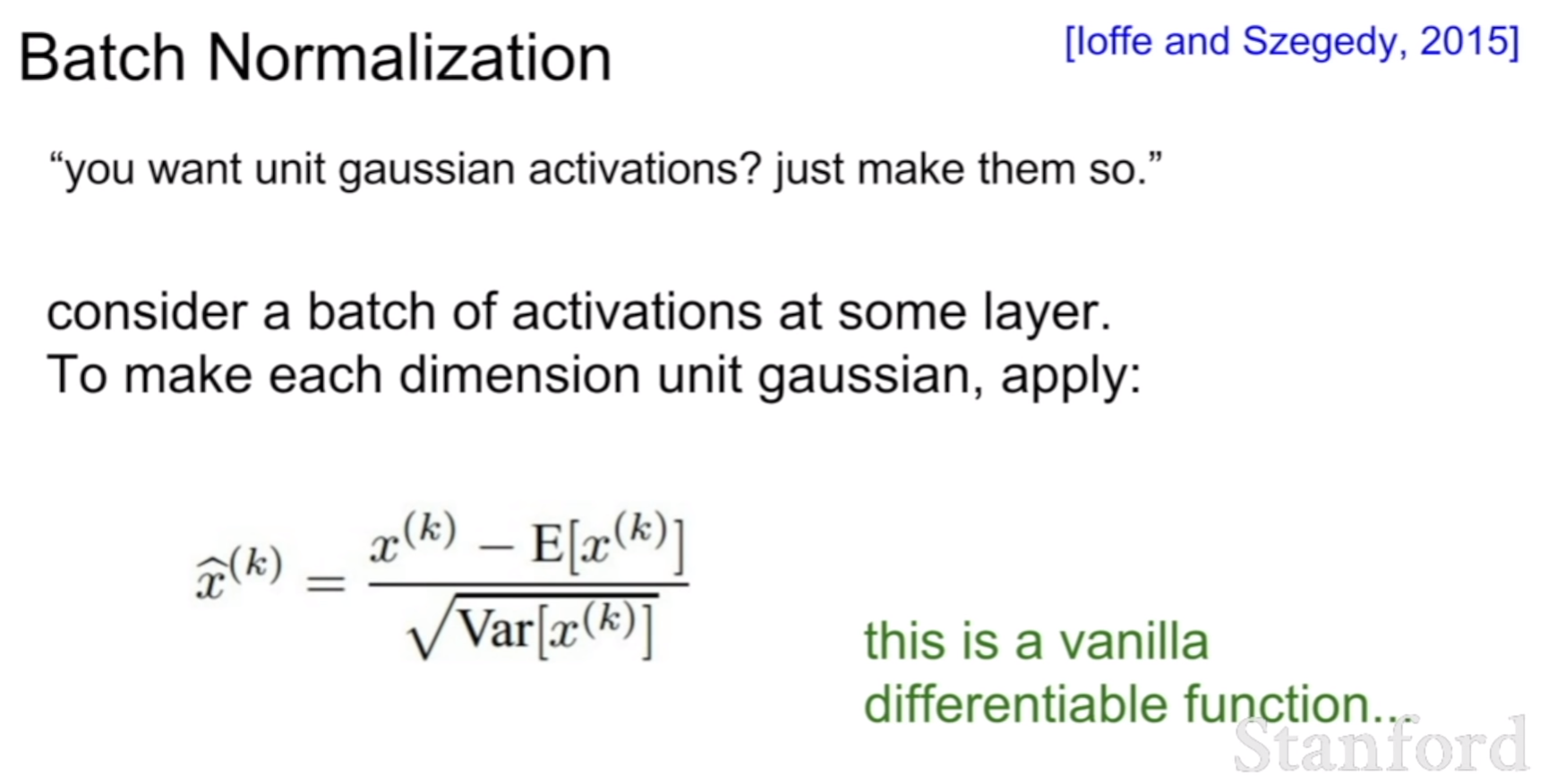
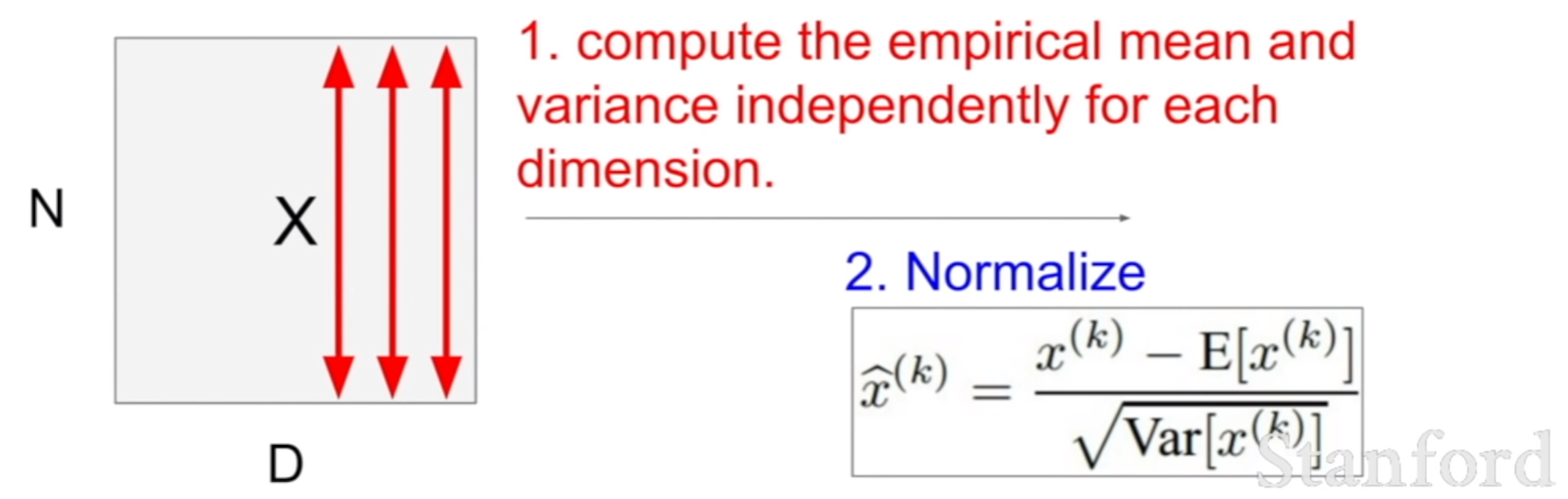
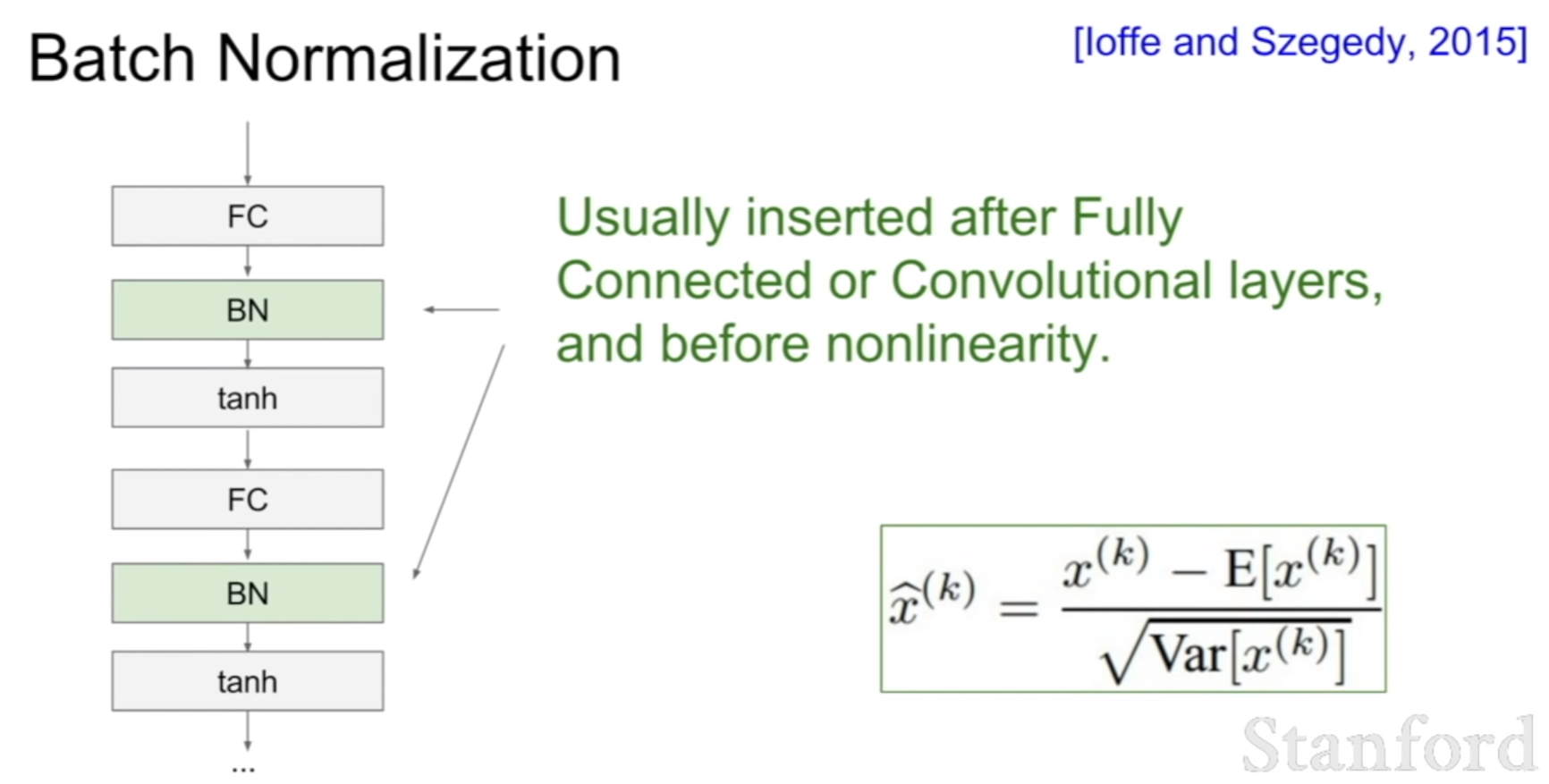
应用Batch Normalization的动机：希望保持数据在送入激活函数前仍然是单元高斯分布。



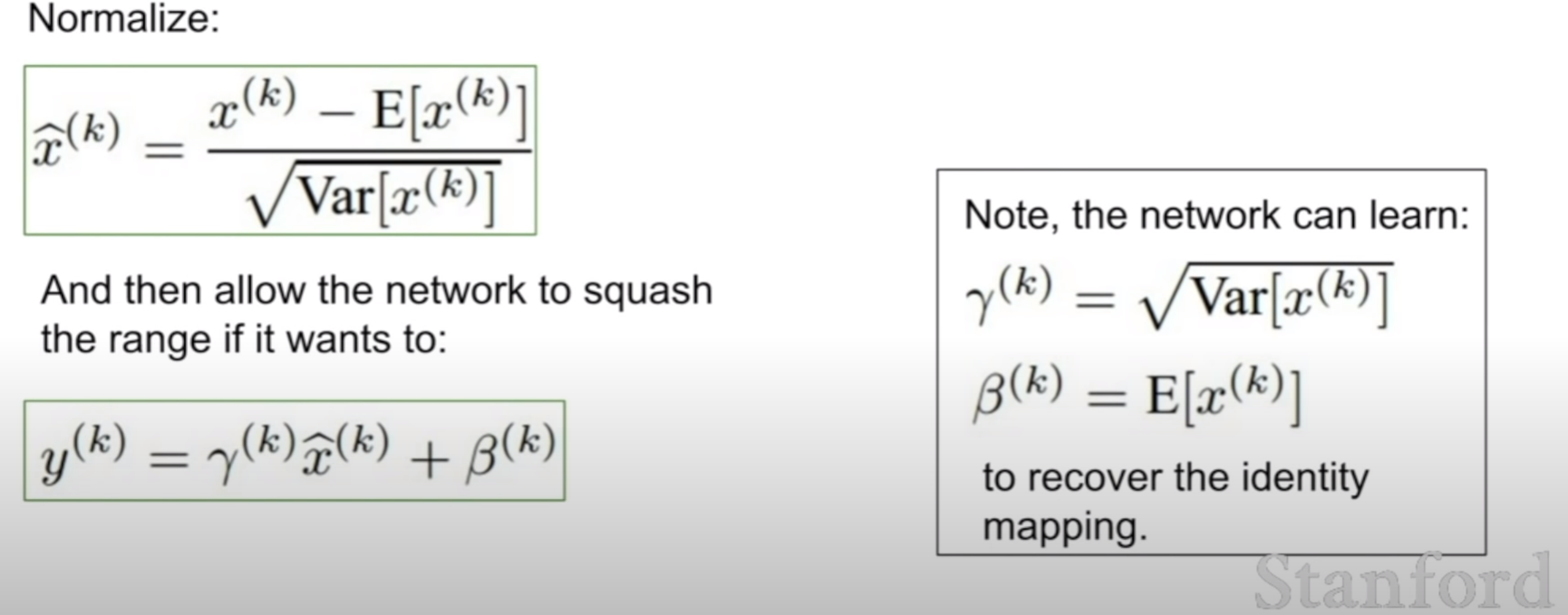


**BN实现**：将当前batch的数据减去这个batch数据的期望（mean），再除以batch数据的方差。**注**：这个normalization需要在不同dimension都进行一样的操作。也就是说，对上图中的每一列都求mean，size为1xD。



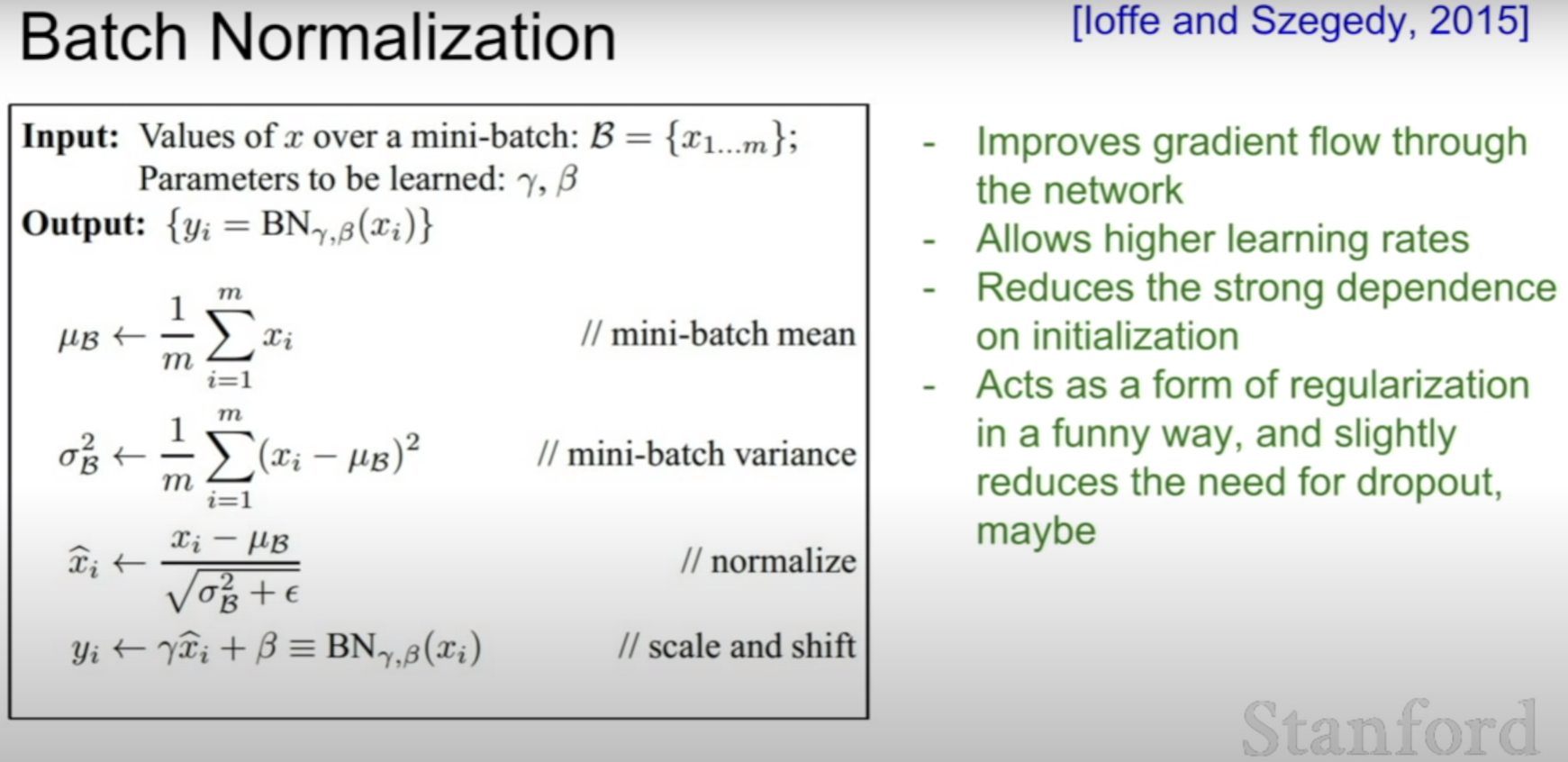
BN通常插入在全连接层或者卷积层之后，在激活层之前。**为什么要在全连接层或者卷积层之后**？因为在前向传播过程中，数据是不断对权重进行累乘的，所以这scaling effect会破坏原来数据的高斯分布形态。

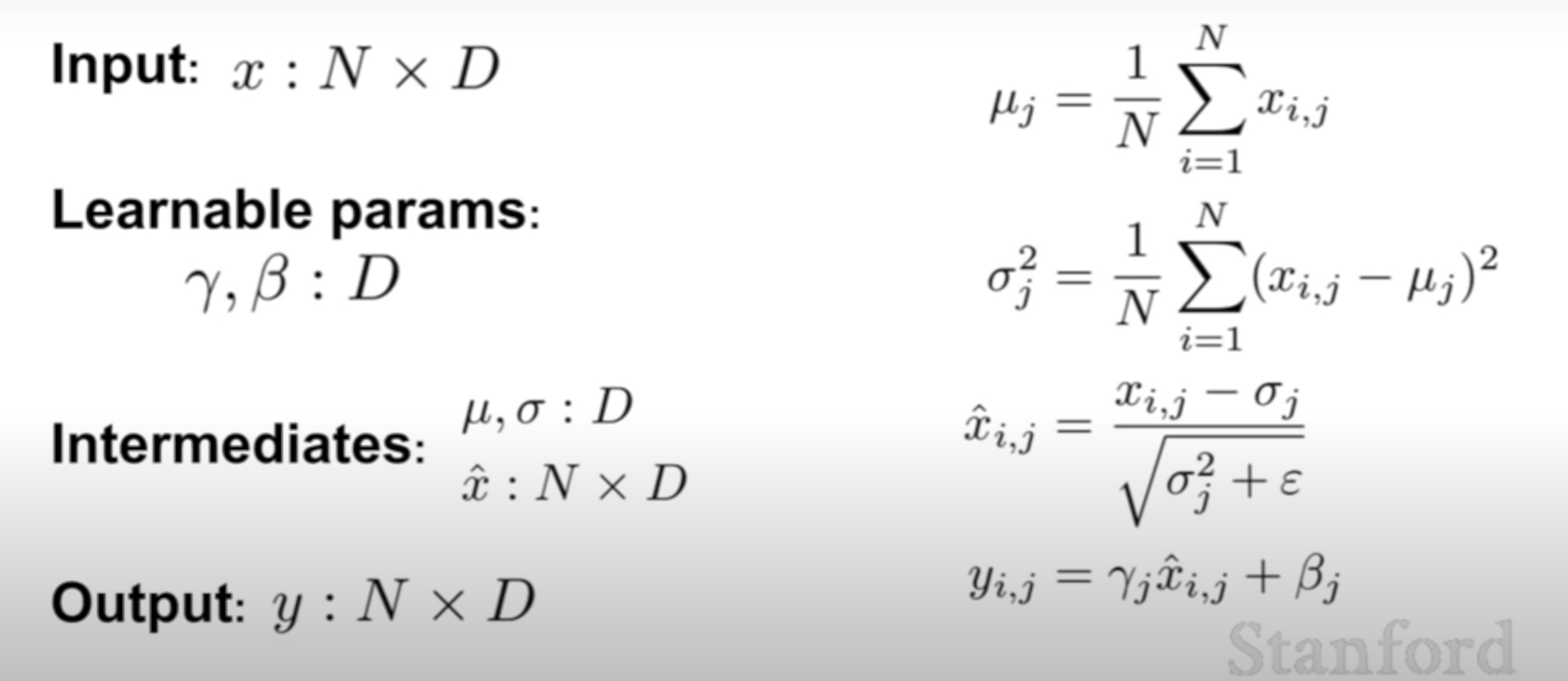
**当输入到tanh激活层，我们是否一定需要采用单元高斯分布输入**？目前不是很清楚。因为采用BN相当于约束输入到tanh非线性的线性机制（即tanh中近似线性的那一段），也就是说我们不要想达到tanh两端饱和的情况。所以我们可以在一定程度上控制这种饱和性。



采用γ进行scaling，采用β进行shift。用这种额外操作的目的是为了方便我们在需要恢复identity function时可以进行恢复。恢复identity map有啥用？因为有时候我们并不知道这种BN是不是最优的方式，对于tanh来说，我们想要合理控制饱和程度。这两个可学习参数对于实现这种控制提供更多灵活性。

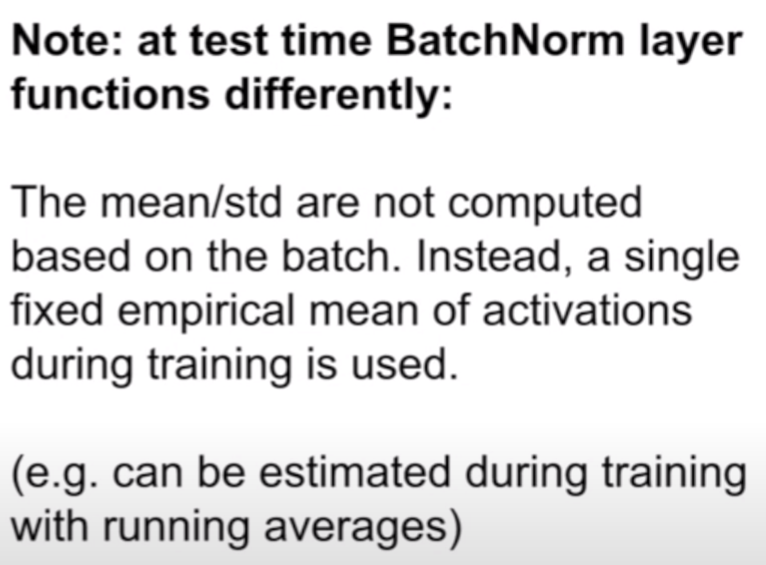
BN总结：





**为什么BN有regularization的效果**？因为我们通过mean和variance对每个输入数据进行了高斯归一化，因此对一个给定的训练样本不再会输出一个确切的值，而是将整个batch的输入都联合一起考虑，相当于对batch输入数据的表征进行扰动，从而产生一定的正则化效果。

**BN会不会改变数据或者特征的结构**？BN的操作是对训练样本进行scale和shift然后形成高斯分布，它修改的只是样本的分布形式，不会对特征结构有影响。而当训练样本经过激活函数后，会对不同的特征进行选择和保留，如果这时候采用BN，就会影响所获取的特征选择，因此我们都是在激活函数前使用BN。**注**：BN改变的只是训练样本，而不是权重。这跟对权重进行高斯初始化的情况完全不同。



测试时期与训练时期中BN的不同点：（1）mean和std不是基于batch样本来计算的，而是采用一个固定的训练过程中的经验激活值的平均值。例如，可以通过在训练中计算平均值而获得。