How to Start Training: The Effect of Initialization and Architecture

Texas A & M University / MIT

NeurIPS 2018

Review: Well-known initialization methods

- Xavier (Glorot) initialization
  - When using Sigmoid or Tanh
  - $W \sim N\left(0, \frac{1}{n_{in}+n_{out}}\right)$ or $W \sim U\left(-\sqrt{\frac{6}{n_{in}+n_{out}}}, +\sqrt{\frac{6}{n_{in}+n_{out}}}\right)$

Variance: $\frac{1}{n_{in}+n_{out}}, \frac{2}{n_{in}+n_{out}}$

- He (Kaiming) initialization
  - When using ReLU
  - $W \sim N\left(0, \frac{2}{n_{in}}\right)$ or $W \sim U\left(-\sqrt{\frac{6}{n_{in}}}, +\sqrt{\frac{6}{n_{in}}}\right)$

Variance: $\frac{2}{n_{in}}, \frac{2}{n_{in}}$

* Variance of $U(a, b)$ is $\frac{1}{12}(b-a)^2$

PyTorch

- Default initialization method is He (Kaiming) uniform
  - Convolution, Linear, …
  - https://github.com/pytorch/pytorch/blob/master/torch/nn/modules/conv.py
  - https://github.com/pytorch/pytorch/blob/master/torch/nn/modules/linear.py
  - See method “reset_parameters” of each class
  - https://discuss.pytorch.org/t/whats-the-default-initialization-methods-for-layers/3157/20

TL; DR

- Two failure mode in training neural network
  - FM1: The mean length scale in the final layer increases / decreases exponentially with the depth
  - FM2: The empirical variance of length scales across layers grows exponentially with the depth

TL; DR

- Conclusions
  - The mean and variance of activations in a neural network are both important in determining whether training begins
  - FM1 is dependent on weight initialization
  - For fully connected and convolutional networks, FM2 is dependent on architecture
  - For residual networks, FM2 is largely independent of the architecture

Observation: fc-nets

- What is “length scale”?
  - Norm of activation vector

$$M_j := \frac{1}{n_j} \left\| act^{(j)} \right\|^2$$

Where j is the index of layer and n_j is the width of j-th layer in fully-connected ReLU network

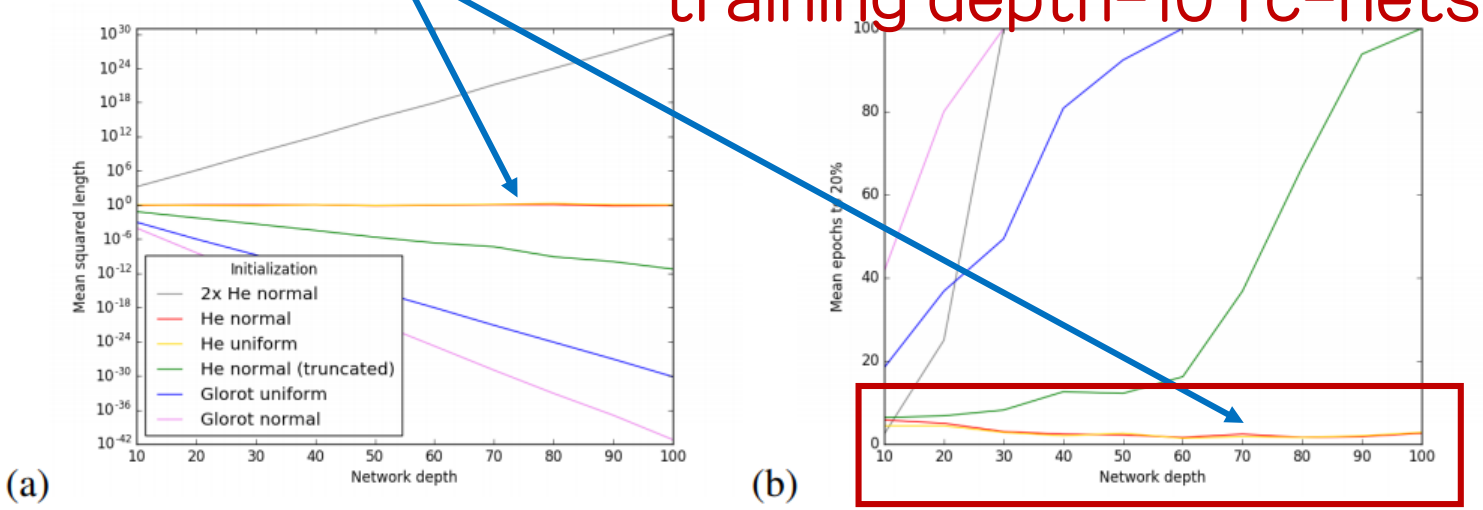
→ FM1: The mean length scale in the final layer increases / decreases exponentially with the depth

(a) 1,000 repeated experiments per point
(b) 5 repeated experiments per point

Observation: fc-nets

Is $\frac{2}{n_{in}}$ a good variance?

Training depth-100 fc-nets is faster than training depth-10 fc-nets!



최신 버전은 안그럼

Figure 1: Comparison of the behavior of differently initialized fully connected networks as depth increases. Width is equal to depth throughout. Note that in `He normal (truncated)`, the normal distribution is truncated at two standard deviations from the mean, as implemented e.g. in Keras and PyTorch. For `2x He normal`, weights are drawn from a normal distribution with twice the variance of `He normal`. (a) Mean square length M_d (log scale), demonstrating exponential decay or explosion unless variance is set at $2/\text{fan-in}$, as in `He normal` and `He uniform` initializations; (b) average number of epochs required to obtain 20% test accuracy when training on vectorized MNIST, showing that exponential decay or explosion of M_d is associated with reduced ability to begin training.

* Width is equal to depth in this experiment

Observation: Residual networks

- We can define residual networks by recursion

$$f_0(x) = x, \qquad f_l(x) = f_{l-1}(x) + \eta_l f_l\big(f_{l-1}(x)\big), \qquad l = 1, \dots, L$$

  - Where η_l is "scale" and we can also consider sequences $\{\eta_l\}$
- Mean length scale of last residual layer: $M_L = \|f_L(x)\|^2$

  → FM1: The mean length scale in the final layer increases / decreases exponentially *with the depth*
  - Grows exponentially with the sum of the scales $\sum_{l=1}^{L} \eta_l$

Observation: Residual networks

$$\eta_l = b^l, \qquad b \in \{1, 0.9, 0.75, 0.5\}$$

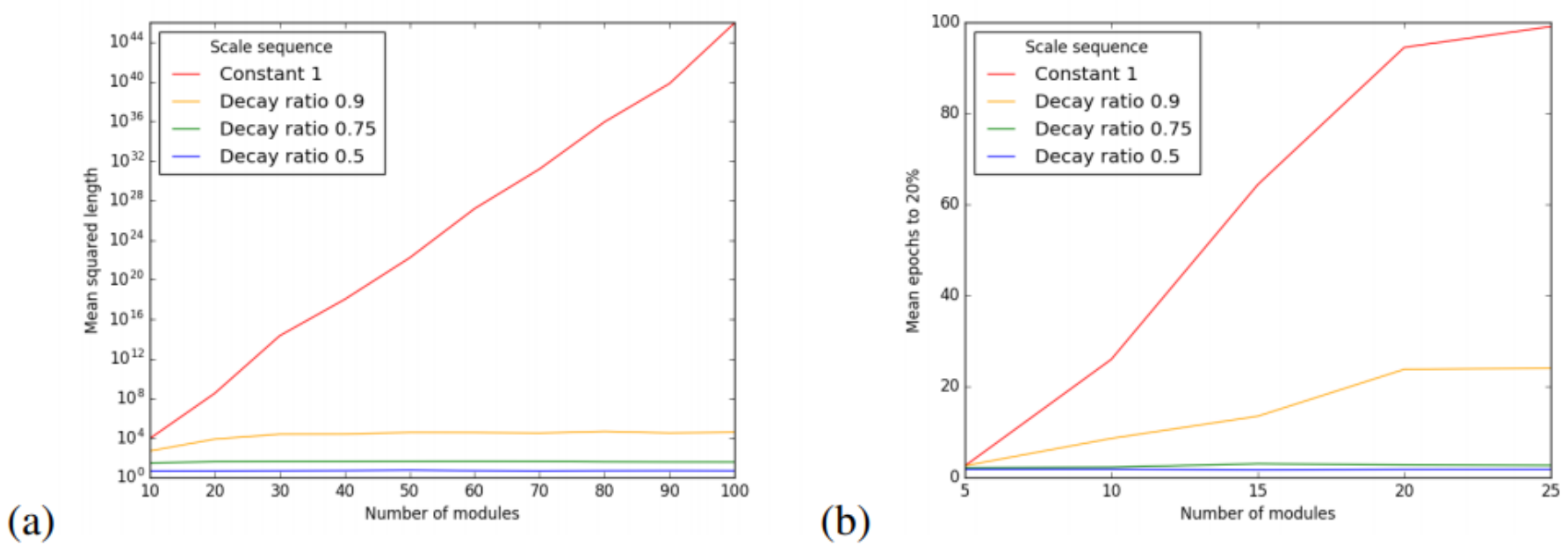
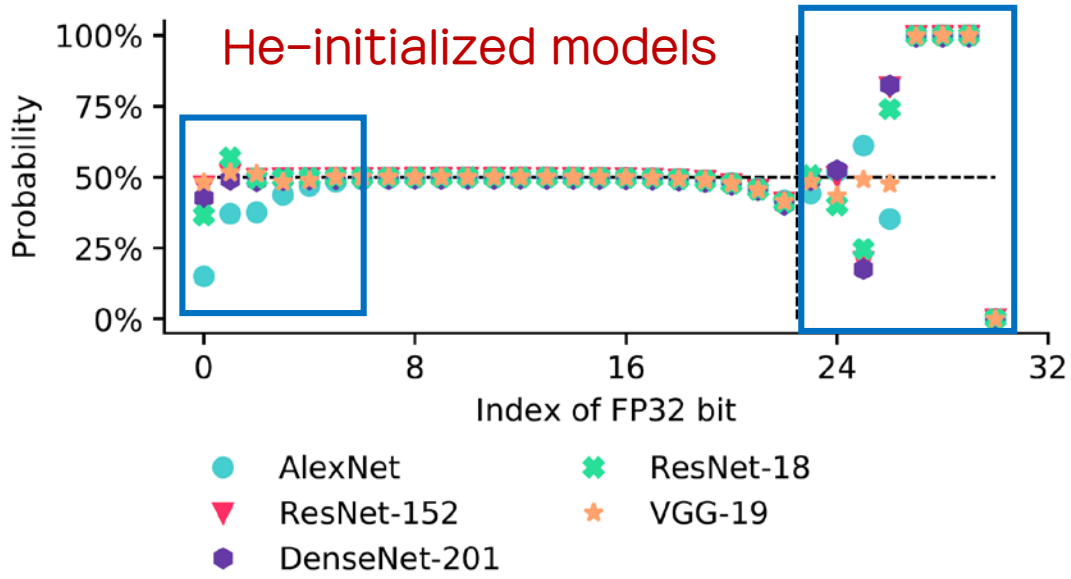


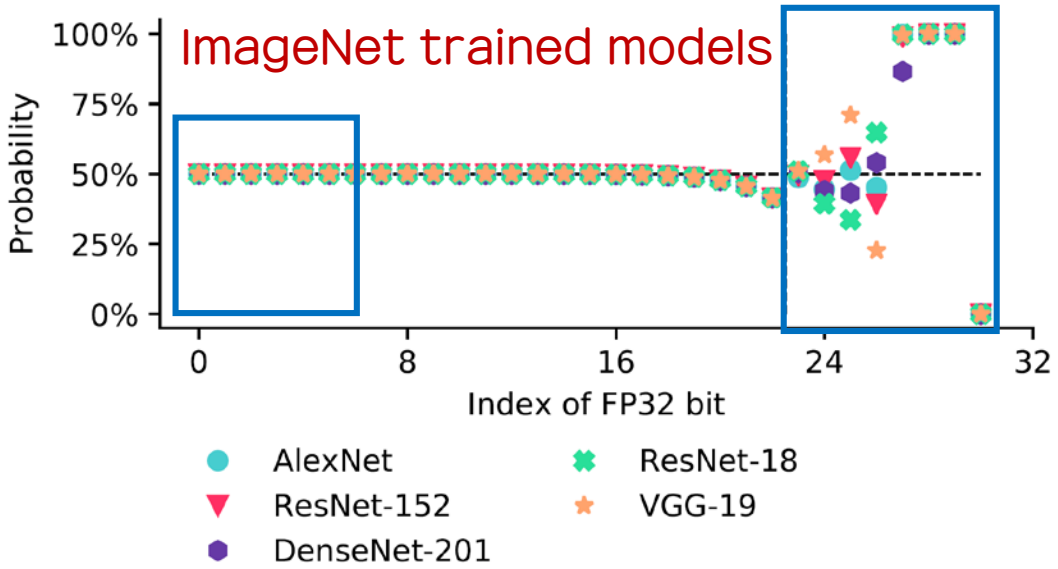
Figure 2: Comparison of the behavior of differently scaled ResNets as the number of modules increases. Each residual module here is a single layer of width 5. (a) Mean length scale M_L^{res}, which grows exponentially in the sum of scales η_ℓ; (b) average number of epochs to 20% test accuracy when training on MNIST, showing that M_L^{res} is a good predictor of initial training performance.

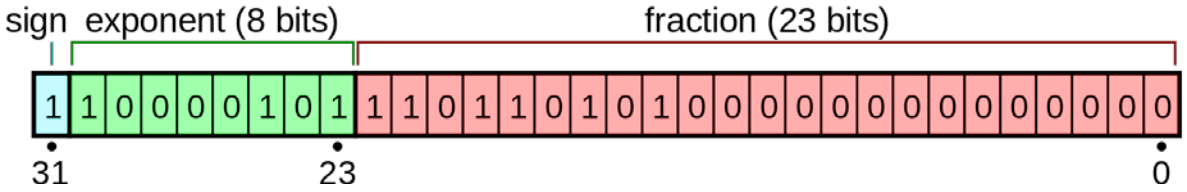
* Experimented with He-initialized ResNets, repeated 100 times

Bit-level view of NN parameters

- FM2 는 생략…
  - I apologize… To show you this, I've focused your attention…







$= -118.625 = (-1)^{1_2} \times 1.110110101\cdots_2 \times 2^{10000101_2}$

Back to the initialization

- Starting from the curiosity
  - 초기 분포를 학습된 모델의 분포와 완벽하게 맞춰 초기화한다면 어떻게 될까?
  - i.i.d 조건을 만족시키기는 쉽지 않음
  - NN parameters (slightly) tend to follow i.i.d random variables…

Layer type		# of layers	Max. Z-score Frac. 0-7	Max. Z-score Frac. 8-15	Max. Z-score Frac. 16-22	Avg. Z-score Image
Conv	Weight	20	3.877	4.528	3.943	626.747
FC	Weight	1	1.053	0.653	2.083	
	Bias	1	0.744	1.645	0.327	
BN	Weight	20	2.109	1.992	2.478	
	Bias	20	2.147	2.004	2.048	

Table 3: Result of runs tests on ResNet-18 parameters and Imagenet validation images. A high Z-score can be interpreted as an evidence that the corresponding sequence of the variables are not i.i.d.

Back to the initialization

- 아키텍쳐마다 학습했을 때 나타나는 고유한 분포가 있는가?
- 학습 이후 나타나는 분포는 데이터셋과 무관한가?
- 아키텍쳐별 고유한 분포로 초기화를 한 모델의 초기 성능은 어떤가?
- 아키텍쳐별 고유한 분포로 초기화한 모델의 학습 속도는 어떤가?
- 기존 방법 (xavier, he 등) 으로 초기화한 모델의 파라미터 분포는 언제부터 아키텍쳐별 고유 분포를 따르는가?
- MLM 등의 많이 쓰이는 pre-training method 를 대체할 수 있는가?

Question

1. 학습되지 않은 모델의 파라미터와 학습된 모델의 파라미터를 구분할 수 있는가?

2. 구분할 수 있다면, 학습된 모델의 파라미터를 바로 생성할 수 있는가?

3. 그렇다면 자기 자신의 기능과 동일한 기능을 하는 모델의 파라미터를 생성할 수 있는가? (Neural Quine)
   - Set of parameters, Cyclic group, …

4. Program synthesis 와 비슷하게 Neural net synthesis 가 가능한가?
   - Different from *Neural Architecture Search (NAS)*