Distributed Deep Learning Using Hopsworks

CGI Trainee Program Workshop

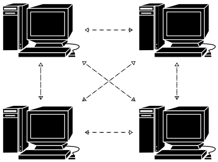
Kim Hammar
kim@logicalclocks.com

Before we start..

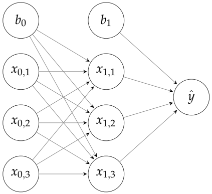
1. Register for an account at: `www.hops.site`
2. Follow the instructions at: `http://bit.ly/2EnZQgW`

DISTRIBUTED COMPUTING + DEEP LEARNING = ?

Distributed Computing



Deep Learning



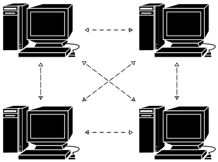
Why Combine the two?

2em1[1] Chen Sun et al. "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In: *CoRR* abs/1707.02968 (2017). arXiv: 1707.02968. URL: http://arxiv.org/abs/1707.02968.

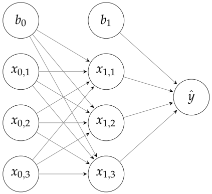
2em1[2] Jeffrey Dean et al. "Large Scale Distributed Deep Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1223–1231.

DISTRIBUTED COMPUTING + DEEP LEARNING = ?

Distributed Computing



Deep Learning



Why Combine the two?

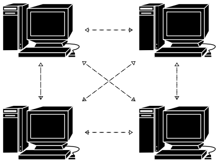
- We like challenging problems ☺

2em1[1] Chen Sun et al. "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In: *CoRR* abs/1707.02968 (2017). arXiv: 1707.02968. URL: http://arxiv.org/abs/1707.02968.

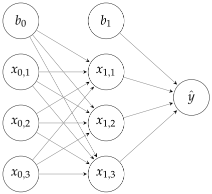
2em1[2] Jeffrey Dean et al. "Large Scale Distributed Deep Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1223–1231.

DISTRIBUTED COMPUTING + DEEP LEARNING = ?

Distributed Computing



Deep Learning



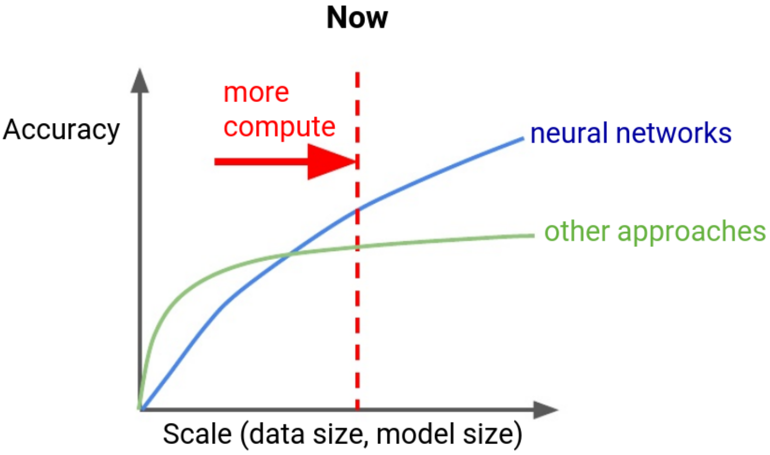
Why Combine the two?

- We like challenging problems ☺
- More productive data science
- Unreasonable effectiveness of data[1]
- To achieve state-of-the-art results[2]

2em1[1] Chen Sun et al. “Revisiting Unreasonable Effectiveness of Data in Deep Learning Era”. In: *CoRR* abs/1707.02968 (2017). arXiv: 1707.02968. URL: http://arxiv.org/abs/1707.02968.

2em1[2] Jeffrey Dean et al. “Large Scale Distributed Deep Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1223–1231.

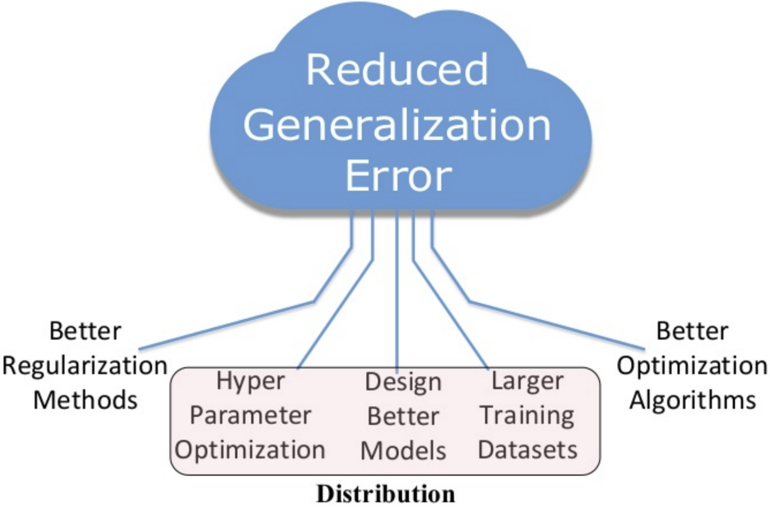
DISTRIBUTED DEEP LEARNING (DDL): PREDICTABLE SCALING



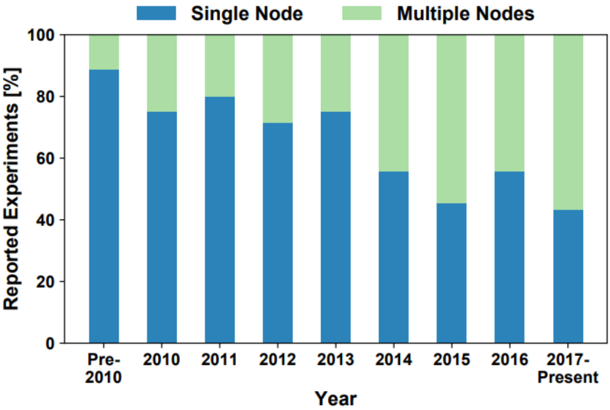
3

2em1[3] Jeff Dean. *Building Intelligent Systems withLarge Scale Deep Learning*. https://www.scribd.com/document/355752799/Jeff-Dean-s-Lecture-for-YC-AI. 2018.

DISTRIBUTED DEEP LEARNING (DDL): PREDICTABLE SCALING



DDL IS NOT A SECRET ANYMORE



(b) Training with Single vs. Multiple Nodes [4]

2em1[4] Tal Ben-Nun and Torsten Hoefler. "Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis". In: *CoRR* abs/1802.09941 (2018). arXiv: 1802.09941. URL: http://arxiv.org/abs/1802.09941.

DDL IS NOT A SECRET ANYMORE

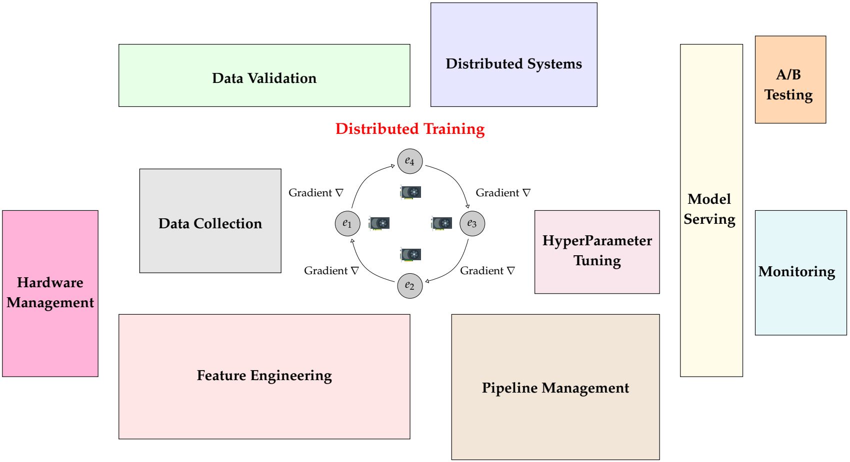
Frameworks for DDL

Kubeflow
Microsoft CNTK
HOROVOD
BigDL
Amazon SageMaker
TensorflowOnSpark
Distributed TF
CaffeOnSpark
PYTORCH
APACHE Spark
mxnet
DL4J
TonY
Chainer MN

Companies using DDL

facebook
Google
UBER
zalando
YAHOO!
NETFLIX
amazon
LinkedIn
Baidu 百度
IBM
Microsoft

DDL REQUIRES AN ENTIRE SOFTWARE/INFRASTRUCTURE STACK



OUTLINE

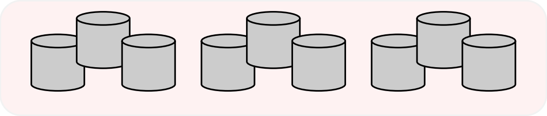
1. **Hopsworks**: Background of the platform
2. **Managed Distributed Deep Learning** using HopsYARN, HopsML, PySpark, and Tensorflow
3. **Black-Box Optimization (Hyperparameter Tuning)** using Hopsworks, Metadata Store, PySpark, and Maggy[5]
4. **Feature Store** data management for machine learning
5. **Coffee Break**
6. **Demo**, end-to-end ML pipeline
7. **Hands-on Workshop**, try out Hopsworks on our cluster in Luleå

2em1[5] Moritz Meister and Sina Sheikholeslami. *Maggy*. https://github.com/logicalclocks/maggy. 2019.

HOPSWORKS

HOPSWORKS

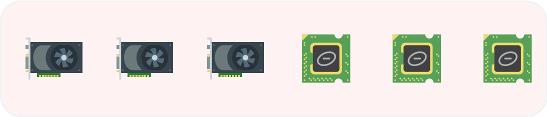
HopsFS



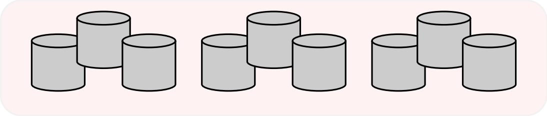
HOPSWORKS

HopsYARN

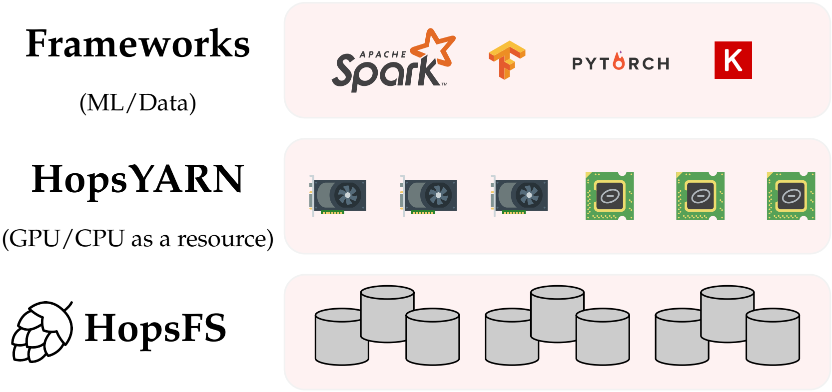
(GPU/CPU as a resource)



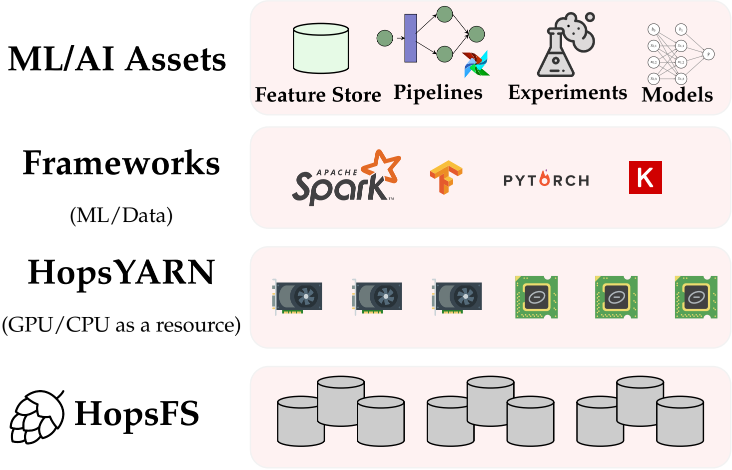
HopsFS



Hopsworks


Frameworks
(ML/Data)
APACHE Spark
PYTORCH
K
HopsYARN
(GPU/CPU as a resource)
HopsFS

HOPSWORKS

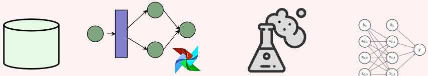

ML/AI Assets
Feature Store
Pipelines
Experiments
Models
Frameworks
(ML/Data)
APACHE Spark
PYTORCH
K
HopsYARN
(GPU/CPU as a resource)
HopsFS

HOPSWORKS

APIs

```python
from hops import featurestore
from hops import experiment
featurestore.get_features([
                         "average_attendance",
                         "average_player_age"])
experiment.collective_all_reduce(features, model)
```

ML/AI Assets



Feature Store Pipelines Experiments Models

Frameworks

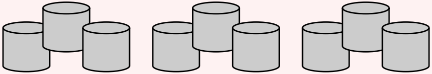
(ML/Data)

APACHE Spark PYTORCH K

HopsYARN

(GPU/CPU as a resource)

HopsFS



Hopsworks

APIs

```
from hops import featurestore
from hops import experiment
featurestore.get_features([
                    "average_attendance",
                    "average_player_age"])
experiment.collective_all_reduce(features, model)
```

ML/AI Assets

Frameworks

(ML/Data)

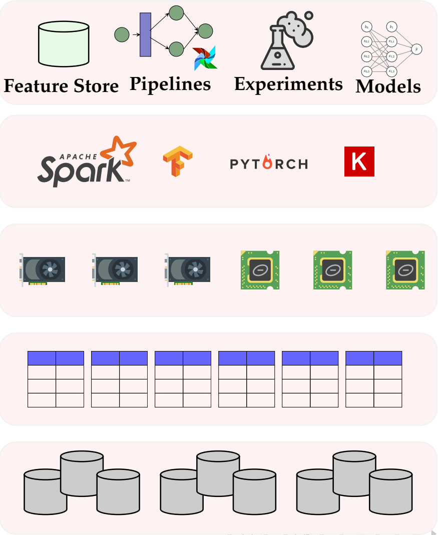
HopsYARN

(GPU/CPU as a resource)

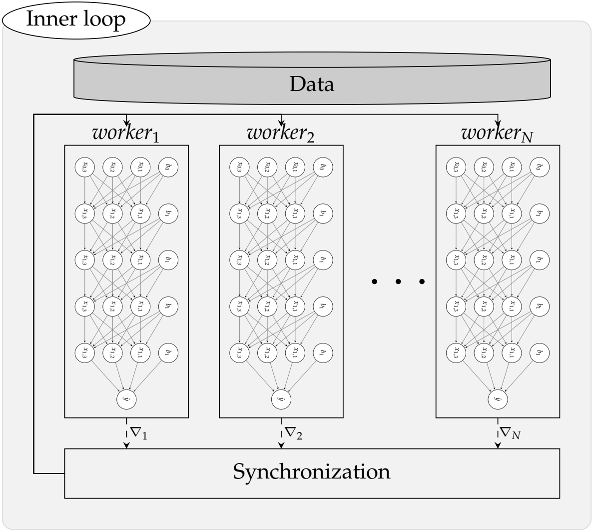
Distributed Metadata

(Available from REST API)

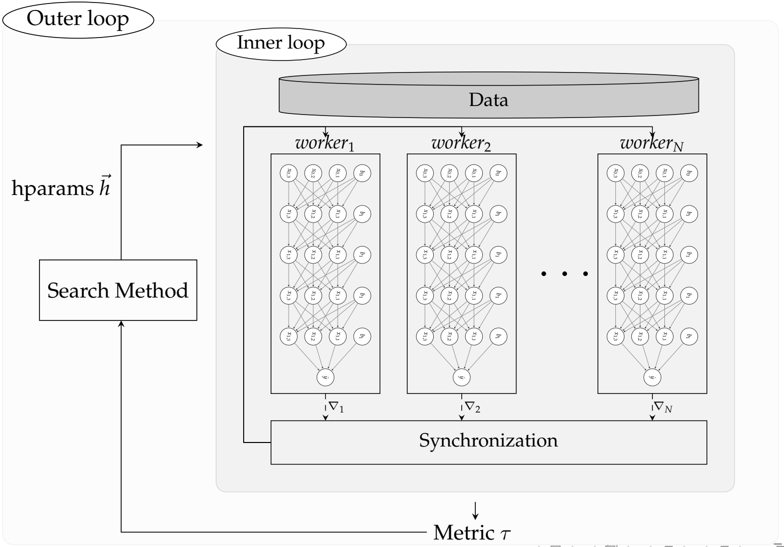
HopsFS



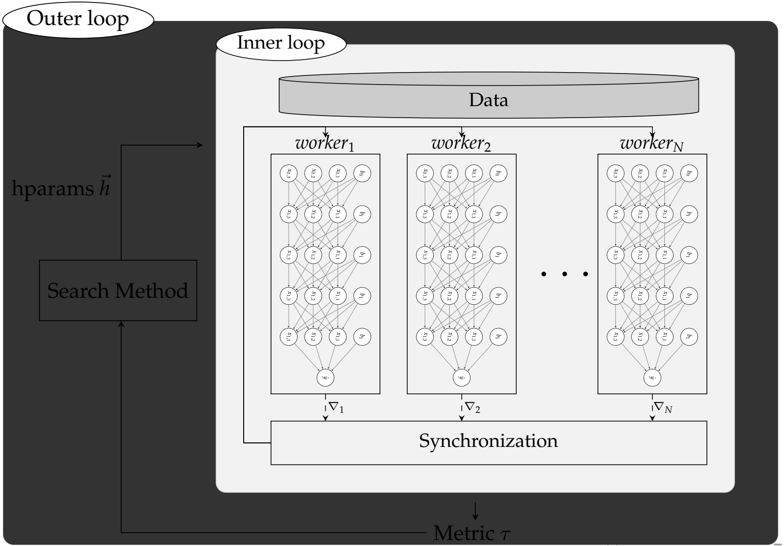
INNER AND OUTER LOOP OF LARGE SCALE DEEP LEARNING



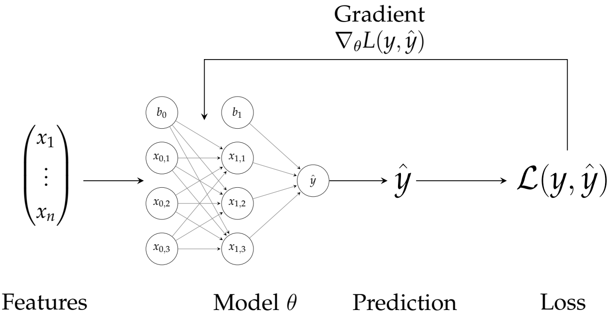
INNER AND OUTER LOOP OF LARGE SCALE DEEP LEARNING



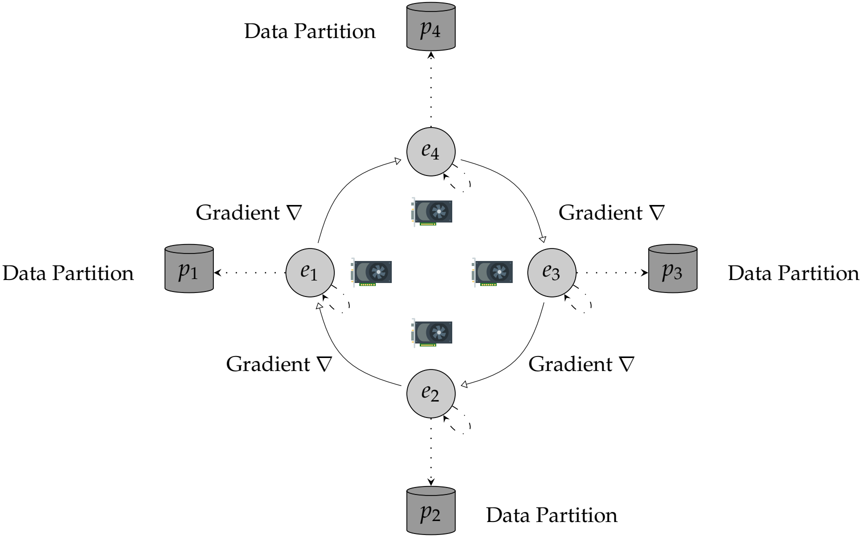
INNER AND OUTER LOOP OF LARGE SCALE DEEP LEARNING



INNER LOOP: DISTRIBUTED DEEP LEARNING

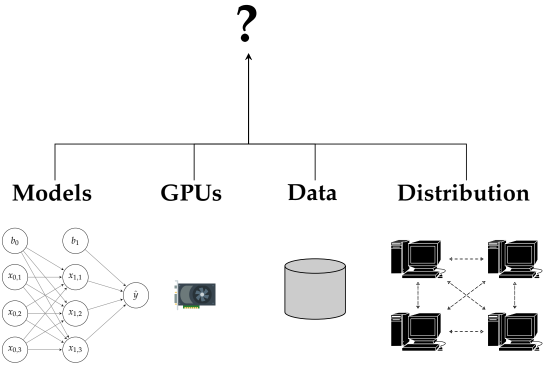


INNER LOOP: DISTRIBUTED DEEP LEARNING



DISTRIBUTED DEEP LEARNING IN PRACTICE

- Implementation of distributed algorithms is becoming a **commodity** (TF, PyTorch etc)
- **The hardest part of DDL is now**:
  - Cluster management
  - Allocating GPUs
  - Data management
  - Operations & performance



Hopsworks DDL Solution

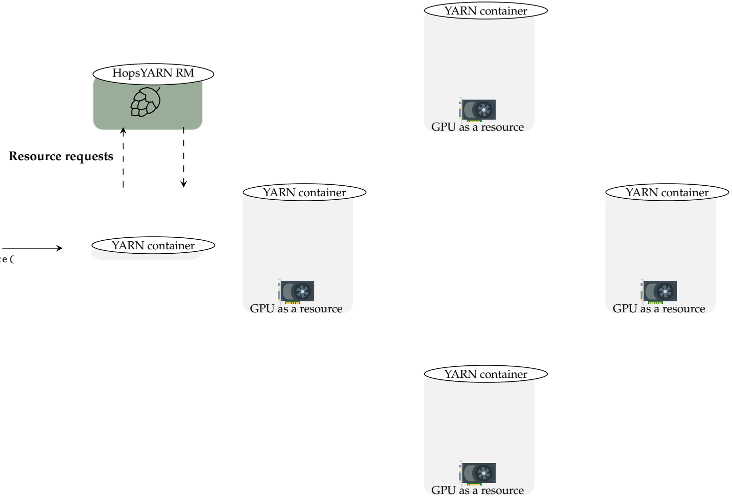
HOPSWORKS DDL SOLUTION

```python
from hops import experiment
experiment.collective_all_reduce(train_fn)
```

HOPSWORKS DDL SOLUTION

```python
from hops
import experiment
experiment.
collective_all_reduce(
train_fn
)
```

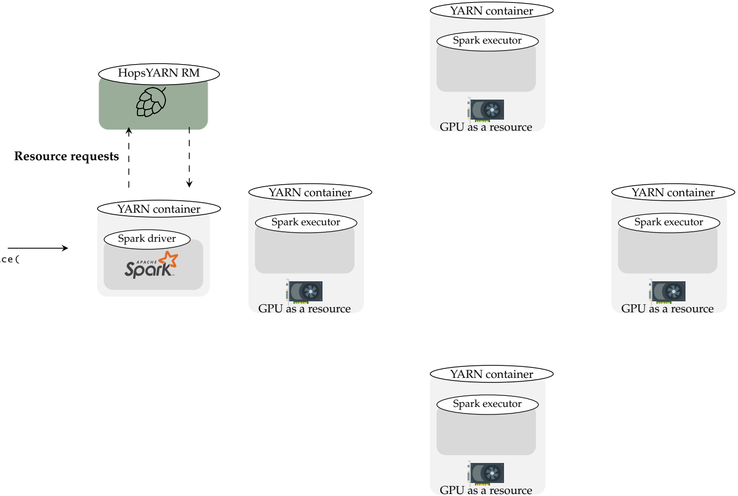
Client API



HOPSWORKS DDL SOLUTION

```
from hops
import experiment
experiment.
collective_all_reduce(
train_fn
)
```

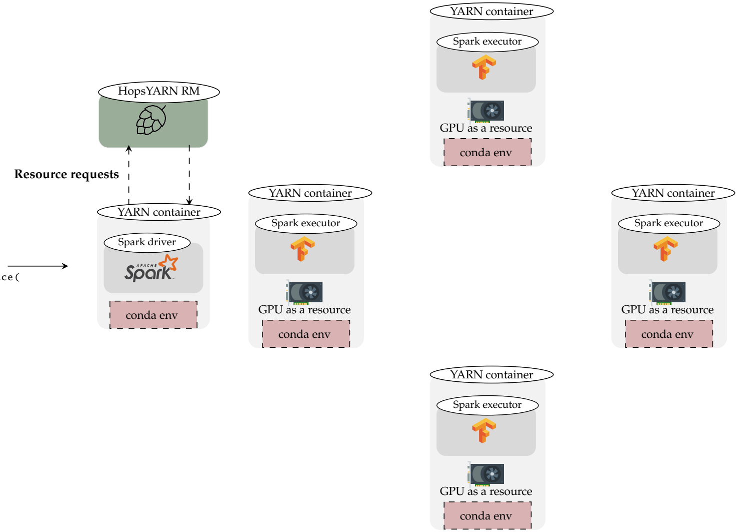
Client API



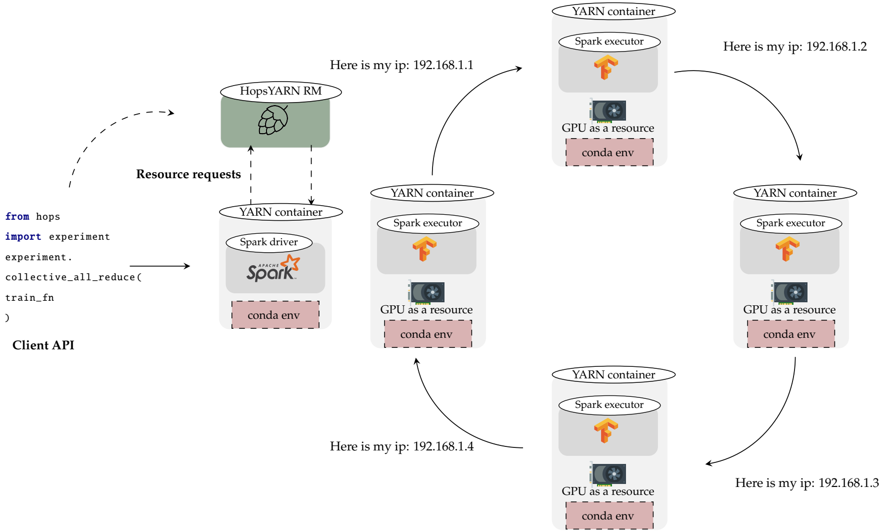
HOPSWORKS DDL SOLUTION

```
from hops
import experiment
experiment.
collective_all_reduce(
train_fn
)
```

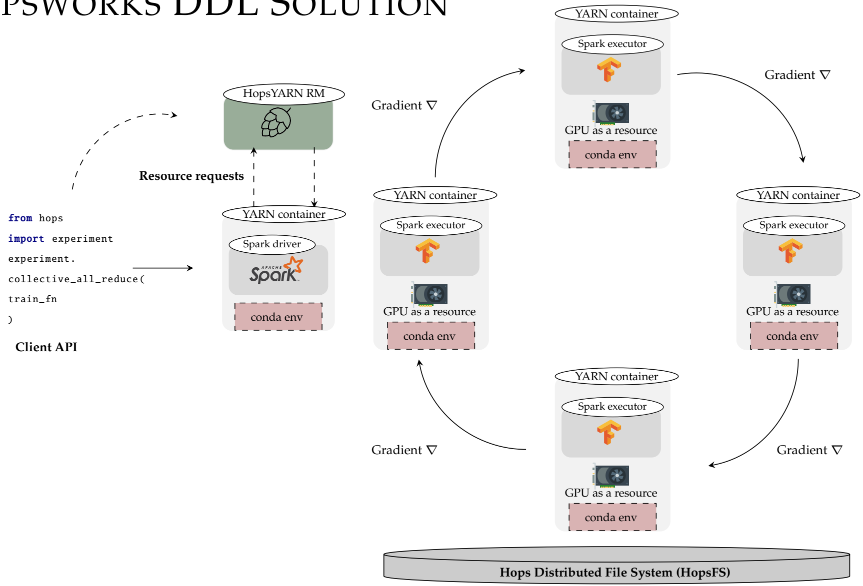
Client API



HOPSWORKS DDL SOLUTION



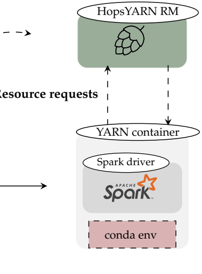
HOPSWORKS DDL SOLUTION

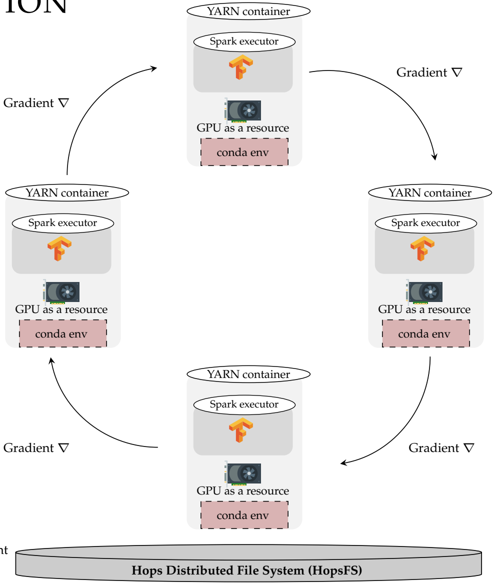


HOPSWORKS DDL SOLUTION

```
from hops
import experiment
experiment.
collective_all_reduce(
train_fn
)
```

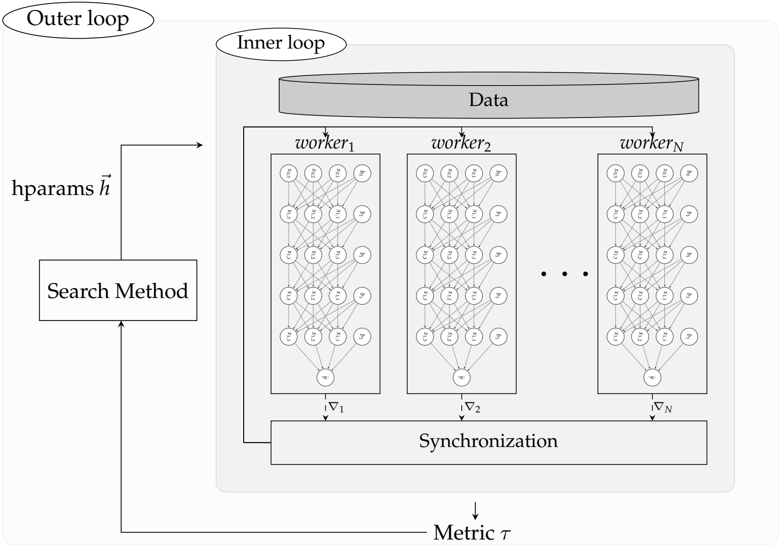
Client API



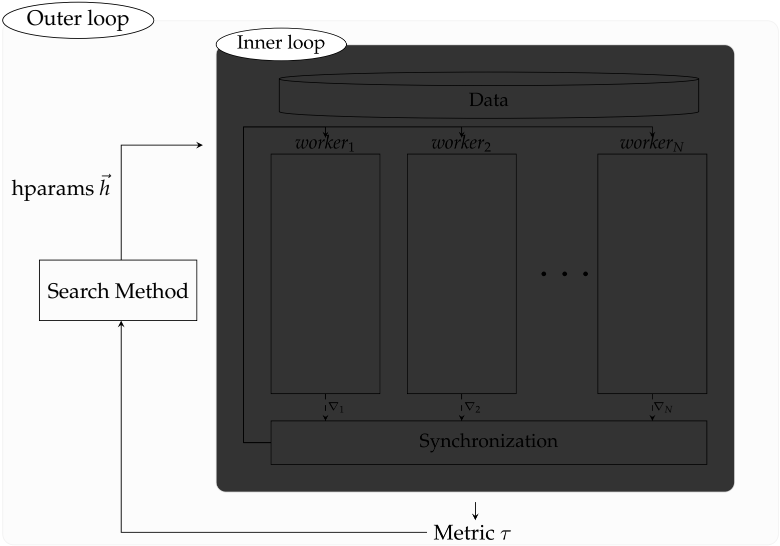


- Hide complexity behind simple API
- Allocate resources using pyspark
- Allocate GPUs for spark executors using HopsYARN
- Serve sharded training data to workers from HopsFS
- Use HopsFS for aggregating logs, checkpoints and results
- Store experiment metadata in metastore
- Use dynamic allocation for interactive resource management

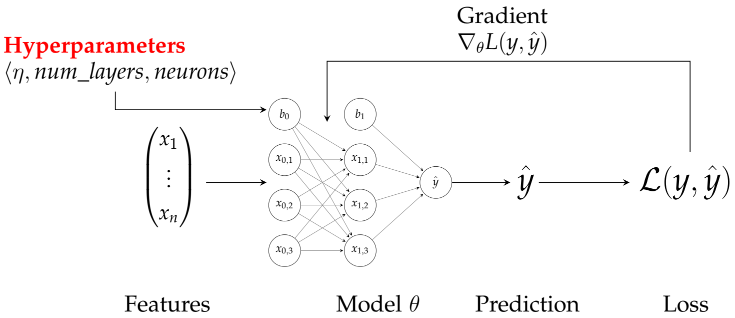
OUTER LOOP: BLACK BOX OPTIMIZATION



OUTER LOOP: BLACK BOX OPTIMIZATION



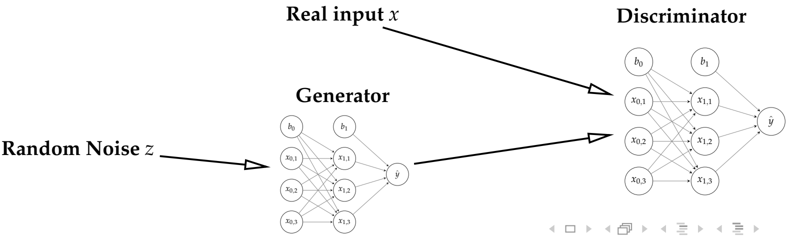
OUTER LOOP: BLACK BOX OPTIMIZATION



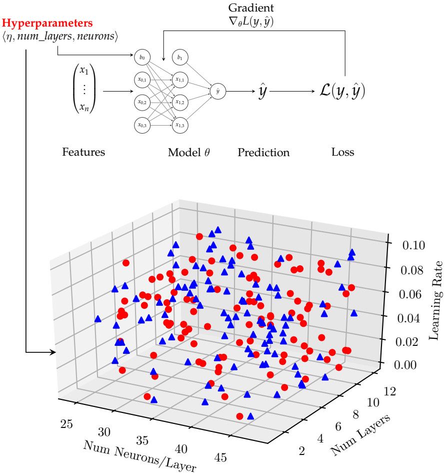
Outer Loop: Black Box Optimization

Example Use-Case from one of our clients:

- Goal: Train a One-Class GAN model for fraud detection
- Problem: GANs are extremely sensitive to hyperparameters and there exists a very large space of possible hyperparameters.
- Example hyperparameters to tune: learning rates η, optimizers, layers.. etc.

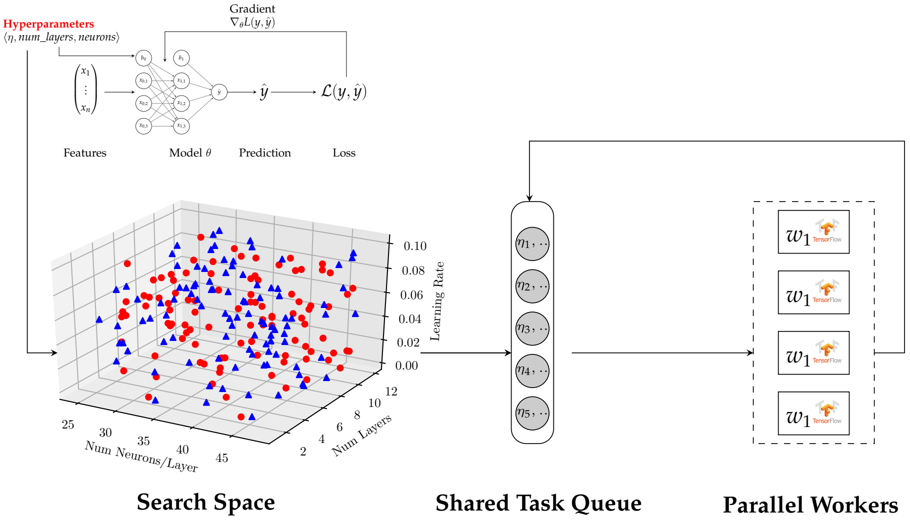


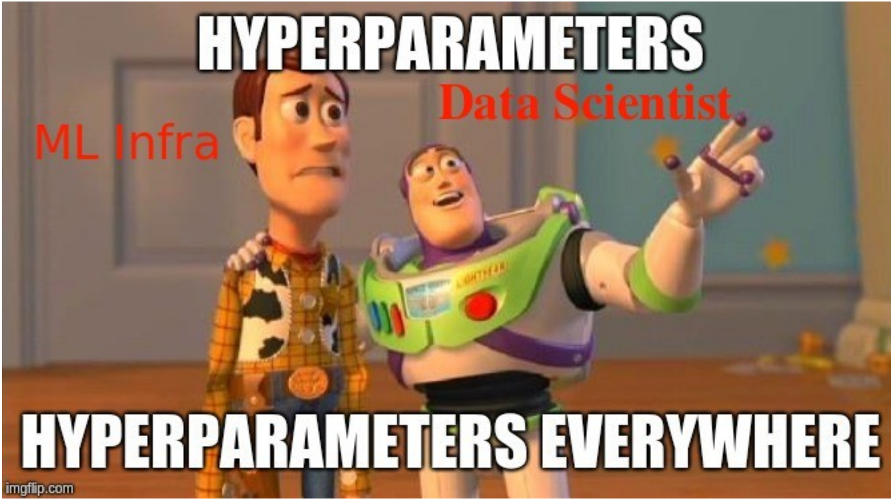
OUTER LOOP: BLACK BOX OPTIMIZATION



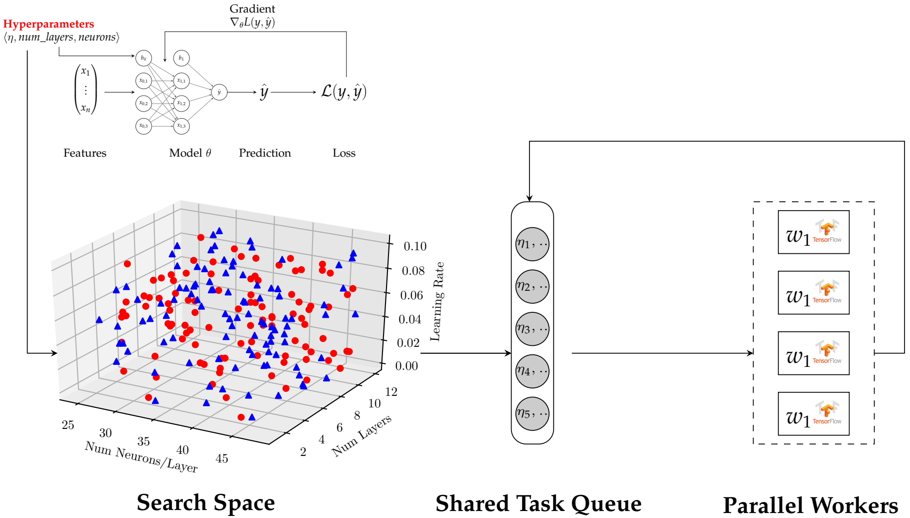
Search Space

OUTER LOOP: BLACK BOX OPTIMIZATION

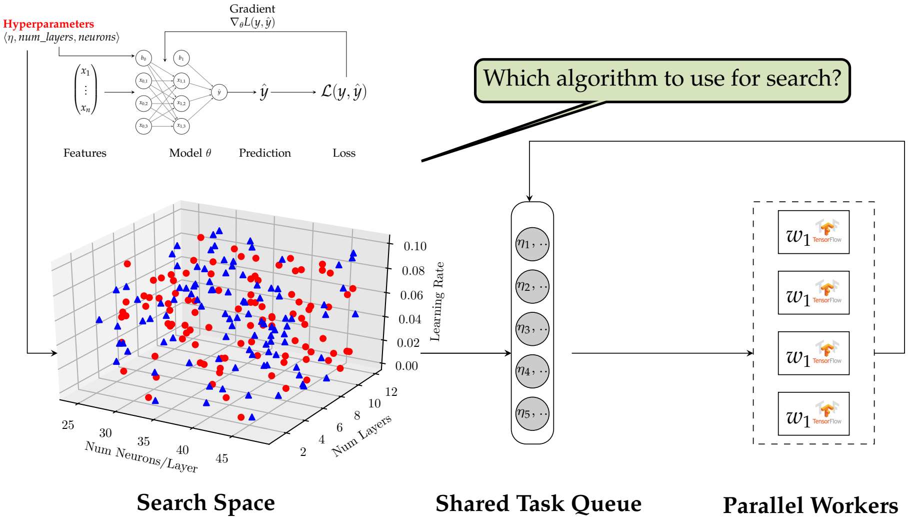



HYPERPARAMETERS
ML Infra
Data Scientist
HYPERPARAMETERS EVERYWHERE
imgflip.com

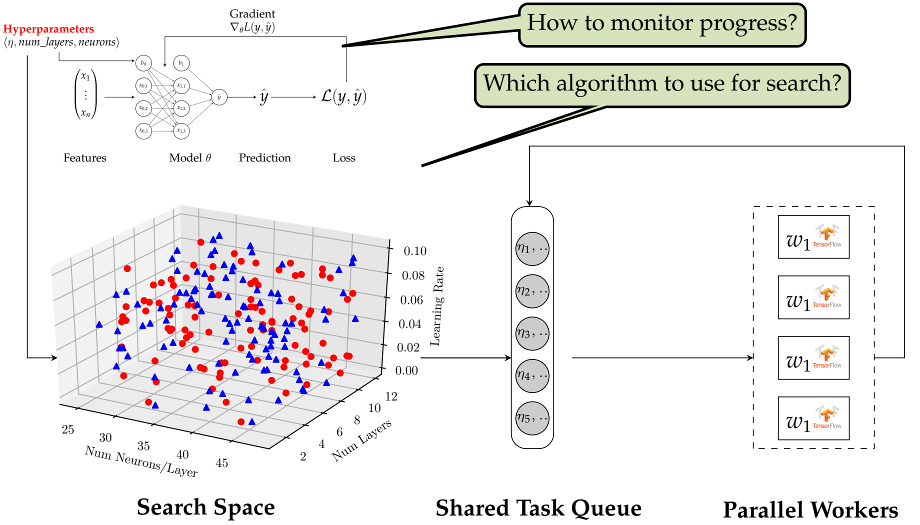
OUTER LOOP: BLACK BOX OPTIMIZATION



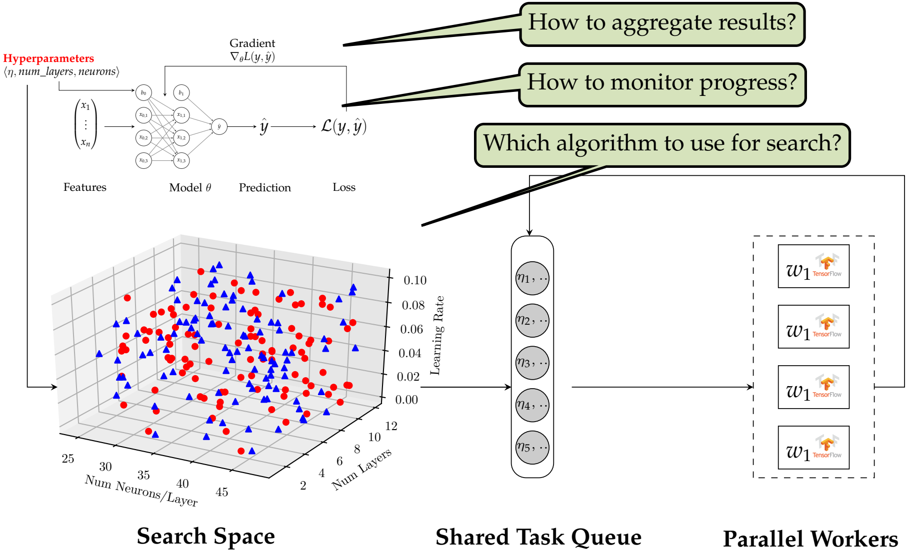
OUTER LOOP: BLACK BOX OPTIMIZATION



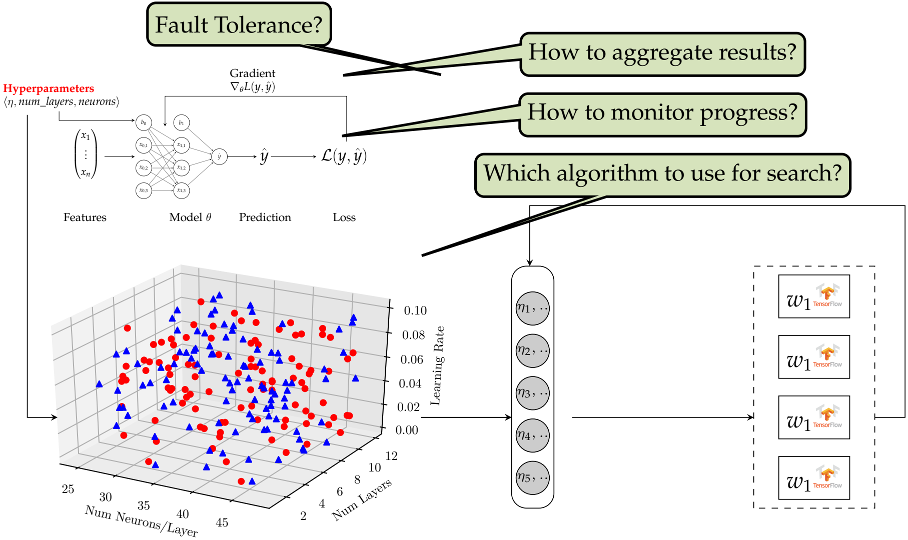
OUTER LOOP: BLACK BOX OPTIMIZATION



OUTER LOOP: BLACK BOX OPTIMIZATION



OUTER LOOP: BLACK BOX OPTIMIZATION



Search Space **Shared Task Queue** **Parallel Workers**

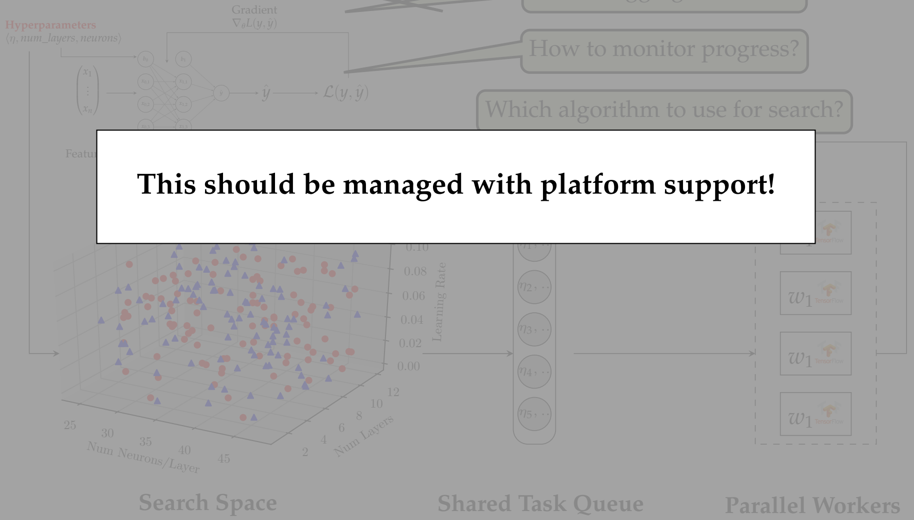
OUTER LOOP: BLACK BOX OPTIMIZATION

Fault Tolerance?

How to aggregate results?

How to monitor progress?

Which algorithm to use for search?



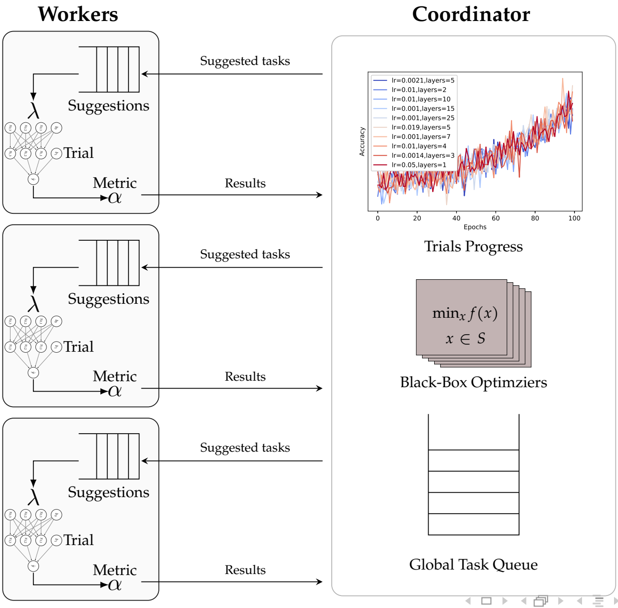
This should be managed with platform support!

Parallel Experiments

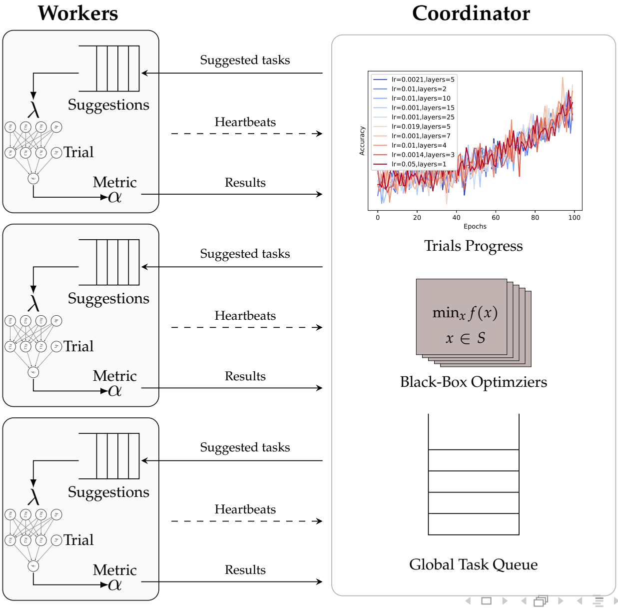
```python
from hops import experiment

experiment.random_search(train_fn)
```

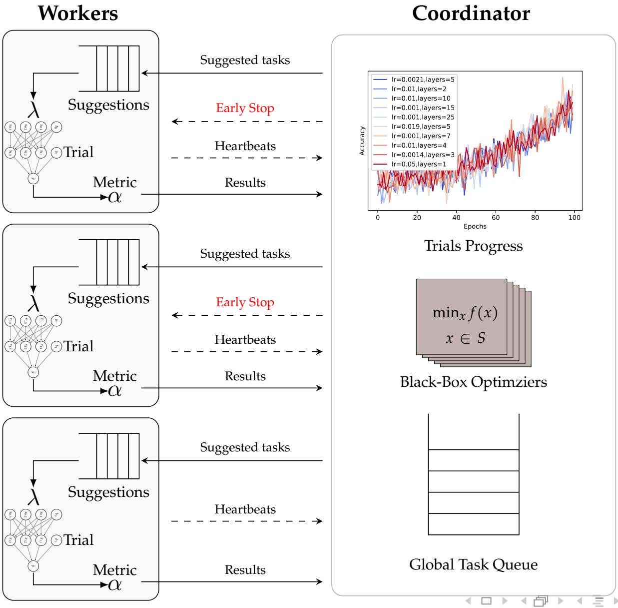
ASYNCHRONOUS SEARCH WORKFLOW



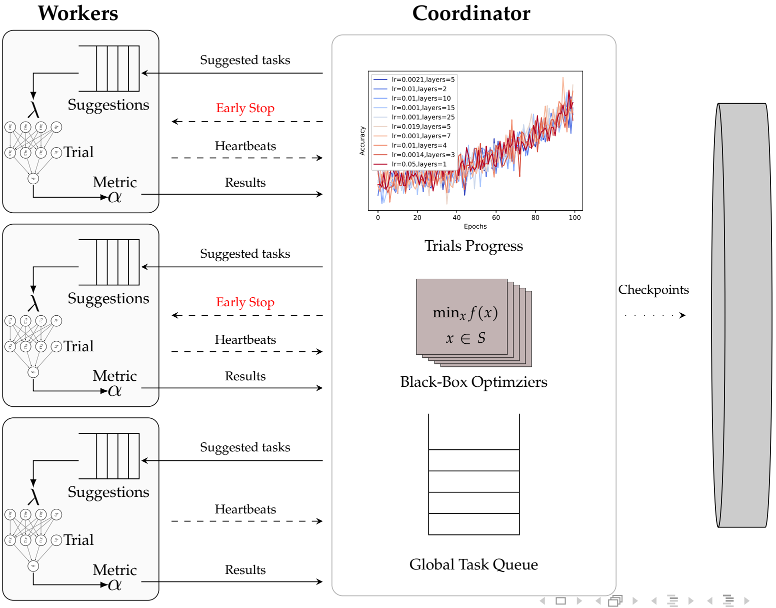
ASYNCHRONOUS SEARCH WORKFLOW



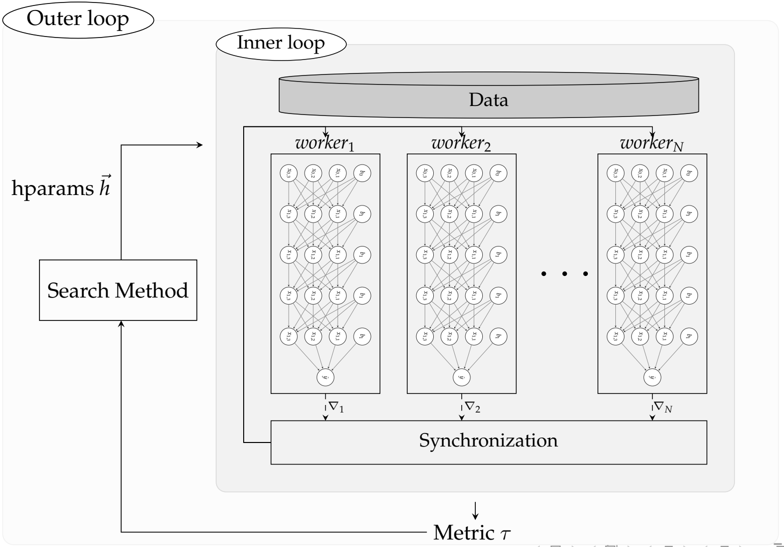
ASYNCHRONOUS SEARCH WORKFLOW



ASYNCHRONOUS SEARCH WORKFLOW



INNER AND OUTER LOOP OF LARGE SCALE DEEP LEARNING

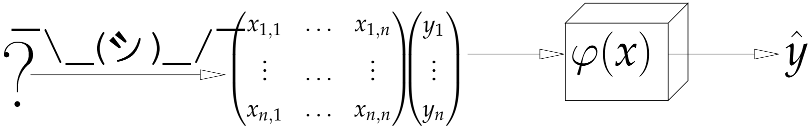


FEATURE STORE

FEATURE STORE

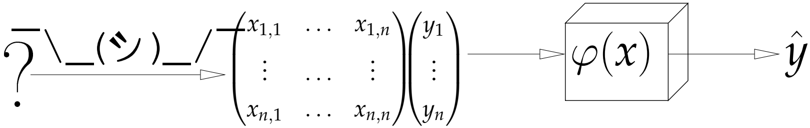
$$\begin{pmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \dots & \vdots \\ x_{n,1} & \dots & x_{n,n} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \longrightarrow \boxed{\varphi(x)} \longrightarrow \hat{y}$$

FEATURE STORE



2em1[6] **scaling_michelangelo**.

FEATURE STORE



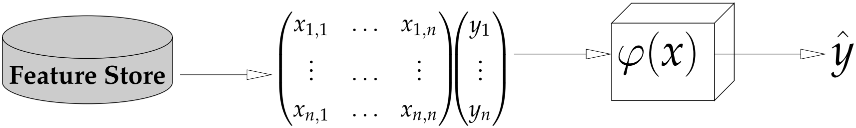
"Data is the hardest part of ML and the most important piece to get right.

Modelers spend most of their time selecting and transforming features at training time and then building the pipelines to deliver those features to production models."

- Uber[6]

2em1[6] **scaling_michelangelo**.

FEATURE STORE



"Data is the hardest part of ML and the most important piece to get right.

Modelers spend most of their time selecting and transforming features at training time and then building the pipelines to deliver those features to production models."

- Uber[7]

2em1[7] **scaling_michelangelo**.

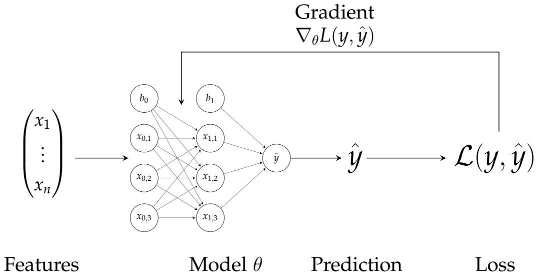
WHAT IS A FEATURE?

A feature is a measurable property of some data-sample

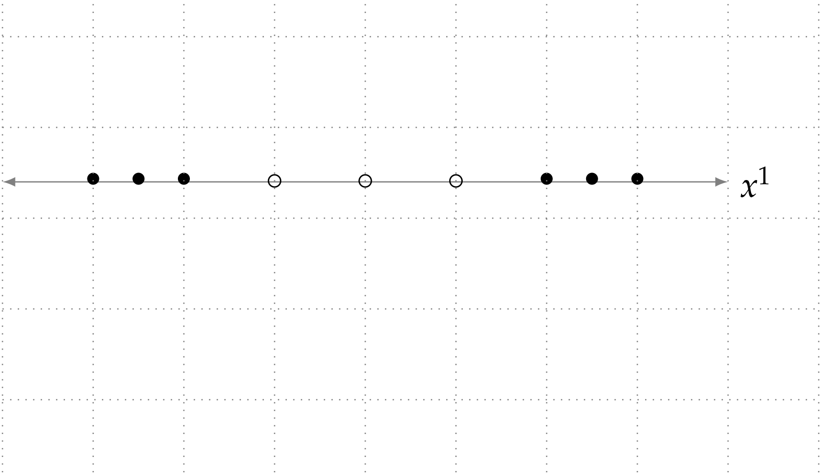
A feature could be..

- An aggregate value (min, max, mean, sum)
- A raw value (a pixel, a word from a piece of text)
- A value from a database table (the age of a customer)
- A derived representation: e.g an embedding or a cluster

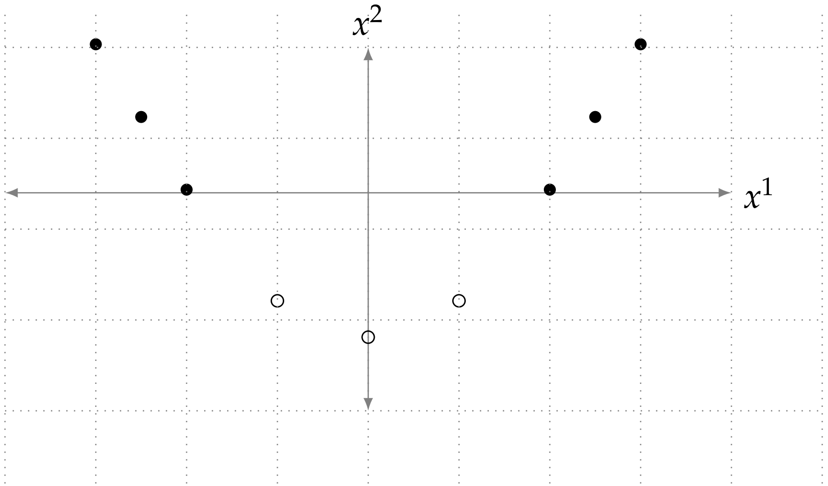
<u>Features</u> are the fuel for AI systems:



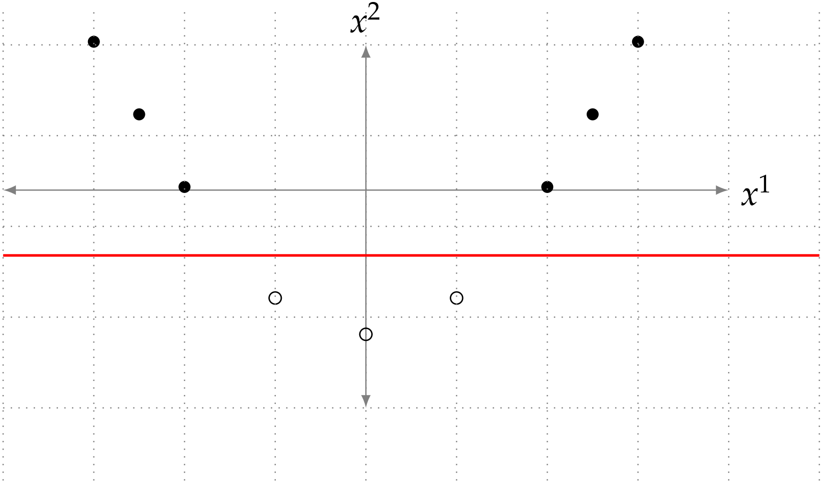
FEATURE ENGINEERING IS CRUCIAL FOR MODEL PERFORMANCE



FEATURE ENGINEERING IS CRUCIAL FOR MODEL PERFORMANCE



Feature Engineering is Crucial for Model Performance

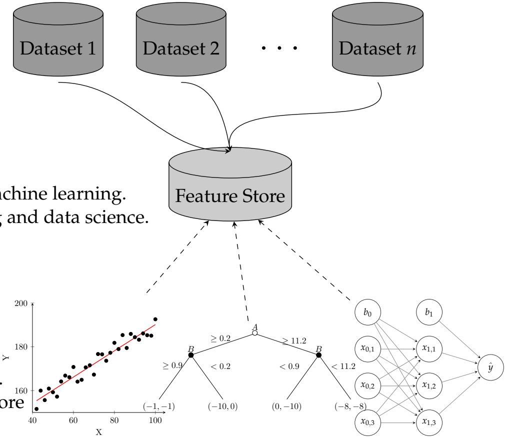


Disentangle Your ML Pipelines with a Feature Store

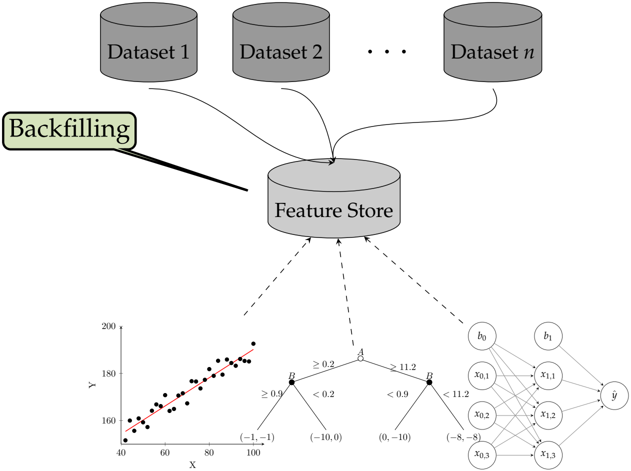
Data Sources

Feature Store
A data management platform for machine learning.
The interface between data engineering and data science.

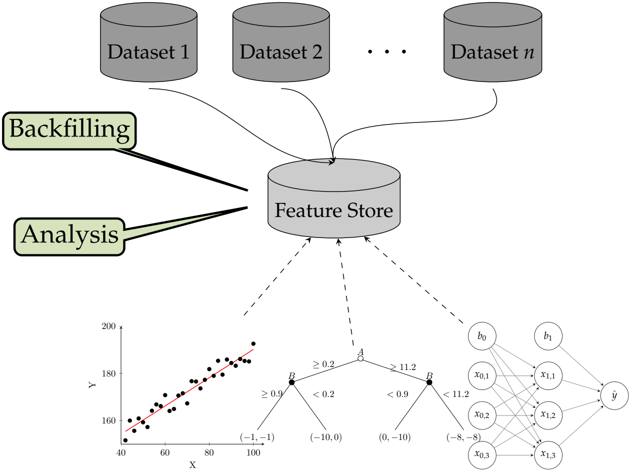
Models
Models are trained using sets of features.
The features are fetched from the feature store
and can overlap between models.



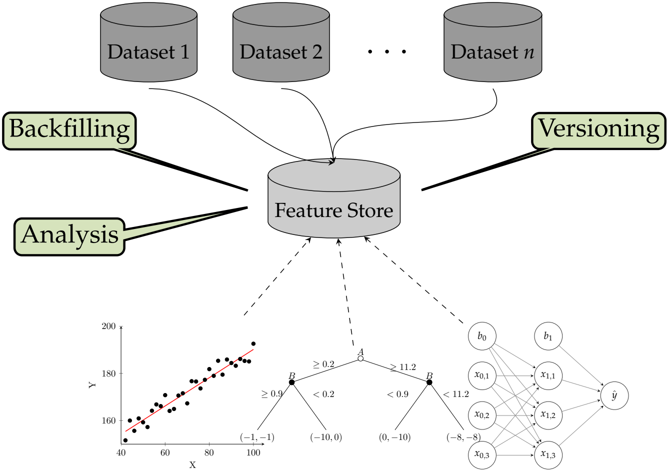
DISENTANGLE YOUR ML PIPELINES WITH A FEATURE STORE



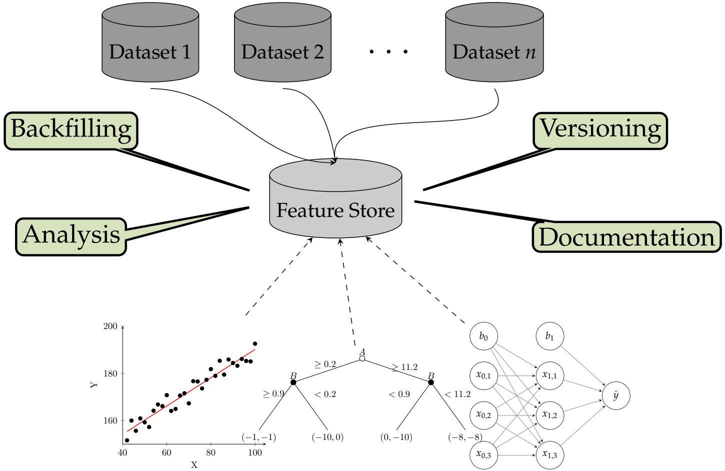
DISENTANGLE YOUR ML PIPELINES WITH A FEATURE STORE



DISENTANGLE YOUR ML PIPELINES WITH A FEATURE STORE



Disentangle Your ML Pipelines with a Feature Store



SUMMARY

- Deep Learning is going distributed
- Algorithms for DDL are available in several frameworks
- Applying DDL in practice brings a lot of operational complexity
- Hopsworks is a platform for scale out deep learning and big data processing
- Hopsworks makes DDL simpler by providing simple abstractions for distributed training, parallel experiments and much more..

@hopshadoop **@logicalclocks**

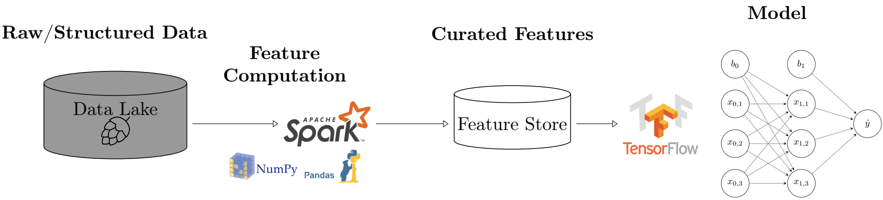
www.hops.io www.logicalclocks.com

LOGICAL CLOCKS

We are open source:
https://github.com/logicalclocks/hopsworks
https://github.com/hopshadoop/hops

Thanks to Logical Clocks Team: Jim Dowling, Seif Haridi, Theo Kakantousis, Fabio Buso, Gautier Berthou, Ermias Gebremeskel, Mahmoud Ismail, Salman Niazi, Antonios Kouzoupis, Robin Andersson, Alex Ormenisan, Rasmus Toivonen and Steffen Grohsschmiedt.

Demo-Setting



Hands-on Workshop

1. If you haven't registered, do it now on hops.site
2. Cheatsheet: http://snurran.sics.se/hops/kim/workshop_cheat.txt

Exercise 1 (Hello Hopsworks)

1. Create a Deep Learning Tour Project on Hopsworks
2. Start a Jupyter Notebook with the config:
   - "Experiment" Mode
   - 1 GPU
   - 4000 (MB) memory for the driver (appmaster)
   - 8000 (MB) memory for the executor
   - Rest can be default
3. Create a new "PySpark" notebook
4. In the first cell, write:

```
print("Hello Hopsworks")
```

5. Execute the cell (Ctrl + <Enter>)

EXERCISE 2 (DISTRIBUTED HELLO HOPSWORKS WITH GPU)

1. Add a new cell with the contents:

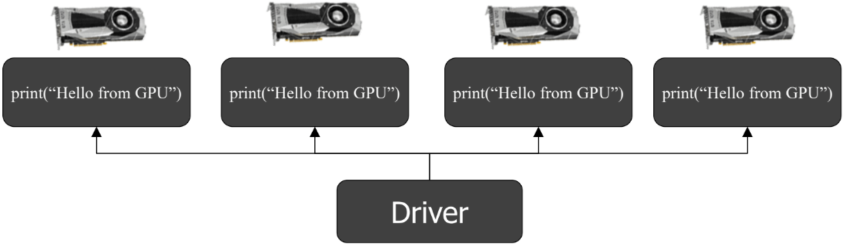
```python
def executor():
    print("Hello from GPU")
```

2. Add a new cell with the contents:

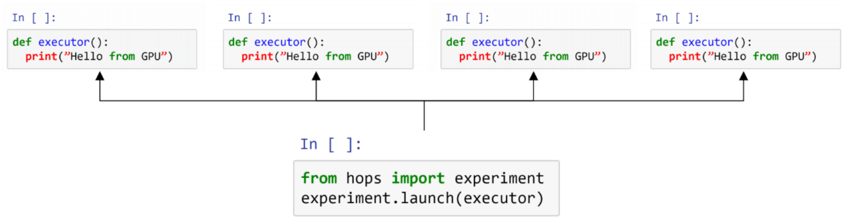
```python
from hops import experiment
experiment.launch(executor)
```

3. Execute the two cells in order (Ctrl + <Enter>)
4. Go to the Application UI

Exercise 2 (Distributed Hello Hopsworks With GPU)



EXERCISE 2 (DISTRIBUTED HELLO HOPSWORKS WITH GPU)



EXERCISE 3 (LOAD MNIST FROM HOPSFS)

1. Add a new cell with the contents:

```
from hops import hdfs
import tensorflow as tf
def create_tf_dataset():
    train_files = [hdfs.project_path() +
                    "TestJob/data/mnist/train/train.tfrecords"]
    dataset = tf.data.TFRecordDataset(train_files)
    def decode(example):
        example = tf.parse_single_example(example,{
                    'image_raw': tf.FixedLenFeature([], tf.string),
                    'label': tf.FixedLenFeature([], tf.int64)})
        image = tf.reshape(tf.decode_raw(example['image_raw'],
                        tf.uint8), (28,28,1))
        label = tf.one_hot(tf.cast(example['label'], tf.int32), 10)
        return image, label
    return dataset.map(decode).batch(128).repeat()
```

EXERCISE 3 (LOAD MNIST FROM HOPSFS)

1. Add a new cell with the contents:

```
create_tf_dataset()
```

2. Execute the two cells in order (Ctrl + <Enter>)

Exercise 4 (Define CNN Model)

```
from tensorflow import keras
def create_model():
    model = keras.Sequential()
    model.add(keras.layers.Conv2D(filters=32, kernel_size=3, padding='same',
                                  activation='relu', input_shape=(28,28,1)))
    model.add(keras.layers.BatchNormalization())
    model.add(keras.layers.MaxPooling2D(pool_size=2))
    model.add(keras.layers.Dropout(0.3))
    model.add(keras.layers.Conv2D(filters=64, kernel_size=3,
                                  padding='same', activation='relu'))
    model.add(keras.layers.BatchNormalization())
    model.add(keras.layers.MaxPooling2D(pool_size=2))
    model.add(keras.layers.Dropout(0.3))
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(128, activation='relu'))
    model.add(keras.layers.Dropout(0.5))
    model.add(keras.layers.Dense(10, activation='softmax'))
    return model
```

EXERCISE 4 (DEFINE CNN MODEL)

1. Add a new cell with the contents:

   ```
   create_model().summary()
   ```

2. Execute the two cells in order (Ctrl + <Enter>)

EXERCISE 5 (DEFINE & RUN THE EXPERIMENT)

1. Add a new cell with the contents:

```
from hops import tensorboard
from tensorflow.python.keras.callbacks import TensorBoard
def train_fn():
    dataset = create_tf_dataset()
    model = create_model()
    model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),metrics=['accuracy'])
    tb_callback = TensorBoard(log_dir=tensorboard.logdir())
    model_ckpt_callback = keras.callbacks.ModelCheckpoint(
                          tensorboard.logdir(), monitor='acc')
    history = model.fit(dataset, epochs=50,
                steps_per_epoch=80, callbacks=[tb_callback])
    return history.history["acc"][-1]
```

EXERCISE 5 (DEFINE & RUN THE EXPERIMENT)

1. Add a new cell with the contents:

```
experiment.launch(train_fn)
```

2. Execute the two cells in order (Ctrl + <Enter>)
3. Go to the Application UI and monitor the training progress

REFERENCES

- Example notebooks https://github.com/logicalclocks/hops-examples
- HopsML[8]
- Hopsworks[9]
- Hopsworks' feature store[10]
- Maggy https://github.com/logicalclocks/maggy

2em1[8] Logical Clocks AB. *HopsML: Python-First ML Pipelines.* https://hops.readthedocs.io/en/latest/hopsml/hopsML.html. 2018.

2em1[9] Jim Dowling. *Introducing Hopsworks.* https://www.logicalclocks.com/introducing-hopsworks/. 2018.

2em1[10] Kim Hammar and Jim Dowling. *Feature Store: the missing data layer in ML pipelines?* https://www.logicalclocks.com/feature-store/. 2018.