Transformers

Transformers acts as the model-definition framework for state-of-the-art machine learning models in text, computer vision, audio, video, and multimodal model, for both inference and training.

It centralizes the model definition so that this definition is agreed upon across the ecosystem. `transformers` is the pivot across frameworks: if a model definition is supported, it will be compatible with the majority of training frameworks (Axolotl, Unsloth, DeepSpeed, FSDP, PyTorch-Lightning, …), inference engines (vLLM, SGLang, TGI, …), and adjacent modeling libraries (llama.cpp, mlx, …) which leverage the model definition from `transformers`.

We pledge to help support new state-of-the-art models and democratize their usage by having their model definition be simple, customizable, and efficient.

There are over 1M+ Transformers model checkpoints on the Hugging Face Hub you can use.
Explore the Hub today to find a model and use Transformers to help you get started right away.

Features

Transformers provides everything you need for inference or training with state-of-the-art pretrained models. Some of the main features include:

- Pipeline: Simple and optimized inference class for many machine learning tasks like text generation, image segmentation, automatic speech recognition, document question answering, and more.
- Trainer: A comprehensive trainer that supports features such as mixed precision, torch.compile, and FlashAttention for training and distributed training for PyTorch models.
- generate: Fast text generation with large language models (LLMs) and vision language models (VLMs), including support for streaming and multiple decoding strategies.

Design

Read our Philosophy to learn more about Transformers' design principles.

Transformers is designed for developers and machine learning engineers and researchers. Its main design principles are:

1. Fast and easy to use: Every model is implemented from only three main classes (configuration, model, and preprocessor) and can be quickly used for inference or training with Pipeline or Trainer.
2. Pretrained models: Reduce your carbon footprint, compute cost and time by using a pretrained model instead of training an entirely new one. Each pretrained model is reproduced as closely as possible to the original model and offers state-of-the-art performance.

Learn

If you're new to Transformers or want to learn more about transformer models, we recommend starting with the LLM course. This comprehensive course covers everything from the fundamentals of how transformer models work to practical applications across various tasks. You'll learn the complete workflow, from curating high-quality datasets to fine-tuning large language models and implementing reasoning capabilities. The course contains both theoretical and hands-on exercises to build a solid foundational knowledge of transformer models as you learn.

Installation

Transformers works with PyTorch, TensorFlow 2.0, and Flax. It has been tested on Python 3.9+, PyTorch 2.1+, TensorFlow 2.6+, and Flax 0.4.1+.

Virtual environment

A virtual environment helps manage different projects and avoids compatibility issues between dependencies. Take a look at the Install packages in a virtual environment using pip and venv guide if you're unfamiliar with Python virtual environments.
Create and activate a virtual environment in your project directory with venv.

Copied

```
python -m venv .env
source .env/bin/activate
```

Python

You can install Transformers with pip or uv.

pip is a package installer for Python. Install Transformers with pip in your newly created virtual environment.

Copied

```
pip install transformers
```

For GPU acceleration, install the appropriate CUDA drivers for PyTorch and TensorFlow.

Run the command below to check if your system detects an NVIDIA GPU.

Copied

```
nvidia-smi
```

To install a CPU-only version of Transformers and a machine learning framework, run the following command.

Copied

```
pip install 'transformers[torch]'
uv pip install 'transformers[torch]'
```

Test whether the install was successful with the following command. It should return a label and score for the provided text.

Copied

```
python -c "from transformers import pipeline; print(pipeline('sentiment-analysis')('hugging face is the best'))"
[{'label': 'POSITIVE', 'score': 0.9998704791069031}]
```

Source install

Installing from source installs the *latest* version rather than the *stable* version of the library. It ensures you have the most up-to-date changes in Transformers and it's useful for experimenting with the latest features or fixing a bug that hasn't been officially released in the stable version yet.
The downside is that the latest version may not always be stable. If you encounter any problems, please open a GitHub Issue so we can fix it as soon as possible.

Install from source with the following command.

Copied

```
pip install git+https://github.com/huggingface/transformers
```

Check if the install was successful with the command below. It should return a label and score for the provided text.

Copied

```
python -c "from transformers import pipeline; print(pipeline('sentiment-analysis')('hugging face is the best'))"
[{'label': 'POSITIVE', 'score': 0.9998704791069031}]
```

Editable install

An editable install is useful if you're developing locally with Transformers. It links your local copy of Transformers to the Transformers repository instead of copying the files. The files are added to Python's import path.

Copied

```
git clone https://github.com/huggingface/transformers.git
cd transformers
pip install -e .
```

You must keep the local Transformers folder to keep using it.

Update your local version of Transformers with the latest changes in the main repository with the following command.

Copied

```
cd ~/transformers/
git pull
```

conda

conda is a language-agnostic package manager. Install Transformers from the conda-forge channel in your newly created virtual environment.

Copied

```
conda install conda-forge::transformers
```

Set up

After installation, you can configure the Transformers cache location or set up the library for offline usage.

Cache directory

When you load a pretrained model with from_pretrained(), the model is downloaded from the Hub and locally cached.

Every time you load a model, it checks whether the cached model is up-to-date. If it's the same, then the local model is loaded. If it's not the same, the newer model is downloaded and cached.

The default directory given by the shell environment variable `TRANSFORMERS_CACHE` is `~/.cache/huggingface/hub`. On Windows, the default directory is `C:\Users\username\.cache\huggingface\hub`.

Cache a model in a different directory by changing the path in the following shell environment variables (listed by priority).

1. HF_HUB_CACHE or TRANSFORMERS_CACHE (default)
2. HF_HOME
3. XDG_CACHE_HOME + /huggingface (only if HF_HOME is not set)
   Older versions of Transformers uses the shell environment variables PYTORCH_TRANSFORMERS_CACHE or PYTORCH_PRETRAINED_BERT_CACHE. You should keep these unless you specify the newer shell environment variable TRANSFORMERS_CACHE.

Offline mode

To use Transformers in an offline or firewalled environment requires the downloaded and cached files ahead of time. Download a model repository from the Hub with the snapshot_download method.

Refer to the Download files from the Hub guide for more options for downloading files from the Hub. You can download files from specific revisions, download from the CLI, and even filter which files to download from a repository.

Copied

```
from huggingface_hub import snapshot_download

snapshot_download(repo_id="meta-llama/Llama-2-7b-hf", repo_type="model")
```

Set the environment variable HF_HUB_OFFLINE=1 to prevent HTTP calls to the Hub when loading a model.

Copied

```
HF_HUB_OFFLINE=1 \

python examples/pytorch/language-modeling/run_clm.py --model_name_or_path
meta-llama/Llama-2-7b-hf --dataset_name wikitext ...
```

Another option for only loading cached files is to set local_files_only=True in from_pretrained().

Copied

```
from transformers import LlamaForCausalLM

model = LlamaForCausalLM.from_pretrained("./path/to/local/directory",
local_files_only=True)
```