⚖️

Pytorch and NNs (1)

https://youtube.com/playlist?list=PLKnIA16_Rmvboy8bmDCjwNHgTaYH2puK7&si=CDuAGicJy77ZAgTy

https://www.eletreby.me/blog/getting-started-with-pytorch-dataset-and-dataloader

https://www.eletreby.me/blog/understanding-pytorch-computational-graphs

https://www.eletreby.me/blog/training-loop

- Its an open source deep learning library
- Updates in pytorch over the years - Torchscipt - serialisation - ONIX (open neural netwrok exchange , can use model from any lib to any lib)
- Keras is a high level api for tensforlow similaror pytorch lighting is for pytorch.
- TPU - tesnfor processing unit
- Core features of pytorch :
  - Tensor computation
  - GPU acceleration
  - dyanmic comp graph (pure NN ko as a graph rep krte)- easy to debug
  - automatic differnetiation - jitne bhi NN - backpropogation - calcualting gradiants - autograd
  - distributed training
  - interperability wiht other libraries (work with other libs , also ONNX support)
- torch vs TF :
  - torch mainly supports python ; tf is compatible with many langs cpp, java, swift, js)
  - deployment tf wins
  - both offer high perfs - optimized dynamic graphs ; pehle static comp grpah the but fir
  - High level api -
  fast.ai and pytorch lighting ; keras for tf

- Mobile and embedded model - pytorch is still developing but tf wins here - tensofrlow lite and tensorflow.js for browser.
  - pytorch is easier to learn than tf

-

Core PyTorch Modules

Module	Description
`torch`	The core module providing multidimensional arrays (tensors) and mathematical operations on them.
`torch.autograd`	Automatic differentiation engine that records operations on tensors to compute gradients for optimization.
`torch.nn`	Provides a neural networks library, including layers, activations, loss functions, and utilities to build deep learning models.
`torch.optim`	Contains optimization algorithms (optimizers) like SGD, Adam, and RMSprop used for training neural networks.
`torch.utils.data`	Utilities for data handling, including the `Dataset` and `DataLoader` classes for managing and loading datasets efficiently.
`torch.jit`	Supports Just-In-Time (JIT) compilation and TorchScript for optimizing models and enabling deployment without Python dependencies.
`torch.distributed`	Tools for distributed training across multiple GPUs and machines, facilitating parallel computation.
`torch.cuda`	Interfaces with NVIDIA CUDA to enable GPU acceleration for tensor computations and model training.
`torch.backends`	Contains settings and allows control over backend libraries like cuDNN, MKL, and others for performance tuning.
`torch.multiprocessing`	Utilities for parallelism using multiprocessing, similar to Python's `multiprocessing` module but with support for CUDA tensors.
`torch.quantization`	Tools for model quantization to reduce model size and improve inference speed, especially on edge devices.
`torch.onnx`	Supports exporting PyTorch models to the ONNX (Open Neural Network Exchange) format for interoperability with other frameworks and deployment.

PyTorch Domain Libraries

Library	Description
`torchvision`	Provides datasets, model architectures, and image transformations for computer vision tasks.
`torchtext`	Tools and datasets for natural language processing (NLP), including data preprocessing and vocabulary management.
`torchaudio`	Utilities for audio processing tasks, including I/O, transforms, and pre-trained models for speech recognition.
`torcharrow`	A library for accelerated data loading and preprocessing, especially for tabular and time series data (experimental).
`torchserve`	A PyTorch model serving library that makes it easy to deploy trained models at scale in production environments.
`pytorch_lightning`	A lightweight wrapper for PyTorch that simplifies the training loop and reduces boilerplate code, enabling scalable and reproducible models.

Popular PyTorch Ecosystem Libraries

Library	Description
Hugging Face Transformers	Provides state-of-the-art pre-trained models for NLP tasks like text classification, translation, and question answering, built on PyTorch.
Fastai	High-level library that simplifies training fast and accurate neural nets using modern best practices, built on top of PyTorch.
PyTorch Geometric	Extension library for geometric deep learning, including graph neural networks and 3D data processing.
TorchMetrics	A modular metrics API for PyTorch, compatible with PyTorch Lightning and provides standardized implementations of many common metrics.
TorchElastic	Enables dynamic scaling of PyTorch distributed training jobs, allowing for elasticity in resource management.
Optuna	An automatic hyperparameter optimization software framework, integrating well with PyTorch for tuning models.

Catalyst	Provides high-level features for training neural networks, focusing on reproducibility and fast experimentation.
Ignite	High-level library to help with training neural networks in PyTorch, offering a lightweight engine for training and evaluating models.
AllenNLP	An NLP research library built on PyTorch, designed to support researchers in deep learning for NLP.
Skorch	A scikit-learn compatible wrapper for PyTorch that allows the use of PyTorch models with scikit-learn utilities and APIs.
PyTorch Forecasting	High-level library for time series forecasting, making it easy to build, train, and evaluate complex models.
TensorBoard for PyTorch	Allows visualization of training metrics, model graphs, and other useful data within TensorBoard for PyTorch models.

Tensors in PyTorch (v imp fundamental topic)

- Tensors are kinda like Data structures ; multi dim array designed for mathematical and comp eff.
- 0 Dimensional Tensor - scalar(a single number) - eg: loss value after a forward pass [yhat - y]
- 1-dim tensor (a list of num) - arrays - eg embeddings
- 2-dim vector (2d grid of numbs) - a greyscale image; 3dim -rgb images(3 channels) ; 4d tensor - batches of rgb images ; 5d tensor - video data - combination of multiple images- batch of 10 videos.

- Why is it useful tho? - tensor arithmatic can be performed in parallel in gpus

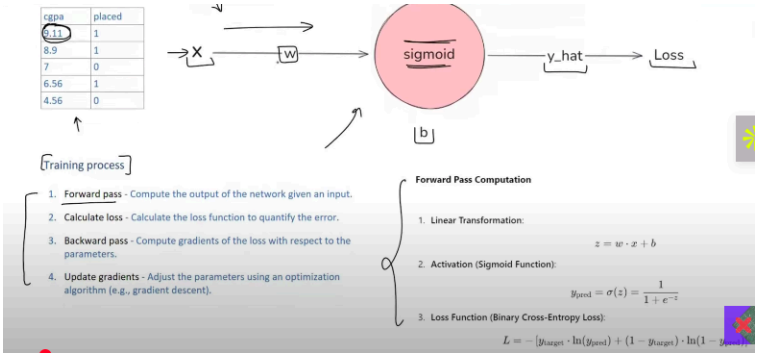
---

- Where are tensors used?
  - training data (images , text etc)
  - weights and biases - learnable parameters stores in tensors (Wx+b) - forward pass
  - during backprop - gradiant calculation

> https://colab.research.google.com/drive/1rexzXduHHRHJqggGPEDhI82LHt2X-PLA

AutoGrad - automatic diffing tool (torch)

- Its easy to code a derivative of function like x**2 , sin(y) - nested functions have difficulat to calc derivates
- Why are we studying this tho - nested functions and calc derivatives in deeply connected to deep learning.
-



w = weight , b = bias
- Z is fed into a activation function = ypred = wx+b
- get loss - L = -[ytarget*ln(ypred) + (1-ytarget)*ln(1-ypred)] =
Binary cross entropy loss.