State Reification Networks

Alex Lamb, Jonathan Binas, Anirudh Goyal, Sandeep Subramanian, Denis Kazakov, Ioannis Mitliagkas, Yoshua Bengio, Michael Mozer
Reification in Cognitive Psychology

- Human visual perception involves interpreting scenes that can be noisy, missing features, or ambiguous.

- Reification refers to the fact that the output of perception is a coherent whole, not the raw features.
Reification in Machine Learning

- **Models** are more useful for prediction than are the *raw data*.
- If that’s true for real-world data, might it also be true for data that originate from within the model (i.e., its hidden states)?
- Reification = exchanging inputs with points that are likely under the model.
Examples of Reification in Machine Learning

- **Batch normalization**
  - Performs extremely well, yet only considers 1st and 2nd moments

- **Radial Basis Function Networks**
  - Projects to “prototypes” around each class ➔ very restrictive

- **Generative Classifiers**
  - Requires extremely strong generative model, poor practical performance
State Reification

Input Space
State Reification

- Hidden states can have simpler statistical structure

Input Space

Hidden Space
Explicit Frameworks for State Reification

- Two frameworks for different model types
  - Denoising Autoencoder (CNNs and RNNs)
  - Attractor Networks (RNNs)

\[ L = L_{task}(x, y) + \lambda_{rec} L_{rec}(h) \]
## Task Overview

<table>
<thead>
<tr>
<th>Architecture</th>
<th>State reification</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>Denoising autoencoder</td>
<td>Generalization and adversarial robustness</td>
</tr>
<tr>
<td>RNN</td>
<td>Attractor net</td>
<td>Parity, Majority Function, Reber Grammar, Sequence Symmetry</td>
</tr>
<tr>
<td>RNN</td>
<td>Denoising autoencoder</td>
<td>Accumulating errors with free running sequence generation</td>
</tr>
</tbody>
</table>
## Task Overview

<table>
<thead>
<tr>
<th>Architecture</th>
<th>State reification</th>
<th>Task</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>Denoising autoencoder</td>
<td>Generalization and adversarial robustness</td>
<td></td>
</tr>
<tr>
<td>RNN</td>
<td>Attractor net</td>
<td>Parity, Majority Function, Reber Grammar, Sequence Symmetry</td>
<td></td>
</tr>
<tr>
<td>RNN</td>
<td>Denoising autoencoder</td>
<td>Accumulating errors with free running sequence generation</td>
<td></td>
</tr>
</tbody>
</table>
Denoising Autoencoder

**Input-output mapping with one hidden layer**

**DAE that produces reified output**

**Integrated architecture**
Denoising Autoencoder

\[ \mathcal{L}_{\text{rec}}(x) = \frac{1}{N} \sum_{n=1}^{N} \left( \left\| r_{\theta} \left( x^{(n)} + a^{(n)} \right) - x^{(n)} \right\|_2^2 \right) \]

\[ a^{(n)} \sim \mathcal{N}(0, \sigma^2 I) \]

\( r_{\theta} \) Learned denoising function.

\[ \frac{r_{\sigma}(x) - x}{\sigma^2} \rightarrow \frac{\partial \log p(x)}{\partial x} \quad \text{as} \quad \sigma \to 0. \]

(Alain and Bengio, 2012)
Adversarial Robustness Setup

- Projected Gradient Descent Attack (PGD):

\[ x^{t+1} = \Pi_{x+S} \left( x^t + \alpha \text{sgn} (\nabla_x L_{task}(x, y)) \right) \]

- Train with adversarial examples and DAE reconstruction loss:

\[ L = L_{task}(x, y) + L_{task}(\tilde{x}, y) + \lambda_{rec} \sum_{i \in S} L^i_{rec}(h_i) \]
Adversarial Robustness → Improving Generalization

- Improves generalization in adversarial robustness from training set to test set.

<table>
<thead>
<tr>
<th>Model</th>
<th>PGD Accuracy baseline</th>
<th>(20 steps) SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>PreActResNet18</td>
<td>37.87</td>
<td>39.20</td>
</tr>
<tr>
<td>WideResNet28-10</td>
<td>43.28</td>
<td>44.06</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>PGD Steps</th>
<th>Attack Epsilon</th>
<th>PGD Accuracy CNN</th>
<th>CNN+</th>
<th>CNN+SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>7</td>
<td>0.03</td>
<td>33.0</td>
<td>34.2</td>
<td>45.0</td>
</tr>
<tr>
<td>Normal</td>
<td>50</td>
<td>0.03</td>
<td>31.6</td>
<td>32.5</td>
<td>42.1</td>
</tr>
<tr>
<td>Normal</td>
<td>200</td>
<td>0.03</td>
<td>31.4</td>
<td>32.2</td>
<td>41.5</td>
</tr>
<tr>
<td>Normal</td>
<td>100</td>
<td>0.03</td>
<td>35.3</td>
<td>39.2</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>100</td>
<td>0.04</td>
<td>24.8</td>
<td>28.0</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>100</td>
<td>0.06</td>
<td>14.3</td>
<td>15.6</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>100</td>
<td>0.08</td>
<td>12.0</td>
<td>13.0</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>100</td>
<td>0.10</td>
<td>11.7</td>
<td>12.9</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>100</td>
<td>0.20</td>
<td>10.2</td>
<td>11.3</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>100</td>
<td>0.30</td>
<td>8.4</td>
<td>9.6</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>100</td>
<td>0.03</td>
<td>33.4</td>
<td></td>
<td>40.1</td>
</tr>
<tr>
<td>Noiseless Attack BPDA, Skip-DAE</td>
<td>100</td>
<td>0.03</td>
<td>33.4</td>
<td>67.1</td>
<td></td>
</tr>
</tbody>
</table>
Adversarial Robustness - some analysis

- Reconstruction error is larger on adversarial examples.
- When the autoencoder is in the hidden states, this detection doesn’t require a high-capacity autoencoder.
# Experiments

<table>
<thead>
<tr>
<th>Architecture</th>
<th>State reification</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>Denoising autoencoder</td>
<td>Generalization and adversarial robustness</td>
</tr>
<tr>
<td>RNN</td>
<td>Attractor net</td>
<td>Parity, Majority Function, Reber Grammar, Sequence Symmetry</td>
</tr>
<tr>
<td>RNN</td>
<td>Denoising autoencoder</td>
<td>Accumulating errors with free running sequence generation</td>
</tr>
</tbody>
</table>
Attractor Net

Network whose dynamics can be characterized as moving downhill in energy, arriving at stable point.
Attractor Net Dynamics

Output \[ y = \tanh(v_0 + U_0 a_\infty) \]

Update \[ a_t = W \tanh(a_{t-1}) + x^+ \]

Initialization \[ a_0 = 0 \]
\[ x^+ = v_I + U_I \tanh^{-1}(x) \]

To achieve attractor dynamics (Koiran, 1994):
\[ w_{ij} = w_{ji} \]
\[ w_{ii} \geq 0 \]
Attractor Net Training: Denoising by Convergent Dynamics

Set of target states \( \{\xi_1, \ldots, \xi_n\} \)

\[
E = |y - \xi_i|^2
\]

\[
x = \xi_i + \eta \quad \eta \sim \mathcal{N}(0, \sigma^2)
\]
Attractor Nets in RNNs

- In an imperfectly trained RNN, feedback at each step can inject noise
  - Noise can amplify over time

- Suppose we could ‘clean up’ the representation at each step to reduce that noise?
  - May lead to better learning and generalization
State-Reified RNN
State-Reified RNN
Parity Task

- 10 element sequences
- Training on 256 sequences

1001000101 ➞ 0
0010101011 ➞ 1

novel sequences

noisy sequences

![Graphs showing proportion correct for different architectures and datasets](chart.png)
Majority Function

- 100 sequences, length 11-29

01001000101 $\rightarrow$ 0
11010111011 $\rightarrow$ 1

Novel sequences

Noisy sequences
Reber Grammar

- Grammatical or not?
- Vary training set size

BTTXPVE $\rightarrow 0$
BPTTVPSE $\rightarrow 1$
Symmetry

- Is sequence symmetric?
- 5 symbols, filler, 5 symbols

![Graph showing proportion correct for different architectures and filler lengths](image)

**Examples:**
- ACAFBXBFACA → 1
- ACAFBXBFABA → 0
## Experiments

<table>
<thead>
<tr>
<th>Architecture</th>
<th>State reification</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>Denoising autoencoder</td>
<td>Generalization and adversarial robustness</td>
</tr>
<tr>
<td>RNN</td>
<td>Attractor net</td>
<td>Parity, Majority Function, Reber Grammar, Sequence Symmetry</td>
</tr>
<tr>
<td>RNN</td>
<td>Denoising autoencoder</td>
<td>Accumulating errors with free running sequence generation</td>
</tr>
</tbody>
</table>
Identifying Failures in Teacher Forcing

- Train LSTM on character-level Text8 dataset for language modeling.
- Train a denoising autoencoder on the hidden states while doing teacher forcing

<table>
<thead>
<tr>
<th>Sampling Steps</th>
<th>Reconstruction Error Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.00</td>
</tr>
<tr>
<td>50</td>
<td>1.03</td>
</tr>
<tr>
<td>180</td>
<td>1.12</td>
</tr>
<tr>
<td>300</td>
<td>1.34</td>
</tr>
</tbody>
</table>
Open Problems

- How well does state reification scale to harder tasks and larger datasets?

- Denoising autoencoders with quadratic loss may not be ideal for reification.
  - Maybe GANs or better generative models could help?

- Thinking about how the states are changed to make reification easier (are these changes ideal or not)?
  - For example, reification might be made easier by having more compressed representations.
Questions?

- Can also email questions to any of the authors!