The Machine Learning Angle for Open Source Science

Linux Foundation Member Summit 2023

J. Setpal

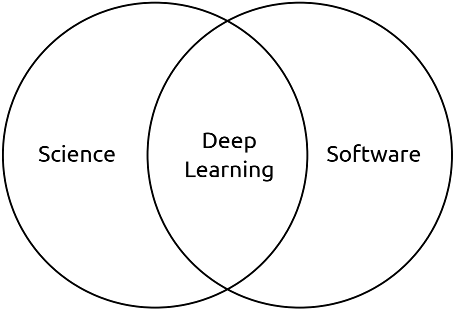
October 25, 2023

PURDUE UNIVERSITY

ML – Software or Science?

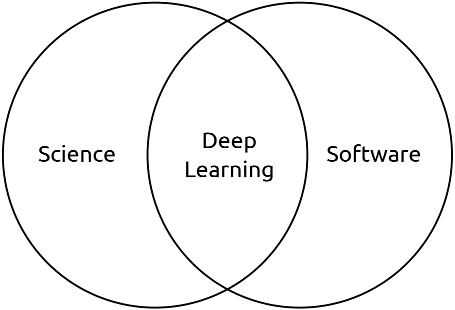
ML – Software or Science?

My argument: **Both**.



ML – Software or Science?

My argument: **Both**.

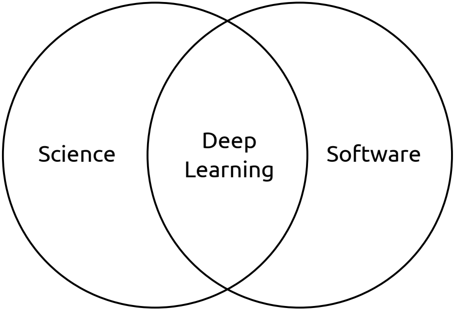


Now, some questions arise:

a. Why both?

ML – Software or Science?

My argument: **Both**.



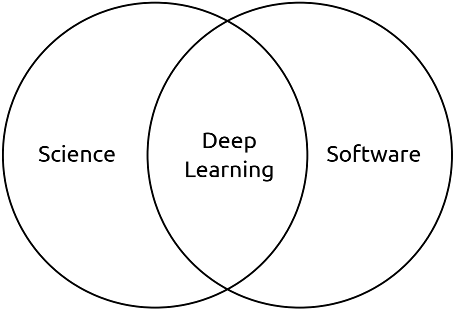
Now, some questions arise:

a. Why both?

b. How does it help?

ML – Software or Science?

My argument: **Both**.



Now, some questions arise:

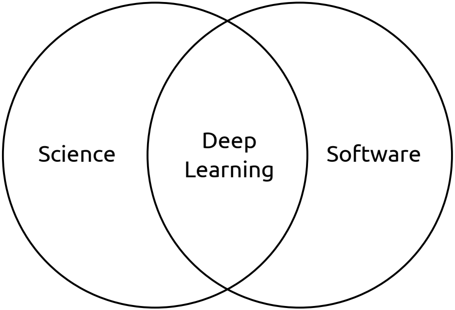
a. Why both?

b. How does it help?

c. Why didn't I include 'both' as an option during the poll?

ML – Software or Science?

My argument: **Both**.



Now, some questions arise:

a. Why both?

b. How does it help?

c. Why didn't I include 'both' as an option during the poll?

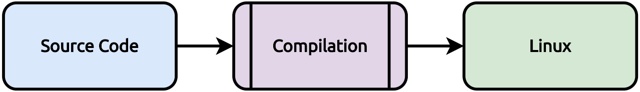
Let's talk about it.

Let's Recontextualize ML Development

Idea: training $\approx$ compilation

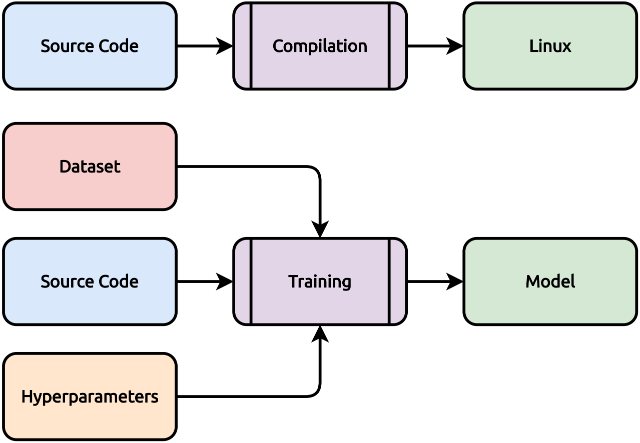
Let's Recontextualize ML Development

Idea: training $\approx$ compilation



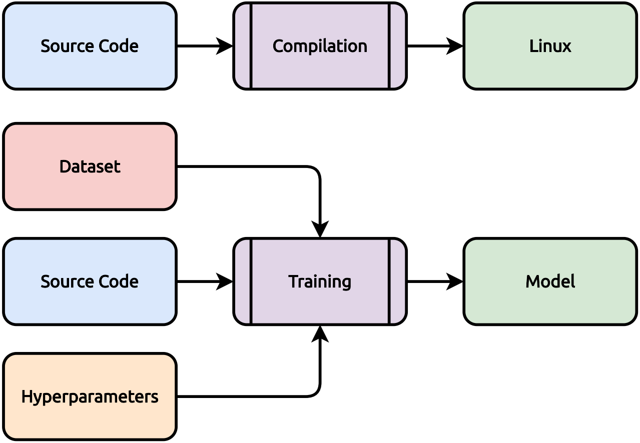
Let’s Recontextualize ML Development

Idea: training $\approx$ compilation



Let's Recontextualize ML Development

Idea: training $\approx$ compilation



Key Difference: time(training) $>>$ time(compilation)

Reproducibility for Open Source Science

Machine Learning is a **science**.

Reproducibility for Open Source Science

Machine Learning is a **science**. Sometimes, the results of the experiments are production-ready. Then, it's also **software**.

Reproducibility for Open Source Science

Machine Learning is a **science**. Sometimes, the results of the experiments are production-ready. Then, it's also **software**.

Consequence: Traditional 'open source' *is not enough*.

Reproducibility for Open Source Science

Machine Learning is a **science**. Sometimes, the results of the experiments are production-ready. Then, it's also **software**.

Consequence: Traditional 'open source' *is not enough*.
Idea: Free and Open Source Science = Open Source + Reproducibility.

Reproducibility for Open Source Science

Machine Learning is a **science**. Sometimes, the results of the experiments are production-ready. Then, it's also **software**.

Consequence: Traditional 'open source' *is not enough*.
Idea: Free and Open Source Science = Open Source + Reproducibility.
Bonus: We can reuse the 'FOSS' acronym!

Reproducibility for Open Source Science

Machine Learning is a **science**. Sometimes, the results of the experiments are production-ready. Then, it's also **software**.

Consequence: Traditional 'open source' *is not enough*.
Idea: Free and Open Source Science = Open Source + Reproducibility.
Bonus: We can reuse the 'FOSS' acronym!

Important Note

This still is a partial answer. The democratization of accelerated hardware is still a **significant challenge** we fail to address.

How can we achieve this?

Step 0: Accept[1] that **not everything can be open**.

[1] begrudgingly

How can we achieve this?

Step 0: Accept[1] that **not everything can be open**. The maximal approach won't work.

[1] begrudgingly

How can we achieve this?

Step 0: Accept[1] that **not everything can be open**. The maximal approach won't work.

This is primarily owing to data privacy, and extends to model parameters.[2]

[1] begrudgingly

[2] `https://unlearning-challenge.github.io/`

How can we achieve this?

Step 0: Accept[1] that **not everything can be open**. The maximal approach won't work.

This is primarily owing to data privacy, and extends to model parameters.[2]

However, we *should* expect:

a. Documentation.

[1] begrudgingly

[2] `https://unlearning-challenge.github.io/`

How can we achieve this?

Step 0: Accept[1] that **not everything can be open**. The maximal approach won't work.

This is primarily owing to data privacy, and extends to model parameters.[2]

However, we *should* expect:

a. Documentation.
b. Synthetic Dataset Samples.

[1] begrudgingly
[2] `https://unlearning-challenge.github.io/`

How can we achieve this?

Step 0: Accept[1] that **not everything can be open**. The maximal approach won't work.

This is primarily owing to data privacy, and extends to model parameters.[2]

However, we *should* expect:

a. Documentation.
b. Synthetic Dataset Samples.
c. Training and Inference Code.

[1] begrudgingly
[2] `https://unlearning-challenge.github.io/`

How can we achieve this?

Step 0: Accept[1] that **not everything can be open**. The maximal approach won't work.

This is primarily owing to data privacy, and extends to model parameters.[2]

However, we *should* expect:

a. Documentation.
b. Synthetic Dataset Samples.
c. Training and Inference Code.
d. Descriptive whitepaper.

[1] begrudgingly
[2] `https://unlearning-challenge.github.io/`

How can we achieve this?

Step 0: Accept[1] that **not everything can be open**. The maximal approach won't work.

This is primarily owing to data privacy, and extends to model parameters.[2]

However, we *should* expect:

a. Documentation.
b. Synthetic Dataset Samples.
c. Training and Inference Code.
d. Descriptive whitepaper.
e. **Permissive Licensing**.

[1] begrudgingly
[2] `https://unlearning-challenge.github.io/`

How can we achieve this?

Step 0: Accept[1] that **not everything can be open**. The maximal approach won't work.

This is primarily owing to data privacy, and extends to model parameters.[2]

However, we *should* expect:

a. Documentation.
b. Synthetic Dataset Samples.
c. Training and Inference Code.
d. Descriptive whitepaper.
e. **Permissive Licensing**.

So; where do we go from here?

[1] begrudgingly
[2] `https://unlearning-challenge.github.io/`

Death to Jupyter Notebooks

Jupyter Notebooks are *fantastic* for experimentation, but unusable in a production context.

Death to Jupyter Notebooks

Jupyter Notebooks are *fantastic* for experimentation, but unusable in a production context.

What **not** to do: `https://github.com/jinensetpal/archimede.git`

Death to Jupyter Notebooks

Jupyter Notebooks are *fantastic* for experimentation, but unusable in a production context.

What **not** to do: `https://github.com/jinensetpal/archimede.git`
Because:

a. There's no real entrypoint.

Death to Jupyter Notebooks

Jupyter Notebooks are *fantastic* for experimentation, but unusable in a production context.

What **not** to do: `https://github.com/jinensetpal/archimede.git`
Because:

a. There's no real entrypoint.
b. Random pickled objects.

Death to Jupyter Notebooks

Jupyter Notebooks are *fantastic* for experimentation, but unusable in a production context.

What **not** to do: `https://github.com/jinensetpal/archimede.git`
Because:

a. There's no real entrypoint.
b. Random pickled objects.
c. No version control.

Death to Jupyter Notebooks

Jupyter Notebooks are *fantastic* for experimentation, but unusable in a production context.

What **not** to do: `https://github.com/jinensetpal/archimede.git`
Because:

a. There's no real entrypoint.
b. Random pickled objects.
c. No version control.

Despite being computationally inexpensive, and having open source {code, data, hyperparameters}, it's not *actually* helpful.

Module-Based Development

Idea: Every project is a python package.

Module-Based Development

Idea: Every project is a python package. Modules establish *sections* (ex. data processing, model architecture) within the repository.

Module-Based Development

Idea: Every project is a python package. Modules establish *sections* (ex. data processing, model architecture) within the repository.

Include a `const.py` that contains **hyperparameters**.

Module-Based Development

Idea: Every project is a python package. Modules establish *sections* (ex. data processing, model architecture) within the repository.

Include a `const.py` that contains **hyperparameters**.

Then, from the root of the repository you'd run:

```
$ python -m path.to.file
```

Module-Based Development

Idea: Every project is a python package. Modules establish *sections* (ex. data processing, model architecture) within the repository.

Include a `const.py` that contains **hyperparameters**.

Then, from the root of the repository you'd run:

```
$ python -m path.to.file
```

Consequences:

a. Structures the codebase; components are easy to identify and debug.

Module-Based Development

Idea: Every project is a python package. Modules establish *sections* (ex. data processing, model architecture) within the repository.

Include a `const.py` that contains **hyperparameters**.

Then, from the root of the repository you'd run:

```
$ python -m path.to.file
```

Consequences:

a. Structures the codebase; components are easy to identify and debug.
b. `$PYTHONPATH` is resolved automatically, relative imports work!

Module-Based Development

Idea: Every project is a python package. Modules establish *sections* (ex. data processing, model architecture) within the repository.

Include a `const.py` that contains **hyperparameters**.

Then, from the root of the repository you'd run:

```
$ python -m path.to.file
```

Consequences:

a. Structures the codebase; components are easy to identify and debug.
b. `$PYTHONPATH` is resolved automatically, relative imports work!

Great Example: Ultralytics' YOLOv8.
Great Template: Cookie Cutter Data Science

Version Control it All

git is a brilliant tool that allows us to version control code; but what about data?

Version Control it All

git is a brilliant tool that allows us to version control code; but what about data?

Enter **DVC** (Data Version Control).

Version Control it All

git is a brilliant tool that allows us to version control code; but what about data?

Enter **DVC** (Data Version Control). It enables us to add, track, push, pull and checkout data.

Version Control it All

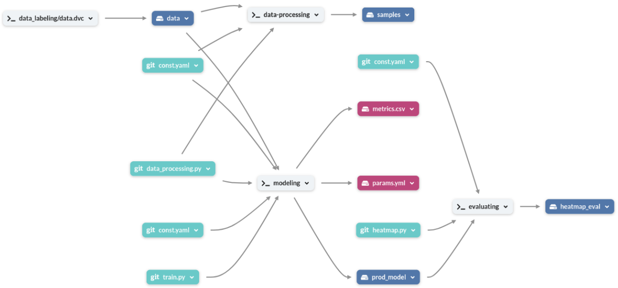
git is a brilliant tool that allows us to version control code; but what about data?

Enter **DVC** (Data Version Control). It enables us to add, track, push, pull and checkout data.

Consequence: Data is now tracked. It's associated with a specific commit, and can be diffed.

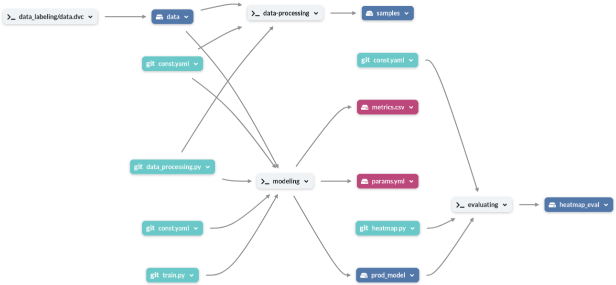
Directed Acyclic Graph for Execution

DVC also allows us to structure code execution as a **DAG** (Directed Acyclic Graph), elucidating code / data / hyperparameter relationships:



Directed Acyclic Graph for Execution

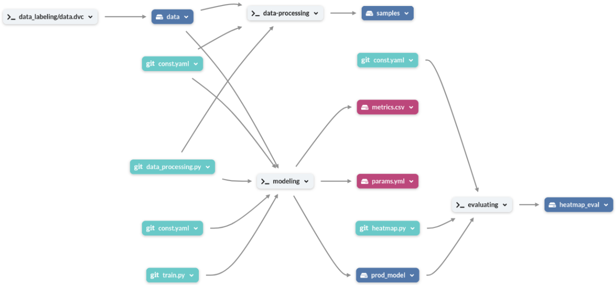
DVC also allows us to structure code execution as a **DAG** (Directed Acyclic Graph), elucidating code / data / hyperparameter relationships:



Crucially, it gives us an **entrypoint** and a project overview.

Directed Acyclic Graph for Execution

DVC also allows us to structure code execution as a **DAG** (Directed Acyclic Graph), elucidating code / data / hyperparameter relationships:



Crucially, it gives us an **entrypoint** and a project overview.

Projects can be reproduced using: `$ dvc repro`

Systematic Experiment & Model Tracking

Next, we target the unpredictability of training.

Systematic Experiment & Model Tracking

Next, we target the unpredictability of training.

We are not guaranteed a minima. Therefore, we track **metrics** and **hyperparameters**, to find the best set for a given run.

Systematic Experiment & Model Tracking

Next, we target the unpredictability of training.

We are not guaranteed a minima. Therefore, we track **metrics** and **hyperparameters**, to find the best set for a given run.

MLFlow helps track and compare various experiments and parameters.

Systematic Experiment & Model Tracking

Next, we target the unpredictability of training.

We are not guaranteed a minima. Therefore, we track **metrics** and **hyperparameters**, to find the best set for a given run.

MLFlow helps track and compare various experiments and parameters.

In addition, it allows tagging runs, registering models, and deploying a target model-as-a-service using Docker.

Systematic Experiment & Model Tracking

Next, we target the unpredictability of training.

We are not guaranteed a minima. Therefore, we track **metrics** and **hyperparameters**, to find the best set for a given run.

MLFlow helps track and compare various experiments and parameters.

In addition, it allows tagging runs, registering models, and deploying a target model-as-a-service using Docker.

This tool manages the experiment-model **lifecycle**.

A Single Source of Truth

All these resources are still disjoint; they need a way to come together.

A Single Source of Truth

All these resources are still disjoint; they need a way to come together.

DagsHub solves this. It's a remote host for:

a. DVC.

A Single Source of Truth

All these resources are still disjoint; they need a way to come together.

DagsHub solves this. It's a remote host for:

a. DVC.

b. MLFlow.

A Single Source of Truth

All these resources are still disjoint; they need a way to come together.

DagsHub solves this. It's a remote host for:

a. DVC.
b. MLFlow.
c. Label Studio.

A Single Source of Truth

All these resources are still disjoint; they need a way to come together.

DagsHub solves this. It's a remote host for:

a. DVC.
b. MLFlow.
c. Label Studio.

It also connects with **GitHub** and **Colaboratory**, and allows for big-picture view of the project.

The Reproducibility Checklist

Finally, we need to ensure that research can be replicated **by third parties**.

The Reproducibility Checklist

Finally, we need to ensure that research can be replicated **by third parties**.

For this, we use Dr. Pineau's **Reproducibility Checklist**.

The Reproducibility Checklist

Finally, we need to ensure that research can be replicated **by third parties**.

For this, we use Dr. Pineau's **Reproducibility Checklist**. Critical ideas:

a. **Models and algorithms** require clear descriptions of assumptions, settings and time-complexity analyses.

The Reproducibility Checklist

Finally, we need to ensure that research can be replicated **by third parties**.

For this, we use Dr. Pineau's **Reproducibility Checklist**. Critical ideas:

a. **Models and algorithms** require clear descriptions of assumptions, settings and time-complexity analyses.
b. **Datasets** must include statistics, splits, and pre-processing procedure.

The Reproducibility Checklist

Finally, we need to ensure that research can be replicated **by third parties**.

For this, we use Dr. Pineau's **Reproducibility Checklist**. Critical ideas:

a. **Models and algorithms** require clear descriptions of assumptions, settings and time-complexity analyses.
b. **Datasets** must include statistics, splits, and pre-processing procedure.
c. **Code** must specify requirements and code for training, inference as well as any pre-trained models.

The Reproducibility Checklist

Finally, we need to ensure that research can be replicated **by third parties**.

For this, we use Dr. Pineau's **Reproducibility Checklist**. Critical ideas:

a. **Models and algorithms** require clear descriptions of assumptions, settings and time-complexity analyses.
b. **Datasets** must include statistics, splits, and pre-processing procedure.
c. **Code** must specify requirements and code for training, inference as well as any pre-trained models.
d. **Experiments** must include the range of hyperparameters, the best configuration, the evaluation statistic and training hardware.

The Reproducibility Checklist

Finally, we need to ensure that research can be replicated **by third parties**.

For this, we use Dr. Pineau's **Reproducibility Checklist**. Critical ideas:

a. **Models and algorithms** require clear descriptions of assumptions, settings and time-complexity analyses.

b. **Datasets** must include statistics, splits, and pre-processing procedure.

c. **Code** must specify requirements and code for training, inference as well as any pre-trained models.

d. **Experiments** must include the range of hyperparameters, the best configuration, the evaluation statistic and training hardware.

As a consequence, we can realistically evaluate the claims made by the paper's authors.

Extensibility + the Overarching Principle

This is a sample framework intended to establish a baseline approach.

Extensibility + the Overarching Principle

This is a sample framework intended to establish a <u>baseline</u> approach.

The goal is to extend this on a case-by-case basis; these concepts apply generally.

Extensibility + the Overarching Principle

This is a sample framework intended to establish a baseline approach.

The goal is to extend this on a case-by-case basis; these concepts apply generally.

To adapt the **approach** to your use-case:

a. Find differences from the established standard.

Extensibility + the Overarching Principle

This is a sample framework intended to establish a baseline approach.

The goal is to extend this on a case-by-case basis; these concepts apply generally.

To adapt the **approach** to your use-case:

a. Find differences from the established standard.
b. Identify the parameters required to recreate the experimental setup.

Extensibility + the Overarching Principle

This is a sample framework intended to establish a baseline approach.

The goal is to extend this on a case-by-case basis; these concepts apply generally.

To adapt the **approach** to your use-case:

a. Find differences from the established standard.
b. Identify the parameters required to recreate the experimental setup.
c. Set hard / soft requirements based on criticality to replication & user privacy.

QA / Discussion

Questions for me?

Thank you!

Have an awesome rest of your day!

Slides: https://www.cs.purdue.edu/homes/jsetpal/slides/lfms.pdf

Homepage: https://jinensetpal.github.io/
Email: jsetpal@cs.purdue.edu
{**Git, Dags**}**Hub:** @jinensetpal
Twitter: @48bitmachine