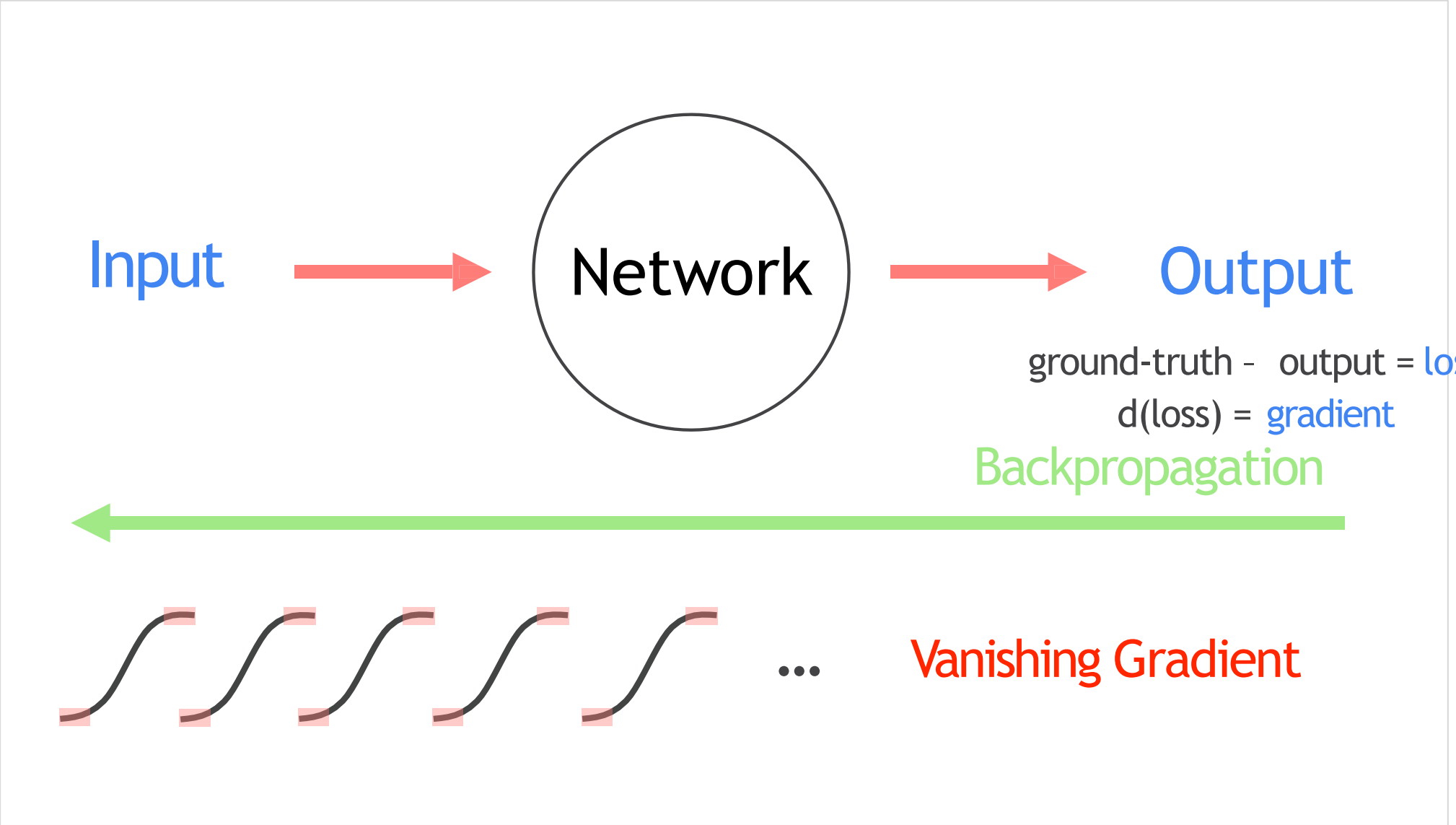

Go back in time
2000s

Geofferey Hinton’s summary of findings up to today

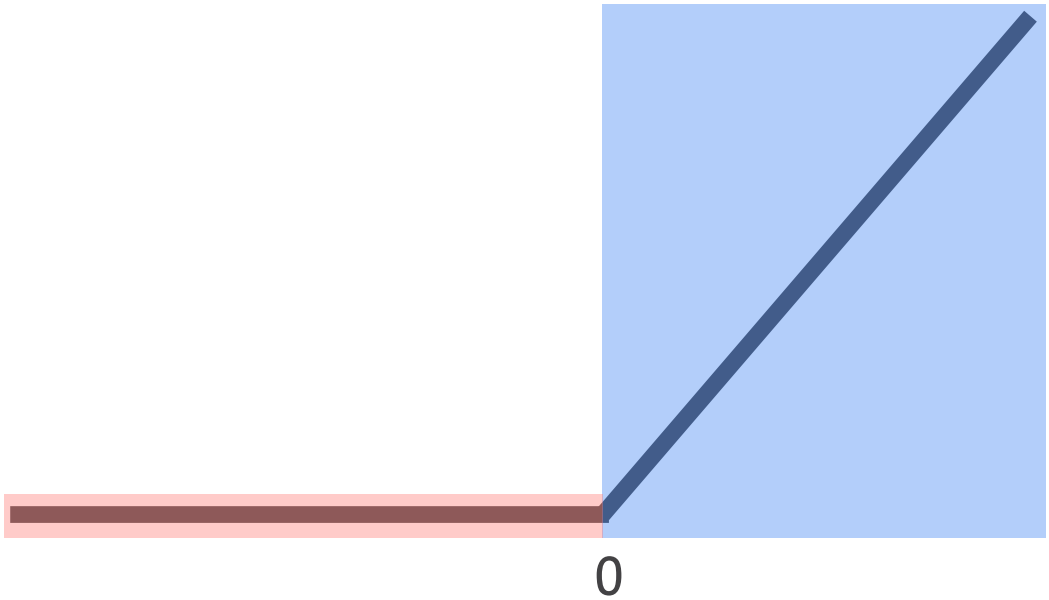
- Our labeled datasets were thousands of times too small
- Our computers were millions of times too slow
- We initialized the weights in a stupid way
- We used the wrong type of non-linearity

Problem of Sigmoid



Why Relu ?

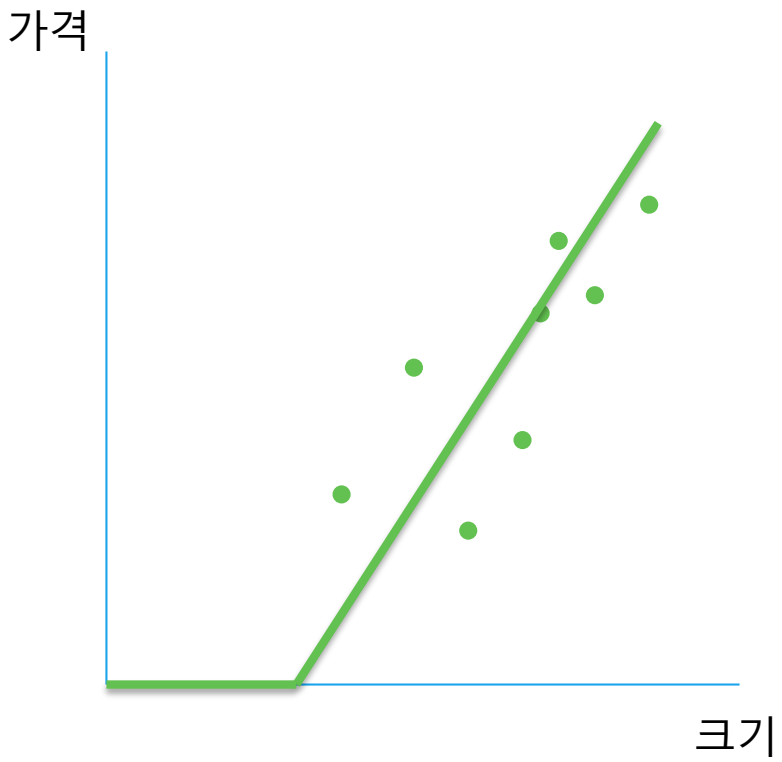
$$f(x) = \max(0, x)$$



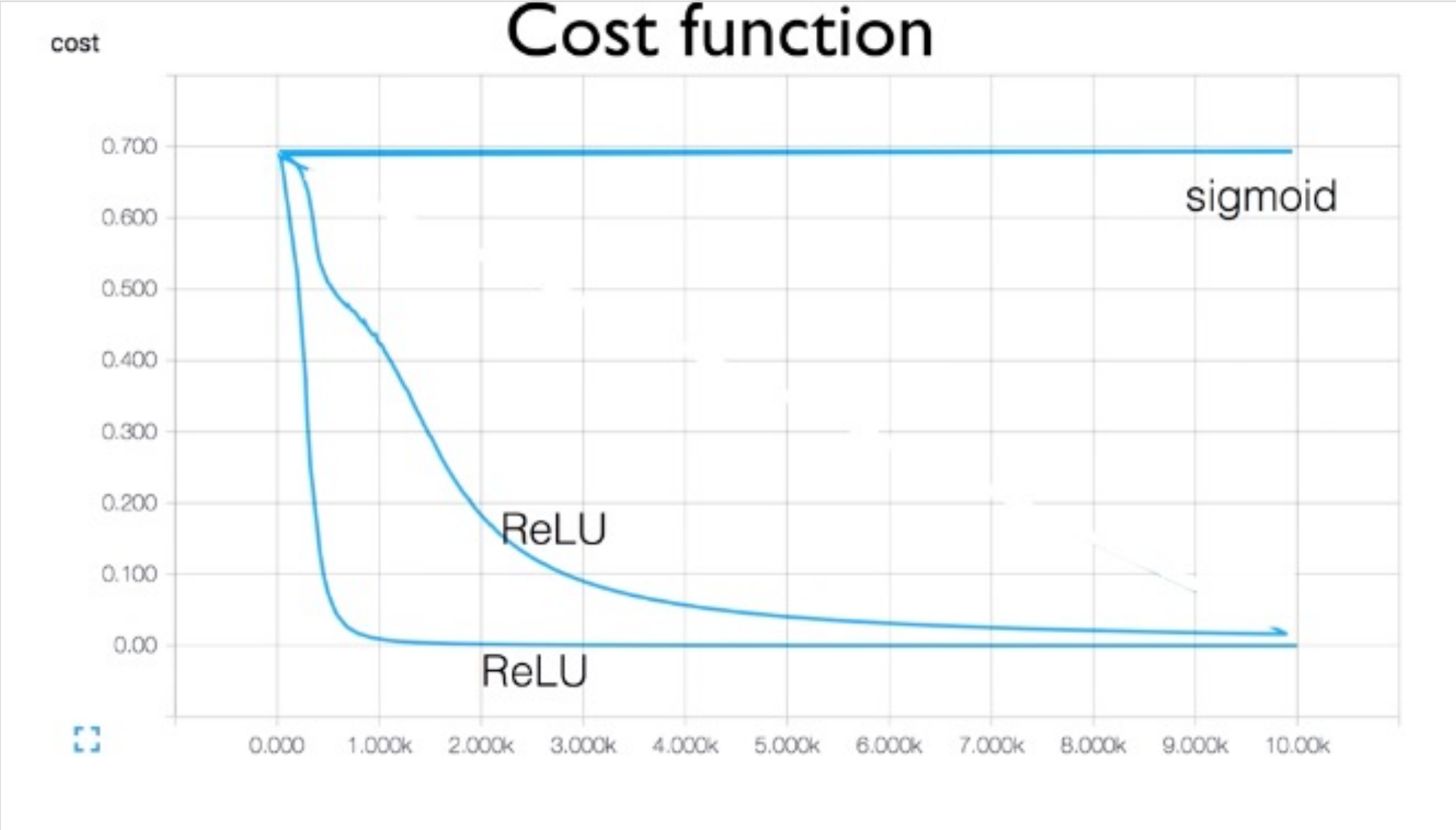
tf.keras.activations → sigmoid, tanh, relu, elu, selu

tf.keras.layers → leaky relu

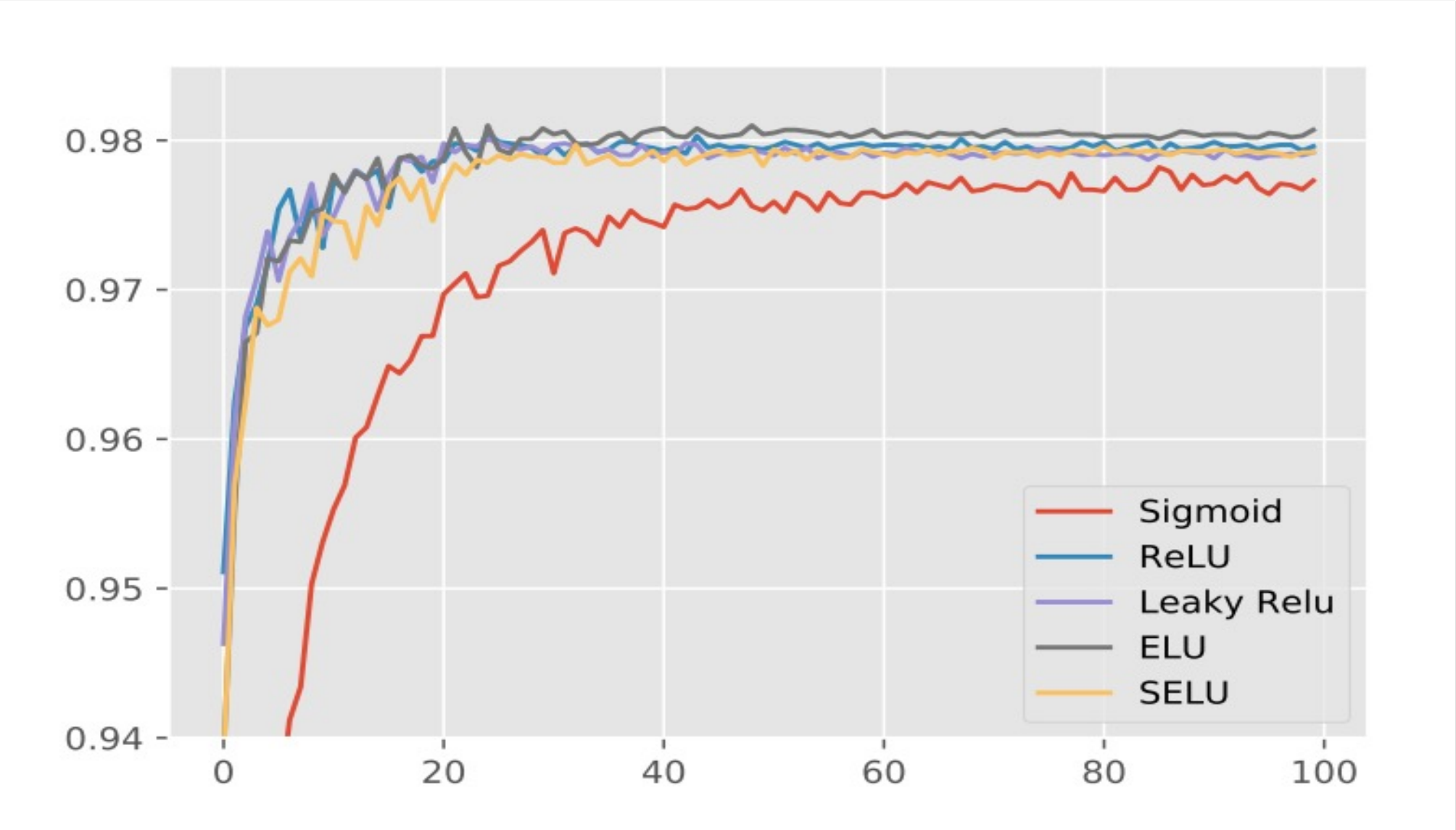
ReLU?



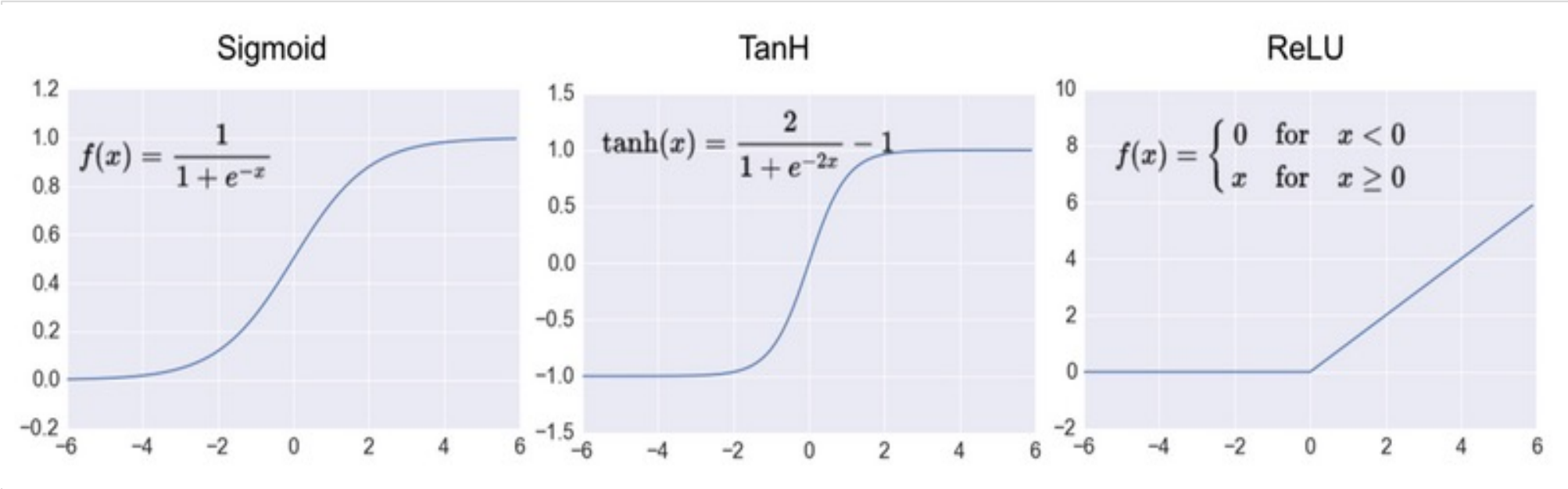
Sigmoid vs ReLU

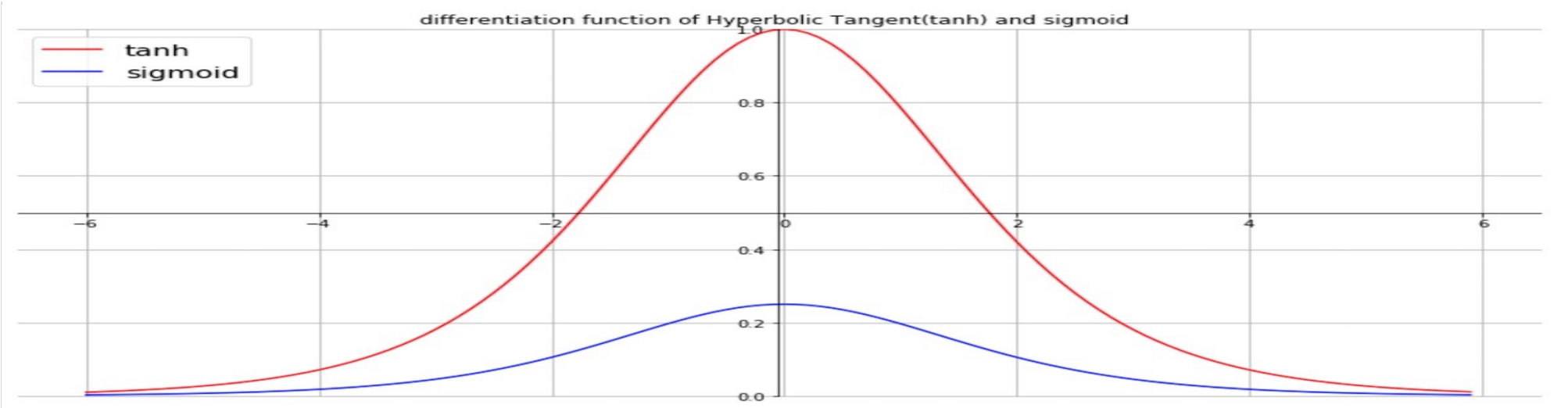


Cost function



Activation function



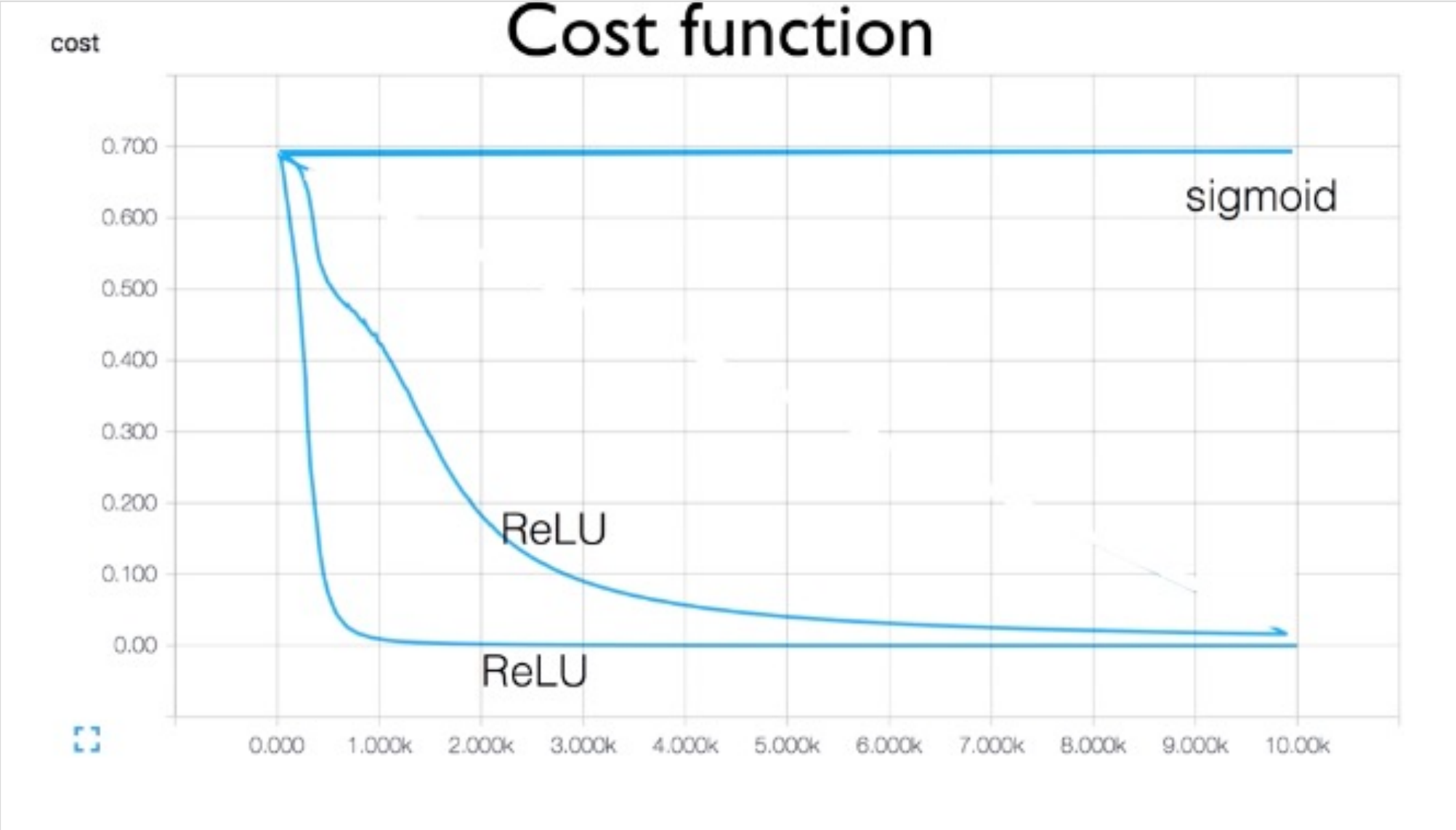


PRACTICE

Geofferey Hinton's summary of findings up to today

- Our labeled datasets were thousands of times too small
- Our computers were millions of times too slow
- We initialized the weights in a stupid way
- ~~We used the wrong type of non-linearity~~

Same ReLU but different speed.. Why?



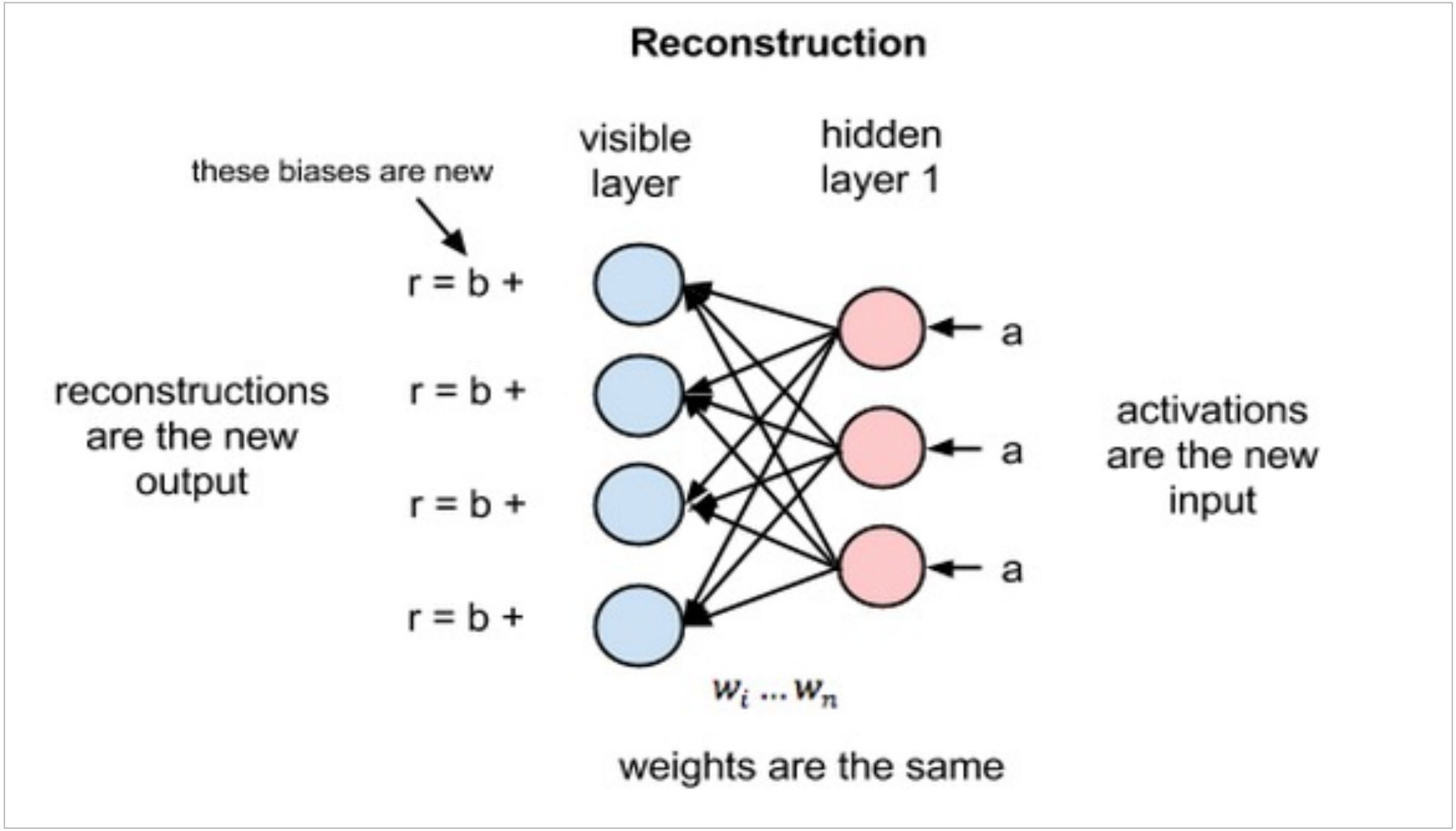
Breakthrough

- In 2006, Hinton, Simon Osindero, and Yee-Whye Teh published, "A fast learning algorithm for deep belief nets"
- Yoshua Bengio et al. in 2007 with "Greedy Layer-Wise Training of Deep Networks"

Breakthrough in 2006 and 2007 by Hinton and Bengio

- Neural networks with many layers really could be trained well, if the weights are initialized in a clever way rather th an randomly.
- Deep machine learning methods are more efficient for difficult problems than shallow methods.
- Rebranding to ***Deep Nets***, ***Deep Learning***

Greedy Layer-Wise Training

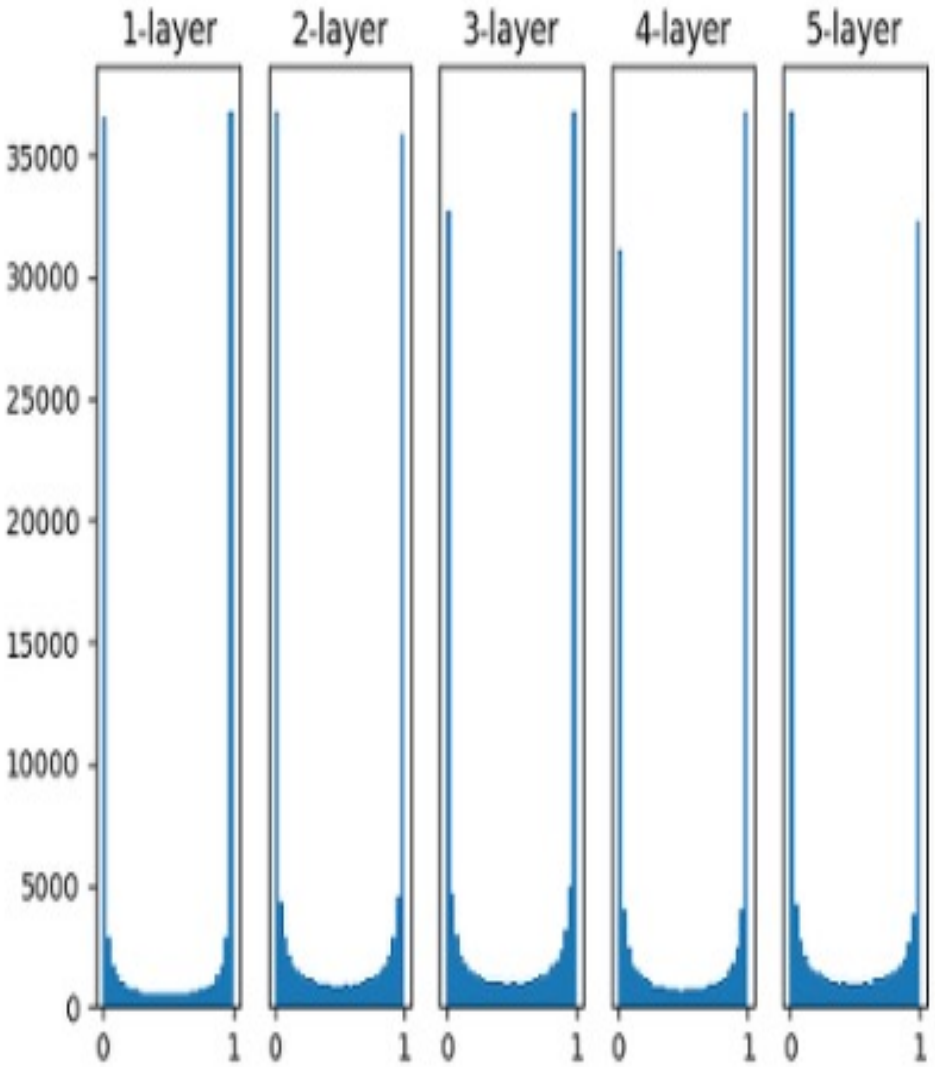


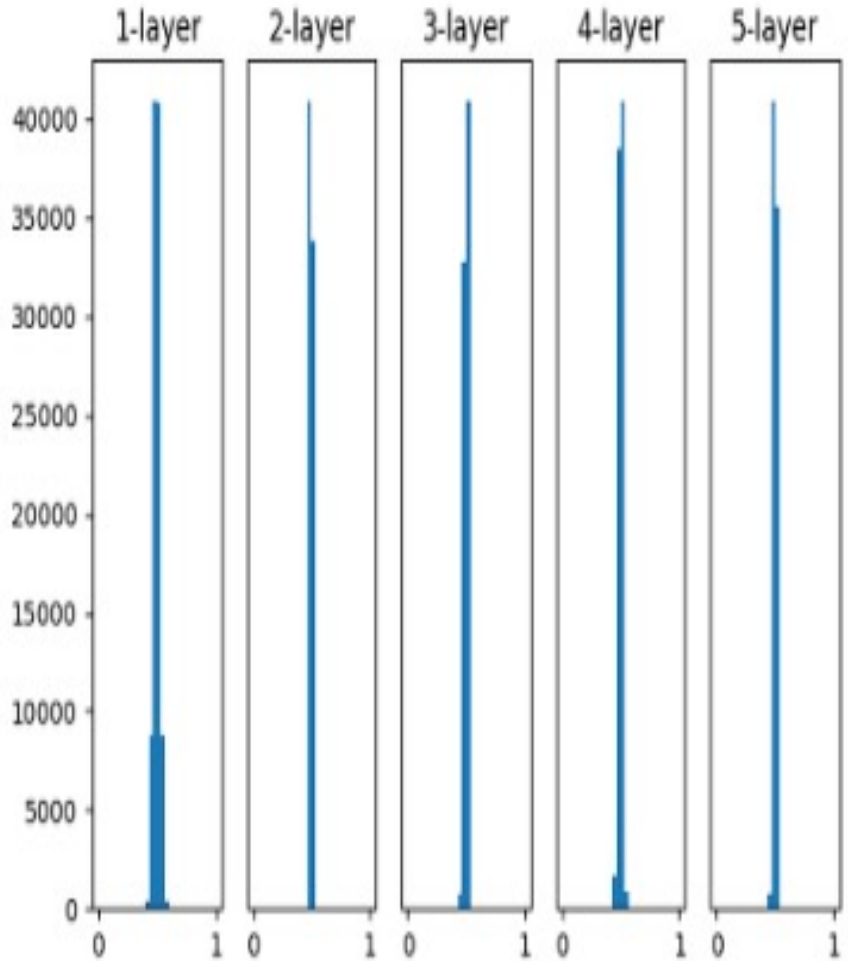
DBN 사용 vs 미사용 모델 성능 비교 논문 결과

Table 1. Algorithm accuracy (%)

Algorithm	Accuracy 1	Accuracy 2	Accuracy 3	Accuracy 4	Accuracy 5	Mean Accuracy
DBN	95.00	90.00	95.00	98.75	91.25	94.00
BP	50.00	25.00	50.00	50.00	25.00	40.00
P-DBN	98.75	92.50	98.75	91.25	92.50	94.75
P-BP	50.00	50.00	25.00	50.00	26.25	40.25

Gradient vanishing vs Gaussian mixture weight initialization

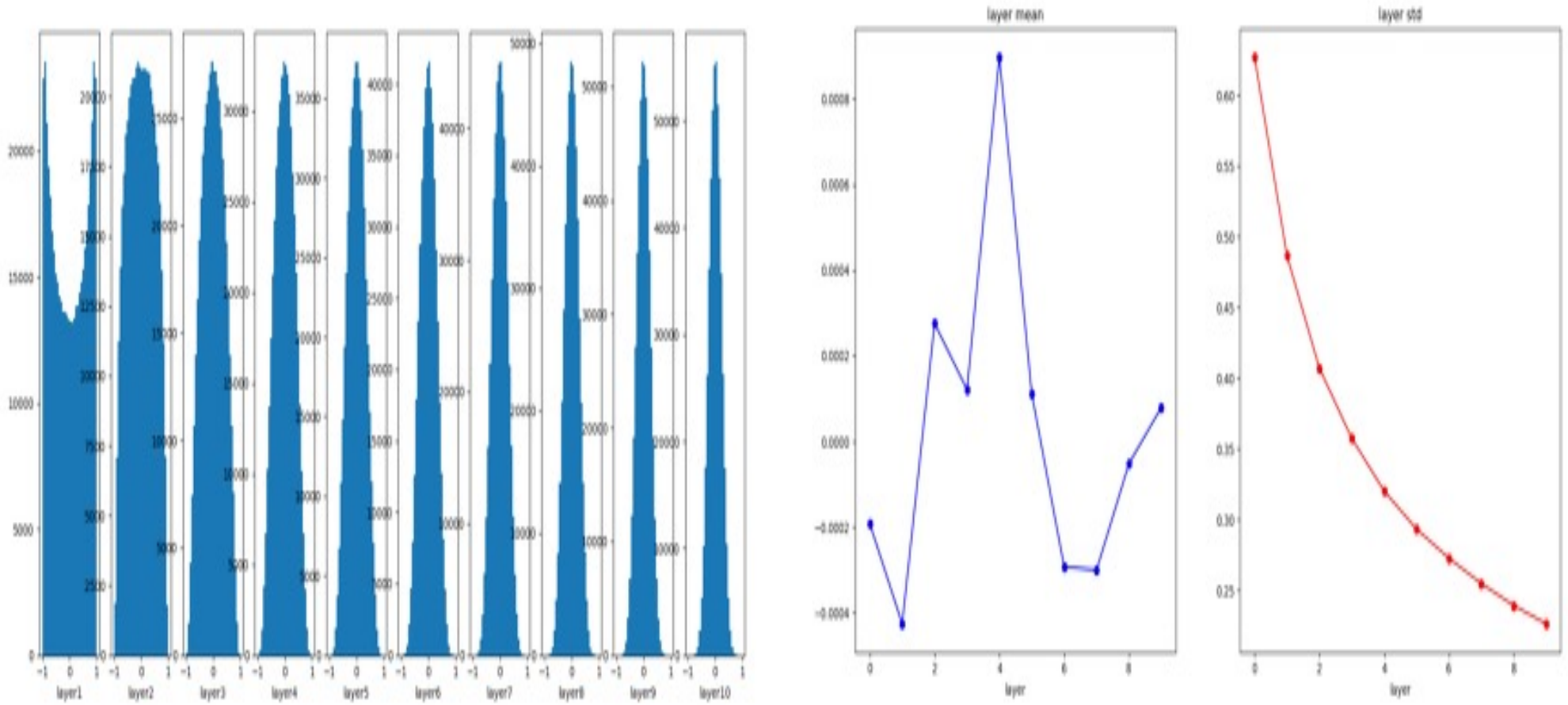




출처: Standford university

Xavier Initialization

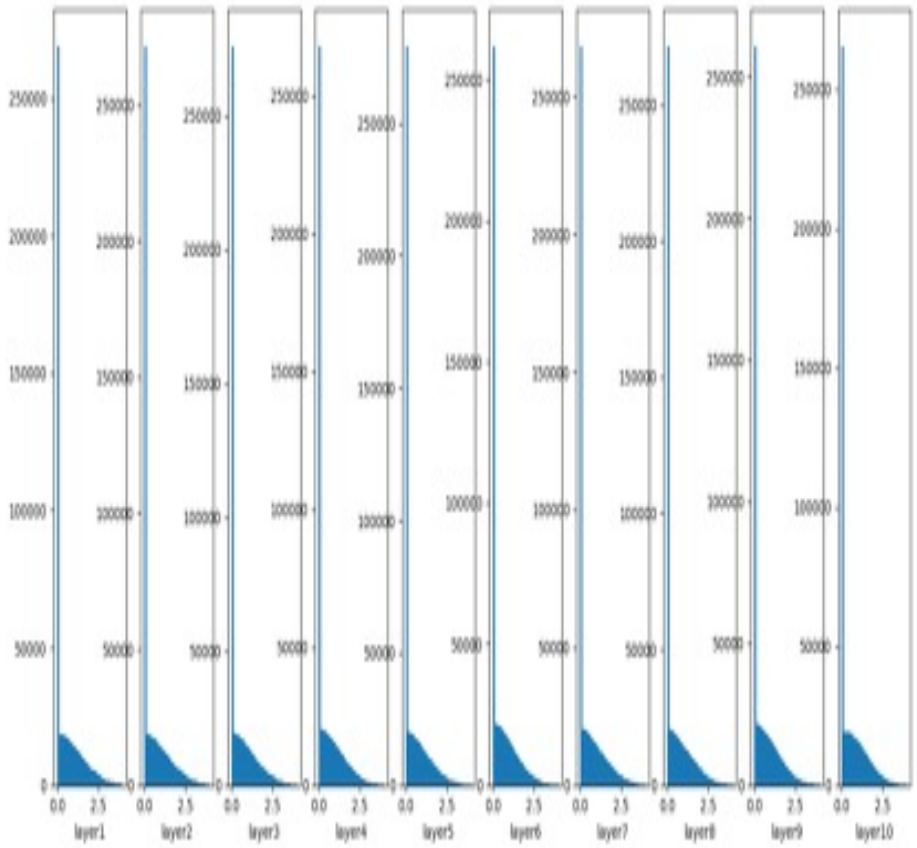
w = np.random.randn(n_input, n_output) / sqrt(n_input)

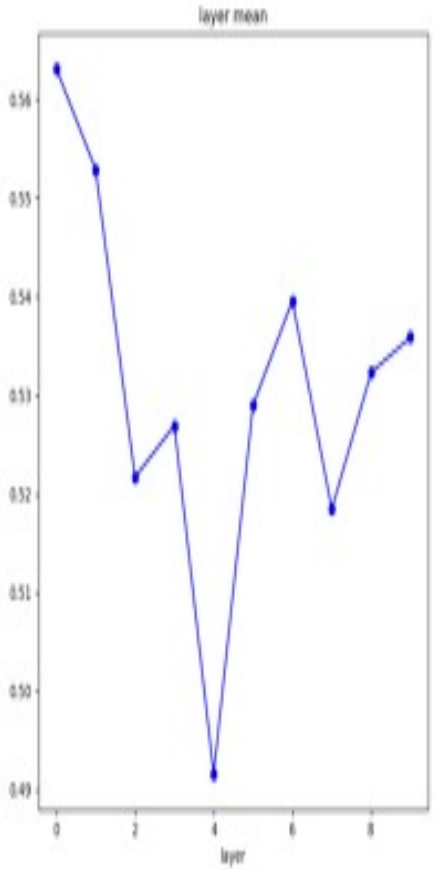


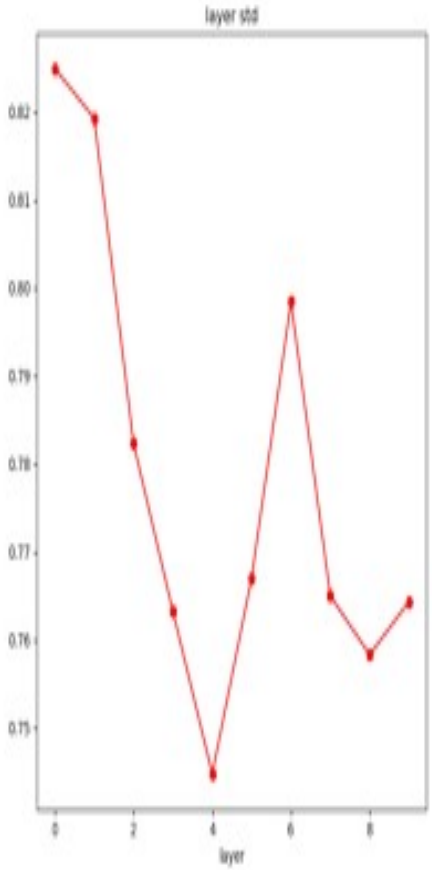
출처: Standford university

He Initialization

w = np.random.randn(n_input, n_output) / sqrt(n_input/2)

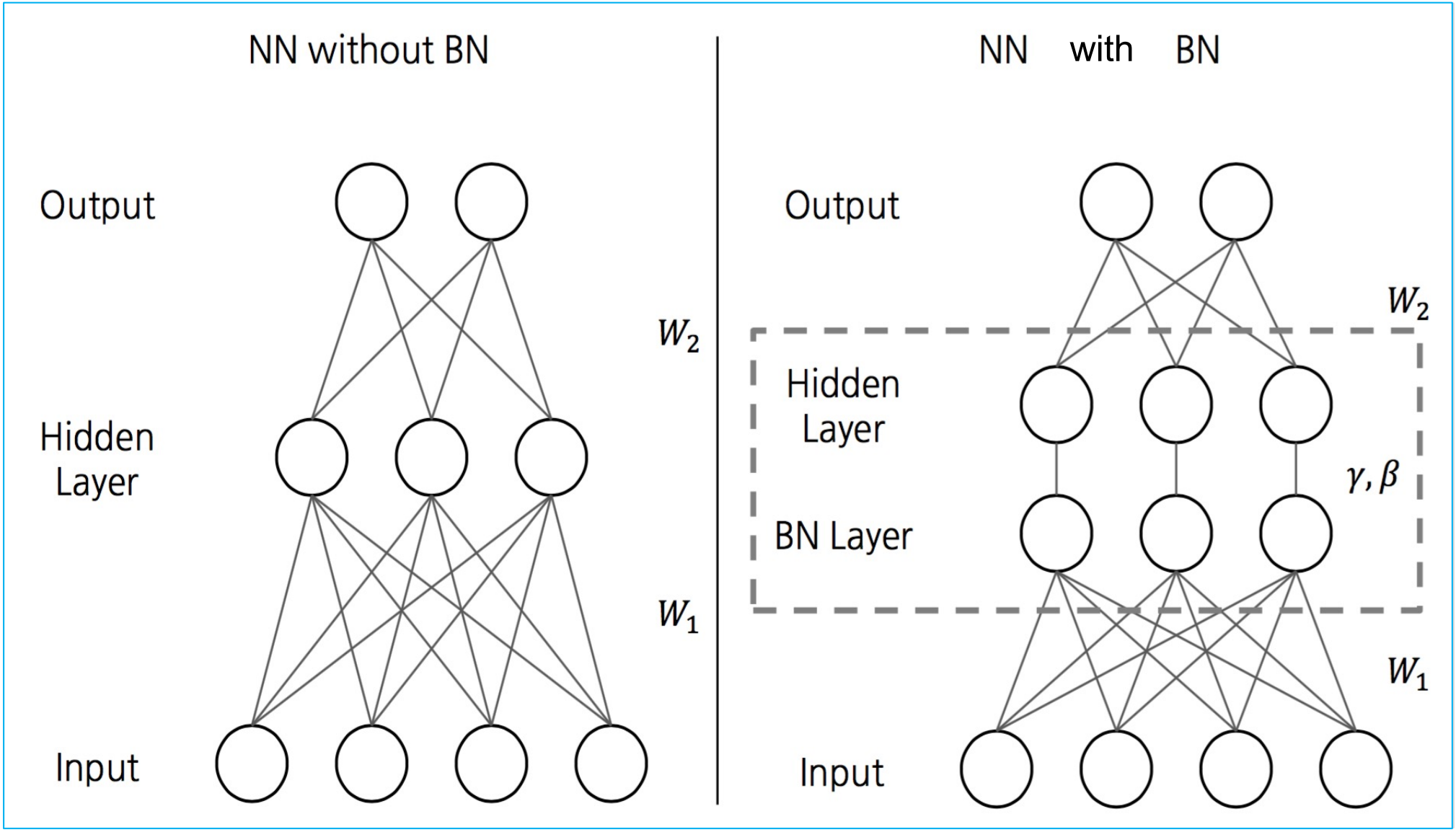




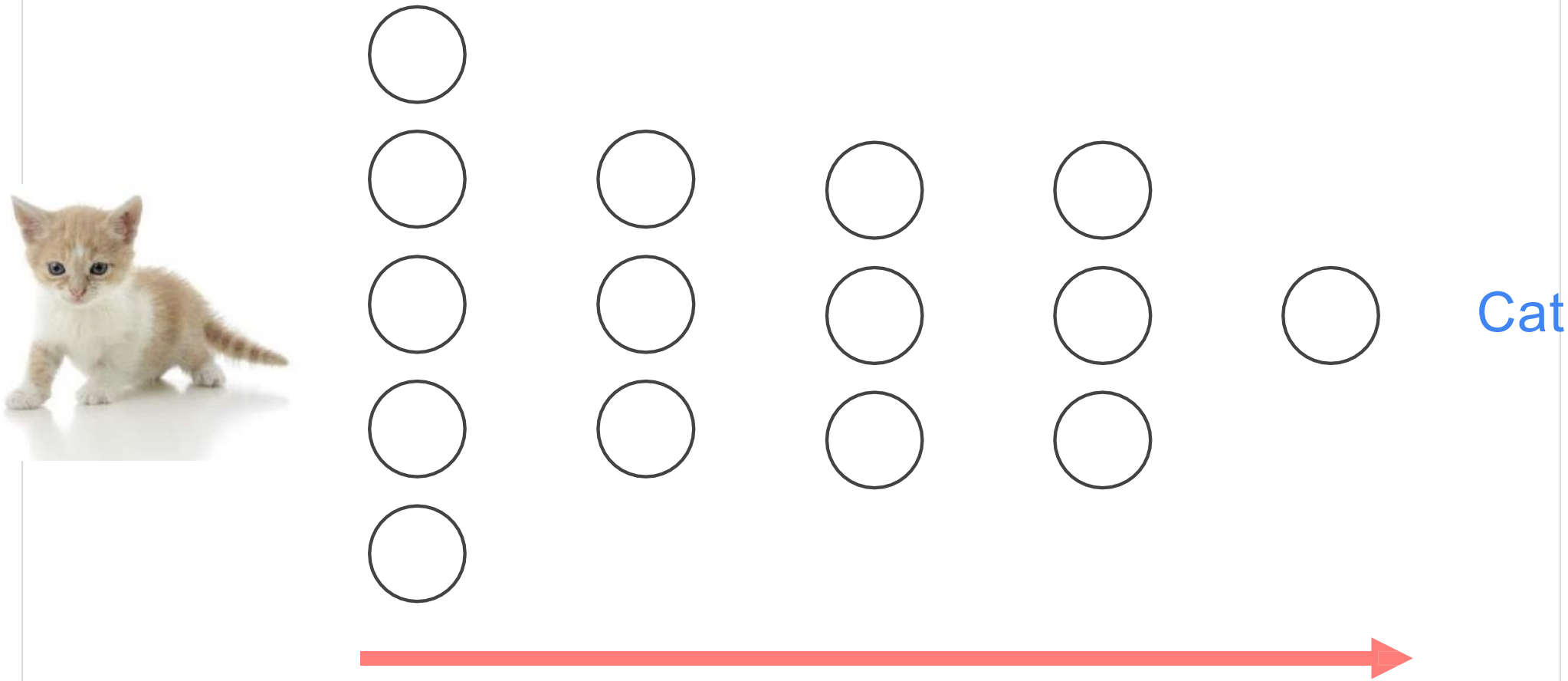


출처: Standford university

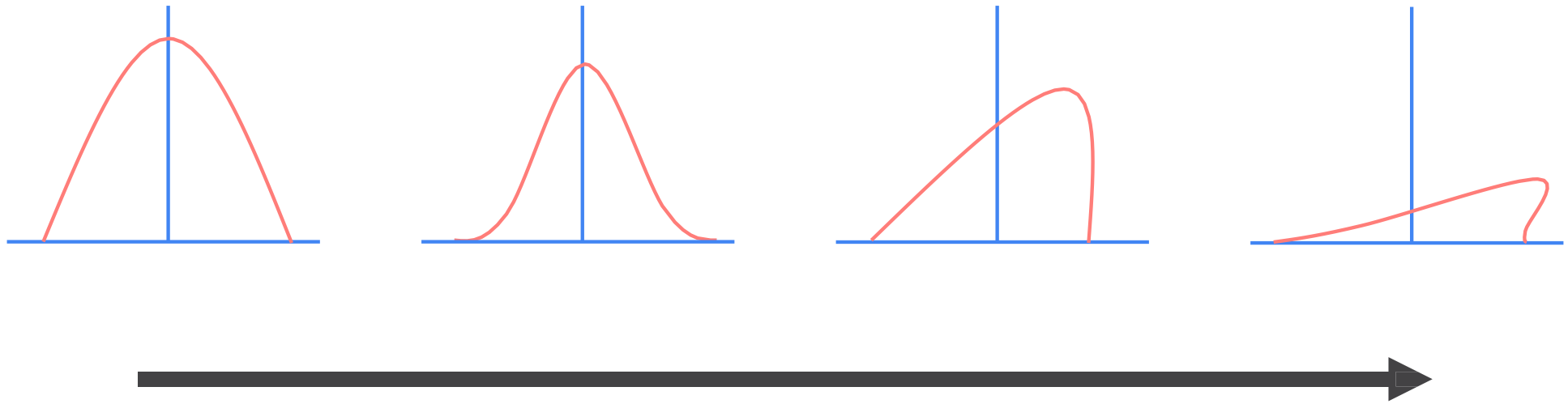
Batch Normalization



Batch Normalization

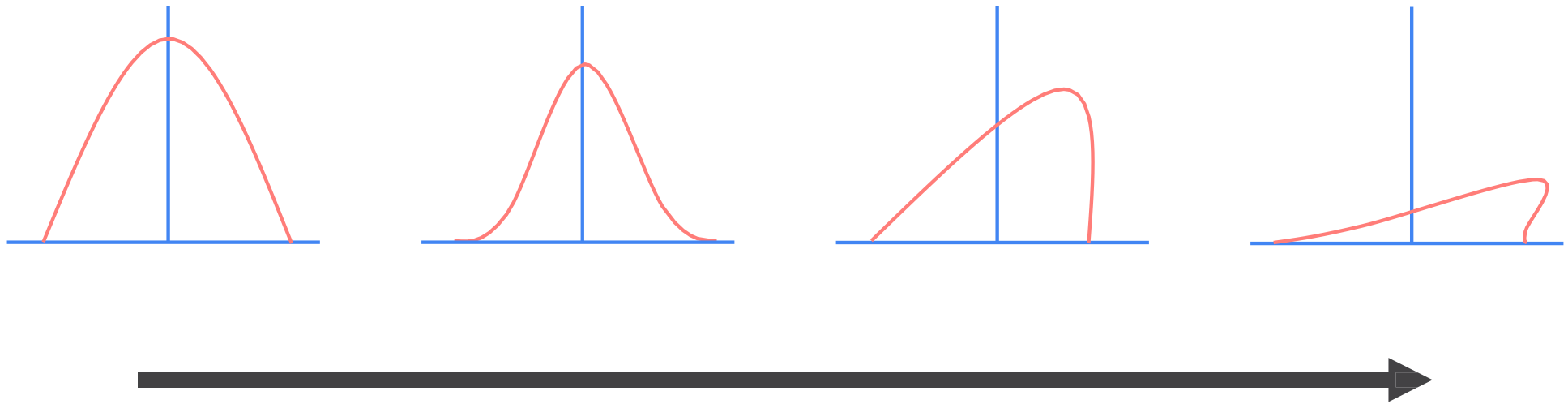


Batch Normalization



Internal Covariate Shift

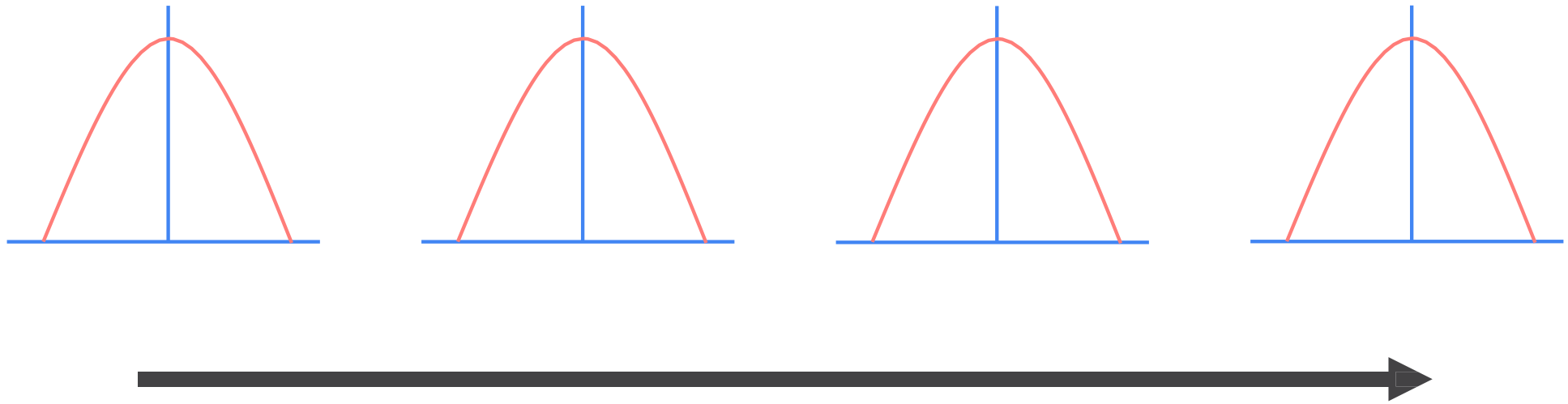
Batch Normalization



$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$y = \gamma \hat{x} + \beta$$

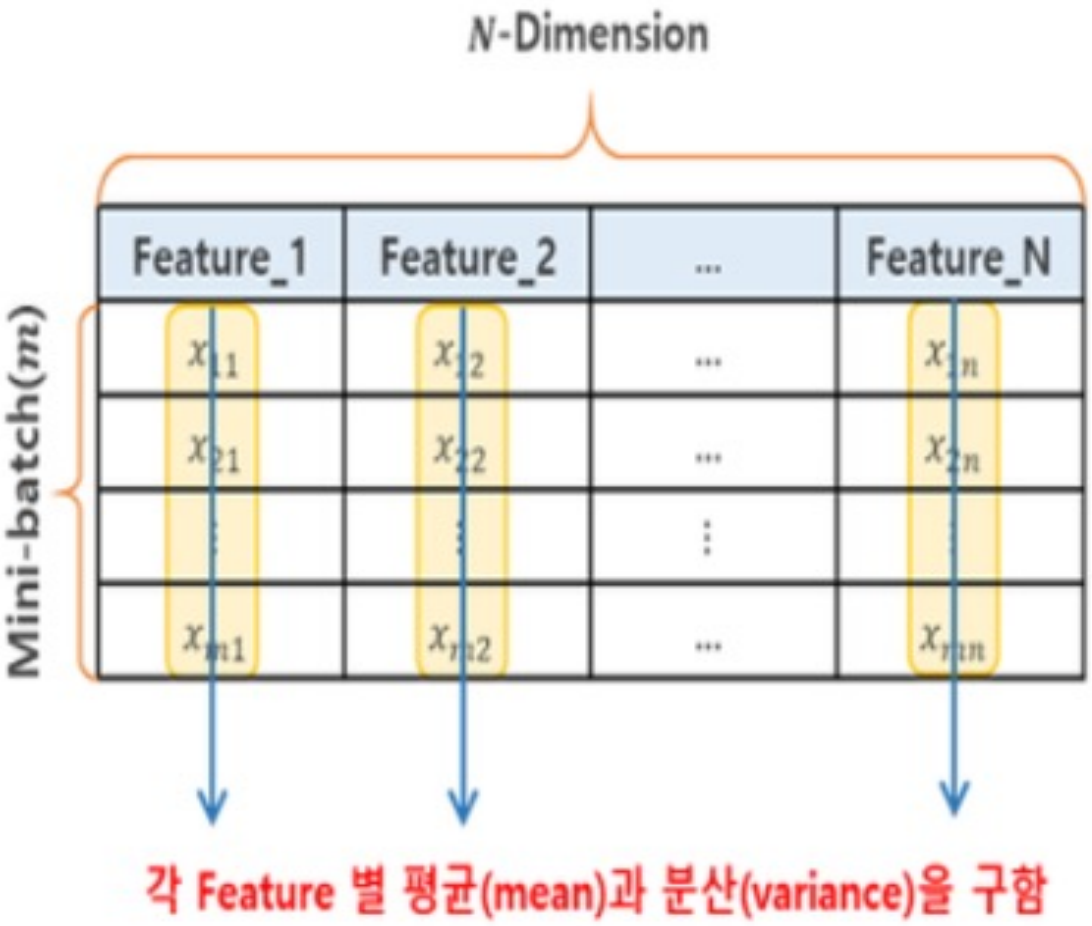
Batch Normalization



$$\bar{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \in}}$$

$$y = \gamma \hat{x} + \beta$$

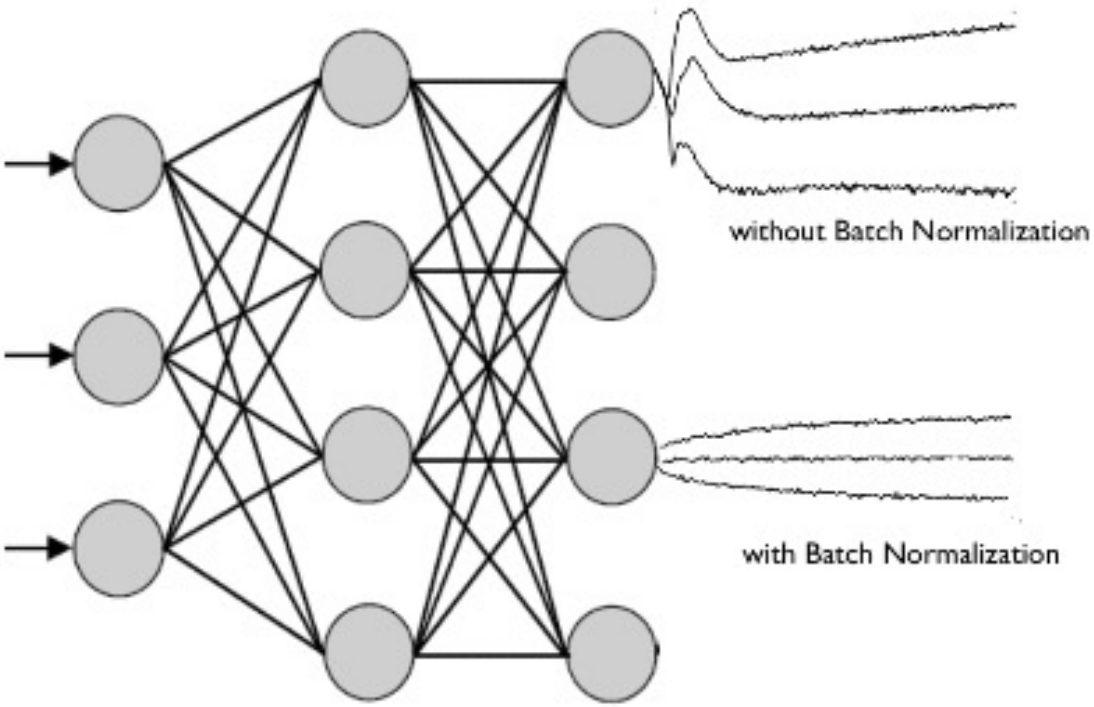
Batch Normalization



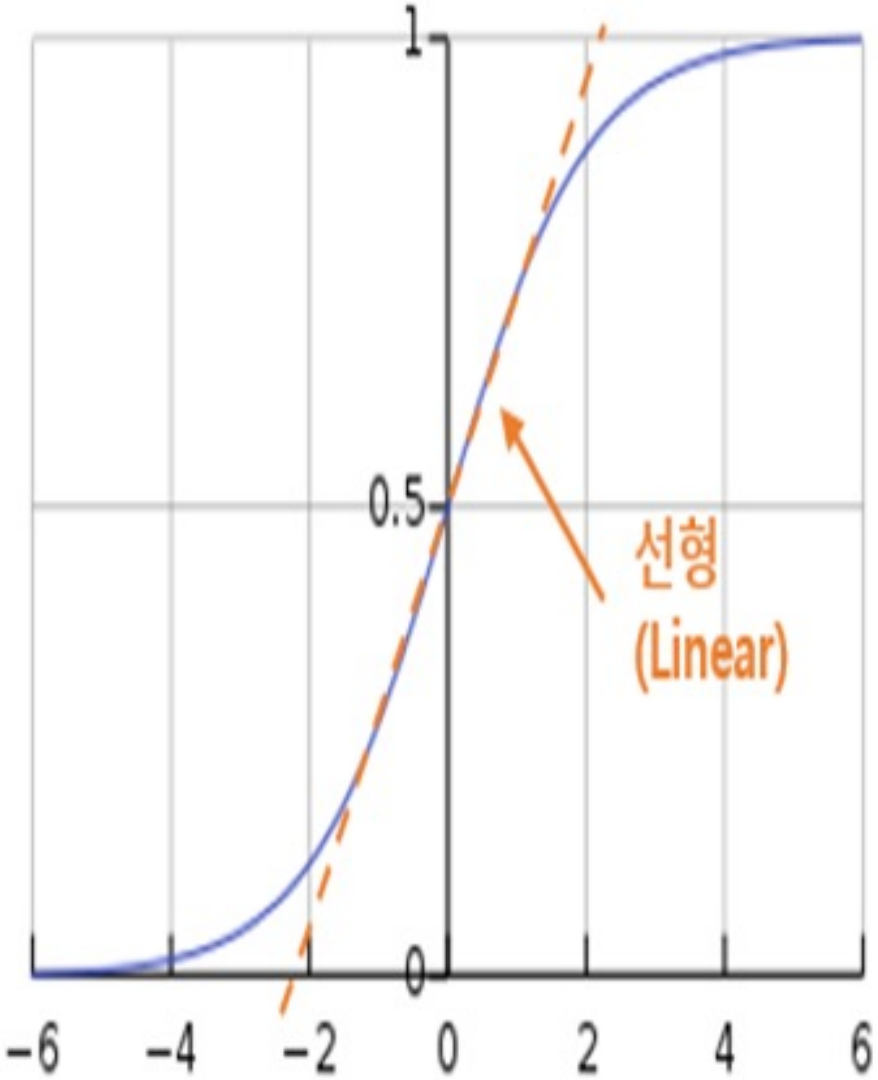
Normalize

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

Batch Normalization



Batch Normalization



Advantages of Batch Normalization

1. BN enables higher learnig rate.
2. BN regularizes the model
3. 초기값에 크게 의존하지 않습니다.

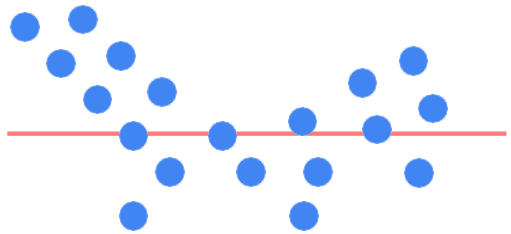
Geofferey Hinton's summary of findings up to today

- Our labeled datasets were thousands of times too small → 레이블링 데이터셋 구축
- Our computers were millions of times too slow → 병렬처리
- ~~We initialized the weights in a stupid way~~
- ~~We used the wrong type of non-linearity~~

딥러닝에서도 과적합?

Dropout

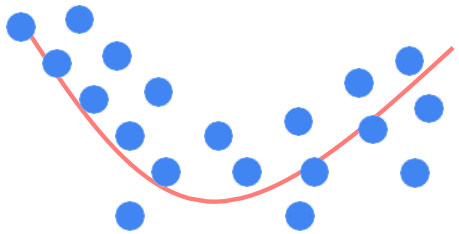
Dropout



Under-fitting

Train ↓

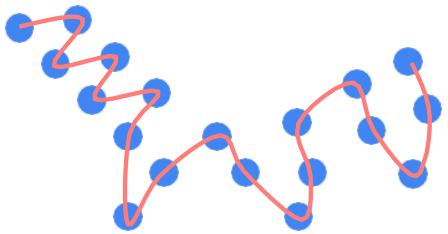
Test ↓



Good

Train =

Test ↑

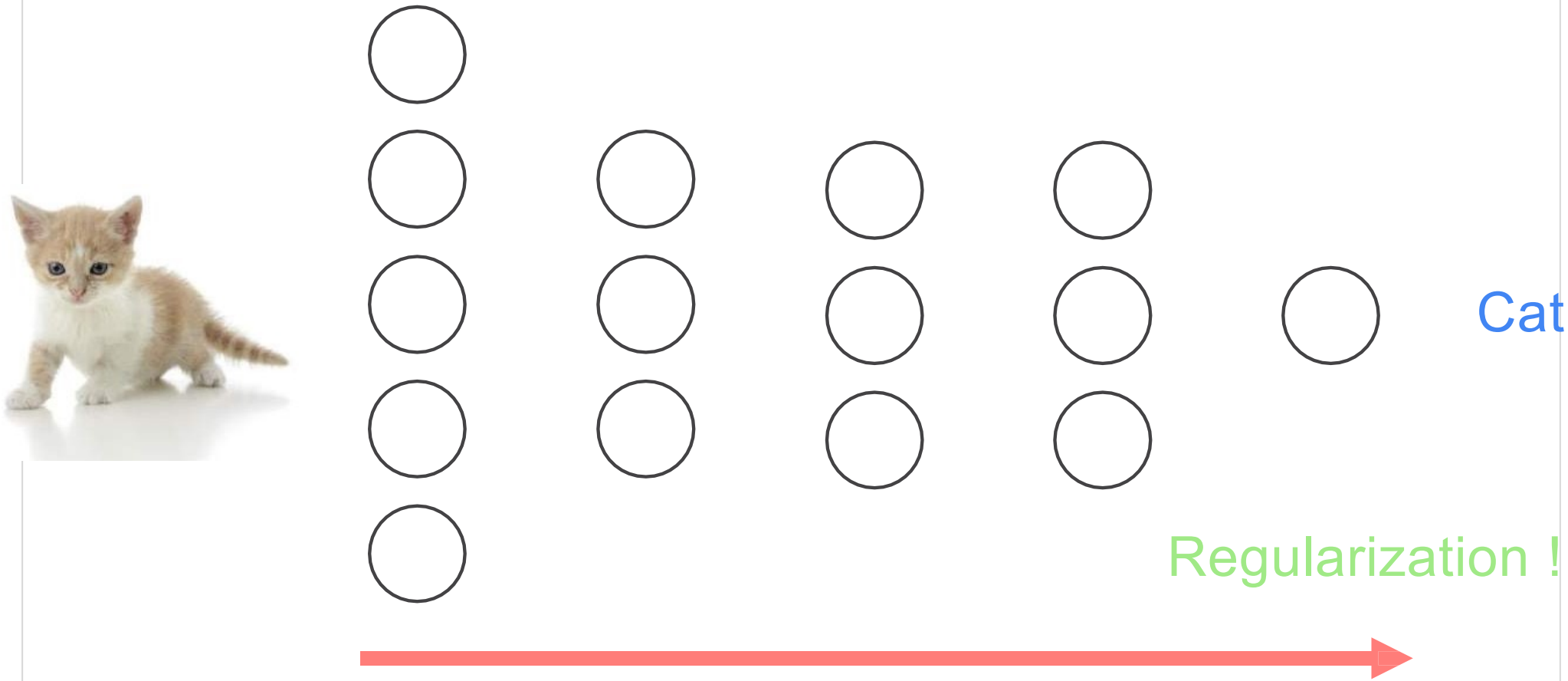


Over-fitting

Train ↑

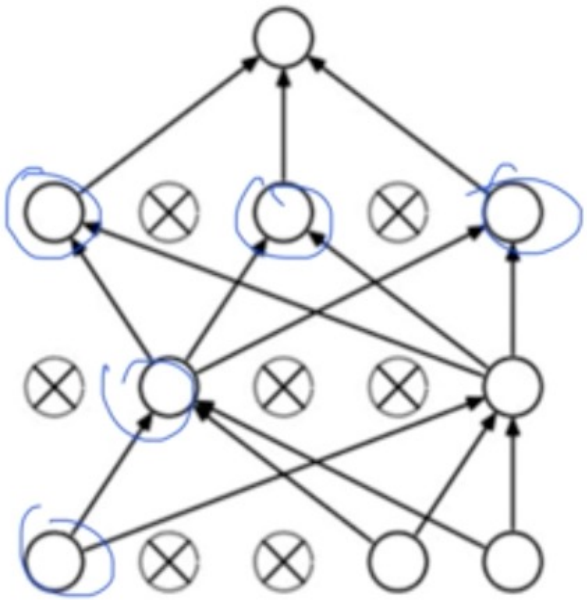
Test ↓

Dropout



Dropout

Waaaait a second…
How could this possibly be a good idea?



Forces the network to have a redundant representation.