02:14:55 SECTION 3: hyperpamaters, AdamW, gradient clipping

improved the performance by about 11x right because we started at about 1,000 milliseconds per step and we're now down to like 93 milliseconds so that's uh quite good and we're uh doing a much better job of utilizing our GPU resources so I'm going to now turn to more algorithmic changes uh and improvements to the actual optimization itself and what we would like to do is we would like to follow the hyper parameters that are mentioned in the GP G2 or gpt2 gpt3 paper now sadly gpt2 is uh doesn't actually say too much it's very nice of them that they released the model weights and the code but the paper itself is extremely vague as to the optimization details uh the code itself that they released as well the code we've been looking at this is just the inference code so there's no training code here and very few hyp parameters so this

doesn't also tell us too much so for that we have to turn to the gpt3 paper and um in the depending of the gpt3 paper um they have a lot more hyper parameters here for us to use and the gpt3 paper in general is a lot more detailed as to uh all of the you know small details that go into the model training but gpt3 U models were never released so gbt2 we have the weights but no details and gpt3 we have lots of details but no weights so um but roughly speaking gpt2 and gpt3 architectures are very very similar and um basically there are very few changes the context length was expanded from 1024 to 2048 and that's kind of like the major change uh and some of the hyper parameters around the Transformer have changed but otherwise they're pretty much the same model it's just that gpt3 was trained for a lot longer on a bigger data set and uh has a lot more thorough evaluations uh and the gpt3 model is 175 billion instead

of 1.6 billion um in the gpt2 so long story short we're going to go to gp3 paper to follow along some the hyper parameters so to train all the versions of gpt3 we use atom with beta 1 beta 2 of9 and .95 so let's swing over here and make sure that the betas parameter which you can see here defaults to 0.9 and 999 is actually set to 0.9 and .95 and then the Epsilon parameter uh you can see is the default is 1 in8 and this is also one in8 let's just uh put it in so that works expit uh now next up they say we clip the gra Global Norm of the gradient at 1.0 so what this is referring to is that once we calculate the gradients right after l. backward um we basically have the gradients at all the parameter tensors and what people like to do is basically uh clip them to have some kind of a maximum Norm so in pytor this is fairly easy to do uh it's one line of code here that we have to insert right after we calcul Cal the gradients

and what this utility function is doing is um it's calculating the global Norm of the parameters so every single par um gradient on all the parameters you square it and you add it all up and you take a big square root of that and that's the norm of the parameter V Vector basically it's the it's the length of it if you if you'd like to look at it that way and we are basically making sure that its length is no more than 1.0 and we're going to clip it and the reason that people like to use this is that uh sometimes you can get unlucky during your optimization maybe it's a bad data batch or something like that and if you get very unlucky in the batch you might get really high loss and really high loss could lead to a really high gradient and this could basically uh shock your model and shock the optimization so people like to use a gradient Norm clipping uh to prevent the model from um basically getting too big of shocks in terms of the gradient magnet ude

and uh the upper bound it in this way it's a bit of a hacky solution it's about like a patch on top of like deeper issues uh but uh people still do it fairly frequently now the clip grad Norm Returns the norm of the gradient which I like to always visualize uh because um it is useful information and sometimes you can look at the norm of the gradient and if it's well behaved things are good if it's climbing things are bad and they're destabilizing during training sometimes you could get a spike in the norm and that means there's some kind of an issue or an instability so the norm here will be a norm uh and let's do a uh 4f or something like that and I believe this is just a float and so we should be able to uh print that uh so that's Global gradient clipping now they go into the details of the learning rate uh scheduler so they don't just use a fixed learning rate like we do here for 3 E4 but there's actually basically a cosine DK learning rate

schedule um it's got a warm-up and it's got a cosine DEC to 10% over some Horizon um and so we're going to implement uh this in a second I just like to see Norm printed here okay there we go so what happened here is the norm is actually really high in the beginning 30 or so and you see that as we continue training it kind of like stabilizes um at values below one um and this is not that crazy uncommon for the norm to be high in the very first few stages basically What's Happening Here is the model is completely random and so there's a ton of learning happening very early in the network but that learning is kind of like um you know it's mostly learning the biases of the output tokens and so it's a bit of an unstable time uh but the network usually stabilizes in a very few iterations so this looks very relatively reasonable to me except usually I would expect this looks a little bit funky that we go from 28 to 6 to 2 and then to 10 um

it's not completely insane but it's just kind of a little bit funky um okay so let's now get to the learning rate schuer so the learning