ADALTAS

Distributed Machine Learning

Petra KAFERLE DEVISSCHERE

Scaling of compute

- **Vertical (Scale-up)**
    - addition of hardware resources to the existing computer (CPUs, RAM…)
    - for ML:
        - GPUs
        - Single Instruction Multiple Data (SIMD) architecture
        - general-purpose accelerators
    - Application Specific Circuits (ASICs)
        - circuit optimized for a specific task
        - example: Tensor Processing Unit (TPU) to accelerate tensor-based calculations in TensorFlow
        - TPU uses Multiple Instructions, Multiple Data (MIMD) architecture
- **Horizontal (Scale-out)**
    - cheaper than scale-up
    - better resilience in the case of failure
    - data read/writes are parallelized better over multiple machines
    - problem: not all ML algorithms can be parallelized in this manner

https://arxiv.org/pdf/1912.09789.pdf

Distributed ML Ecosystem

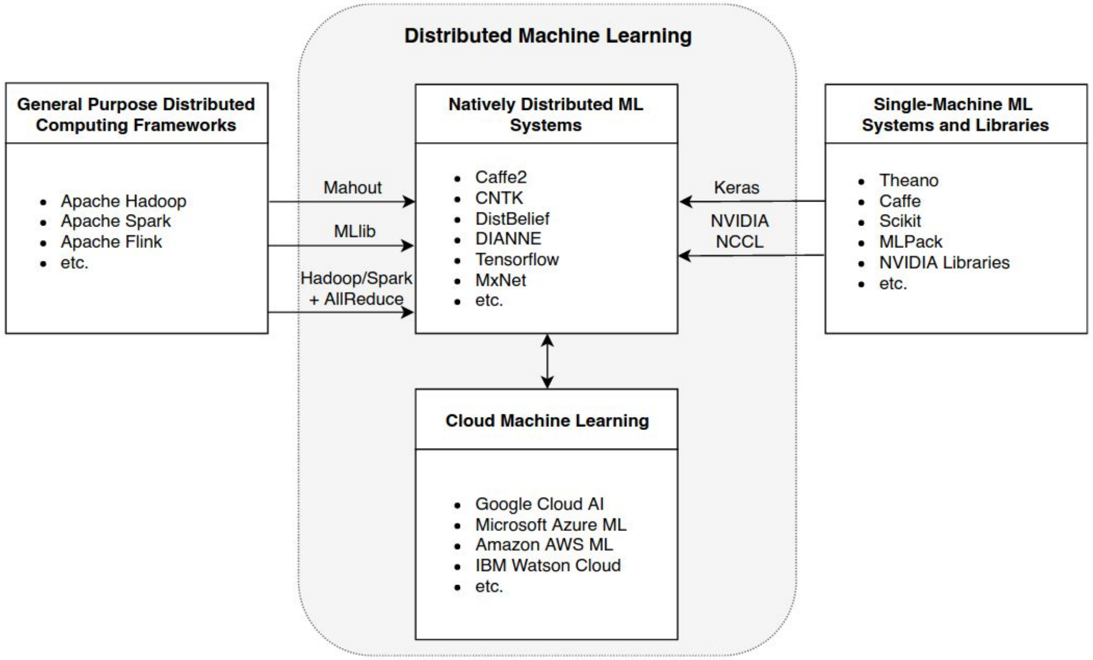


Fig. 4. Distributed Machine Learning Ecosystem. Both general purpose distributed frameworks and single-machine ML systems and libraries are converging towards Distributed Machine Learning. Cloud emerges as a new delivery model for ML.

https://arxiv.org/pdf/1912.09789.pdf

General overview of ML

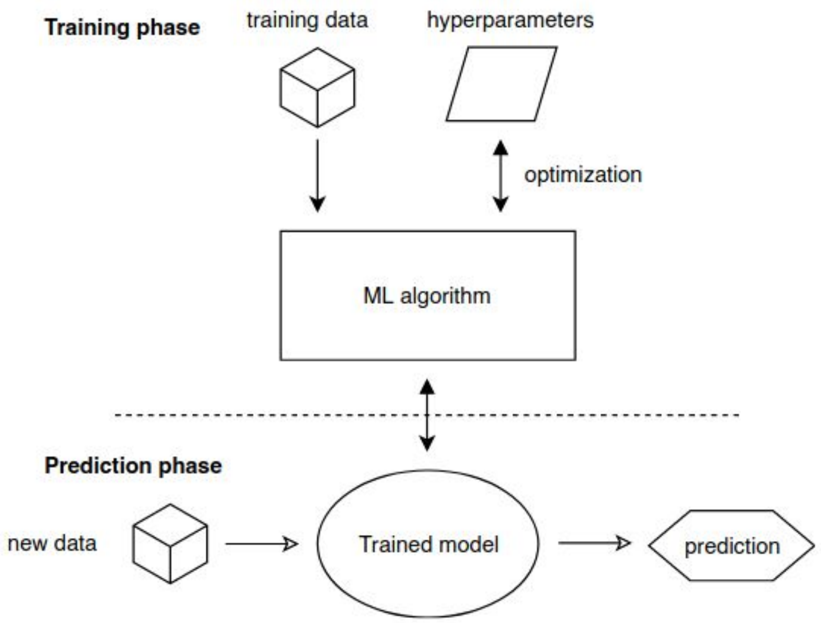


Fig. 1. General Overview of Machine Learning. During the training phase a ML model is optimized using training data and by tuning hyper parameters. Then the trained model is deployed to provide predictions for new data fed into the system.

Parallelism in distributed ML

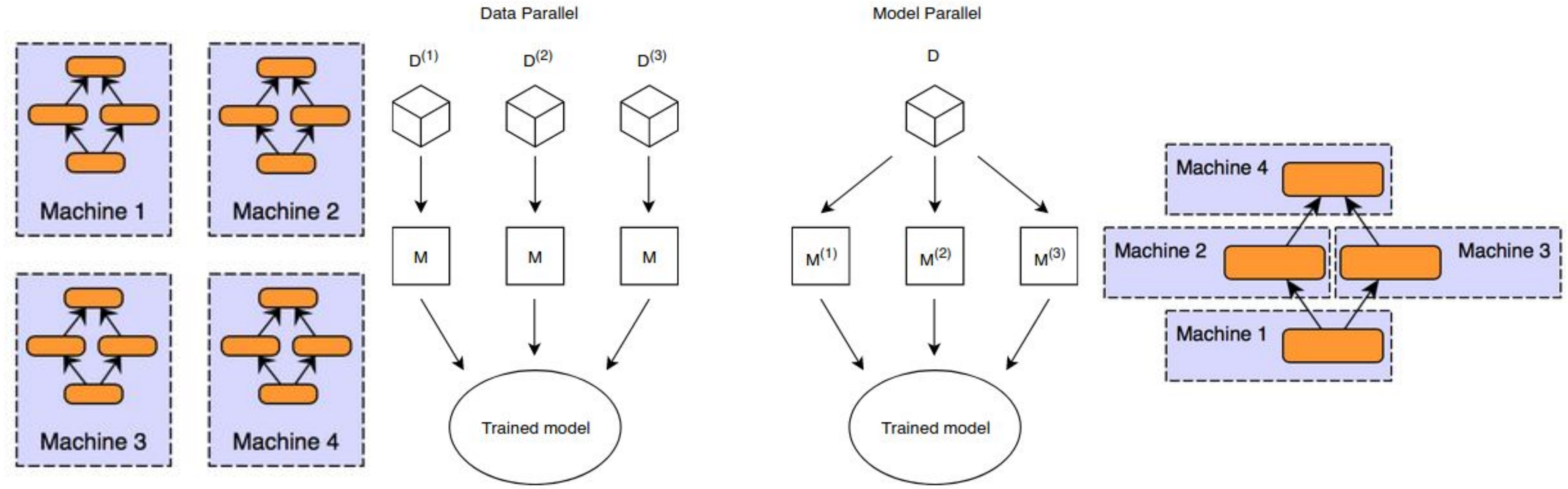


Fig. 2. Parallelism in Distributed Machine Learning. Data parallelism trains multiple instances of the same model on different subsets of the training dataset, while model parallelism distributes parallel paths of a single model to multiple nodes.

https://xiandong79.github.io/Intro-Distributed-Deep-Learning

Why data parallelism?

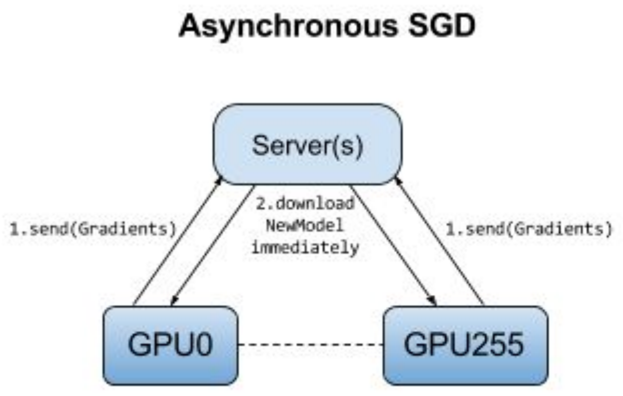
- Dataset doesn’t fit in the RAM
- More data gives better results than fine-tuning a model

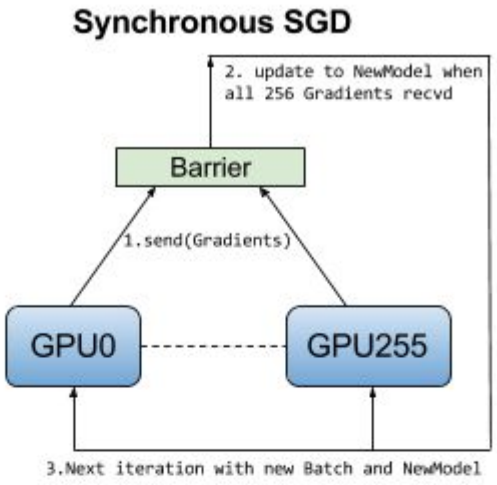
Synchronisation of parameters

After each calculation, the new parameters need to be communicated to the parameter server:

- synchronous manner:
    - the model update happens after all of the workers submitted their results
    - hard to scale
    - can result in idle resources
- asynchronous manner
    - the model update happens immediately, but not with all the params
    - only one model at the time is up-to-date
    - poor implementation can slow down the training

Example: Synchronous and asynchronous SGD




Synchronous SGD
2. update to NewModel when all 256 Gradients recvd
Barrier
1.send(Gradients)
GPU0
GPU255
3.Next iteration with new Batch and NewModel

Spark:

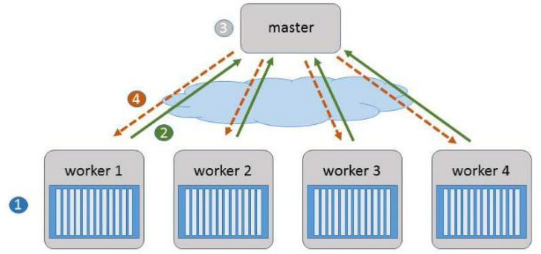


Figure 1: Four-stage algorithmic pattern for synchronous distributed learning algorithms. Arrows indicate the synchronous communication per round.

https://www.oreilly.com/content/distributed-tensorflow/

https://arxiv.org/pdf/1612.01437.pdf

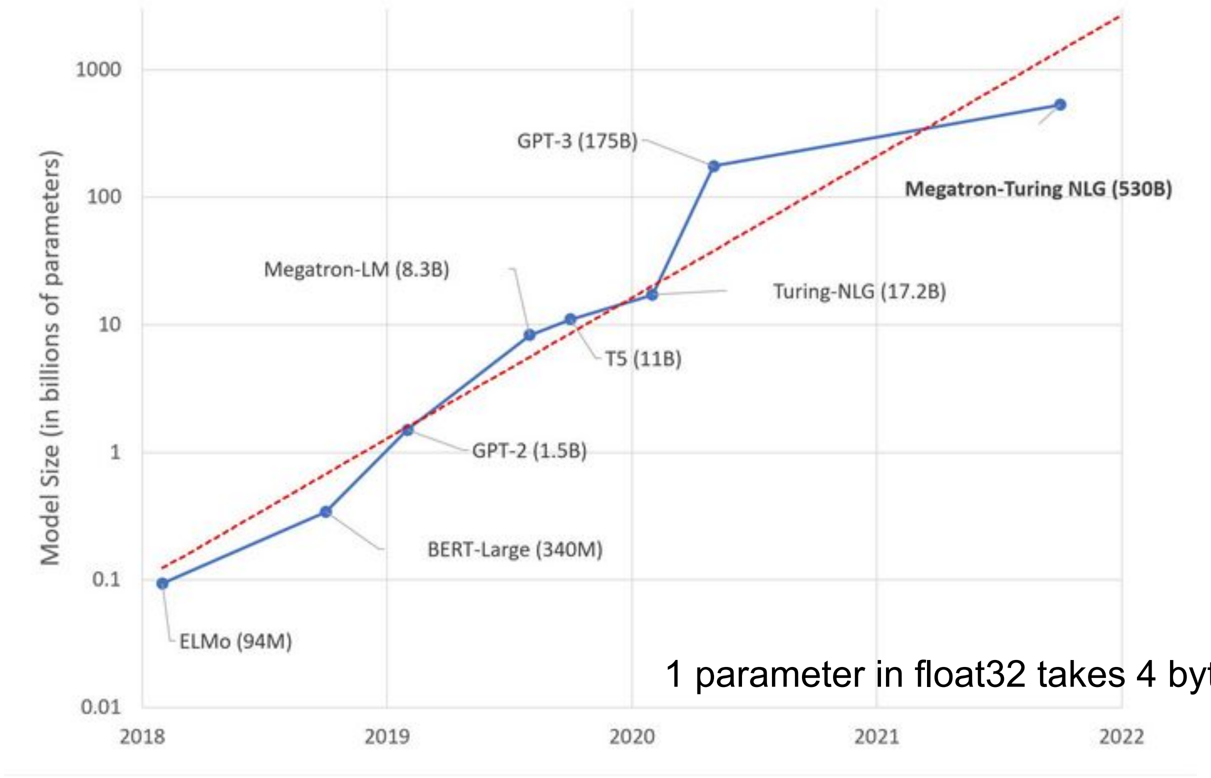
Data parallelism - Summary

1. **Dataset** is partitioned and **distributed** across many workers
2. Each worker also has (random) **initial model parameters**
3. On each worker, the **same algorithm** calculates new model parameters
4. The **parameters** are **returned** to the parameter server
    i. In case of Spark, Spark driver serves as parameter server
5. Parameter server **aggregates** the parameters and **updates** the model

Why model parallelism?

- Model doesn’t fit in the RAM
- Mostly used in Deep Learning, where models can be very big

Number of parameters in NLP models



1 parameter in float32 takes 4 bytes in memory

https://huggingface.co/blog/large-language-models

Example: Model parallelism

ADALTAS

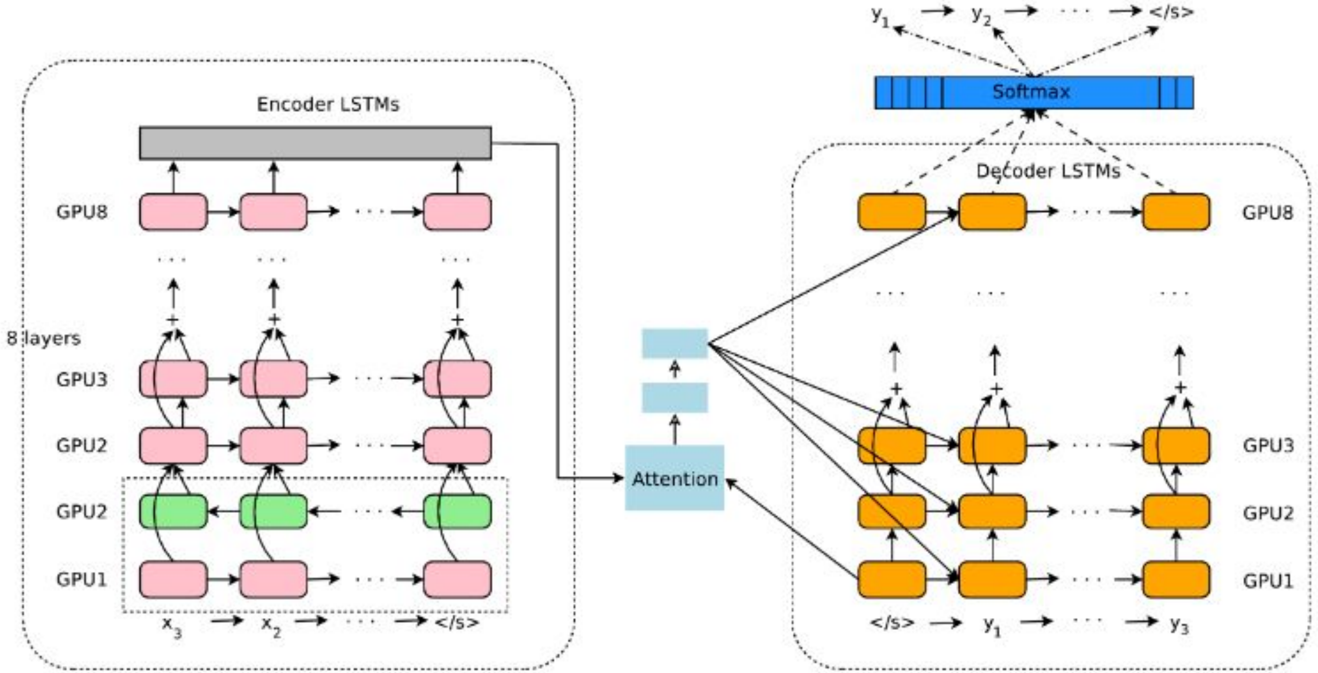


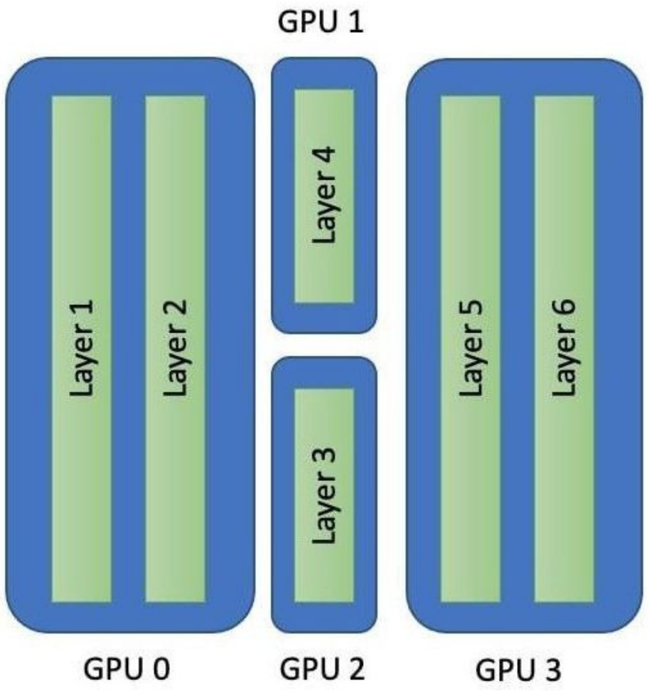
Figure 1: The model architecture of GNMT, Google's Neural Machine Translation system. On the left is the encoder network, on the right is the decoder network, in the middle is the attention module. The

https://arxiv.org/pdf/1609.08144.pdf

What is parallel in model parallelism?

ADALTAS

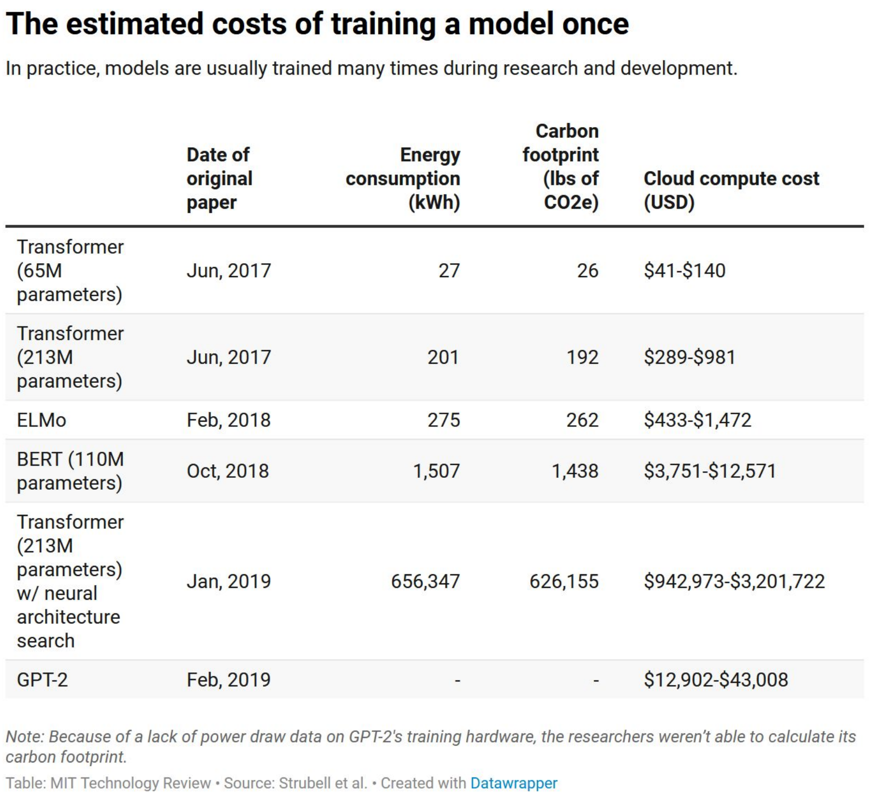
- It's mostly distributed (but not necessarily parallel)
- Passing parameters between the layers can be:
  - synchronous
    - only one GPU working at the time
    - slow, sequential
  - asynchronous
    - fast, parallel



Model parallelism - Summary

1. **Model** is split in parts and **distributed** across many workers (or GPUs)
2. Each worker also has **the same dataset**
3. Since the output of the n-th worker will be used as input of n+1-th worker, the training is mostly **sequential**
4. Asynchronous training can offer a speed-up (but not all algorithms support it)

Models and their environmental impact



The estimated costs of training a model once

In practice, models are usually trained many times during research and development.

	Date of original paper	Energy consumption (kWh)	Carbon footprint (lbs of CO2e)	Cloud compute cost (USD)
Transformer (65M parameters)	Jun, 2017	27	26	$41-$140
Transformer (213M parameters)	Jun, 2017	201	192	$289-$981
ELMo	Feb, 2018	275	262	$433-$1,472
BERT (110M parameters)	Oct, 2018	1,507	1,438	$3,751-$12,571
Transformer (213M parameters) w/ neural architecture search	Jan, 2019	656,347	626,155	$942,973-$3,201,722
GPT-2	Feb, 2019	-	-	$12,902-$43,008

Note: Because of a lack of power draw data on GPT-2's training hardware, the researchers weren't able to calculate its carbon footprint.

Table: MIT Technology Review • Source: Strubell et al. • Created with Datawrapper

Common carbon footprint benchmarks

in lbs of CO2 equivalent

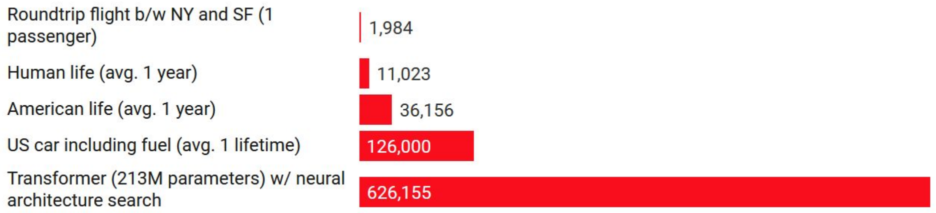


Chart: MIT Technology Review • Source: Strubell et al. • Created with Datawrapper

https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/