00:56:42 optimization loop: overfit a single batch

so let's do the optimization now um so here we have the loss is this is how we get the loss but now basically we want a load for Loop here so 4 I in range let's do 50 steps or something like that uh let's create an Optimizer object in pytorch um and so here we are using the atom um Optimizer which is an alternative to the stochastic radian descent Optimizer SGD that we were using so SGD is a lot simpler atom is a bit more involved and I actually specifically like the atom W variation because in my opinion it kind of just like fixes a bug um so adom w is a bug fix of atom is what I would say when we go to the documentation for atom W oh my gosh we see um that it takes a bunch of hyper parameters and it's a little bit more complicated than the SGD we were looking at before uh because in addition to basically updating the parameters with the gradient

uh scaled by the Learning rate it keeps these buffers around and it keeps two buffers the m and the V which it calls the first and the second moment so something that looks a bit like momentum and something that looks a bit like RMS prop if you're familiar with it but you don't have to be it's just kind of a normalization that happens on each gradient element individually and speeds up the optimization especially for language models but I'm not going to go into the detail right here we're going to treat it as a bit of a black box and it just optimizes um the objective faster than SGD which is what we've seen in the previous lectures so let's use it as a black box in our case uh create the optimizer object and then go through the optimization the first thing to always make sure the co-pilot did not forget to zero the gradients so um always remember that you have to start with a zero gradient then when you get

your loss and you do a DOT backward dot backward adds to gradients so it deposits gradients it it always does a plus equals on whatever the gradients are which is why you must set them to zero so this accumulates the gradient from this loss and then we call the step function on the optimizer to um update the parameters and to um decrease the loss and then we print a step and the loss do item is used here because loss is a tensor with a single element do item will actually uh convert that to a single float and this float will live not will will live on the CPU so this gets to some of the internals again of the devices but loss is a is a tensor with a single element and it lifts on GPU for me because I'm using gpus when you call item P torch behind the scenes will take that one-dimensional tensor ship it back to the CPU uh memory and convert it into a float that we can just print so this is the optimization and this

should probably just work let's see what happens actually sorry let me instead of using CPU override let me delete that so this is a bit faster for me and it runs on Cuda oh expected all tensors to be on the same device but found at least two devices Cuda zero and CPU so Cuda zero is the zeroth GPU because I actually have eight gpus on this box uh so the zeroth GPU in my box and CPU and model we have moved to device but when I was writing this code I actually introduced a bug because buff we never moved to device and you have to be careful because you can't just do buff dot two of device um it's not stateful it doesn't convert it to be a device it instead uh returns pointer to a new memory which is on the device so you see how we can just do model that two a device that does not apply to tensors you have to do buff equals um b.2 device and then this should work okay so what do we expect to see we

expect to see a reasonable loss in the beginning and then we continue to optimize just the single batch and so we want to see that we can overfit this single batch we can we can crush this little batch and we can perfectly predict the indices on just this little batch and indeed that is roughly what we're seeing here so um we started off at roughly 10.82 11 in this case and then as we continue optimizing on this single batch without loading new examples we are making sure that we can overfit a single batch and we are getting to very very low loss so the Transformer is memorizing this single individual batch and one more thing I didn't mention is uh the learning rate here is 3 E4 which is a pretty good default for most uh optimizations that you want to run at a very early debugging stage so this is our simple inter Loop and uh we are overfitting a single batch and this looks good so now what uh what comes next is we don't just

want to overfit a single batch we actually want to do an optimization so we actually need to iterate these XY batches and create a little data loader uh that makes sure that we're always getting a fresh batch and that we're actually optimizing a reasonable objective so let's do that next okay so this is what I came up with