial \mathcal{L}}{\partial \theta} = - \frac{\partial \mathcal{L}}{\partial z^*} (J_{g_\theta}^{-1} |_{z^*}) \frac{\partial f_\theta(z^*; x)}{\partial \theta} \]
Backward pass: 2nd Approach

Procedure:

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

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.

\[ J_{g_{\theta}}^{-1} \big|_{z_{1:T}^{[i+1]}} \approx B_{g_{\theta}}^{[i+1]} = B_{g_{\theta}}^{[i]} + \frac{\Delta z^{[i+1]}}{\Delta z^{[i+1]}} \mathbf{g}_{\theta} \Delta g^{[i+1]} \mathbf{g}_{\theta} \Delta z^{[i+1]} \mathbf{g}_{\theta} \top \]
DEQ

\[ g_0(z^*; x) = 0 \]
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

\[ g_0(z^*; x) = 0 \]
\[ g_0'(z^{**}; z^*) = 0 \]
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: \( z^{i+1} = \sigma^i(W^i z^i + b^i), \quad i = 0, \ldots, L - 1, \quad z^0 = x \)
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: \( z^{[i+1]} = \sigma^{[i]}(W^{[i]}z^{[i]} + b^{[i]}), \quad i = 0, \ldots, L - 1, \quad z^{[0]} = x \)

We want: \( \tilde{z}^{[i+1]} = \sigma(W_{z}\tilde{z}^{[i]} + W_{x}x + \tilde{b}), \quad i = 0, \ldots, L - 1. \)
Guarantees:

Universality of Weight-tied, Input-injected Networks

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

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

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

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]
### Word-level Language Modeling w/ Penn Treebank (PTB)

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

Bai et al. [1]
### Results: WikiText-103

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

Bai et al. [1]
## Results: WikiText-103

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

Bai et al. [1]
Results: WikiText-103

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

Bai et al. [1]
## Results: WikiText-103

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

Bai et al. [1]
Results: Broyden’s Method

Bai et al. [1]
### Results: Runtime

<table>
<thead>
<tr>
<th></th>
<th>DEQ / 18-layer Transformer</th>
<th>DEQ / 70-layer TrellisNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>2.82×</td>
<td>2.40×</td>
</tr>
<tr>
<td>Inference</td>
<td>1.76×</td>
<td>1.64×</td>
</tr>
</tbody>
</table>

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\textsuperscript{1}  
Jianlong Chang\textsuperscript{2}  
Xiao Luo\textsuperscript{1}  
Jinan Sun\textsuperscript{1}  
Zhouchen Lin\textsuperscript{1}  
Qi Tian\textsuperscript{2*}  
\textsuperscript{1}Peking University, Beijing, China  
\textsuperscript{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

[2]: http://implicit-layers-tutorial.org/deep_equilibrium_models/
[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
[8]: Wang, Haixin, Jianlong Chang, Xiao Luo, Jinan Sun, Zhouchen Lin and Qi Tian. LION: Implicit Vision Prompt Tuning.
[10]: Shaojie Bai, J. Zico Kolter, Vladlen Koltun. Trellis Networks For Sequence Modeling
Results: Fixpoint

Trellis Network

Difference Norm $\|f(x) - y\|

Layers

Weight-Tied/Universal Transformer

Difference Norm $\|f(x) - y\|

Layers