03:28:23 evaluation: HellaSwag, starting the run

the optimization running here and it's stepping and we're on step 6,000 or so so we're about 30% through training now while this is training I would like to introduce one evaluation that we're going to use to supplement the validation set and that is the H swag eval so hos swag comes from this paper back in 2019 so it's a 5-year-old eval now and the way H swag works is there is basically a sentence completion data set so it's a multiple choice for every one of these questions we have uh basically a shared context like a woman is outside with a bucket and a dog the dog is running around trying to avoid bath she a Rises the bucket off with soap and blow dry the dog's head B uses a hose to keep it from getting soapy C gets the dog wet and it runs away again or D gets into a bathtub with the dog and so basically the idea is that these multiple

choice are constructed so that one of them is a natural continuation of the um sentence and the others are not and uh the others might not make sense like uses the host to keep it from getting soaped that makes no sense and so what happens is that models that are not trained very well are not able to tell these apart but models that have a lot of World Knowledge and can tell uh which um and can tell a lot about the world will be able to create these completions and these sentences are sourced from activity net and from Wiki how and at the bottom of the uh paper there's kind of like a cool chart of the kinds of domains in Wiki house so there's a lot of sentences from computers and electronics and Homes and Garden and it has kind of a broad coverage of the kinds of things you need to know about the world in order to find the most likely completion and um the identity of that of that completion one more thing that's kind of interesting

about H swag is the way it was constructed is that the incorrect um options are deliberately um adversarially sourced so they're not just random sentences they're actually sentences generated by language models and they're generated in such a way that language models basically find them difficult but humans find them easy and so they mentioned that humans have a 95% accuracy on this set but at the time the state-of-the-art language models had only 48% and so at the time this was a good Benchmark now you can read the details of this paper to to learn more um the thing to point out though is that this is 5 years ago and since then what happened to H swag is that it's been totally just uh um solved and so now the language models here are 96% so basically the 4% the last 4% is probably errors in the data set or the questions are really really hard and so basically this data set is kind of crushed with respect to

language models but back then the best language model was only at about 50% uh but this is how far things got but still the the reason people like H swag and it's not used by the way in gpt2 but in gpt3 there is H swag eval and lots of people use H swag and so for gpt3 we have results here that are cited so we know what percent accuracies gpt3 um attains at all these different model checkpoints for H swag eval and the reason people like it is because H swag is a smooth eval and it is an eval that offers quote unquote early signal uh so early signal means that even small language models are going to start at the random chance of 25% but they're going to slowly improve and you're going to see 25 26 27 Etc and uh you can see slow Improvement even when the models are very small and it's very early so it's smooth it has early signal and um it's been around for a long time so that's why people kind of

like this eval uh now the way that we're going to evaluate this is as follows as I mentioned we have a shared context and this is kind of like a multiple choice task but instead of giving the model a multiple choice question and asking it for A B C or D uh we can't do that because these models when they are so small as we are seeing here the models can't actually do multiple choice they don't understand the concept of associating a label to one of the options of multiple choice uh they don't understand that so we have to give it to them in a native form and the native form is a token completion so here's what we do we construct a batch of four rows and uh T tokens whatever that t happens to be then the shared context that is basically the context for the for choices the tokens of that are shared across all of the rows and then we have the four options so we kind of like lay them out and then only one of the

options is correct in this case label three option three and so um this is the correct option and option one two and for are incorrect now these options might be of different lengths so what we do is we sort of like take the longest length and that's the size of the batch B BYT and then some of these uh here are going to be pded Dimensions so they're going to be unused and so we need the tokens we need the correct label and we need a mask that tells us which tokens are active and the mask is then zero for these uh padded areas so that's how we construct these batches and then in order to get the language model to predict A B C or D the way this works is basically we're just going to look at the tokens their probabilities and we're going to pick the option that gets the lowest or the highest average probability for the token so for the tokens because that is the most likely completion according to the language

model so we're just going to look at the um probabilities here and average them up across the options and pick the one with the highest probability roughly speaking so this is how we're going to do H swag um and this is I believe also how uh gpt3 did it um this is how gpt3 did it as far as I know but you should note that some of the other evals where you might see H swag may not do it this way they may do it in a multiple choice format where you sort of uh give the the context a single time and then the four completions and so the model is able to see all the four options before it picks the best possible option and that's actually an easier task for a model because you get to see the other options when you're picking your choice um but unfortunately models at our size can't do that only models at a bigger size are able to do that and so our models are actually slightly handicapped in this way that they are not going to see the other

options they're only going to see one option at a time and they just have to assign probabilities and the correct option has to win out in this metric all right so let's now implement this very briefly and incorporate it into our script okay so what I've done here is I've introduced a new file called hell swag. py that you can take a look into and I'm not going to to step through all of it because uh this is not exactly like deep code deep code it's kind of like a little bit tedious honestly because what's happening is I'm downloading hsac from GitHub and I'm rendering all of its examples and there are a total of 10,000 examples I am rendering them into this format um and so here at the end of this render example function you can see that I'm returning the tokens uh the tokens of this um 4xt uh array of Tokens The Mask which tells us which parts are the options and everything else is zero and the label that is the correct label and so that

allows us to then iterate the examples and render them and I have an evaluate function here which can load a um gpt2 from huging face and it runs the eval here um and it basically just calculates uh just as I described it predicts the option that has the lowest or the highest prob ility and the way to do that actually is we can basically evaluate the cross entropy loss so we're basically evaluating the loss of predicting the next token in a sequence and then we're looking at the row that has the lowest average loss and that's the uh option that we pick as the prediction and then we do some stats and prints and stuff like that so that is a way to evaluate L swag now if you go up here I'm showing that for GPT 2124m if you run this script you're going to see that H swag gets 29.5% um so that's the performance we get here now remember that random Chan is 25% so we haven't gone too far and gpt2 XL which is the

biggest the gpt2 gets all the way up to 49% roughly so uh these are pretty low values considering that today's state-ofthe-art is more like 95% uh so these are definitely older models by now and then there's one more thing called Uther harness which is a very piece of infrastructure for running evals for language models and they get slightly different numbers and I'm not 100% sure what the discrepancy is for these um it could be that they actually do the multiple choice uh instead of just the completions and that could be the um uh the discrepancy but I'm not 100% sure about that i' have to take a look but for now our script reports 2955 and so that is the number that we'd like to beat if we are training a GPD 2124m from scratch and ourselves um so now I'm going to go into actually incorporating this eval into our main training script and um and basically because we want to evaluate it in a periodic

manner so that we can track H swag and how it evolves over time and see when when and if we cross uh this 2955 um sort of region so let's now walk through some of the changes to train gpt2 thatp the first thing I did here is I actually made use compile optional kind of and I disabled it by default and the problem with that is the problem with compile is that unfortunately it does make our code faster but it actually breaks the evaluation code and the sampling code it gives me a very gnarly message and I don't know why so hopefully by the time you get to the codebase when I put it up on GitHub uh we're going to fix that by then but for now I'm running without torch compile which is why you see this be a bit slower so we're running without torch compile I also create cre a log directory log where we can place our log.txt which will record the train loss validation loss and the H swag accuracies so a very simple text file and

we're going to uh open for writing so that it sort of starts empty and then we're going to append to it I created a simple variable that um helps tell us when we have a last step and then basically periodically inside this Loop every 250th iteration or at the last step we're going to evaluate the validation loss and then every 250th iteration um we are going to evaluate H swag but only if we are not using compile because compile breaks it so I'm going to come back to this code for evaluating H swag in a second and then every 250th iteration as well we're also going to sample from the model and so you should recognize this as our ancient code from way back when we started the video and we're just sampling from the model and then finally here um these are if we're not after we validate sample and evaluate hell swag we actually do a training step here and so this is one step of uh training and you should be pretty familiar with all of what

this does and at the end here once we get our training laws we write it to the file so the only thing that changed that I really added is this entire section for H swag eval and the way this works is I'm trying to get all the gpus to collaborate on the H swag and so we're iterating all the examples and then each process only picks the examples that assigned to it so we sort of take I and moded by the world size and we have to make it equal to rank otherwise we continue and then we render an example put it on the GPU we get the low jits then I create a helper function that helps us basically predict the option with the lowest loss so this comes here the prediction and then if it's correct we sort of keep count and then if multiple processes were collaborating on all this then we need to synchronize their stats and so the way one way to do that is to package up our statistics here into tensors which we can then call this. alberon and

sum and then here we sort of um unwrap them from tensors so that we just have ins and then here the master process will print and log the hellis swag accuracy so that's kind of the that's kind of it and that's what I'm running right here so you see this optimization here and uh we just had a generation and this is Step 10,000 out of about 20,000 right so we are halfway done and these are the kinds of samples that uh we are getting at this stage so let's take a look hello I'm a language model so I'd like to use it to generate some kinds of output hello I'm a language model and I'm a developer for a lot of companies Al language model uh let's see if I can find fun one um I don't know you can go through this yourself but certainly the predictions are getting less and less random uh it seems like the model is a little bit more self-aware and using language uh that is a bit more uh specific to it being language model hello I'm

a language model and like how the language is used to communicate I'm a language model and I'm going to be speaking English and German okay I don't know so let's just wait until this optimization finishes and uh we'll see what kind of samples we get and we're also going to look at the train Val and the hway accuracy and see how we're doing with respect to