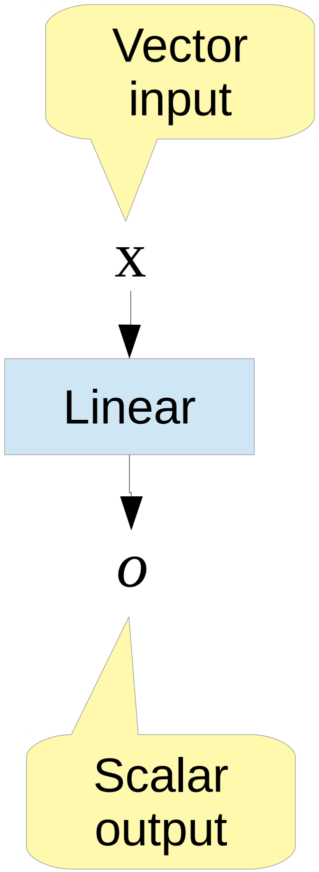
Normalization

Why Normalize?



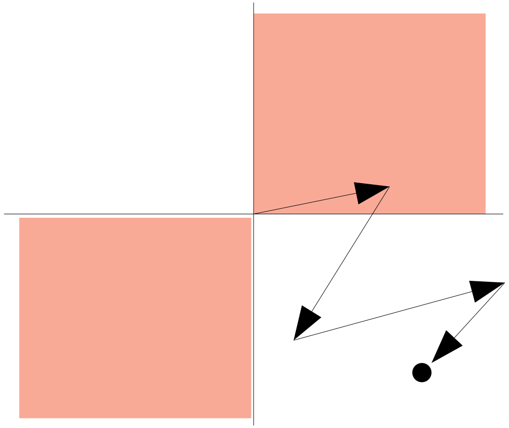
$$o = \mathrm{w}\,\mathrm{x}$$

$$\frac{\partial \ell(o)}{\partial \mathrm{w}} = \left(\frac{\mathrm{d}\,\ell(o)}{\mathrm{d}\,o}\right)\mathrm{x}^{\mathrm{T}}$$

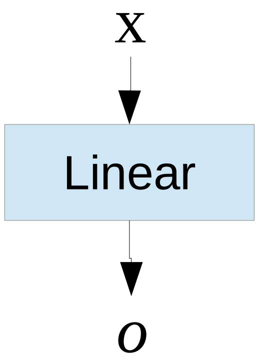
Scalar derivative

$$\mathrm{x}_i > 0$$

$$\left(\frac{\partial \ell(o)}{\partial \mathrm{w}}\right)_i \geq 0 \qquad \left(\frac{\partial \ell(o)}{\partial \mathrm{w}}\right)_i \leq 0$$



Why Normalize?

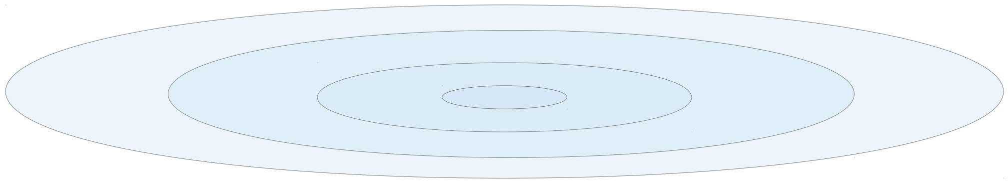


$$o = \mathrm{w}\,\mathrm{x}$$

$$\frac{\partial \ell(o)}{\partial \mathrm{w}} = \left(\frac{\mathrm{d}\,\ell(o)}{\mathrm{d}\,o}\right)\mathrm{x}^{T}$$

$$\mathrm{x} \in \mathbb{R}^2 \qquad |\mathrm{x}_1| \ll |x_2|$$

$$\frac{\partial \ell(o)}{\partial \mathrm{w}_1} \ll \frac{\partial \ell(o)}{\partial \mathrm{w}_2}$$



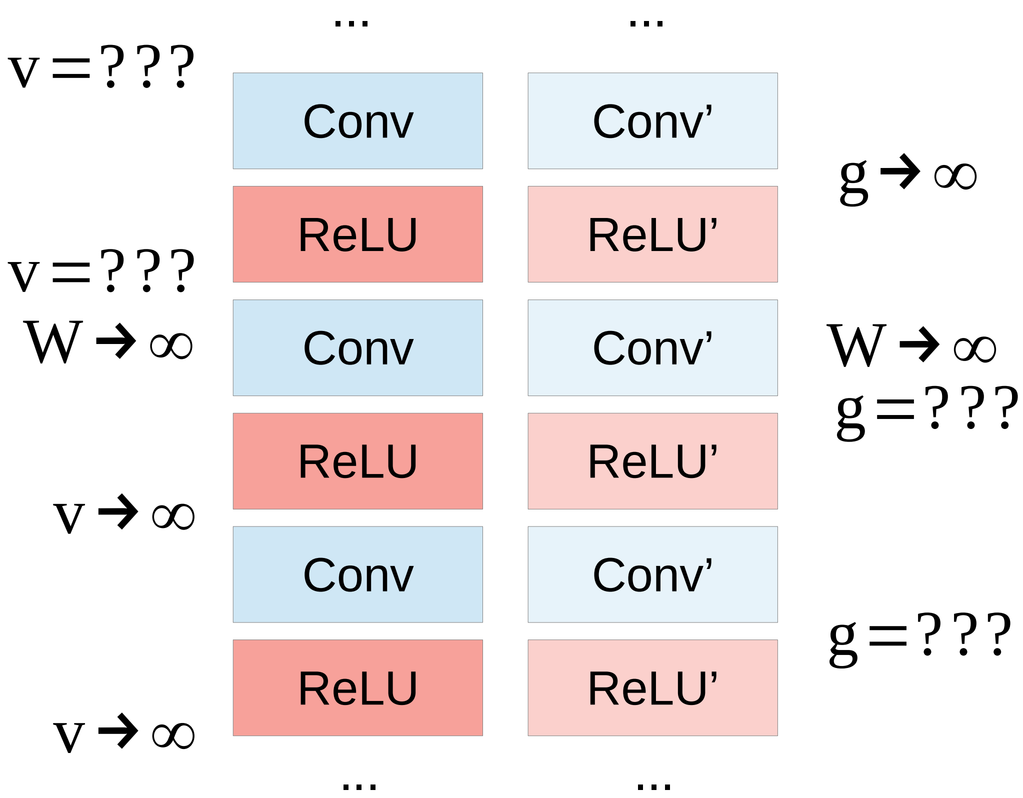
Input Normalization

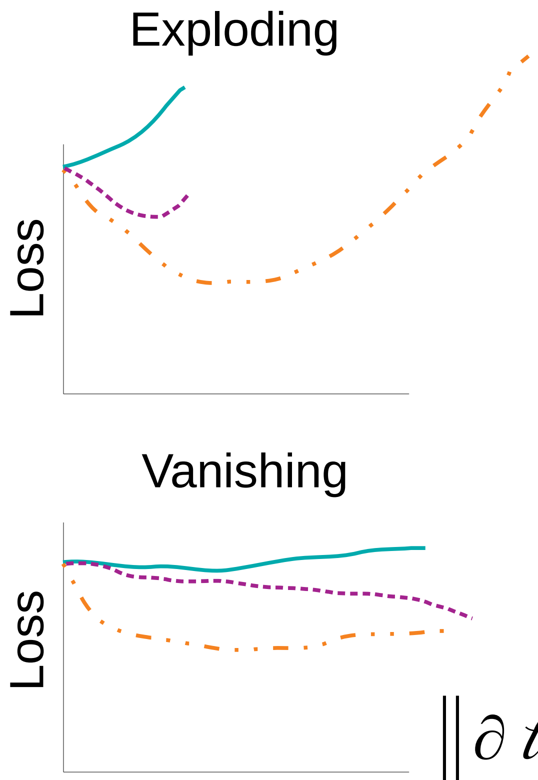
$$x_i \qquad \frac{x_i - \mu_x}{\sigma_x}$$

Average over all elements of all inputs

For images, compute mean and standard deviation for each channel – that is, one red mean, one blue mean, and one green mean.

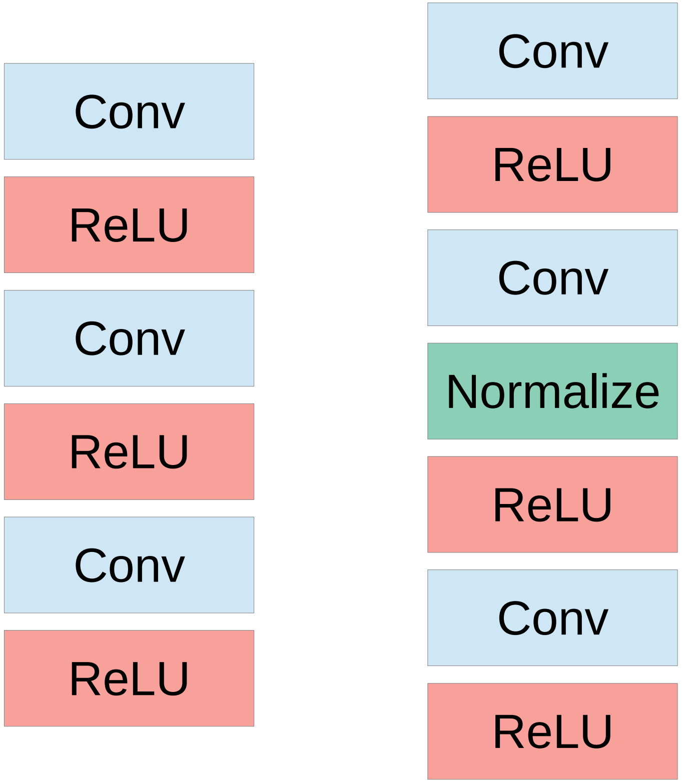
Vanishing / Exploding Gradients





$$\left\|\frac{\partial \ell(o)}{\partial W_i}\right\| \ll \|W_i\|$$

Normalization



$$y = \alpha x + \beta$$

$$E[y] = 0$$

$$Var[y] = 1$$

Batch Normalization

Conv

BatchNorm

ReLU

$$y = \alpha x + \beta$$

Over the entire batch

$$E[y] = 0$$

$$\mathrm{Var}[y] = 1$$

$$x \in \mathbb{R}^{B \times C \times H \times W}$$

$$y_{i,c,j,k} = \frac{x_{i,c,j,k} - \mu_c}{\sigma_c}$$

$$\mu_c = \frac{1}{BHW} \sum_{i,j,k} x_{i,c,j,k}$$

$$\sigma_c^2 = \frac{1}{BHW} \sum_{i,j,k} \left(x_{i,c,j,k} - \mu_c \right)^2$$

Batch Normalization

- ✔ Keeps the activation magnitudes in check
- ✔ Deals with badly scaled weights
- ✗ Mixes gradient information between inputs
  - Mitigated by large batches

$$x \in \mathbb{R}^{B \times C \times H \times W}$$

$$x_{i,c,j,k} \rightarrow \infty$$

$$\mu_c \rightarrow \infty \quad \sigma_c \rightarrow \infty$$

BatchNorm at Test Time

- Usually we don’t test on a batch of data.

- Keep a running average of the mean and standard deviation during training, then save those values.

Layer Normalization

- Same as BatchNorm, but we compute statistics per input rather than per channel.
- Prevents cross-talk.
- Training and testing are the same.
- In practice, works well for sequence models but not in computer vision.

$$\mathrm{x} \in \mathbb{R}^{B \times C \times H \times W}$$

$$\mu_i = \frac{1}{CHW} \sum_{c,j,k} \mathrm{x}_{i,c,j,k}$$

$$\sigma_i^2 = \frac{1}{CHW} \sum_{c,j,k} \left(\mathrm{x}_{i,c,j,k} - \mu_i \right)^2$$

Instance Normalization

- Compute statistics per input *and* per channel
  - Sum over *only* spatial locations
- Statistics are unstable
- Not so good in recognition
- Works okay for image generation and computer graphics

$$x \in \mathbb{R}^{B \times C \times H \times W}$$

$$\mu_{i,c} = \frac{1}{HW} \sum_{j,k} x_{i,c,j,k}$$

$$\sigma_{i,c}^2 = \frac{1}{HW} \sum_{j,k} \left(x_{i,c,j,k} - \mu_{i,c} \right)^2$$

Group Normalization

- Compute statistics over groups of channels
  - Between instance normalization and layer normalization
- More flexible than layer normalization, more stable than instance normalization.

$$x \in \mathbb{R}^{B \times C \times H \times W}$$

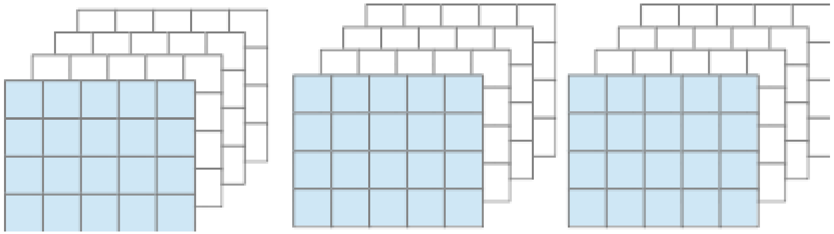
$$S = \lfloor C / G \rfloor$$

$$\mu_{i,g} = \frac{1}{SHW} \sum_{j,k} \sum_{c=S(g-1)}^{Sg-1} x_{i,c,j,k}$$

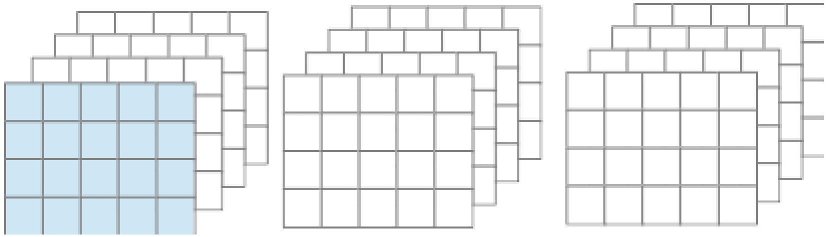
$$\sigma_{i,g}^2 = \frac{1}{SHW} \sum_{j,k} \sum_{c=S(g-1)}^{Sg-1} \left(x_{i,c,j,k} - \mu_{i,g}\right)^2$$

Summary

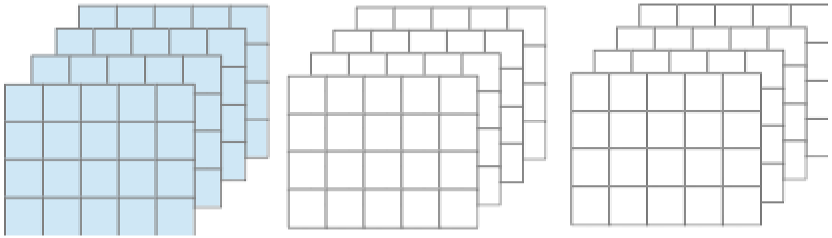
Batch normalization



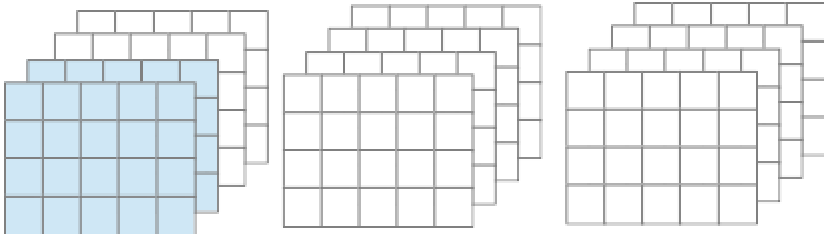
Instance normalization



Layer normalization

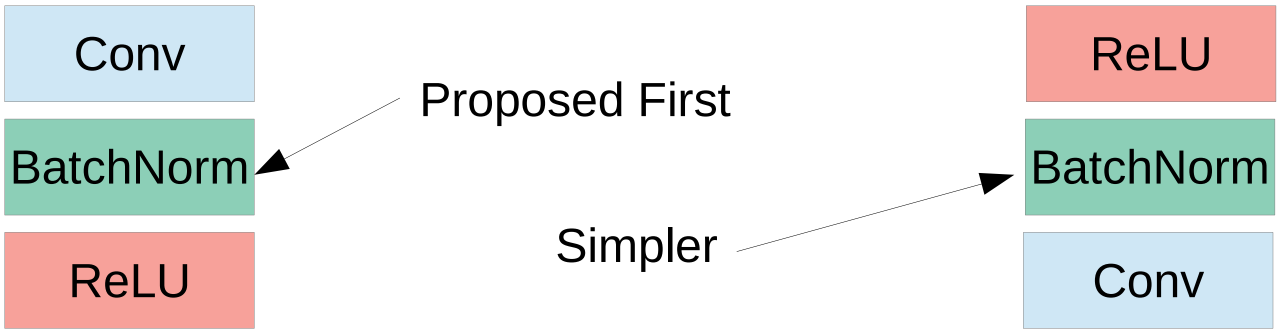


Group normalization



Normalization in Practice

$$y=\frac{x-\mu_c}{\sigma_c}\gamma+\beta$$



- No bias needed in Conv
- Activations are zero mean
  - ReLU will zero out half of activations
- Learn a scale and bias parameter in the normalization layer (`affine=True`)

- Scale and bias in the normalization layer are optional (`affine=False`).
- Conv is unchanged

NOTE: Do not normalize after linear layers (statistical estimates are too unstable)