Normalization
Why Normalize?

\[ o = w x \]

\[ \frac{\partial \mathcal{L}(o)}{\partial w} = \left( \frac{d \mathcal{L}(o)}{d o} \right) x^T \]

\[ x_i > 0 \]

\[ \left( \frac{\partial \mathcal{L}(o)}{\partial w} \right)_i \geq 0 \]

\[ \left( \frac{\partial \mathcal{L}(o)}{\partial w} \right)_i \leq 0 \]
Why Normalize?

\[ o = w \cdot x \]

\[ \frac{\partial \ell(o)}{\partial w} = \left( \frac{d \ell(o)}{d o} \right) x^T \]

\[ x \in \mathbb{R}^2 \quad |x_1| \ll |x_2| \]

\[ \frac{\partial \ell(o)}{\partial w_1} \ll \frac{\partial \ell(o)}{\partial w_2} \]
Input Normalization

\[ X_i \rightarrow \frac{X_i - \mu_x}{\sigma_x} \]

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

\[ v \rightarrow \infty \]

\[ v = ??? \]

\[ \text{Conv} \quad \text{Conv'} \]

\[ \text{ReLU} \quad \text{ReLU'} \]

\[ W \rightarrow \infty \]

\[ W \rightarrow \infty \]

\[ g \rightarrow \infty \]

\[ g = ??? \]

\[ g = ??? \]

\[ v \rightarrow \infty \]

\[ v \rightarrow \infty \]

\[ \text{Vanishing} \]

\[ \text{Exploding} \]

\[ \left\| \frac{\partial \ell(o)}{\partial W_i} \right\| \ll \| W_i \| \]
Normalization

\[ y = \alpha x + \beta \]

\[ \mathbb{E}[y] = 0 \]

\[ \text{Var}[y] = 1 \]
Batch Normalization

$y = \alpha x + \beta$

$\mathbb{E}[y] = 0$

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

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

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

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

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

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

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

\[ x_{i,c,j,k} \to \infty \]

\[ \mu_c \to \infty \quad \sigma_c \to \infty \]
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.

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

\[
\mu_i = \frac{1}{CHW} \sum_{c,j,k} x_{i,c,j,k}
\]

\[
\sigma_i^2 = \frac{1}{CHW} \sum_{c,j,k} (x_{i,c,j,k} - \mu_i)^2
\]
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.

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

\[S = \left\lfloor \frac{C}{G} \right\rfloor\]

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

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

Batch normalization

Instance normalization

Layer normalization

Group normalization
Normalization in Practice

\[ y = \frac{x - \mu_c}{\sigma_c} y + \beta \]

- 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)