Survey of Normalization Techniques

By: Hiva Mohammadzadeh

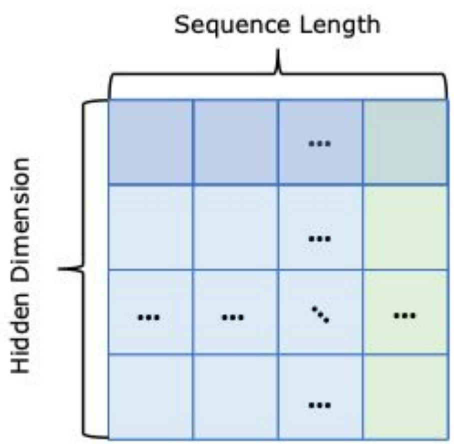
Why we need Normalization?

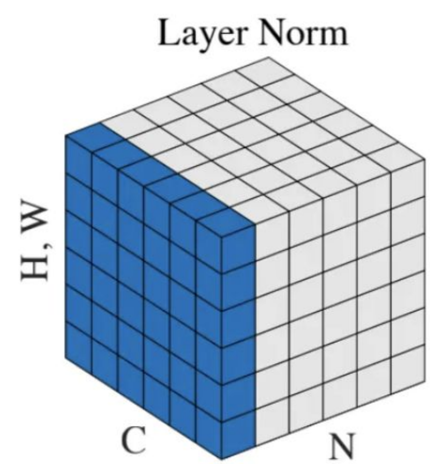
- Normalization is a technique that is very effective in accelerating and stabilizing the learning process of a Neural Network.
- Makes the network unbiased to higher value features.
- Makes optimization faster because it prevents exploding weights and restricts them to a certain range.
- It also helps with regularizing the network.

Layer Norm

- Normalizes input across the features.
- Every example in batch(N) is normalized across [C,H,W] dimensions.
- Speeds up and stabilizes training.
- It does *re-centring* by making model insensitive to shift noises on both inputs and weights. It also does *re-scaling* which keeps the output representations intact when both inputs and weights are randomly scaled.
- Better performance in RNNs and Transformer based models and tasks.
- Applied at test time.

$$p_{out} = \frac{p_{in} - \mu_t}{\sigma_t}\gamma_{e+\beta_e}$$

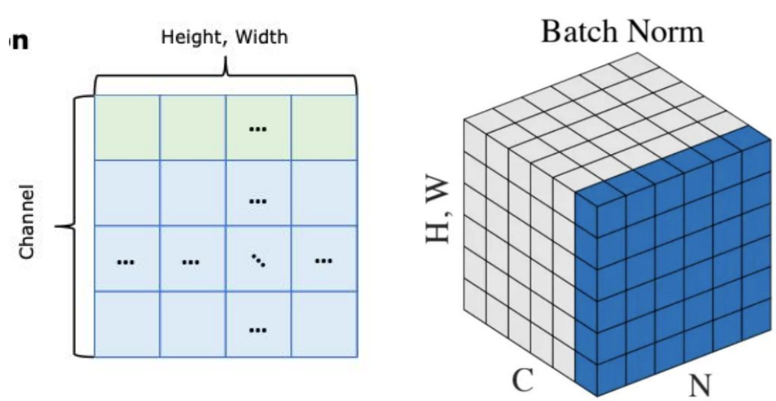




Batch Norm

- Normalizing across the mini-batch dimension.
- Normalizes features by subtracting the mean and dividing the feature by its mini-batch standard deviation.
- Eases optimization and enables very deep networks to converge.
- It also serves as a regularization technique.
- Problems:
  - BN's error increases rapidly when the batch size becomes smaller. Need a large batch size
  - Not good in RNNs because sequences from different samples can have different lengths→ need separate layer for each timestep. More space consuming.
  - Better performance in CNN tasks.

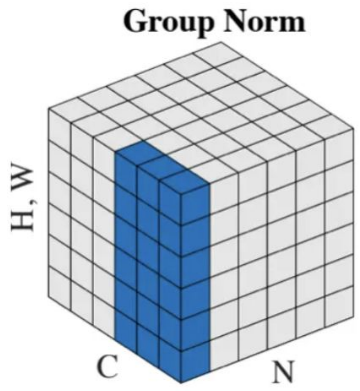
$$p_{out} = \frac{p_{in} - \mu_c}{\sigma_c}\gamma_{c+\beta_c}$$



Group Norm

- Divides the channels for each training example into groups and computes within each group the mean and variance for normalization.
- Independent of batch sizes → more stable
- Group is a sub-vector computed with respect to a cluster
- If we put each each channel into a single group, it becomes Layer Normalization.
- If we put each channel into different groups, it becomes instance norm.
- Makes computation independent of the batch sizes

$$\mu_i = \frac{1}{m} \sum_{k \in \mathcal{S}_i} x_k, \quad \sigma_i = \sqrt{\frac{1}{m} \sum_{k \in \mathcal{S}_i} (x_k - \mu_i)^2 + \epsilon},$$



Root Mean Square Layer Norm

- RMSNorm is an extension of layerNorm that computes the root mean square (RMS) value of the activations across all feature dimensions and channels, resulting in a single scalar value per example.
- More effective than LayerNorm in the presence of data with high variance.
- Gives the model re-scaling invariance property and implicit learning rate adaptation ability. Don't need the re-centering.
- Only one pass over the data instead of 2 that were needed in Layer normalization.
- Computationally simpler and more efficient.

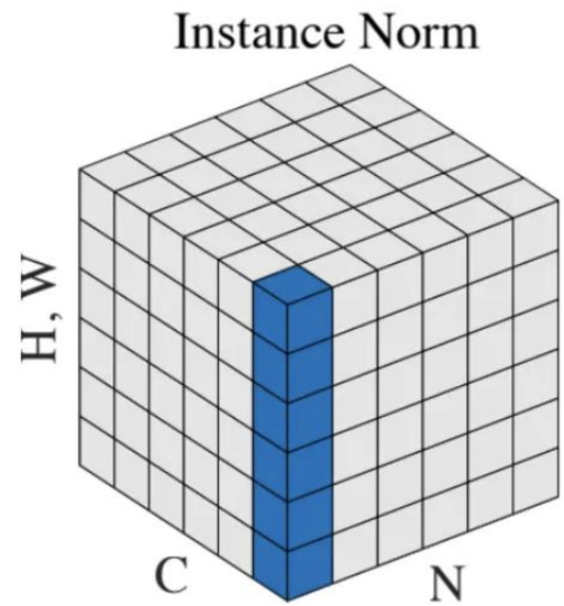
$$\bar{a}_i = \frac{a_i}{\text{RMS}(\mathbf{a})} g_i, \quad \text{where } \text{RMS}(\mathbf{a}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} a_i^2}.$$

Instance Norm

- Like Layer Norm but normalizes across each channel in each training example.
- Applied at test time
- It helps the network be agnostic to the contrast of the original image.

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

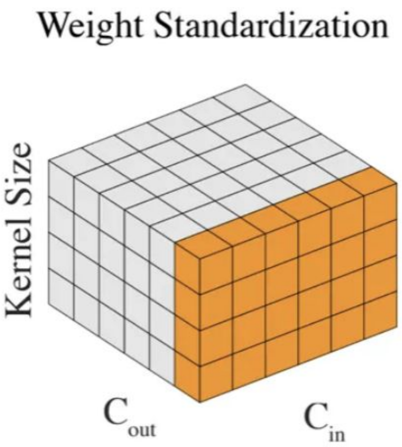
$$\sigma_{ti}^2 = \frac{1}{HW} \sum_{l=1}^{W} \sum_{m=1}^{H} (x_{tilm} - mu_{ti})^2.$$



Weight Norm

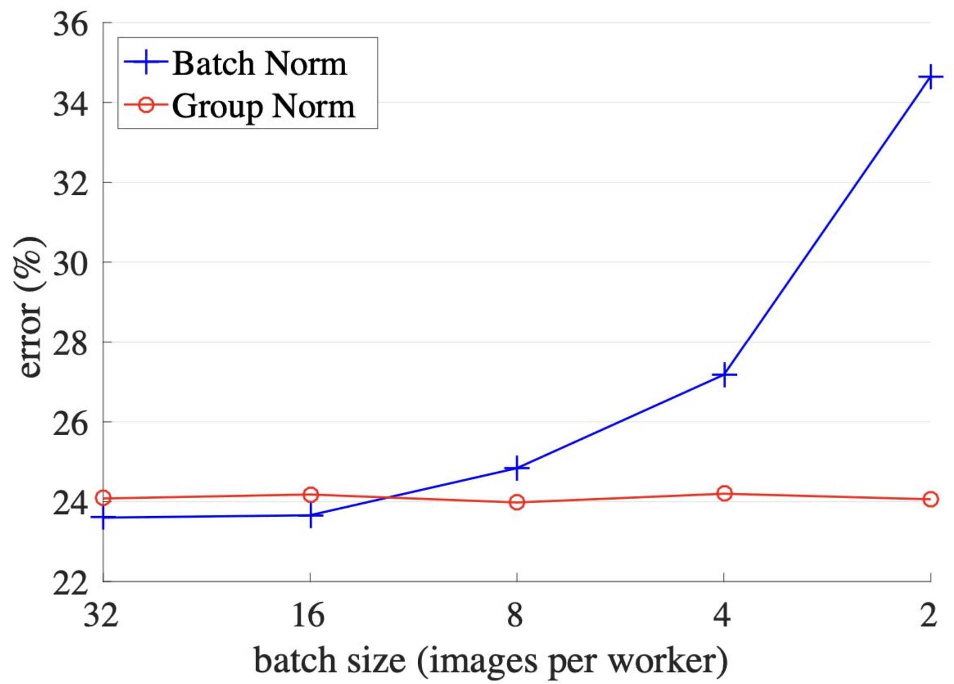
- Normalizes the weights of the layer.
- It separates the weight vector from its direction.
- Then just uses weight normalization instead of dividing by the variance.
- We can achieve a more smooth loss landscape and more stable training.
- Better performance in CNN tasks.

$$\boldsymbol{w} = \frac{g}{\|\boldsymbol{v}\|}\boldsymbol{v}$$



Extra visualization

ImageNet classification error vs. batch sizes



Batch Norm Algo

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

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

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

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

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

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

ϵ is the stability constant in the equation.

Group Norm Algo

```
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
```

Figure 3. Python code of Group Norm based on TensorFlow.

Performance Comparison on MNIST