Understanding Deep Learning Requires Rethinking Generalization

Chiyuan Zhang\textsuperscript{1} Samy Bengio\textsuperscript{3} Moritz Hardt\textsuperscript{3} Benjamin Recht\textsuperscript{2} Oriol Vinyals\textsuperscript{4}

\textsuperscript{1}Massachusetts Institute of Technology
\textsuperscript{2}University of California, Berkeley
\textsuperscript{3}Google Brain
\textsuperscript{4}Google DeepMind

ICLR, 2017
Presenter: Arshdeep Sekhon
Generalization Error

1. Generalization error = test error − training error
2. A network that generalizes well has comparable performance on the test and training set
3. $p >> n$ in neural networks, still low generalization error
4. Question: What makes a NN with good generalization different from one that generalizes poorly?
Traditional View of generalization

1 Model Family
2 Complexity Measures:
   1 Rademacher Complexity
   2 Uniform Stability
   3 VC dimension
3 Regularization
   1 Explicit Regularization: weight decay, dropout, etc
   2 Implicit Regularization: early stopping, batch norm, etc
Experiments with the following modifications of input and labeled data:

1. original data
2. partially corrupted labels: independently with probability $p$, the label of each image is corrupted as a uniform random class
3. Randomize labels completely: No relationship between data and labels
4. shuffled pixels: same random permutation of pixels to all images
5. Random Pixels: different random permutation of pixels to all images
6. Gaussian: Use gaussian to generate random pixels

Ideally, should affect training procedure as there is no relationship between input and output.
Results

Figure: Randomization tests results

1. Training Error zero: fits the data perfectly/Overfitting
2. No changes in training procedure
3. More corruption slows convergence
Implications

1. **Rademacher Complexity:**

\[
E_\sigma \left[ \sup_{h \in H} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(x_i) \right]
\]

(1)

where \(\sigma_1, \sigma_1, \sigma_1, \in +1, -1\) are iid random variables

Indicates how well a model in the hypothesis class fits a random assignment.
Implications

1. Rademacher Complexity:

\[
E_{\sigma}\left[ \sup_{h \in H} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(x_i) \right]
\]

(1)

where \( \sigma_1, \sigma_1, \sigma_1, \in [+1, -1] \) are iid random variables
Indicates how well a model in the hypothesis class fits a random assignment.

2. Because the NNs fit the training data perfectly, \( R(H) \approx 1 \). But, this is the upper bound for Rademacher complexity. Generalization is between zero and the worst case.
Implications

1. Rademacher Complexity:

\[ E_\sigma \left[ \sup_{h \in H} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(x_i) \right] \]  

where \( \sigma_1, \sigma_1, \sigma_1, \in +1, -1 \) are iid random variables

Indicates how well a model in the hypothesis class fits a random assignment.

2. Because the NNs fit the training data perfectly, \( R(H) \approx 1 \). But, this is the upper bound for Rademacher complexity. *Generalization is between zero and the worst case.*

3. Uniform Stability: Uniform stability of an algorithm A measures how sensitive the algorithm is to the replacement of a single example. A property of the algorithm/Has no relationship to data/distribution of labels
### Key Observations:

1. Even with regularization, networks generalize fine.
2. Even with regularization, training error is still zero: fit perfectly.

### Table: Training and Test Accuracy with Regularization

<table>
<thead>
<tr>
<th>Model</th>
<th># Params</th>
<th>Random Crop</th>
<th>Weight Decay</th>
<th>Train Accuracy</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>1,649,402</td>
<td>yes</td>
<td>yes</td>
<td>100.0</td>
<td>89.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>yes</td>
<td>no</td>
<td>100.0</td>
<td>89.31</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
<td>yes</td>
<td>100.0</td>
<td>86.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
<td>100.0</td>
<td>85.75</td>
</tr>
<tr>
<td>(fitting random labels)</td>
<td>no</td>
<td>no</td>
<td></td>
<td>100.0</td>
<td>9.78</td>
</tr>
<tr>
<td>Inception w/o BatchNorm</td>
<td>1,649,402</td>
<td>no</td>
<td>yes</td>
<td>100.0</td>
<td>83.00</td>
</tr>
<tr>
<td>(fitting random labels)</td>
<td>no</td>
<td>no</td>
<td></td>
<td>100.0</td>
<td>82.00</td>
</tr>
<tr>
<td>Alexnet</td>
<td>1,387,786</td>
<td>yes</td>
<td>yes</td>
<td>99.90</td>
<td>81.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>yes</td>
<td>no</td>
<td>99.82</td>
<td>79.66</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
<td>yes</td>
<td>100.0</td>
<td>77.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
<td>100.0</td>
<td>76.07</td>
</tr>
<tr>
<td>(fitting random labels)</td>
<td>no</td>
<td>no</td>
<td></td>
<td>99.82</td>
<td>9.86</td>
</tr>
<tr>
<td>MLP 3x512</td>
<td>1,735,178</td>
<td>no</td>
<td>yes</td>
<td>100.0</td>
<td>53.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
<td>100.0</td>
<td>52.39</td>
</tr>
<tr>
<td>(fitting random labels)</td>
<td>no</td>
<td>no</td>
<td></td>
<td>100.0</td>
<td>10.48</td>
</tr>
<tr>
<td>MLP 1x512</td>
<td>1,209,866</td>
<td>no</td>
<td>yes</td>
<td>99.80</td>
<td>50.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>no</td>
<td>no</td>
<td>100.0</td>
<td>50.51</td>
</tr>
<tr>
<td>(fitting random labels)</td>
<td>no</td>
<td>no</td>
<td></td>
<td>99.34</td>
<td>10.61</td>
</tr>
</tbody>
</table>

---

**Figure:** Regularization and Generalization
Implicit Regularization and Generalization

1. Early Stopping
2. Batch Normalization

Figure: Implicit Regularization

3. Continue to perform well without regularization
1. Regularization improves generalization ability.
2. Not the key reason for generalization.
Model Expressivity

1. Old/Previous View: What functions can be expressed by certain classes of neural networks?

2. Finite Sample Expressivity: Given n samples of d dimension, parameters required to express any function?
Theorem: Finite Sample Expressivity

Theorem:
There exists a two-layer neural network with ReLU activations and $2n + d$ weights that can represent any function on a sample of size $n$ in $d$ dimensions.

Proof:
Lemma 1:
For any interleaving sequences of $n$ real numbers, $b_1 < x_1 < b_2 < \cdots < b_n < x_n$, the $n \times n$ matrix $A = \max[x_i - b_j, 0]$ has full rank.

Proof:
Theorem: Finite Sample Expressivity

Consider the function:

$$c(x) = \sum_{j=1}^{n} w_j \left[ \max < a, x > - b_j, 0 \right]$$  \hspace{1cm} (2)

1. This can be expressed as a 2 layer ReLU network
2. $S = z_1, \ldots, z_n$
3. $x_i = < a, z_i >$
4. Choose $a, b$ such that the interleaving property $b_1 < x_1 < b_2 < \cdots, b_n < x_n$, is satisfied.
5. Reduces to $y = Aw$
6. because $A$ is invertible by the lemma,
7. Find suitable weights $w$
1. Traditional Views fail to explain generalization
2. Regularization methods are not sufficient or necessary for explaining generalization
3. Optimization is easy even if the resulting model does not generalize well