

---

# Projects

02476 Machine Learning Operations  
Nicki Skafte Detlefsen

---

# The “case”

You are just hired as an MLOps engineer at an start-up.

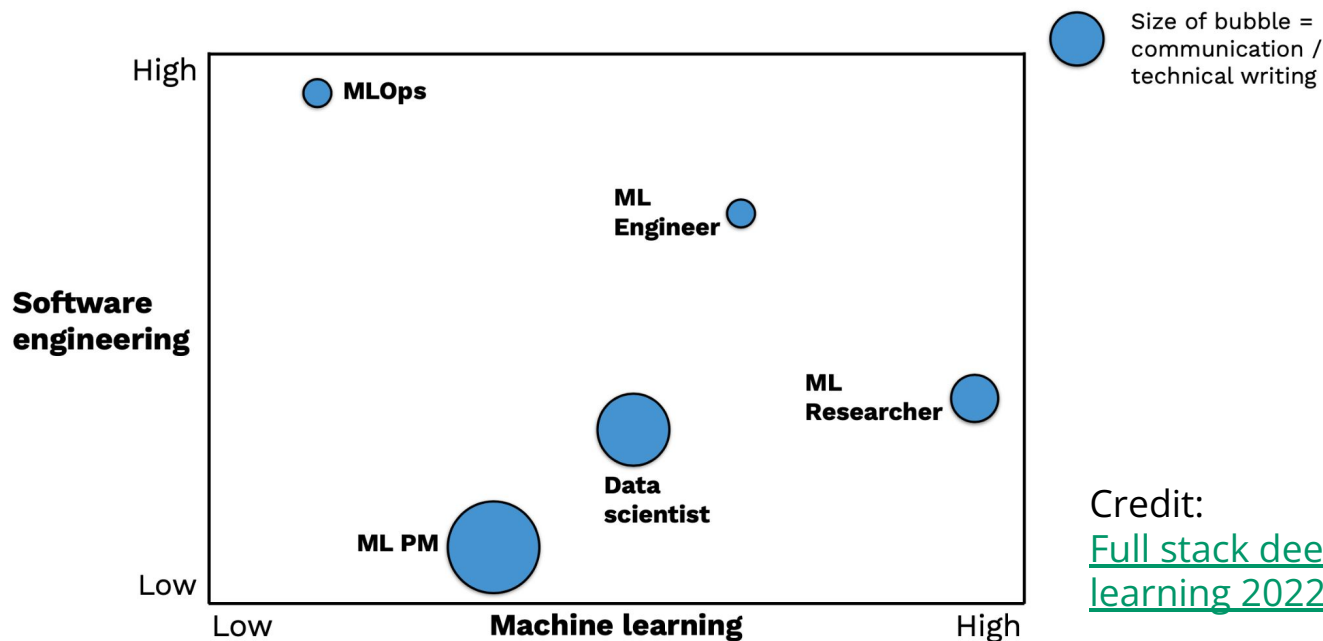
Your first job:

*Develop an MLOps pipeline to solve a specific task for the company*

Importantly: You are judged not by how great the model is but how fast you can setup a pipeline to solve the task.

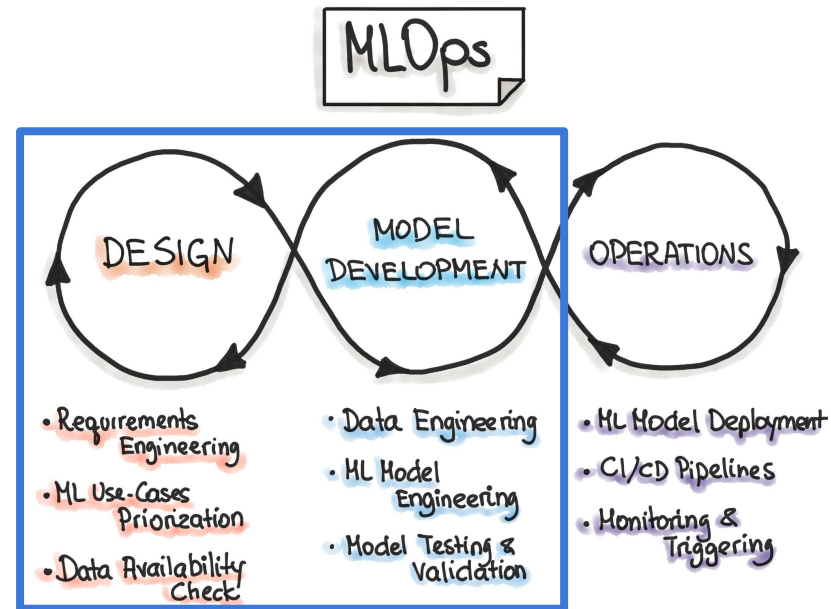
# Why you should not care about the model

You do not need to focus on the model, that's the ML researchers job



# How to solve the problem

- You already have all the tools for the pipeline, you just need a good starting model.
- Your base framework is Pytorch
- You turn your attention towards open-source projects build on top of Pytorch

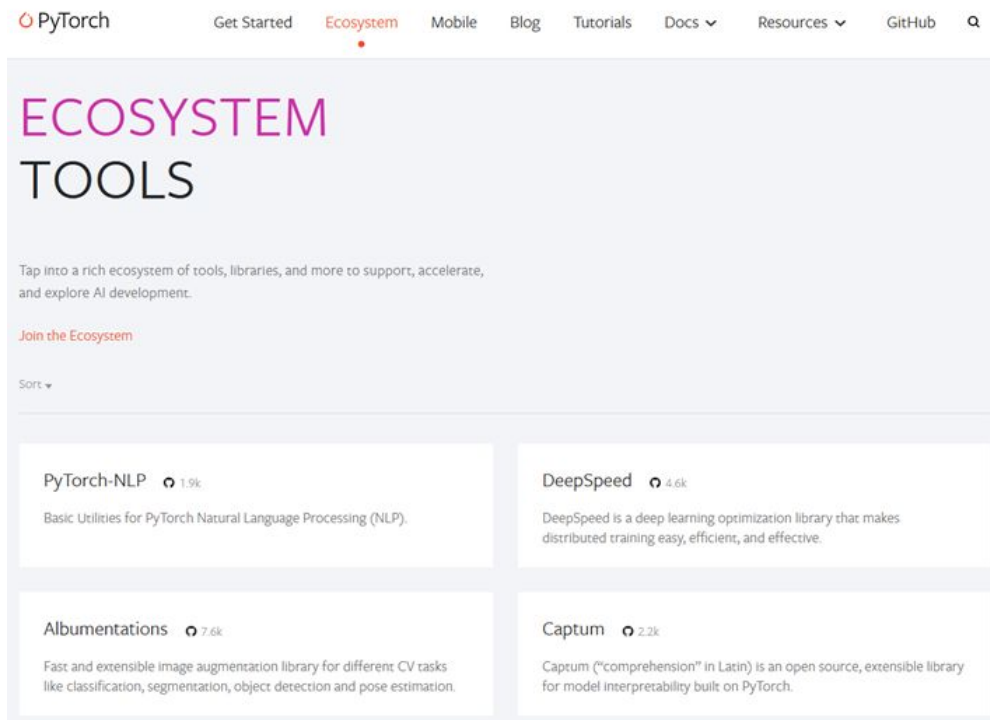


Trying to fast track this part

# The Pytorch Ecosystem

Collection of frameworks build to be used in combination with Pytorch

Note: Not a complete list of all frameworks



The screenshot shows the 'PyTorch Ecosystem Tools' page. The header includes the PyTorch logo and navigation links: 'Get Started', 'Ecosystem' (highlighted), 'Mobile', 'Blog', 'Tutorials', 'Docs', 'Resources', and 'GitHub'. The main heading is 'ECOSYSTEM TOOLS'. Below it, a subheading reads: 'Tap into a rich ecosystem of tools, libraries, and more to support, accelerate, and explore AI development.' A link 'Join the Ecosystem' is present. A 'Sort' dropdown menu is visible. The page displays four tool cards:

- 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

My own biased division

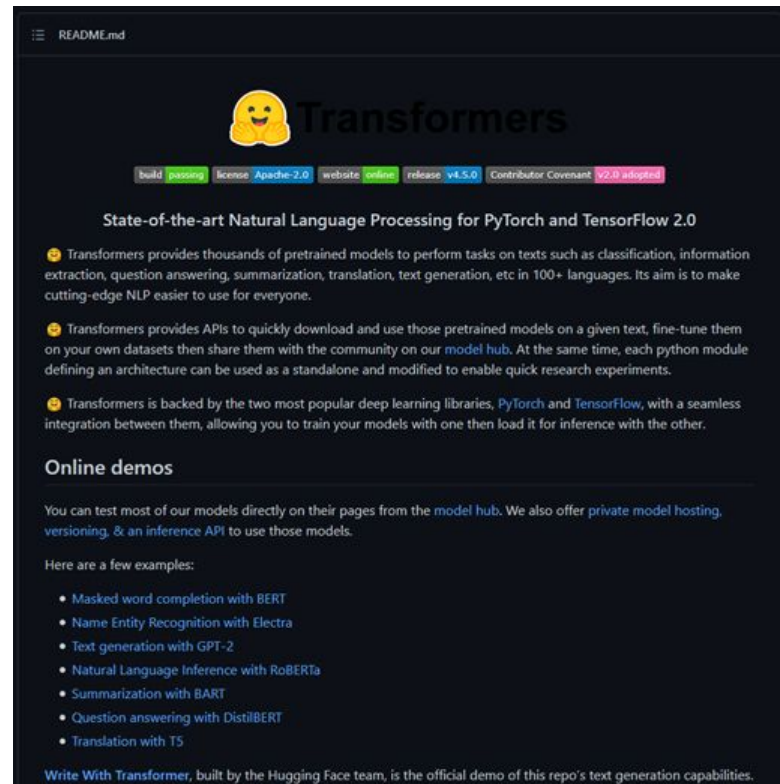
Data specific frameworks	Training frameworks	Utility frameworks
Transformers	fastai	Albumentations
Detectron2	Ray	PySyft
Pytorch geometric	Pytorch Lightning	Pyro
Flair	Horovod	Optuna
AllenNLP	DeepSpeed	Hydra
ParlAI	ONNX Runtime	Pytorch Metric Learning
DGL	skorch	Einops
PyTorch3D	Ignite	
MMF	Polyaxon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

# Project case 1: Natural Language Processing

Framework: Transformers

<https://github.com/huggingface/transformers>

Provides state-of-the-art NLP models for both Pytorch, Jax and Tensorflow.

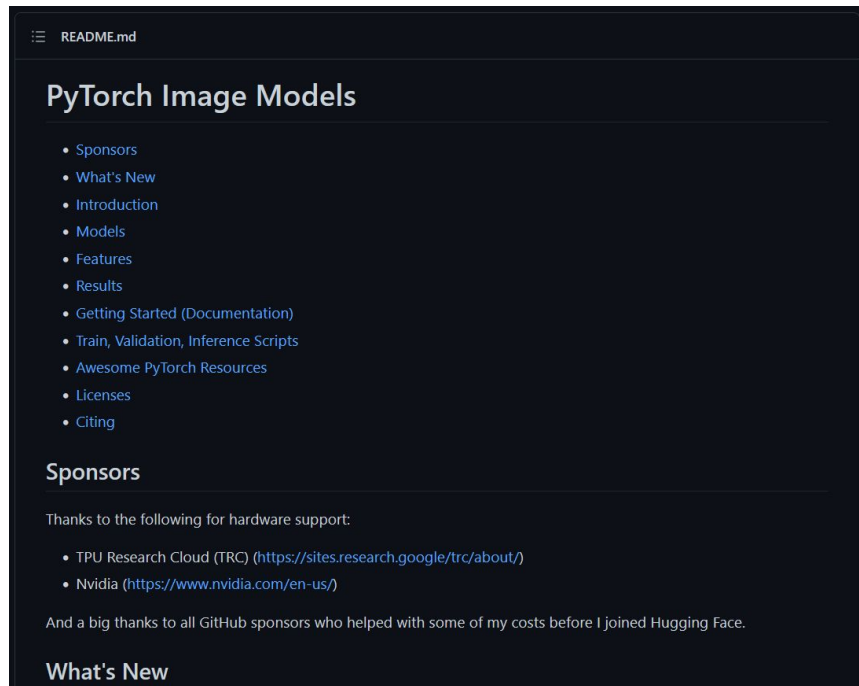


# Project case 2: Computer vision

Framework: Pytorch Image Models

<https://github.com/rwightman/pytorch-image-models>

Also known as TIMM. Image models, scripts, pretrained weights.





# Project case 3: Graphs and points

Framework: Pytorch Geometric

[https://github.com/pyg-team/pytorch\\_geometric](https://github.com/pyg-team/pytorch_geometric)

Graph Neural Network Library for PyTorch to work on irregular data such as graphs and points.

PyTorch geometric

PyTorch package 1.6.3 build passing docs passing codecov 84% contributions welcome slack pyg

[Documentation](#) | [Paper](#) | [Colab Notebooks](#) | [External Resources](#) | [OGB Examples](#)

PyTorch Geometric (PyG) is a geometric deep learning extension library for PyTorch.

It consists of various methods for deep learning on graphs and other irregular structures, also known as *geometric deep learning*, from a variety of published papers. In addition, it consists of an easy-to-use mini-batch loader for many small and single giant graphs, multi-gpu-support, a large number of common benchmark datasets (based on simple interfaces to create your own), and helpful transforms, both for learning on arbitrary graphs as well as on 3D meshes or point clouds. [Click here to join our Slack community!](#)

**OGB-LSC** @ KDD Cup 2021  
— Large-Scale Challenge —

OGB is hosting a **large-scale graph machine learning challenge (OGB-LSC)** at KDD Cup 2021 from March 15th to June 8th in order to discover innovative solutions for large-scale node classification, link prediction and graph regression. We are looking forward to your participation!

PyTorch Geometric makes implementing Graph Neural Networks a breeze (see [here](#) for the accompanying tutorial). For example, this is all it takes to implement the edge convolutional layer:

```
import torch
from torch.nn import Sequential as Seq, Linear as Lin, ReLU
from torch_geometric.nn import MessagePassing
```

# How to get an good idea?

The screenshot shows the GitHub repository for **kornia**, an Open Source Differentiable Computer Vision Library for PyTorch. The repository is maintained by **edgarriba** and has 11 branches and 16 tags. The file list on the left includes directories like `.circleci`, `.github`, `docker`, `docs`, `examples`, `kornia`, `packaging`, `test`, and `tutorials`, as well as files like `.codecov.yml`, `.gitconfig`, `.gitignore`, `CHANGELOG.md`, `CITATION.md`, `CODE_OF_CONDUCT.md`, and `CONTRIBUTING.rst`.

The right sidebar provides additional information:

- About:** Open Source Differentiable Computer Vision Library for PyTorch. Includes tags for `machine-learning`, `computer-vision`, `image-processing`, and `pytorch`.
- Releases:** 16 releases. The latest release is **Morphological operators, Dee...** from 21 days ago.
- Packages:** No packages published.
- Used by:** 290 projects. A blue box highlights this section, showing icons for various projects and a count of +282.

# How to get an good idea?

**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**

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

```

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")

# check up column names
data.columns = data.columns.str.strip().tolist()

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

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

# print it out from
train.head()
    
```

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.

[Got it](#) [Learn more](#)

# Summary

Pick a framework (try running their notebooks/examples!):

- Project 1: Natural Language Processing
- Project 2: Computer vision
- Project 3: Graphs and points

Brainstorm a project.

- It does not have to be particular big as you only have **4½** full days for working on it

Write a small (max 1 page) project description including:

- What model do intent to implement
- What data are you going to use
- How you think the chosen framework can be incorporated

# Checklist

## Week 1

- ☐ 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

**You do not need to do everything to pass, the list is meant to be exhaustive.**

# Hands-in: Today no later than 17:00

<input type="checkbox"/>	Assignment	<input type="checkbox"/>
<input type="checkbox"/>	No Category	
<input type="checkbox"/>	Exercises	<p>🚩  <a href="#">mygithub link.txt</a> (37 Bytes)</p> <p>🚩  <a href="#">mlops_gcloud_exercise.zip</a> (157.13 MB)</p> <p>Big zip file with everything + github link</p>
<input type="checkbox"/>	Project description	<p><b>Text Submission 1</b></p> <p><span>Unevaluated</span> Friday, 7 January 2022 2:13 PM</p> <p><a href="https://github.com/[redacted]/Project-MLOps-[redacted]">https://github.com/[redacted]/Project-MLOps-[redacted]</a></p> <p>Github link</p>

# Exam format

Thursday 20/1 – evaluation by either Nicki or Søren

## Group presentation

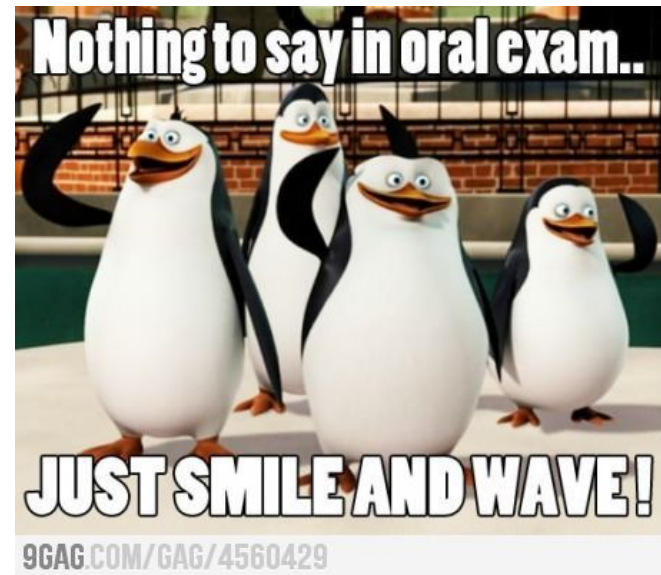
- 6 minutes of powerpoint showcase
- 10 minutes of discussion

## 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



## A bit of advice

1. Document everything - take screenshots of your work
2. Parallelize the work - many points on the checklist are independent
3. Commit frequently - no merge conflicts
4. Use each others strengths - some are good at coding, other at modelling



# Meme of the day

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

