03:43:05 SECTION 4: results in the morning! GPT-2, GPT-3 repro

gpt2 okay good morning so focusing For a Moment On The jupyter Notebook here on the right I created a new cell that basically allows us to visualize the the train Val and Hela and um the hel score and you can step through this it basically like parses the log file that we are writing and um a lot of this is just like boring ma plot lip code but basically this is what our optimization looks like so we ran for 19,731 billion tokens which is whoops oh my gosh which is one Epoch of the sample 10B of webd on the left we have the loss and the in blue we have the training loss in Orange we have the validation loss and red as a horizontal line we have the opening IG gpt2 124 M model checkpoint when it's just evaluated on the validation set of um of this fine web edu uh so you can see that we are surpassing this orange is below the red so we're surpassing the validation set of this data set and like I mentioned the data set distribution is very different from what gpt2 trained on so this is not an exactly fair comparison but it's a good cross check uh to uh to look at now we would ideally like something that is withheld and comparable and somewhat standard um and so for us that is helis swag and so on here we see the H swag progress we made from 25% all the way here in red we see the open gpt2 124 M model in red so it achieves this h bag here and the the gpt3 model 124 M which was trained on 300 billion tokens achieves green so that's over here so you see that we basically surpassed the gbt2 24m uh model right here uh which is uh really nice now interestingly we were able to do so with only training on 10 billion tokens while gpt2 was trained on 100 billion tokens so uh for some reason we were able to get away with significantly fewer tokens for training there are many possibilities to as to why we could match or surpass this accuracy um with only 10 million training so number one um it could be that opening gbt2 was trained on a much wider data distribution so in particular fine web edu is all English it's not multilingual and there's not that much math and code um and so math and code and multilingual could have been stealing capacity from the original gpt2 model and um basically that could be partially the reason why uh this is not working out there's many other reasons um so for example the H swag eval is fairly old uh maybe 5 years or so it is possible that aspects of H swag in some way or even identically have made it into the training Set uh of fine web we don't know for sure but if that was the case then we are basically looking at the training curve instead of the validation curve so long story short this is not a perfect eval and there's some caveats here uh but at least we have some confidence that that we're not doing something completely wrong and um and uh it's probably the case that when people try to create these data sets they try to make sure that test sets that are very common are not part of the training set for example uh when hugging face created the fine web BDU they use H swag as an eval so I would hope that they make sure that they D duplicate and that there's no hella swag in the training set but we can't be sure uh the other thing I wanted to address briefly is look at this loss curve this looks really this looks really wrong here I don't actually know 100% what this is and I suspect it's because the uh 10 billion sample of fine web edu was not properly shuffled um and there's some issue here uh with the data that I don't fully understand yet and there's some weird periodicity to it um and because we are in a very lazy way sort of serializing all the tokens and just iterating all them from scratch without doing any permutation or any random sampling ourselves I think we're inheriting some of the ordering that they have in the data set so uh this is not ideal but hopefully by the time you get to this repo uh some of these things by the way will hopefully be fixed and I will release this build n GPT repo and right now it looks a little ugly and preliminary uh so hopefully by the time you get here it's nicer but down here I'm going to show aada and I'm going to talk about about some of the things that happened after the video and I expect that we will have fixed uh the small issue uh but for now basically this shows that uh our training is not uh completely wrong and it shows that uh we're able to surpass the accuracy with only 10x the token budget um and possibly it could be also that the data set may have improved so uh the original uh gpt2 data set was web text it's possible that not a lot of care and attention went into the data set this was very early in llms whereas now there's a lot more scrutiny on good practices around uh D duplication filtering uh quality filtering and so on and it's possible that the data that we're training on is just of higher quality per token and that could be giving us a boost as well so a number of cave has to think about but for now uh we're pretty happy with this um and yeah now the next thing I was interested in is as you see it's a morning now so there was an overnight and I wanted to basically see how far I could push the result so uh to do an overnight run I basically did instead of one Epoch which took roughly two hours I just did a times four so that that would take eight hours while I was sleeping and so we did four Epoch or roughly 40 billion uh tokens of training and I was trying to see how far we could get um and so this was the only change and I reran the script and when I point uh and read the log file at uh at the 40b uh this is what the curve look like okay so to narrate this number one we are seeing this issue here here with the periodicity through the different Epoch and something really weird with the fine web edu data set and that is to be determined uh but otherwise we are seeing that the H swag actually went up by a lot and we almost we almost made it uh to the GPT 324m accuracy uh up here uh but not quite so uh it's too bad that I didn't sleep slightly longer um and uh I think if this was an uh five Epoch run we may have gotten here now one thing to point out is that if you're doing multi Epoch runs uh we're not actually being very careful in our data loader and we're not um I this data loader goes through the data in exactly the same format and exactly the same order and this is kind of suboptimal and you would want to look into extensions where you actually permute the data uh randomly you permute the documents around in Every Single Shard on every single new Epoch um and po even permute the shards and that would go a long way into decreasing the pricity and it's also better for the optimization so that you're not seeing things ident in the identical format and you're introducing some of the some uh Randomness in how the documents follow each other because you have to remember that in every single row these documents follow each other and then there's the end of text token and then the next document so the documents are currently glued together in the exact same identical manner but we actually want to break break up the documents and shuffle them around because the order of the documents shouldn't matter and they shouldn't um basically we want to break up that dependence because it's a kind of a spous correlation and so our data lad is not currently doing that and that's one Improvement uh you could think of making um the other thing to point out is we're almost matching gpt3 accuracy with only 40 billion tokens gpt3 trained on 300 billion tokens so again we're seeing about a 10x um Improvement here with respect to learning efficiency uh the other thing I wanted to and I don't actually know exactly what to attribute this to other than some of the things that I already mentioned previously for the previous run uh the other thing I wanted to briefly mention is uh the max LR here I saw some people already play with this a little bit in a previous related repository um and it turns out that you can actually almost like three xas so it's possible that the maximum learning rate can be a lot higher and for some reason the gpt3 hyper parameters that we are inheriting are actually extremely conservative and you can actually get away with a Higher

Learning rate and it would train faster so a lot of these hyper parameters um are quite tunable and feel free to play with them and they're probably not set precisely correctly and um it's possible that you can get away with doing this basically and if you wanted to exactly be faithful to gpt3 you would also want to make the following difference you'd want to come here and the sequence length of gpt3 is 2x it's 20 48 instead of 1,24 so you would come here change this to 248 for T and then if you want the exact same number of tokens uh half a million per iteration or per step you want to then decrease this to 32 so they still multiply to half a mil so that would give your model sequence length equal to that of gpt3 and in that case basically the um the models would be roughly identical as far as I'm as far as I'm aware because again gpt2 and gpt3 are very very similar models now we can also look at some of the samples here from the model that was trained overnight so this is the optimization and you see that here we stepped all the way to 76290 also or so and these are the hos mag we achieved was 33.2 4 and these are some of the samples from the model and you can see that if you read through this and pause the video briefly you can see that they are a lot more coherent uh so um and they're actually addressing the fact that it's a language model almost so uh hello I'm a language model and I try to be as accurate as possible um I'm a language model not a programming language I know how to communicate uh I use Python um I don't know if you pause this and look at it and then compare it to the one to the model that was only trained for 10 billion uh you will see that these are a lot more coherent and you can play with this uh yourself one more thing I added to The Code by the way is this chunk of code here so basically right after we evaluate the validation loss if we are the master process in addition to logging the validation loss every 5,000 steps we're also going to save the checkpoint which is really just the state dictionary of the model and so checkpointing is nice just because uh you can save the model and later you can uh use it in some way if you wanted to resume the optimiz ation then in addition to saving the model we have to also save the optimizer State dict because remember that the optimizer has a few additional buffers because of adom so it's got the m and V and uh you need to also resume the optimizer properly you have to be careful with your RNG seeds uh random number generators and so on so if you wanted to exactly be able to resume optimization you have to think through the state of the of the training process but if you just want to save the model this is how you would do it and one one nice reason why you might want to do this is because you may want to evaluate the model a lot more carefully so here we are only kind of like winging the hell swag eval but you may want to use something um nicer like for example the Luther uh Luther evaluation hardness evaluation hardness hardness um so this is a way to also evaluate language models and um so it's possible that um you may want to use basically different infrastructure to more thoroughly evaluate the models on different um evaluations and compare it to the opening gbt2 model on many other um tasks like for example that involve math code or different languages and so on so this is a nice functionality to have as well um and then the other thing I wanted to mention is that everything we've built here this is only the pre-training step so um the GPT here is a it dreams documents it just predicts the next to you can't talk to it like you can talk to chat GPT uh chat GPT if you wanted to talk to the model we have to fine-tune it into the chat format and it's not actually like that complicated if you're looking at supervised fine-tuning or sft really what that means is we're just swapping out a data set into a data set that is a lot more conversational and there's a user assistant user assistant kind of structure and we just fine-tune on it and then we um we basically fill in the user tokens and we sample the assistant tokens it's not a lot more deeper than that uh but basically we swap out the data set and continue training uh but for now we're going to stop at uh pre-training one more thing