

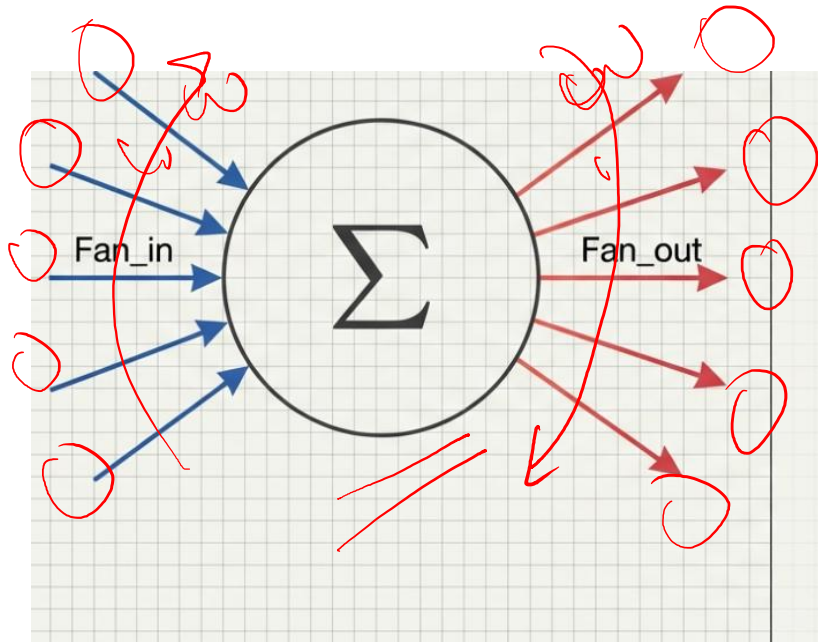
# Deep Learning Frameworks

Model Initialization, Tensorflow, Keras, Dataset, Model Creation,  
Training, Knowledge Distillation

<https://tinyurl.com/dlframeworks>  
<https://github.com/sakharamg/DeepLearningFrameworks>

# A Good Start

## Linear Layers

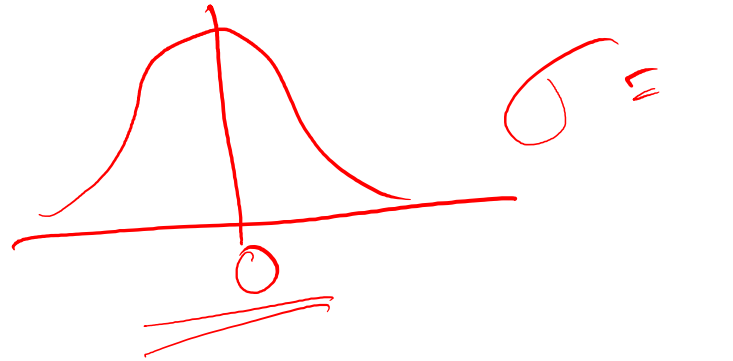


$$y_j = \sum_{i=1}^{\text{fan\_in}} W_{ji} x_i$$

Consider a single layer (ignoring bias). The output  $y$  is the weighted sum of inputs  $x$ . As we sum more inputs (higher  $\text{fan\_in}$ ), the natural tendency is for the output variance to grow.

- **Fan\_in:** The number of input connections coming *into* the neuron.
- **Fan\_out:** The number of output connections going *out* to the next layer.

$$\text{Var}(y) \approx \text{fan\_in} \cdot \text{Var}(W) \cdot \text{Var}(x)$$

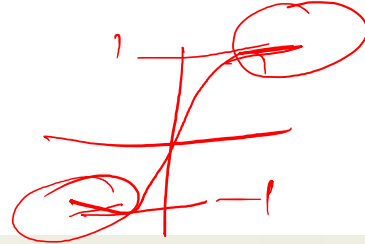


### Derivation Logic:

1. Assuming inputs and weights are independent with mean 0.
2. If 'fan\_in' is large, the output variance explodes.
3. To keep Output Variance  $\approx$  Input Variance, we must force:

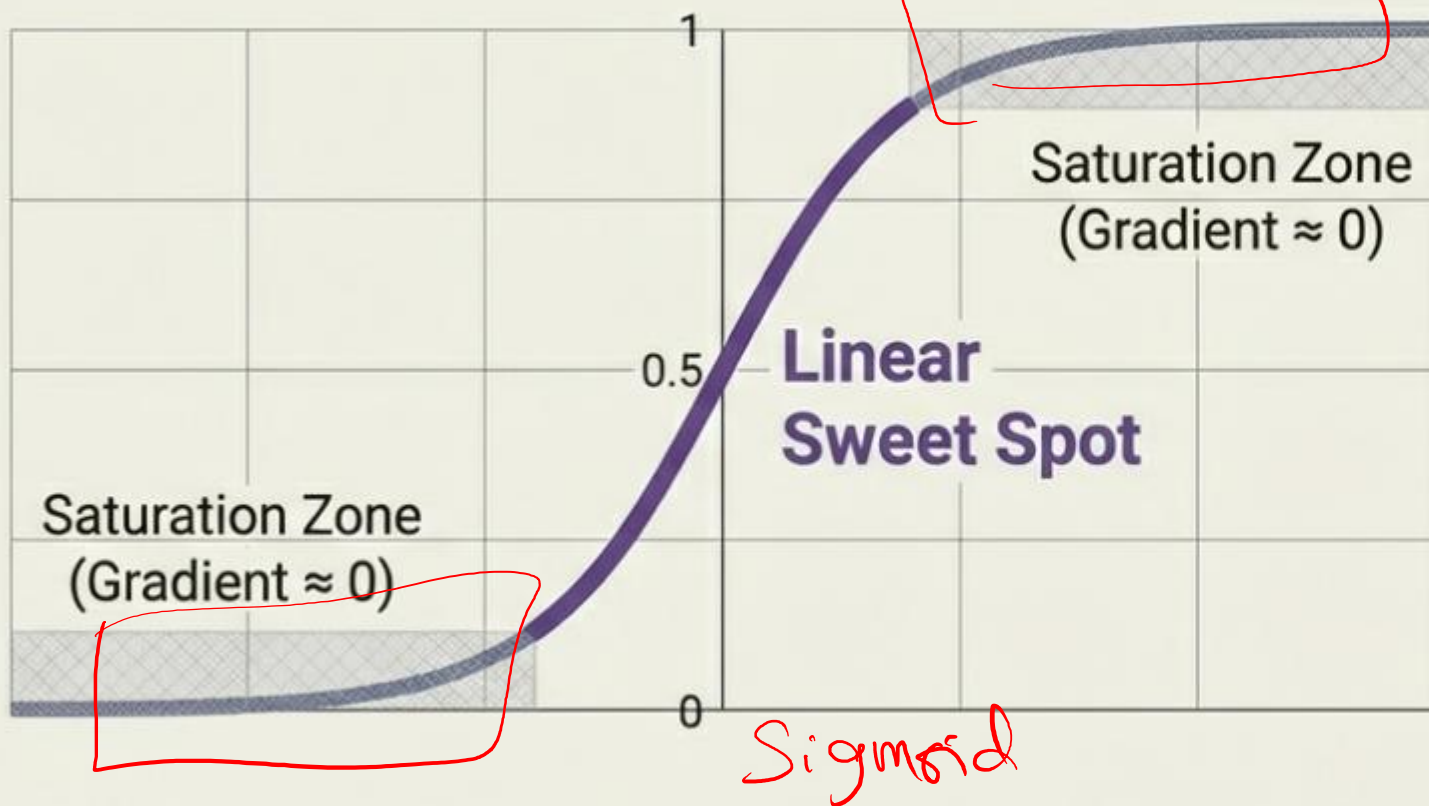
$$\text{Var}(W) \approx \frac{1}{\text{fan\_in}}$$

# Sigmoid and Tanh



`conv1 = torch.nn.Conv2d(...)`  
`torch.nn.init.xavier_uniform(conv1.weight)`

## Solution: Xavier / Glorot Initialization



The Trap: These functions saturate at the tails. If the signal is too large, gradients vanish. We must keep the signal in the “linear sweet spot”.

$\approx 0$

$\frac{f_i + f_o}{2}$

$$\text{Var}(W) = \frac{2}{\text{fan\_in} + \text{fan\_out}}$$

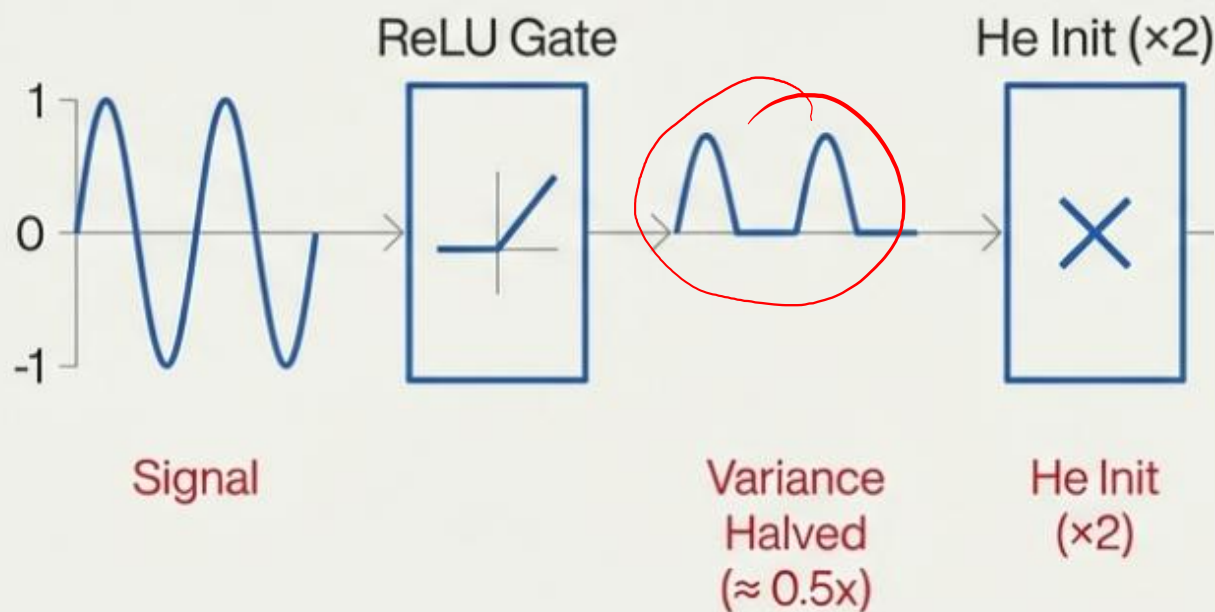
**The Harmonic Mean.** Since we cannot favor fan\_in or fan\_out, **Xavier** takes the average to maintain a compromise between forward and backward stability.

$N(0, \frac{2}{f_i + f_o})$

# ReLU

$-\infty \text{ to } \infty \Rightarrow 0 \text{ to } \infty$

## Solution: He / Kaiming Initialization



$$\text{Var}(W) = \frac{2}{\text{fan\_in}}$$

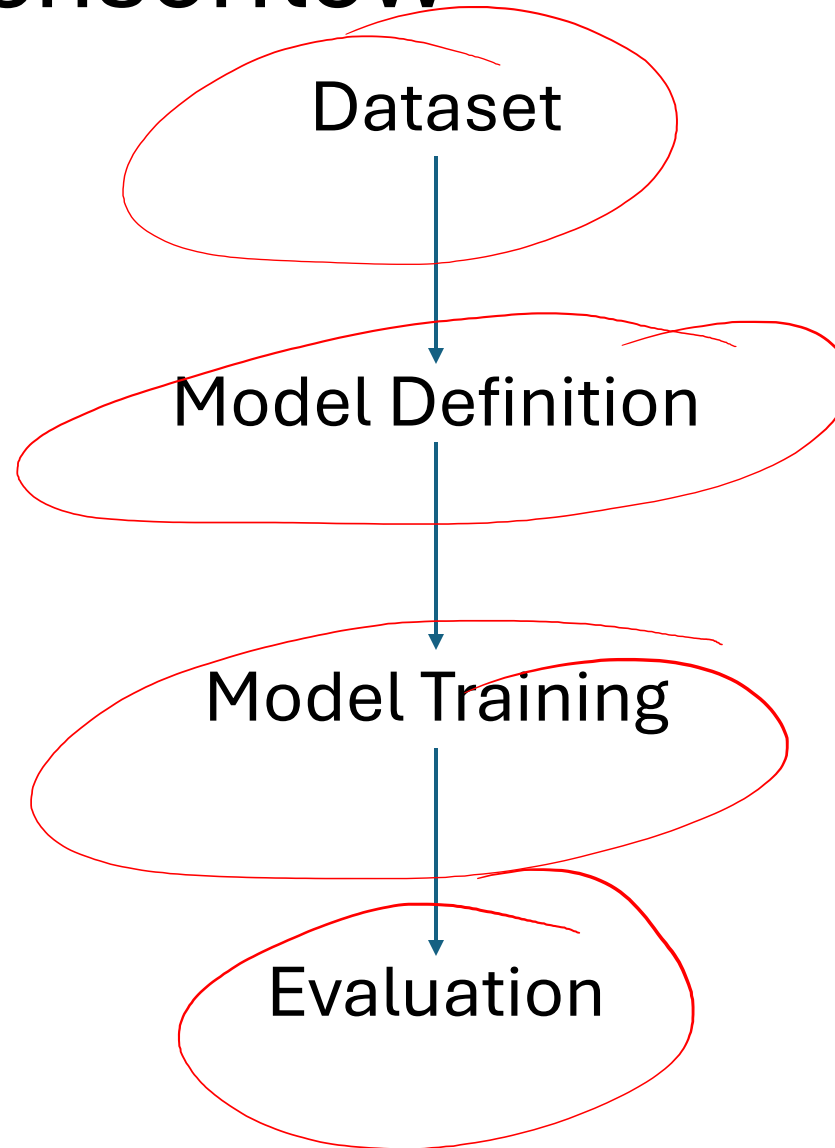
**The Trap:** ReLU kills half the signal. If the network destroys half the variance, the initialization must double the variance to compensate. This preserves the energy of the signal through deep layers.

$\frac{2}{\text{fan\_in}}$   $\rightarrow \frac{4}{\text{fan\_in}}$

$\frac{2}{\text{fan\_in}}$

for out

# PyTorch to Tensorflow



# PyTorch to Tensorflow

```
graph TD; Dataset --> MD[Model Definition]; MD --> MT[Model Training];
```

- `from torch.utils.data import Dataset, DataLoader`
- `import torchvision.transforms as T`
- `import torch.nn as nn`
- `import torch.nn.functional as F`
- `torch.optim`
- Training Loop (fixed skeleton)

Dataset

Model Definition

Model Training

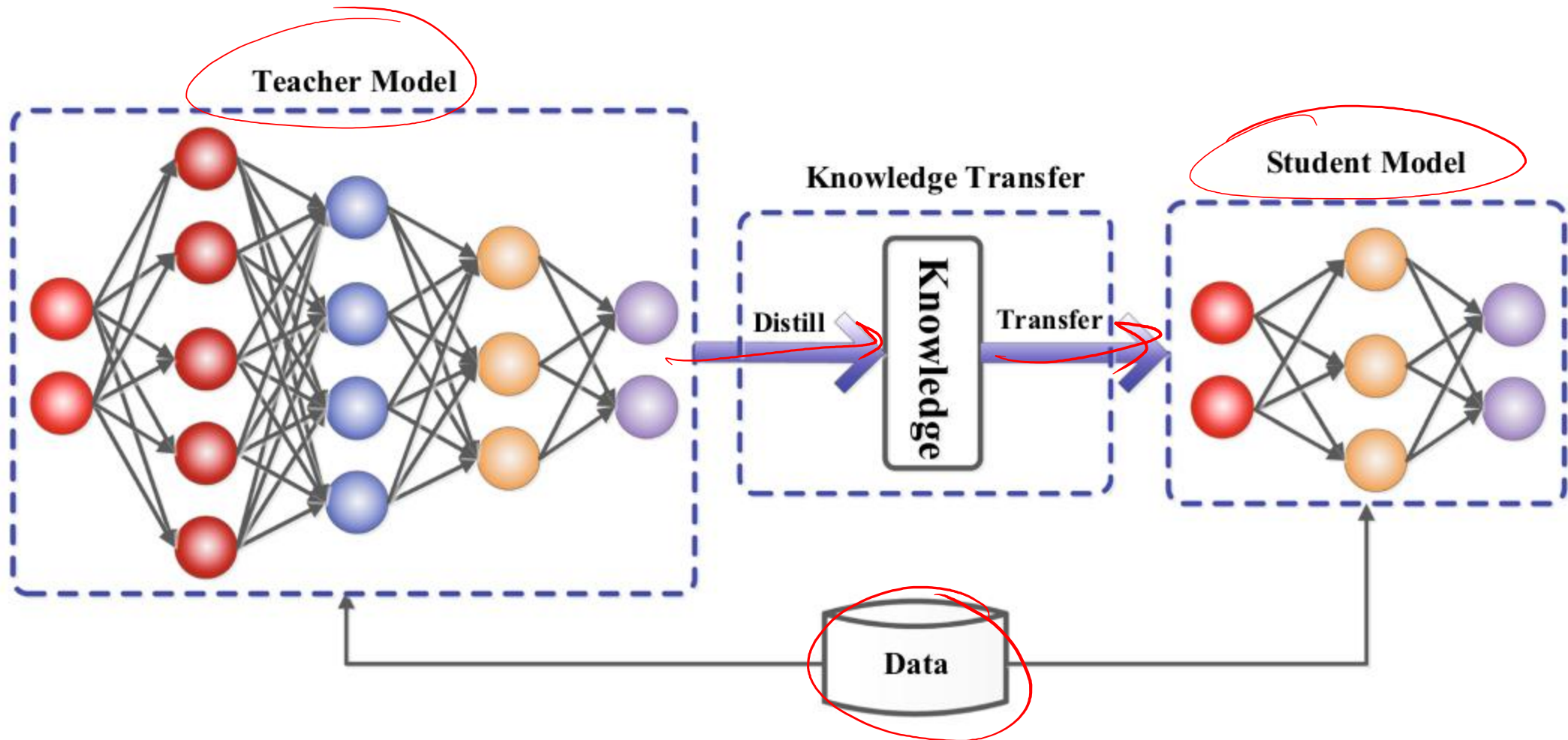
`import tensorflow.keras as keras`

- `tf.keras.optimizers`
- `model.compile`
- `model.fit`

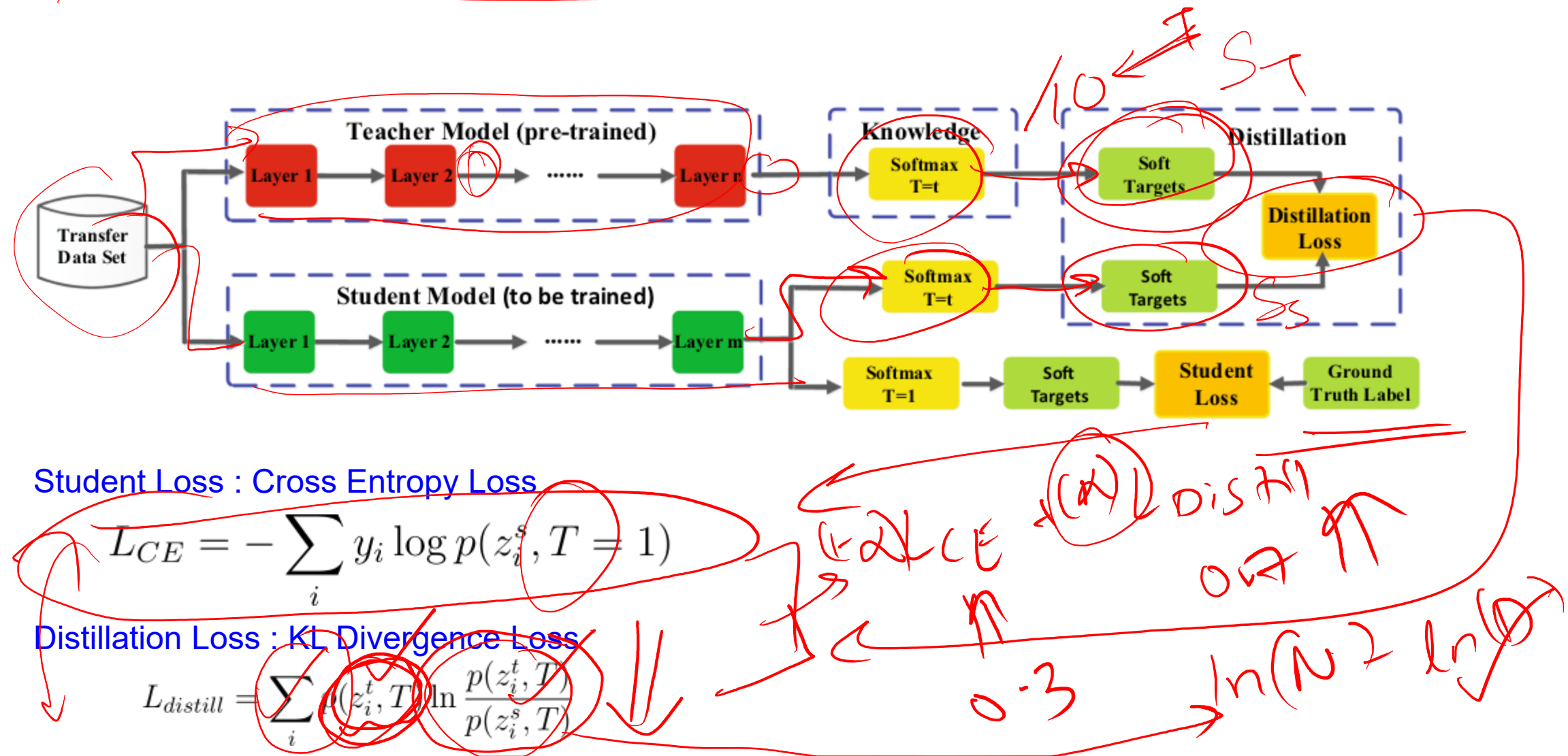
# Lab: Fully Connected Network for predicting Fashion Item (FashionMNIST)

PT  
TF

# Knowledge Distillation



# Response-based KD



$$y_h \rightarrow \frac{e^{z_i/T}}{\sum_{j=1}^{10} e^{z_j/T}}$$

Teacher o/p

20

Output of model(T=1)

Softened Output(T=10)

5.9049452e-14

0.0350216

3.8963577e-10

0.08438793

**1.0000000e+00**

**0.7365706**

1.5487779e-11

0.06112422

1.1989066e-22

0.00473253

7.1172944e-18

0.01420515

2.3661837e-17

0.0160184

2.5463203e-16

0.02031448

1.8852514e-15

0.02481712

6.4822944e-25]

0.00280794

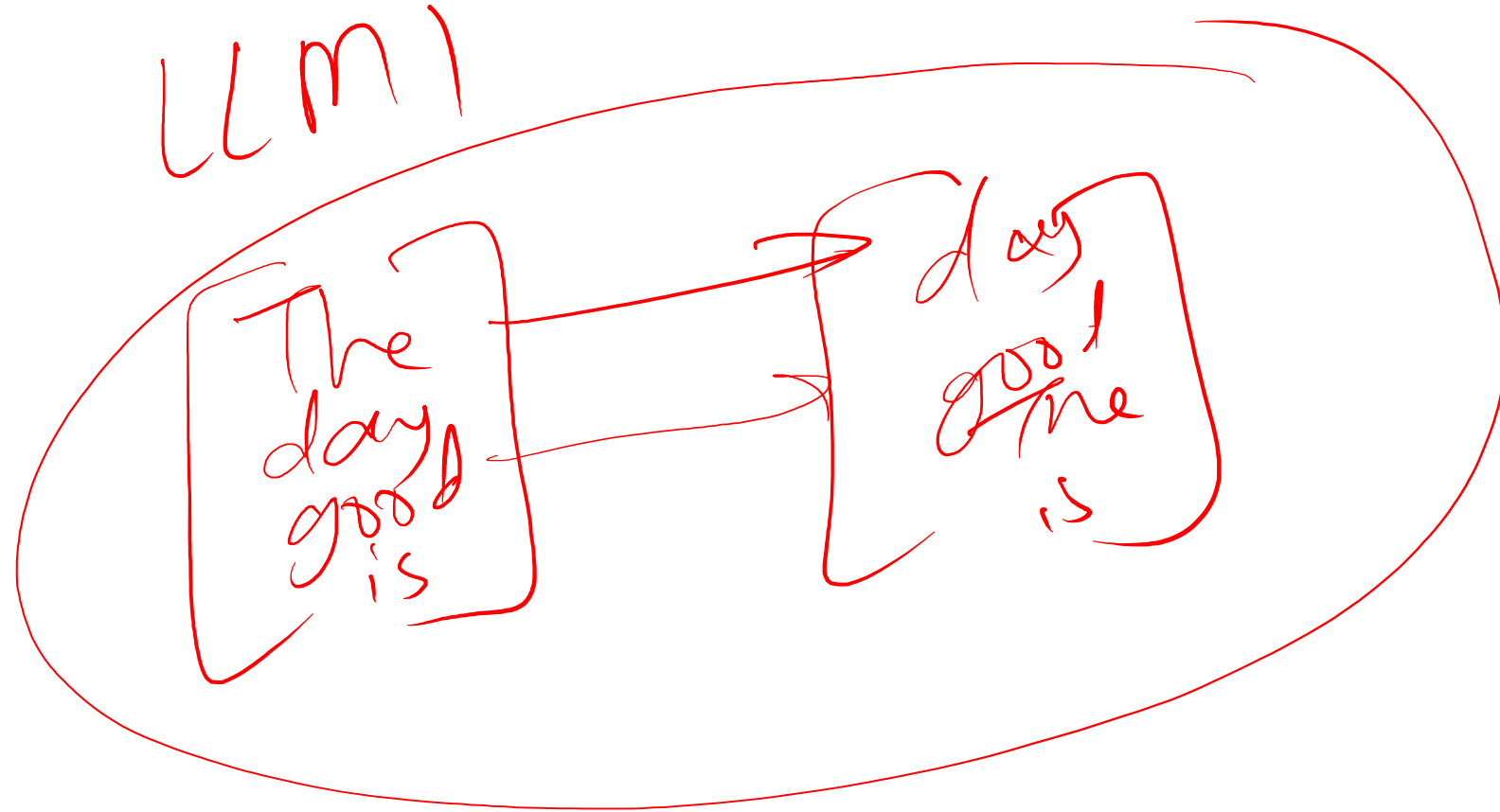
0.999 →

1/10

softmax o/p

0-0-0-0-0-0-0-0-0-0

# Lab



Thank You

# Appendix