Various Normalizations & Standardization

Through normalizing the output of a network

→ reduces training time & increases training stability

[Types]

- Batch Normalization
- Weight Normalization
- Layer Normalization
- Group Normalization
- Weight Standardization

Batch Normalization

minibatch x_i —Batch Normalization→ y_i

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

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

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

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

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

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

learnable parameters

} ← probably, the model might perform better with some other mean and covariance

→ Stable & acceleterated training

△ Cons

- requires large enough batch sizes to effectively approximate mean and variance from mini-batch
- different training & test calculation: during inference time, the BN layer doesn't calculate the mean and variance from the test mini-batch, but uses fixed mean & variance from pre-calculated training data
- not applicable to RNNs

Weight Normalization

$w = \frac{g}{\|v\|} v$ } suggested using two parameters instead of w for gradient descent:

- g: the length of the weight vector
- v: the direction of the weight vector

○ pros:
- applicable to RNNs

△ cons:
- less stable

Group Normalization

Applied along feature direction, but divides the features into certain groups and normalizes each group separately

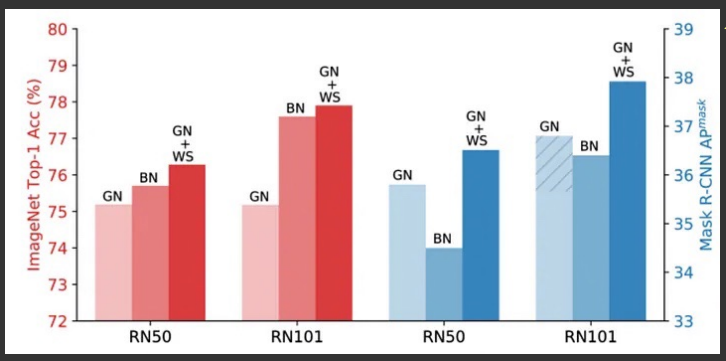
# of groups: tuned as a hyper parameter

- better than layer normalization

9' Weight Standardization

: transforming the weights of any layer to have zero mean and unit variance

For any given layer with shape $(N, *)$ where $*$ represents 1 or more dimensions, weight standardization, transforms the weights along the $*$ dimension(s)



performance of weight standardization + group normalization $\geq$ performance of batch normalization

Helpful when large batch sizes are incapable due to memory constraints

In practice, normalization layers are used in between the Linear / Conv / RNN layer and the ReLU non-linearity (or hyperbolic tangent etc) so that when the activations reach the non-linear activation, the activations are equally centered around zero.