

The pytorch ecosystem

Machine Learning Operations

Nicki Skafte Detlefsen,

Postdoc

DTU Compute

The ecosystem



Collection of frameworks
build to be used in
combination with
Pytorch

PyTorch

Get Started Ecosystem Mobile Blog Tutorials Docs ▾ Resources ▾ GitHub 🔍

ECOSYSTEM TOOLS

Tap into a rich ecosystem of tools, libraries, and more to support, accelerate, and explore AI development.

[Join the Ecosystem](#)

Sort ▾

PyTorch-NLP 🔒 1.9k

Basic Utilities for PyTorch Natural Language Processing (NLP).

DeepSpeed 🔒 4.6k

DeepSpeed is a deep learning optimization library that makes distributed training easy, efficient, and effective.

Albumentations 🔒 7.6k

Fast and extensible image augmentation library for different CV tasks like classification, segmentation, object detection and pose estimation.

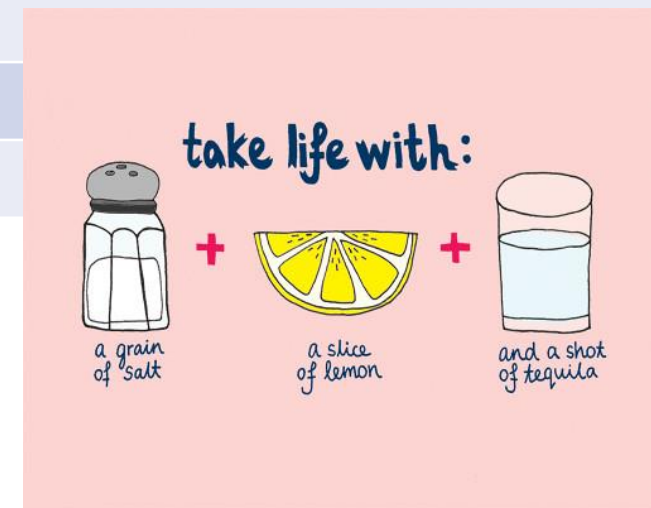
Captum 🔒 2.2k

Captum ("comprehension" in Latin) is an open source, extensible library for model interpretability built on PyTorch.

Framework categorizing



Data specific frameworks	Training frameworks	Utility frameworks
Transformers	fastai	Albumentations
Detectron2	Ray	PySyft
Pytorch geometric	Pytorch Lightning	Pyro
Flair	Horovod	Optuna
AllenNLP	DeepSpeed	Hydra
ParlAI	ONNX Runtime	Pytorch Metric Learning
DGL	skorch	Einops
PyTorch3D	Ignite	
MMF	Polyaxon	
Kornia		

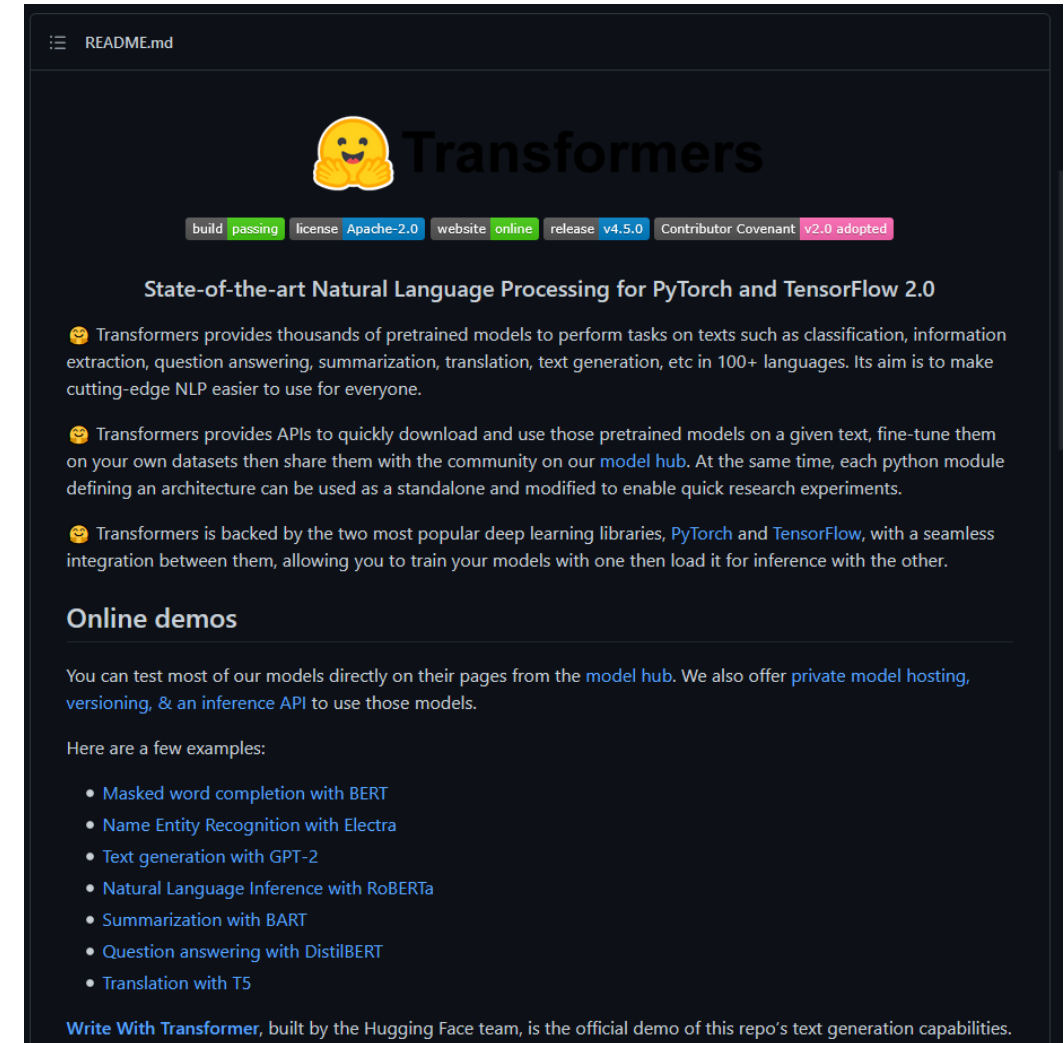


Project 1: NLP



Framework: Transformers (Huggingface)

- <https://github.com/huggingface/transformers>
- State-of-the-art NLP models
- Most starred framework in the ecosystem



Project 2: CV



Framework: Kornia

- <https://github.com/kornia/kornia>
- Differentiable computer vision algorithms

The screenshot shows the GitHub README for the Kornia library. At the top is the Kornia logo, a 3D cube with red, green, and blue faces. Below the logo is a horizontal bar with various status indicators: License (Apache 2.0), tests-cpu-versions (passing), tests-cuda-versions (failing), codecov (96%), pypi package (0.5.0), and docs (passing). The main text describes Kornia as a differentiable computer vision library for PyTorch, consisting of routines and modules for solving generic computer vision problems. It uses PyTorch as its main backend for efficiency and reverse-mode auto-differentiation. Below the text is a code block showing a simple example of using Kornia to rotate a video frame. To the right of the code is a 2x2 grid of images showing a cow on a bridge with a full moon in the background, demonstrating the effect of random rotation. The bottom section is titled 'Overview' and describes the library's composition and its focus on tensor-based operations.

README.md

kornia

License: Apache 2.0 tests-cpu-versions: passing tests-cuda-versions: failing codecov: 96% pypi package: 0.5.0 docs: passing

Kornia is a differentiable computer vision library for PyTorch.

It consists of a set of routines and differentiable modules to solve generic computer vision problems. At its core, the package uses *PyTorch* as its main backend both for efficiency and to take advantage of the reverse-mode auto-differentiation to define and compute the gradient of complex functions.

```
import torch
import kornia

frame: torch.Tensor = load_video_frame(...)

out: torch.Tensor = (
    kornia.random_rotation(frame, 15.)
)
```

Overview

Inspired by existing packages, this library is composed by a subset of packages containing operators that can be inserted within neural networks to train models to perform image transformations, epipolar geometry, depth estimation, and low-level image processing such as filtering and edge detection that operate directly on tensors.

At a granular level, Kornia is a library that consists of the following components:

Project 3: Graphs and points



Framework: Pytorch Geometric

- https://github.com/rusty1s/pytorch_geometric
- Neural networks on graphs and point clouds



Getting a good idea



master 11 branches 16 tags Go to file Add file Code

edgarriba update new kornia logo e36ca3d 2 days ago 1,533 commits

.circleci	upgrade ci workflow with pytorch 1.8 (#892)	29 days ago
.github	Create CODEOWNERS (#947)	2 days ago
docker	[Feat] Add tpu support for the losses module (#834)	3 months ago
docs	update new kornia logo	2 days ago
examples	Updated doc & example for augmentation (#583)	8 months ago
kornia	Fixed the issue of NaN gradients by adding epsilon in focal loss (#924)	2 days ago
packaging	remove pytorch version variable	8 months ago
test	Deprecate some augmentation functionals (#943)	2 days ago
tutorials	Fixed tests and docs (#654)	7 months ago
.codecov.yml	Create .codecov.yml (#735)	6 months ago
.gitconfig	reorganize color module	2 years ago
.gitignore	Update gitignore to avoid version.py	2 years ago
CHANGELOG.md	create CHANGELOG and update for 0.4.1 (#726)	6 months ago
CITATION.md	Create CITATION.md (#949)	2 days ago
CODE_OF_CONDUCT.md	add code of conduct file	2 years ago
CONTRIBUTING.rst	Update CONTRIBUTING.rst (#316)	17 months ago

About

Open Source Differentiable Computer Vision Library for PyTorch

kornia.org

machine-learning computer-vision image-processing pytorch

Readme

View license

Releases 16

Morphological operators, Dee... Latest 21 days ago

+ 15 releases

Packages

No packages published

Used by 290

+ 282

Getting a good idea



Projects - dtu_mlops

Kaggle: Your Machine Learning

https://www.kaggle.com

kaggle Competitions Datasets Code Discussions Courses

Search Sign In Register

Start with more than a blinking cursor

Kaggle offers a no-setup, customizable, Jupyter Notebooks environment. Access free GPUs and a huge repository of community published data & code.

REGISTER WITH GOOGLE

Register with Email

Predict Malicious Websites: XGBoost

Python notebook using data from Malicious and benign Websites - 4 views

Version 6

This kernel has an XGBoost model that predicts whether a website is malicious or not.

```
In [1]: import numpy as np
import pandas as pd
import xgboost as xgb

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.utils.multiclass import unique_labels

data = pd.read_csv("../input/dataset.csv")

# clean up column names
data.columns = data.columns.\
    str.strip().\
    str.lower()

# remove non-numeric columns
data = data.select_dtypes(['number'])

# split data into training & testing
train, test = train_test_split(data, shuffle=True)

# peak @ dataframe
train.head()
```

	url_length	number_special_characters	content_length	tcp_conversation_exchange	dist_remote_tcp_port	remote_ip	app_byte
344	37	9	102.0	1	0	1	66
77	26	5	NaN	0	0	0	0
1093	63	10	733.0	2	1	5	700

Inside Kaggle you'll find all the code & data you need to do your data science work. Use over 50,000 public datasets and 400,000 public notebooks to conquer any analysis in no time.

We use cookies on Kaggle to deliver our services, analyze web traffic, and improve your experience on the site. By using Kaggle, you agree to our use of cookies.

<https://www.kaggle.com/code/new>

Got it Learn more

Summary



- Pick a framework (try running their notebooks/examples!):
 - Project 1: NLP
 - Project 2: CV
 - Project 3: Graphs and points
- Brainstorm a project. It does not have to be particularly big as you only have 4½ full days for working on it
- Write a small (max 1 page) project description including:
 - What model do intend to implement
 - What data are you going to use
 - How you think the chosen framework can be incorporated

Checklist (also in todays readme)



- - [] Create a git repository
- - [] Make sure that all team members have write access to the github repository
- - [] Create a dedicated environment for you project to keep track of your packages (using conda)
- - [] Create the initial file structure using cookiecutter
- - [] Fill out the `make_dataset.py` file such that it downloads whatever data you need and
- - [] Add a model file and a training script and get that running
- - [] Remember to fill out the `requirements.txt` file with whatever dependencies that you are using
- - [] Remember to comply with good coding practices (`pep8`) while doing the project
- - [] Do a bit of code typing and remember to document essential parts of your code
- - [] Setup version control for your data or part of your data
- - [] Construct one or multiple docker files for your code
- - [] Build the docker files locally and make sure they work as intended
- - [] Write one or multiple configurations files for your experiments
- - [] Used Hydra to load the configurations and manage your hyperparameters
- - [] When you have something that works somewhat, remember at some point to do some profiling and see if you can optimize your code
- - [] Use wandb to log training progress and other important metrics/artifacts in your code
- - [] Use pytorch-lightning (if applicable) to reduce the amount of boilerplate in your code

Hand-in



- By 17:00 today handin link to github repository on DTU Learn

Exam format



Thursday 24/6 – evaluation by Nicki and Søren

- Group presentation
 - 6 minutes of powerpoint showcase
 - 10 minutes of discussion
- What you will be evaluated on:
 - How well you have included what we teach you in the course
- What you will NOT be evaluated on
 - How epic your deep learning model is

Some good advice



1. Document everything
 - Take screenshots of your work

2. Parallize work
 - Many of the checkpoints are independent of each other

Meme of the day

**When someone asks why you never stops
talking about machine learning**

