00:52:53 cross entropy loss

going to adjust the forward function of this NN module in the model and in particular we're not just going to be returning logits but also we're going to return the loss uh and we're going to not just pass in the input in thees but also the targets uh in y and now we will print not Lo just. shape anymore we're actually going to print the loss function and then c. exit of zero so that we skip some of the sampling logic so now let's swing up to the forward function which gets called there because now we also have these optional targets and when we get the targets we can also calculate uh the loss and remember that we want to basically return uh log just loss and loss by default is none but um let's put this here if uh targets is not none then we want to calculate loss and co-pilot is already getting excited here and calculating the what looks to be correct

loss it is using the cross entropy loss as is documented here uh so this is a function in pytorch under the functional now what is actually happening here because it looks a little bit scary uh basically uh the F that cross entropy does not like multi-dimensional inputs it can't take a b BYT by vocap size so what's happening here is that we are flattening out this three-dimensional tensor into just two Dimensions the First Dimension is going to be calculated automatically and it's going to be B * T and then the last Dimension is vocap size so basically this is uh flattening out this three-dimensional tensor of logits to just be two- dimensional B * T all individual examples and vocap size on uh in terms of the length of each row and then it's also flattening out the targets which are also two-dimensional at this stage but we're going to just flatten them out so they're just a single tensor of B * T and this can then pass into

cross entropy to calculate a loss which we return so this should basically at this point run because this is not too complicated so let's run it and let's see if we should be printing the loss and here we see that we printed 11 uh roughly and so um and notice that this is the tensor of a single element which is this number 11 now we also want to be able to calculate a reasonable uh kind of starting point for a random rationalized Network so we covered this in previous videos but our vocabulary size is 50257 at initialization of the network you would hope that um every vocab element is getting roughly a uniform probability uh so that we're not favoring at initialization any token way too much we're not confidently wrong at initialization so what we're hoping is that the probability of any arbitrary token is roughly 1 over 50,2 57 and now we can sanity check the loss because remember that the cross entropy loss is just basically

the negative um log likelihood so if we now take this probability and we take it through the natural logarithm and then we do the negative that is the loss we expect at initialization and we covered this in previous videos so I would expect something around 10.82 and we're seeing something around 11 so it's not way off this is roughly the probability I expect at initialization so that tells me that the at initialization or probability distribtion is roughly diffused it's a good starting point and we can now uh perform the optimization and tell the network which elements you know should follow correctly in what order so at this point we can do a l step backward calculate the gradients and do an optimization so let's get to that okay