



# Natural Language Processing

## 第六周 激活函数与齐次化

庞彦

yanpang@gzhu.edu.cn

# Overview



## CONTENTS

01



激活函数

02



齐次化



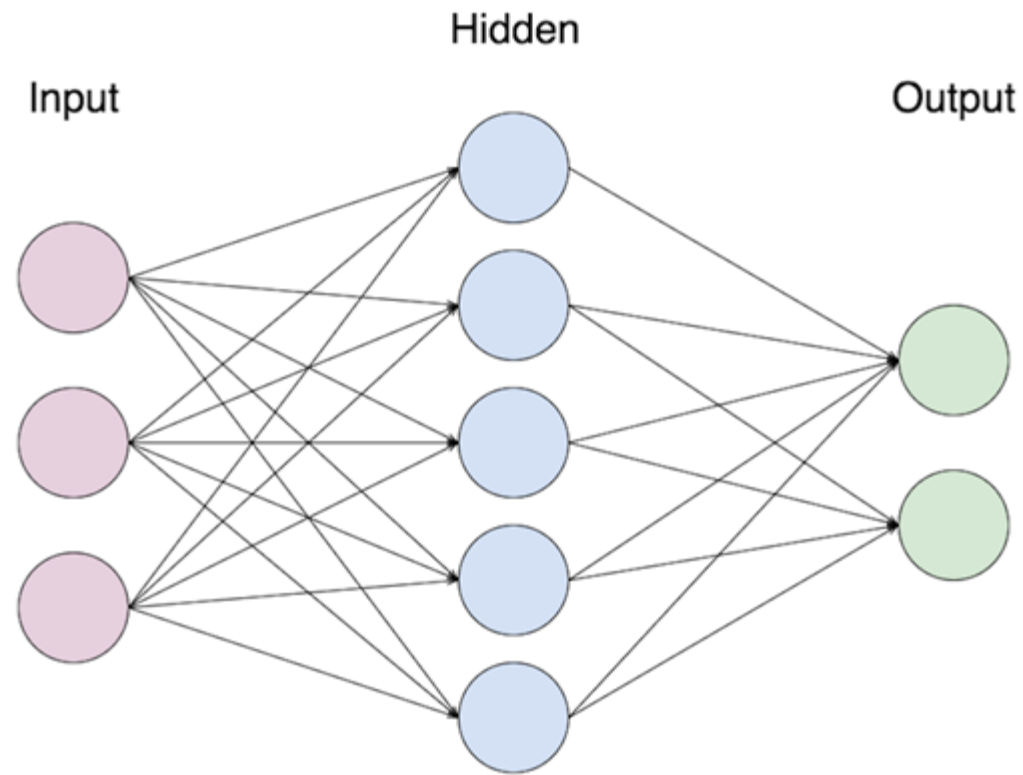
01

# Recurrent Neural Network

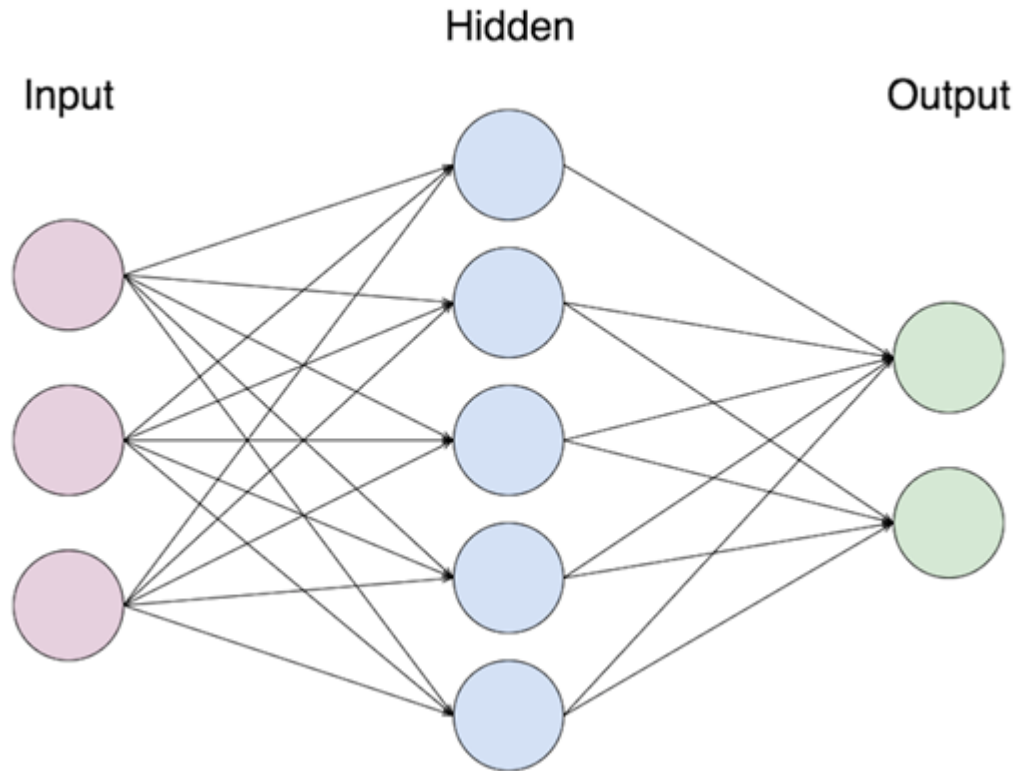
循环神经网络

Spring 2023

# Neural Network



# Neural Network



$$H = XW_h + b_h$$

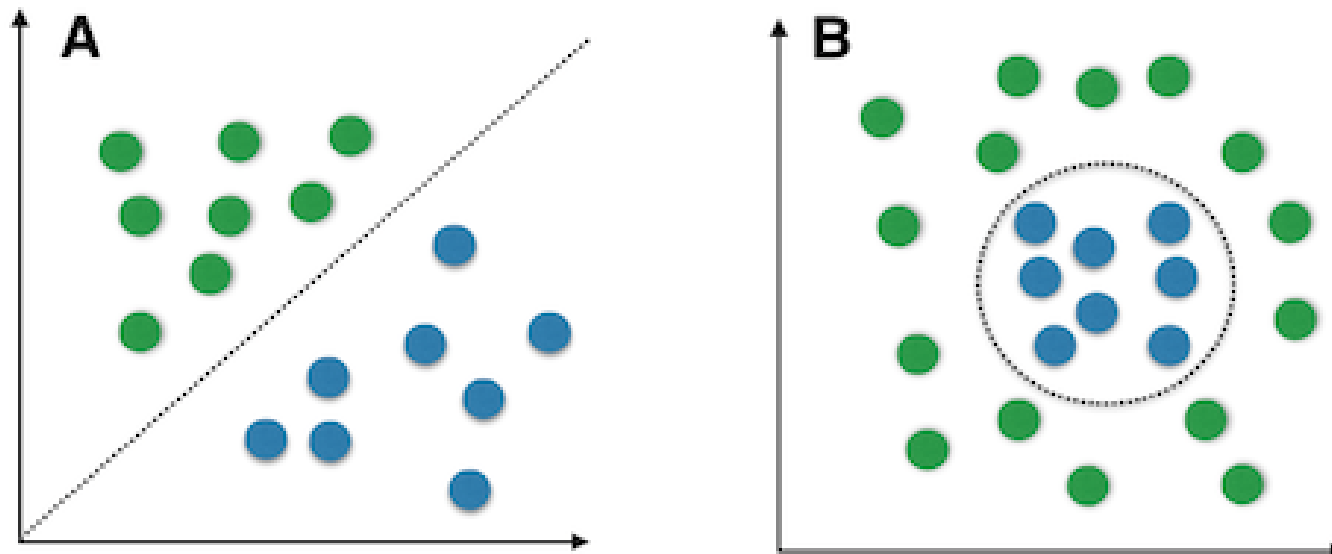
$$O = HW_o + b_o$$

$$\begin{aligned} O &= (XW_h + b_h)W_o + b_o \\ &= XW_hW_o + b_hW_o + b_o \\ &= XW + b \end{aligned}$$

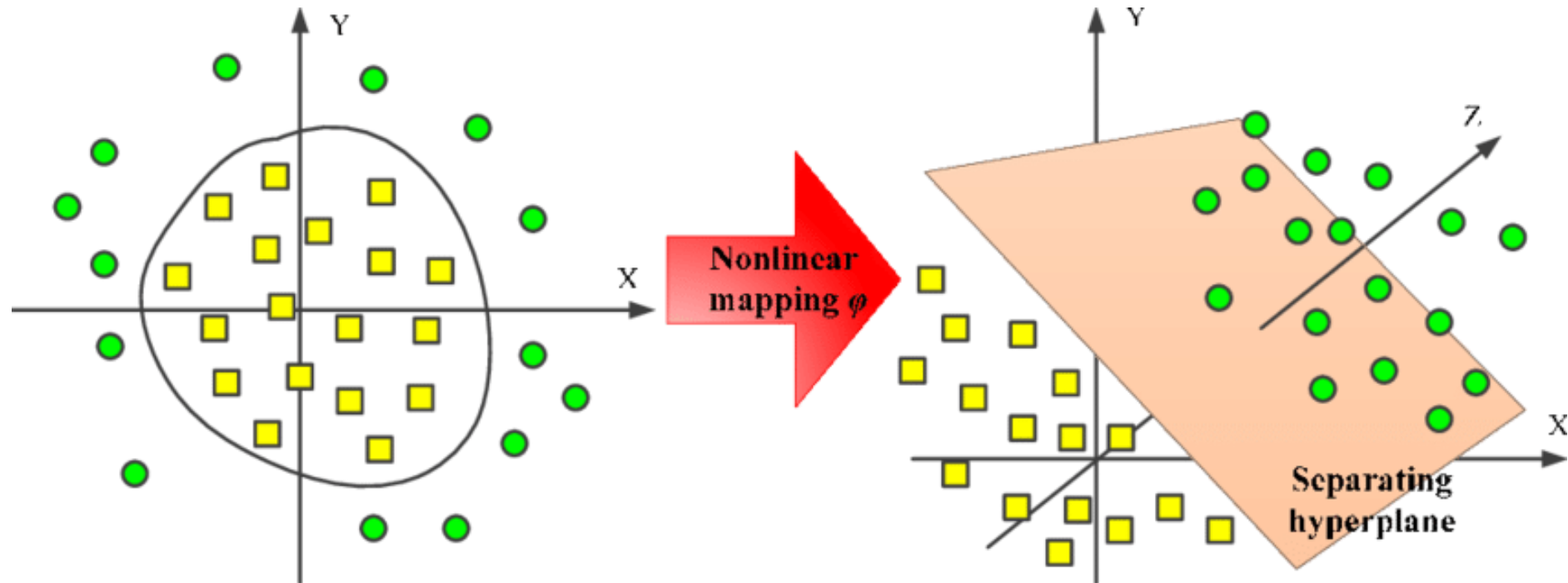
# Nonlinear



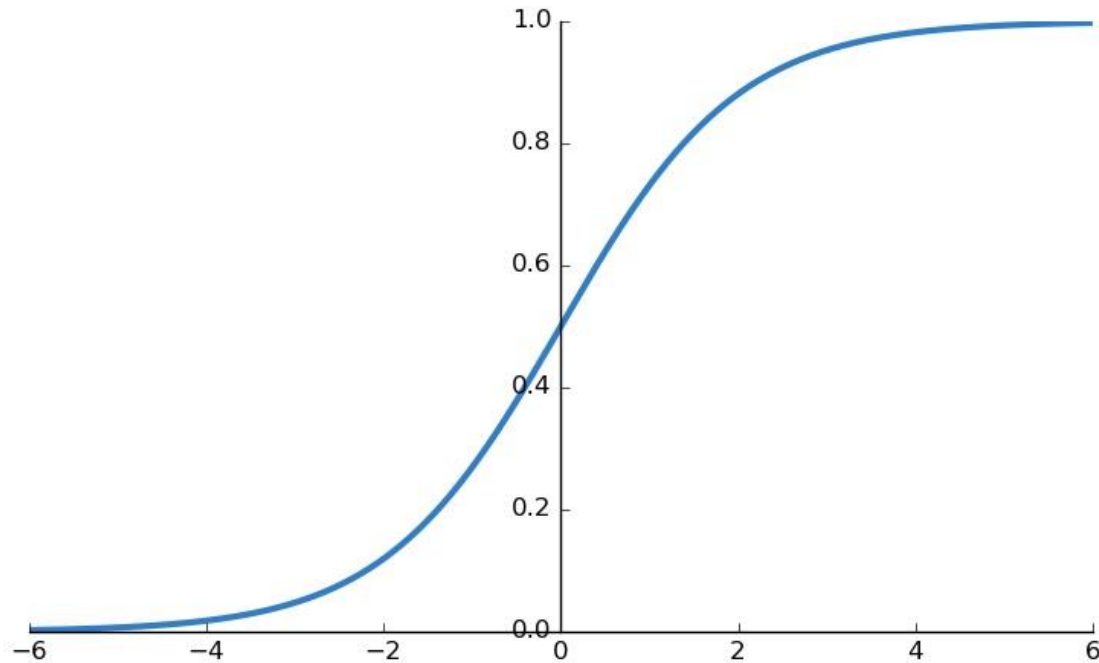
## Linear vs. nonlinear problems



# Nonlinear



# Sigmoid

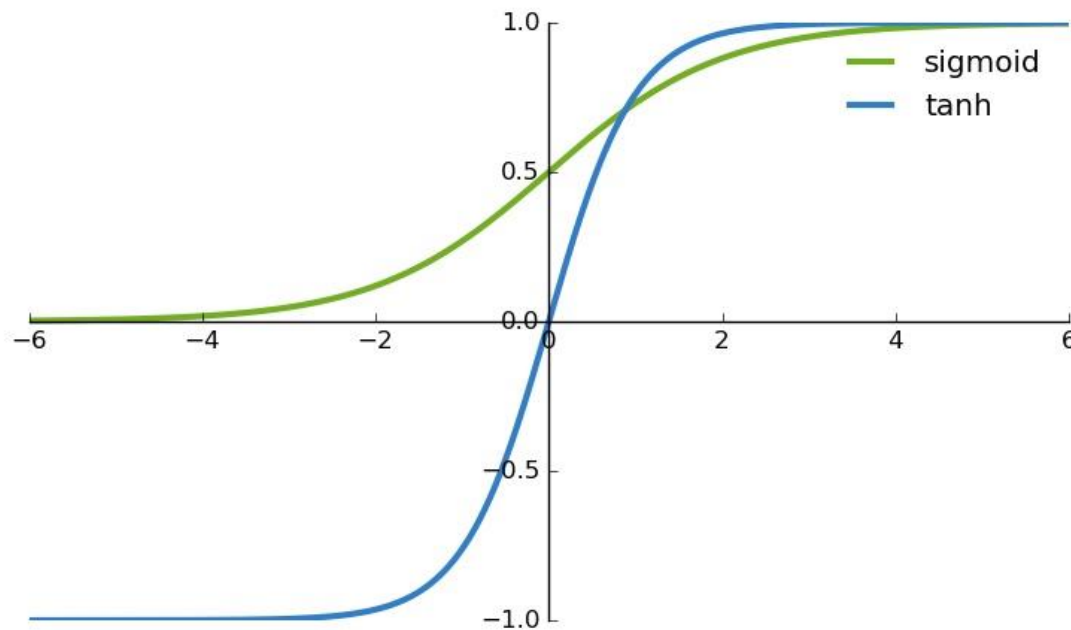


$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\begin{aligned}\sigma'(z) &= \frac{e^{-z}}{(1 + e^{-z})^2} \\ &= \sigma(z)(1 - \sigma(z))\end{aligned}$$



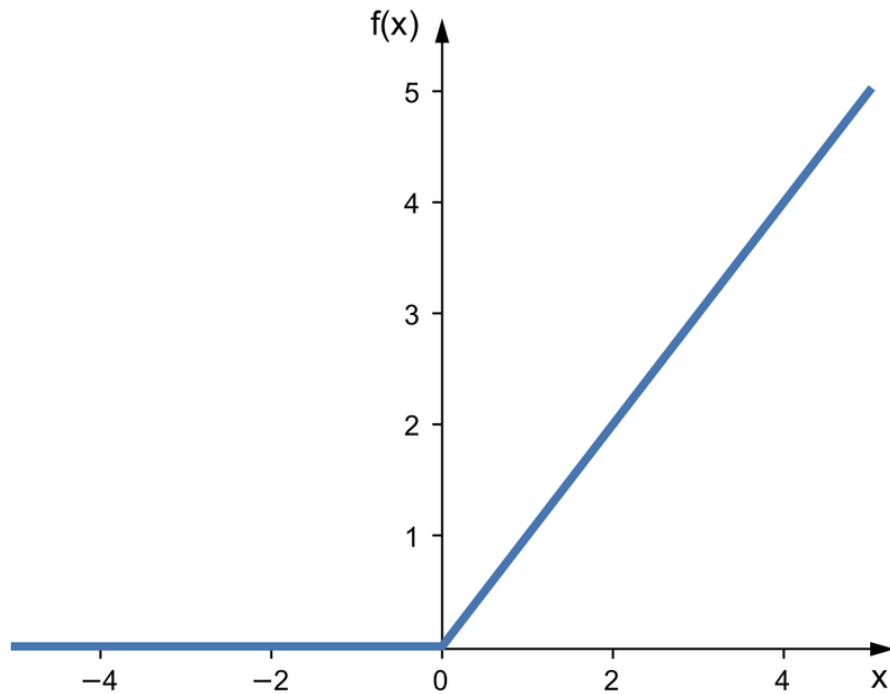
# tanh



$$\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$\begin{aligned}\sigma'(z) &= \frac{4}{(e^z + e^{-z})^2} \\ &= 1 - \sigma(z)^2\end{aligned}$$

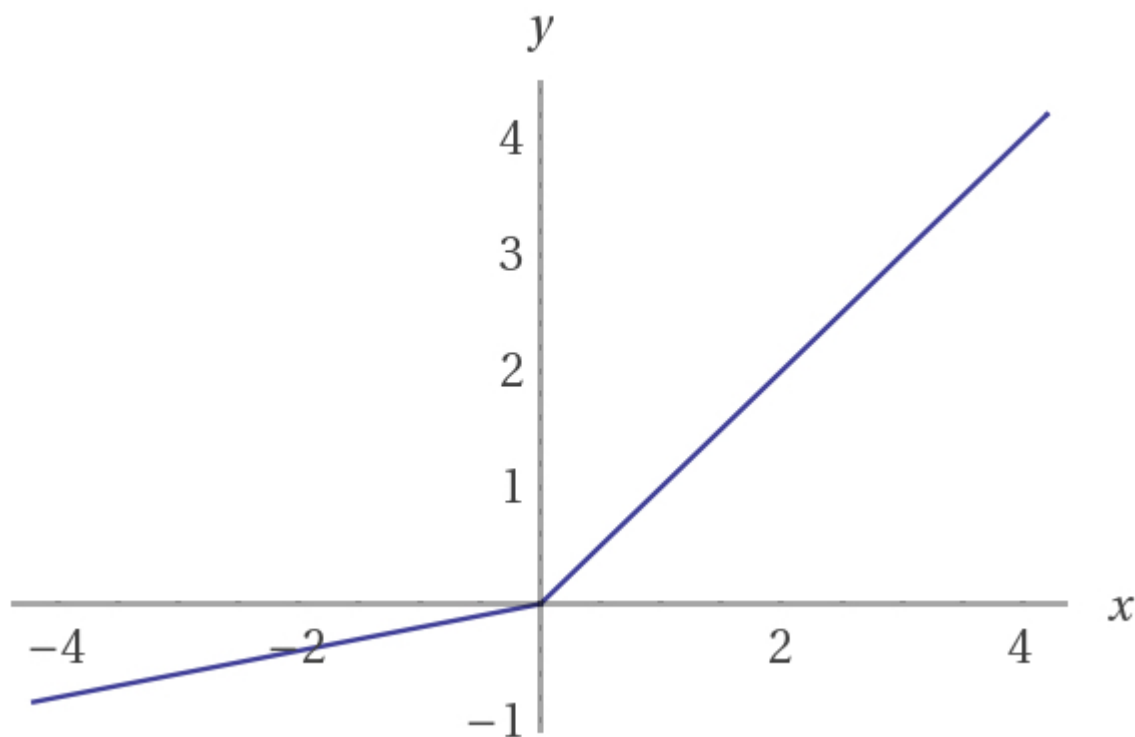
# ReLU: Rectified Linear Unit



$$\sigma(z) = \begin{cases} z, & \text{if } z > 0 \\ 0, & \text{if } z < 0 \end{cases}$$

$$\sigma'(z) = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{if } z < 0 \end{cases}$$

# Leaky ReLU



$$\sigma(z) = \begin{cases} z, & \text{if } z > 0 \\ az, & \text{if } z < 0 \end{cases}$$

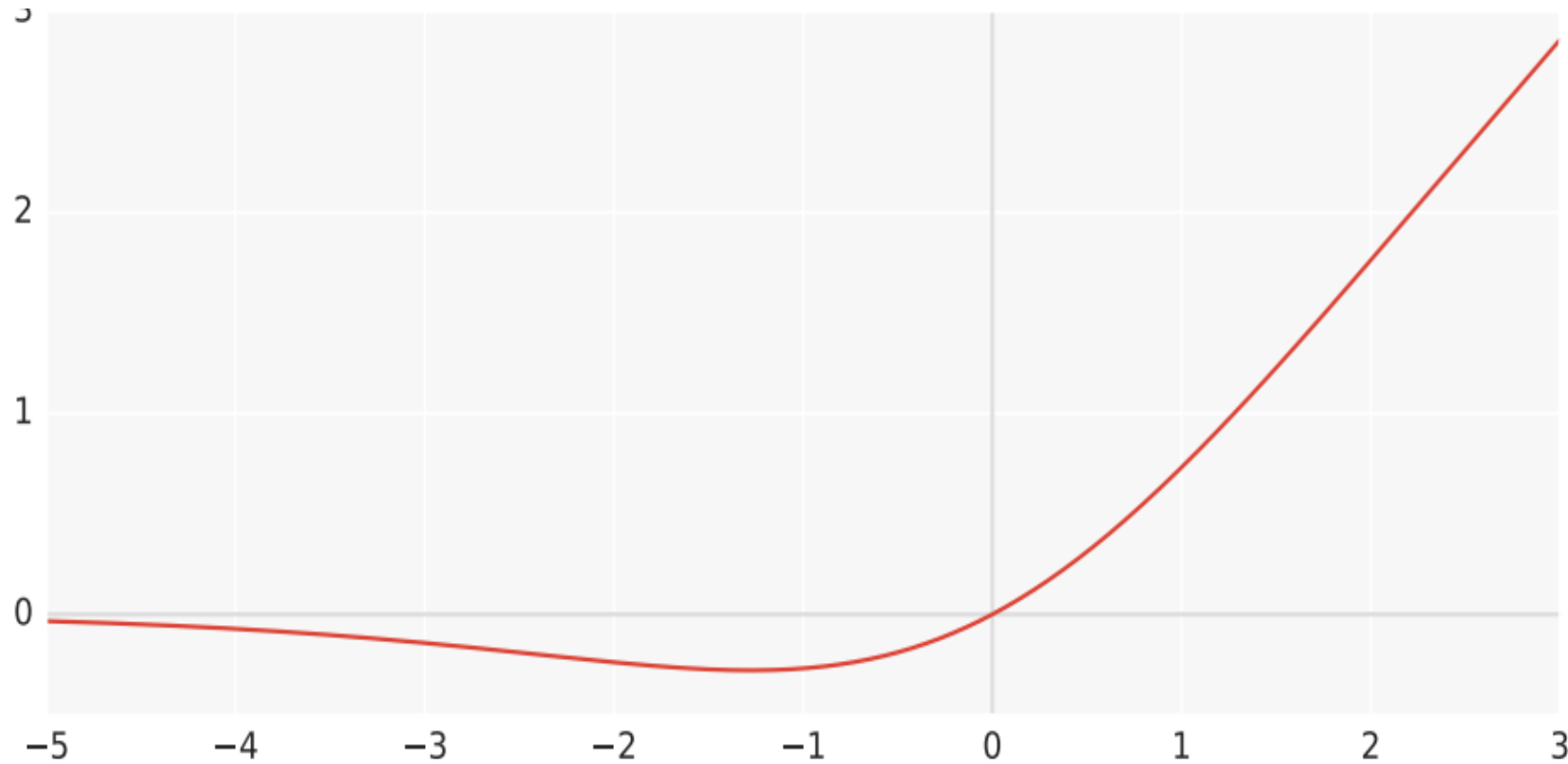
$$\sigma'(z) = \begin{cases} 1, & \text{if } z > 0 \\ a, & \text{if } z < 0 \end{cases}$$

# More ReLU



Identity	Sigmoid	TanH	ArcTan
ReLU	Leaky ReLU	Randomized ReLU	Parameteric ReLU
Binary	Exponential Linear Unit	Soft Sign	Inverse Square Root Unit (ISRU)
Inverse Square Root Linear	Square Non-Linearity	Bipolar ReLU	Soft Plus

# Swish



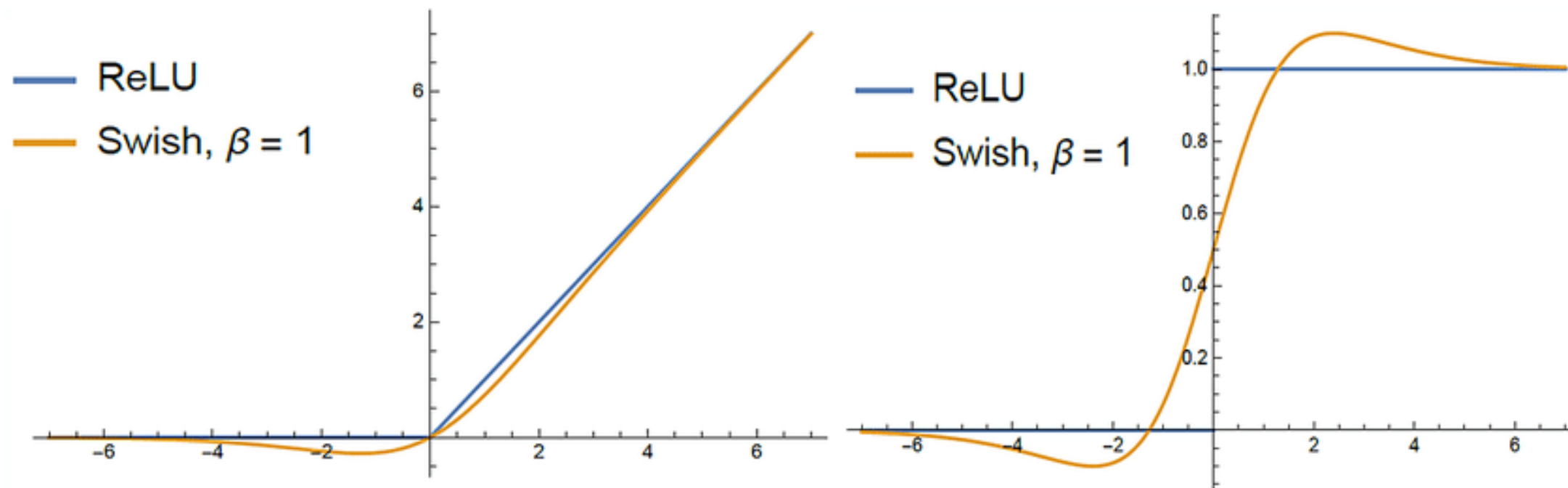
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$
$$f(z) = z \cdot \sigma(\beta z)$$

# Swish



$$f(z) = z \cdot \sigma(\beta z)$$

$$f(z)' = \beta f(z) + \sigma(\beta z)(1 - \beta f(z))$$



# Swish

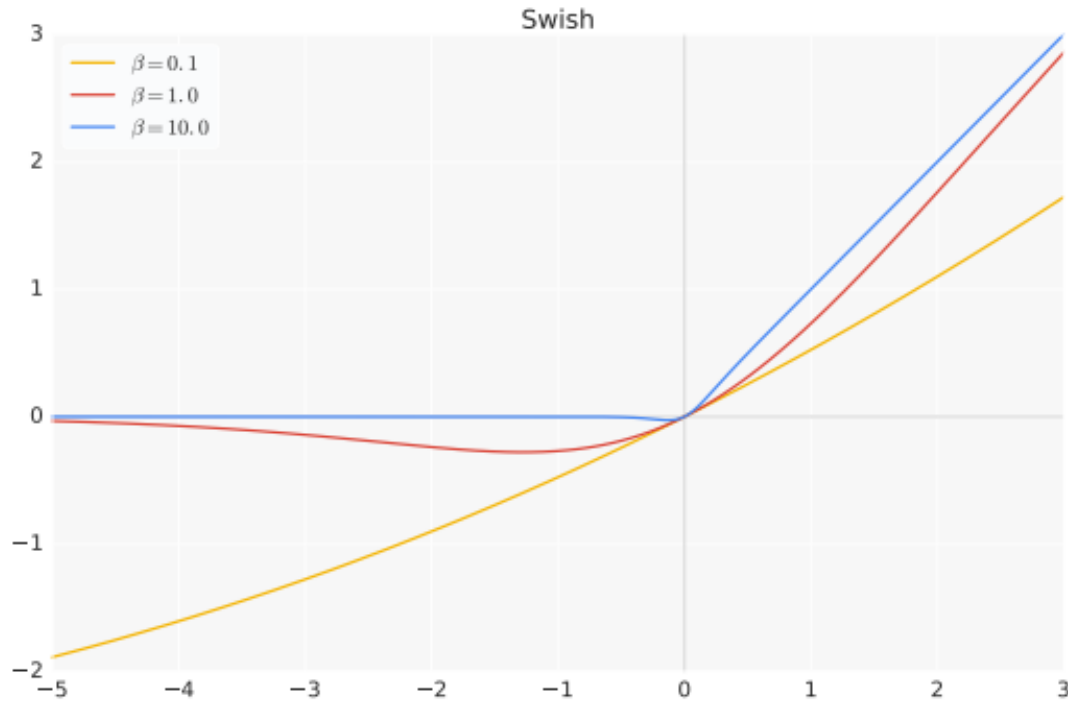


Figure 4: The Swish activation function.

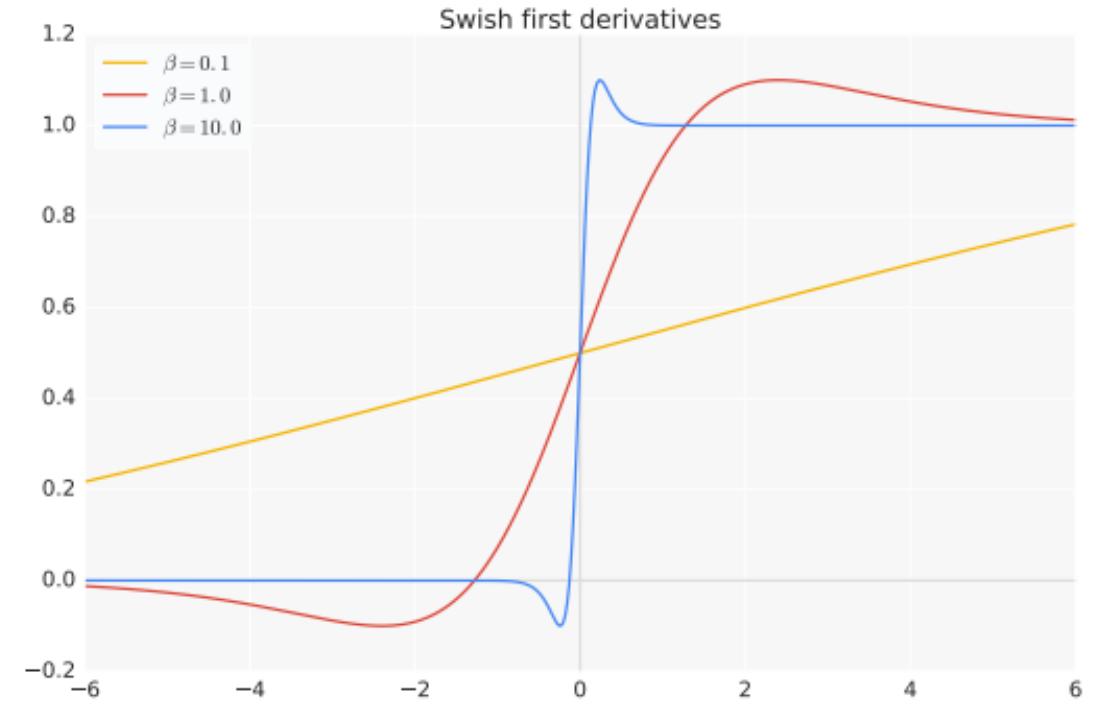
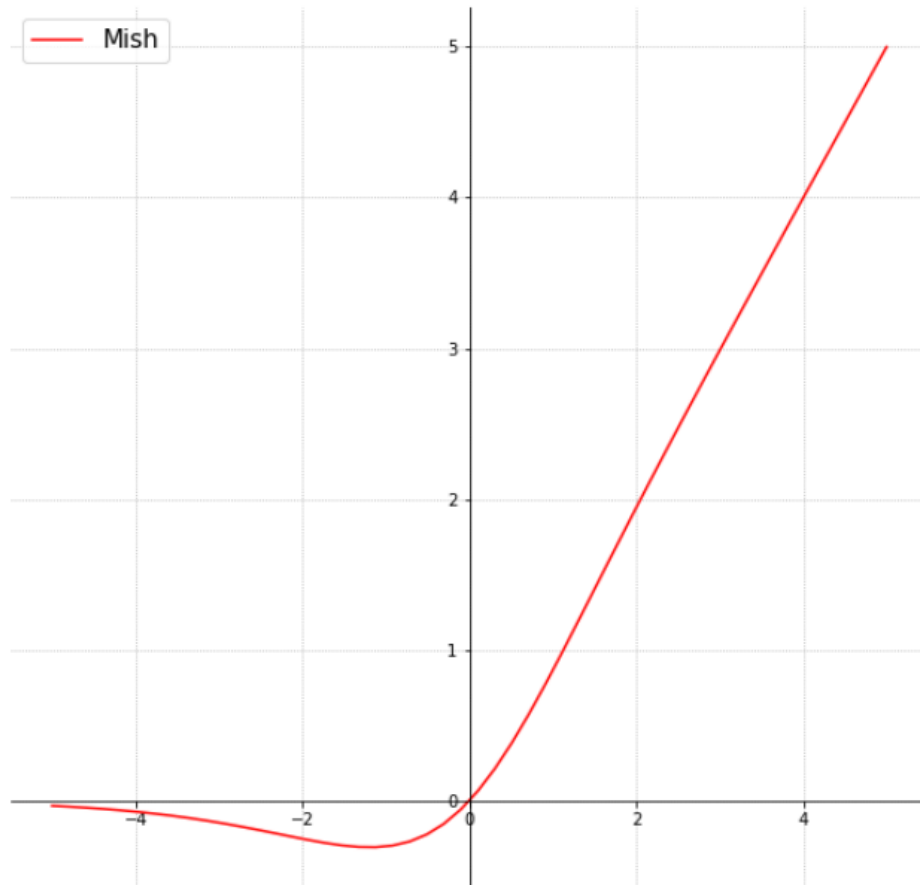


Figure 5: First derivatives of Swish.

# Mish



$$f(z) = z \cdot \tanh(\ln(1 + e^z))$$

$$f(z)' = \frac{e^z \omega}{\delta^2}$$

$$\omega = 4(z + 1) + 4e^{2z} + e^{3z} + e^z(4z + 6)$$

$$\delta = 2e^z + e^{2z} + 2$$

Activation Function	Mean Accuracy	Mean Loss	Standard Deviation (Accuracy)
Mish	<b>87.48%</b>	<b>4.13%</b>	<b>0.3967</b>
Swish	87.32%	4.22%	0.414





# 02 | Normalization

## 齐次化

Spring 2023

# Batch Normalization



Why do we need Normalization ? 为什么我们需要齐次化？

**Normalization** techniques can decrease your model's training time by a huge factor. Let me state some of the benefits of using Normalization. 齐次化技术可以大大减少模型的训练时间。

- ✓ It normalizes each feature so that they maintains the contribution of every feature, as some feature has higher numerical value than others. This way our network can be **unbiased**(to higher value features).  
它对每个特征进行归一化，以便它们保持每个特征的贡献，因为某些特征的数值高于其他特征。这样我们的网络就可以无偏见（对更高价值的特征）。
- ✓ It **reduces Internal Covariate Shift**. It is the change in the distribution of network activations due to the change in network parameters during training. To improve the training, we seek to reduce the internal covariate shift.  
它减少了内部协变量偏移。它是由于训练期间网络参数的变化而导致的网络激活分布的变化。为了改进训练，我们寻求减少内部协变量偏移。

# Batch Normalization



Why do we need Normalization ? 为什么我们需要齐次化？

**Normalization** techniques can decrease your model's training time by a huge factor. Let me state some of the benefits of using Normalization. 齐次化技术可以大大减少模型的训练时间。

- ✓ It makes the **Optimization faster** because normalization doesn't allow weights to explode all over the place and restricts them to a certain range. 它使优化更快，因为规范化不允许权重在整个地方爆炸并将它们限制在一定范围内。
- ✓ An unintended benefit of Normalization is that it helps network in **Regularization**(only slightly, not significantly). 归一化的一个意想不到的好处是它有助于正则化中的网络（仅轻微，不显著）。

# Batch Normalization



How Normalization layers behave in Distributed training ? 标准化层在分布式训练中的表现如何？

Which Normalization technique should you use for your task like CNN, RNN, style transfer etc ?

对于 CNN、RNN、风格迁移等任务，您应该使用哪种归一化技术？

What happens when you change the batch size of dataset in your training?

当您在训练中更改数据集的批量大小时会发生什么？

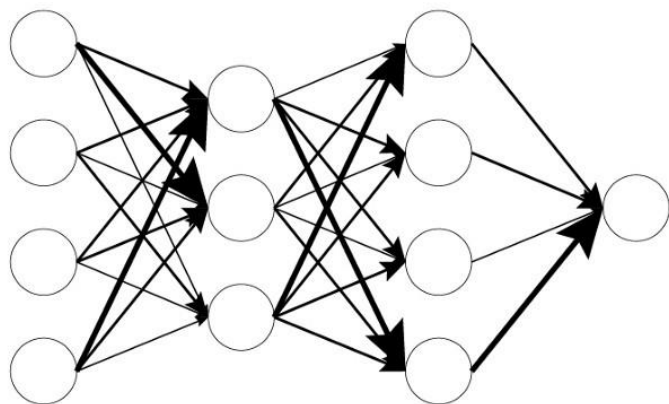
Which norm technique would be the best trade-off for computation and accuracy for your network ?

哪种规范技术将是您网络的计算和准确性的最佳权衡？

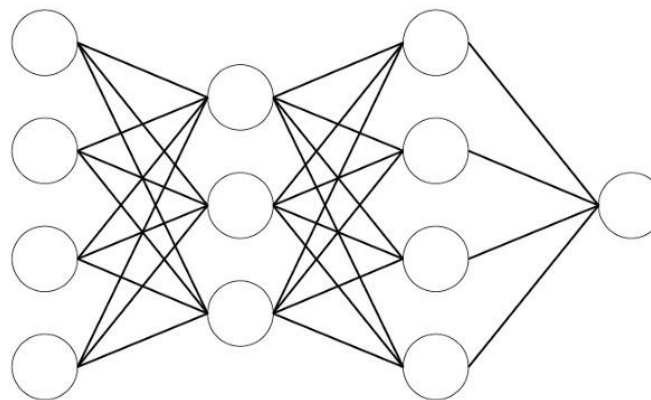
# Batch Normalization



Batch normalization is a method that normalizes activations in a network across the mini-batch of definite size.  
批次齐次化可以在很小的batch上面对激活函数做其次操作。

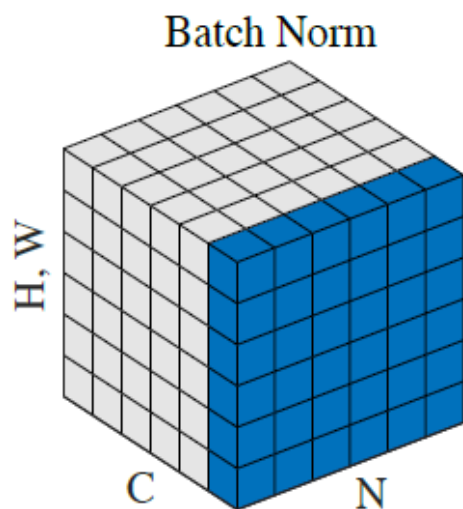


- **Raw** signal
- **High interdependency** between distributions
- **Slow** and **unstable** training



- **Normalized** signal
- **Mitigated interdependency** between distributions
- **Fast** and **stable** training

# Batch Normalization



**Input:** Values of  $x$  over a mini-batch:  $\mathcal{B} = \{x_1 \dots x_m\}$ ;

Parameters to be learned:  $\gamma, \beta$

**Output:**  $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

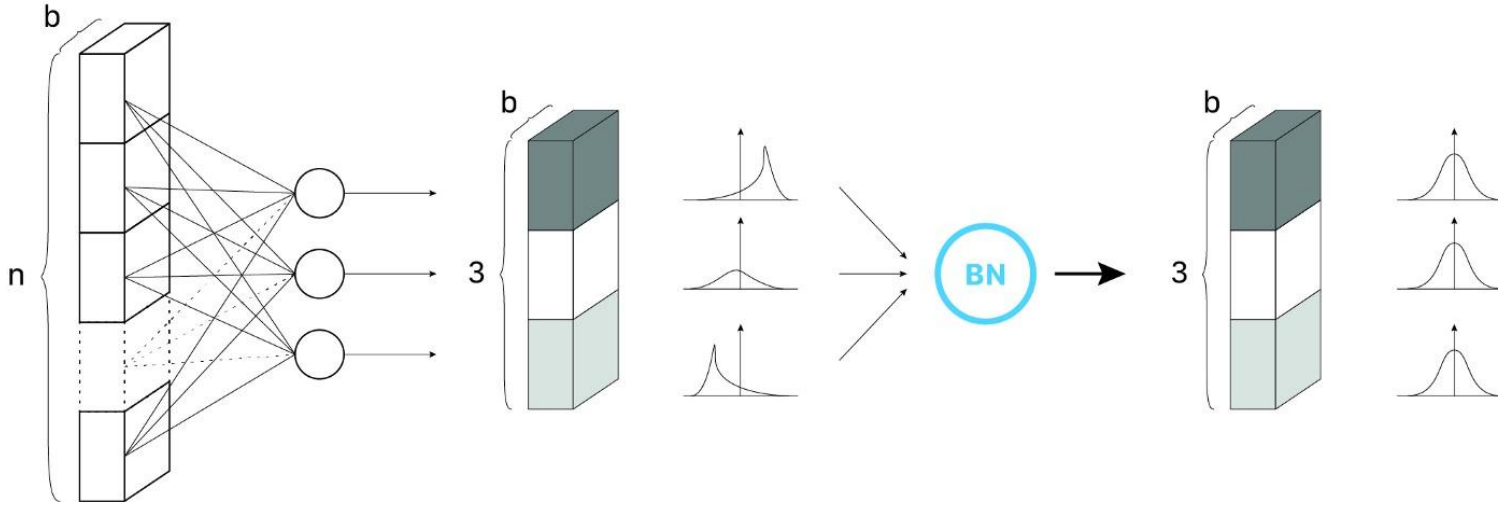
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

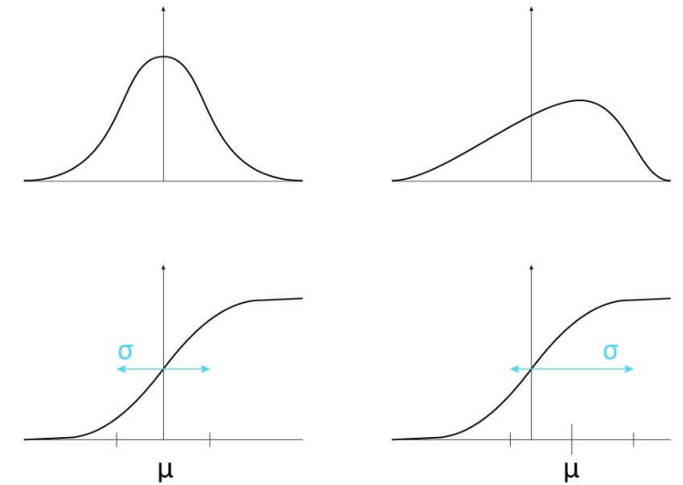
$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

# Batch Normalization



Batch Normalization first step. Example of a 3-neurons hidden layer, with a batch of size  $b$ . Each neuron follows a standard normal distribution.

- ❖  $\gamma$  allows to adjust the standard deviation 标准差;
- ❖  $\beta$  allows to adjust the bias, shifting the curve on the right or on the left side. 偏差



# Batch Normalization



Problems associated with Batch Normalization:

**Variable Batch Size** → If batch size is of 1, then variance would be 0 which doesn't allow batch norm to work. Furthermore, if we **have small mini-batch size** then it becomes too noisy and training might affect. There would also be a problem in distributed training. As, if you are computing in different machines then you have to take same batch size because otherwise  $\gamma$  and  $\beta$  will be different for different systems.

BN对batch的大小非常敏感。

BN实际使用时需要计算并且保存某一层神经网络batch的均值和方差等统计信息。

**Recurrent Neural Network** → In an **RNN**, the recurrent activations of each time-step will have a **different story** to tell(i.e. statistics). This means that we have to fit a separate batch norm layer for each time-step. This makes the model more complicated and space consuming because it forces us to store the statistics for each time-step during training. RNN的深度不是固定的。

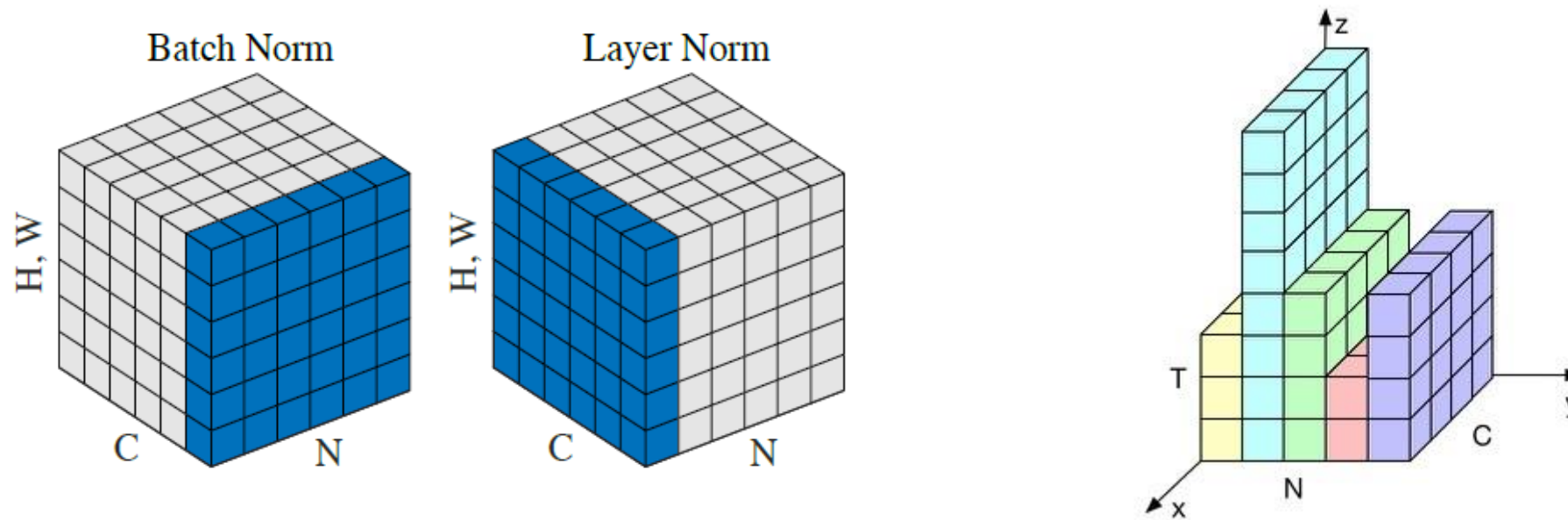


# Layer Normalization



**Layer normalization** improves the **training speed** for various neural network models. 层齐次化可以提升训练速度。

Layer normalization **normalizes input across the features** instead of normalizing input features across the batch dimension in batch normalization. 层齐次化是对每条数据内部做其次操作。



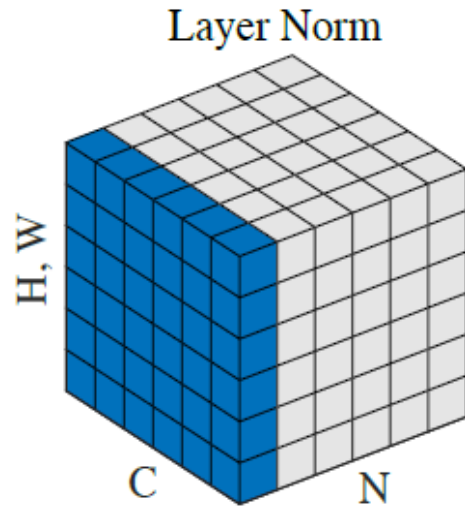
# Layer Normalization

If we would like to represent real numbers, we have to give up perfect precision.

BN与LN的区别在于:

LN中同层神经元输入拥有相同的均值和方差, 不同的输入样本有不同的均值和方差;

BN中则针对不同神经元输入计算均值和方差, 同一个batch中的输入拥有相同的均值和方差。

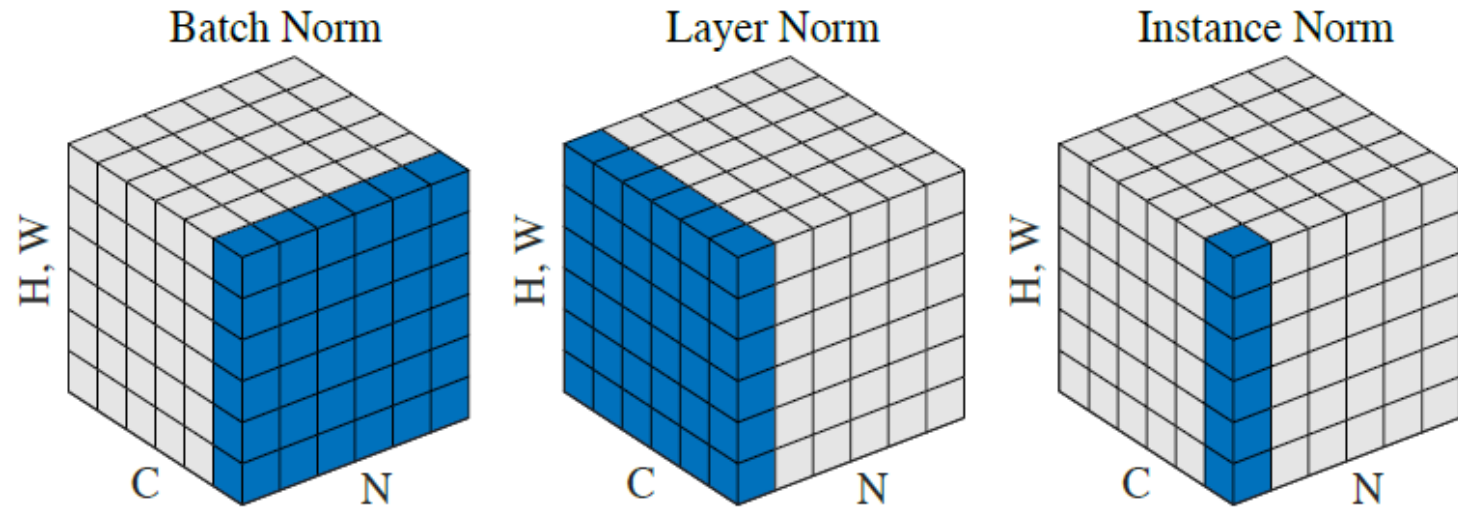


$$\mu^l = \frac{1}{H} \sum_{i=1}^H a_i^l \quad \sigma^l = \sqrt{\frac{1}{H} \sum_{i=1}^H (a_i^l - \mu^l)^2}$$

# Instance Normalization

Layer normalization and **instance normalization** is very similar to each other but the difference between them is that instance normalization **normalizes across each channel** in each **training** example instead of normalizing across input features in a training example. 实例齐次化需要对每个通道做齐次化。

Unlike batch normalization, the **instance normalization layer** is applied **at test time as well** (due to non-dependency of mini-batch). 测试的时候也要做实例齐次化。



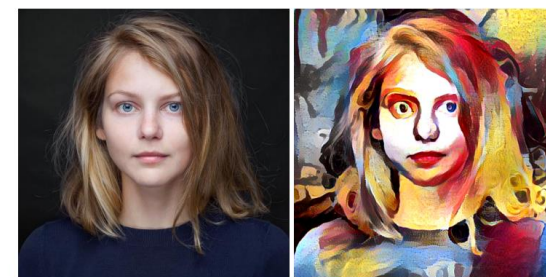
# Instance Normalization



Figure 1: Artistic style transfer example of Gatys et al. (2016) method.

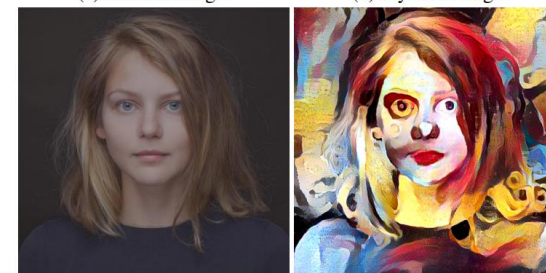
$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}, \quad \mu_{ti} = \frac{1}{HW} \sum_{l=1}^W \sum_{m=1}^H x_{tilm},$$

$$\sigma_{ti}^2 = \frac{1}{HW} \sum_{l=1}^W \sum_{m=1}^H (x_{tilm} - \mu_{ti})^2.$$



(a) Content image.

(b) Stylized image.



(c) Low contrast content image.

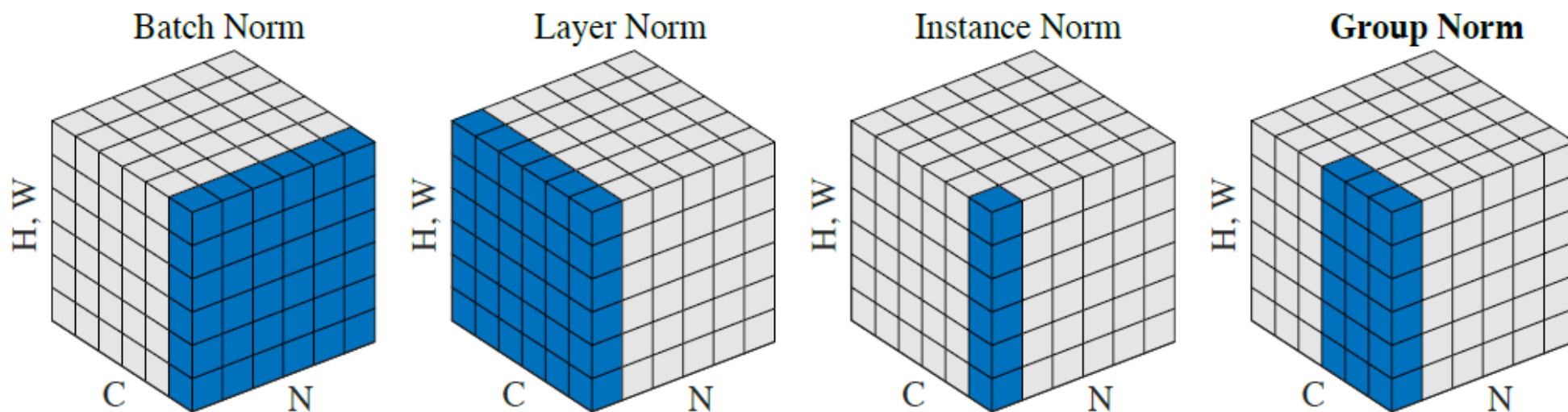
(d) Stylized low contrast image.

# Group Normalization



**Group Normalization** normalizes over group of channels for each training examples. 对每个训练示例的通道组进行归一化

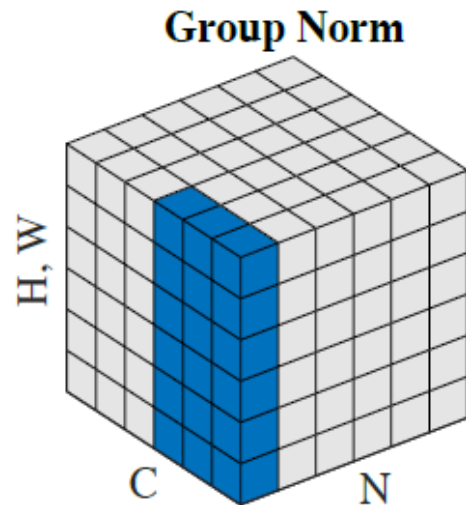
- When we put all the channels into a **single group**, group normalization becomes **Layer normalization**.  
当我们将所有通道放到一个组中时，组归一化就变成了层归一化。
- When we put **each channel into different groups** it becomes **Instance normalization**.  
当我们将每个通道放入不同的组时，它就变成了实例标准化。



# Group Normalization

**Group Normalization** divides the channels into groups and computes within each group the mean and variance for normalization. 将通道分成组，然后计算每组的均值和方差。

The computation of group normalization is independent of batch sizes, and its accuracy is stable in a wide range of batch sizes. 组齐次化的计算量依赖于batch大小，在较大的batch下，其精度相对稳定。



```
def GroupNorm(x, gamma, beta, G, eps=1e-5):  
    # x: input features with shape [N,C,H,W]  
    # gamma, beta: scale and offset, with shape [1,C,1,1]  
    # G: number of groups for GN  
  
    N, C, H, W = x.shape  
    x = tf.reshape(x, [N, G, C // G, H, W])  
  
    mean, var = tf.nn.moments(x, [2, 3, 4], keep_dims=True)  
    x = (x - mean) / tf.sqrt(var + eps)  
  
    x = tf.reshape(x, [N, C, H, W])  
  
    return x * gamma + beta
```

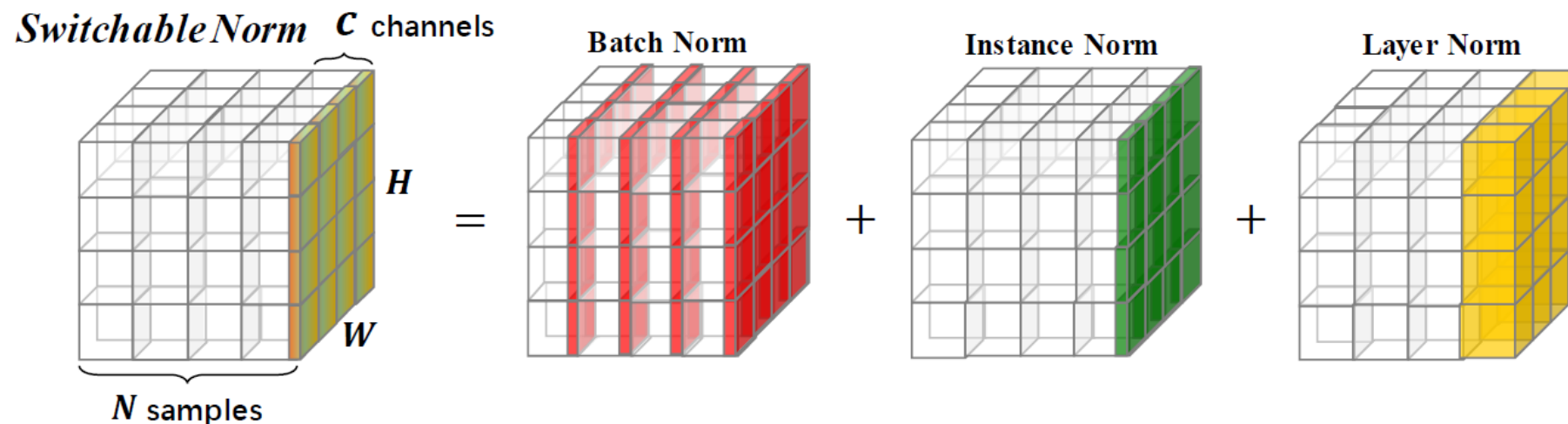


# Switchable Normalization



**Switchable Normalization** is a normalization technique that is able to **learn different normalization operations for different normalization layers** in a deep neural network in an end-to-end manner.

组合：将各种齐次化方式组合到一起，活灵活现。



# Q&A

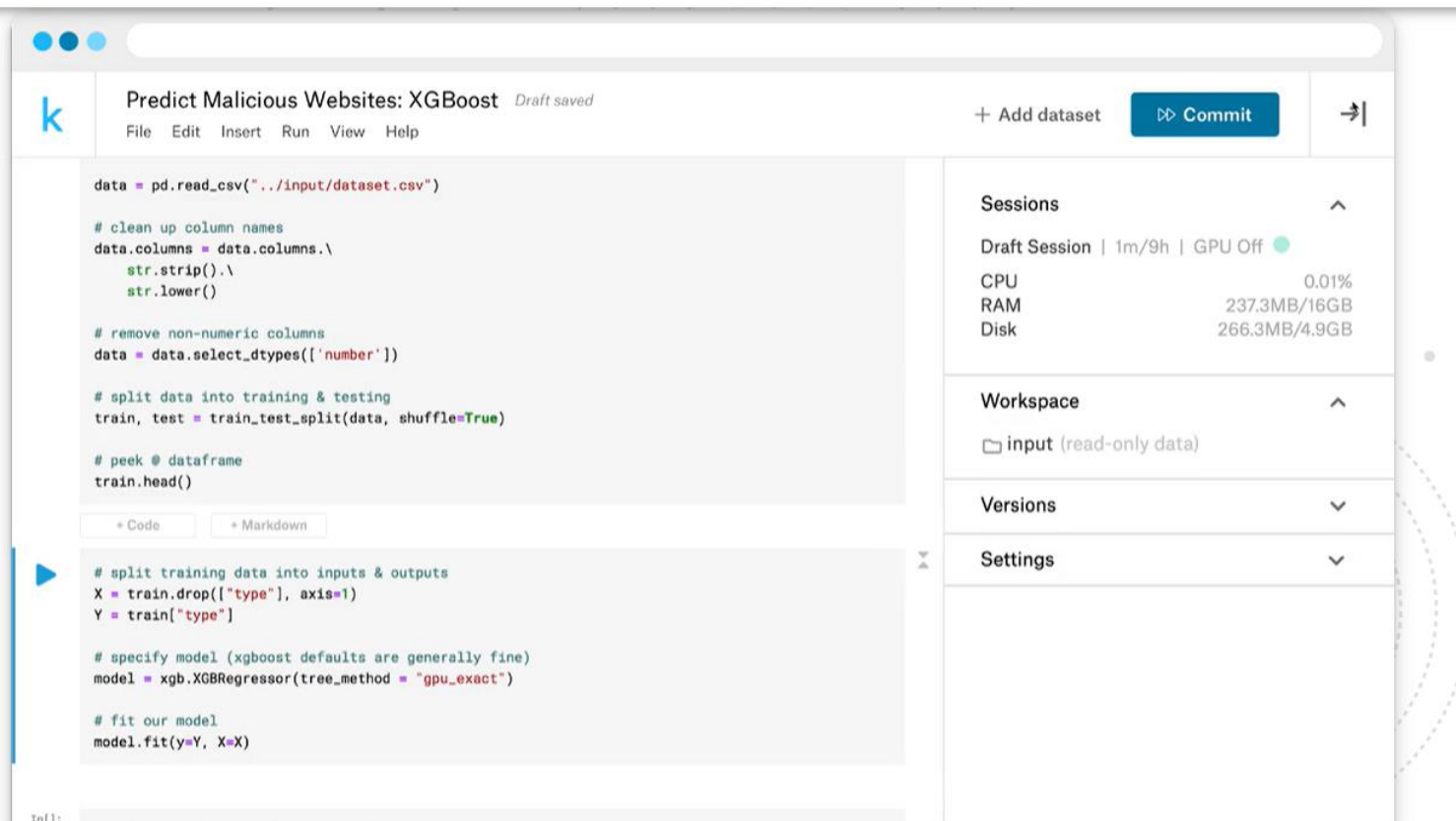


Spring 2023



## Start with more than a blinking cursor

Kaggle offers a no-setup, customizable, Jupyter Notebooks environment. Access GPUs at no cost to you and a huge repository of community published data & code.

[REGISTER WITH GOOGLE](#)[Register with Email](#)

The screenshot shows a Kaggle Notebook interface. The title is "Predict Malicious Websites: XGBoost" with a "Draft saved" status. The notebook is in a Jupyter environment with a menu bar (File, Edit, Insert, Run, View, Help). The code is written in Python and includes the following steps:

```
data = pd.read_csv("../input/dataset.csv")

# clean up column names
data.columns = data.columns.\
    str.strip().\
    str.lower()

# remove non-numeric columns
data = data.select_dtypes(['number'])

# split data into training & testing
train, test = train_test_split(data, shuffle=True)

# peek @ dataframe
train.head()
```

Below the code, there are tabs for "+ Code" and "+ Markdown". The code continues with:

```
# split training data into inputs & outputs
X = train.drop(["type"], axis=1)
Y = train["type"]

# specify model (xgboost defaults are generally fine)
model = xgb.XGBRegressor(tree_method = "gpu_exact")

# fit our model
model.fit(y=Y, X=X)
```

On the right side of the notebook, there is a sidebar with the following sections:

- Sessions**: Draft Session | 1m/9h | GPU Off (0.01% CPU, 237.3MB/16GB RAM, 266.3MB/4.9GB Disk)
- Workspace**: input (read-only data)
- Versions**: (dropdown arrow)
- Settings**: (dropdown arrow)