How Does Batch Normalization Help Optimization?

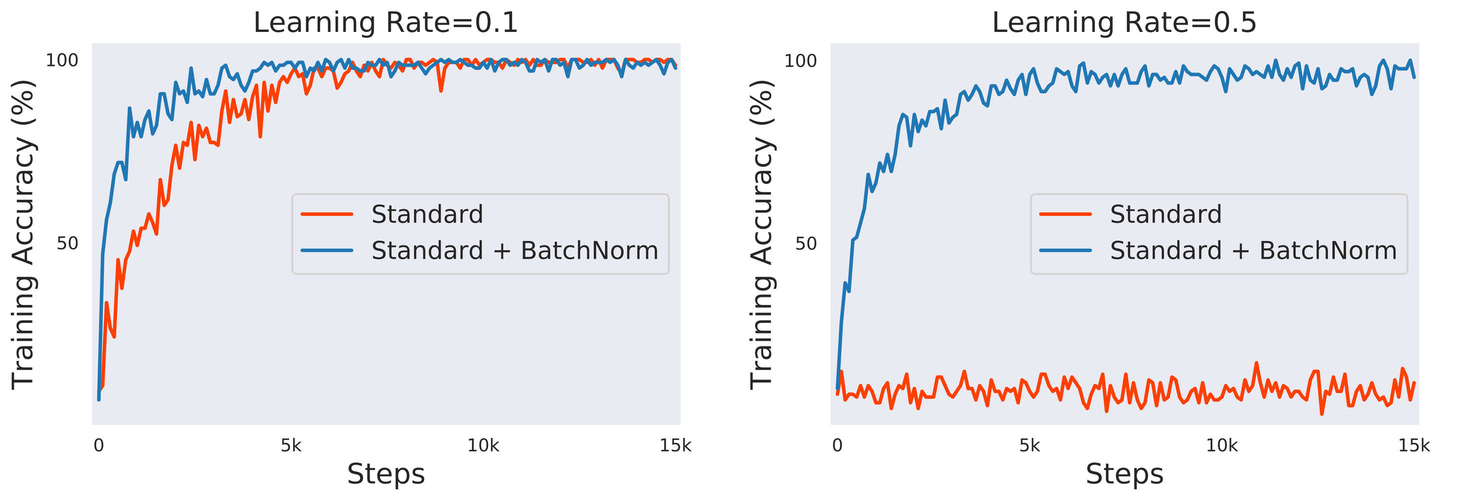
Shibani Santurkar*, Dimitris Tsipras*, Andrew Ilyas*, Aleksander Madry

Massachusetts Institute of Technology

madry-lab.ml

Batch Normalization (BatchNorm)

Faster Convergence — Robust to Hyperparameter Choice



⇒ Used almost by default in most architectures (7k+ citations)

How does BatchNorm help training?

Why does BatchNorm work?

Reducing Internal Covariate Shift (ICS) by normalizing activations

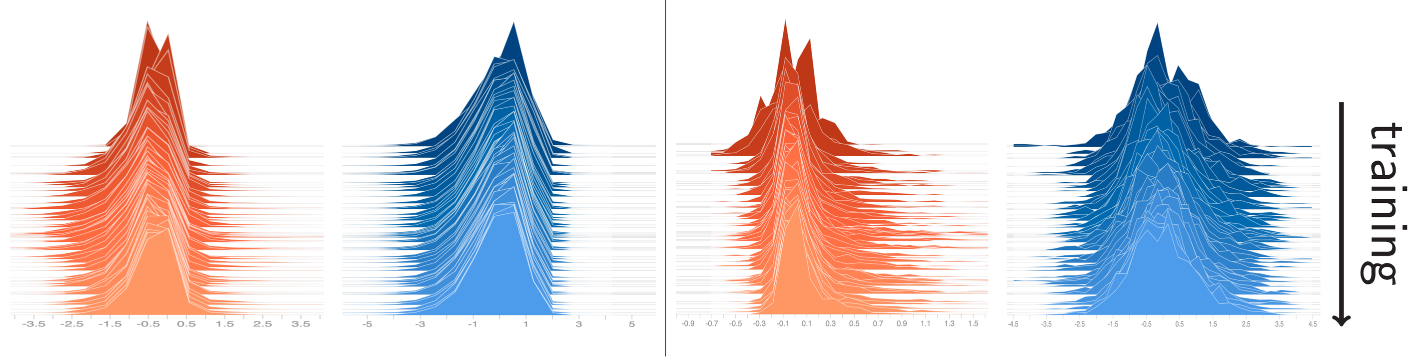
> [When training deep models, the input distribution of each layer changes over time.] The change in the distributions of layers' inputs presents a problem because the layers need to continuously adapt to the new distribution.
>
> **[Ioffe, Szegedy 2015]**

But: Is that really what happens?

A closer look at activation distributions

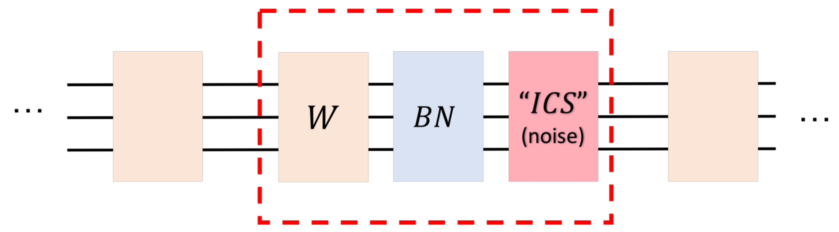
Layer inputs over training:

Layer #3: Standard, +BatchNorm — Layer #11: Standard, +BatchNorm

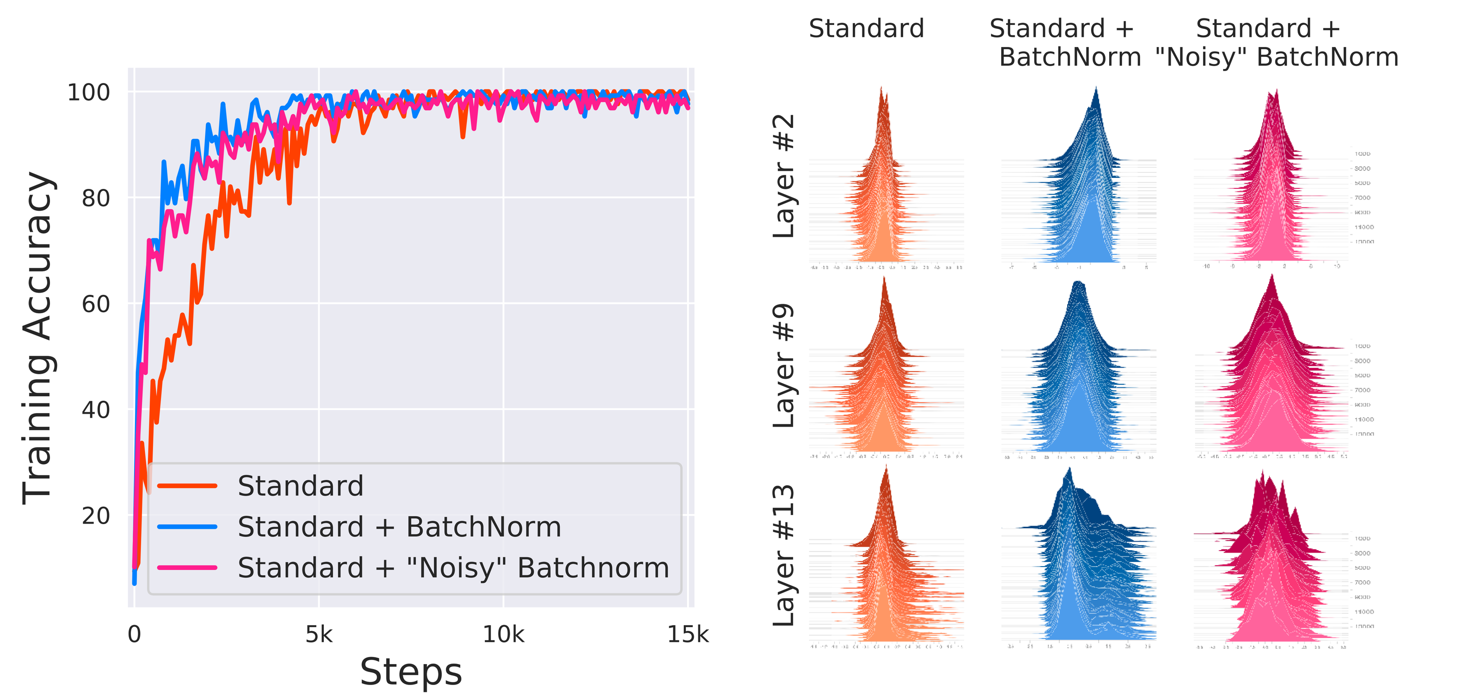


⇒ No apparent difference between models with and without BN

What if we introduce additional (artificial) ICS?



Specifically: We add **time-varying** noise (with **non-zero mean**) to the **outputs** of BatchNorm layers



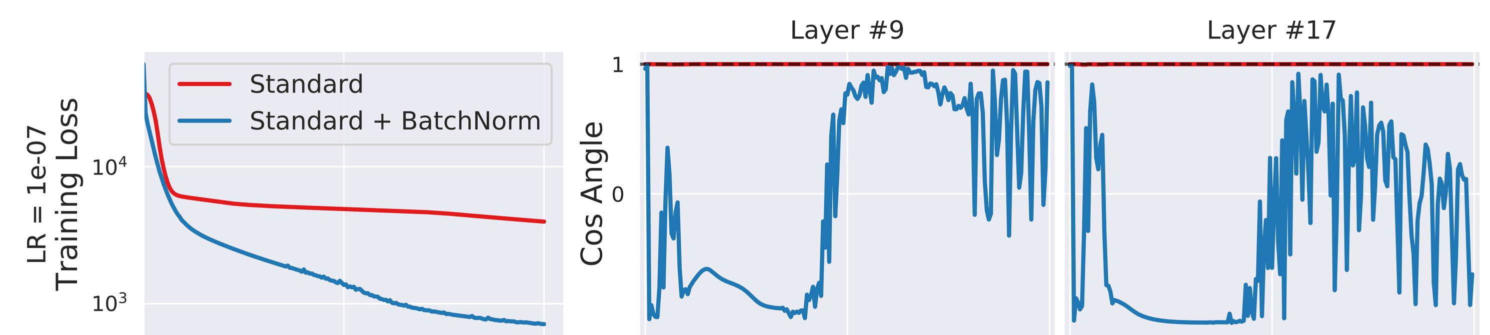
Result: Increased instability, yet **no** apparent decrease in performance

⇒ Stability and performance seem to **not** be strongly connected

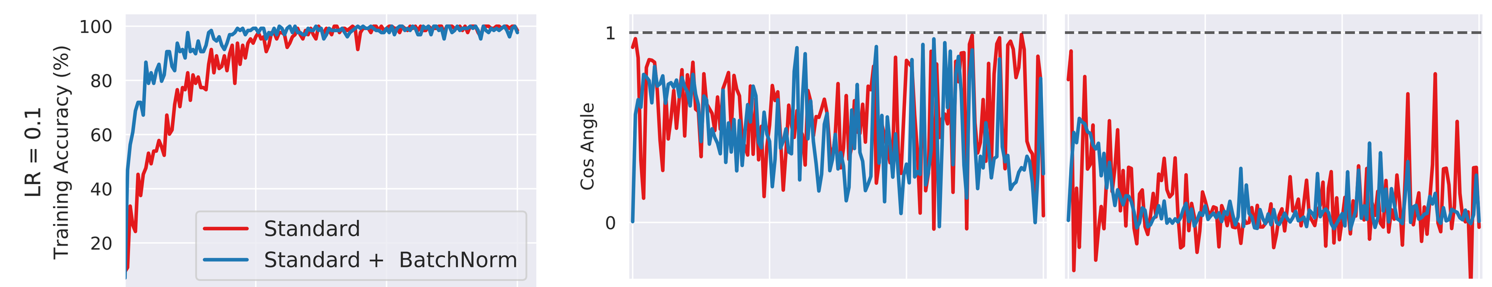
An optimization-based notion of ICS?

Idea: Measure **change in gradient** due to **previous layer updates**

Deep Linear Network



VGG



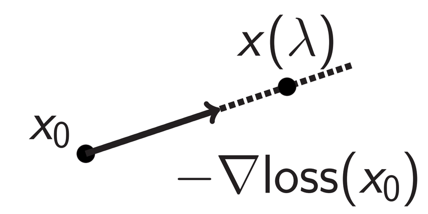
We observe:

→ In deep linear models there is essentially no such change altogether

→ In VGG networks, the changes caused by the updates to previous layers are similar for both standard and batch normalized networks

Roots of BatchNorm's success

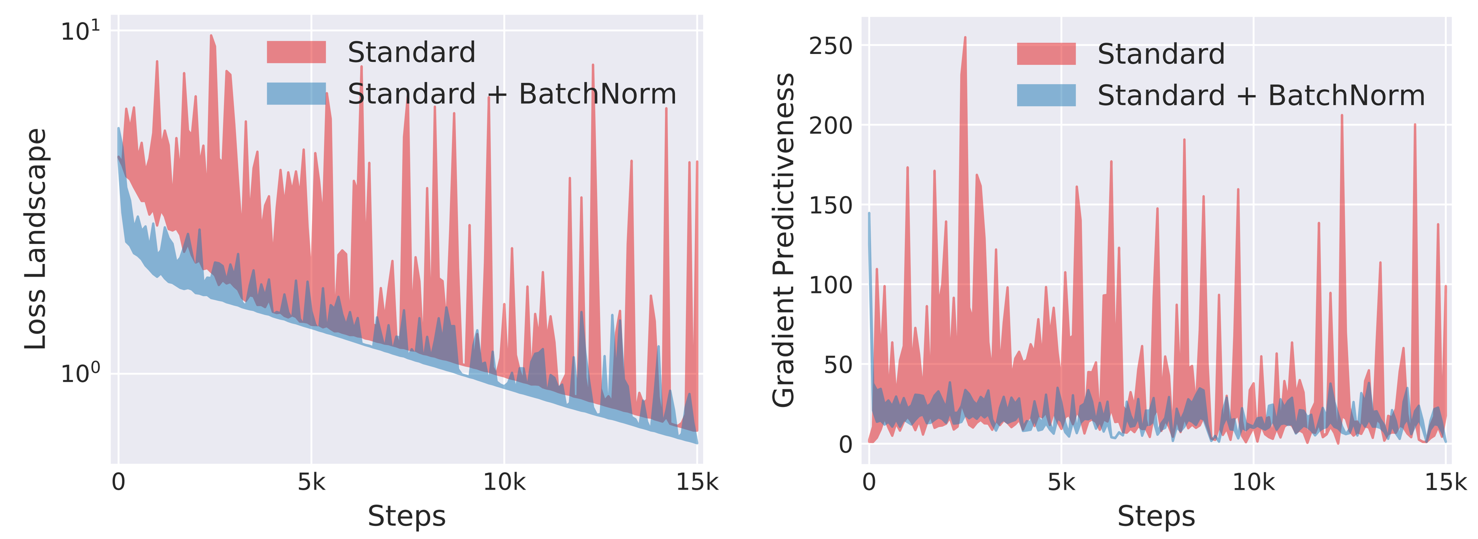
Our approach: Examine the loss and gradient landscape



$$x(\lambda) = x_0 - \lambda \nabla \text{loss}(x_0)$$

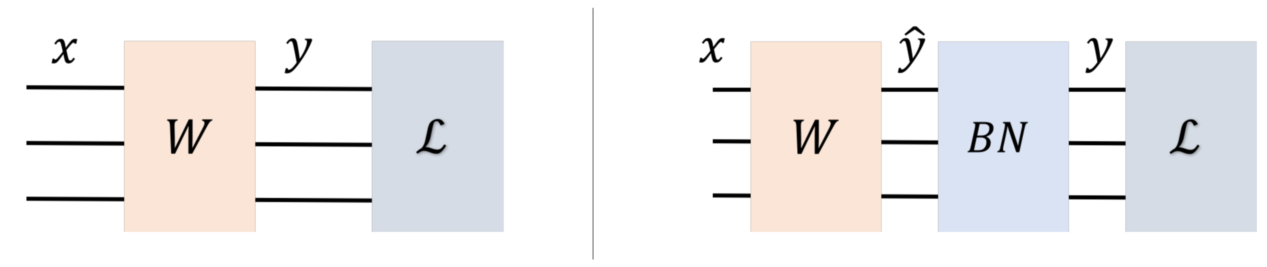
Specifically: Measure variation of loss and gradient over λ

$\text{loss}(x(\lambda))$ — $\|\nabla \text{loss}(x(\lambda)) - \nabla \text{loss}(x_0)\|_2$



⇒ Loss and gradients significantly better behaved for BatchNorm

Impact of adding a BatchNorm layer



We show:

⇒ Loss is **provably** more Lipschitz wrt y

⇒ Gradients wrt y are **provably** more predictive (and hence reliable)

⇒ Translates into similar **worst-case** improvements for W

Future directions

→ Better normalization schemes
(Normalizing by **other norms** offer similar improvements)

→ Understand BatchNorm's impact on generalization

→ **More broadly:** Study the other elements of our DL toolkit in depth

Full version at `arxiv:1805.11604`