00:28:08 loading the huggingface/GPT-2 parameters

so here I've also changed the GPT config so that the numbers here the H parameters agree with the gpt2 124 M model so the maximum sequence length which I call block size here is 124 the number of tokens is 50250 257 which if you watch my tokenizer video know that this is 50,000 m merges BP merges 256 bite tokens the leaves of the BP tree and one special end of text token that delimits different documents and can start generation as well and there are 12 layers there are 12 heads in the attention and the dimension of the Transformers was 768 so here's how we can now load the parameters from hugging face to uh our code here and initialize the GPT class with those parameters so let me just copy paste a bunch of code here and I'm not going to go through this code too slow too quickly too slowly because um honestly it's not that interesting it's not that exciting we're just loading the weights so it's kind of dry but as I mentioned there are four models in this miniseries of gpt2 this is some of the Jupiter code um code that we had here on the right I'm just pting it over these are the hyper parameters of the gpt2 models uh we're creating the config object and creating our own model and then what's Happening Here is we're creating the state dict both for our model and for the hugging face model um and then what we're doing here is we're going over the hugging face model keys and we're copying over those tensors and in the process we are kind of ignoring a few of the buffers they're not parameters they're buffers so for example attention dobias uh that's just used for the autoaggressive mask and so we are ignoring some of those masks and uh that's it and then then one additional kind of annoyance is that this comes from the tensorflow repo and I'm not sure how this is a little bit annoying but some of the weights are transposed from what pytorch would want and so manually I hardcoded the weights that should be transposed and then we transpose them if that is so and then we return this model so the from pre-trained is a Constructor or class method in Python that Returns the GPT object if we just give it the model type which in our case is gpt2 the smallest model that we're interested in so this is the code and this is how you would use it and um we can pop open the terminal here in vs code and we can python train gbt2 pi and fingers crossed okay so we didn't crash and so we can load the weights and the biases and everything else into our Ann module but now let's also get additional confidence that this is working and let's try to actually generate from this