Tunable Efficient Unitary Neural Networks (EUNN) and their application to RNNs

Li Jing*, Yichen Shen*, Tena Dubček, John Peurifoy, Scott Skirlo, Yann LeCun, Max Tegmark, Marin Soljačić

* equal contribution
Gradient Vanishing/Explosion Problem

- During backpropagation through time, hidden to hidden Jacobin matrix is multiplied multiple times.
- Gradient vanishing/explosion makes RNN hard to train.
Conventional Solution: LSTM

- Practically, gradient clipping is required
- Slow to learn long term dependency
New Solution: Unitary/Orthogonal RNN

Unitary/Orthogonal matrices keep the norm of vectors: $\|UX\| = \|X\|$

By enforcing hidden to hidden transition matrix to be unitary/orthogonal, no matter how many time steps are propagated, the norm of the gradient will stay the same.

$$\left\| \prod_{k=t}^{T-1} \frac{\partial h^{(k+1)}}{\partial h^{(k)}} \right\| \sim 1$$
Related Works

- Restricted-capacity Unitary Matrix Parametrization (Arjovsky, ICML 2016)
- Full-capacity Unitary Matrix by projection (Wisdom, NIPS 2016)
- Orthogonal Matrix Parametrization by reflection (Mhammedi, ICML 2017)
- Orthogonal Matrix by regularization (Vorontsov, ICML 2017)
Efficient Unitary Matrix Parametrization

Physical system inspired:

Efficient Unitary Matrix Parametrization

SU(2) element: rotation matrix + relevant phase

- Strict, complete, unique
- Full unitary space
Efficient Unitary Matrix Implementation

• Parallel, Sparse
• O(1) per parameter
• No need to customize gradient

Algorithm 1 Efficient implementation for $F$ with parameters $\theta$ and $\phi$.

Input: input $x$, size $N$; parameters $\theta$ and $\phi$, size $N/2$; constant permutation index list $\text{ind}_1$ and $\text{ind}_2$.

Output: output $y$, size $N$.

1. $v_1 \leftarrow \text{concatenate}(\cos \theta, \cos \theta \ast \exp(i\phi))$
2. $v_2 \leftarrow \text{concatenate}(\sin \theta, -\sin \theta \ast \exp(i\phi))$
3. $v_1 \leftarrow \text{permute}(v_1, \text{ind}_1)$
4. $v_2 \leftarrow \text{permute}(v_2, \text{ind}_1)$
5. $y \leftarrow v_1 \ast x + v_2 \ast \text{permute}(x, \text{ind}_2)$

Tensorflow: https://github.com/jingli9111/EUNN-tensorflow
PyTorch: https://github.com/jingli9111/URNN-PyTorch
Theano: https://github.com/guanaus/EUNN-theano

sparse block diagonal matrix element-wise functions
Tunable Parametrization

- reduce capacity allows larger hidden state
- reduce capacity increases training speed
FFT-style Parametrization

- $n \log(n)$ parameters
- Minimum number of parameters for symmetric unitary space
- Almost same performance as Tunable style with large capacity

$$W = \begin{pmatrix} e^{i\phi} \cos \theta & -e^{i\phi} \sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$
Experiment results

- Copying Memory

EURNN outperforms all other models for long delay time case.

- Pixel Permuted MNIST

EURNN outperforms LSTM in both final performance and training speed.

EURNN tunable-style achieves highest accuracy with least number of parameters.
Experiment results

- Speech spectrum prediction

TIMIT spectrum sampled in 8 GHz. RNN model is required to predict next frame based on previous frames.

EURNN outperforms LSTM with less number of parameters

<table>
<thead>
<tr>
<th>Model</th>
<th>hidden size (capacity)</th>
<th>number of parameters</th>
<th>MSE (validation)</th>
<th>MSE (test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>64</td>
<td>33k</td>
<td>71.4</td>
<td>66.0</td>
</tr>
<tr>
<td>LSTM</td>
<td>128</td>
<td>98k</td>
<td>55.3</td>
<td>54.5</td>
</tr>
<tr>
<td>EURNN (tunable style)</td>
<td>128 (2)</td>
<td>33k</td>
<td>63.3</td>
<td>63.3</td>
</tr>
<tr>
<td>EURNN (tunable style)</td>
<td>128 (32)</td>
<td>35k</td>
<td>52.3</td>
<td>52.7</td>
</tr>
<tr>
<td>EURNN (tunable style)</td>
<td>128 (128)</td>
<td>41k</td>
<td><strong>51.8</strong></td>
<td><strong>51.9</strong></td>
</tr>
<tr>
<td>EURNN (FFT style)</td>
<td>128 (FFT)</td>
<td>34k</td>
<td>52.3</td>
<td>52.4</td>
</tr>
</tbody>
</table>
Conclusion

Efficient Unitary NN (EUNN)

- **Efficient**: $O(1)$ operation per parameter
- **Strict**: strictly unitary parametrization
- **Tunable**: from small subspace to full unitary space, trade off capacity to hidden size
- **Easy implementation**: element-wise functions, no need to implement backpropagation
- **FFT approximation**: provides further speed-up
Future work

• Combine unitary matrices with other mechanisms: gated system, attention mechanism etc

• Go beyond Recurrent Neural Network: Parametrization of Semi-unitary Matrices
Thank you!