🤗 AutoTrain: No-code training for state-of-the-art models

Abhishek Thakur
Hugging Face, Inc.
abhishek@huggingface.co

Abstract

With the advancements in open-source models, training(or finetuning) models on custom datasets has become a crucial part of developing solutions which are tailored to specific industrial or open-source applications. Yet, there is no single tool which simplifies the process of training across different types of modalities or tasks. We introduce 🤗 *AutoTrain* (aka AutoTrain Advanced)—an open-source, no code tool/library which can be used to train (or finetune) models for different kinds of tasks such as: large language model (LLM) finetuning, text classification/regression, token classification, sequence-to-sequence task, finetuning of sentence transformers, visual language model (VLM) finetuning, image classification/regression and even classification and regression tasks on tabular data. 🤗 *AutoTrain Advanced* is an open-source library providing best practices for training models on custom datasets. The library is available at https://github.com/huggingface/autotrain-advanced. AutoTrain can be used in fully local mode or on cloud machines and works with tens of thousands of models shared on Hugging Face Hub and their variations.

Demo screencast: YouTube

1 Introduction

With recent advancements in open-source and open-access state-of-the-art models, the need for standardized yet customizable training of models on downstream tasks has become crucial. However, a universal open-source solution for a diverse range of tasks is still lacking. To address this challenge, we introduce 🤗 *AutoTrain* (also known as AutoTrain Advanced).

AutoTrain is an open-source solution which offers model training for different kinds of tasks such as: large language model (LLM) finetuning, text classification/scoring, token classification, training custom embedding models using sentence transformers (Reimers and Gurevych, 2019), finetuning for visual language models (VLMs), computer vision tasks such as image classification/scoring, object detection and even tabular regression and classification tasks. At the time of writing this paper, a total of 22 tasks: 16 text-based, 4 image-based and 2 tabular based have been implemented.

The idea behind creating AutoTrain is to allow a simple interface for training models on custom datasets without requiring extensive knowledge of coding. AutoTrain is intended for not just no-coders but also for experienced data scientists and machine learning practioners. Instead of writing complex scripts, one can focus on gathering and preparing your data and let AutoTrain handle the training part. AutoTrain UI is shown in Figure 1.

When talking about model training, there are several problems which arise:

Complexity of hyperparameter tuning: Finding the right parameters for tuning models can only be done by significant experimentations and expertise. Improperly tuning the hyperparameters can result in overfitting or underfitting.

Model validation: A good way to make sure the trained models generalize well, is to have a proper valiation set and a proper way to evaluate with appropriate metrics. Overfitting to training data can cause the models to fail in real-world scenarios.

Distributed training: Training models on larger datasets with multi-gpu support can be cumbersome and requires significant changes to codebase. Distributed training requires additional complexity when it comes to synchronization and data handling.

Monitoring: While training a model, its crucial to monitor losses, metrics and artifacts to make sure there is nothing fishy going on.

Maintenance: With ever-changing data, it may be necessary to retrain or fine-tune the model on new data while keeping the training settings consistent.

We introduce the open source *AutoTrain Ad-*

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing
System Demonstrations, pages 419–423

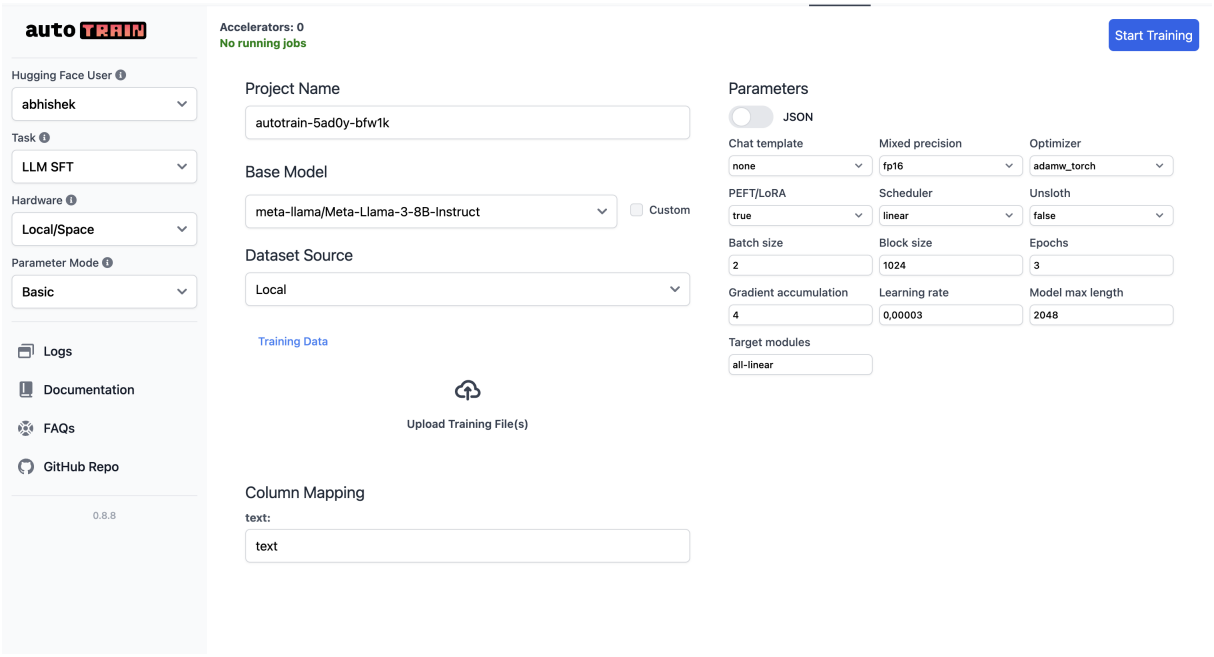

Figure 1: A screenshot of the AutoTrain User Interface (UI)

vanced library to address many of these problems.

2 Related work

In recent years, many AutoML solutions have been developed to automate the training process of machine learning models. Some notable solutions include:

AutoML Solutions AutoSklearn (Feurer et al., 2015), which is an open-source AutoML toolkit built on top of the popular scikit-learn library. AutoSklearn uses Bayesian optimization to automate the process of model selection and hyperparameter tuning.

AutoCompete (Thakur and Krohn-Grimberghe, 2015), which won the Codalab AutoML GPU challenges, builds a framework for tacking machine learning competitions. The code is, however, not open source.

Axolotl (Cloud, 2024) is a CLI tool for finetuning LLMs.

AutoKeras (Jin et al., 2023), developed on top of Keras offers functionalities for various tasks such as image classification, text classification, and regression

Many other closed-source solutions have also been developed by Google, Microsoft, and Amazon. However, all these solutions have some limitations. They are either not open-source, or if they are, they can only handle a limited number of tasks. Many of these solutions are also not no-code, making them inaccessible to non-coders.

With *AutoTrain*, we provide a single interface to deal with many different data format, task, and model combinations, which depending on user's choices is also closely connnected to Hugging Face Hub which enables download, inference and sharing of models with the entire world. Moreover, AutoTrain supports almost all kinds models which are compatible with Hugging Face Transformers (Wolf et al., 2019) library, making it a unique solution to support hundreds of thousands of models for finetuning, including the models which require custom code.

3 Library: *AutoTrain Advanced*

The *AutoTrain Advanced* python library provides a command line interface (CLI), a graphical user interface (GUI/UI) and python SDK to enable training on custom datasets. The datasets can be uploaded/used in different formats such as zip files, CSVs or JSONLs. We provide documentation and walkthroughs on training models for different task and dataset combinations with example hyperparameters, evaluation results and usage of the trained models. The library is licensed as Apache 2.0 license and is available on Github, [1] making it easy for anyone to adopt and contribute.

[1] https://github.com/huggingface/autotrain-advanced

The design of the library has been made keeping in mind both professionals and amateurs who would like to finetune model but don't know where to start and don't want to invest time setting up a separate environment for each of their finetuning tasks. The library lies on the shoulders of giants such as Transformers (Wolf et al., 2019), Hugging Face Datasets (Lhoest et al., 2021), Accelerate (Gugger et al., 2022), Diffusers(von Platen et al., 2022), PEFT (Mangrulkar et al., 2022), TRL (von Werra et al., 2020) and other libraries created by Hugging Face.

AutoTrain uses (Paszke et al., 2019) as the main backend for training the models. For tabular datasets, models from (Van der Walt et al., 2014) and (Chen and Guestrin, 2016) are used as preferred models.

3.1 Component of the AutoTrain Advanced library

There are 3 main components in the *AutoTrain Advanced* library:

Project Configuration: manages the configuration of the project and allows users to set up and manage their training projects. Here, one can specify various settings such as the type of task (e.g., llm finetuning, text classification, image classification), dataset, the model to use, and other training parameters. This step ensures that all necessary configurations are in place before starting the training process.

Dataset Processor: handles the preparation and preprocessing of datasets. It ensures that data is in the right format for training. This component can handle different types of data, including text, images, and tabular data. Dataset processor does cleaning and transformation of dataset, saves time and reduces the potential for errors. A dataset once processed can also be used for multiple projects without requiring to be processed again.

Trainer: is responsible for the actual training process. It manages the training loop, handles the computation of loss and metrics, and optimizes the model. The Trainer also supports distributed training, allowing you to train models on multiple GPUs seamlessly. Additionally, it includes tools for monitoring the training progress, ensuring that everything is running smoothly and efficiently.

3.2 Installation & Usage

Using *AutoTrain* is as easy as pie. In this section we focus briefly on installation and LLM finetuning task. However, the same can be applied to other tasks keeping in mind the dataset format which is provided

Installation AutoTrain Advanced can be easily installed using pip.

```
$ pip install autotrain-advanced
```

It has to be noted that the the pip installation doesnt install pytorch and users must install it on their own. However, a complete package with all the requirements is also available as a docker image.

```
$ docker pull
   huggingface/autotrain-advanced:latest
```

Usage AutoTrain Advanced offers CLI and UI. CLI is based on a AutoTrain Advanced python library. So, users familiar with python can also use the python sdk. To start the UI as shown in Figure 1, one can run the `autotrain app` command:

```
$ autotrain app
```

An example of running training in UI is shown in Figure 2.

Training can also be started using a config file which is in yaml format and the autotrain cli. An example config to finetune llama 3.1 is shown below:

```yaml
task: llm:orpo
base_model: meta-llama/Meta-Llama-3.1-8B
project_name: autotrain-llama
log: tensorboard
backend: local

data:
  path: HuggingFaceH4/no_robots
  train_split: train
  valid_split: null
  chat_template: zephyr
  column_mapping:
    text_column: chosen
    rejected_text_column: rejected
    prompt_text_column: prompt

params:
  block_size: 1024
  model_max_length: 8192
  max_prompt_length: 512
  epochs: 3
  batch_size: 2
  lr: 3e-5
  peft: true
  quantization: int4
```

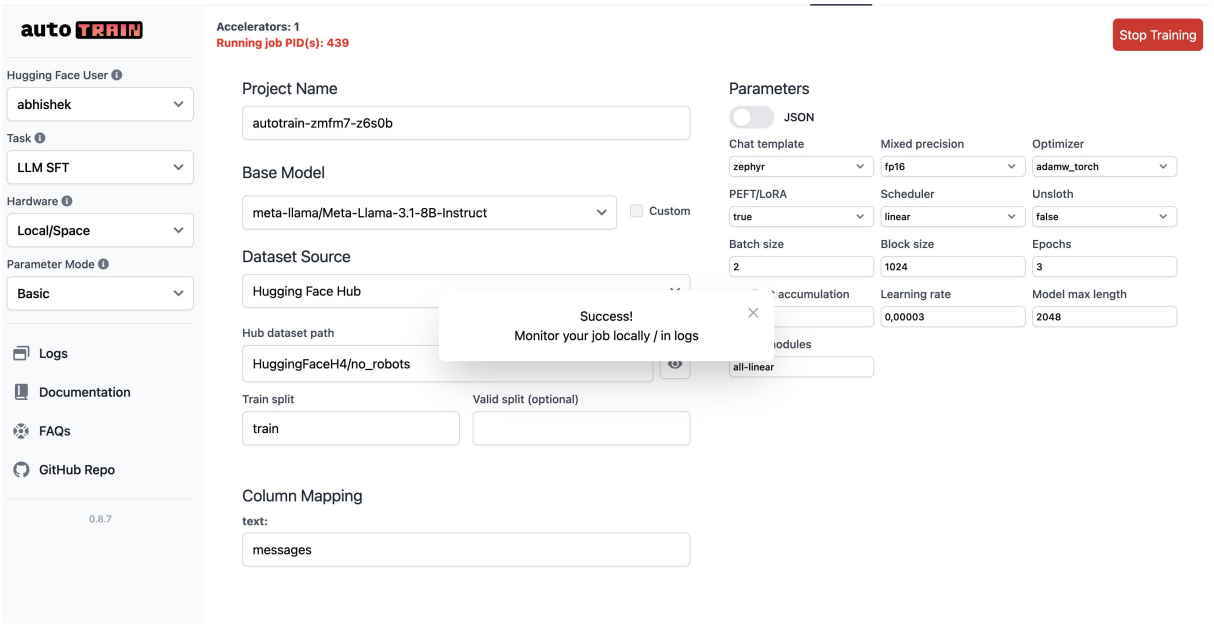

Figure 2: Finetuing an LLM in AutoTrain UI

```
  target_modules: all-linear
  padding: right
  optimizer: adamw_torch
  scheduler: linear
  gradient_accumulation: 4
  mixed_precision: fp16

hub:
  username: ${HF_USERNAME}
  token: ${HF_TOKEN}
  push_to_hub: true
```

The above config file shows how a `llama-3.1-8B` model from the Hugging Face hub can be finetuned on `HuggingFaceH4/no_robots` dataset which is also available on Hugging Face Hub. If the user wants to use a local dataset and model, they can do that too by following the documentation. In this specific case, a local dataset can be provided as a JSONL file. To start the training, `autotrain -config` command is used:

```
$ autotrain --config config.yml
```

The training process starts tensorboard (Abadi et al., 2015) which can be used to monitor the training and metrics and generated during the training process. The users can also monitor the training logs in terminal if they started the training using the CLI or in the UI logs section.

The trained model, depending on user's choice, can also be pushed to Hugging Face Hub, thus making it accessible to hundreds of thousands of users across the world. The trained models are also compatible with major inference providers (huggingface, aws, google cloud, etc.) which makes deployment and consumption easy for both coders and non-coders.

4 Conclusion

In this paper, we introduce *AutoTrain* (aka AutoTrain Advanced), which is an open source, no-code solution for training (or finetuning) machine learning models on a variety of tasks. AutoTrain addresses common challenges in the model training process, such as dataset processing, hyperparameter tuning, model validation, distributed training, monitoring, and maintenance. By automating these tasks, AutoTrain ensures that users can efficiently build high-performing models without needing extensive coding knowledge or experience. Additionally, AutoTrain supports a diverse range of tasks, including llm finetuning, text classification, image classification, and regression, and even tabular data classification/regression, thus, making it a versatile tool for various applications.

Limitations

AutoTrain tries to generalize the training process for a given model - dataset combination as much as possible, however, there might be situations in which custom changes might be required. For example, AutoTrain doesnt provide support for sample weights, model merging, or ensembling yet.

We are gathering issues faced by users and implementing them to address these limitations.

Acknowledgements

We thank the many contributors to the Hugging Face open source ecosystem. We also thank the different teams at Hugging Face: the open-source team, the infrastructure team, the hub team, frontend and backend teams and others.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM.

Axolotl AI Cloud. 2024. Axolotl: A tool for streamlining fine-tuning of ai models. `https://github.com/axolotl-ai-cloud/axolotl`. Accessed: 2024-08-06.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28 (2015)*, pages 2962–2970.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. `https://github.com/huggingface/accelerate`.

Haifeng Jin, François Chollet, Qingquan Song, and Xia Hu. 2023. Autokeras: An automl library for deep learning. *Journal of Machine Learning Research*, 24(6):1–6.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, et al. 2021. Datasets: A community library for natural language processing. *arXiv preprint arXiv:2109.02846*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Abhishek Thakur and Artus Krohn-Grimberghe. 2015. Autocompete: A framework for machine learning competition.

Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. 2014. scikit-image: image processing in python. *PeerJ*, 2:e453.

Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. 2022. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.