"How does batch normalization help optimization?"

Shibani Santurka, Dimitris Tsipras, Andrew Ilyas & Aleksander Madry

MIT

NIPS 2018

Well… how does it?

???

"… because it reduces "internal covariate shift" (ICS), i.e., controlling and stabilizing the distributions fed into each layer."

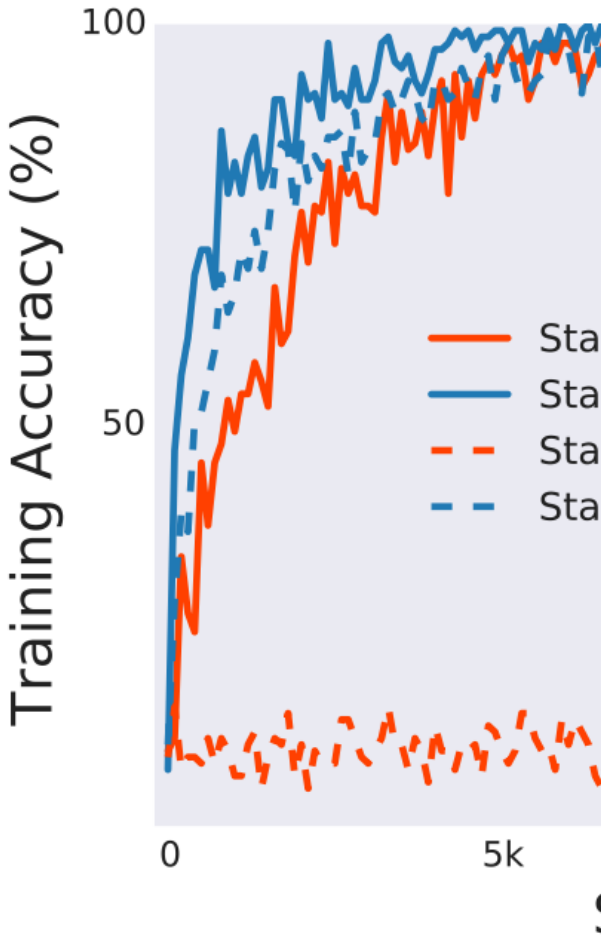
“Even though this explanation is widely accepted, we seem to have little concrete evidence supporting it. **The chief goal of this paper is to address this**.”

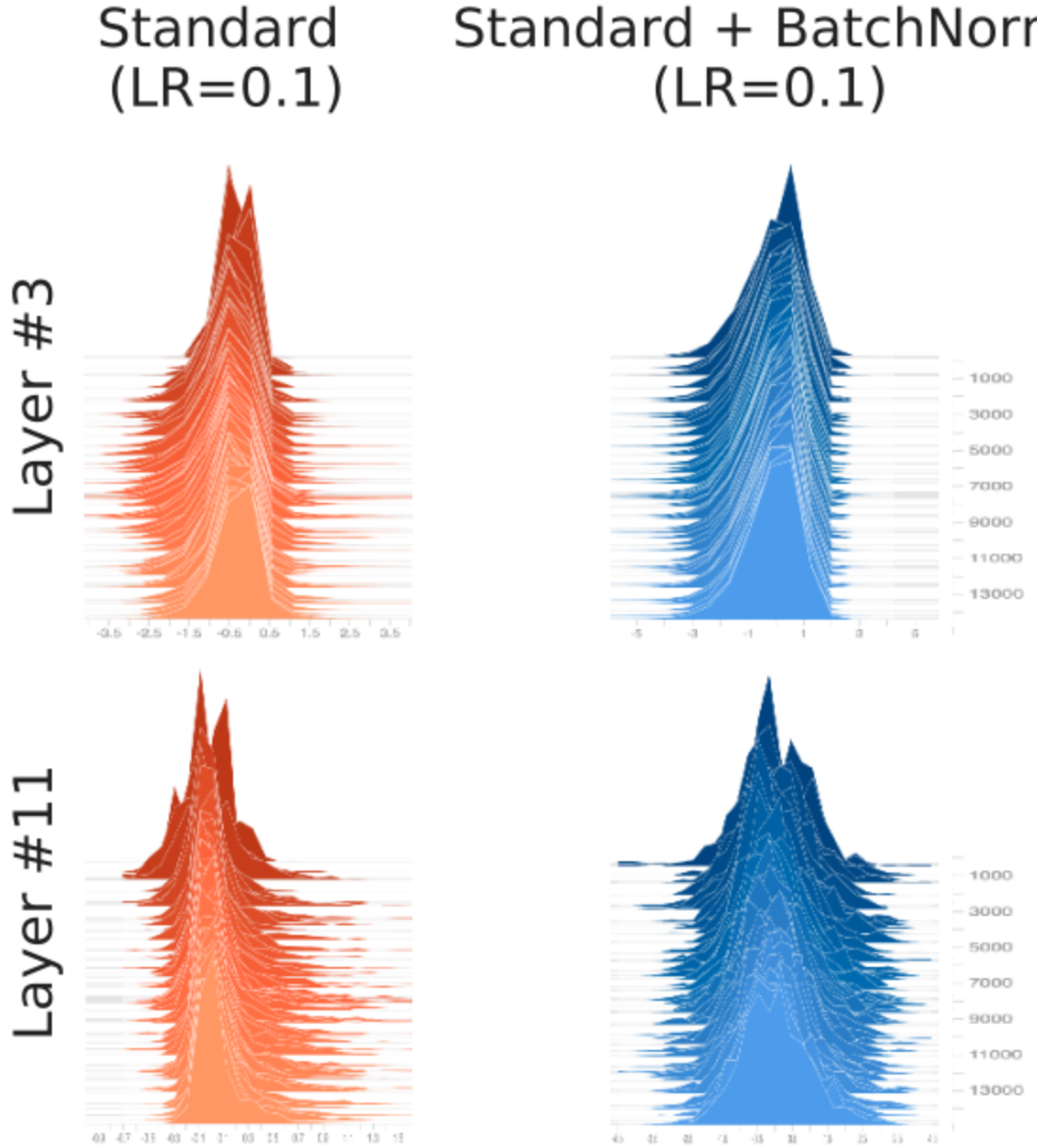
Author's Main Questions:

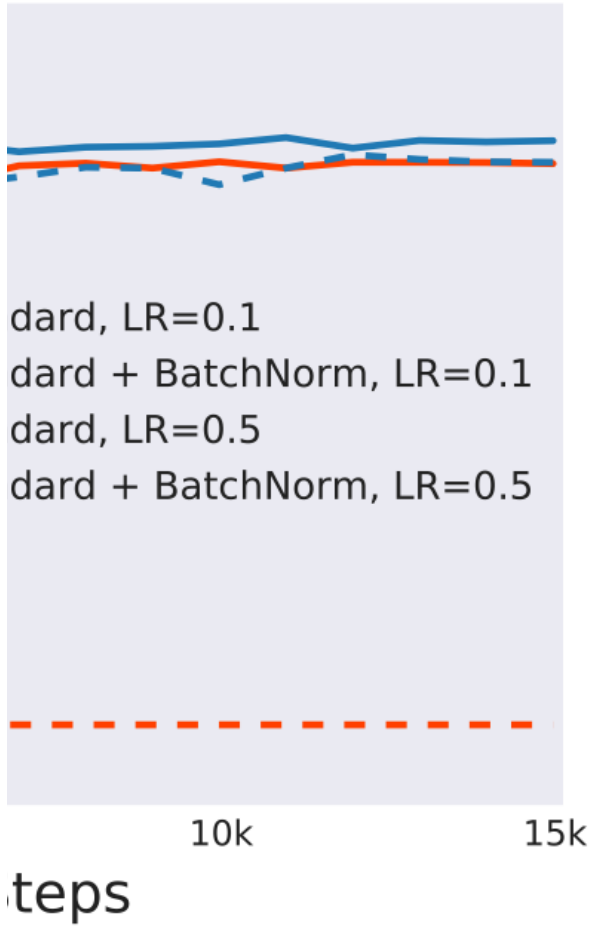
(1) "Is the effectiveness of BatchNorm indeed related to internal covariate shift?"

(2) "Is BatchNorm's stabilization of layer input distributions even effective in reducing ICS?"

VGG train



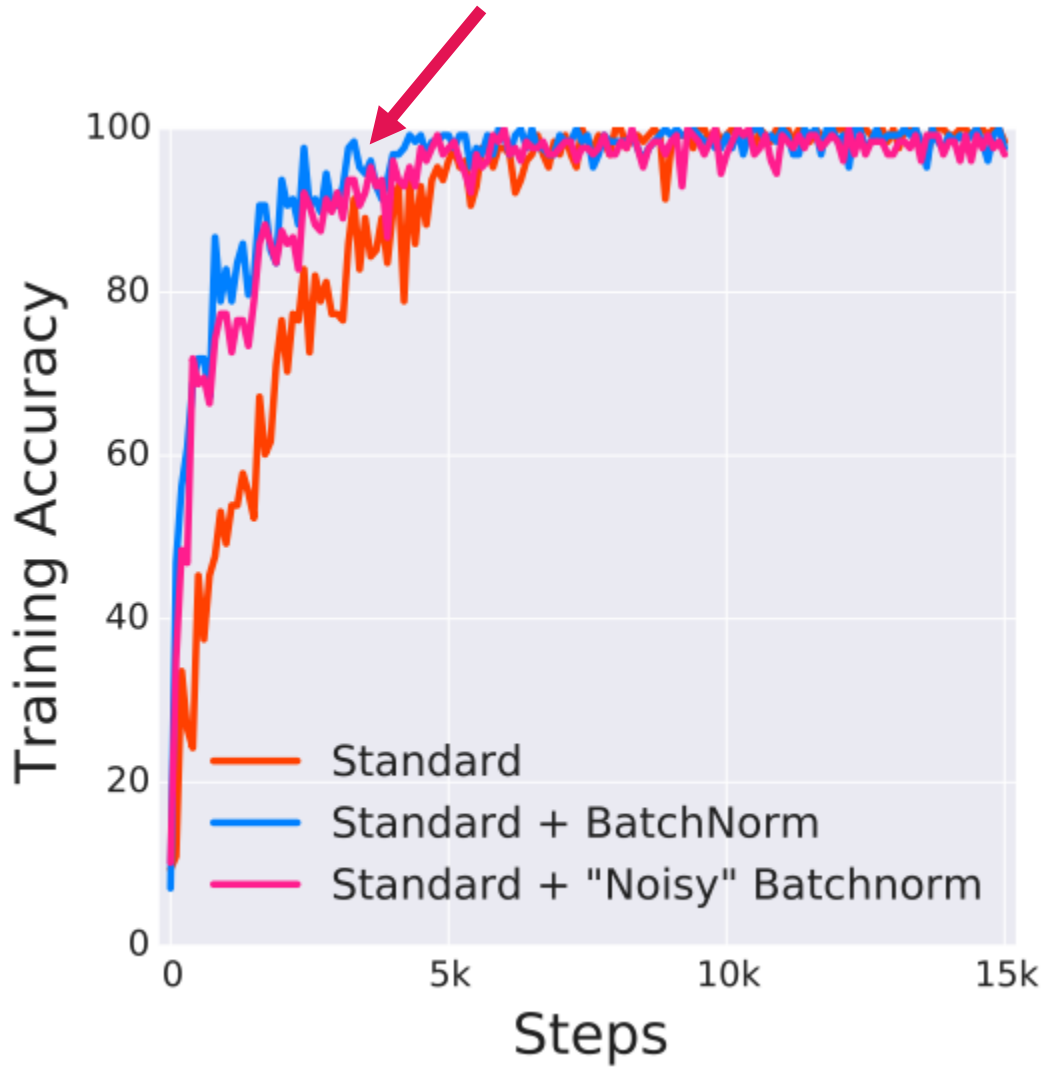


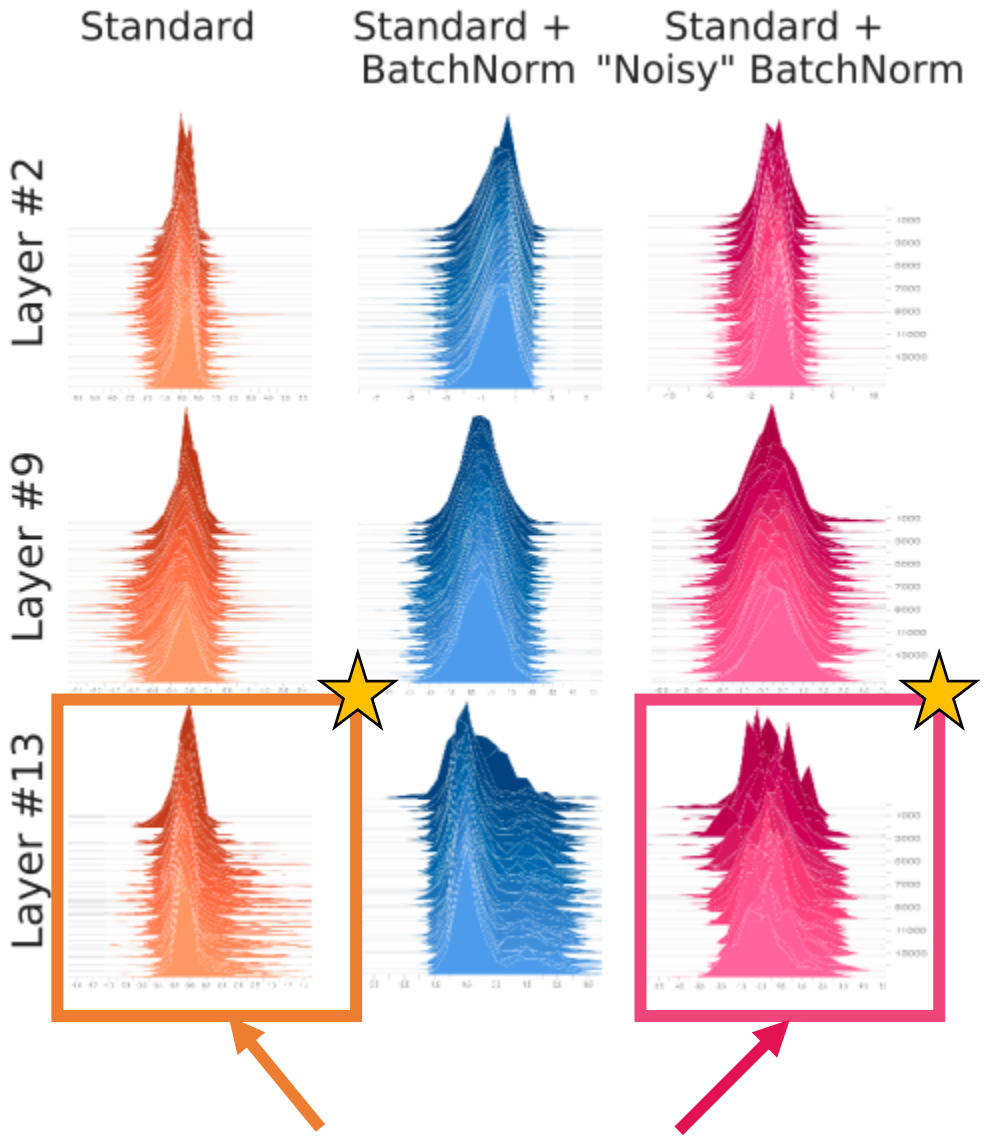


o BN)

teps

Standard + “Noisy” ≌ Standard + Batchnorm





(ICS): Standard << Standard + “Noisy” BN

"The difference between $\boldsymbol{G}$ and $\boldsymbol{G'}$ thus reflects the *change in the optimization landscape* of W^i caused by the changes to its input."

LR = 0.1
Training Accuracy (%)

"The conventional understanding of BatchNorm suggests that the addition of BatchNorm layers in the network should increase the correlation between $\boldsymbol{G}$ and $\boldsymbol{G'}$, thereby reducing ICS…"

Okay… then…

“WHY DOES BATCH NORM WORK?”

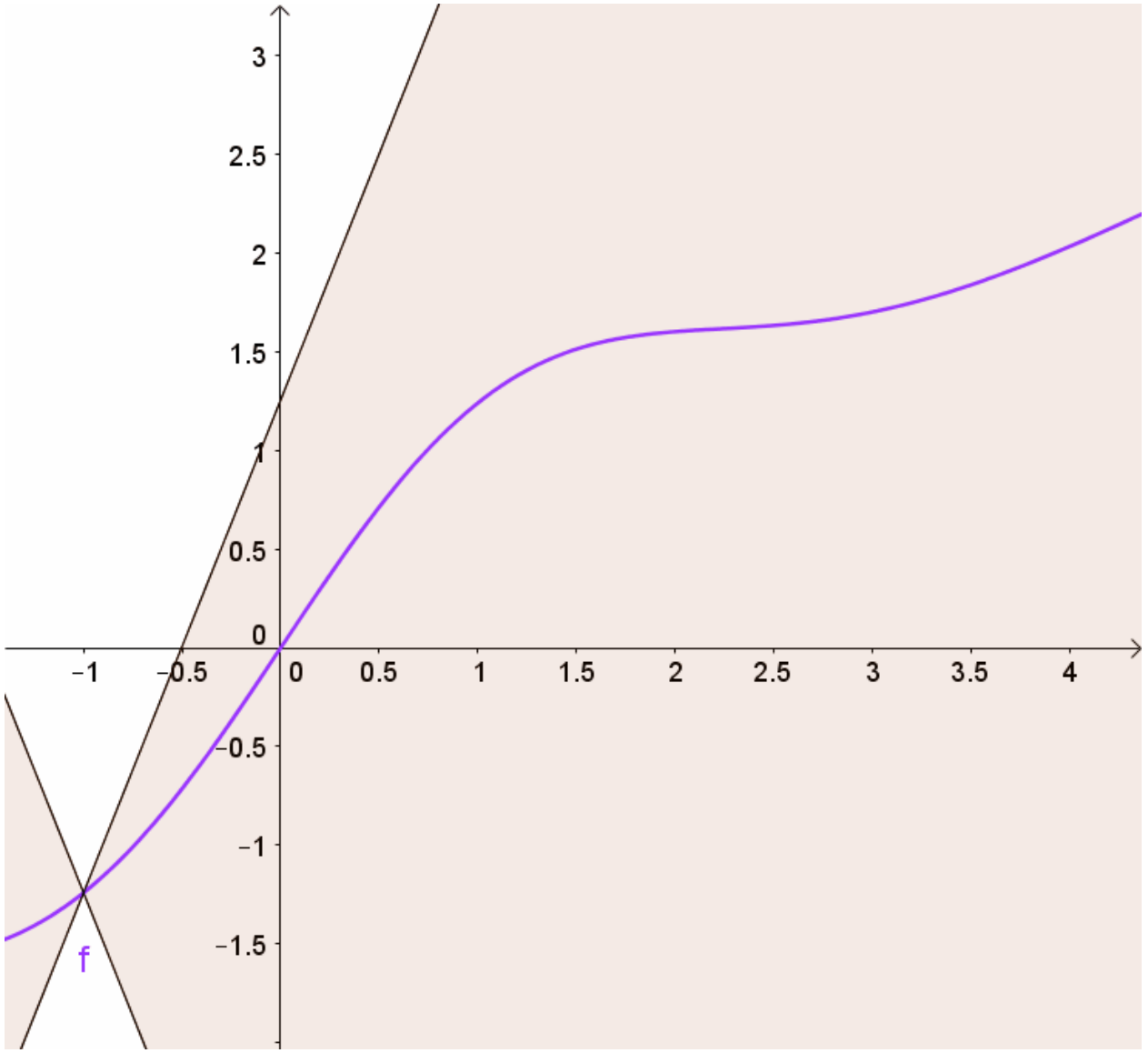
Authors: Smoothness is the key

“Indeed, we identify the key impact that BatchNorm has on the training process: it reparametrizes the underlying optimization problem to ***make its landscape significantly more smooth***.”

… *smoothness of the optimization landscape*…?

Lipschitzness

For a “**Lipschitz continuous**” function, there exists a double cone (white) whose origin can be moved along the graph so that the whole graph always stays outside the double cone



Why is “**higher Lipschitzness**” in both the loss and gradient of loss a good thing?

Implications of Higher Lipschitzness

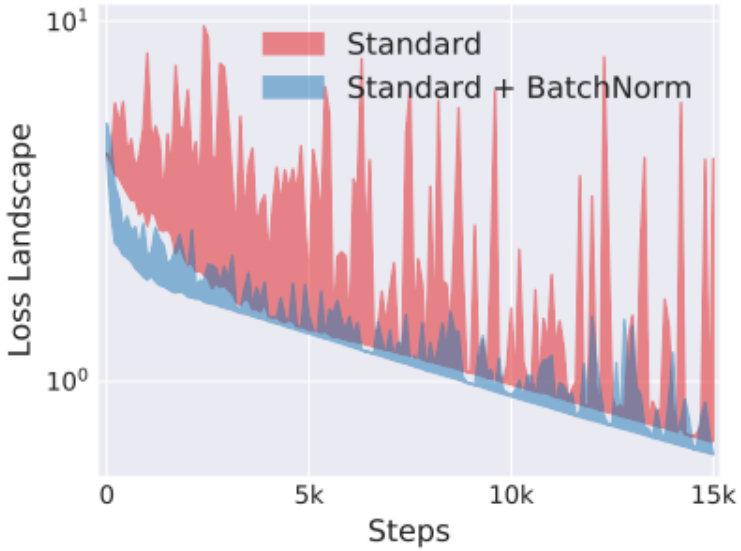
On loss: Exploding gradients, local minima and other “kinks” become much less of an issue.

On <u>gradient</u> of loss: “*Now, the key implication of BatchNorm’s reparametrization is that it makes the gradients more reliable and predictive*…”

“It thus enables any (gradient–based) training algorithm ***to take larger steps without the danger of running into a sudden change of the loss landscape*** *such as flat region (corresponding to vanishing gradient) or sharp local minimum (causing exploding gradients).*”

Quantifying Smoothness

Variation in Loss



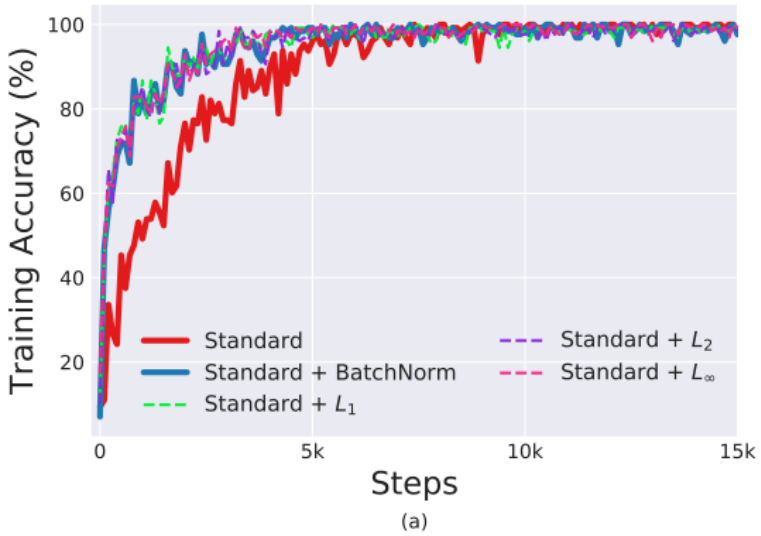
(a) loss landscape

Experiment:

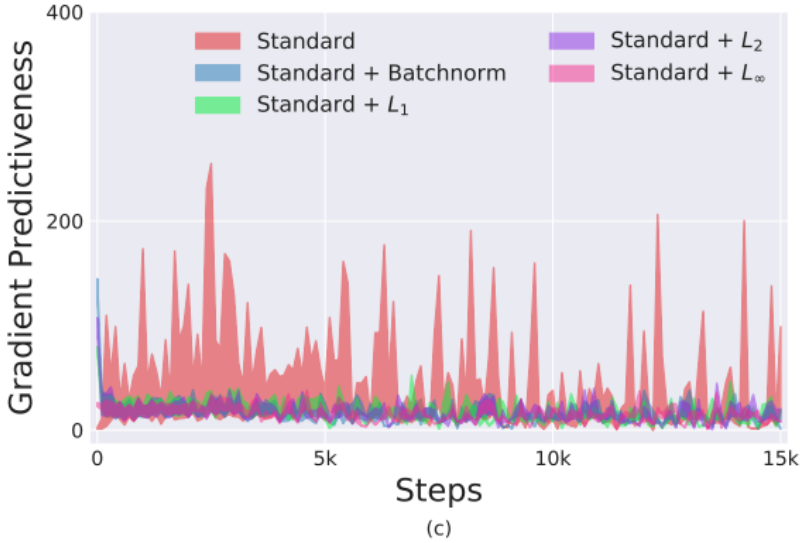
Train VGG by normalizinig each layer with its ℓ_p**–norm (**$p = 1, 2, \ldots$**)** *before* shifting the mean. ***<u>These distributions are no longer gaussian.</u>***

Compare with BN.

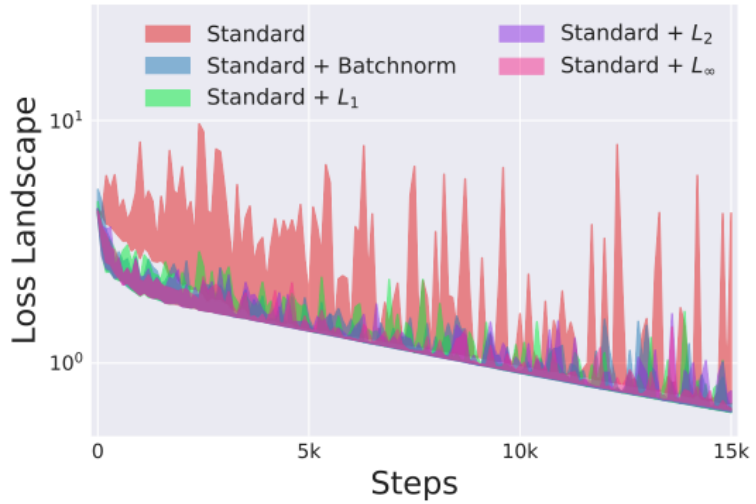
VGG Trained w/ Different Normalization Schemes

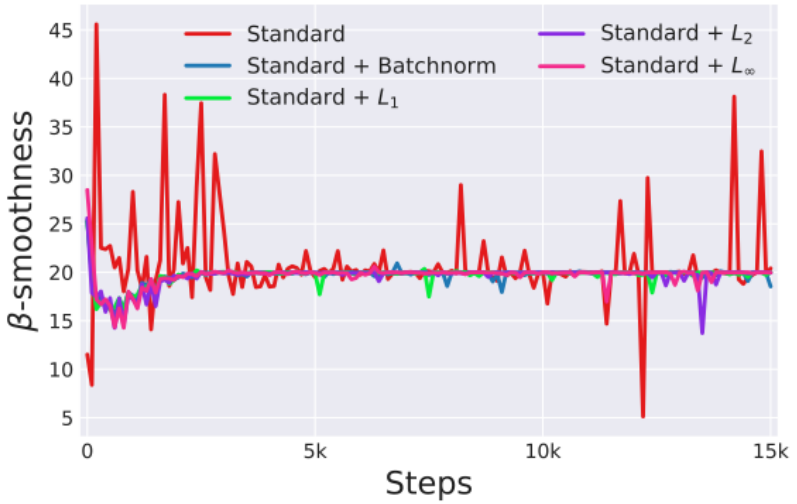


(a)



(c)





VGG Trained w/ Different Normalization Schemes

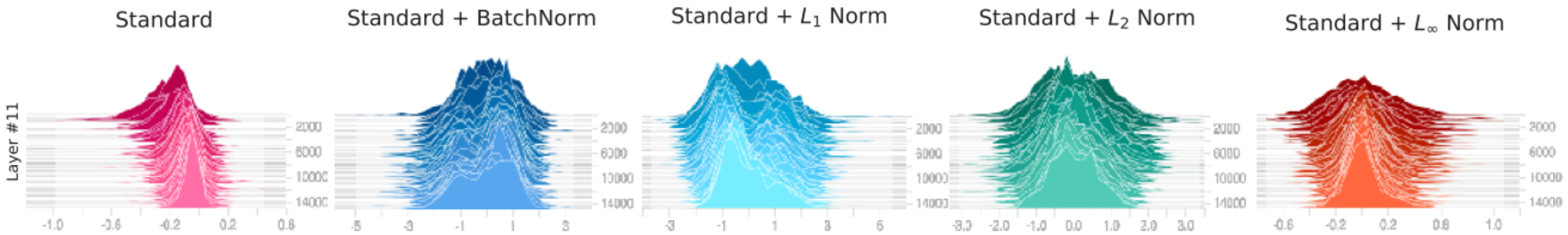


Figure 14: Activation histograms for the VGG network under different normalizations. Here, we randomly sample activations from a given layer and visualize their distributions. Note that the ℓ_p-normalization techniques leads to larger distributional covariate shift compared to normal networks, yet yield improved optimization performance.

Other non-gaussian normalization methods demonstrated improved performance over BN.

As long as the normalization scheme used results in increased smoothness of the loss landscape… it seems more than likely that BN is *not* the "**best**" normalization paradigm.