crosscoder-model-diff-replication

Public

1 Branch 0 Tags Go to file t Go to file Add file Code

ckkissane Update README.md

e221e8d · 4 months ago

.gitignore	first commit	4 months ago
README.md	Update README.md	4 months ago
analysis.py	first commit	4 months ago
buffer.py	first commit	4 months ago
crosscoder.py	update default sav...	4 months ago
train.py	first commit	4 months ago
trainer.py	first commit	4 months ago
utils.py	first commit	4 months ago

README

TLDR

Open source replication of Anthropic's Crosscoders for Model Diffing. The crosscoder was trained to model diff the Gemma-2 2b base and IT residual stream at the middle layer.

See this blog post for more details.

Reading This Codebase

This implementation is adapted from Neel Nanda's code at https://github.com/neelnanda-io/Crosscoders

- `train.py` is the main training script for the crosscoder. Run it with `python train.py`.
- `trainer.py` contains the pytorch boilerplate code that actually trains the crosscoder.
- `crosscoder.py` contains the pytorch implementation of the crosscoder. It was implemented to diff two different models, but

About

Open source replication of Anthropic's Crosscoders for Model Diffing

Readme
Activity
34 stars
1 watching
14 forks
Report repository

Releases

No releases published

Packages

No packages published

Languages

Python 100.0%

should be easily hackable to work with an arbitrary number of models.

- `buffer.py` contains code to extract activations from both models, concatenate them, and store them in a buffer which is shuffled and periodically refreshed during training.
- `analysis.py` is a short notebook replicating some of the results from the Anthropic paper. See this colab notebook for a more comprehensive demo.

It won't work out of the box, but hopefully it's pretty hackable. Some tips:

- In `train.py` I just set the cfg by editing the code, rather than using command line arguments. You'll need to change the "wandb_entity" and "wandb_entity" in the cfg dict.
- You'll need to create a checkpoints dir in `/workspace/crosscoder-model-diff-replication/checkpoints` (or change this path in the code). I would sanity check this with a short test run to make sure your weights will be properly saved at the end of training.
- We load training data from https://huggingface.co/datasets/ckkissane/pile-lmsys-mix-1m-tokenized-gemma-2 as a global tensor called all_tokens, and pass this to the Trainer in `train.py`. This is very hacky, but should be easy to swap out if needed.
- In `buffer.py` we separately normalize both the base and chat activations such that they both have average norm sqrt(d_model). This should be handled for you during training, but note that you'll also need to normalize activations during analysis (or fold the normalization scaling factors into the crosscoder weights).