# 数学基础

## <https://arxiv.org/pdf/1610.04161.pdf>

WHY DEEP NEURAL NETWORKS FOR FUNCTION APPROXIMATION?

## 

## 概率密度函数

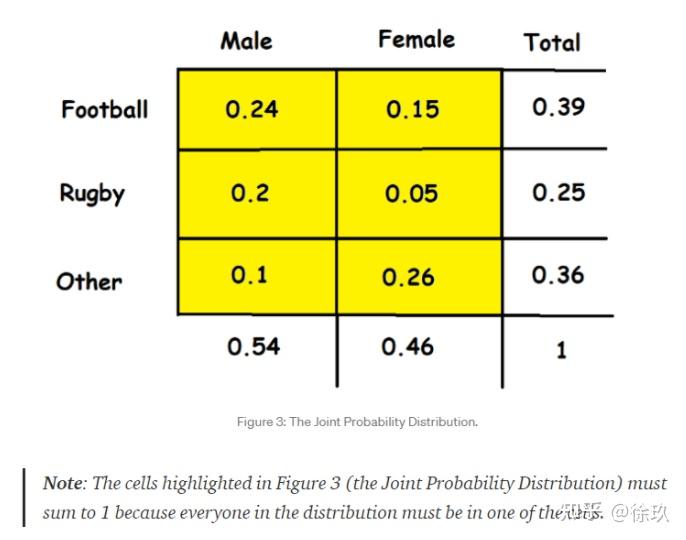
https://zhuanlan.zhihu.com/p/48140593

## 边缘概率

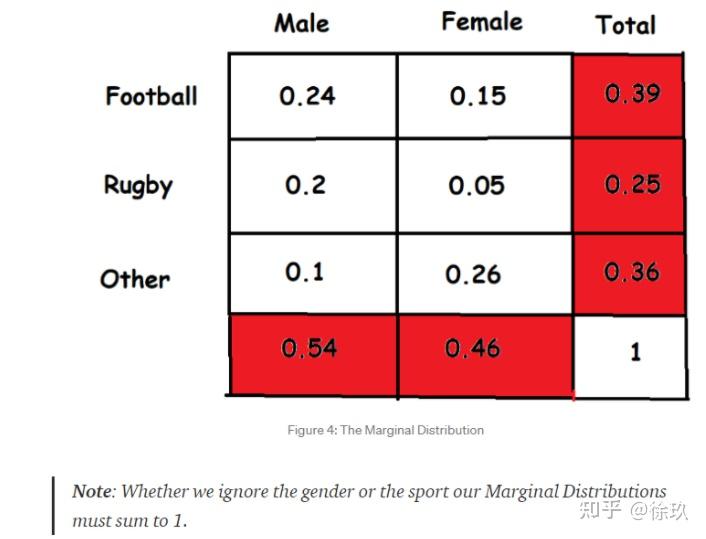
联合概率 Joint Probability：

两个事件同时发生的概率：P(A & B)

下图**标黄部分是联合概率**，联合概率是对称的，即 P(Male and Football) = P(Football and Male)



边缘概率 Marginal Probability：对多类事件存在情况（联合分布问题），只关注其中一个事件的概率： P(A & B)。下图标红色部分就是边缘概率



<https://zhuanlan.zhihu.com/p/53005534>

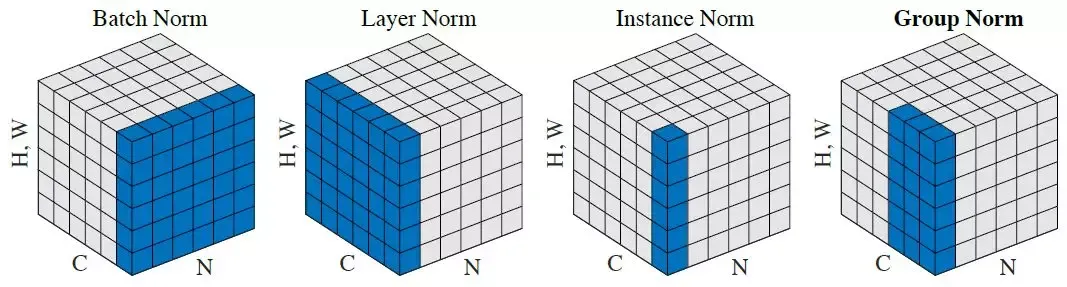
https://zhuanlan.zhihu.com/p/519209543

## 均值为0，方差为1

https://juejin.cn/s/%E6%A0%87%E5%87%86%E6%AD%A3%E6%80%81%E5%88%86%E5%B8%83%E7%9A%84%E5%9D%87%E5%80%BC%E4%B8%BA0%2C%E6%96%B9%E5%B7%AE%E4%B8%BA1

均值和方差会影响分布吗？

在正态分布中，均值和方差是两个非常重要的参数。 事实上，均值和方差在正态分布中有着密切的关系。 具体来说，正态分布的均值决定了分布的中心位置，而方差则决定了分布的分散程度。 方差越大，分布的形态就越平坦，越接近于均匀分布；方差越小，分布的形态就越陡峭，越接近于一个尖峰。



NHW CHW HW

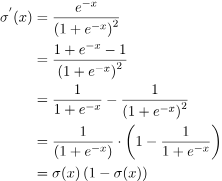
<https://medium.com/syncedreview/facebook-ai-proposes-group-normalization-alternative-to-batch-normalization-fb0699bffae7>

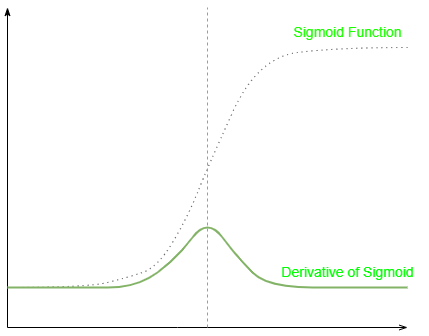
## 梯度消失和梯度爆炸exploding or vanishing gradients,

梯度消失：

Vanishing is when as backpropagation occurs, the gradients normally get smaller and smaller, gradually approaching zero. This leaves the weights of the initial or lower layers unchanged, causing the Gradient Descent to never converge to the optimum.

Sigmoid：



## Linear-regression

<https://www.zhihu.com/column/c_1369429467543306240>

## CNN vs FC

The main difference is that CNN uses convolution operation to process the data, which has some benefits for working with images. In that way, CNNs reduce the number of parameters in the network. Also, convolution layers consider the context in the local neighborhood of the input data and construct features from that neighborhood.

## RNN vs CNN vs Linear

* CNNs are commonly used to solve problems involving spatial data, such as images. RNNs are better suited to analyzing temporal and sequential data, such as text or videos.
* CNNs and RNNs have different architectures. CNNs are feedforward neural networks that use filters and pooling layers, whereas RNNs feed results back into the network.
* In CNNs, the size of the input and the resulting output are fixed. A CNN receives images of fixed size and outputs a predicted class label for each image along with a confidence level. In RNNs, the size of the input and the resulting output can vary.
* Common use cases for CNNs include [facial recognition](https://www.techtarget.com/searchenterpriseai/definition/facial-recognition), medical analysis and image classification. Common use cases for RNNs include machine translation, [natural language processing](https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP), sentiment analysis and speech analysis.

## Gradient-descent

https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21

用上一个点的值以及梯度来预测下一个点的值。

对于同样的tolerance，learn\_rate越大，速度越快。

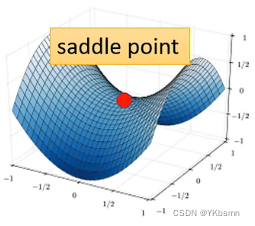
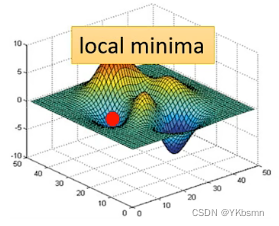
对于同样的learn\_rate, 的tolerance越大，速度越快。

学习的基本方法。通常来说，我们对误差的要求是确定的。所以，可以调整的参数只有learn rate了。那么，怎么调整这个呢？一种增大这个的方法是Batch Normization。

SGD:

<https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31>

https://www.zhihu.com/question/264189719/answer/291167114



## 

## Forward propagation

https://www.youtube.com/watch?v=sNTtUV9yE\_M

## Backprog calculus

https://www.youtube.com/watch?v=tIeHLnjs5U8

<https://www.youtube.com/watch?v=wqPt3qjB6uA>

## 

## Why need deep neural networks

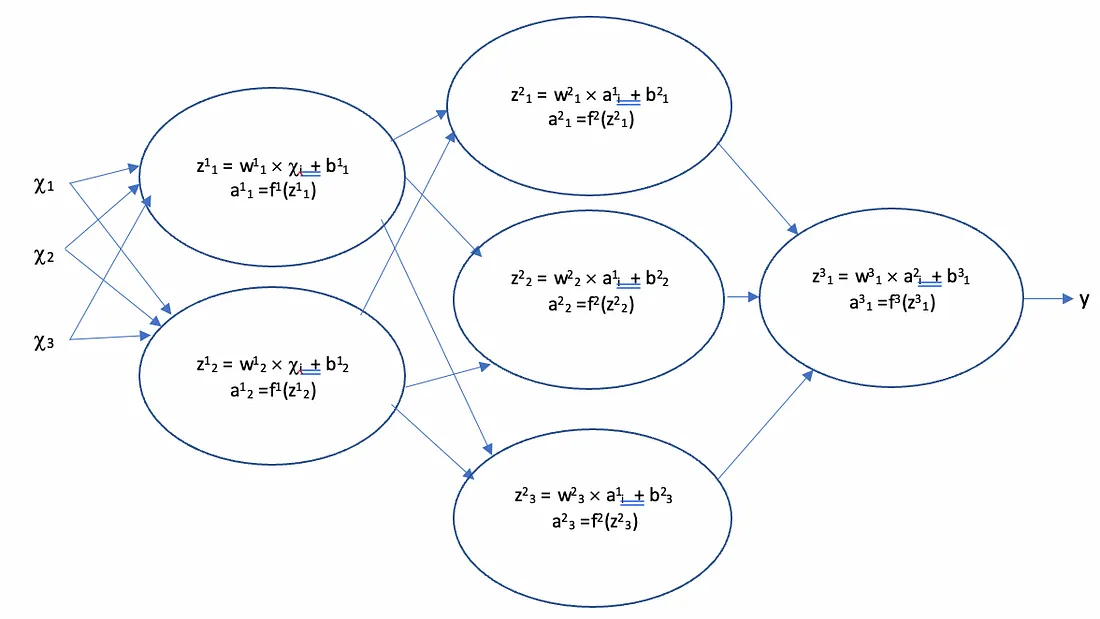
## Why Converge harder when network is deeper

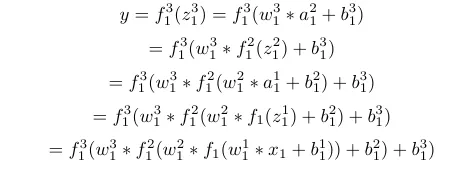
https://medium.com/@dreading/why-are-deeper-neural-networks-harder-to-train-587edf422ac2

## Why Activation function

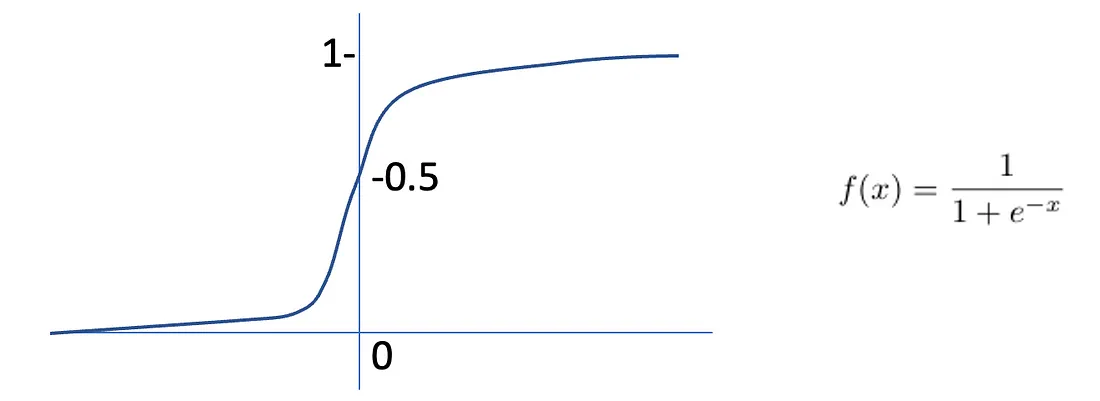
To introduce non-linear

<https://towardsdatascience.com/the-importance-and-reasoning-behind-activation-functions-4dc00e74db41>



Sigmoid：



“This outputs a value between 0 and 1, making it useful for binary classification problems as one can set a threshold “probability” value. Above this value the output can be classed as 1, and below it the output will be 0. **This makes it a popular choice for the final output node of a network**.”

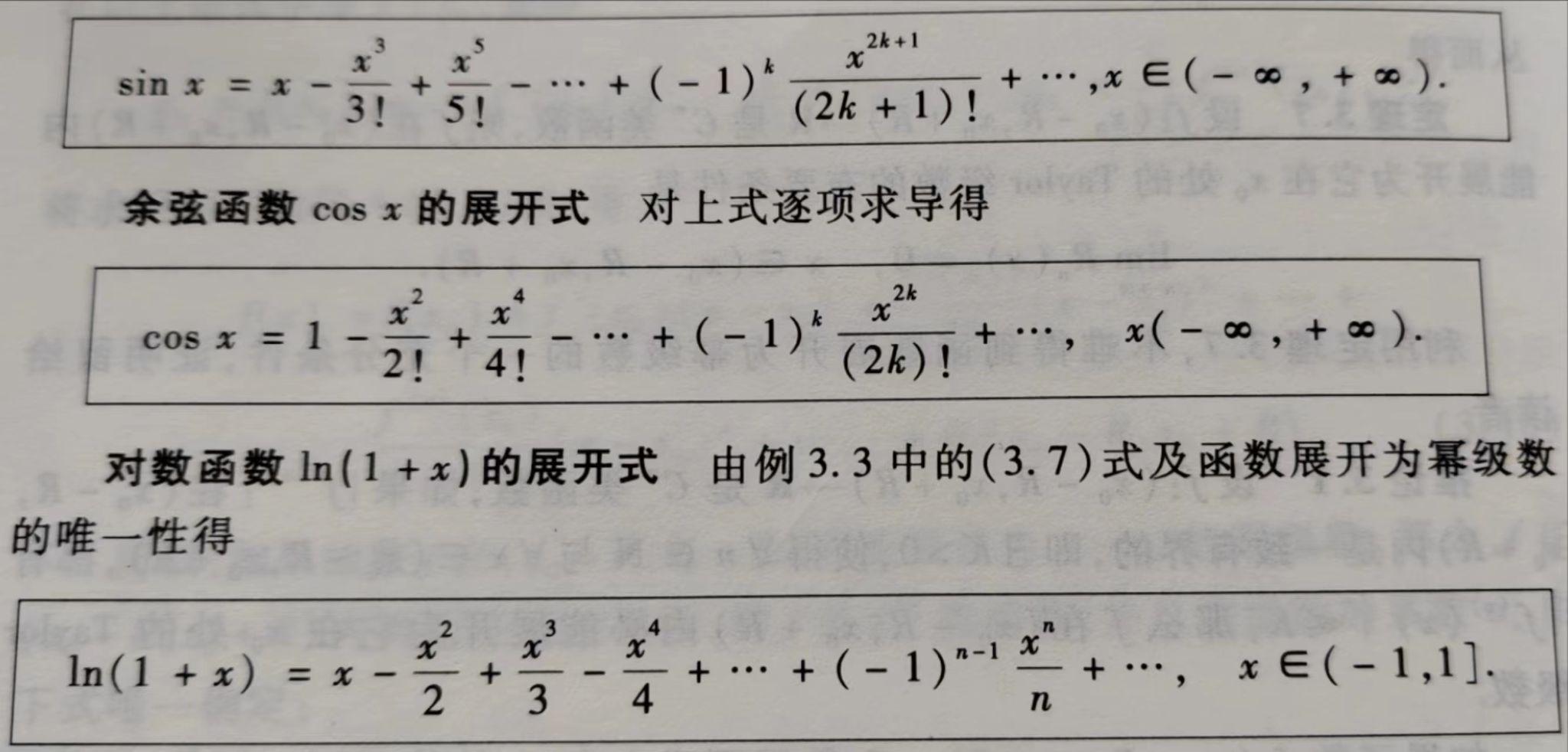
https://hausetutorials.netlify.app/posts/2019-12-01-neural-networks-deriving-the-sigmoid-derivative/#:~:text=The%20derivative%20of%20the%20sigmoid%20function%20%CF%83(x)%20is%20the,1%E2%88%92%CF%83(x).

我理解问题确实是非线性的。这样的网络也确实构成了一个非线性的网络。

第一，非线性问题一定是可以描述的吗？

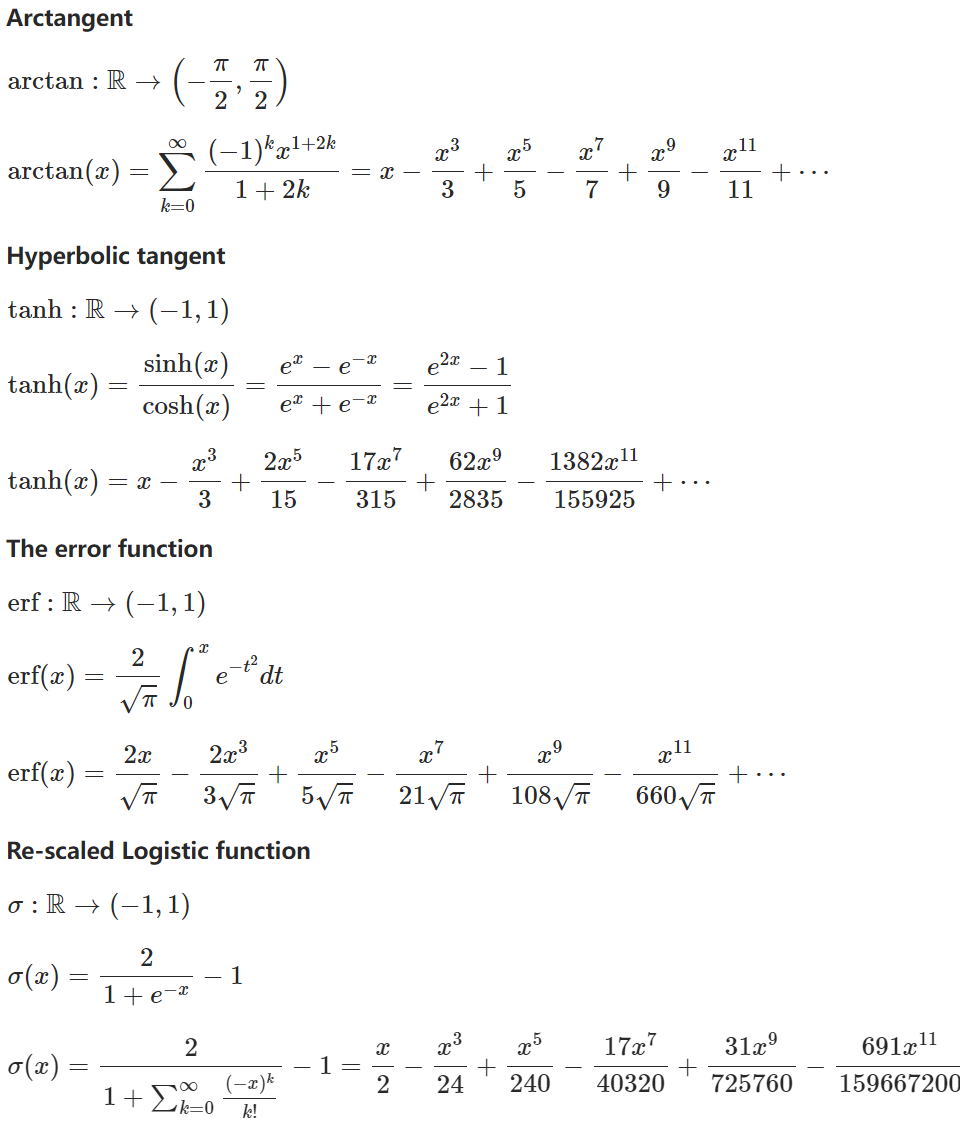
第二，为什么神经网络这样的非线性描述可以构成一个解？

微积分里面有个泰勒级数，满足一定条件的函数都可以展开成泰勒级数。譬如：



我们把左边的函数等价为各种现实的非线性问题，右边等价为网络。虽然左边的问题（CV，NLP等）形态各异，但是右边的问题都是级数（网络）。这是数学的传统。

进一步，如果我们把激活函数展开成级数：



多个原来收敛级数经过加减后，还是收敛的吗？

## Gradient descent methods and linear regression

https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931

# Machine learning terms

## Β-smoothness

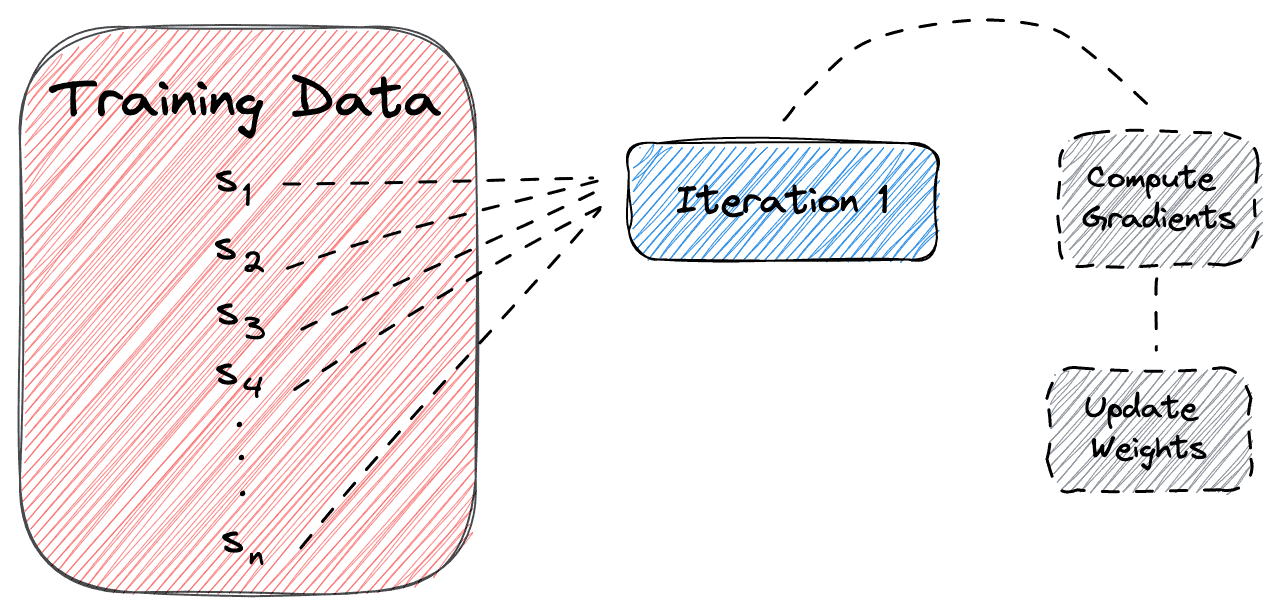
## Backpropagation Algorithm

TBD: https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

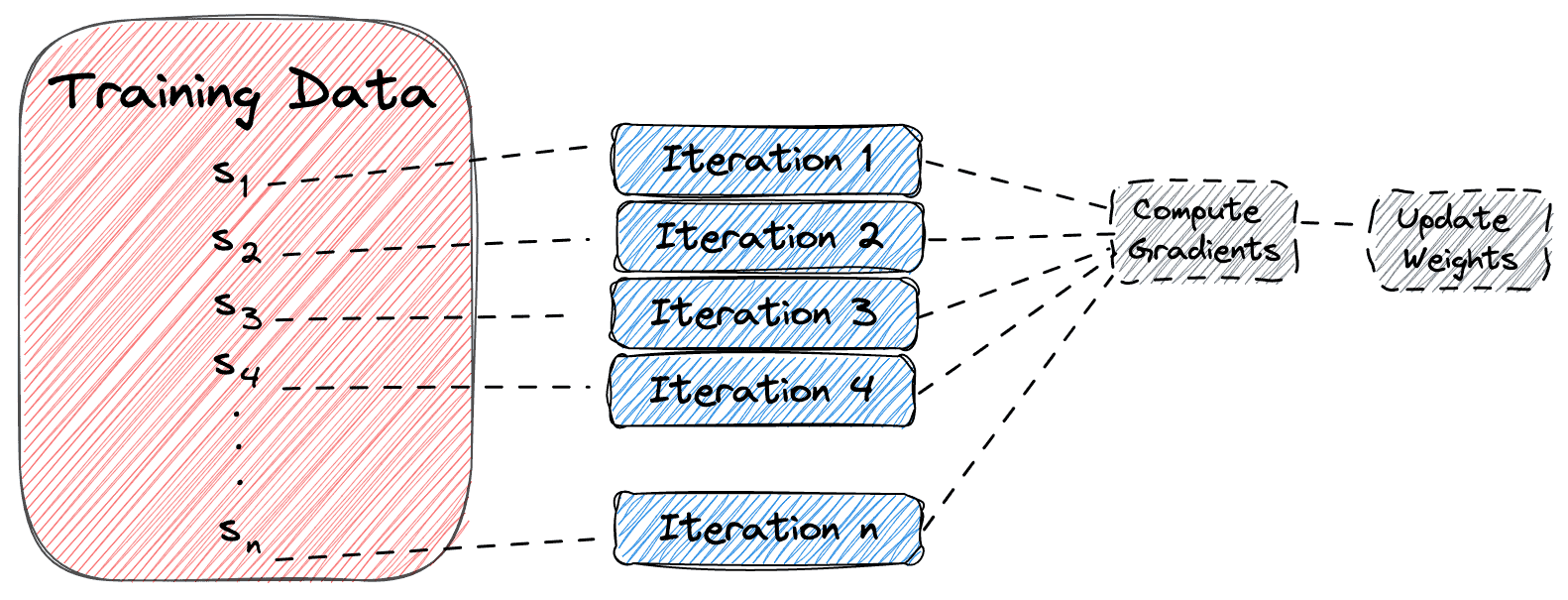
## Batch vs Mini-batch

From: <https://www.baeldung.com/cs/epoch-vs-batch-vs-mini-batch>

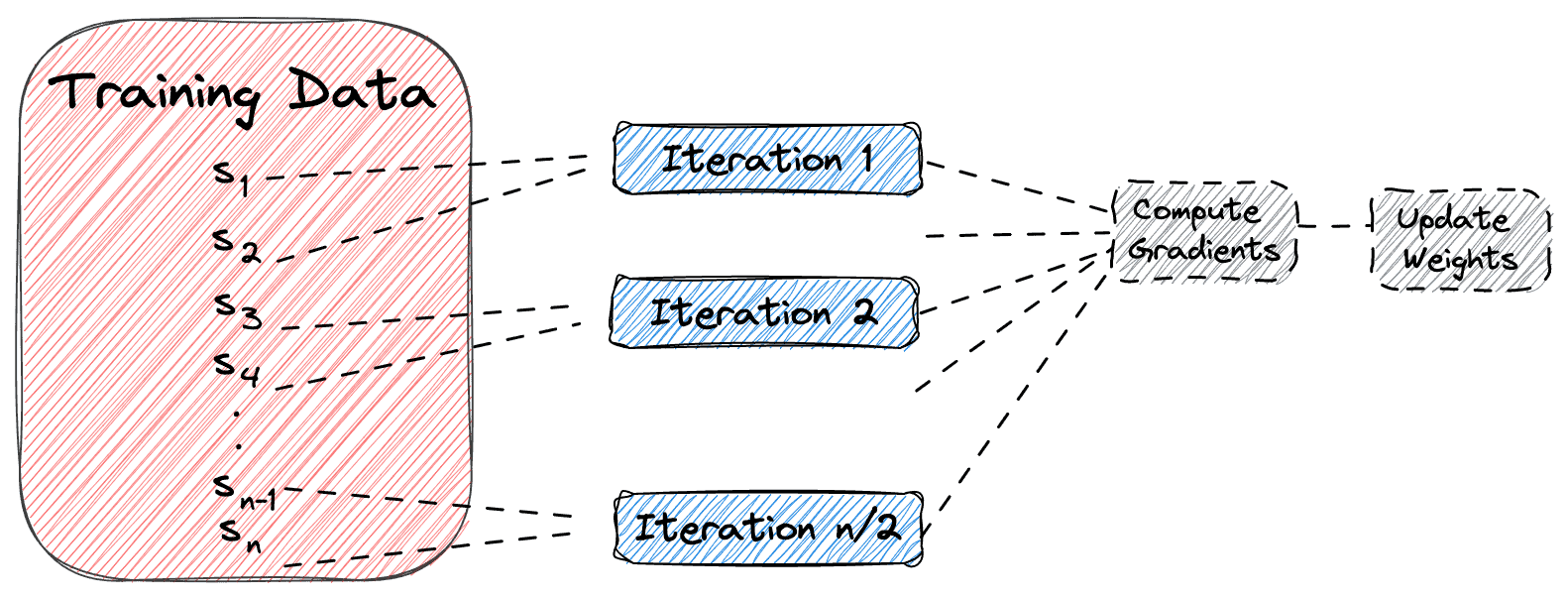
Batch Gradient Descent:



Stochastic Gradient Descent:

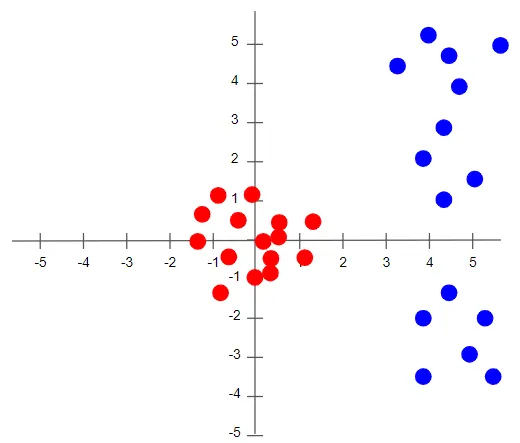
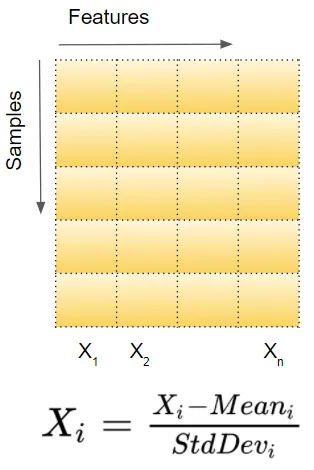


Mini-batch Gradient Descent:



# Normalization

## What is norm



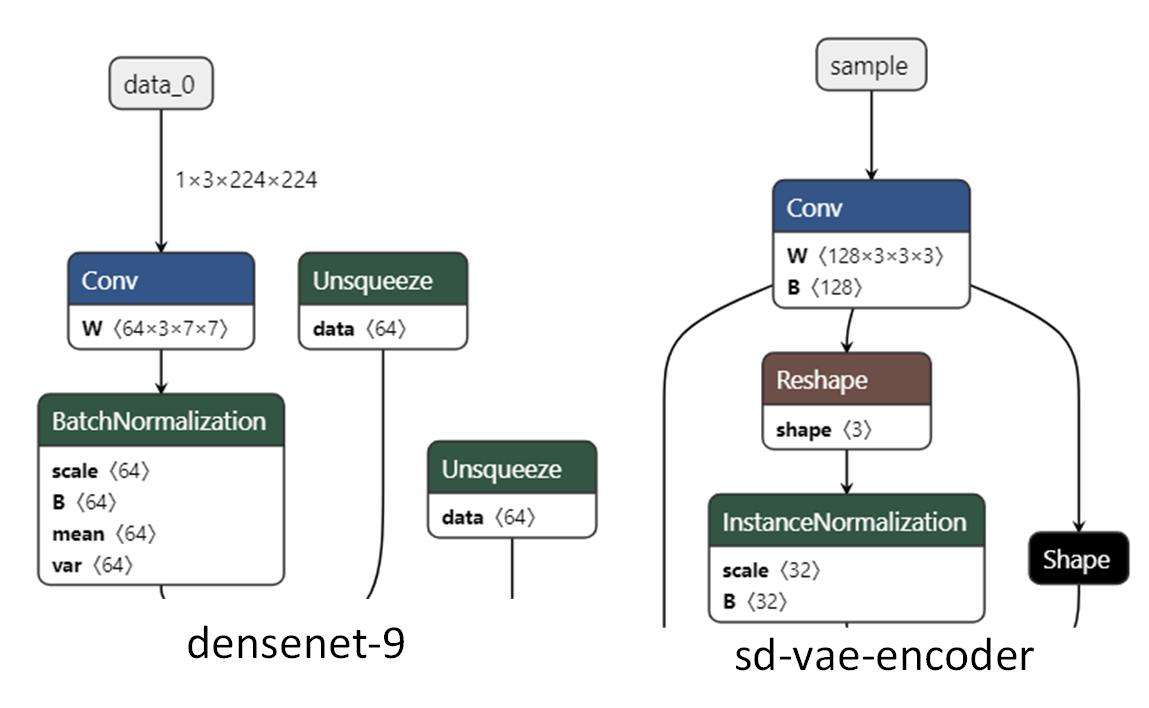
From: https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

## Why Norm

本质原因：

1. **我们要识别的东西，不是只有一个特征**。而且，我们需要在同一个网络里面，实现对同一物体的多个特征的识别。（为什么不可以在多个网络里面，每个网络仅仅识别一个特征？）。每个特征的值的范围相差很大，譬如身高最大是2米左右，体重有100千克多。

我理解模型最开始的输入，应该有Norm。但是我找了几个模型，没有发现对输入进行Normalization。



有个过程和Norm类似，叫Preprocessing，应用在ResNet50里面，譬如ResNet50（[https://deeplizard.com/learn/video/hRKEQhiqIU4）](https://deeplizard.com/learn/video/hRKEQhiqIU4%EF%BC%89) 里面就规定了这样的Preprocessing：

let meanImageNetRGB = {

red: 123.68,

green: 116.779,

blue: 103.939

};

let centeredRGB = {

red: tf.gather(tensor, indices[0], 2)

.sub(tf.scalar(meanImageNetRGB.red))

.reshape([50176]),

green: tf.gather(tensor, indices[1], 2)

.sub(tf.scalar(meanImageNetRGB.green))

.reshape([50176]),

blue: tf.gather(tensor, indices[2], 2)

.sub(tf.scalar(meanImageNetRGB.blue))

.reshape([50176])

};

但是，这个原因不适合解释为什么中间会有Norm。

那么，为什么存在Layer norm？应该是每个feature对应一个norm（譬如BN，IN）？但是存在某些对象，它的多个Feature之间是非常类似的分布，这就意味着多个Feature一起进行Norm就可以了。

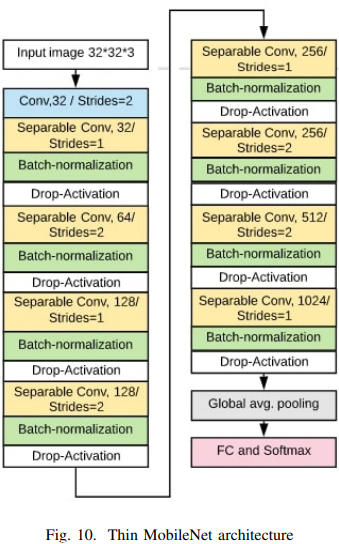
1. **为什么中间层也会有Norm（譬如ResNet50）？**

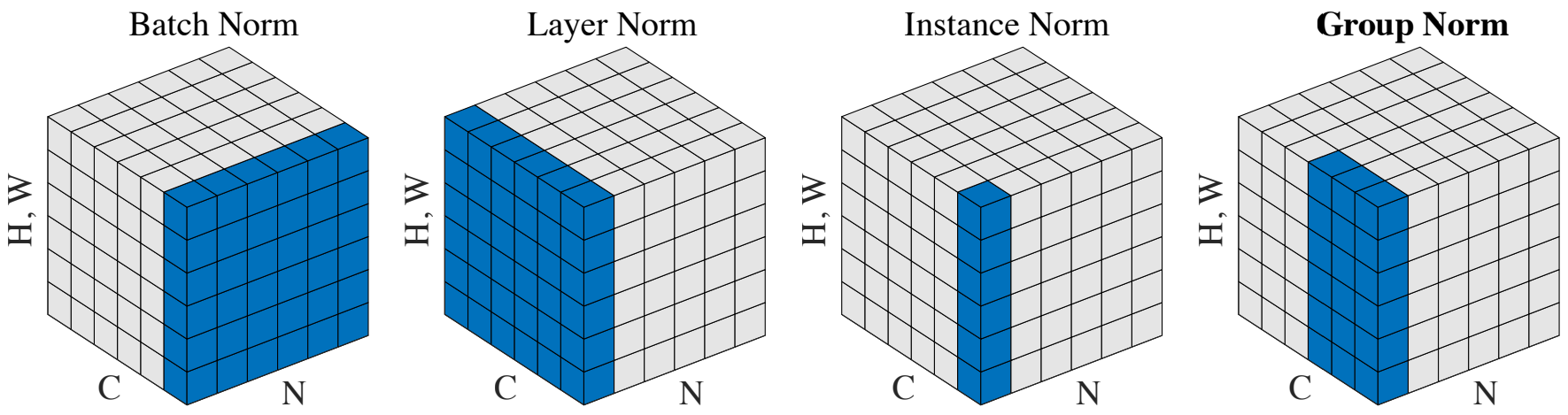
（独立同分布：independent and identically distributed， 所谓同分布就是所有特征具有相同的均值和方差。为什么均值和方差相同分布就相同？Sufficient statistic）

## Why does MobileNet have no Norm and ResNet has BatchNorm?

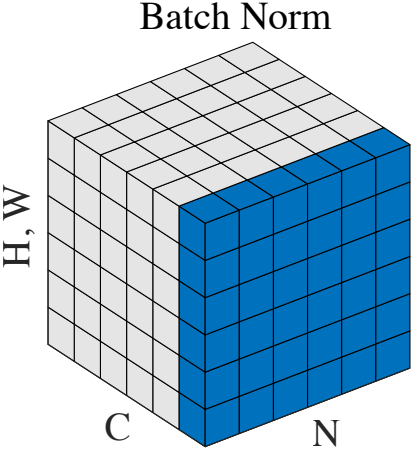
Someone added batchnorm layer in MobileNet dubbed Thin-MobileNet, from

<https://scholarworks.iupui.edu/server/api/core/bitstreams/a7fbc815-0f25-480a-bce1-0cb231238b66/content>





# BN



Batch

Mini-batch

What is batch norm

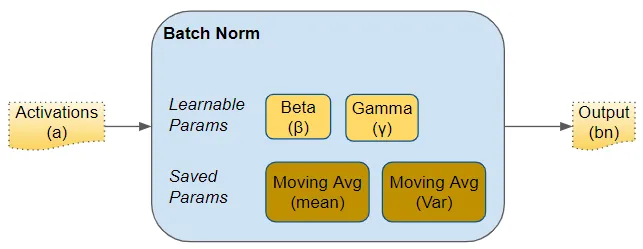
## How BN works

### Training

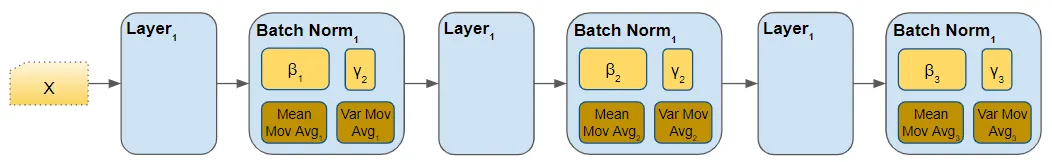
From：https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

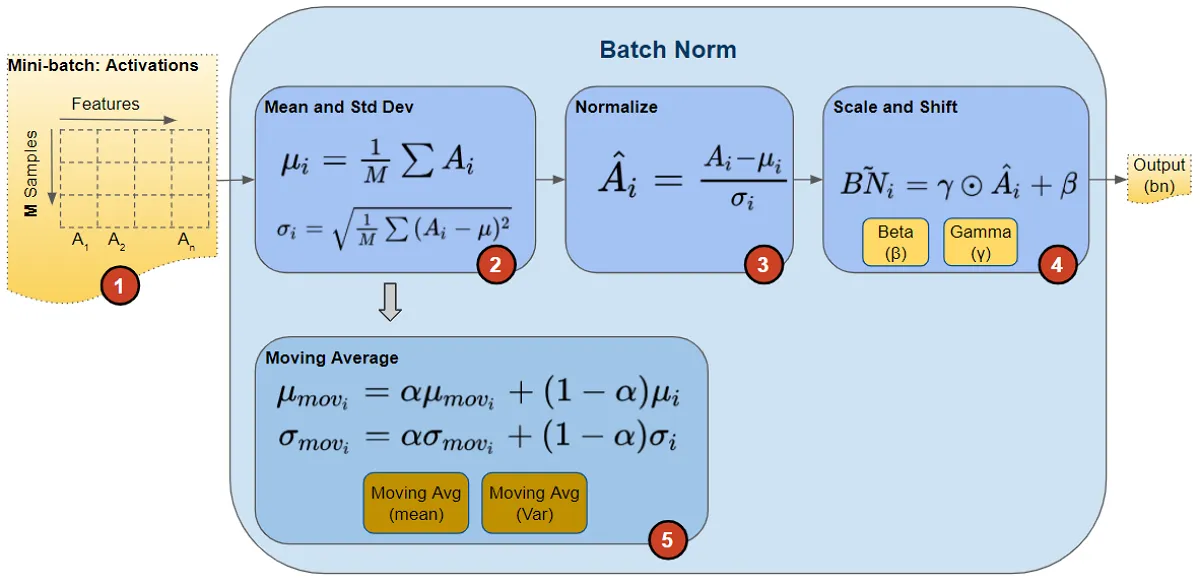
* Two learnable parameters called beta and gamma.
* Two non-learnable parameters (Mean Moving Average and Variance Moving Average) are saved as part of the ‘state’ of the Batch Norm layer.

If single layer:



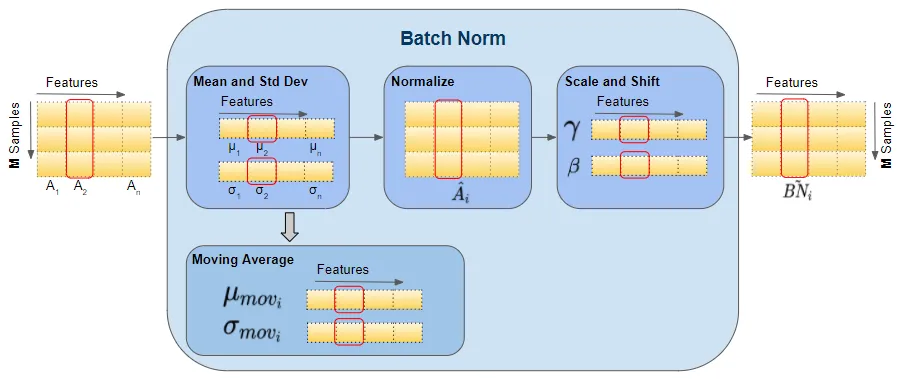
If multi layer:

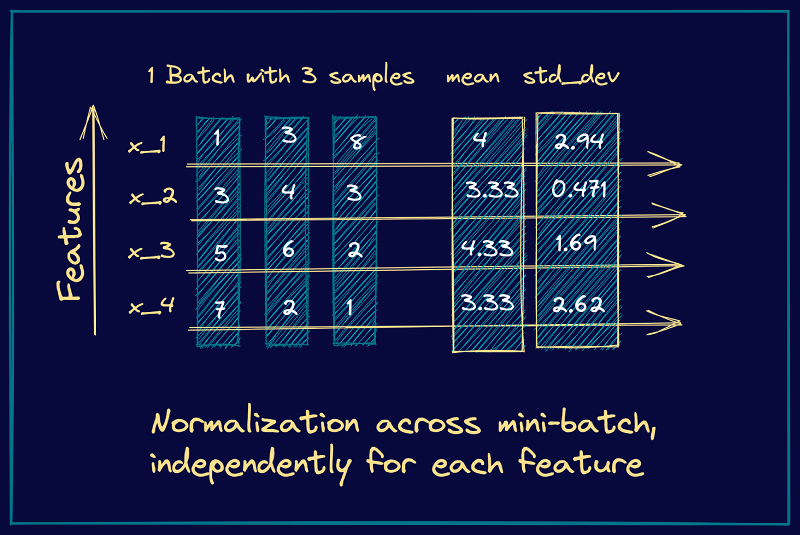




M是mini-batch数据的大小。

这里的均值是似乎把所有Feature求和，但是ORT里面的Inference是Per Channel的。不过作者稍后给出了另一个图：





### Why scale and shift?

注意：输出的数据均值为方差为gamma2，均值为beta。不是均值为0，方差为1的最好处理么？为什么又scale shift回去了？

From：https://zhuanlan.zhihu.com/p/33173246

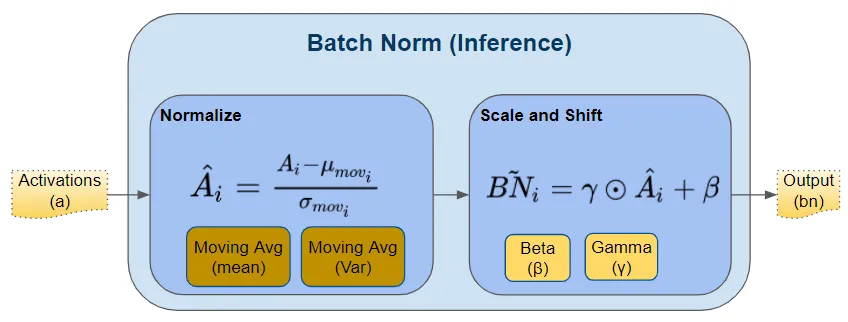
“答案是——为了保证模型的表达能力不因为规范化而下降。

我们可以看到，第一步的变换将输入数据限制到了一个全局统一的确定范围（均值为 0、方差为 1）。下层神经元可能很努力地在学习，但不论其如何变化，其输出的结果在交给上层神经元进行处理之前，将被粗暴地重新调整到这一固定范围。难道底层神经元就在做无用功吗？所以，**为了尊重底层神经网络的学习结果**，我们将规范化后的数据进行再平移和再缩放，使得每个神经元对应的输入范围是针对该神经元量身定制的一个确定范围（均值为beta，方差为gamma2）。rescale 和 reshift 的参数都是可学习的，这就使得 Normalization 层可以学习如何去尊重底层的学习结果。”

From： https://www.sciencedirect.com/science/article/pii/S0020025523005200#:~:text=2.1.,-1.&text=According%20to%20the%20original%20BatchNorm,process%20of%20zeroing%20the%20mean

According to the original BatchNorm paper, **the purpose of the scale and shift parameters γ and β is to restore information** that may be lost through the normalization process of zeroing the mean. This scale and shift serves to re-parametrize the activations in a way that allows for the same family of functions to be expressed, but with trainable parameters that may make it easier for the model to learn via [gradient descent](https://www.sciencedirect.com/topics/engineering/gradient-descent) [[6]](https://www.sciencedirect.com/science/article/pii/S0020025523005200#br0060).

### Inference



@compute @workgroup\_size(64, 1, 1)

fn main(@builtin(global\_invocation\_id) global\_id : vec3<u32>,

@builtin(workgroup\_id) workgroup\_id : vec3<u32>,

@builtin(local\_invocation\_id) local\_id : vec3<u32>) {

let global\_idx = global\_id.x; let local\_idx = local\_id.x;

if (global\_idx >= uniforms.outputSize) { return; }

var outputIndices = global\_idx \* 4;

let cOffset = 0u;

let scale = scale[cOffset];// scale&bias is per channel!

let bias = bias[cOffset];

let inputMean = inputMean[cOffset];

let inputVar = inputVar[cOffset];

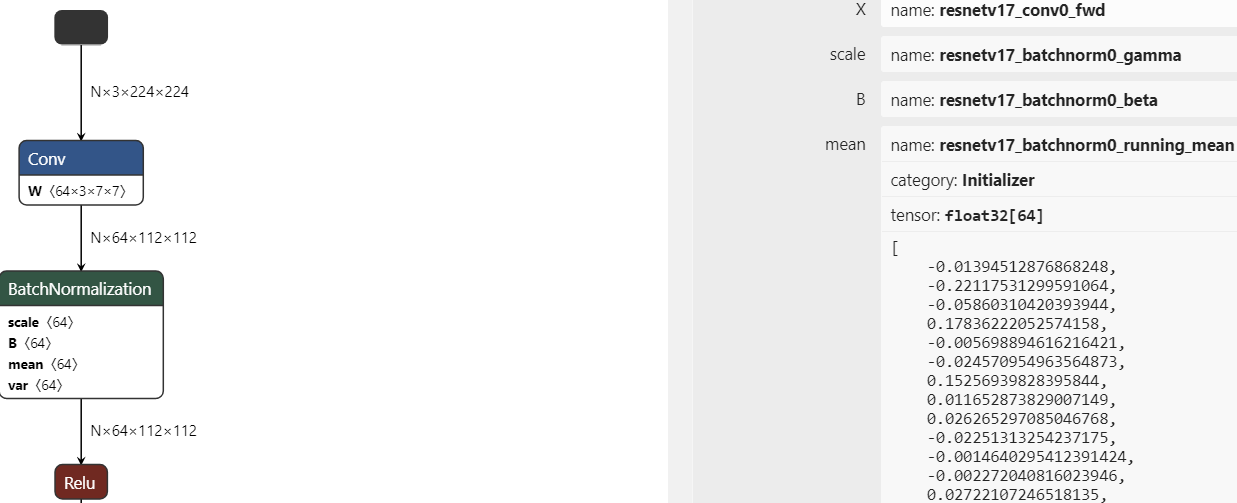
let x = x[global\_idx];

let value = (x - inputMean) \* inverseSqrt(inputVar + epsilon) \* scale + bias;

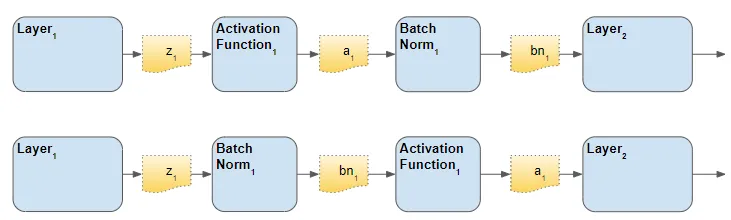
y[global\_idx]=value;

}

ResNet50v1-12:



### Where to place BN



### Math-TBD

https://arxiv.org/pdf/1502.03167.pdf

<https://medium.com/@sofeikov/batch-normalisation-formulas-derivation-253df5b75220>

### Why BN

About： Internal Covariate Shift， From: https://www.zhihu.com/question/38102762/answer/85238569

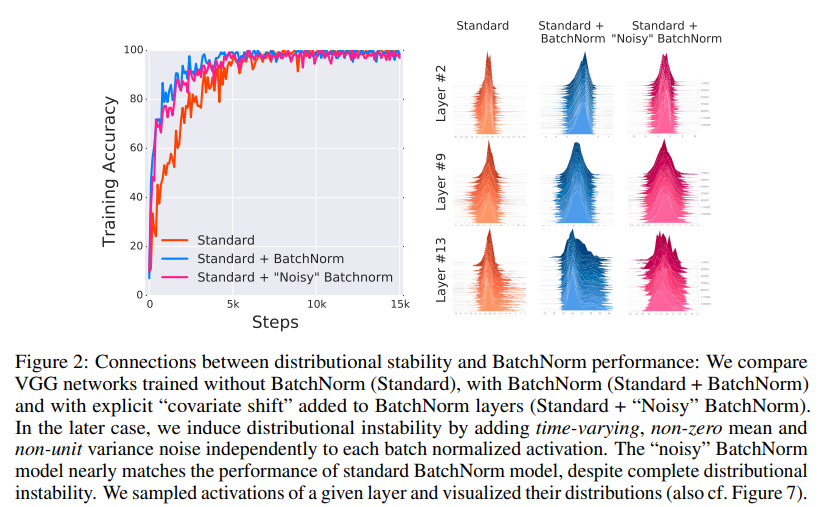
机器学习中的一个经典假设：“源空间（source domain）和目标空间（target domain）的数据分布（distribution）是一致的”。如果不一致，那么就出现了新的机器学习问题，如，transfer learning/domain adaptation等。

而covariate shift就是分布不一致假设之下的一个分支问题，它是指源空间和目标空间的条件概率是一致的，但是其边缘概率不同。的确，对于神经网络的各层输出，由于它们经过了层内操作作用，其分布显然与各层对应的输入信号分布不同，而且差异会随着网络深度增大而增大，可是它们所能“指示”的样本标记（label）仍然是不变的，这便符合了covariate shift的定义。由于是对层间信号的分析，也即是“internal”的来由。”

相反的观点：How Does Batch Normalization Help Optimization?

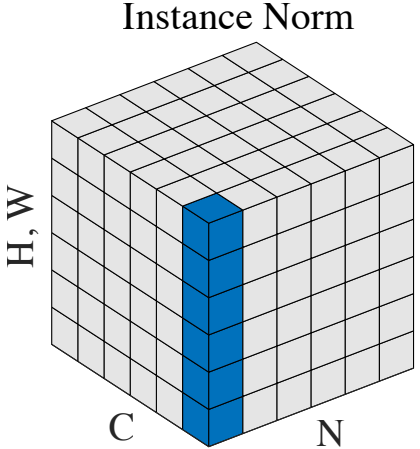
<https://proceedings.neurips.cc/paper_files/paper/2018/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf>:

“The popular belief is that this effectiveness stems from controlling the change of the layers’ input distributions during training to reduce the so-called **“internal covariate shift”**. In this work, we demonstrate that such distributional stability of layer inputs has little to do with the success of BatchNorm. Instead, we uncover a more fundamental impact of BatchNorm on the training process: **it makes the optimization landscape significantly smoother**. This smoothness induces a more predictive and stable behavior of the gradients, allowing for faster training. ”



What is smoothness?

# IN



数据的范围不同。数据的范围不同决定了均值和方差不同。

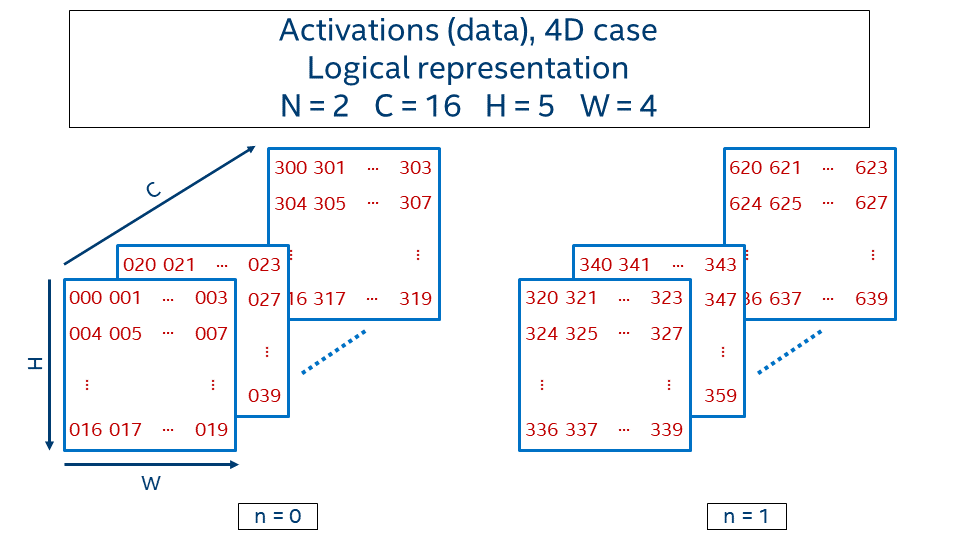
在Inference的实现上，均值和方差是对输入样本求得的。

## 原始实现：

<https://github.com/microsoft/onnxruntime/commit/c3f04251c74c296da982caf683a16e06af2722d4>

<https://github.com/microsoft/onnxruntime/commit/c3f04251c74c296da982caf683a16e06af2722d4#diff-e0ef8b5f7230b59783e698fd241b6487ccc4f181f785e8da27e0ea5656ca669d>

对于图片类型的输入，instance-norm的输入是N x C x H x W。NCHW的布局是这样的：



也就是说：第一个batch：第一个feature的hw，第二个feature的hw，，，第二个batch。

value**(**n**,** c**,** h**,** w**)** **=** n **\*** CHW **+** c **\*** HW **+** h **\*** W **+** w

我们先看GPU Compute的配置情况。假设有64个Channel，那么对应一个WG，每个WG有64线程。这个时候，总的输出是：

输出平均值：64个；

输出output value：64xHxW个。

每个线程的输出是：

输出平均值：1个；

输出output value：HxW个。

const xShape = inputs[0].dims;

const scale = inputs[1];

const bias = inputs[2];

const outputShape = xShape;

const outputSize = ShapeUtil.size(outputShape);

const axis = 2;

const normCount = ShapeUtil.sizeToDimension(xShape, axis); //[0,2)=>0，1

const normSize = ShapeUtil.sizeFromDimension(xShape, axis);//[2,4)=>2，3

const C = xShape[1];

instance-norm本质上是对hw维度进行norm，所以这里的normSize是xShape[2]\*xShape[3]。

最初版本里面，一个normCount/64 个work group。每个workgroup有64个线程。在batch等于1的时候，normCount就等于channel。

记住这个情况后，我们来看寻址方式。

| Channel | 0 | | | | | | 1 | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | HxW | | | | | | HxW | | | | | |
| DataId | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| GlobalID | 0 | | | | | | 1 | | | | | |

当batch=1时，GlobalID其实就是Channnel。

所以offset = global\_idx \* normSize对应的就是每个线程处理的数据的起点，长度是normSize（HxW）。

let offset = global\_idx \* normSize;

if (offset + normSize >= ${outputSize}) { return; }

var mean: ${dataType} = 0;

for (var h: u32 = 0u; h < normSize; h++) {

mean = mean + x[h + offset];

}

mean = mean / normSizeTyped;

这里求出的平均值，就是每个样本，也就是HxW对应数据的平均值。类似的方差也是这样求出来的。

前面提到了global\_idx在batch是1的时候，就等于channel。当batch不等于1 的时候，global\_idx 其实是batch和channel的乘积。所以：global\_idx % C得到的其实是channel的ID，而global\_idx / C向上取整得到的是batch id。所以有了下面的代码：

let invStdDev = 1 / sqrt(squaredNorm / normSizeTyped + epsilon);

let channelScale = invStdDev \* scale[global\_idx % C];

let channelShift = bias[global\_idx % C] - mean \* channelScale;

要注意：scale/bias和均值不一样，他们是针对该channel的所有样本得到的。

## Shared memory：

<https://github.com/microsoft/onnxruntime/pull/17491/files#diff-e0ef8b5f7230b59783e698fd241b6487ccc4f181f785e8da27e0ea5656ca669dR59>

Compute mean：

<https://github.com/microsoft/onnxruntime/commit/22947109f2747097e79bf253c4555e7e9984b4bd#diff-e0ef8b5f7230b59783e698fd241b6487ccc4f181f785e8da27e0ea5656ca669d>

NCHW layout：

<https://github.com/microsoft/onnxruntime/pull/18123>

BN vs IN:

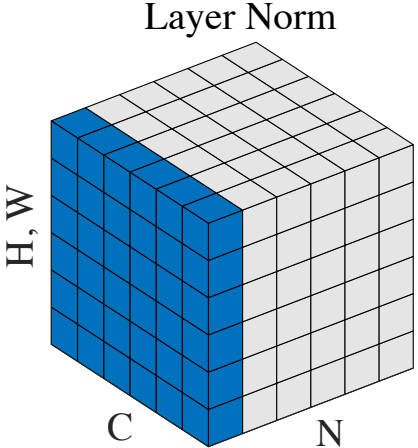
1. IN: single training sample, BN: whole mini-batch of samples.

2. BN is dependent on the batch size, during inference, one example => BN has different training and inference, average and variance are from training; IN has the same training and inference, and needs to calculate mean and variance during inference.

# LN

Batch normalization normalizes each feature independently across the mini-batch.

Layer normalization normalizes each of the inputs in the batch independently across all features-- **This requires that different features should be similar.**



layer normalization vs instance normalization: