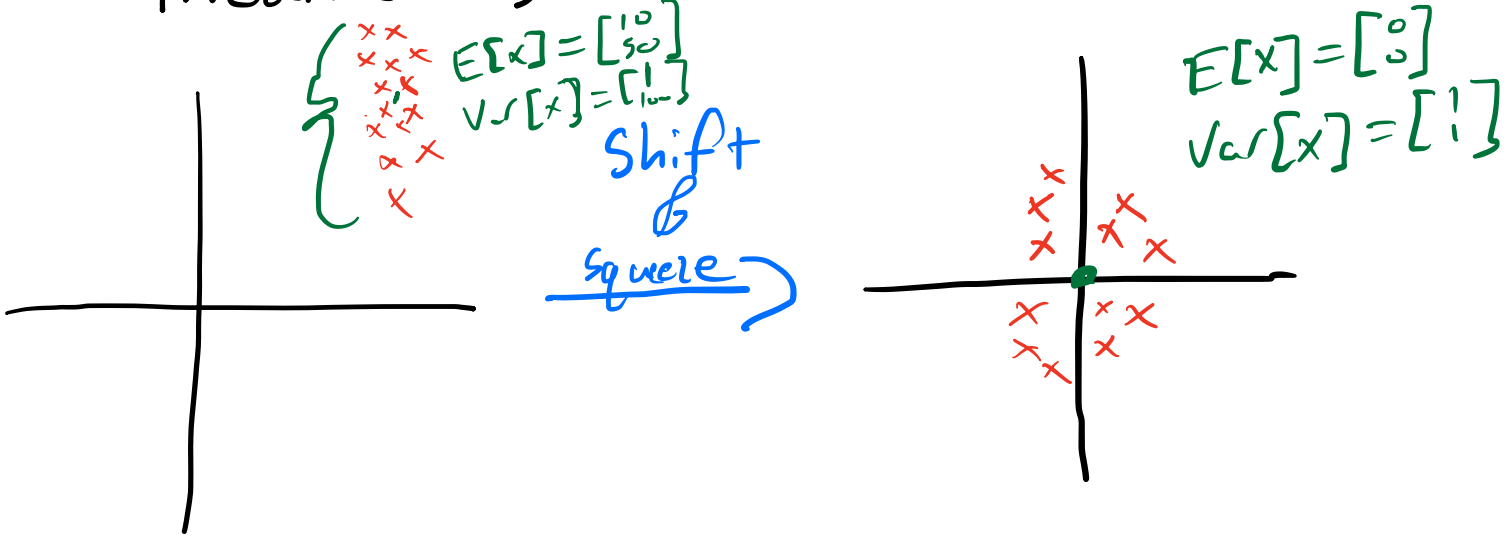
Last time

Normalization: keep data at consistent mean ($E[x]$) and variance ($Var[x]$).



Motivation

1) Gradient Scale

$$\left|\frac{dL}{dw_1}\right| = |x|\,|\sigma'(xw_1+b_1)| \prod_{i=2}^{l} \left|\frac{d\phi_i}{d\phi_{i-1}}\right| \dots$$

(Prevent scale of data from affecting gradient)

2) Scale mismatch

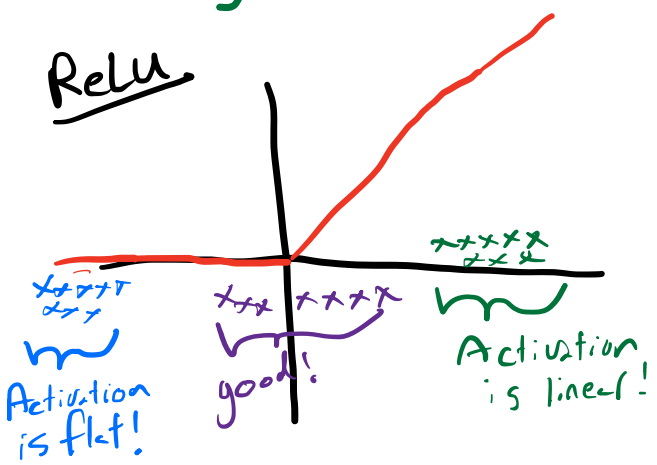
e.g. $x_1 = \begin{bmatrix} 2152 \\ 0.02 \end{bmatrix}$ $x_2 = \begin{bmatrix} 1092 \\ 0.017 \end{bmatrix}$ ...

We saw why this is a problem!

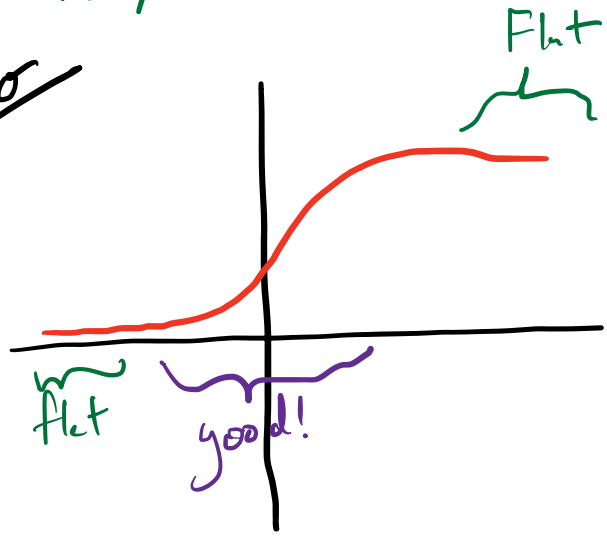
Motivation

3) Activations only make sense near 0

ReLU



or



Normalization

$$\text{Norm}(x) = \frac{x - E[x]}{\sqrt{\text{Var}[x] + \epsilon}}$$

Prevent div. by 0! $\epsilon \ll 1$

Estimate $E[x]$, $\text{Var}[x]$ from dataset or Batch

$$E[x] \approx \bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

$$\text{Var}[x] \approx s^2 = \frac{1}{N-1} \sum_{i=1}^{N} \left(x_i - \bar{x} \right)^2$$

Unbiased estimator

$$E[Norm(x)] = E\left[\frac{x - E[x]}{\sqrt{Var[x] + \epsilon}}\right]$$

$$= \frac{1}{\sqrt{Var[x] + \epsilon}}\left[E[x] - E[x]\right] = 0$$

Batch Normalization

$$\text{BatchNorm}(x) = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}}$$

ϵ: prevent div. by 0!

Estimate $E[x]$, $Var[x]$ from dataset or **Batch**

← Batch sampled for SGD

$$E[x] \approx \bar{x} = \frac{1}{B}\sum_{i=1}^{B} x_i$$

$$Var[x] \approx s^2 = \frac{1}{B-1}\sum_{i=1}^{B}\left(x_i - \bar{x}\right)^2$$

$B-1$: Unbiased estimator

Deep Network

$$f(x) = \phi_\ell(\phi_{\ell-1}(\dots \phi_1(x)))w_0 + b_0$$

Composing feature transforms

e.g.

$$f(x) = \sigma(\sigma(\dots\sigma(x^T W_1 + b_1)^T W_2 + b_2)\dots)^T W_\ell + b_\ell)w_0 + b_0$$

or

$$\phi_1 = \sigma(x^T W_1 + b_1)$$
$$\phi_2 = \sigma(\phi_1^T W_2 + b_2)$$
$$\vdots$$
$$\phi_\ell = \sigma(\phi_{\ell-1}^T W_\ell + b_\ell)$$

$$f = \phi_\ell^T w_0 + b_0$$
$$L = \text{Loss}(f, y)$$

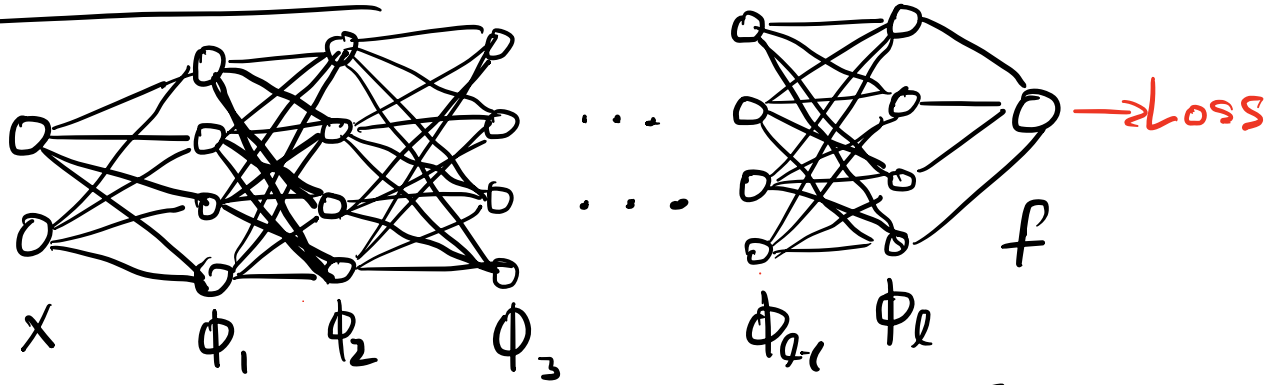
↓

e.g. $(f-y)^2$ MSE

↳ Is our internal representation normalized? Consistent?

Deep Network

ℓ-Layer network



or

$$\phi_1 = \sigma(x^T W_1 + b_1)$$
$$\phi_2 = \sigma(\phi_1^T W_2 + b_2)$$
$$\vdots$$
$$\phi_\ell = \sigma(\phi_{\ell-1}^T W_\ell + b_\ell)$$

$$f = \phi_\ell^T W_0 + b_0$$
$$L = \text{Loss}(f, y)$$

e.g. $(f-y)^2$

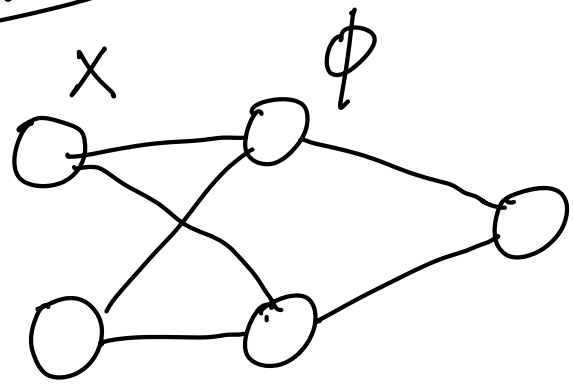
$E[x] = 0$ $\quad Var[x] = 1$ w/ Norm.

$E[\phi_{\ell-1}] = ?$ $\quad Var[\phi_{\ell-1}] = ?$

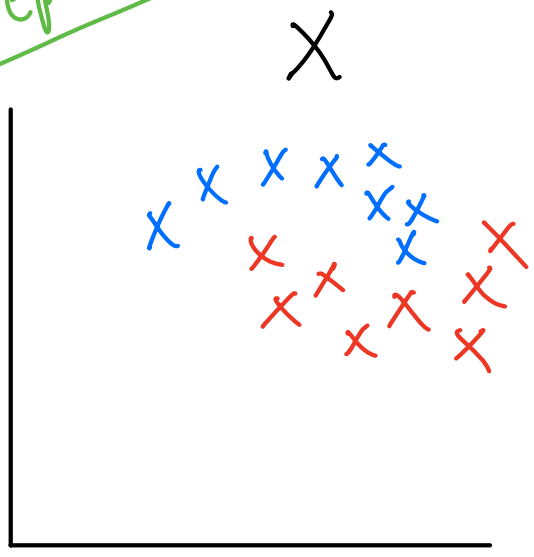
Depend on $W_1 \ldots W_{\ell-1}$ which could change

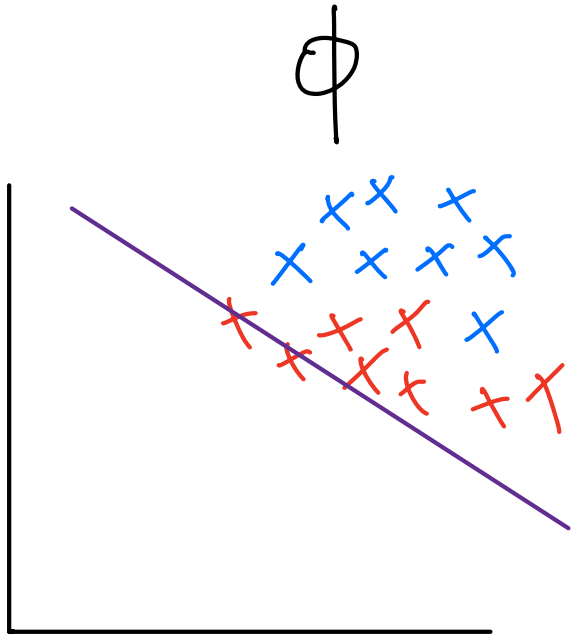
Internal covariate shift

Simple network

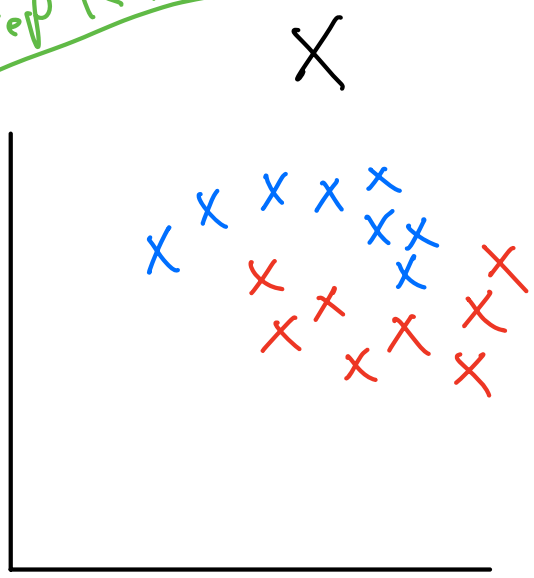


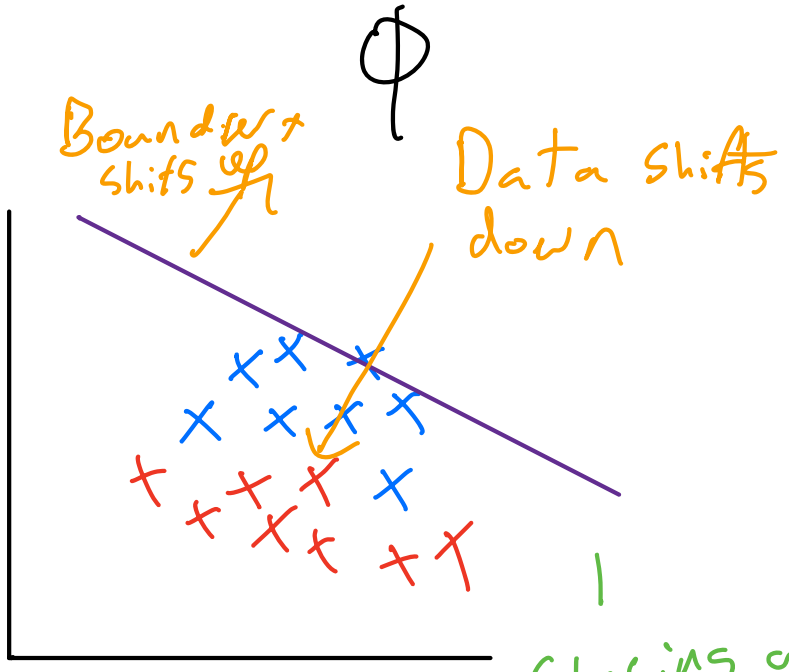
Step K





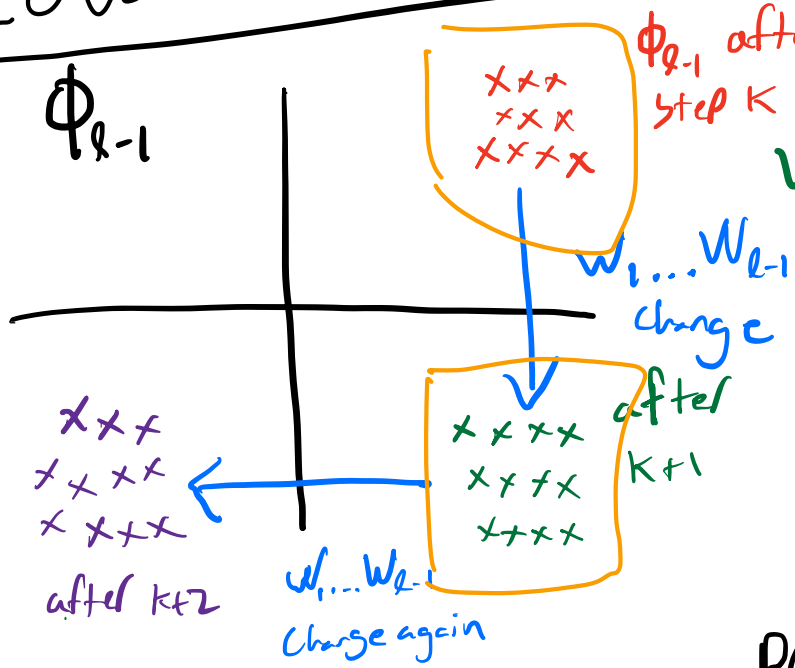
Step K+1





Chasing a moving target!

Covariate Shift



Layer ℓ

$$\phi_\ell = \sigma(\phi_{\ell-1} W_\ell + b_\ell)$$

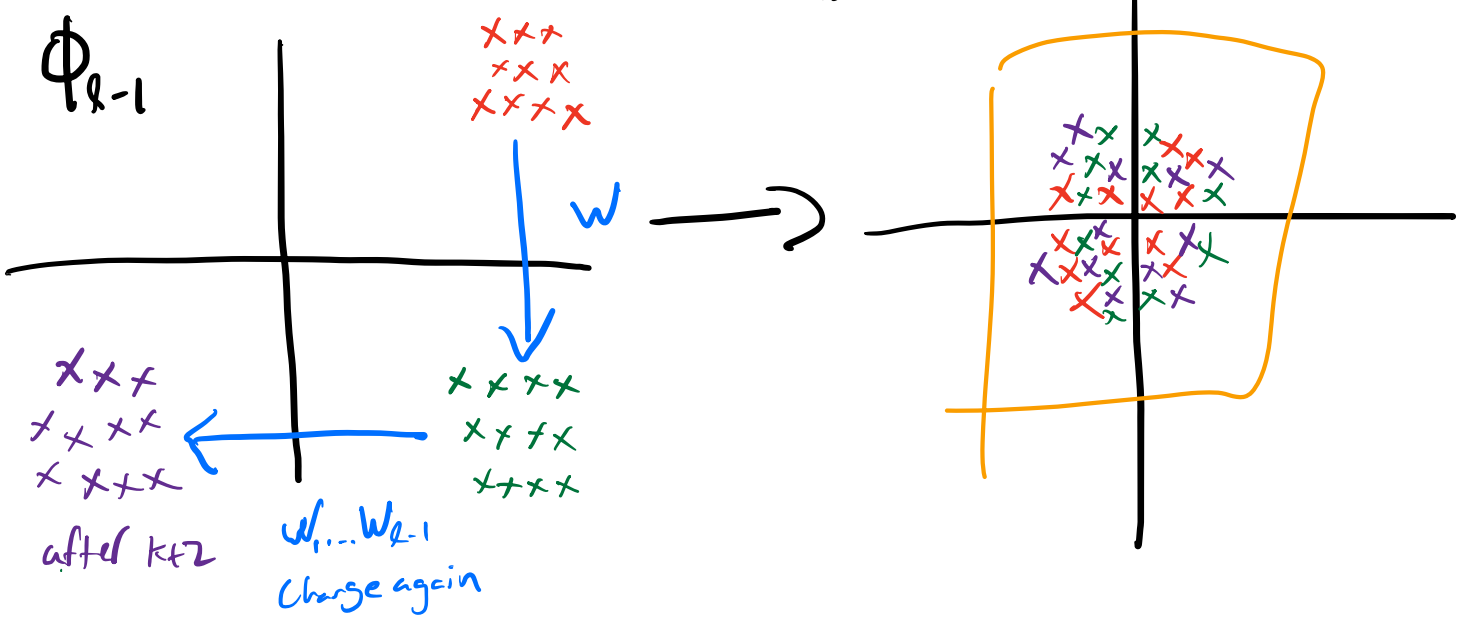
$$W_\ell^{(k+1)} \leftarrow W_\ell^{(k)} - \alpha \phi_{\ell-1}^{(k)} \sigma'\left(\phi_{\ell-1}^{(k)} W_\ell^{(k)} + b_\ell\right)$$

update using old $\phi_{\ell-1}^{(k)}$

$$f(x) = \sigma\left(\phi_{\ell-1}^{(k+1)} W_\ell^{(k+1)} + b_\ell\right) \dots$$

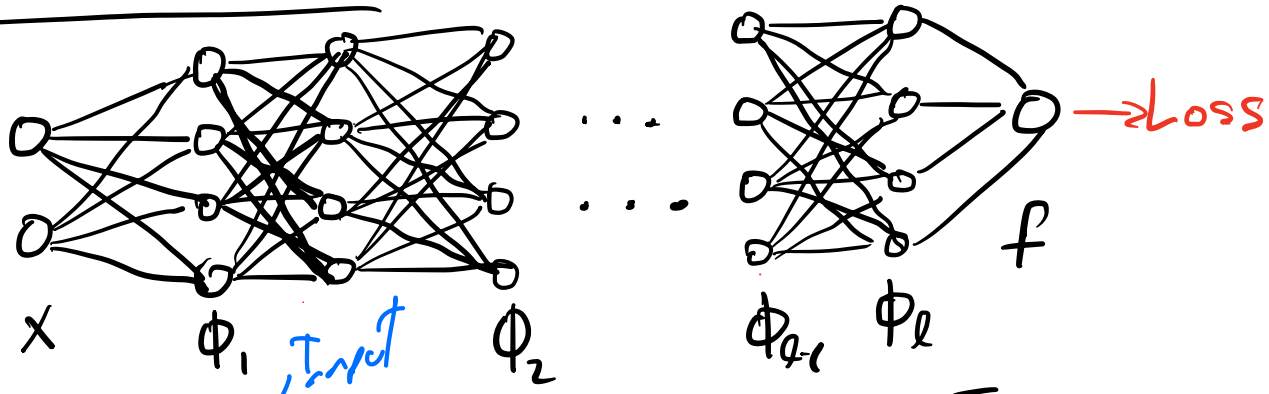
Predict using new $\phi_{\ell-1}^{(k+1)}$

Batch Norm



Can apply BatchNorm IN the network

Deep Network w/ Batch Norm!



Input

or

$$\phi_1 = \sigma(BN(x)^T W_1 + b_1)$$

$$\phi_2 = \sigma(BN(\phi_1)^T W_2 + b_2)$$

↑ output of $(\ell-1)^{st}$ layer

$$\vdots$$

$$\phi_\ell = \sigma(BN(\phi_{\ell-1})^T W_\ell + b_\ell)$$

$$f = BN(\phi_\ell)^T W_0 + b_0$$

$$L = Loss(f, y)$$

↓

e.g. $(f-y)^2$

$E[x] = 0$ $Var[x] = 1$ w/ Norm.

$E[BN(\phi_{\ell-1})] = 0$ $Var[BN(\phi_{\ell-1})] = 1$

Deep Network w/ Batch Norm

$E[BN(\phi_{l-1})] = 0 \quad Var[BN(\phi_{l-1})] = 1$

$E[\phi_{l-1}]$, $Var[\phi_{l-1}]$ still change every step

∴ We need to use Batch Norm and update $\bar{x} \approx E[x]$ and $s^2 \approx Var[x]$ at every step

($E[\phi_{l-1}]$ for layer l)

Test time

Problem: At test time we might want to make a prediction on a single obs.

Single obs. → $B = 1$

$\bar{x} = \frac{1}{1} x_1$ $\quad s^2 = \frac{1}{1-1}(x_1 - \bar{x})^2$

$x_1 - \bar{x} = 0$! $\quad$ $= 0 \rightarrow$ divide by 0!

Batch Norm at test time

In practice: Maintain Running Average of mean and variance estimates as we go. (using EMA!)

At step k: $\bar{X}^{(k)}_{Avg.} \leftarrow \beta \bar{X}^{(k-1)}_{Avg.} + (1-\beta)\bar{X}^{(k)}$ $\beta \in [0,1]$ typically ≥ 0.9

Est. using current batch

$S^{2\,(k)}_{Avg.} \leftarrow \beta S^{2\,(k-1)}_{Avg.} + (1-\beta) S^{2\,(k)}$

At test:

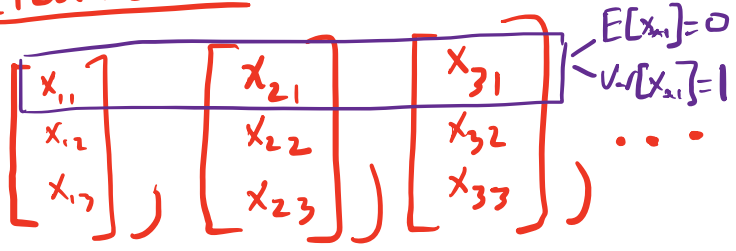
$$\underset{\text{test}}{\text{BatchNorm}}(x) = \frac{x - \bar{X}_{Avg.}}{\sqrt{S^2_{Avg.} + \varepsilon}}$$

Use the same estimates every time we evaluate

LayerNorm

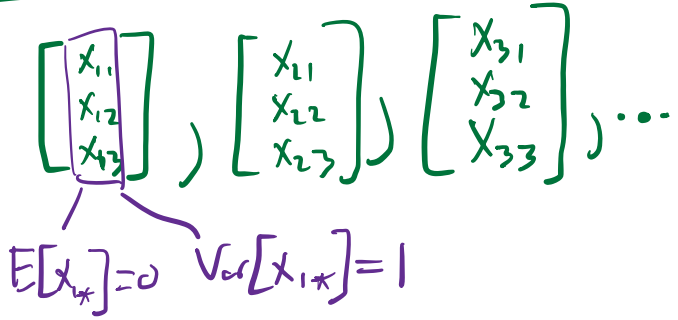
What if we don't want to do separate things at train and test time?

Batch norm



Find $\bar{x} \approx E[x]$, $s \approx Var[x]$ by Averaging over observations for each dimension

LayerNorm



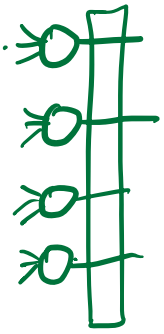
Alt. Average over dimensions within an observation

Layer Norm

$$\text{LayerNorm}(x_i) = \frac{x_i - \bar{x}_i}{\sqrt{s_i^2 + \varepsilon}}$$

$$\bar{x}_i = \frac{1}{d}\sum_{j=1}^{d} x_{ij} \qquad s^2 = \frac{1}{d-1}\sum_{j=1}^{d}(x_{ij} - \bar{x}_i)^2$$

In network



$E[x] = 0, \text{Var}[x] = 1$

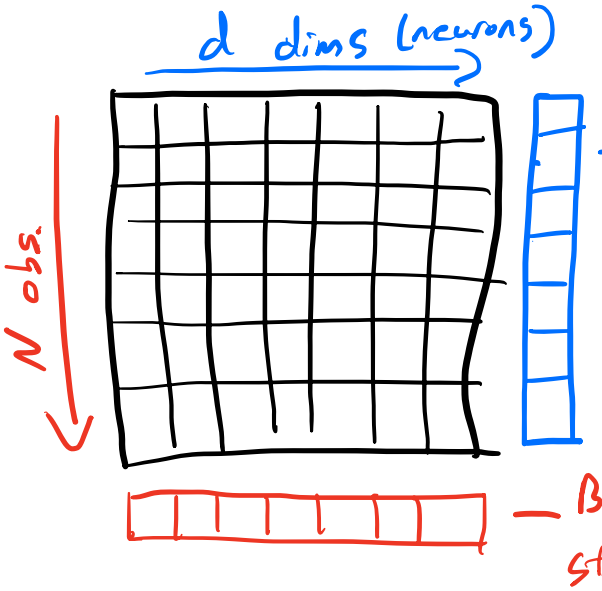
Can be applied w/ Batch size 1! As long as $d > 1$

Batch Norm Vs. Layer Norm

Input matrix

X

$N \times d$



Layer norm statistics per row

Batch norm statistics per Column

Batch Norm Usually preferred, but not always possible

[preserves more info about each observation]

Batch Norm caveats

Covariate shift explanation is controversial!

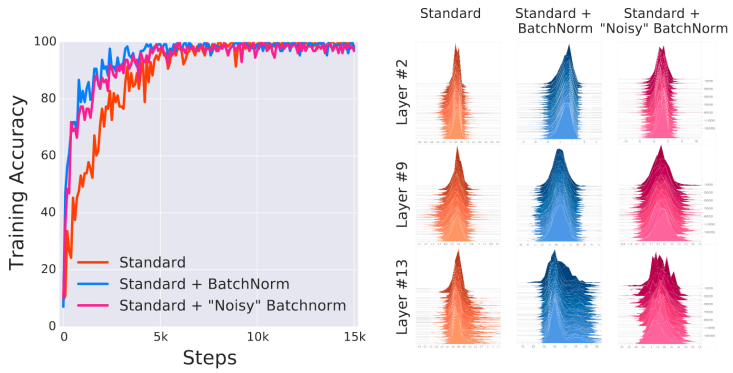


Figure 2: Connections between distributional stability and BatchNorm performance: We compare VGG networks trained without BatchNorm (Standard), with BatchNorm (Standard + BatchNorm) and with explicit "covariate shift" added to BatchNorm layers (Standard + "Noisy" BatchNorm). In the later case, we induce distributional instability by adding *time-varying*, *non-zero* mean and *non-unit* variance noise independently to each batch normalized activation. The "noisy" BatchNorm model nearly matches the performance of standard BatchNorm model, despite complete distributional instability. We sampled activations of a given layer and visualized their distributions (also cf. Figure 7).

Alternative Claim: Smooths Loss Landscape

(a) loss landscape (b) gradient predictiveness (c) "effective" β-smoothness

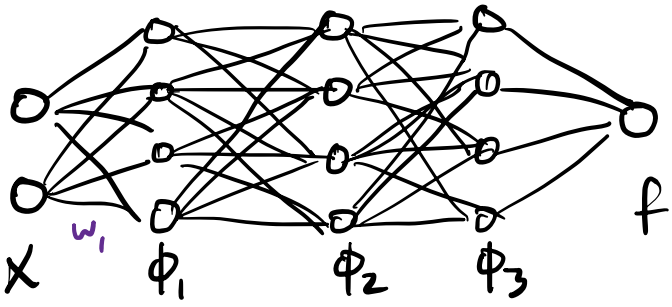
Figure 4: Analysis of the optimization landscape of VGG networks. At a particular training step, we measure the variation (shaded region) in loss (a) and ℓ_2 changes in the gradient (b) as we move in the gradient direction. The "effective" β-smoothness (c) refers to the maximum difference (in ℓ_2-norm) in gradient over distance moved in that direction. There is a clear improvement in all of these measures in networks with BatchNorm, indicating a more well-behaved loss landscape. (Here, we cap the maximum distance to be $\eta = 0.4\times$ the gradient since for larger steps the standard network just performs worse (see Figure 1). BatchNorm however continues to provide smoothing for even larger distances.) Note that these results are supported by our theoretical findings (Section 4).

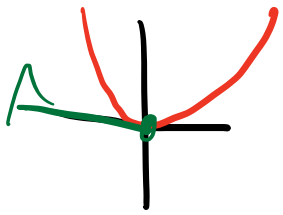
Some debate over orderings

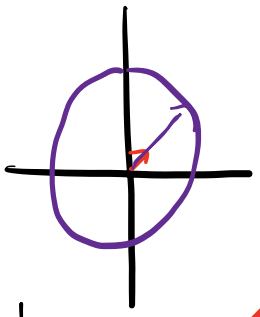
$\sigma(BN(x)W+b)$ vs. $\sigma(BN(Wx+b))$

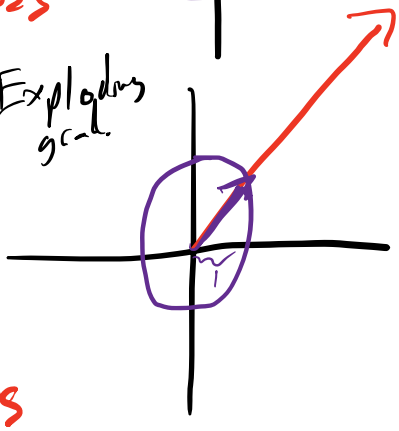
↳ More common these days

Residual Networks

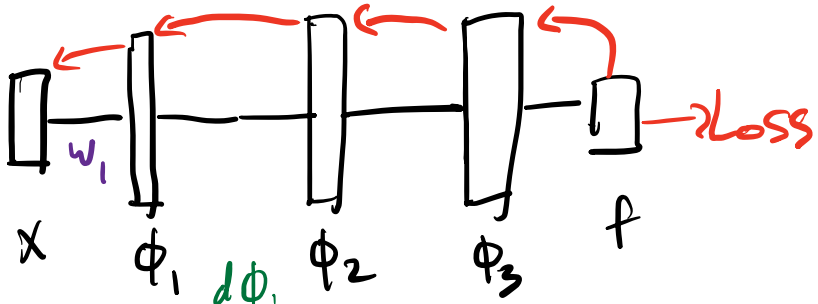








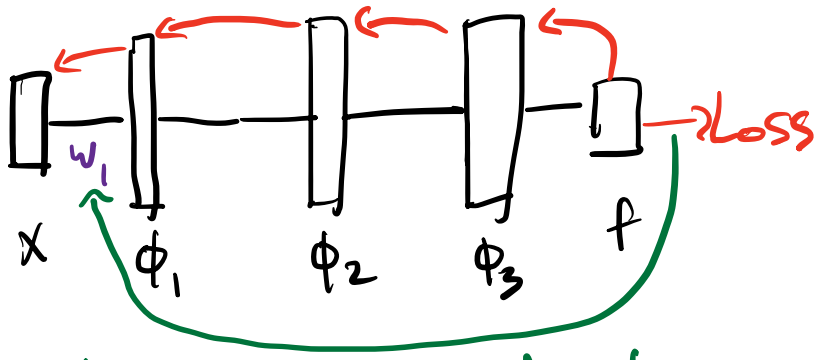
Simplified



$$\frac{dL}{dw_1} = x\sigma'(x^T w_1)\frac{d\phi_1}{dw_1}\frac{d\phi_2}{d\phi_1}\cdot\frac{d\phi_3}{d\phi_2}\cdots\frac{df}{d\phi_k}\cdot\frac{dLoss}{df}$$

Info from Loss gets diluted (Vanishing gradients)

Residual Networks



What if we had a direct connection?

$$\frac{dL}{dw_1} = x\sigma'(x^T w_1)\underbrace{\frac{d\phi_2}{d\phi_1}\cdot\frac{d\phi_3}{d\phi_2}\cdots\frac{df}{d\phi_n}}\cdot\frac{dLoss}{df} + \underbrace{\frac{dL}{d\phi_1}\cdot\frac{d\phi_1}{dw_1}}$$

$\left|\frac{d\phi_i}{d\phi_{i-1}}\right| < 1 \rightarrow 0$

Loss depends directly on ϕ_1

Residual function

$$\hat{f}(x) \rightarrow f(x) = \hat{f}(x) + x$$

Add input back to output!

Residual Layer

Layer: $\phi_i = \sigma(\phi_{i-1}^T w_i + b_i)$

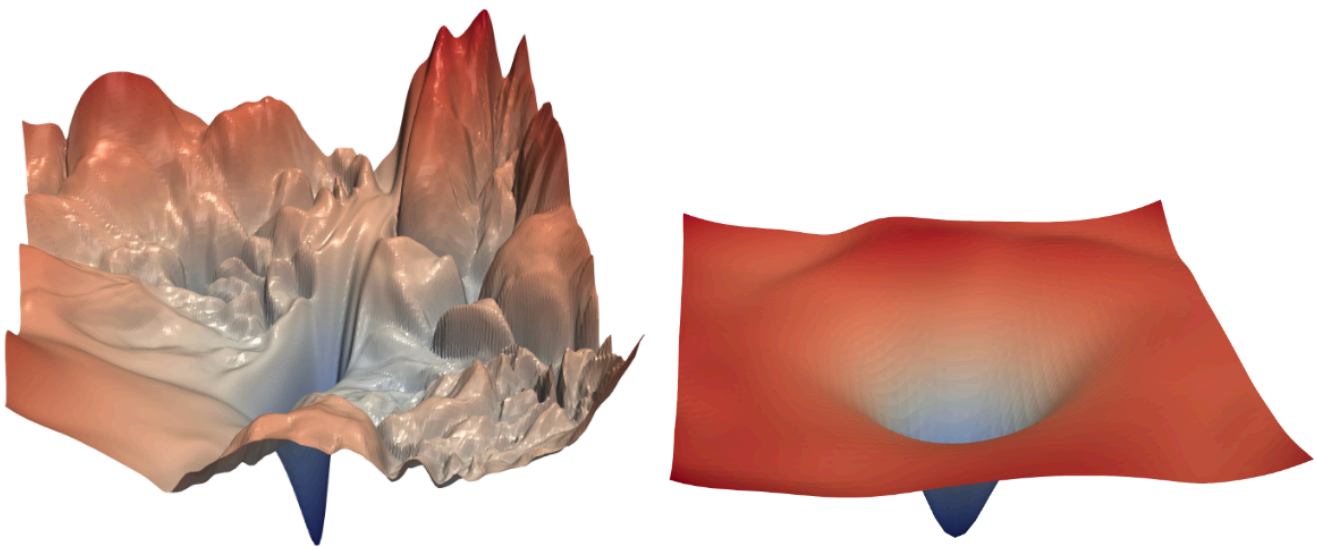
Next value

Residual Layer: $\phi_i = \sigma(\phi_{i-1}^T w_i + b_i) + \phi_{i-1}$

$$\frac{d\phi_i}{d\phi_{i-1}} = \frac{d}{d\phi_{i-1}}\left[\sigma(\phi_{i-1}^T w_i + b_i) + \phi_{i-1}\right]$$

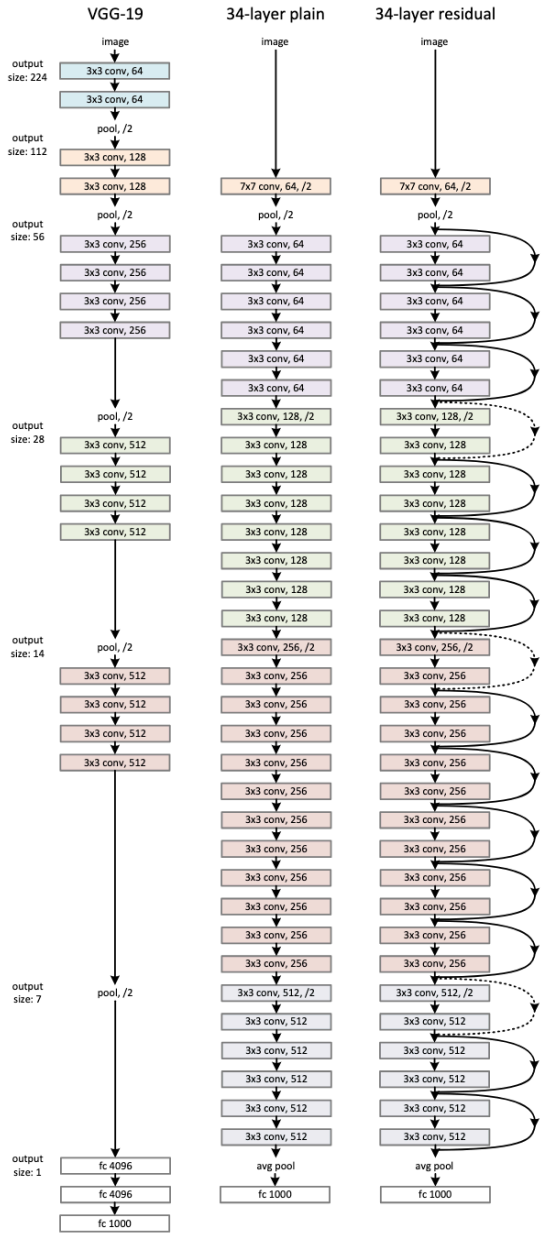
$$= \sigma'(\phi_{i-1}^T w_i + b_i) w_i + 1$$

$$\frac{d\phi_i}{d\phi_{i-1}} = \frac{d\hat{\phi}_i}{d\phi_{i-1}} + 1$$



(a) without skip connections (b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.



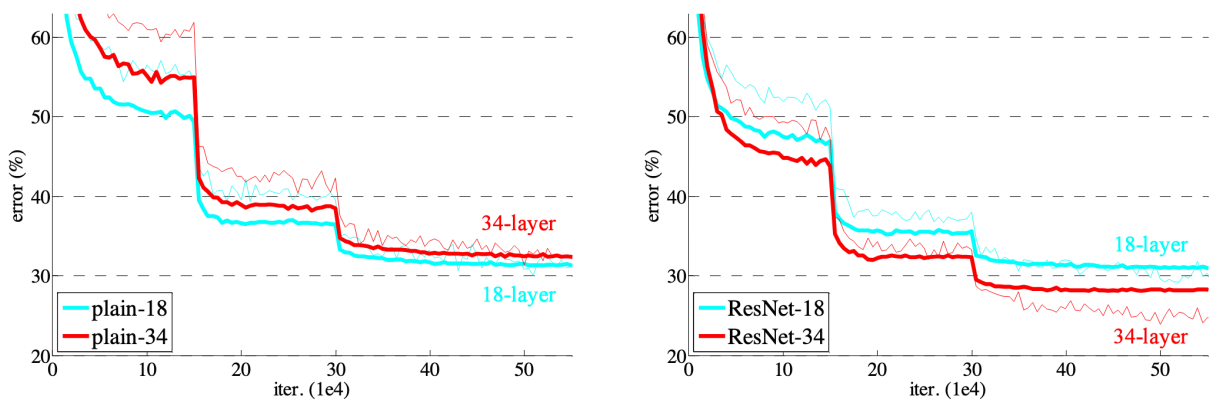


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.