LECTURE 16

WINTER 2021
APPLIED MACHINE LEARNING
CIHANG XIE

SLIDE CREDIT:
NARGES NOROUZI

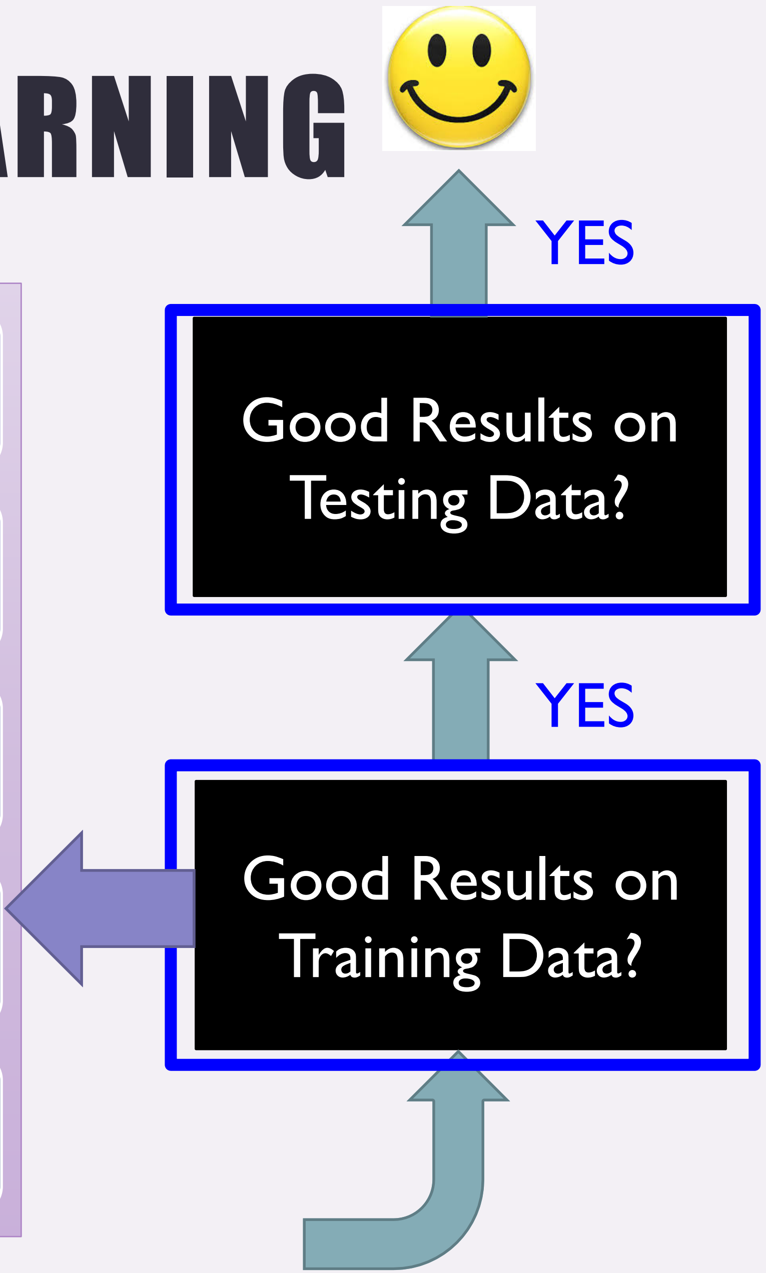
HWS & QUIZZES

- 4 HWs in total (50%)
  - You can earn extra credits by completing the bonus questions
- 5 Quizzes in total (30%)
  - Quiz 1 **5 pts**
  - Quiz 2 **7 pts**
  - Quiz 3 **6 pts**
  - Quiz 4 **8 pts**
  - **Quiz 5 10 pts**

Total quiz credit will be capped at 30 pts

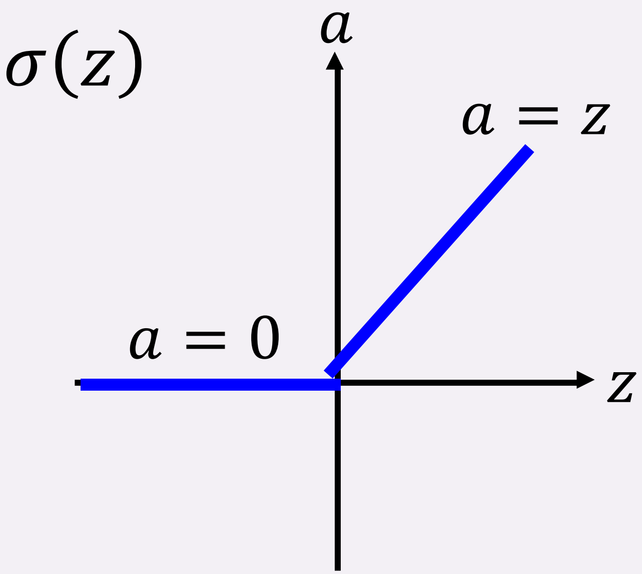
RECIPE FOR DEEP LEARNING

- Choosing proper loss
- Mini-batch
- New activation function
- Adaptive Learning Rate
- Momentum



RELU

- Rectified Linear Unit (ReLU)



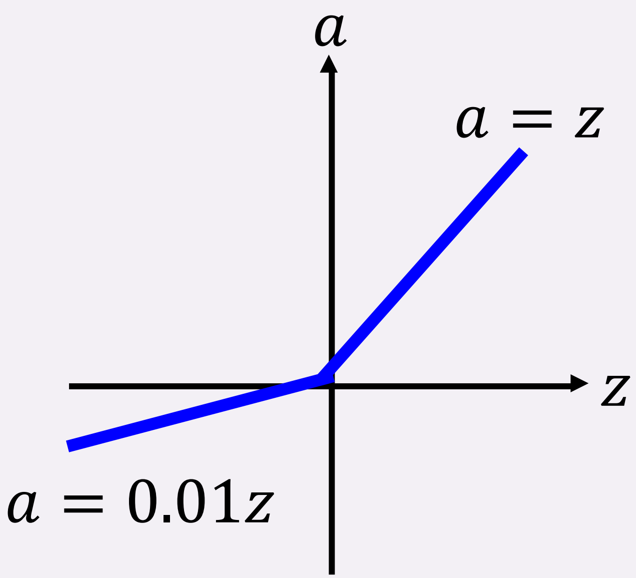
[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

Reason:

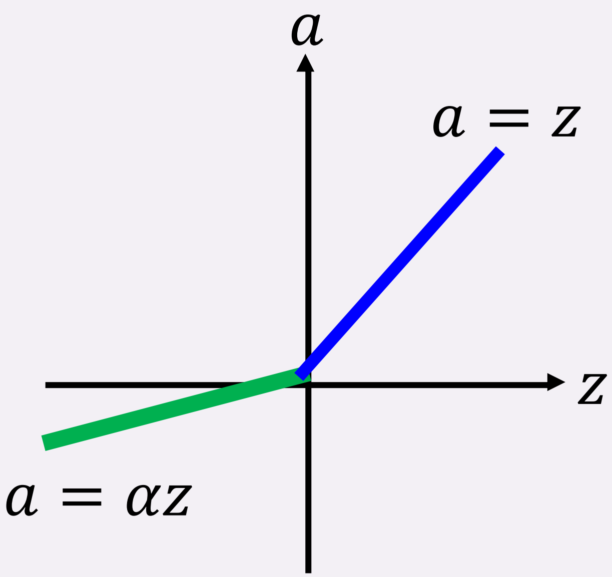
1. Fast to compute
2. Biological reason
3. Vanishing gradient problem

RELU - VARIANT

$Leaky\ ReLU$



$Parametric\ ReLU$



α also learned by gradient descent

RELU – "SMOOTH" VARIANT

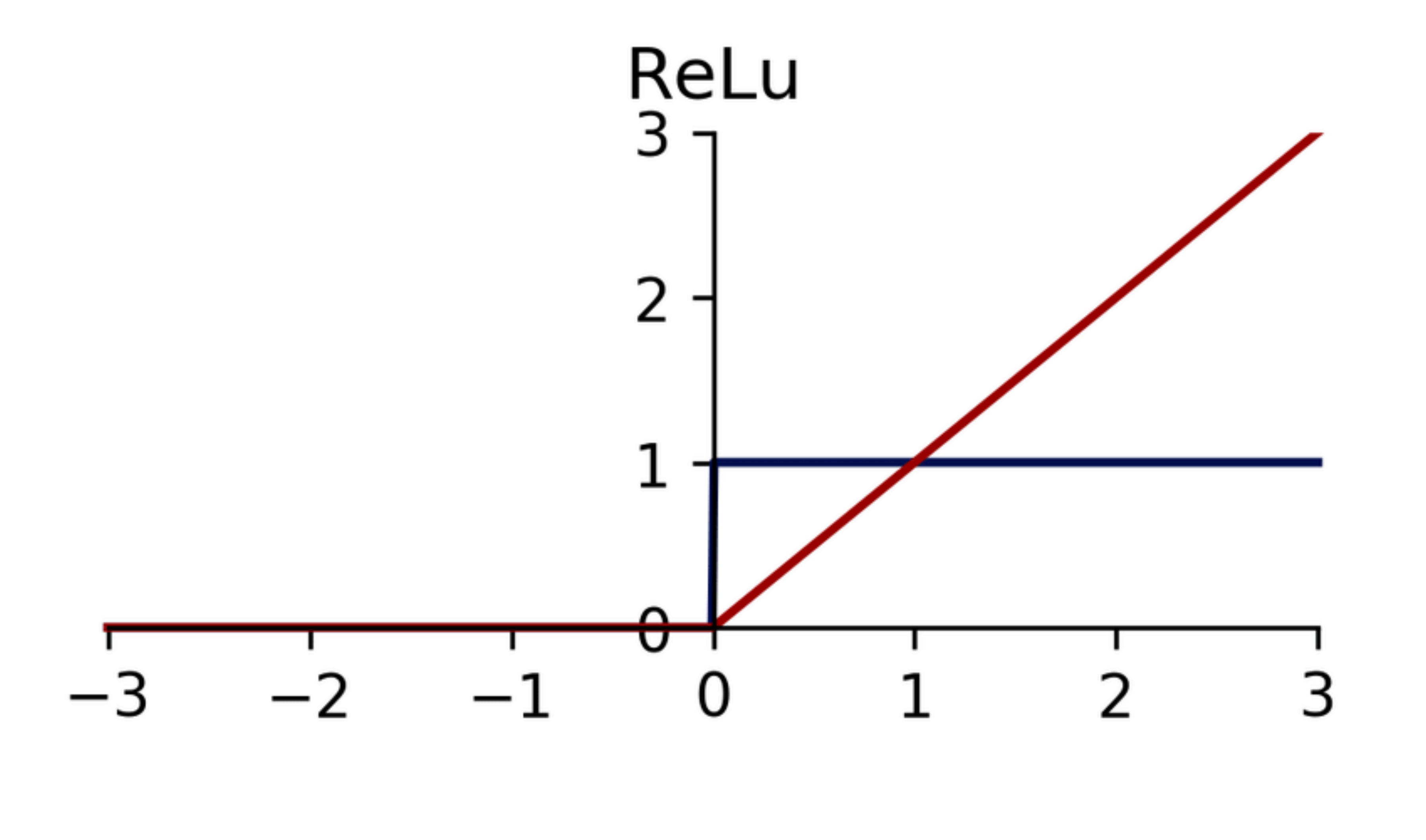
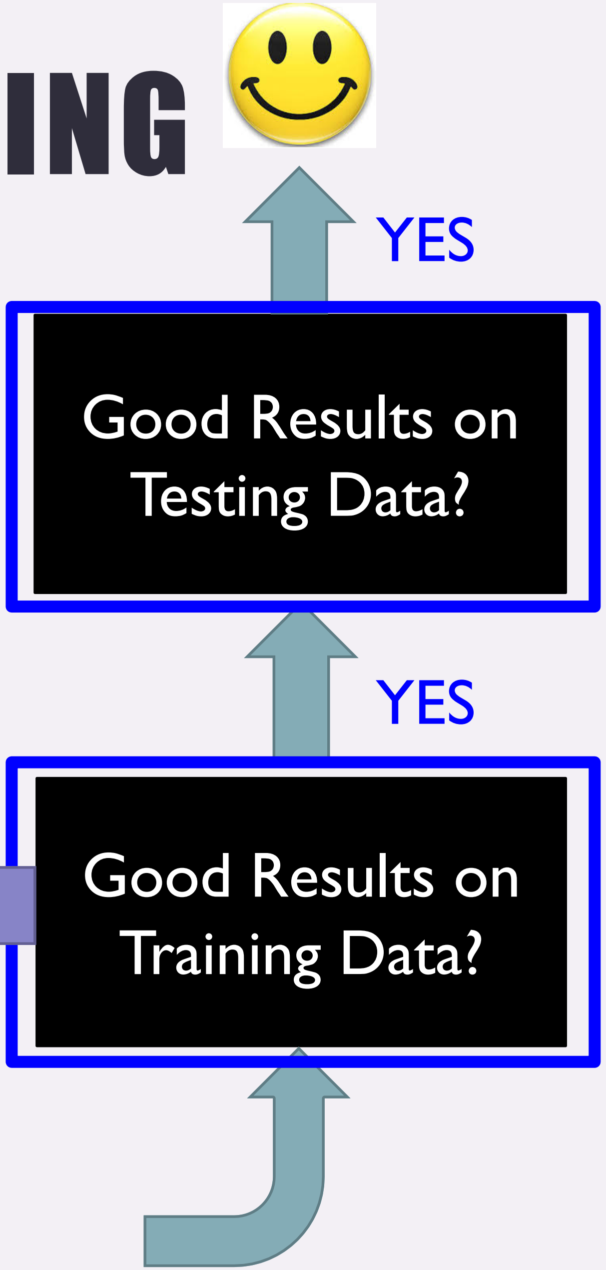


Image credit: https://arxiv.org/pdf/2006.13311.pdf

RECIPE FOR DEEP LEARNING

- Choosing proper loss
- Mini-batch
- New activation function
- Adaptive Learning Rate
- Momentum



ADAGRAD

Original: $\theta \leftarrow \theta - \alpha \partial L \, / \, \partial \theta$

Adagrad: $\theta \leftarrow \theta - \boxed{\alpha_\theta} \partial L \, / \, \partial \theta$

Parameter-dependent learning rate

$$\alpha_\theta = \frac{\alpha}{\sqrt{\boxed{\sum_{i=0}^{t} (g^i)^2}}}$$

constant

g^i is $\partial L \, / \, \partial \theta$ obtained at the i^{th} update

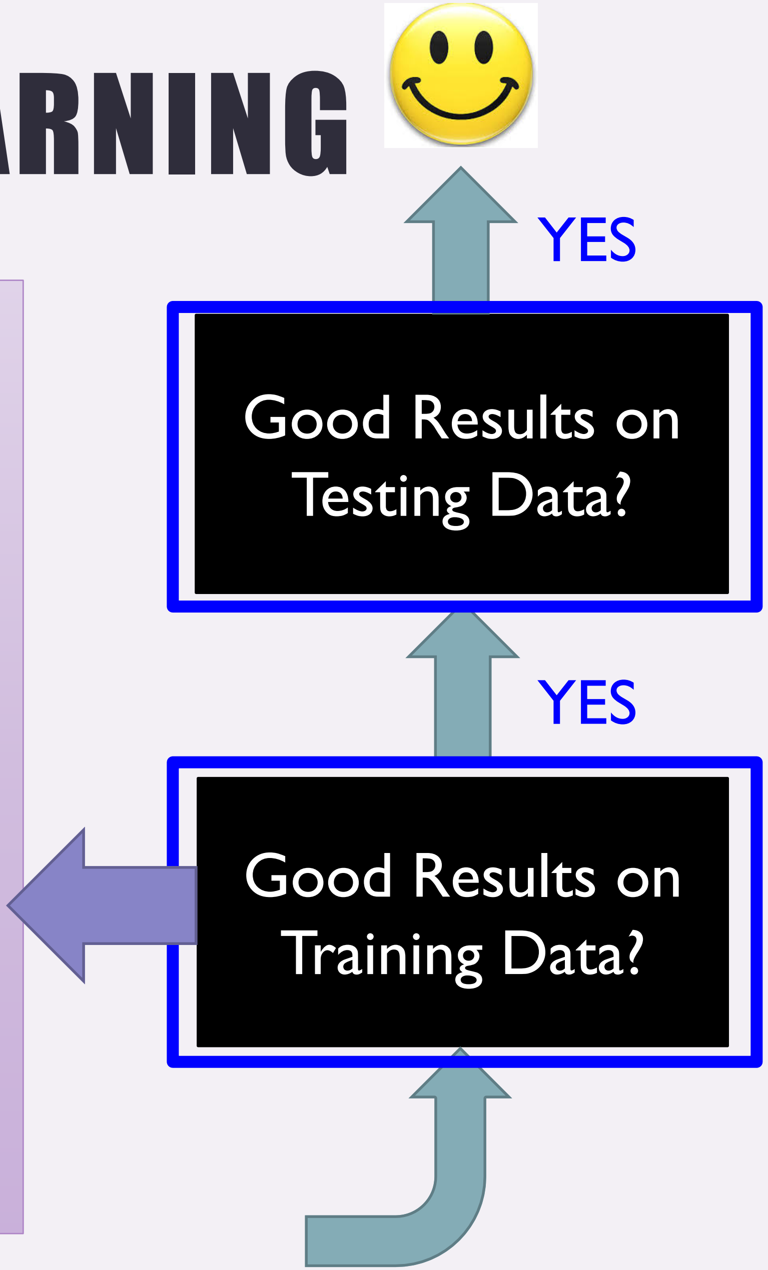
Summation of the square of the previous derivatives

TODAY

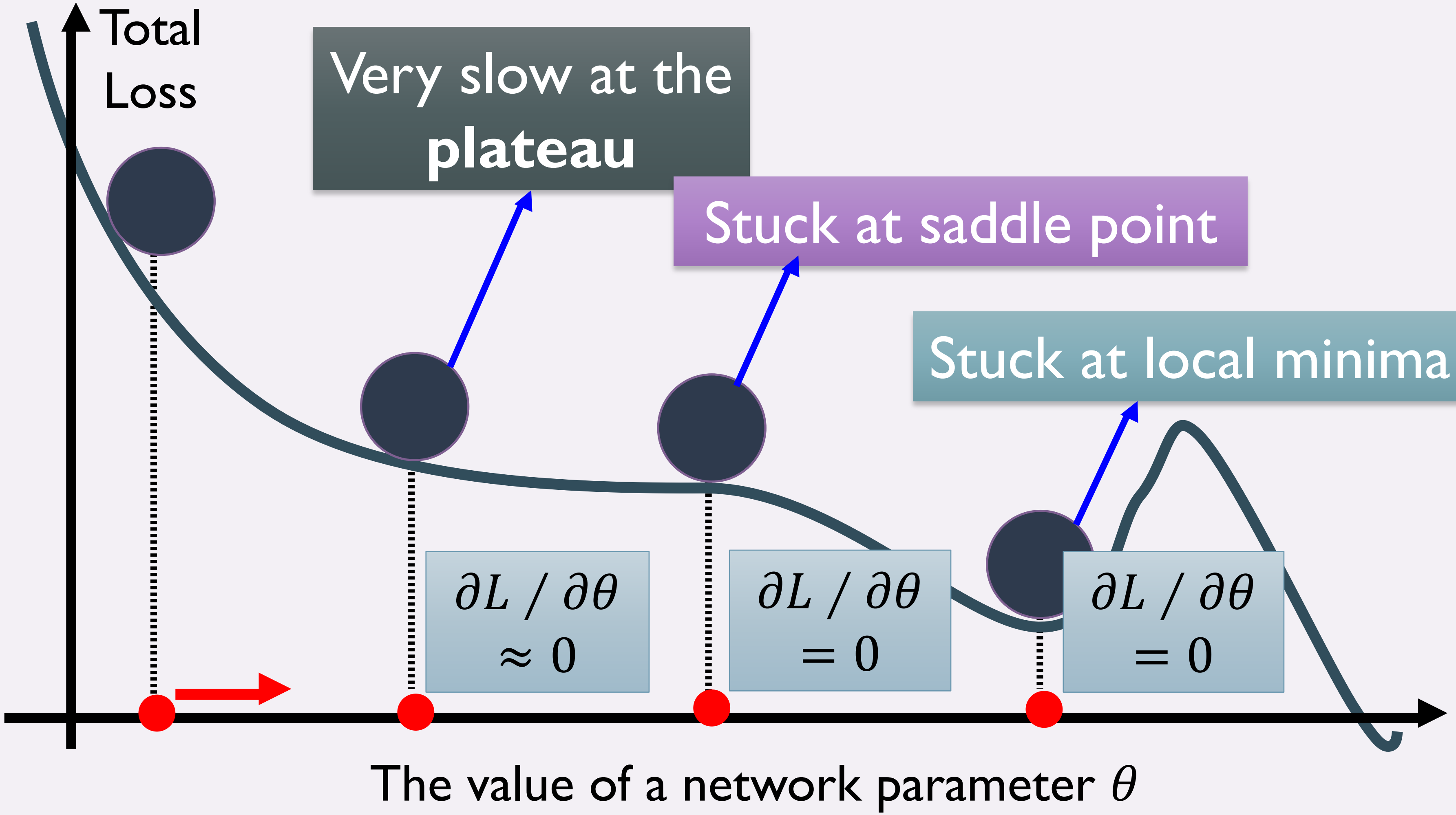
- Today: More deep learning topics
  - Momentum
  - Avoiding overfitting
    - Data augmentation
    - Early stopping
    - Regularization
    - Dropout technique

RECIPE FOR DEEP LEARNING

- Choosing proper loss
- Mini-batch
- New activation function
- Adaptive Learning Rate
- Momentum

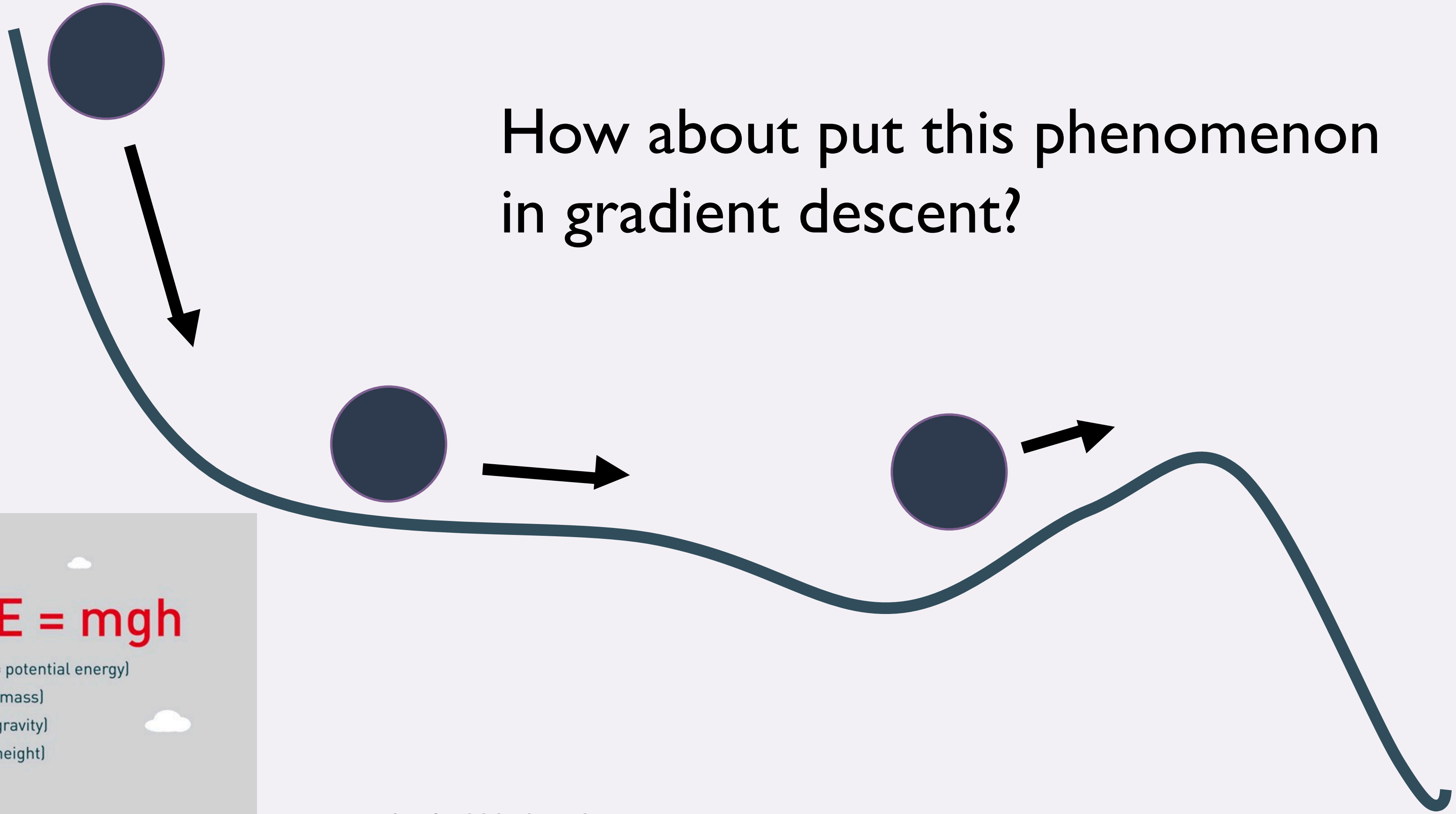


HARD TO FIND OPTIMAL NETWORK PARAMETERS

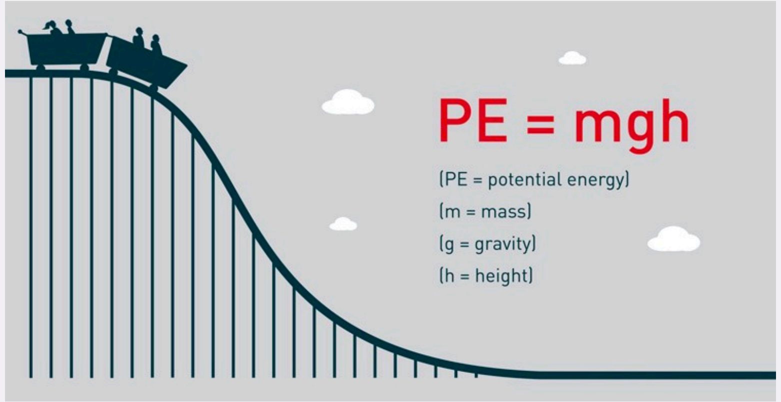


IN PHYSICAL WORLD ……

- Momentum



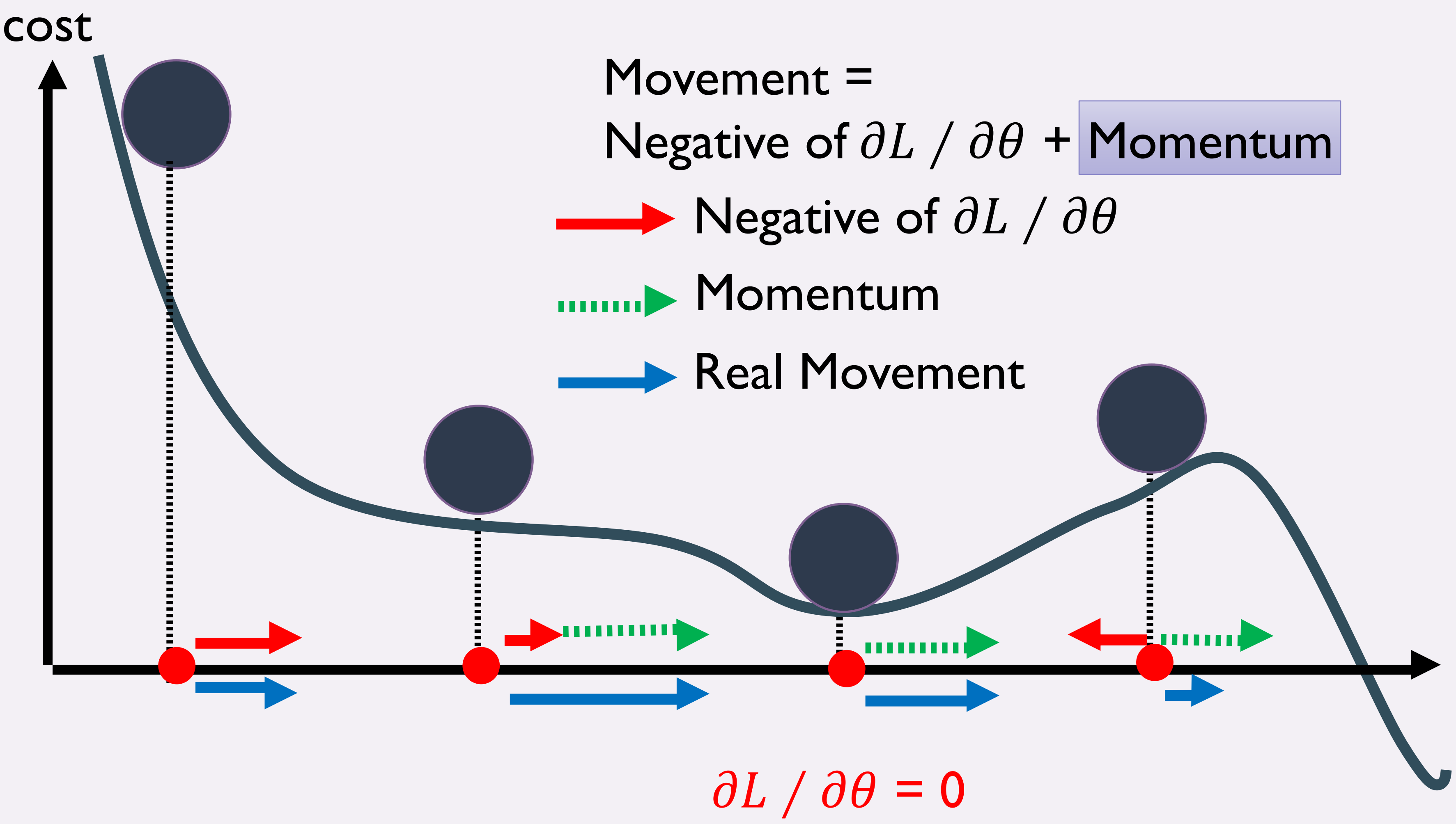
How about put this phenomenon in gradient descent?



MOMENTUM

Still not guarantee reaching global minima, but give some hope ……

Movement =
Negative of ∂L / $\partial \theta$ + Momentum



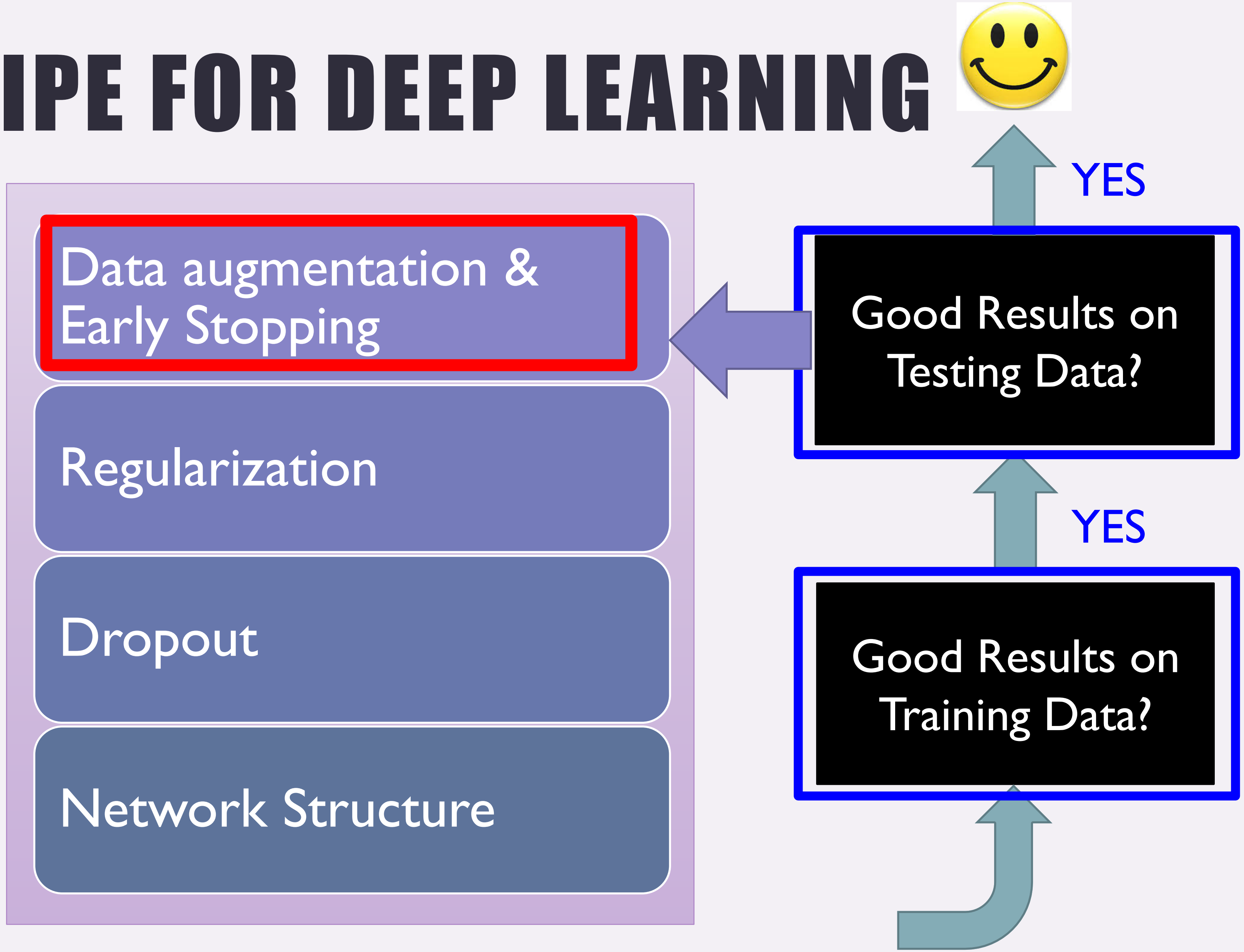
MOMENTUM

- Momentum update (t is the iteration number)

$$v^t = \mu \times v^{t-1} - \alpha \times \frac{\partial L}{\partial \theta} \; (integrate\ velocity)$$
$$\theta^t \mathrel{+}= v^t$$

- v is initialized at 0 (from the top of the hill).
- μ is called "coefficient of momentum". Think about it as coefficient of friction of the surface. This variable damps the energy of the system, allowing v to stop.
- Pros: can be used to handle noisy gradients + can handle extremely small gradients
- Cons: introduces further complexity to the model

RECIPE FOR DEEP LEARNING



DATA AUGMENTATION

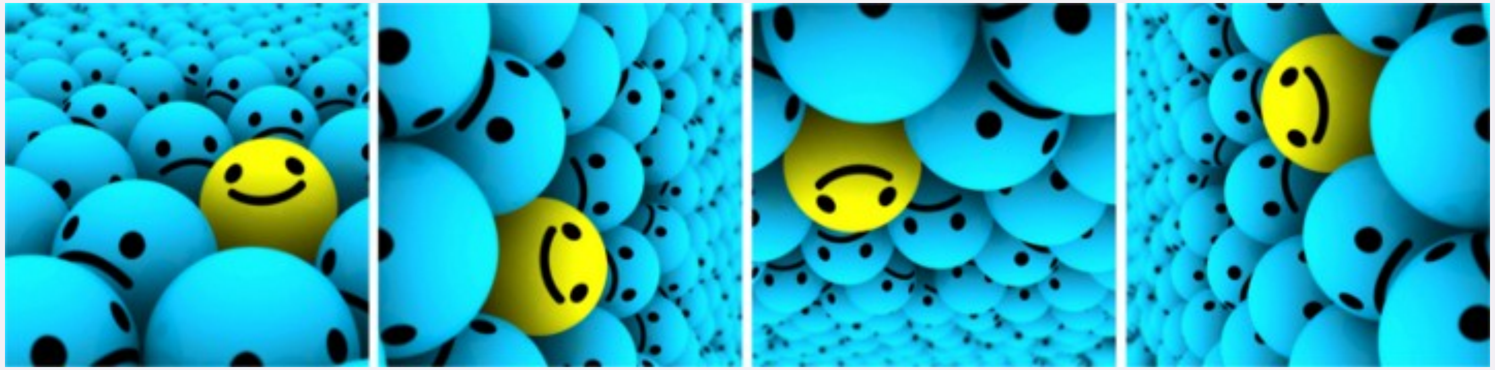
- Have more training data
- ***Create*** more training data

Image credit: https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

POPULAR DATA AUGMENTATION

flip

rotation



crop

Image credit: https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

POPULAR DATA AUGMENTATION

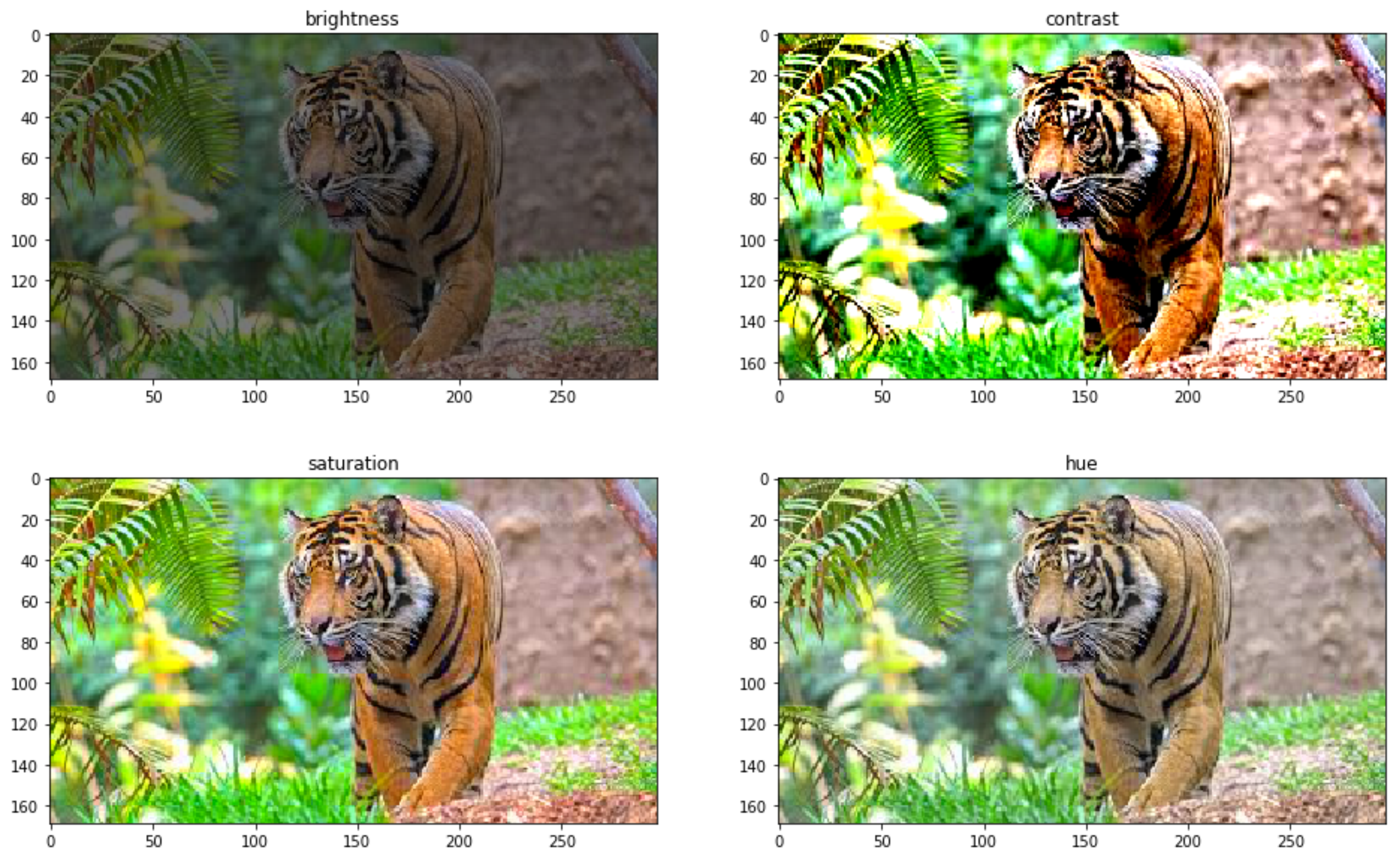


Image credit: https://iq.opengenus.org/data-augmentation/

EARLY STOP

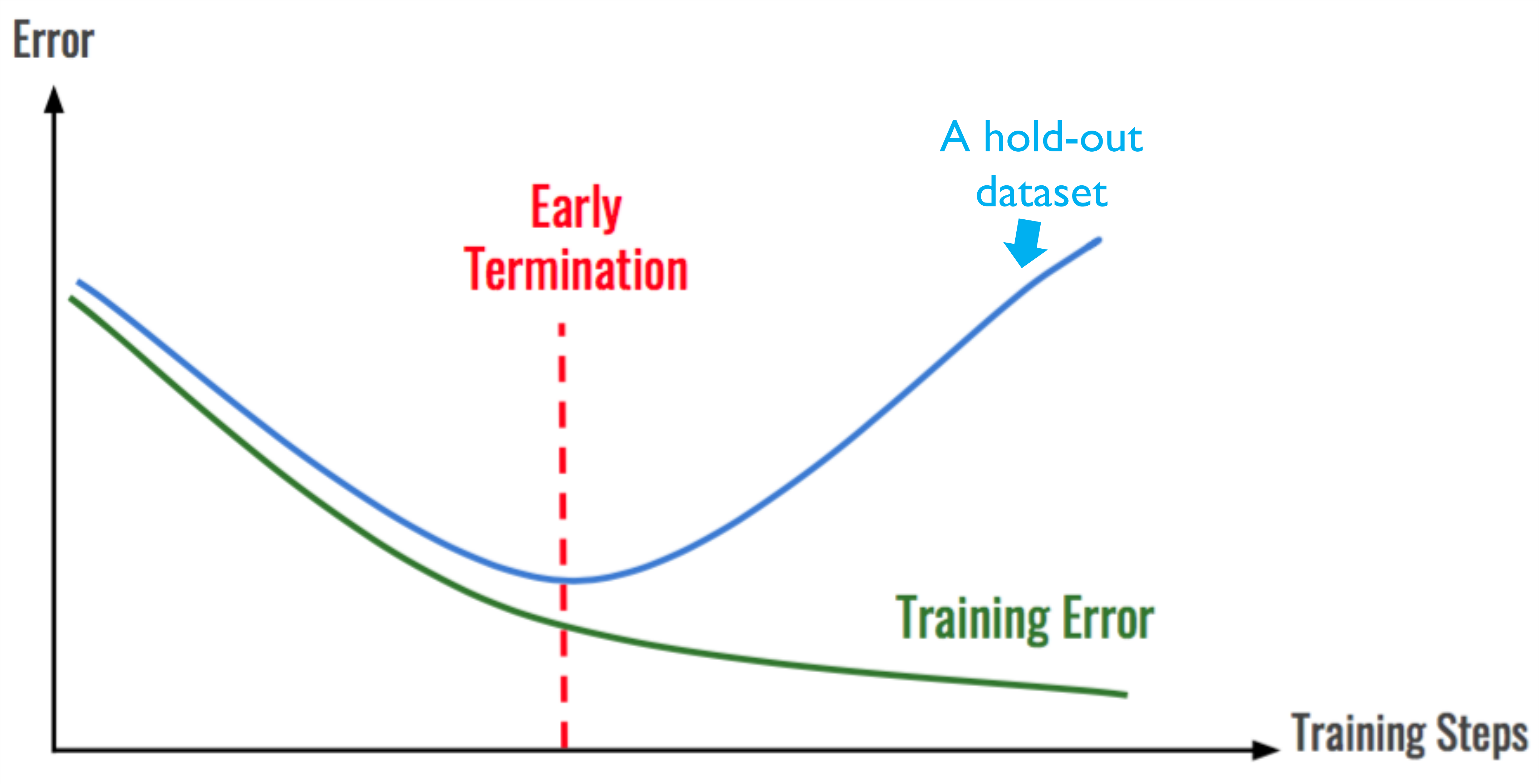
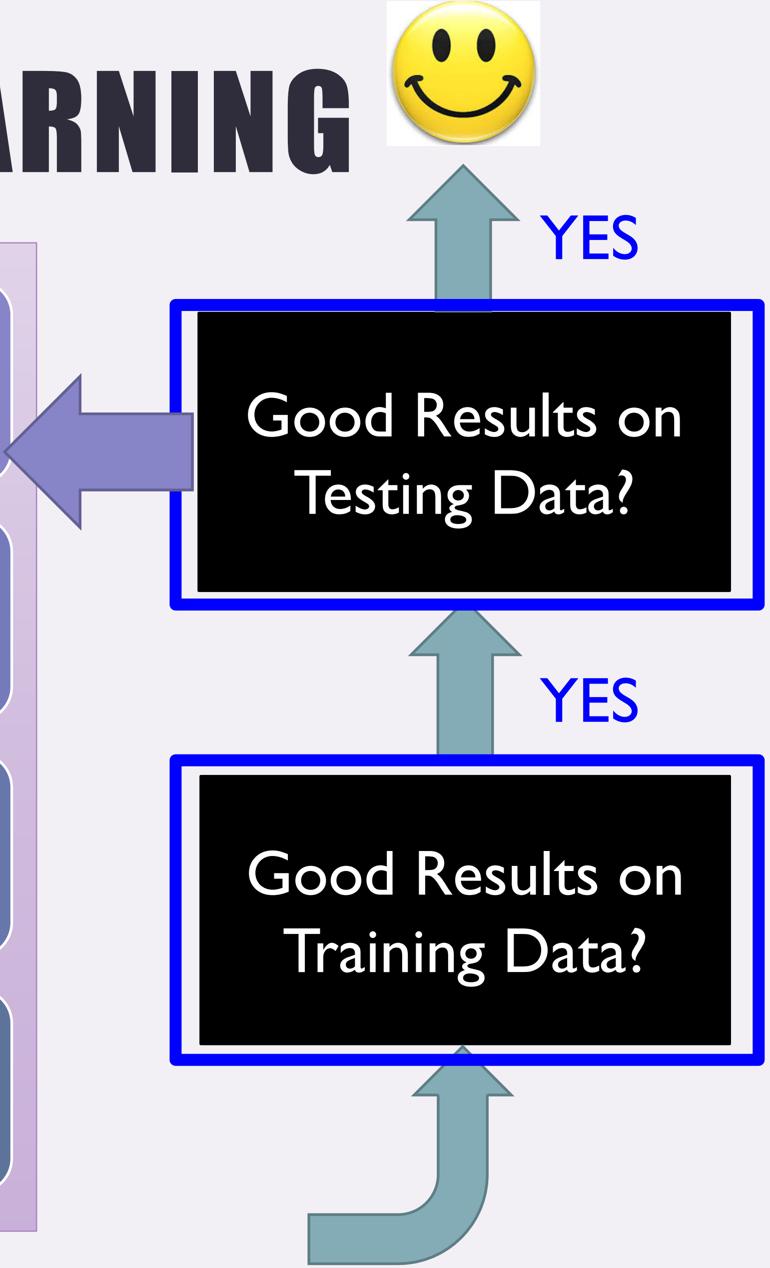


Image credit: https://medium.com/analytics-vidhya/early-stopping-with-pytorch-to-restrain-your-model-from-overfitting-dce6de4081c5

RECIPE FOR DEEP LEARNING

- Early Stopping
- Regularization
- Dropout
- Network Structure



OVERFITTING REVISED: REGULARIZATION

- A regularizer is an additional criteria to the loss function to make sure that we don't overfit
- It's called a regularizer since it tries to keep the parameters more normal/regular
- It is a bias on the model forces the learning to prefer certain types of weights over others

$$TrainLoss(\theta) = \frac{1}{|D_{train}|} \sum_{(x,y) \in D_{train}} Loss(x, y, \theta)$$

$$\min_{\theta \in \mathbb{R}^d} TrainLoss(\theta) + \lambda\, regularizer(\theta)$$

COMMON REGULARIZERS

- Sum of the weights

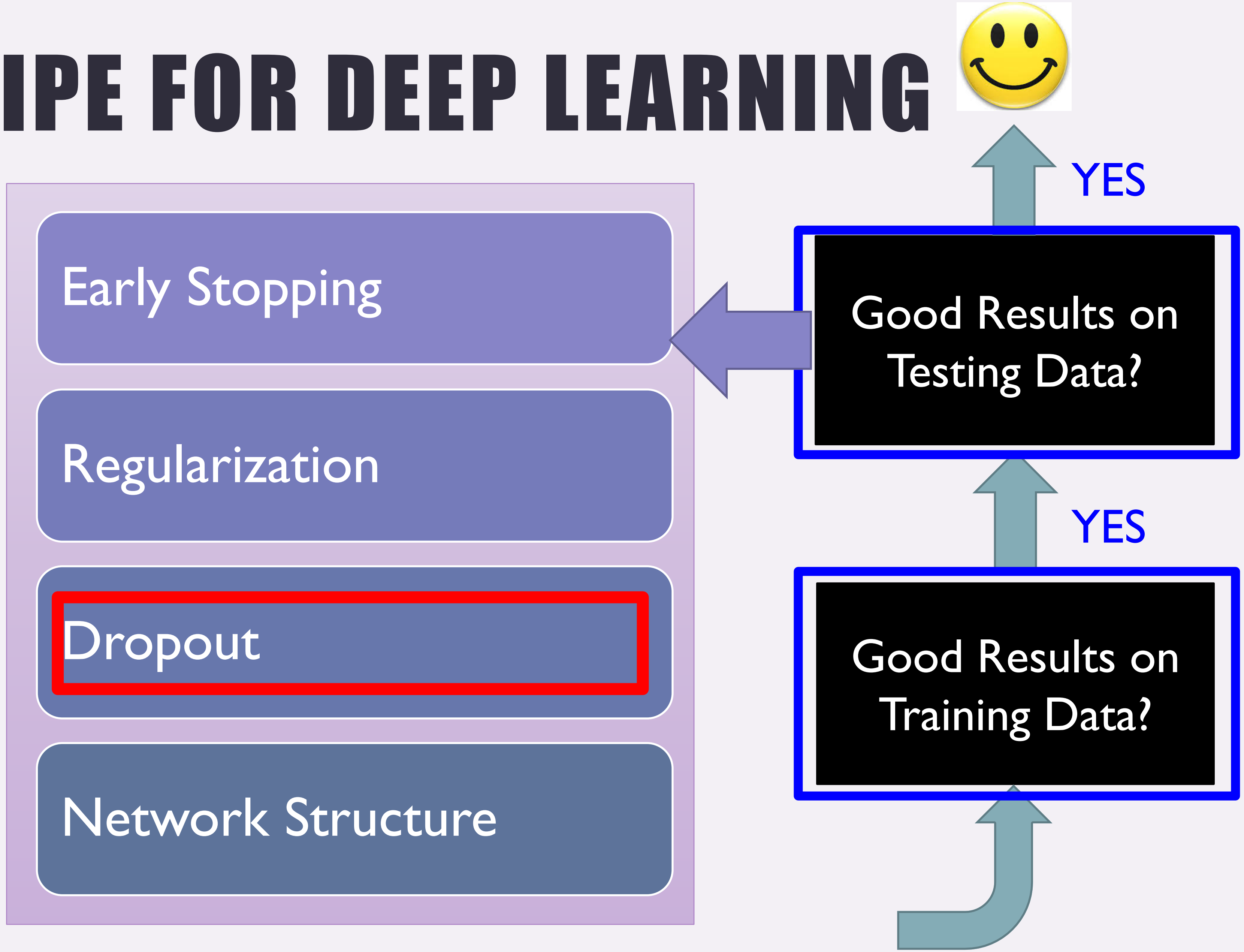
$$r(\theta) = \sum_{\theta_j} |\theta_j|$$

- Sum of the squared weights

$$r(\theta) = \sqrt{\sum_{\theta_j} \left|\theta_j\right|^2}$$

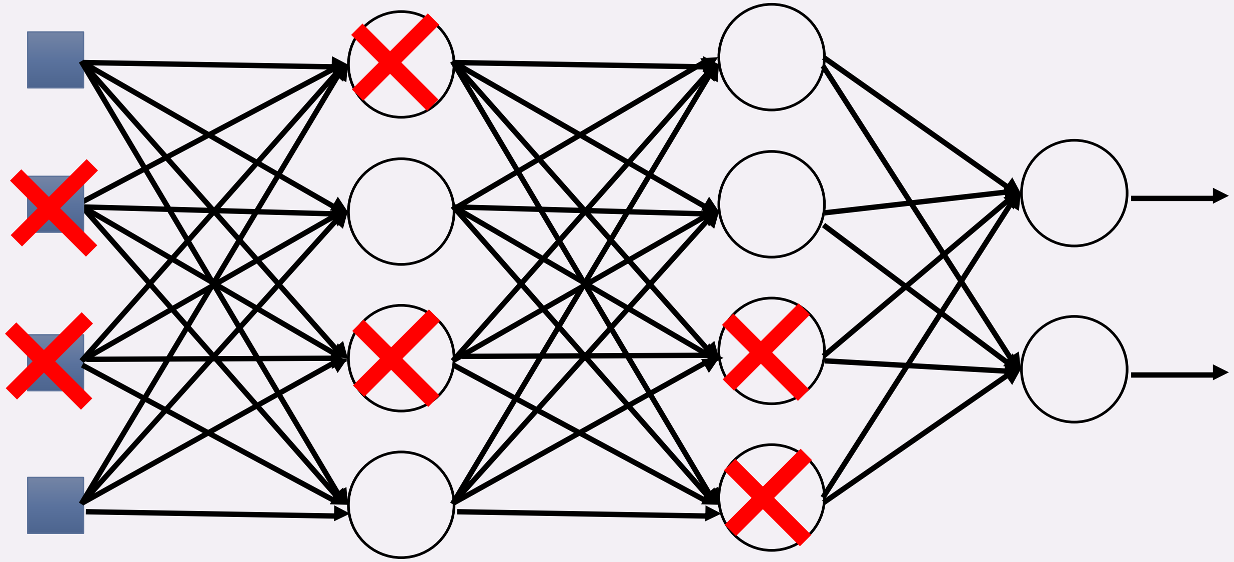
Squared weights penalizes large values more.

RECIPE FOR DEEP LEARNING



DROPOUT

Training:

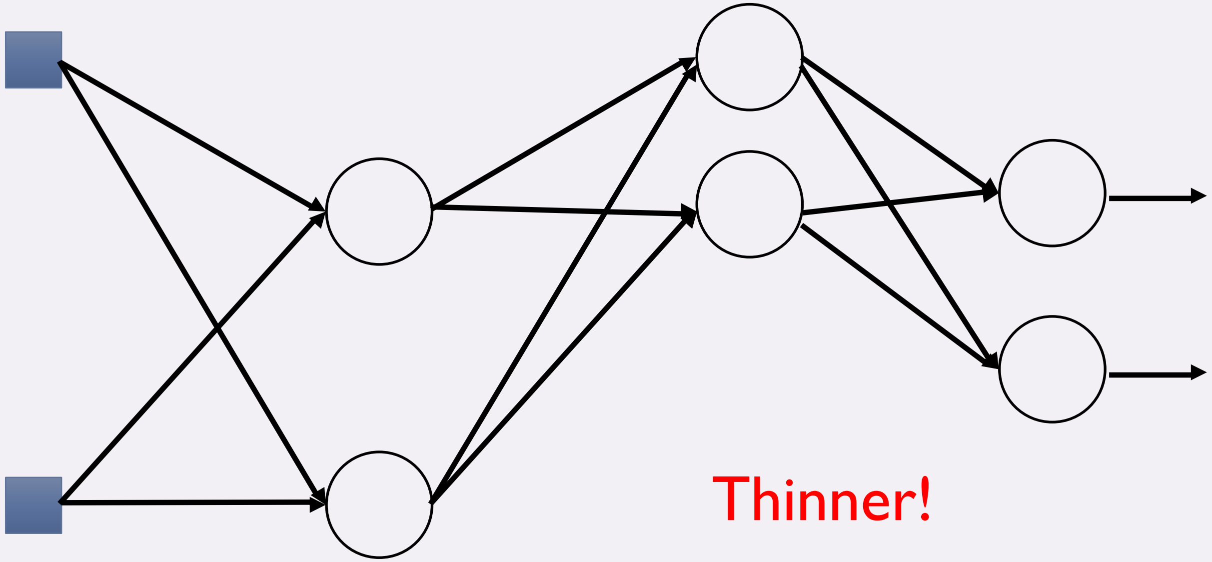


➢ **Each time before updating the parameters**

- Each neuron has $p\%$ to dropout

DROPOUT

Training:



- **Each time before updating the parameters**
  - Each neuron has $p\%$ to dropout

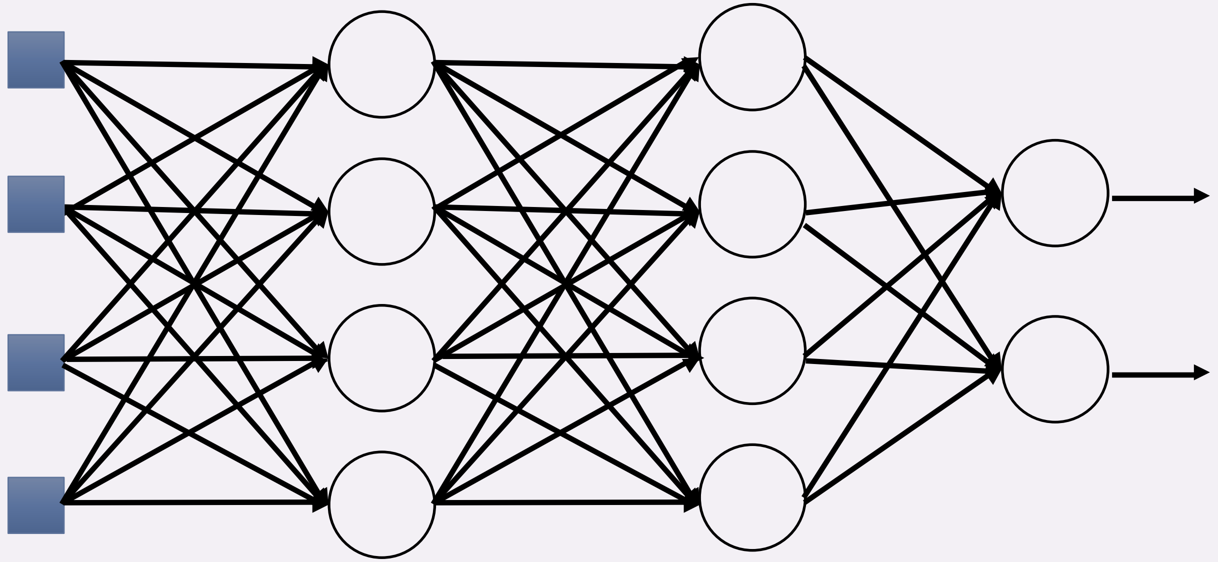
  The structure of the network is changed.

  - Using the new network for training

For each mini-batch, we resample the dropout neurons

DROPOUT

Testing:



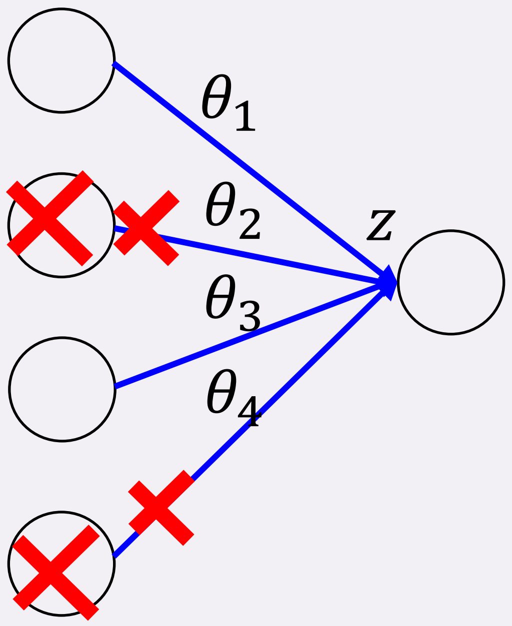
- **No dropout**
  - If the dropout rate at training is $p\%$, all the weights times $1 - p\%$
  - Assume that the dropout rate is 50%.
    If a weight $\theta = 1$ by training, set $\theta = 0.5$ for testing.

DROPOUT - INTUITIVE REASON

- Why the weights should multiply $(1 - p)\%$ (dropout rate) when testing?

<u>Training of Dropout</u>

Assume dropout rate is 50%

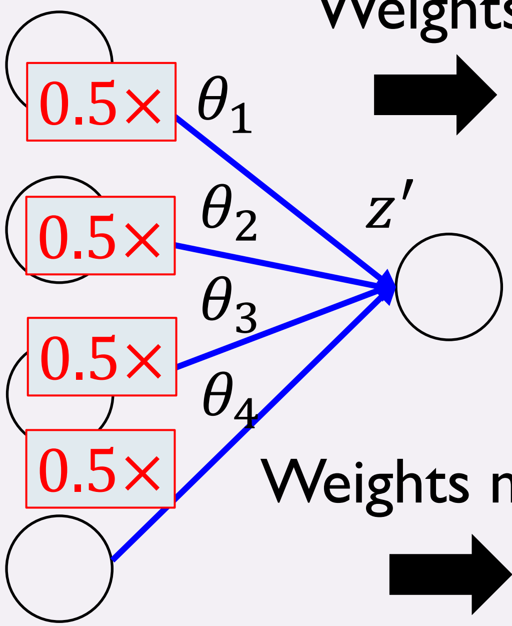


<u>Testing of Dropout</u>

No dropout

Weights from training

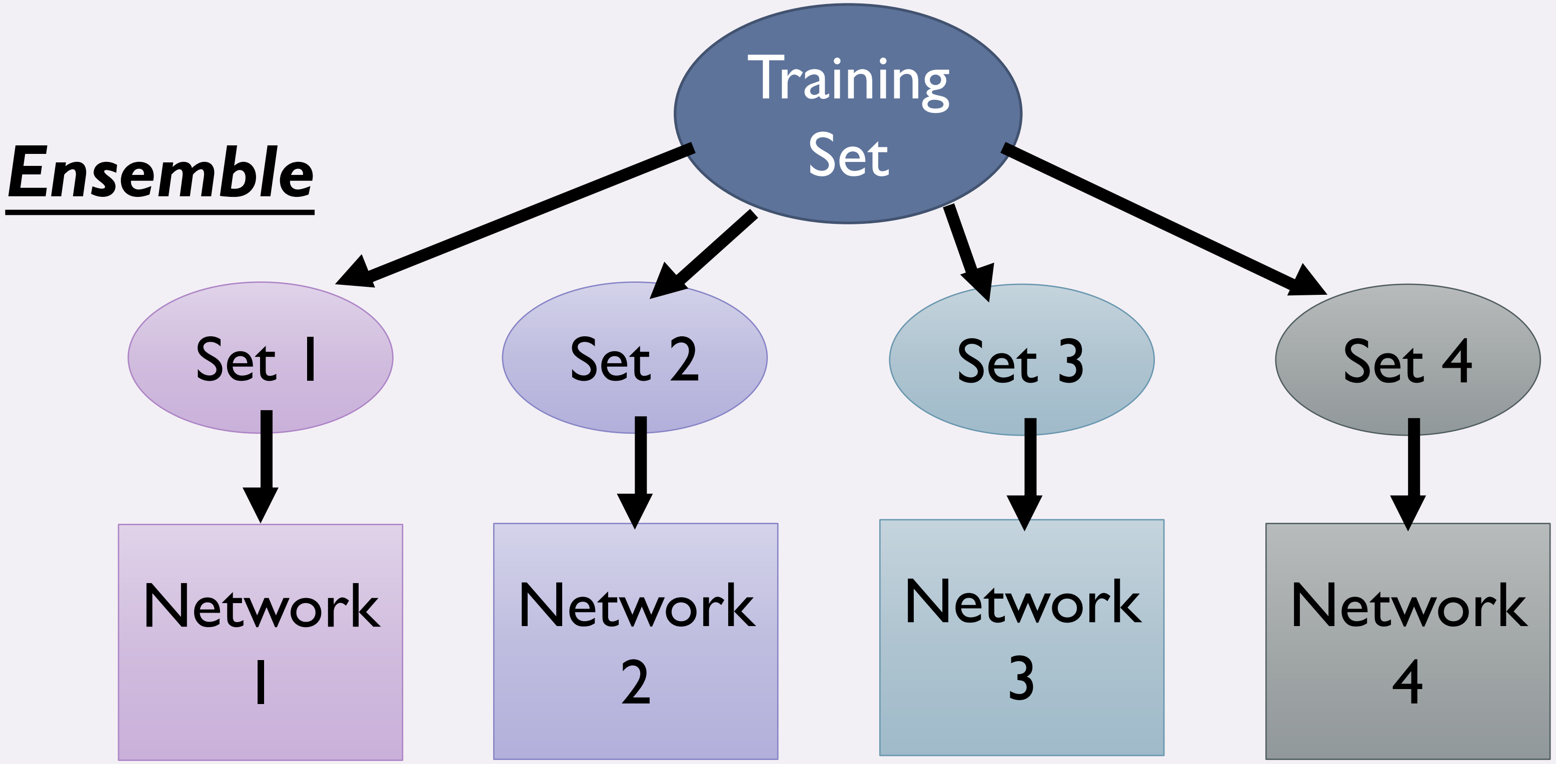
$\Rightarrow z' \approx 2z$



Weights multiply $1 - p\%$

$\Rightarrow z' \approx z$

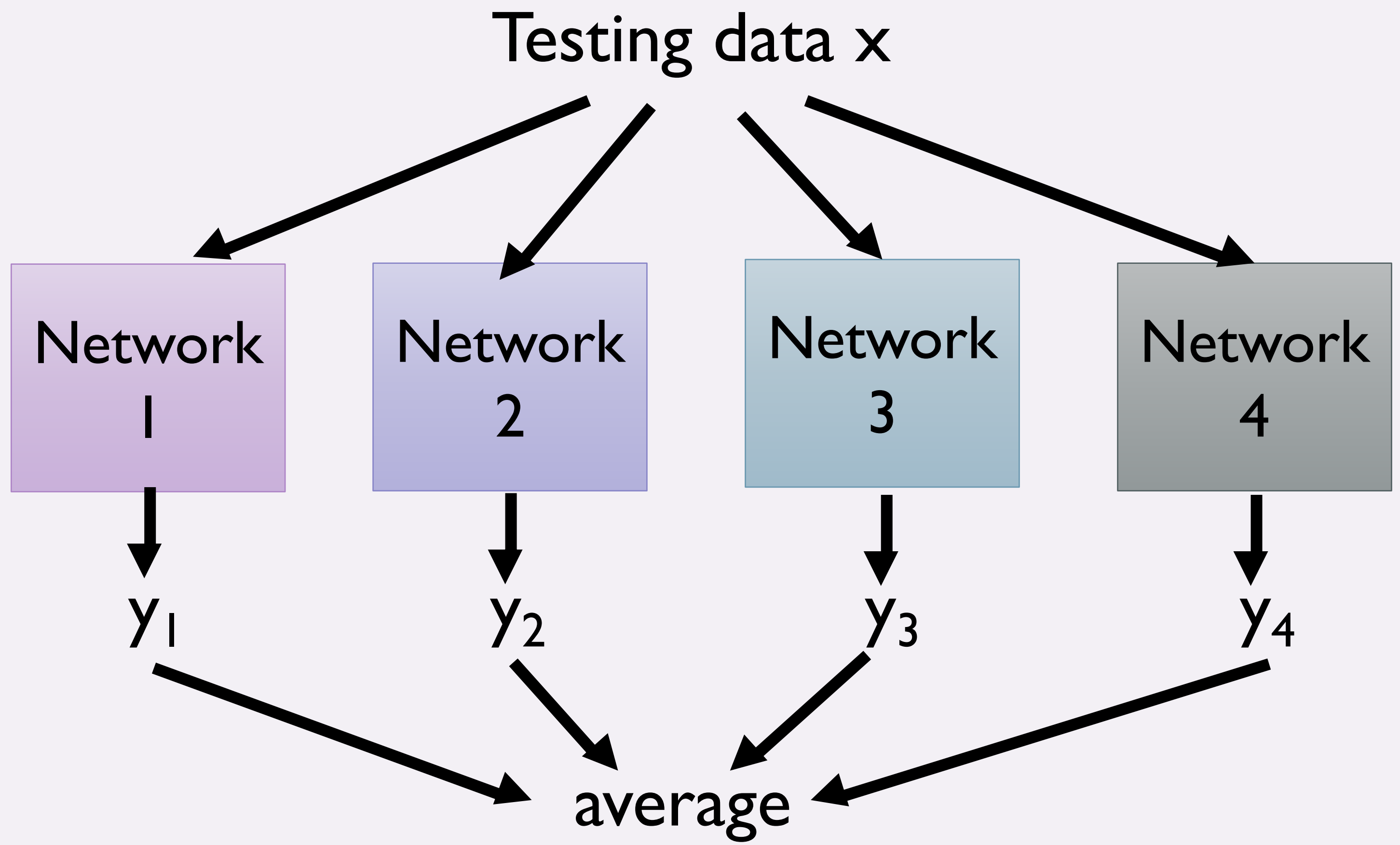
DROPOUT IS A KIND OF ENSEMBLE.



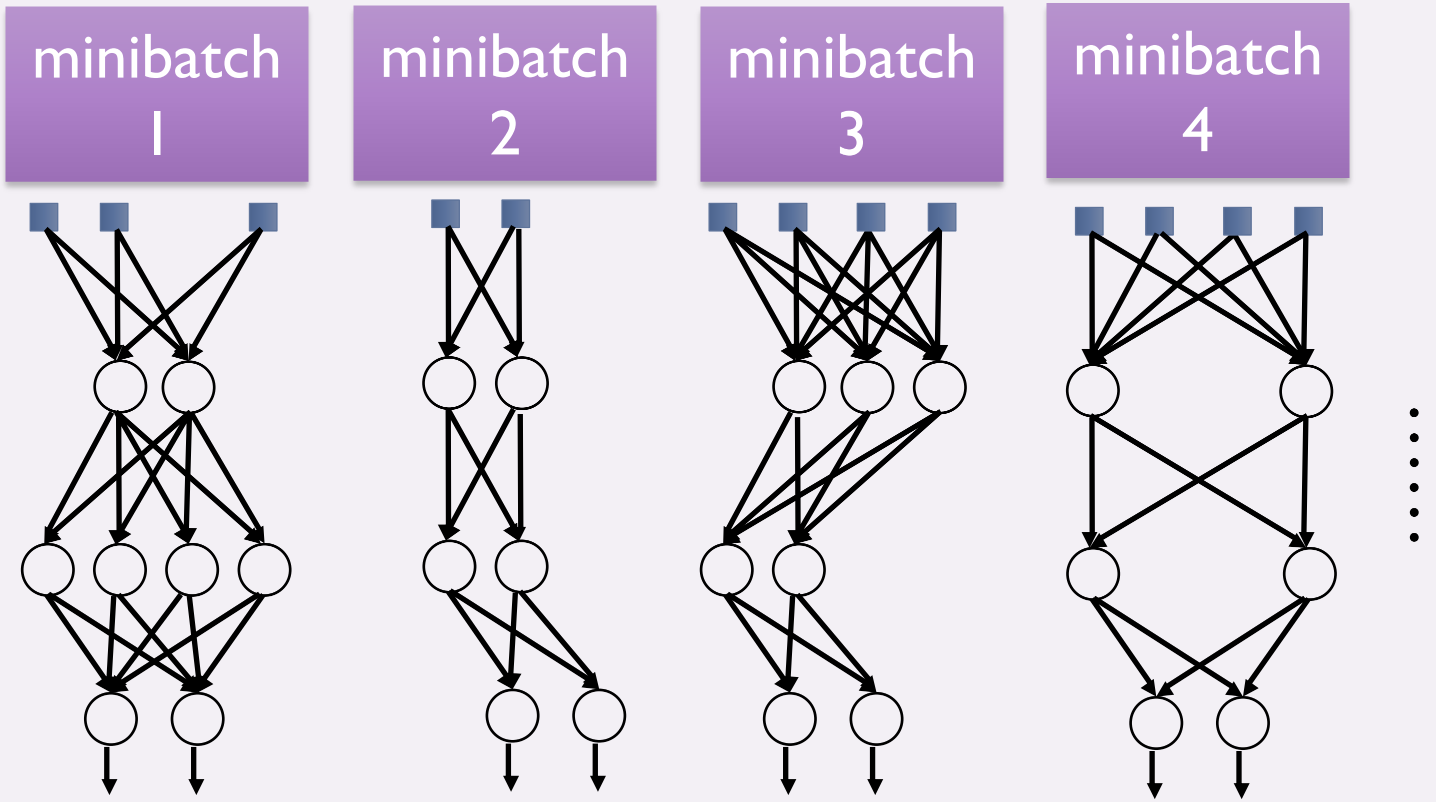
Train a bunch of networks with different structures

DROPOUT IS A KIND OF ENSEMBLE.

Ensemble



DROPOUT IS A KIND OF ENSEMBLE.



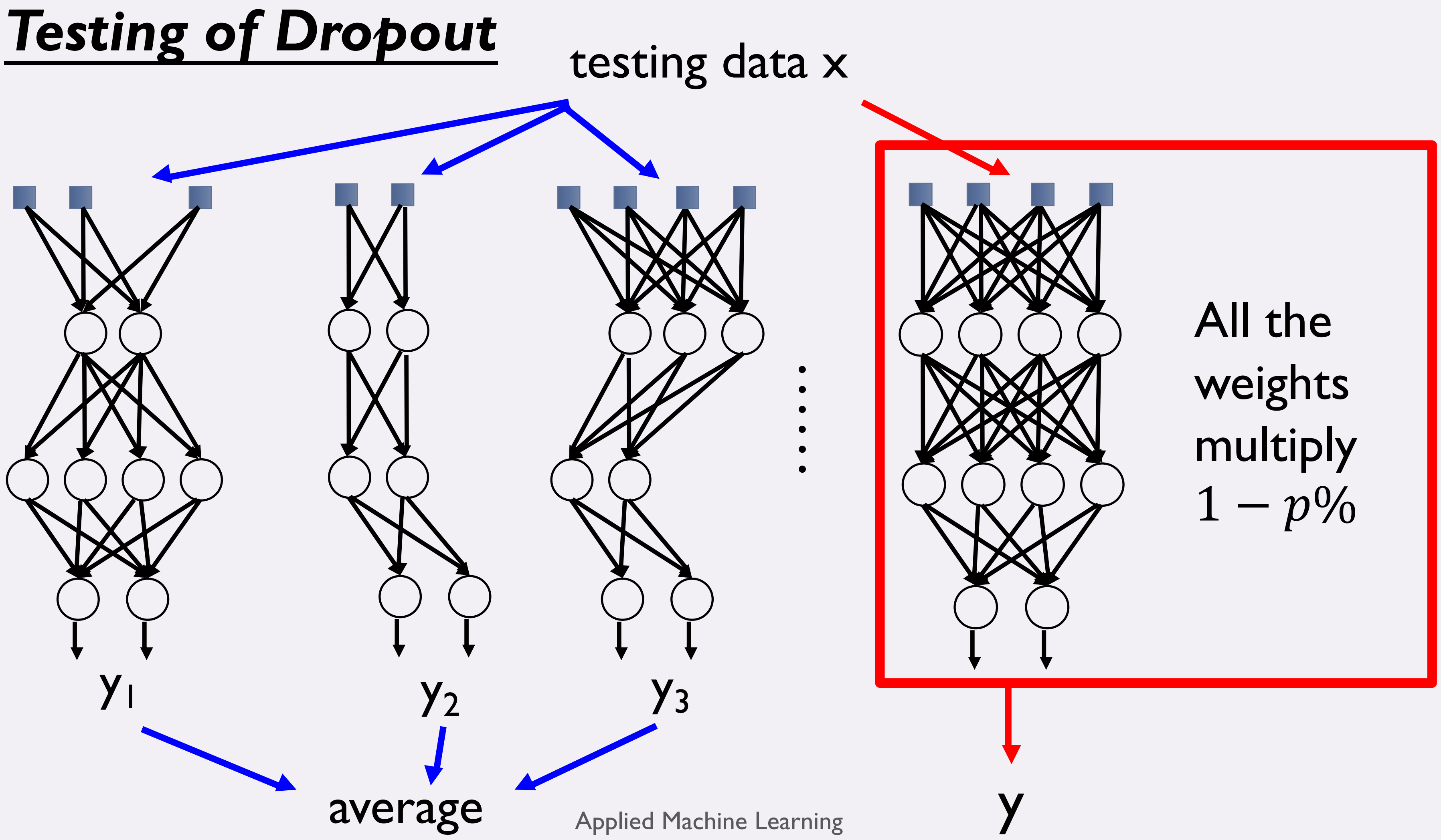
Training of Dropout

M neurons

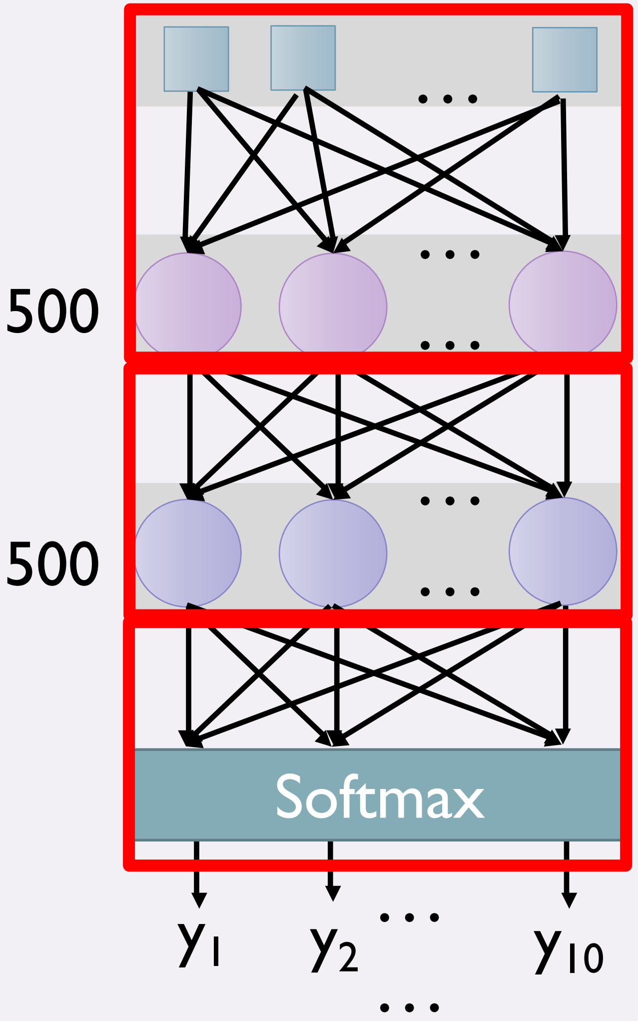
2^M possible networks

- ➢Using one mini-batch to train one network
- ➢Some parameters in the network are shared

DROPOUT IS A KIND OF ENSEMBLE.

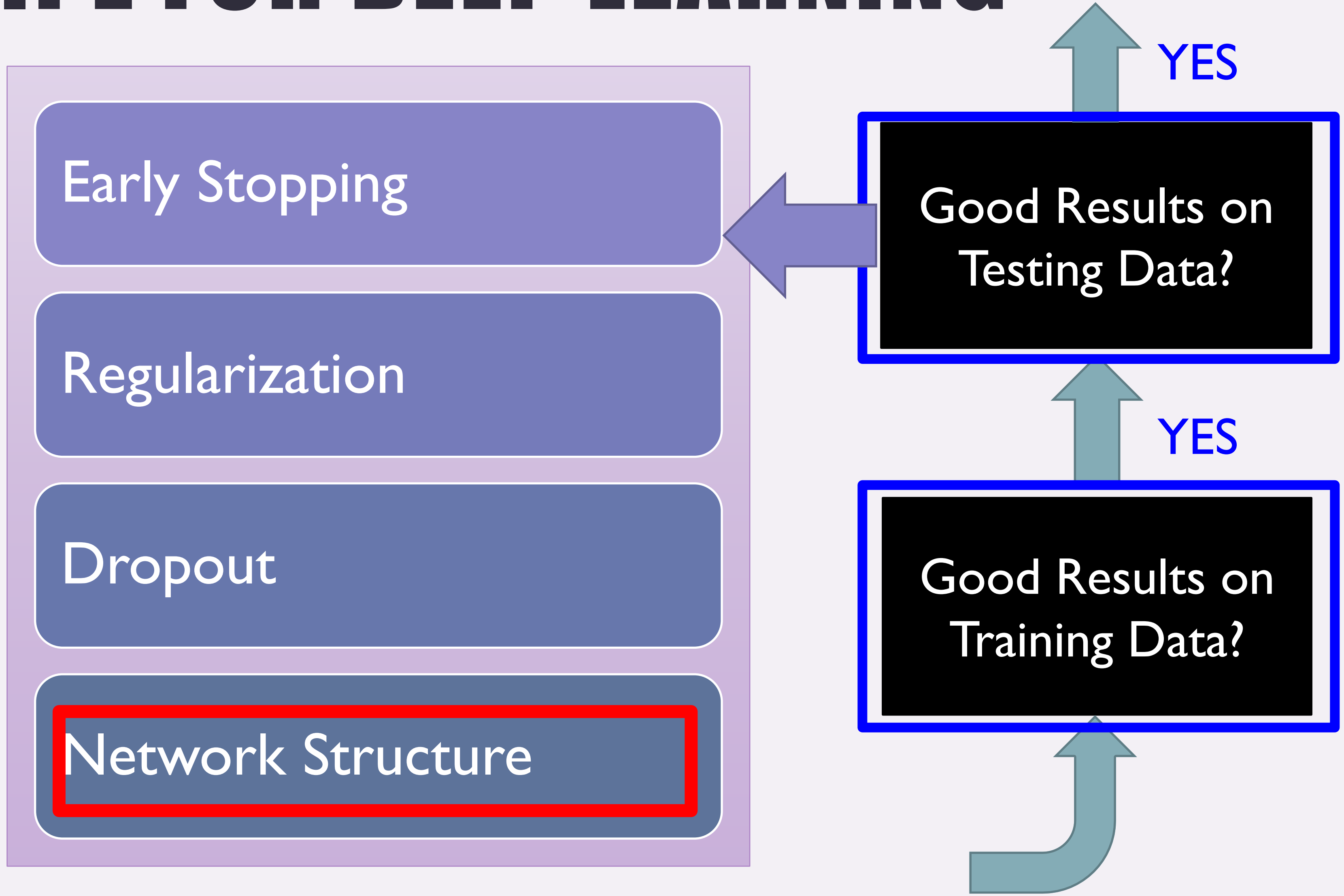


DEMO




```
model = Sequential()

model.add(Dense(units=500,
                input_dim=28*28,
                activation='sigmoid'))

model.add(Dropout(0.8))

model.add(Dense(units=500,
                activation='sigmoid'))

model.add(Dropout(0.6))

model.add(Dense(units=10,
                activation='softmax'))
```

RECIPE FOR DEEP LEARNING



VARIANTS OF NEURAL NETWORKS

- Convolutional Neural Network (CNN)
- Transformer
- Graph Neural Network (GNN)
- Recurrent Neural Network (RNN)

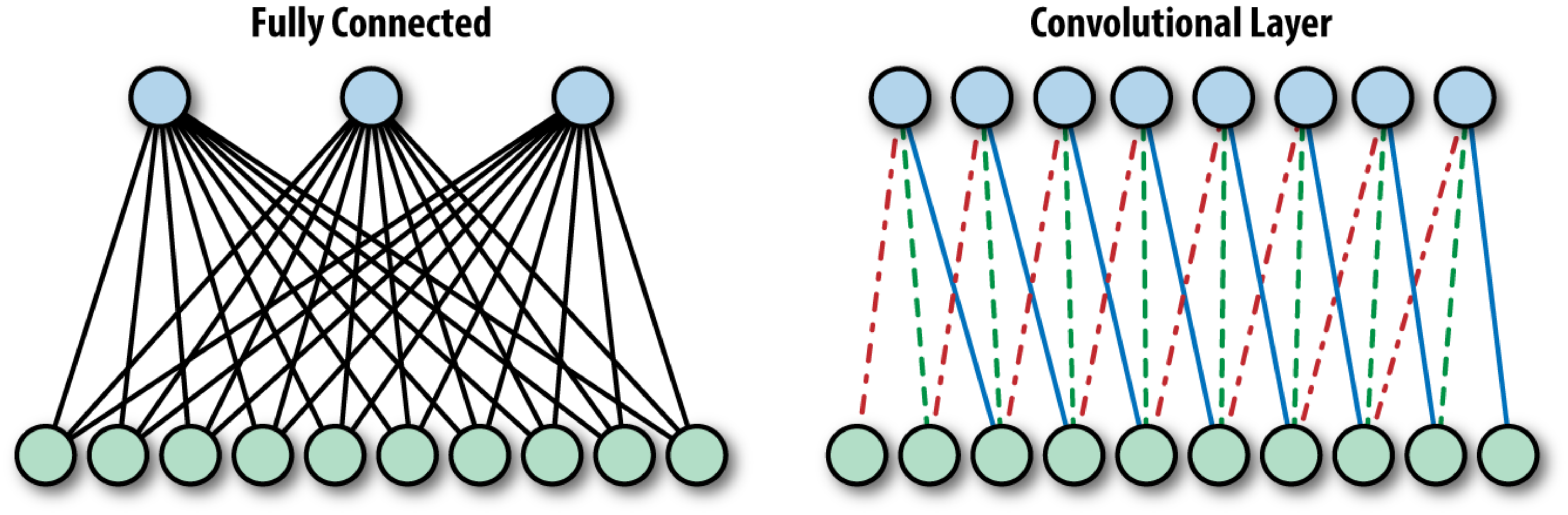


Image credit: https://www.oreilly.com/library/view/learning-tensorflow/9781491978504/ch04.html

QUESTIONS?