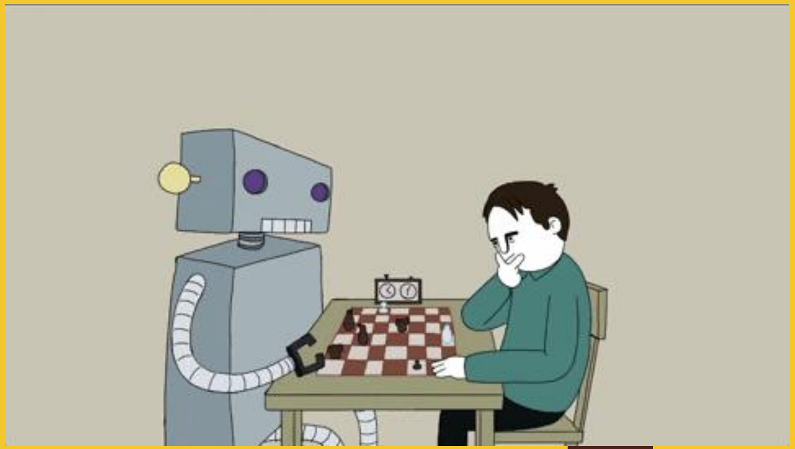
CSE4006 DEEP LEARNING



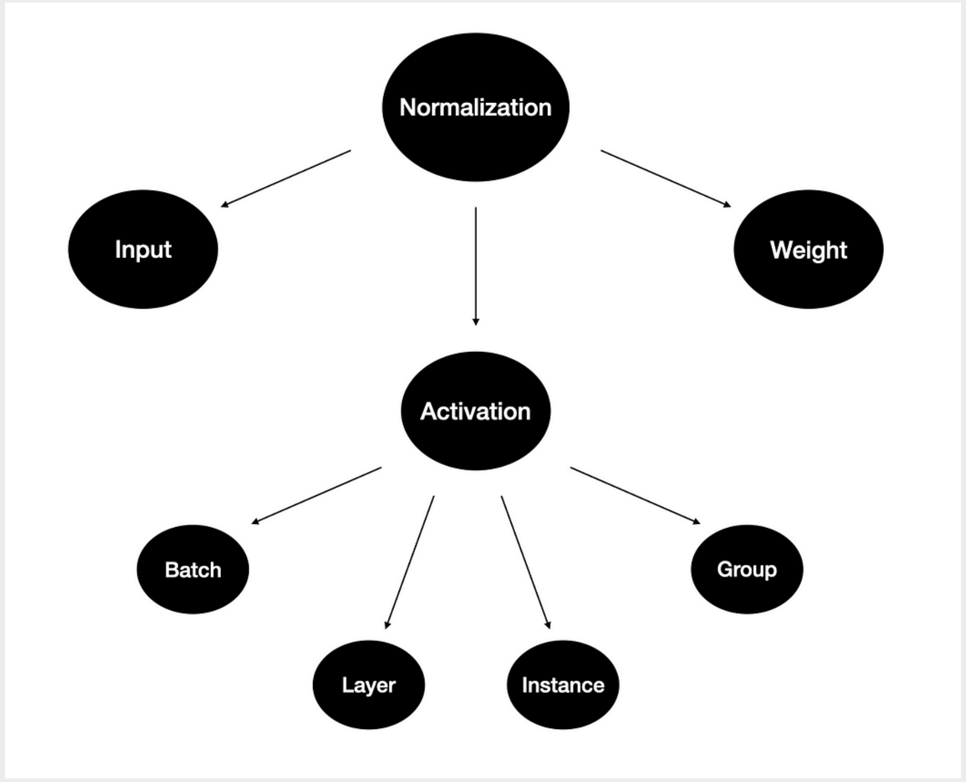
Dr K G Suma
Associate Professor
School of Computer Science and Engineering

Module No. 3
Convolution Neural Networks
7 Hours

- Convolutional networks optimization
- Loss functions in classifiers
- Convolution layers
- Max pool layers
- VGG
- Google Net
- ResNet
- Dropout
- Normalization
- Rules update
- Data augmentation
- Transfer learning
- Analysis of pre trained models

Normalization

- Over the years, researchers have come up with different methods to **accelerate** and **stabilize** the learning process. Normalization is one technique that proved to be very effective in doing this.



Why Normalization?

- Imagine that we have two features and a simple neural network. One is **age** with a range between 0 and 65, and another is **salary** ranging from 0 to 10,000. We feed those features to the model and calculate gradients.

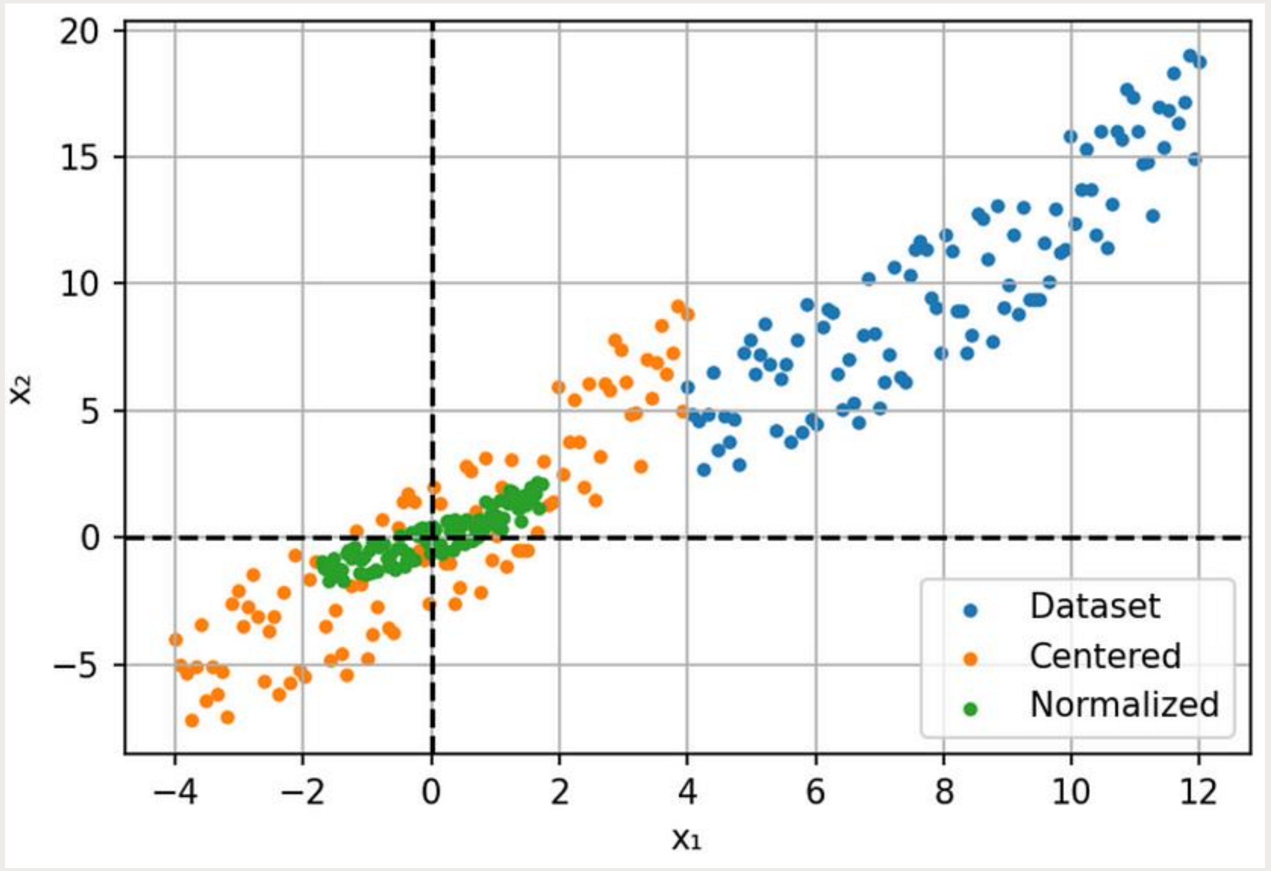
$$W_n = W_n - \alpha(y - \hat{y})x_n$$

- Different scales of inputs cause different weights updates and optimizers steps towards the minimum. It also makes the shape of the loss function disproportional. In that case, we need to use **a lower learning rate** to not overshoot, which means a slower learning process.
- The solution is input normalization. It scales down features by **subtracting** the **mean** (centering) and dividing by **the standard deviation**.

$$\mu_n = \frac{1}{m}\sum_{i=0}^{m} x_{ni} \quad - \quad mean$$

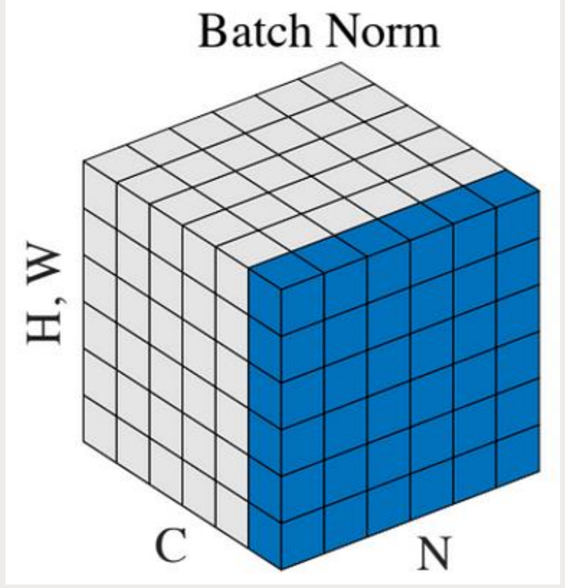
$$\sigma_n = \sqrt{\frac{1}{m}\sum_{i=0}^{m}(x_{ni} - \mu_n)^2} \quad - \quad std$$

$$\hat{x}_n = \frac{x_n - \mu_n}{\sigma_n} \quad - \quad normalized\ input$$



This process is also called '*whitening*', where the values have 0 mean and *unit* variance. It provides **faster convergence** and more **stable training**.

Batch Normalization



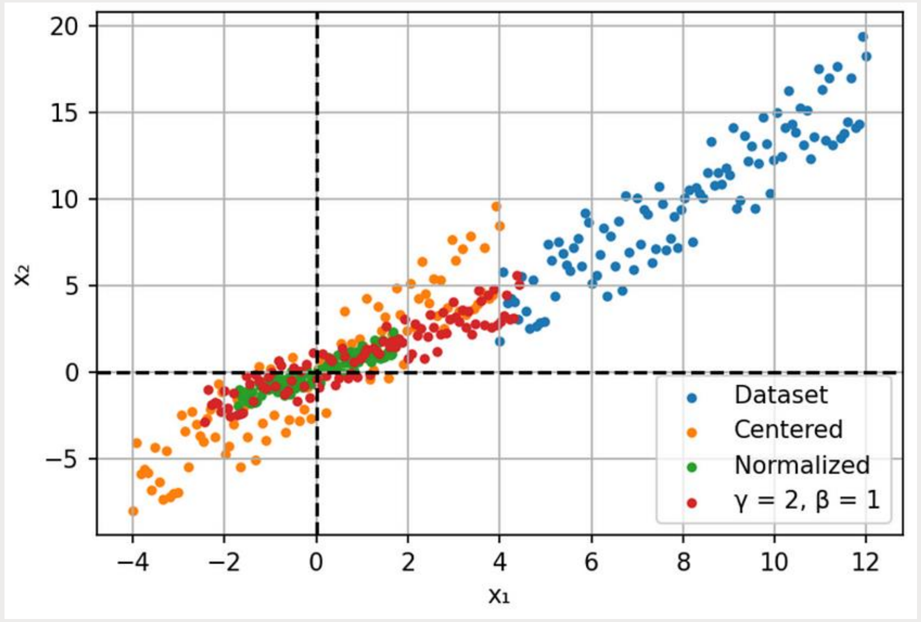
N — batch, C — channels, H,W— spatial width and height

- In 2015 Sergey Ioffe and Christian Szegedy picked up that idea to solve the internal covariate shift issue. In plain English it means that the input layer distribution is **constantly changing** due to weight update. In this case, the following layer always needs to adapt to the new distribution. It causes slower convergence and unstable training.

- Batch Normalization presents a way to **control** and **optimize** the distribution after each layer. The process is identical to the input normalization, but we add two **learnable** parameters, **γ**, and **β**.

Batch Normalization

- These two parameters are learned along the network using backpropagation. They optimize the distribution by **scaling**(γ) and **shifting**(β) activations.



Batch Normalization

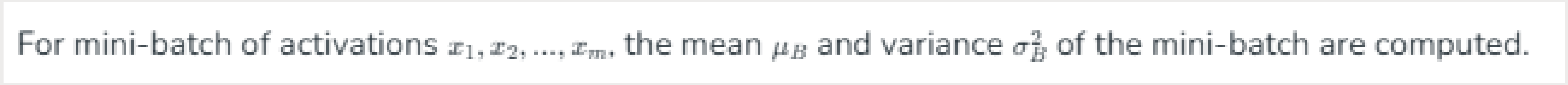
- Batch normalization is a deep learning approach that has been shown to **significantly improve the efficiency and reliability of neural network models**. It is particularly useful for training very deep networks, as it can help to reduce the internal covariate shift that can occur during training.
- **Batch normalization is a supervised learning method for normalizing the interlayer outputs of a neural network.** As a result, the next layer receives a "reset" of the output distribution from the preceding layer, allowing it to analyze the data more effectively.
- The term "internal covariate shift" is used to describe the effect that updating the parameters of the layers above it has on the distribution of inputs to the current layer during deep learning training. This can make the optimization process more difficult and can slow down the convergence of the model.
- Since normalization guarantees that no activation value is too high or too low, and since it enables each layer to learn independently from the others, this strategy leads to quicker learning rates.
- By standardizing inputs, the "dropout" rate (the amount of information lost between processing stages) may be decreased. That ultimately leads to a vast increase in precision across the board.

How does batch normalization work?

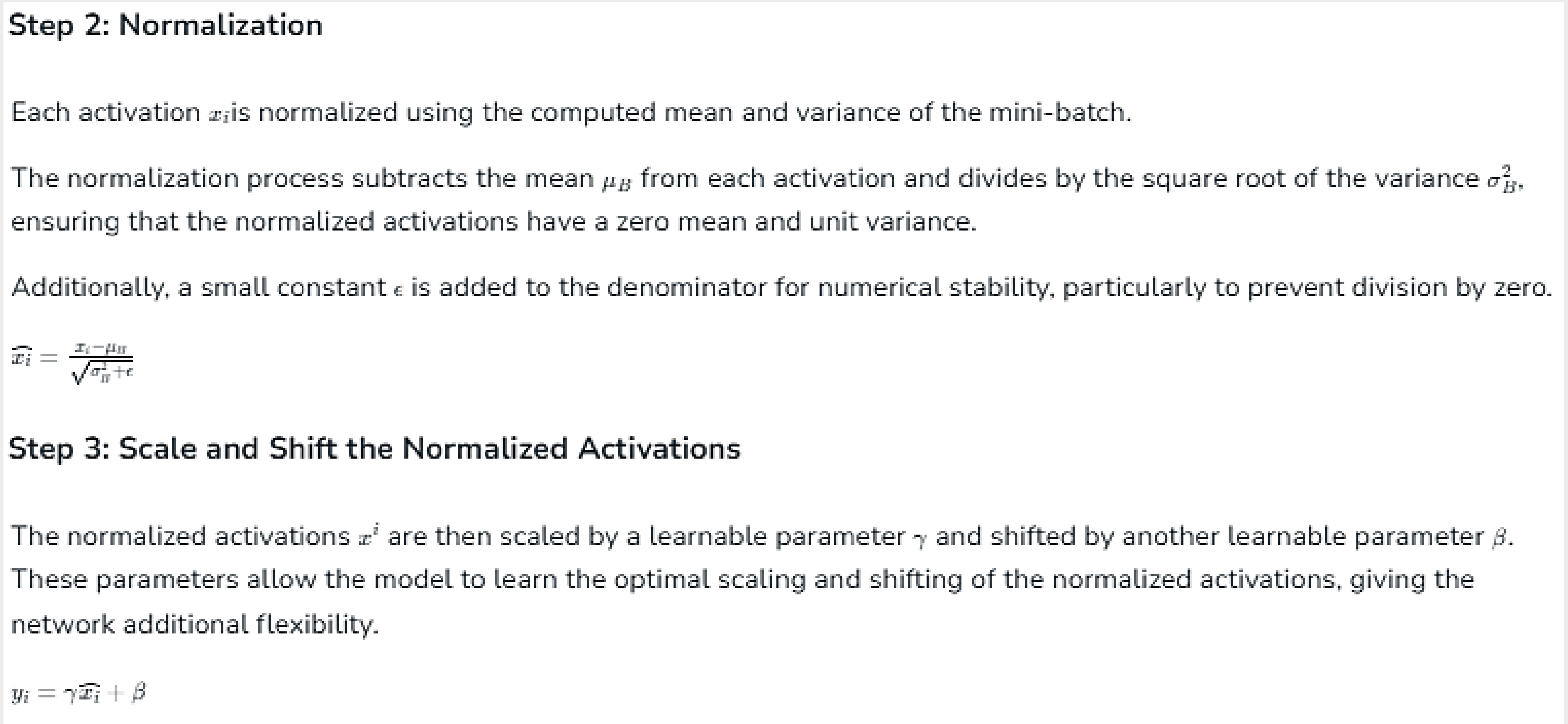
- Batch normalization is a technique used to improve the performance of a deep learning network by first removing the batch mean and then splitting it by the batch standard deviation.
- When applied to a layer, batch normalization multiplies its output by a standard deviation parameter (gamma) and adds a mean parameter (beta) to it as a secondary trainable parameter. Data may be "denormalized" by adjusting just these two weights for each output, thanks to the synergy between batch normalization and gradient descents. Reduced data loss and improved network stability were the results of adjusting the other relevant weights.
- **The goal of batch normalization is to stabilize the training process and improve the generalization ability of the model.** It can also help to reduce the need for careful initialization of the model's weights and can allow the use of higher learning rates, which can speed up the training process.

Fundamentals of Batch Normalization

Step 1: Compute the Mean and Variance of Mini-Batches



For mini-batch of activations $x_1, x_2, ..., x_m$, the mean μ_B and variance σ_B^2 of the mini-batch are computed.



Step 2: Normalization

Each activation $x_i$is normalized using the computed mean and variance of the mini-batch.

The normalization process subtracts the mean μ_B from each activation and divides by the square root of the variance σ_B^2, ensuring that the normalized activations have a zero mean and unit variance.

Additionally, a small constant ϵ is added to the denominator for numerical stability, particularly to prevent division by zero.

$$\widehat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

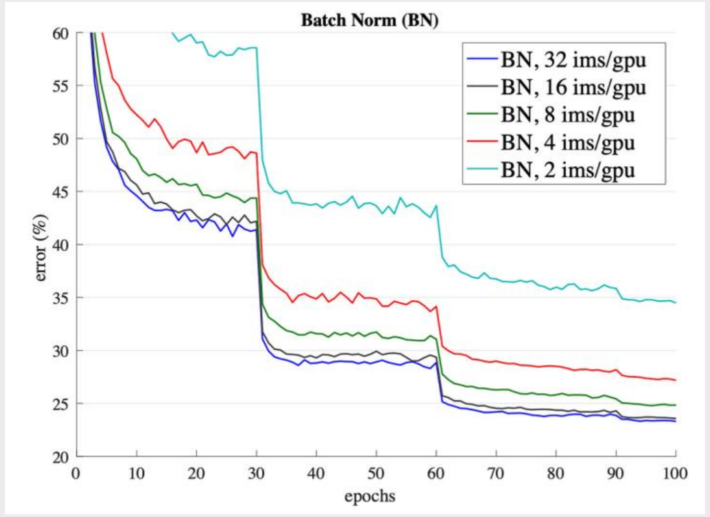
Step 3: Scale and Shift the Normalized Activations

The normalized activations x^i are then scaled by a learnable parameter γ and shifted by another learnable parameter β. These parameters allow the model to learn the optimal scaling and shifting of the normalized activations, giving the network additional flexibility.

$$y_i = \gamma\widehat{x_i} + \beta$$

Batch Normalization

- Since we have fixed distributions, we can increase the learning rate and speed up the convergence. Besides computational boost, BN also serves as a **regularisation** technique. The noise generated by approximation of the dataset's statistics removes the need for a Dropout.
- But it's a double-edged sword. This estimation is only tolerable for larger batches. When the number of examples is smaller, the performance decreases dramatically.



Benefits of Batch Normalization

- **Faster Convergence:** Batch Normalization reduces internal covariate shift, allowing for faster convergence during training.
- **Higher Learning Rates:** With Batch Normalization, higher learning rates can be used, can to speed up the training process without the risk of divergence.
- **Regularization Effect:** Batch Normalization introduces a slight regularization effect that reduces the need for adding regularization techniques like dropout.
- **Stabilize the training process**. Batch normalization can help to reduce the internal covariate shift that occurs during training, which can improve the stability of the training process and make it easier to optimize the model.
- **Improves generalization**. By normalizing the activations of a layer, batch normalization can help to reduce overfitting and improve the generalization ability of the model.

Batch normalization overfitting

- While batch normalization can help to reduce overfitting, it is not a guarantee that a model will not overfit.
- Overfitting can still occur if the model is too complex for the amount of training data, if there is a lot of noise in the data, or if there are other issues with the training process. It is important to use other regularization techniques like dropout, and to monitor the performance of the model on a validation set during training to ensure that it is not overfitting.

Batch Normalization

- Another downside of the BN is **testing time**. Let’s use a self-driving car as an example. You pass a single frame recorded by the camera during the driving rather than the batch of images. In this case, the network has to use pre-computed mean and variance from training, which might lead to **different results**.
- The significance of this problem pushed the community to create alternative methods to avoid dependency on the batch.

Conclusion

- Batch Normalization is a powerful technique for stabilizing the training of deep neural networks. By normalizing the inputs of each layer, it addresses issues like vanishing gradients and accelerates convergence.
- In Keras, integrating Batch Normalization into your models is simple and can lead to significant improvements in performance. As you explore the depths of deep learning, consider harnessing the power of Batch Normalization to enhance the training of your neural networks.

Weight Normalization

- Weight Normalization was introduced as an alternative to BatchNorm, addressing some of its limitations. Here we want to reparametrize the weights of the network such that we separate the direction of the weight vectors from their magnitude. So we write a weight vector *w* as:

where *a* is a scalar and *v* is a vector of the same length as *w*.

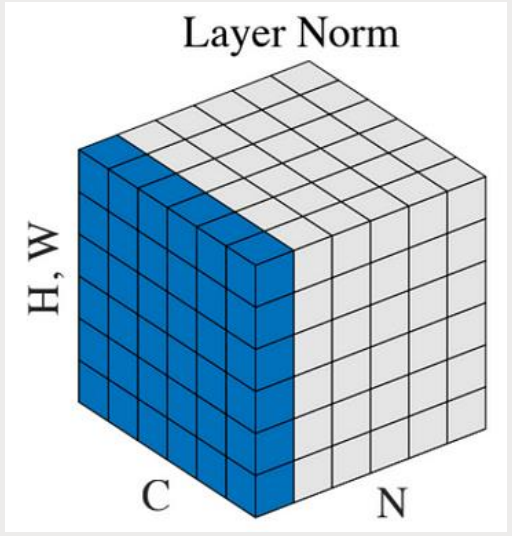
It decouples the importance of the mean and variance from the batch and incorporates them as a weight factor

$$w = \frac{a * v}{||v||}$$

Layer Normalization

- In layer normalization we normalize across the feature dimension, applying the same normalization to every hidden unit in the layer.

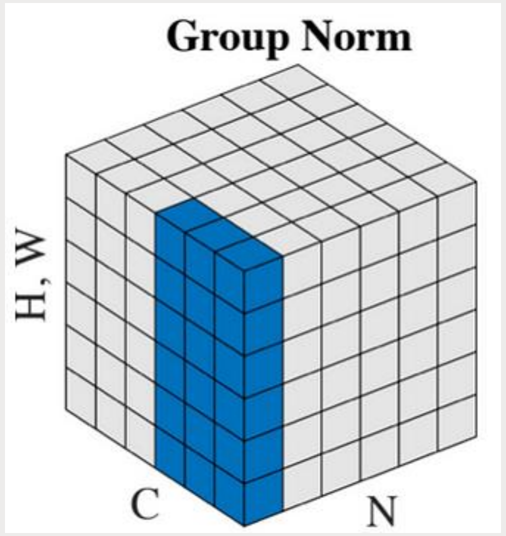
- Layer Normalization works well with small mini-batches or different layer sizes. It is particularly effective in recurrent neural networks and is invariant to the scaling of weights and inputs.



Group Normalization

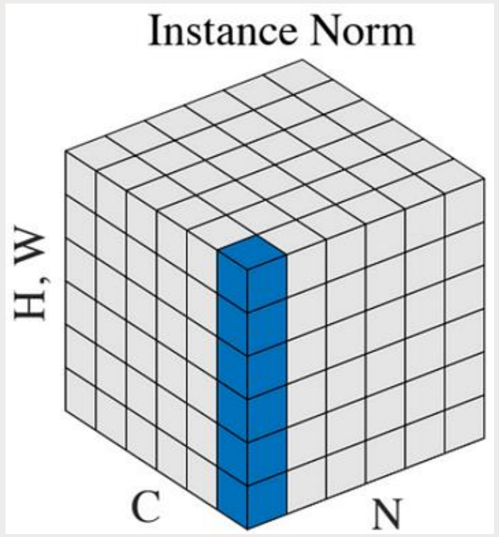
- Group Normalization is indeed very similar to Layer Normalization since we compute the mean and standard deviation for each training example, but instead of computing it across all channels for each example, we do so across subsets of channels.

- The key advantage of Group Normalization over Layer Normalization lies in its ability to capture channel-wise relationships while still maintaining some level of independence between different feature groups.



Instance Normalization

- Instance normalization simply applies normalization to each instance in a batch independently. Similar to batch normalization but we normalize for only one element of the batch independently. Instead of computing one mean and standard deviation for the batch, we compute n of them.



Spectral Normalization

- Spectral normalization was a method originally proposed to improve training in Generative Adversarial Networks (GANs) to address the mode collapse issue.
- Spectral normalization simply makes training more stable by controlling the Lipschitz constant of the network. This Lipschitz property guarantees a smooth and sensitive function, which is often desirable for creating a measure of distance between inputs and hidden states of the network denoted by *f*.

$$L_1 * ||x_1 - x_2||_X \leq ||f(x_1) - f(x_2)||_F \leq L_2 * ||x_1 - x_2||_X$$