of one of the lower weights can flip the relationship between $\hat{h}_{l-1}$ and y. These situations are very rare. Without normalization, nearly every update would have an extreme effect on the statistics of h_{l-1}. Batch normalization has thus made this model significantly easier to learn. In this example, the ease of learning of course came at the cost of making the lower layers useless. In our linear example, the lower layers no longer have any harmful effect, but they also no longer have any beneficial effect. This is because we have normalized out the first and second order statistics, which is all that a linear network can influence. In a deep neural network with nonlinear activation functions, the lower layers can perform nonlinear transformations of the data, so they remain useful. Batch normalization acts to standardize only the mean and variance of each unit in order to stabilize learning, but allows the relationships between units and the nonlinear statistics of a single unit to change.

Because the final layer of the network is able to learn a linear transformation, we may actually wish to remove all linear relationships between units within a layer. Indeed, this is the approach taken by Desjardins *et al.* (2015), who provided the inspiration for batch normalization. Unfortunately, eliminating all linear interactions is much more expensive than standardizing the mean and standard deviation of each individual unit, and so far batch normalization remains the most practical approach.

Normalizing the mean and standard deviation of a unit can reduce the expressive power of the neural network containing that unit. In order to maintain the expressive power of the network, it is common to replace the batch of hidden unit activations $\boldsymbol{H}$ with $\boldsymbol{\gamma H'} + \boldsymbol{\beta}$ rather than simply the normalized $\boldsymbol{H'}$. The variables $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are learned parameters that allow the new variable to have any mean and standard deviation. At first glance, this may seem useless—why did we set the mean to $\mathbf{0}$, and then introduce a parameter that allows it to be set back to any arbitrary value $\boldsymbol{\beta}$? The answer is that the new parametrization can represent the same family of functions of the input as the old parametrization, but the new parametrization has different learning dynamics. In the old parametrization, the mean of $\boldsymbol{H}$ was determined by a complicated interaction between the parameters in the layers below $\boldsymbol{H}$. In the new parametrization, the mean of $\boldsymbol{\gamma H'} + \boldsymbol{\beta}$ is determined solely by $\boldsymbol{\beta}$. The new parametrization is much easier to learn with gradient descent.

Most neural network layers take the form of $\phi(\boldsymbol{XW} + \boldsymbol{b})$ where ϕ is some fixed nonlinear activation function such as the rectified linear transformation. It is natural to wonder whether we should apply batch normalization to the input $\boldsymbol{X}$, or to the transformed value $\boldsymbol{XW} + \boldsymbol{b}$. Ioffe and Szegedy (2015) recommend