Batch Normalization (element-wise operation)

why use it: backpropagation tends to training more on the top layers, once bottom layer's weight changed, what we learned before need to be learned again. Hard to converge, internal covariate shift

batch norm ~~allows~~ stablize training process and allows faster convergence

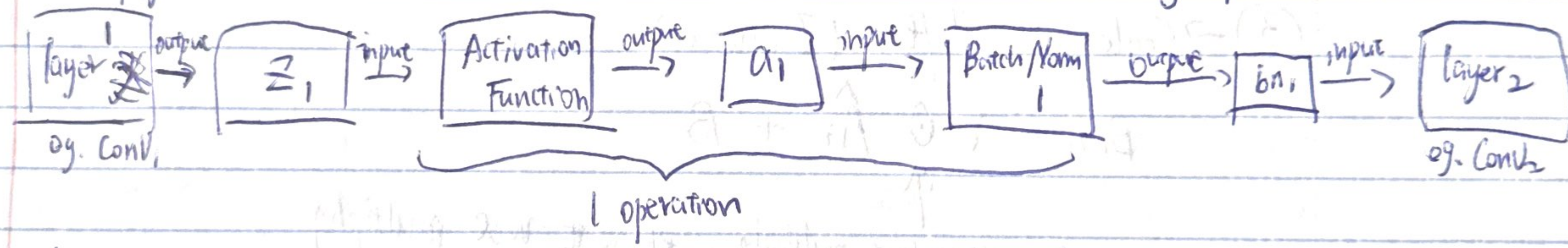
benefit: allows larger learning rate, loss exploding/vanishing gradience

where to use: normalize the input for fist layer and each (hidden) layers

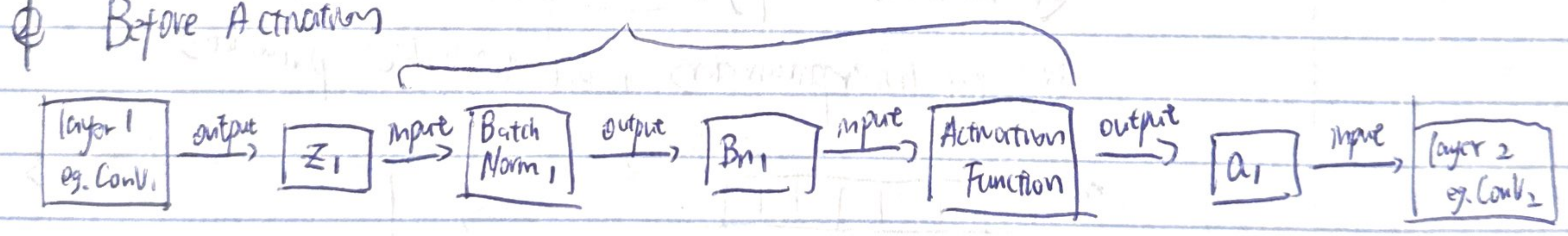
① before activation ② After Activation

② ~~After~~ Activation

itself just ~~like~~ a layer



① Before Activation



Parameters (Each Batch Norm Layer has its own copy of parameters)

learnable: β, γ

saved: mean μ, var σ

Input

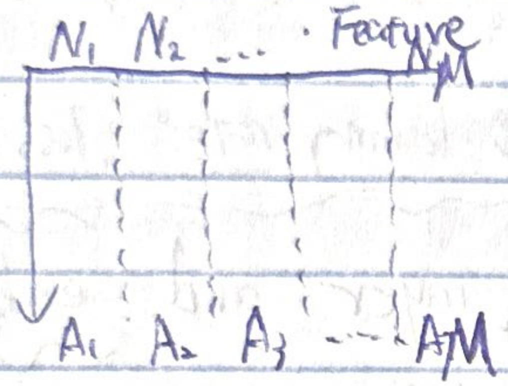
CNN (N, C, H, W) N = # of image in one mini-batch, C = # of feature maps (channels)

Application Process

① Mini-Batch

(N, C, H, W)

each sample is a feature



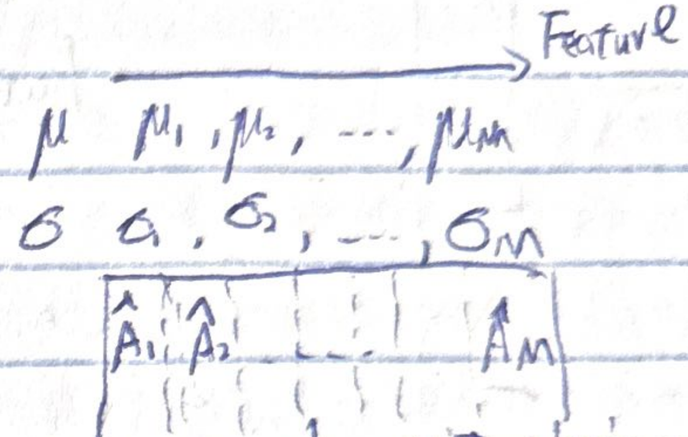
each feature is a vector/matrix map.

$\rightarrow$ ② Normalize

$$\mu_i = \frac{1}{m}\sum A_i = \frac{1}{N \times H \times W}\sum^{N}\sum^{H}\sum^{W} x_{N,H,W}$$

$$\sigma_i = \sqrt{\frac{1}{M}\sum (A_i - \mu)^2}$$

$$\hat{A}_i = \frac{A_i - \mu_i}{\sigma_i}$$



Normalized Value Now have 0 mean and unit Variance / std (1)

③ $\rightarrow$ Scale and Shift (Innovation)

$$BN_i = \gamma \odot \hat{A}_i + \beta$$

☆ not matrix multiply, element-wise multiply

☆ not hyperparameters, but trainable parameters