Batch Norm at test time

(SPEECH)
Batch

(DESCRIPTION)
Text, Batch Normalization. Batch Norm at test time. Website, deep learning, dot, A.I.

(SPEECH)
norm processes your data one mini batch at a time, but the test time you might need to process the examples one at a time.

Let's see how you can adapt your network to do that.

(DESCRIPTION)
New slide, Batch Norm at test time.

(SPEECH)
Recall that during training, here are the equations you'd use to implement batch norm.

Within a single mini batch, you'd sum over that mini batch of the ZI values to compute the mean.

So here, you're just summing over the examples in one mini batch.

I'm using M to denote the number of examples in the mini batch not in the whole training set.

Then, you compute the variance and then you compute Z norm by scaling by the mean and standard deviation with Epsilon added for numerical stability.

And then Z total is taking Z norm and rescaling by gamma and beta.

So, notice that mu and sigma squared which you need for this scaling calculation are computed on the entire mini batch.

But the test time you might not have a mini batch of 6428 or 2056 examples to process at the same time.

So, you need some different way of coming up with mu and sigma squared.

And if you have just one example, taking the mean and variance of that one example, doesn't make sense.

So what's actually done?

In order to apply your neural network and test time is to come up with some separate estimate of mu and sigma squared.

And in typical implementations of batch norm, what you do is estimate this using a exponentially weighted average where the average is across the mini batches.

So, to be very concrete here's what I mean.

Let's pick some layer L and let's say you're going through mini batches X1, X2 together with the corresponding values of Y and so on.

So, when training on X1 for that layer L, you get some mu L. And in fact, I'm going to write this as mu for the first mini batch and that layer.

And then when you train on the second mini batch for that layer and that mini batch,you end up with some second value of mu.

And then for the fourth mini batch in this hidden layer, you end up with some third value for mu.

So just as we saw how to use a exponentially weighted average to compute the mean of Theta one, Theta two, Theta three when you were trying to compute a exponentially weighted average of the current temperature, you would do that to keep track of what's the latest average value of this mean vector you've seen.

So that exponentially weighted average becomes your estimate for what the mean of the Zs is for that hidden layer and similarly, you use an exponentially weighted average to keep track of these values of sigma squared that you see on the first mini batch in that layer, sigma square that you see on second mini batch and so on.

So you keep a running average of the mu and the sigma squared that you're seeing for each layer as you train the neural network across different mini batches.

Then finally at test time, what you do is in place of this equation, you would just compute Z norm using whatever value your Z have, and using your exponentially weighted average of the mu and sigma square whatever was the latest value you have to do the scaling here.

And then you would compute Z total on your one test example using that Z norm that we just computed on the left and using the beta and gamma parameters that you have learned during your neural network training process.

So the takeaway from this is that during training time mu and sigma squared are computed on an entire mini batch of say 64 engine, 28 or some number of examples.

But that test time, you might need to process a single example at a time.

So, the way to do that is to estimate mu and sigma squared from your training set and there are many ways to do that.

You could in theory run your whole training set through your final network to get mu and sigma squared.

But in practice, what people usually do is implement and exponentially weighted average where you just keep track of the mu and sigma squared values you're seeing during training and use and exponentially the weighted average, also sometimes called the running average, to just get a rough estimate of mu and sigma squared and then you use those values of mu and sigma squared that test time to do the scale and you need the head and unit values Z.

In practice, this process is pretty robust to the exact way you used to estimate mu and sigma squared.

So, I wouldn't worry too much about exactly how you do this and if you're using a deep learning framework, they'll usually have some default way to estimate the mu and sigma squared that should work reasonably well as well.

But in practice, any reasonable way to estimate the mean and variance of your head and unit values Z should work fine at test.

So, that's it for batch norm and using it.

I think you'll be able to train much deeper networks and get your learning algorithm to run much more quickly.

Before we wrap up for this week, I want to share with you some thoughts on deep learning frameworks as well.

Let's start to talk about that in the next video.