

Training an M*DEL Expert

Multi-Domain Expert Learning (M*DEL) is an approach to training LLMs for expertise in knowledge domains.

This process involves branching from a base model and training the branch on specific domain data in order to establish an expert, which routing logic is then able to activate in serving inference requests. The result is a framework for domain expertise that is easily-extensible, modular, and efficient.

Contents

- 1 Prerequisites
- 2 Prepare the Compute Instance
- 3 Prepare the Training Data and Settings
- 4 Launch the Training
- 5 Run Inference Testing
- 6 Upload the Expert to HuggingFace

Prerequisites

M*DEL's Aurora model (aurora-m) will be used as the base and RunPod compute resources will be used for training. Also, Weights and Biases will collect information during training for monitoring status, evaluating progress, and to allow comparing subsequent training runs for performance.

Prior to beginning training, the following steps should be taken:

1. Create an account / log in at [HuggingFace](#) and accept the [terms and conditions for using the Aurora model](#).
2. Create a [Weights and Biases](#) account.
3. Create an account / log in at [RunPod](#) and add an SSH key to allow terminal access to cloud instances:

Generate a key locally, if needed:

```
$ ssh-keygen -t ed25519 -C "your_email@example.com"
$ cat ~/.ssh/id_ed25519.pub
```

Paste this into RunPod at Account -> Settings -> Public Keys

In order to connect to instances at RunPod, it is recommended that an SSH client be used via local terminal. This is standard on MacOS and Linux; for Windows, use [PuTTY](#).

Prepare the Compute Instance

Axolotl is an LLM training tool that has been developed for fine-tuning models. A template exists at RunPod to easily launch a container where Axolotl can be used for training an M*DEL expert.

To initialize the compute resources, log in at RunPod and perform the following steps:

1. Start from the [axolotl-runpod template](#) and set the organization Account, as appropriate, from the top-right profile drop-down list.
2. Scroll down to the "Previous Generation" section and click Deploy on the 1xA100 80GB GPU. Leave all settings at defaults and click the Continue and Deploy buttons on subsequent screens. (The quantity of GPUs can be adjusted to make training faster, if desired, by clicking "1x A100 80GB" on the first screen and selecting a greater quantity).
3. Once the axolotl-runpod instance is ready, expand it on the Pods list and find the Connect button to get the command to use for "Basic SSH Terminal" access. Run this command in your local terminal to establish the connection.



The SSH session uses [TMUX](#). This allows for resuming the session if it drops due to a connection issue, but scrolling in TMUX does not function like a standard terminal. See the linked documentation and the [TMUX Cheat Sheet](#) for details.

Prepare the Training Data and Settings

With the compute environment initialized, the configuration and data to use for the training can be put into place.

Retrieve the M*DEL settings and samples for Axolotl:

```
$ cd /workspace/
$ git clone https://github.com/Stillerman/axolotl-mdel
$ cd axolotl-mdel
```

Provide CLI authentication to the terminal for HuggingFace and Weights and Biases accounts:

```
$ huggingface-cli login
[ provide read/write token from https://huggingface.co/settings/tokens ]
$ wandb login
[ provide API key from https://wandb.ai/authorize ]
```

Download the training data that comprises the domain expertise to use for this expert. If the data is accessible from the Internet, then `wget` or `curl` can be used to pull it down to the RunPod instance.

```
$ wget [URL]
```

Alternatively, `scp` can be used to upload data directly from a local file. Get the IP address and port from the "SSH over exposed TCP" details for the axolotl-runpod instance, which can be found by clicking the Connect button for the instance. Then, use those values in the `scp` command, along with the same SSH key used to access the terminal of the instance already:

```
localhost:~$ scp -i ~/.ssh/[ssh-key] -P [port] [local-file] root@[ip-addr]:/workspace/axolotl-mdel
```



To use example data for training, see the `axolotl-mdel/scripts/download-mtg.py` script.

Next, make a configuration file to use for this training run with a meaningful filename; here, it is called "experiment1":

```
$ cp examples/aurora/lora.yml examples/aurora/experiment1.yml
$ vim examples/aurora/experiment1.yml
```

Lines in the `.yml` file that have comments can be edited for subsequent training runs to improve effectiveness and performance. For the first run, be sure to set the "path" under datasets to point to the training data that has been copied to the instance and also set a name for `wandb_project`, which is used to identify the collection of training runs in your Weights and Biases account.

Launch the Training

Training is launched on the compute instance with a single command:

```
$ accelerate launch -m axolotl.cli.train examples/aurora/experiment1.yml
```

This will pre-tokenize the dataset, download the model weights (which will be cached in `/workspace/data/hf_home`), and load checkpoints. Once those steps are complete, the terminal output will show a link to "View run at ..." with a `wandb.ai` URL. This URL provides a dashboard for the training job for monitoring status and performance.[blocked URL](#)

Both the terminal output and WandB dashboard will indicate when the training run is complete, at which point inference can be run.



If the following error appears when launching `axolotl.cli.train`: Error while finding module specification for 'axolotl.cli.train' (ModuleNotFoundError: No module named 'axolotl')

Run the following additional commands within the RunPod instance:

```
$ pip3 install -e '.[deepspeed]'
$ pip uninstall flash_attn
```

This is a workaround for a temporary problem that was encountered and should be resolved shortly, if not already.

Run Inference Testing

Simple testing can be performed on the new expert model by launching an inference server with the .yaml configuration file of the training run:

```
$ accelerate launch -m axolotl.cli.inference examples/aurora/experiment1.yaml --lora_model_dir="./lora-out" --gradio
```

This command will show a link with a gradio.live URL, which provides an interface to the model for testing inference. This is likely not sufficient for chatbook integration, but is robust enough to demonstrate functionality of the new expert.

[blocked URL](#)

Note: If you want to inference using an alternative base model you can run the command without specifying a lora_model_dir. This will work for any base model. Aurora and Starcoder examples below:

```
$ accelerate launch -m axolotl.cli.inference examples/aurora/experiment1.yaml --gradio # this will inference aurora
$ accelerate launch -m axolotl.cli.inference examples/starcoder/lora.yaml --gradio # this will inference starcoder
```

Upload the Expert to HuggingFace

Optionally, the newly-trained expert model can be uploaded to HuggingFace to make it easy to use later. This would be accomplished with the following command:

```
$ huggingface-cli upload [repo-id] [local-path] [path-in-repo] --token=[token with WRITE permission]
```

For example:

```
$ huggingface-cli upload stillerman/aurora-mathematica ./lora-out/ . --token=...
```

This uploads the model to the Hub at HuggingFace and is public, by default. Add the `--private` flag to make it private.