Over-parameterization as a **Catalyst** for Better Generalization of Deep ReLU network

Yuandong Tian

Research Scientist and Manager
Facebook AI Research
Great Empirical Success from Deep Models
How do deep models work?

Input

This is an apple

“Some Nonlinear Transformation”

Output
Three Major Problems

Understanding how Deep Models work

Expressibility

“Neural Network is a universal approximator”
“Deep Models can express functions more efficiently than shallow ones”

Optimization

“Gradient vanishing/exploding”
“Gradient Descent might get stuck at saddle point / local minima”
“Can GD/SGD go to global optima? How fast?”

Generalization

“Does zero training error often lead to overfitting?”
“More parameters might lead to overfitting.”
Supervised Learning

Dataset: \{ (x_i, y_i) \}

Student Network (Learnable Parameters)

Supervision
Student-Teacher Setting

Supervision

By Network Expressibility

Teacher Network (Fixed parameters)

Student Network (Learnable Parameters)

No direct supervision
Why Student-Teacher Setting?

Expressibility
- Provide a target function with bounded complexity

Optimization
- Study fine dynamics behaviors by comparing with teacher

Generalization
- Weight alignment with the teacher yields generalization

Understanding how Deep Models work
Old History of Teacher-Student Setting

\[ \epsilon(J, \xi) \equiv \frac{1}{2} \left[ \sigma(J, \xi) - \zeta \right]^2 = \frac{1}{2} \left[ \frac{1}{K} \sum_{i=1}^{K} g(x_i) - \frac{1}{M} \sum_{n=1}^{M} g(y_n) \right]^2 \]

One layer of trainable parameters

Use Gaussian erf() function as the nonlinearity

Study when the input dimension \( d \to +\infty \) (i.e., thermodynamics limits)

[On-line learning in soft committee machines, Saad & Solla, Phys. Rev 1995]
Student-Teacher Setting (this paper)

$$\min_{\mathcal{W}} J(\mathcal{W}) = \frac{1}{2} \mathbb{E}_x \left[ \| \mathbf{f}_L^*(x) - \mathbf{f}_L(x) \|^2 \right]$$

Teacher Network (Fixed parameters)

(Over-parameterized) Student Network (Learnable Parameters)

No direct supervision
Contributions

Over-parameterization helps in generalization in two ways:

1. Critical point analysis shows that over-parameterization helps student-teacher alignment.

2. Training dynamics analysis shows faster alignment with over-parameterization.
**Notation**

Layer $l$ ($n_l$ nodes)

Layer $l - 1$ ($n_{l-1}$ nodes)

$$f_{l,1}(x) \Rightarrow g_{l,1}(x)$$  

Weight update rule:  

$$\dot{W}_l = \mathbb{E}_x \left[ f_{l-1}(x) g_l^\top(x) \right]$$

**Activation**

$$f_l(x) = \begin{bmatrix} f_{l,1}(x) \\ f_{l,2}(x) \end{bmatrix}$$

**Gradient**

$$g_l(x) = \begin{bmatrix} g_{l,1}(x) \\ g_{l,2}(x) \end{bmatrix}$$

GD: expectation taken over the entire dataset  
SGD: expectation taken over a batch
A Trivial Statement

With over-parameterized student network:

Student aligns with the teacher

\[ g_l(x; \mathcal{W}) = 0, \quad \forall x \in R_0 \]
The Inverse Problem

With over-parameterized student network:

Student aligns with the teacher

\[ g_l(x; \mathcal{W}) = 0, \quad \forall x \in R_0 \]

→ Zero training error leads to good generalization
Lemma 1: Recursive Gradient Rule

For layer $l$, there exists $A_l(x)$ and $B_l(x)$ so that:

$$g_l(x) = D_l(x) \left[ A_l(x)f_l^*(x) - B_l(x)f_l(x) \right]$$

$A_l(x)$ and $B_l(x)$ are piece-wise constant.
Lemma 1: Recursive Gradient Rule

For layer $l$, there exists $A_l(x)$ and $B_l(x)$ so that:

$$g_l(x) = D_l(x) \left[ A_l(x)f_l^*(x) - B_l(x)f_l(x) \right]$$

$A_l(x)$ and $B_l(x)$ are piece-wise constant.
Recursive Formula for $A_l(x)$ and $B_l(x)$

\[ A_l(x) = V_l^T(x)V_l^*(x) \]
\[ B_l(x) = V_l^T(x)V_l(x) \]

Recursive Formula for $V$:

\[ V_{l-1}^*(x) = V_l^*(x)D_l^*(x)W_l^{*T} \]
\[ V_{l-1}(x) = V_l(x)D_l(x)W_l^T \]

Base case:

\[ V_L(x) = V_L^*(x) = I_{C \times C} \]
Main results: Alignment could happen!
Definition of Alignment

An example of “rough” alignment

$E_j$  Activated Region of node $j$

$\partial E_j$  Boundary of node $j$

$\partial E_k$  Boundary of node $k$
Assumption of the dataset

$\rho(x) > 0$

$R_0$

Infinite dataset!
Assumption of the dataset

\( R_0 \)

\( \rho(x) > 0 \)

Infinite dataset!
Assumptions on Teacher Network

• Cannot reconstruct arbitrary teachers
  • e.g., all ReLU nodes are dead

Distinct teacher nodes

Teacher’s boundary are visible in the dataset
Main results: Alignment could happen!

2-layer network

Layer 0

Layer 1

teacher $j$

student $k'$

observer $k$
Definition of “Observation”

Teacher $j$ is observed by a student $k$

$$\partial E_j^* \cap E_k \neq \emptyset$$

Teacher $j$ is observed by a student $k$
Main results: Alignment could happen!

Teacher node $j$ is **observed** by a student node $k$

Teacher $j$ is **aligned with** at least one student $k'$
Why?

The gradient of observer $k$ is 0:

From Lemma 1, $g_k(x) = \alpha_k^T f^*(x) - \beta_k^T f(x) = 0$

If $x \in E_k$
Why?

The gradient of observer $k$ is 0:

From Lemma 1, $g_k(x) = \alpha_k^T f^*(x) - \beta_k^T f(x) = 0$

If $x \in E_k$

ReLUs are linear independent!

Coefficients for teacher $j$ direction must be 0
Why?

The gradient of observer $k$ is 0:

From Lemma 1, $g_k(x) = \alpha_k^T f^*(x) - \beta_k^T f(x) = 0$

If $x \in E_k$

ReLUs are linear independent!

Coefficients for teacher $j$ direction must be 0

Teacher $j$ is aligned with at least one student $k'$ (sum of coefficients = 0)
Why Over-parameterization helps?

More observers!
What happens to unaligned students?

Aligned (can be one-to-many)
Simple 2D experiments

Iteration 0

Iteration 2

Student Boundary
Teacher Boundary
Simple 2D experiments
L-shape curve at convergence

10x over-parameterization

10x, loss=0.00

$\|v_k\|$ - norm of fan-out weights

Student nodes

Normalized correlation of a student node to its best correlated teacher
L-shape curve at convergence

1x, loss=0.00

2x, loss=0.00

5x, loss=0.00

10x, loss=0.00
Noisy Case  $\|g_1(x; \mathcal{W})\|_\infty \leq \epsilon$

For teacher $j$, there exists student $k'$:

weights \hspace{1cm} \sin \theta_{jk'} = \mathcal{O} \left( \frac{\epsilon^{1-\delta}}{|\alpha_{k,j}|} \right)

bias \hspace{1cm} |b^*_{j} - b_{k'}| = \mathcal{O} \left( \frac{\epsilon^{1-2\delta}}{|\alpha_{k,j}|} \right)
How to Prove?

Misalignment leads to small overlap
How to Prove?

Small overlap $\rightarrow$ There exists a datapoint that is far away from all boundaries.
How to Prove?

Pick three points $x_j, x_j^+, x_j^-$ and there will be one with $|g_j(x)| > \epsilon$, which is a contradiction.
Multi-Layer case: Alignment could happen!

\[ \alpha_k^T(x)f^*(x) - \beta_k^T(x)f(x) = 0 \]

Piece wise constant, apply the same logic per region!
For 2-layer:
$$\sqrt{\mathbb{E}_x [\beta_{kk}(x)]} = \| v_k \|$$
Different initialization, Similar Solutions

VGG-19 on CIFAR-100

Training Progresses

[All Neural Networks are Created Equal, Hacohen et al, 2019]
Solutions can be connected by line segments

[Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs, Garipov et al. NeurIPS 2018]
[Essentially No Barriers in Neural Network Energy Landscape, Draxler et al., 2018]
[Explaining Landscape Connectivity of Low-cost Solutions for Multilayer Nets, Kuditipudi et al., 2019]
Our Explanation

Student Solution 1

\[ \mathbf{v}_1 = 0 \]

Unaligned

\[ \mathbf{w}_1^* \quad \mathbf{w}_2^* \quad \mathbf{w}_3^* \]

Unaligned

Student Solution 2

\[ \mathbf{v}_1 = 0 \]

Linear segment

\[ \mathbf{w}_1^* \quad \mathbf{w}_2^* \quad \mathbf{w}_3^* \]

Linear segment

\[ \mathbf{w}_1^* \quad \mathbf{w}_2^* \quad \mathbf{w}_1^* \]
Training Dynamics

Critical Points have nice properties!

*Can we achieve that via training with SGD?*

*Not Easy*
Strong/weak teacher nodes

$\|v_{j1}\| \text{ large}$

$\|v_{j2}\| \text{ small}$

Strong teacher nodes are learned faster
1. Robust to Noise! 😊
2. Hard to learn weak teacher nodes 😞
Training Dynamics

Teacher $j$: $\|\mathbf{v}_j^*\| \propto 1/j^p$

Strong teacher node attracts many students!
Training Dynamics

Losing student node shifts focus.

Teacher $j$: $\|v_j^*\| \propto 1/j^p$
Successful Rate of Teacher Node Reconstruction

$p = 0.5$

$p = 1$

$p = 1.5$

$p = 2$

Successful Recovery Rate

Teacher $j$: $\|v_j^*\| \propto 1/j^p$

---

5 epochs

100 epochs

facebook Artificial Intelligence
Analysis of (approx.) Training Dynamics

For each node $k$, we have:

$$\dot{\mathbf{w}}_k = \|\mathbf{w}_k\| \mathbf{r}_k$$

Where:

$$\mathbf{r}_k = \sum_j \alpha_{jk} \psi(\theta_{jk}) \mathbf{w}_j^* - \sum_{k'} \beta_{k'k} \psi(\theta_{k'k}) \mathbf{w}_{k'} - \nu \mathbf{w}_k$$

$$\psi(\pi) = 0$$
Worst case scenario

\[ \dot{\theta} \propto -\psi(\theta) \sin \theta \propto - (\pi - \theta)^2 \]

\[ \theta_0 \rightarrow \pi, \quad t \rightarrow +\infty \]
How Over-parameterization can help?

More students yield better coverage.
Weak teacher nodes are Slow to train
CIFAR 10

1. Train a teacher network 64-64-64-64.
2. Then prune the teacher network with [0.3,0.5,0.5,0.7] rate.
3. Then train a student network to mimic teacher’s output (before softmax)
Hypothesis: What does a real dataset look like?

Teacher node (ordered by importance)

"Real salient teacher nodes every method gets it right"

"Weak teacher nodes that deep models get it right"

"noise, signals that do no show up in test set"

Early stopping

Evaluation loss
Some Evidences

[Do deep neural networks learn shallow learnable examples first? Mangalam et al, ICML 2019 Workshop]
Future Work

Empirical:
• Large Scale Experiments (ImageNet)
• Relate what analysis tells versus what we see empirically

Theoretical:
• Finite Sample Analysis to achieve a formal generalization bound
• Bottom-up Training Dynamics of deep ReLU networks
• Training Dynamics of student nodes competing against each other (Competitive Lotka–Volterra equations)
Building Scalable Systems for Reinforcement Learning

Presented by Yuandong Tian

Research Scientist and Manager
Facebook AI Research
Crash Course of Reinforcement Learning

State $s_t$ → Reward $r_t$ → Action $a_t$ → Environment → State $s_{t+1}$
Reinforcement Learning works, but expensive

<table>
<thead>
<tr>
<th>Year</th>
<th>Projects</th>
<th>Human Data</th>
<th>Training Resource</th>
<th>Training time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016</td>
<td>DeepMind’s AlphaGo</td>
<td>Yes</td>
<td>~50 GPUs + ? CPUs</td>
<td>~1 week</td>
</tr>
<tr>
<td>2017</td>
<td>DeepMind’s AlphaGo Zero (20 blocks)</td>
<td>No</td>
<td>~2000 TPUs</td>
<td>3 days</td>
</tr>
<tr>
<td>2017</td>
<td>DeepMind’s AlphaZero (20 blocks)</td>
<td>No</td>
<td>~5000 TPUs</td>
<td>8 hours</td>
</tr>
<tr>
<td>2018</td>
<td>OpenAI Five</td>
<td>No</td>
<td>128,000 CPUs + 256 GPUs</td>
<td>Several months</td>
</tr>
<tr>
<td>2019</td>
<td>DeepMind’s AlphaStar</td>
<td>Yes</td>
<td>16,000 CPUs + 3072 TPUv3 cores</td>
<td>44 days</td>
</tr>
</tbody>
</table>
Challenges in large-scale RL Training System

• Trade-offs in a *heterogenous* system
  • **Different kind of objects**: Actor / Environment / Trainer / Replay buffer
  • CPUs / GPUs Allocations
  • Multi-threading versus Multiple Processes, Batching issues
  • Local versus Distributed
  • Synchronization / Asynchronization.
    • On-policy versus off-policy methods
    • Perfect synchronization might NOT give you the best performance

• Mingled Algorithm Design and System Design
  • New System design ↔ New RL algorithm
Distributed System for training RL agent

GORILLA
[Distributed Prioritized Experience Replay, Horgan et al, ICLR 2018]
[Massively Parallel Methods for Deep Reinforcement Learning, AAAI 2015]

Ape-X / R2D2
[Recurrent Experience Replay in Distributed Reinforcement Learning, Kapturowski et al, ICLR 2019]

OpenAI Rapid
**ELF**: RL Framework for Game Research

```
while True:
    batched_states = GameContext.Wait()
    replies = model(batched_states)
    GameContext.Steps(replies)
```

AlphaGoZero / AlphaZero

Generate Training data

Without human knowledge

$\theta_i$ Update Models

Self-Replays

[Silver et al, Mastering the game of Go without human knowledge, Nature 2017]
Generate Self-play Games

Monte Carlo Tree Search with current model $\theta_i$

Training samples for $\theta_{i+1}$
Update Models

Input features (19x19x17): \((X, Y, X_{-1}, Y_{-1}, \ldots, X_{-7}, Y_{-7}, C)\)

Objective:

\[
J(\theta) = (z - V_{\theta})^2 - \pi^T \log p_{\theta} + c\|\theta\|^2
\]
AlphaGo Zero Strength

• 3 days version
  • 4.9M Games, 1600 rollouts/move
  • 20 block ResNet
  • Defeat AlphaGo Lee.

• 40 days version
  • 29M Games, 1600 rollouts/move
  • 40 blocks ResNet.
  • Defeat AlphaGo Master by 89:11
The Mystery of AlphaZero

• Mystery
  • Is the proposed algorithm really universal?
  • Is the bot almighty? Is there any weakness in the trained bot?

• Lack of Ablation Studies
  • What factor is critical for the performance?
  • Is the algorithm robust to random initialization and changes of hyper parameters?
  • Any adversarial samples?

Impressive Results, No code, No model
ELF OpenGo

- System can be trained with 2000 GPUs in 2 weeks (20 block version)
- Superhuman performance against professional players and strong bots.
- Abundant ablation analysis.

We open source the code and the pre-trained model for the Go and ML community

[ELF OpenGo: An Analysis and Open Reimplementation of AlphaZero, Y. Tian et al, ICML 2019]
ELF OpenGo Performance

Vs top professional players

<table>
<thead>
<tr>
<th>Name</th>
<th>ELO (world rank)</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kim Ji-seok</td>
<td>3590 (#3)</td>
<td>5-0</td>
</tr>
<tr>
<td>Shin Jin-seo</td>
<td>3570 (#5)</td>
<td>5-0</td>
</tr>
<tr>
<td>Park Yeonghun</td>
<td>3481 (#23)</td>
<td>5-0</td>
</tr>
<tr>
<td>Choi Cheolhan</td>
<td>3466 (#30)</td>
<td>5-0</td>
</tr>
</tbody>
</table>

Single GPU, 80k rollouts, 50 seconds
Offer unlimited thinking time for the players

Vs professional players

Single GPU, 2k rollouts, 27-0 against Taiwanese pros.

Vs strong bot (LeelaZero)

[158603eb, 192x15, Apr. 25, 2018]: 980 wins, 18 losses (98.2%)
Distributed ELF (version 1, AlphaGoZero)

Selfplay 1 → Selfplay 2 → ... → Selfplay n

Selfplay data

Training procedure (8 GPUs)

Current trained model

Model Zoo

Current best model

Evaluation 1 → Evaluation 2 → ... → Evaluation m

Win rate > 55%

Update best model and next candidate
Distributed ELF (version 1)

Training procedure

Selfplay 1

Selfplay 2

Selfplay n

Open a port
Receive selfplay data via ZeroMQ

Current trained model

Model Zoo

Evaluation Server

Evaluation 1

Evaluation 2

Evaluation m

8 GPUs
Distributed ELF (version 1)

Training procedure (8 GPUs)

300-2k GPUs

Selfplay 1 → Selfplay 2 → ⋯ → Selfplay n

Pick the best model and keep selfplaying

Current best model

Each selfplay client batches 32 parallel games in a batch size of 128

Evaluation 1 → Evaluation 2 → ⋯ → Evaluation m

Model Zoo

facebook Artificial Intelligence
Distributed ELF (version 1)

Training procedure (8 GPUs) → Model Zoo

Selfplay 1 → Selfplay 2 → ... → Selfplay n

Evaluation 1 → Evaluation 2 → ... → Evaluation m

No GPU needed

Update best model and next candidate

Win rate > 55%
Distributed ELF (version 1)

Selfplay 1 ➔ Selfplay 2 ➔ ... ➔ Selfplay n

Training procedure (8 GPUs) ➔ Model Zoo

Evaluation Server ➔ Evaluation 1 ➔ Evaluation 2 ➔ Evaluation m

Send the current model pairs to evaluate

Each evaluation client batches 2 parallel games

100 GPUs
Distributed ELF (v2)

Putting AlphaGoZero and AlphaZero into the same framework

- AlphaGoZero (more synchronization)
- AlphaZero (less synchronization)

Server controls synchronization
Server also does training.
Next Step: RL Assembly

• Backbone infrastructure for ongoing projects (Hanabi, Bridge, etc)
• Reimplementation of SoTA off-policy RL methods like Ape-X and R2D2
• Incorporate OpenGo and SoTA implementation of MCTS.
• Efficient on single machine (SoTA training FPS so far)

Open source soon
Current Projects using ReLA

Contract Bridge

Hanabi

MiniRTSv2

More projects to come!
Thanks!