Why does Batch Norm work?

(SPEECH)
So,

(DESCRIPTION)
Text, Batch Normalization. Why does Batch Norm work? Website, deep learning, dot, A.I.

(SPEECH)
why does batch norm work?

Here's one reason, you've seen how normalizing the input features, the X's, to mean zero and variance one, how that can speed up learning.

So rather than having some features that range from zero to one, and some from one to a 1,000, by normalizing all the features, input features X, to take on a similar range of values that can speed up learning.

So, one intuition behind why batch norm works is, this is doing a similar thing, but further values in your hidden units and not just for your input there.

Now, this is just a partial picture for what batch norm is doing.

There are a couple of further intuitions, that will help you gain a deeper understanding of what batch norm is doing.

Let's take a look at those in this

(DESCRIPTION)
New slide, learning on shifting input distribution.

(SPEECH)
video.

A second reason why batch norm works, is it makes weights, later or deeper than your network, say the weight on layer 10, more robust to changes to weights in earlier layers of the neural network, say, in layer one.

To explain what I mean, let's look at this most vivid example.

Let's see a training on network, maybe a shallow network, like logistic regression or maybe a neural network, maybe a shallow network like this regression or maybe a deep network, on our famous cat detection toss.

But let's say that you've trained your data sets on all images of black cats.

If you now try to apply this network to data with colored cats where the positive examples are not just black cats like on the left, but to color cats like on the right, then your cosfa might not do very well.

So in pictures, if your training set looks like this, where you have positive examples here and negative examples here, but you were to try to generalize it, to a data set where maybe positive examples are here and the negative examples are here, then you might not expect a module trained on the data on the left to do very well on the data on the right.

Even though there might be the same function that actually works well, but you wouldn't expect your learning algorithm to discover that green decision boundary, just looking at the data on the left.

So, this idea of your data distribution changing goes by the somewhat fancy name, covariate shift.

And the idea is that, if you've learned some X to Y mapping, if the distribution of X changes, then you might need to retrain your learning algorithm.

And this is true even if the function, the ground true function, mapping from X to Y, remains unchanged, which it is in this example, because the ground true function is, is this picture a cat or not.

And the need to retain your function becomes even more acute or it becomes even worse if the ground true function shifts as well.

So, how does this problem of covariate shift apply to a neural network?

(DESCRIPTION)
New slide, Why this is a problem with neural networks?

(SPEECH)
Consider a deep network like this, and let's look at the learning process from the perspective of this certain layer, the third hidden layer.

So this network has learned the parameters W3 and B3.

And from the perspective of the third hidden layer, it gets some set of values from the earlier layers, and then it has to do some stuff to hopefully make the output Y-hat close to the ground true value Y.

So let me cover up the nose on the left for a second.

So from the perspective of this third hidden layer, it gets some values, let's call them A_2_1, A_2_2, A_2_3, and A_2_4.

But these values might as well be features X1, X2, X3, X4, and the job of the third hidden layer is to take these values and find a way to map them to Y-hat.

So you can imagine doing great intercepts, so that these parameters W_3_B_3 as well as maybe W_4_B_4, and even W_5_B_5, maybe try and learn those parameters, so the network does a good job, mapping from the values I drew in black on the left to the output values Y-hat.

But now let's uncover the left of the network again.

The network is also adapting parameters W_2_B_2 and W_1B_1, and so as these parameters change, these values, A_2, will also change.

So from the perspective of the third hidden layer, these hidden unit values are changing all the time, and so it's suffering from the problem of covariate shift that we talked about on the previous slide.

So what batch norm does, is it reduces the amount that the distribution of these hidden unit values shifts around.

And if it were to plot the distribution of these hidden unit values, maybe this is technically renormalizer Z, so this is actually Z_2_1 and Z_2_2, and I also plot two values instead of four values, so we can visualize in 2D.

What batch norm is saying is that, the values for Z_2_1 Z and Z_2_2 can change, and indeed they will change when the neural network updates the parameters in the earlier layers.

But what batch norm ensures is that no matter how it changes, the mean and variance of Z_2_1 and Z_2_2 will remain the same.

So even if the exact values of Z_2_1 and Z_2_2 change, their mean and variance will at least stay same mean zero and variance one.

Or, not necessarily mean zero and variance one, but whatever value is governed by beta two and gamma two.

Which, if the neural networks chooses, can force it to be mean zero and variance one.

Or, really, any other mean and variance.

But what this does is, it limits the amount to which updating the parameters in the earlier layers can affect the distribution of values that the third layer now sees and therefore has to learn on.

And so, batch norm reduces the problem of the input values changing, it really causes these values to become more stable, so that the later layers of the neural network has more firm ground to stand on.

And even though the input distribution changes a bit, it changes less, and what this does is, even as the earlier layers keep learning, the amounts that this forces the later layers to adapt to as early as layer changes is reduced or, if you will, it weakens the coupling between what the early layers parameters has to do and what the later layers parameters have to do.

And so it allows each layer of the network to learn by itself, a little bit more independently of other layers, and this has the effect of speeding up of learning in the whole network.

So I hope this gives some better intuition, but the takeaway is that batch norm means that, especially from the perspective of one of the later layers of the neural network, the earlier layers don't get to shift around as much, because they're constrained to have the same mean and variance.

And so this makes the job of learning on the later layers easier.

It turns out batch norm has a second effect, it has a slight regularization effect.

(DESCRIPTION)
New slide, Batch Norm as regularization.

(SPEECH)
So one non-intuitive thing of a batch norm is that each mini-batch, I will say mini-batch X_t, has the values Z_t, has the values Z_l, scaled by the mean and variance computed on just that one mini-batch.

Now, because the mean and variance computed on just that mini-batch as opposed to computed on the entire data set, that mean and variance has a little bit of noise in it, because it's computed just on your mini-batch of, say, 64, or 128, or maybe 256 or larger training examples.

So because the mean and variance is a little bit noisy because it's estimated with just a relatively small sample of data, the scaling process, going from Z_l to Z_2_l, that process is a little bit noisy as well, because it's computed, using a slightly noisy mean and variance.

So similar to dropout, it adds some noise to each hidden layer's activations.

The way dropout has noises, it takes a hidden unit and it multiplies it by zero with some probability.

And multiplies it by one with some probability.

And so your dropout has multiple of noise because it's multiplied by zero or one, whereas batch norm has multiples of noise because of scaling by the standard deviation, as well as additive noise because it's subtracting the mean.

Well, here the estimates of the mean and the standard deviation are noisy.

And so, similar to dropout, batch norm therefore has a slight regularization effect.

Because by adding noise to the hidden units, it's forcing the downstream hidden units not to rely too much on any one hidden unit.

And so similar to dropout, it adds noise to the hidden layers and therefore has a very slight regularization effect.

Because the noise added is quite small, this is not a huge regularization effect, and you might choose to use batch norm together with dropout, and you might use batch norm together with dropouts if you want the more powerful regularization effect of dropout.

And maybe one other slightly non-intuitive effect is that, if you use a bigger mini-batch size, right, so if you use use a mini-batch size of, say, 512 instead of 64, by using a larger mini-batch size, you're reducing this noise and therefore also reducing this regularization effect.

So that's one strange property of dropout which is that by using a bigger mini-batch size, you reduce the regularization effect.

Having said this, I wouldn't really use batch norm as a regularizer, that's really not the intent of batch norm, but sometimes it has this extra intended or unintended effect on your learning algorithm.

But, really, don't turn to batch norm as a regularization.

Use it as a way to normalize your hidden units activations and therefore speed up learning.

And I think the regularization is an almost unintended side effect.

So I hope that gives you better intuition about what batch norm is doing.

Before we wrap up the discussion on batch norm, there's one more detail I want to make sure you know, which is that batch norm handles data one mini-batch at a time.

It computes mean and variances on mini-batches.

So at test time, you try and make predictors, try and evaluate the neural network, you might not have a mini-batch of examples, you might be processing one single example at the time.

So, at test time you need to do something slightly differently to make sure your predictions make sense.

Like in the next and final video on batch norm, let's talk over the details of what you need to do in order to take your neural network trained using batch norm to make predictions.