Let's build GPT: from scratch, in code, spelled out.

We build a Generatively Pretrained Transformer (GPT), following the paper "Attention is All You Need" and OpenAI's GPT-2 / GPT-3. We talk about connections to ChatGPT, which has taken the world by storm. We watch GitHub Copilot, itself a GPT, help us write a GPT (meta :D!) . I recommend people watch the earlier makemore videos to get comfortable with the autoregressive language modeling framework and basics of tensors and PyTorch nn, which we take for granted in this video. My website: https://karpathy.ai

- Google colab for the video: https://colab.research.google.com/drive/1JMLa53HDuA-i7ZBmqV7ZnA3c_fvtXnx-?usp=sharing

- GitHub repo for the video: https://github.com/karpathy/ng-video-lecture

- Playlist of the whole Zero to Hero series so far: https://www.youtube.com/watch?v=VMj-3...

- nanoGPT repo: https://github.com/karpathy/nanoGPT

- *Attention is All You Need paper*: https://arxiv.org/abs/1706.03762

- *OpenAI GPT-3 paper*: https://arxiv.org/abs/2005.14165

- *OpenAI ChatGPT blog post*: https://openai.com/blog/chatgpt/

- The GPU I'm training the model on is from Lambda GPU Cloud, I think the best and easiest way to spin up an on-demand GPU instance in the cloud that you can ssh to: https://lambdalabs.com . If you prefer to work in notebooks, I think the easiest path today is Google Colab.

Suggested exercises:

- **EX1**: The n-dimensional tensor mastery challenge: Combine the `Head` and `MultiHeadAttention` into one class that processes all the heads in parallel, treating the heads as another batch dimension (answer is in nanoGPT).

- **EX2**: Train the GPT on your own dataset of choice! What other data could be fun to blabber on about? (A fun advanced suggestion if you like: train a GPT to do addition of two numbers, i.e. a+b=c. You may find it helpful to predict the digits of c in reverse order, as the typical addition algorithm (that you're hoping it learns) would proceed right to left too. You may want to modify the data loader to simply serve random problems and skip the generation of train.bin, val.bin. You may want to mask out the loss at the input positions of a+b that just specify the problem using y=-1 in the targets (see CrossEntropyLoss ignore_index). Does your Transformer learn to add? Once you have this, swole doge project: build a calculator clone in GPT, for all of +-*/. Not an easy problem. You may need Chain of Thought traces.)

- **EX3**: Find a dataset that is very large, so large that you can't see a gap between train and val loss. Pretrain the transformer on this data, then initialize with that model and finetune it on tiny shakespeare with a smaller number of steps and lower learning rate. Can you obtain a lower validation loss by the use of pretraining?

- **EX4**: Read some transformer papers and implement one additional feature or change that people seem to use. Does it improve the performance of your GPT?

Chapters:

00:34:53 training the bigram model

00:38:00 port our code to a script

Building the "self-attention"

00:42:13 version 1: averaging past context with for loops, the weakest form of aggregation

00:47:11 the trick in self-attention: matrix multiply as weighted aggregation

00:51:54 version 2: using matrix multiply

00:54:42 version 3: adding softmax

00:58:26 minor code cleanup

01:00:18 positional encoding

01:02:00 THE CRUX OF THE VIDEO: version 4: self-attention

01:11:38 note 1: attention as communication

01:12:46 note 2: attention has no notion of space, operates over sets

01:13:40 note 3: there is no communication across batch dimension

01:14:14 note 4: encoder blocks vs. decoder blocks

01:15:39 note 5: attention vs. self-attention vs. cross-attention

01:16:56 note 6: "scaled" self-attention. why divide by sqrt(head_size)

Building the Transformer

01:19:11 inserting a single self-attention block to our network

01:21:59 multi-headed self-attention

01:24:25 feedforward layers of transformer block

01:26:48 residual connections

01:32:51 layernorm (and its relationship to our previous batchnorm)

01:37:49 scaling up the model! creating a few variables. adding dropout

Notes on Transformer

01:42:39 encoder vs. decoder vs. both (?) Transformers

01:46:22 super quick walkthrough of nanoGPT, batched multi-headed self-attention

01:48:53 back to ChatGPT, GPT-3, pretraining vs. finetuning, RLHF

01:54:32 conclusions

Corrections:

00:57:00 Oops "tokens from the future cannot communicate", not "past". Sorry! :)

01:20:05 Oops I should be using the head_size for the normalization, not C