DeepMind

XManager (External Talk)

Updated: 2 August 2022

Presented by Andrew Chen (xmanager-oss@google.com)

DeepMind

Agenda

I. The Problem

II. Our Solution

III. Demo

IV. Design Principles

V. Limitations

The Problem:

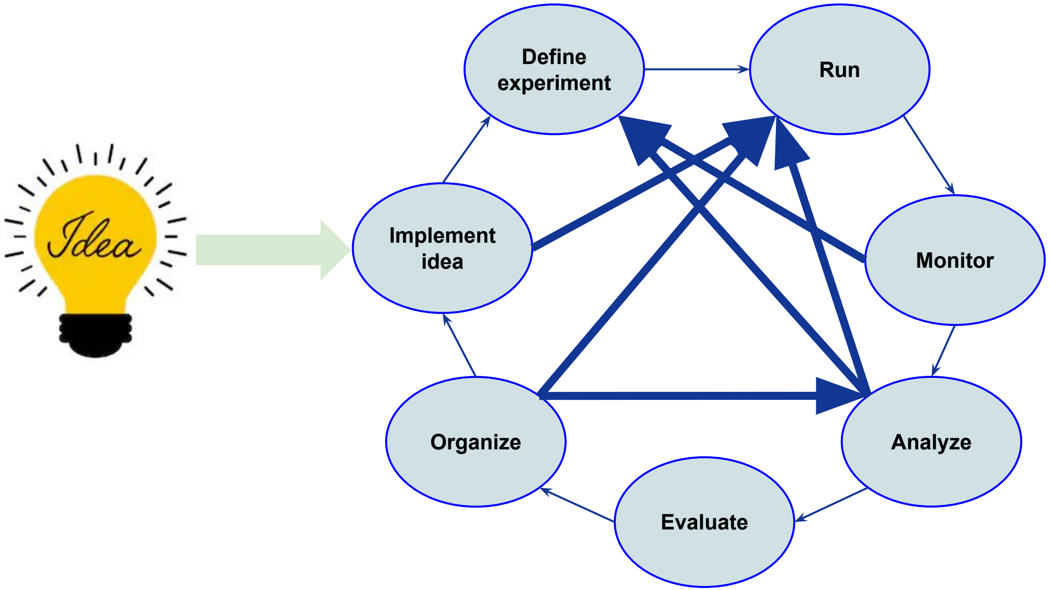
Current ML tools are not designed for Research

Research must be:

Fast, Reproducible, and Collaborative

Challenge 1: Fast Research

Early-stage research is focused on unexplored areas and needs to be highly-flexible.



Challenge 1: Fast Research

How fast can you clone a Github baseline and run it?

- Current tools are too slow/costly to spin up.
- Production services geared towards ML lifecycle management are not designed for research.
- Full-service frameworks causes tight coupling between the ML code and the framework.

Challenge 2: Reproducible Research

How easily can you verify the empirical claims of a paper?

- Running demo code requires setup.
- Experimental setups are not described in the code.

How can you track changes?

- ML code is changing.
- Software dependencies are changing.
- Hyperparameters are changing.

Challenge 3: Collaborative Research

Members of your research team may work at different companies or different universities.

- Physical desktop with GPUs
- Google Borg
- Google Cloud Platform (GCP) Vertex AI
- Kubernetes (K8s)
- On-prem high performance computing (HPC)

And more...

Our Solution:

A simple framework for defining and managing experiments

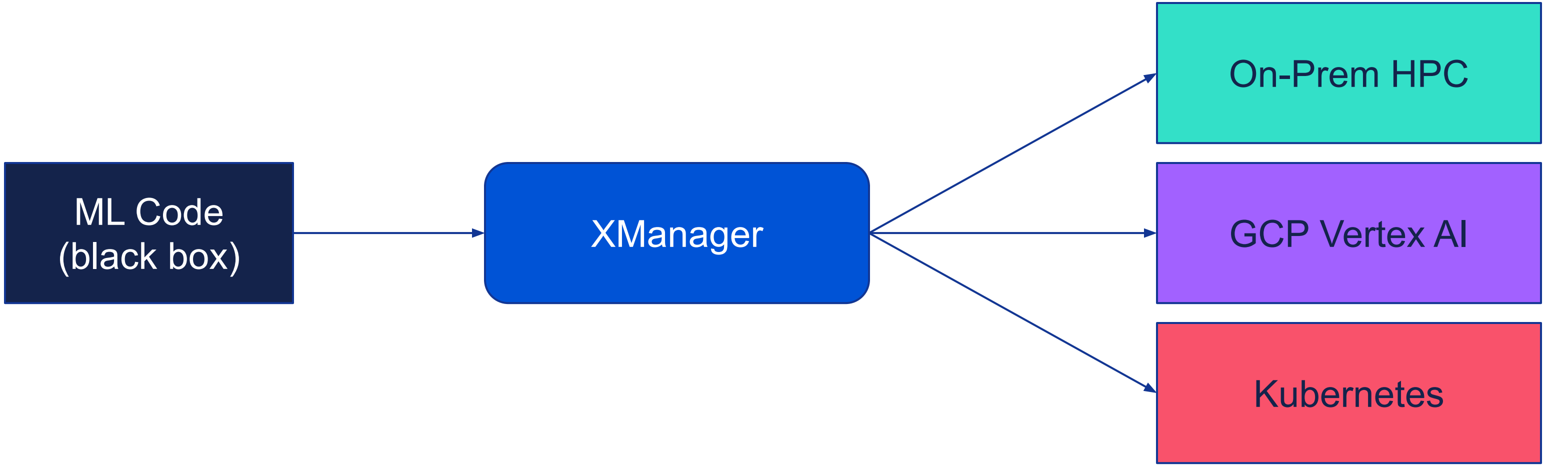
A universal specification for experimentation

XManager is a light-weight, non-invasive, unopinionated, platform-agnostic Python framework for defining ML Experiments.

Take any ML code "as-is" and use XManager to "plug-and-play".

A universal specification for experimentation

XManager abstracts the differences between ML platforms and leaves more time for researchers to do research.



A collaboration tool for industry + academia

The same ML code + the same XManager code:

- Can run on a physical desktops with GPUs
- Can run on proprietary clusters, e.g. Borg
- Can run on open-source clusters (Kubernetes)
- Runs on Cloud-based AI solutions

Modular components that can mix-and-match

The XManager interface allows clients to swap public components with private components without changing the ML code or structure.

```
from xmanager import xm
from xmanager.xm_local  import create_experiment
from xmanager.xm_local  import Vertex

with create_experiment("train") as experiment:
  # assert isinstance(experiment, xm.Experiment)

  job = xm.Job(train_executable, Vertex())
  experiment.add(job)
```

Modular components that can mix-and-match

The XManager interface allows clients to swap public components with private components without changing the ML code or structure.

```python
from xmanager import xm
from xmanager.xm_google import create_experiment
from xmanager.xm_google import Borg

with create_experiment("train") as experiment:
  # assert isinstance(experiment, xm.Experiment)

  job = xm.Job(train_executable, Borg())
  experiment.add(job)
```

A starting point for new research

XManager makes it easy to snapshot/share/run papers with code.

- Sharable ML code.
- Sharable XManager configuration code.
- Sharable Docker images.

Groups using XManager

And more...

Demo Time!

Running Tensorflow on XManager

https://github.com/deepmind/xmanager/tree/main/examples/cifar10_tensorflow

Running Tensorflow on XManager

```
# Import base XManager experiment components.
from xmanager import xm

# Import the execution environments compatible with the open-source XManager codebase.
from xmanager import xm_local
```

Running Tensorflow on XManager

```
def main(_):

  # Declare the experiment you want to create.
  # Open the experiment in a context manager.
  with xm_local.create_experiment(experiment_title='cifar10') as experiment:
```

Running Tensorflow on XManager

```
# Declare the package you want to run.
spec = xm.PythonContainer(
    # Package the current directory that this script is in.
    path='.',
    base_image='gcr.io/deeplearning-platform-release/tf2-gpu.2-6',
    entrypoint=xm.ModuleName('cifar10'),
)
```

Running Tensorflow on XManager

```
# Declare the environment you want to run your package in.
executor = xm_local.Vertex()
```

Running Tensorflow on XManager

```
# Prepare your package to be staged in the execution environment.
[executable] = experiment.package([
    xm.Packageable(
        executable_spec=spec,
        executor_spec=executor.Spec(),
    ),
])
```

Running Tensorflow on XManager

```python
# Declare the hyperparameter sweep or trials to run.
batch_sizes = [64, 1024]
learning_rates = [0.1, 0.001]
trials = list(
  {'batch_size': batch_size, 'learning_rate': learning_rate}
  for batch_size, learning_rate in itertools.product(batch_sizes, learning_rates)
)
```

Running Tensorflow on XManager

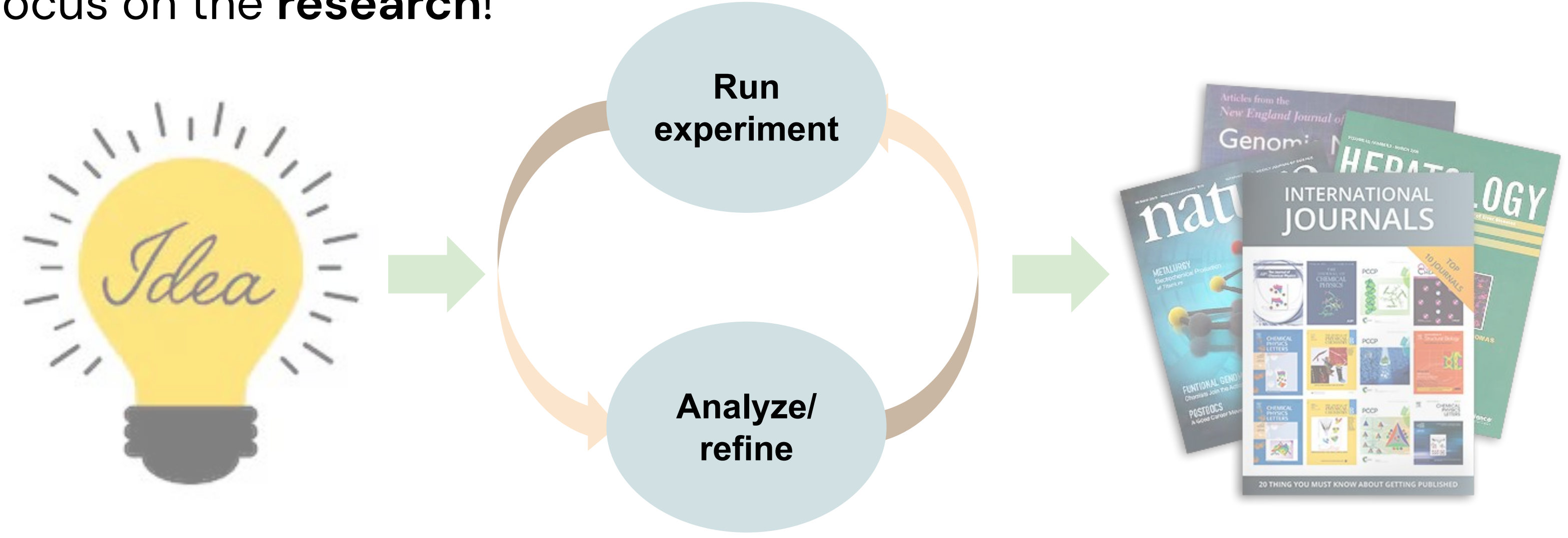
```
# For each hyperparameter set, create a job.
for hyperparameters in trials:
  experiment.add(
      xm.Job(
          executable=executable,
          executor=executor,
          args=hyperparameters,
      ))
```

XManager Design Principles

Tailored for Research

ML research is about *science*.
Real-world applications are about *software engineering*.

Focus on the **research**!



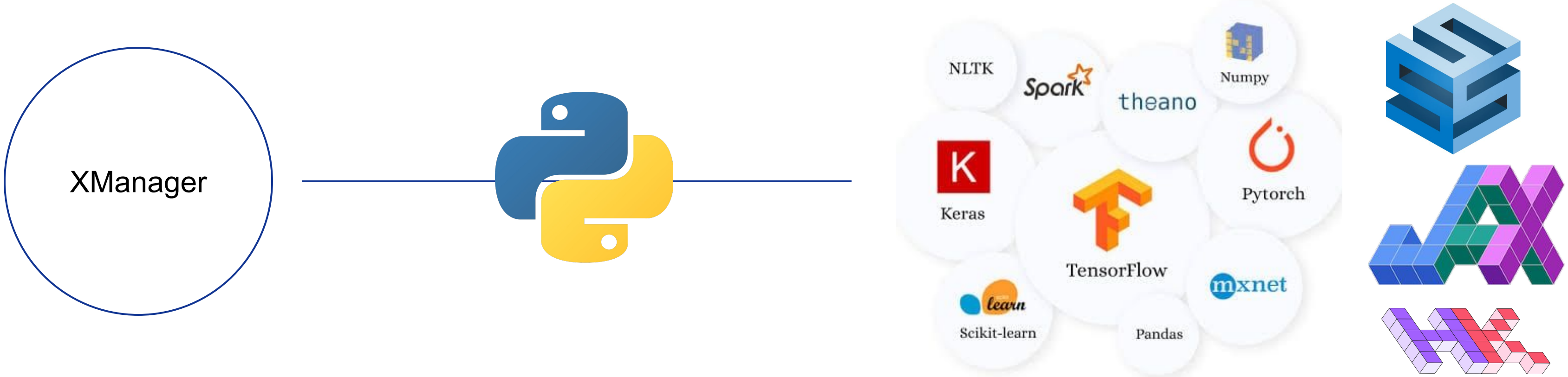
Tailored for Research

What researchers care about:

- ML Code
- ~~Data extraction/analysis~~
- ~~Model validation~~
- ~~Serving infrastructure~~
- ~~CI/CD pipelines~~

Python is the language of ML

XManager is built for ML researchers, and researchers write ML code (Tensorflow, PyTorch, JAX) in Python.



Modularity

XManager is intended for many different executable types:

- Python code
- C++ code
- Docker image
- Shell script
- Binary package

Running on many different platforms:

- Physical desktop
- GCP Vertex AI
- Kubernetes
- On-prem/remote HPCs

Extensibility

The XManager interface can be extended to further support new:

- Executable types (binaries, packages, configs)
- Executor types (AWS, Azure, etc.)
- Scheduler flows (xm_local, xm_google, etc.)

Limitations

Not Included

- No graphical user-interface.
  - Use GCP Vertex AI instead
- No metric storage.
  - Use Tensorboard instead
- No job status tracking.
  - Use Vertex Training instead
- No resource sharing or queueing.
  - Rely on K8s resource quotas or GCP compute quotas.
- No dataset versioning.
  - Re-running using the exact same dataset isn't part of XM.