

NOTICES AND DISCLAIMERS

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration.

No product or component can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. For more complete information about performance and benchmark results, visit <http://www.intel.com/benchmarks>.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit <http://www.intel.com/benchmarks>.

Intel® Advanced Vector Extensions (Intel® AVX) provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more at <http://www.intel.com/go/turbo>.

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Agenda

15:45-17:00, Distributed Training and Federated Learning

- **15:45 – 16:15** Distributed Deep Learning Training
- **16:15 – 16:45** Federated Learning
- **16:45 – 17:00** Quiz Time

DISTRIBUTED DEEP LEARNING TRAINING

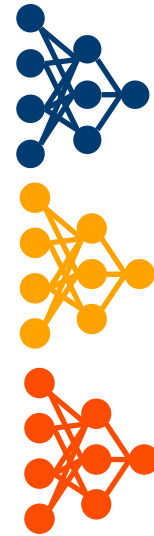


DISTRIBUTED TRAINING

MODEL PARALLELISM

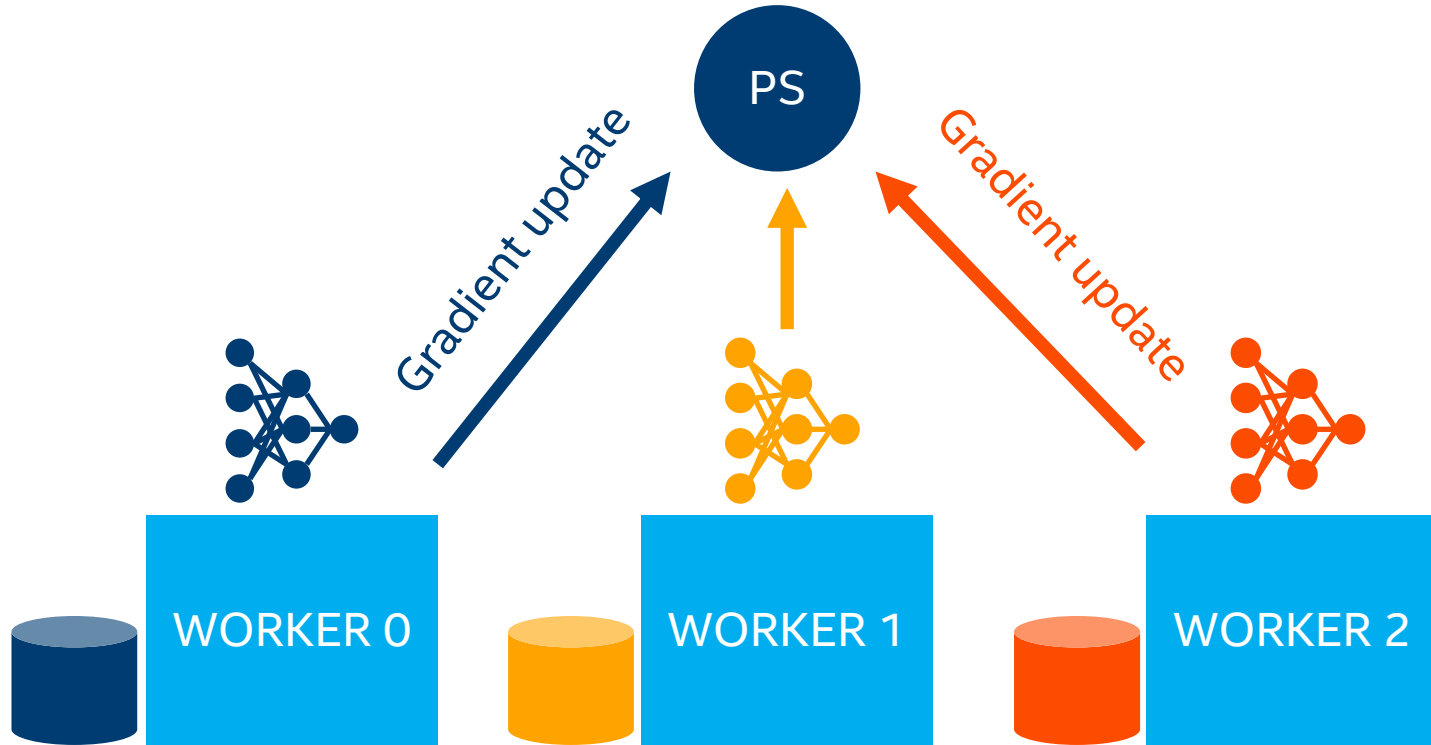


DATA PARALLELISM



A worker could be a CPU, a GPU, a socket, or multiple cores.

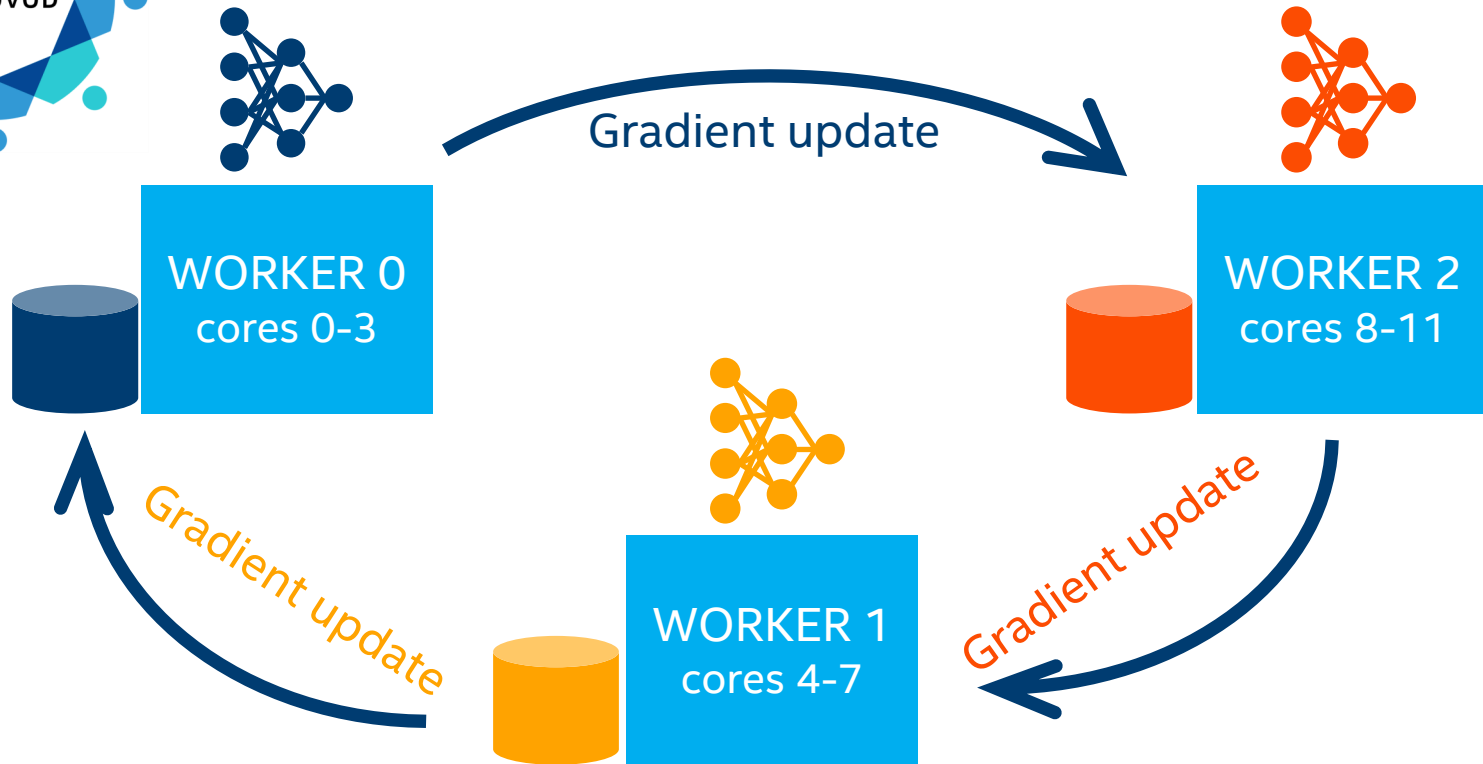
PARAMETER SERVER



Tree using gRPC calls



HOROVOD



<https://arxiv.org/abs/1802.05799v3>

MESSAGE PASSING INTERFACE (MPI)

```
$ mpirun -H 192.168.1.100,192.168.1.105 hostname  
aipg-infra-07.intel.com  
aipg-infra-09.intel.com
```

```
$ mpirun -H host1,host2,host3 python hello.py  
Hello World!  
Hello World!  
Hello World!
```

CHANGES TO TENSORFLOW

1

```
import tensorflow as tf
import horovod.tensorflow as hvd
```

2

```
hvd.init()
```

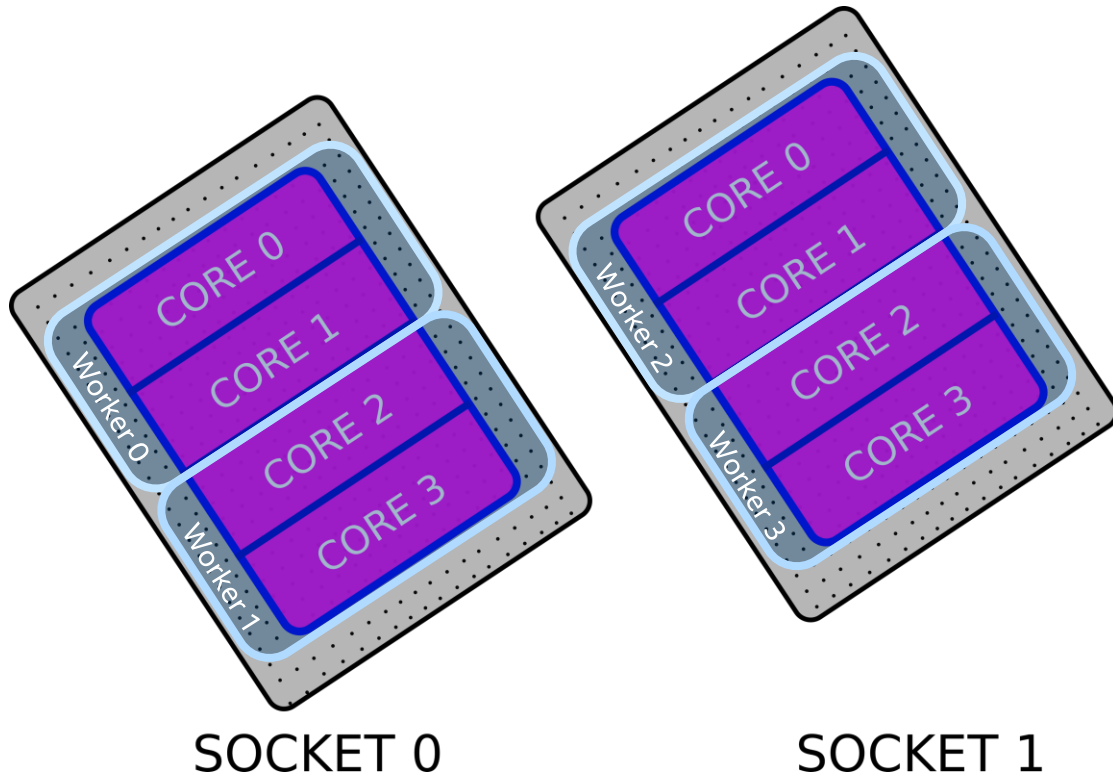
3

```
opt = tf.train.AdagradOptimizer(0.01 * hvd.size())
opt = hvd.DistributedOptimizer(opt)
```

4

```
hooks = [hvd.BroadcastGlobalVariablesHook(0)]
```


SOCKETS & CORES



SOCKET

Receptacle on the motherboard for one physically packaged processor.

CORE

A complete private set of registers, execution units, and queues to execute a program.

MULTIPLE WORKERS PER CPU

```
$ mpirun  
-H hostA,hostB,hostC  
-np 6  
--map-by ppr:1:socket:pe=2  
--oversubscribe  
--report-bindings  
python train_model.py
```

MULTIPLE WORKERS PER CPU

```
$ mpirun  
-H hostA, hostB, hostC  
-n 6  
-ppn 2  
-print-rank-map  
-genv I_MPI_PIN_DOMAIN=socket  
-genv OMP_NUM_THREADS=24  
-genv OMP_PROC_BIND=true  
-genv KMP_BLOCKTIME=1  
python train_model.py
```

MULTIPLE WORKERS PER CPU

| | | SOCKET 0 | SOCKET 1 |
|----|-------|---------------|---------------|
| R0 | hostA | [BB/BB/././.] | [./././././.] |
| R1 | hostA | [./././././.] | [BB/BB/././.] |
| R2 | hostB | [BB/BB/././.] | [./././././.] |
| R3 | hostB | [./././././.] | [BB/BB/././.] |
| R4 | hostC | [BB/BB/././.] | [./././././.] |
| R5 | hostC | [./././././.] | [BB/BB/././.] |

HOROVODRUN COMMAND

```
horovodrun -np 4 -H node-4984:2,node-4985:2 --binding-args="--  
map-by ppr:2:socket:pe=10" --mpi-args="--report-bindings" python  
train_horovod.py
```

 [facebookincubator](#) / [gloo](#)

About

Collective communications library with various primitives for multi-machine training.

Advanced: Run Horovod with Open MPI

In some advanced cases you might want fine-grained control over options passed to Open MPI. To learn how to run Horovod training directly using Open MPI, read [Run Horovod with Open MPI](#).

Run Horovod with Intel(R) MPI

`horovodrun` automatically converts some parameters to the format supported by Intel(R) MPI `mpirun`. The set of allowed options includes `-np`, `-H` and ssh arguments (`-p`, `-i`). Intel(R) MPI `mpirun` does not support MCA parameters, but you can set some of the options via [environment variables](#). For additional information refer to [Intel\(R\) MPI official documentation](#).

IntelAI / unet

Watch 15 Star 149 Fork 67

Code Issues 3 Pull requests Actions Projects Wiki Security

master unet / 3D / Go to file Add file

tonyreina Update 3D notebook 2 days ago History

| | | |
|------------------------|---|--------------|
| .. | | |
| images | Adding plot | 3 months ago |
| 3d_unet.ipynb | Update 3D notebook | 2 days ago |
| README.md | Update README.md | 3 months ago |
| argparser.py | Adding 3D inference notebook | 3 months ago |
| dataloader.py | rotate 90 on 3D data loader to give correct orientation | 3 months ago |
| horovod_command.sh | Updating license | 3 months ago |
| model.py | Updating 2D to TensorFlow 2 and OpenVINO 2021 | 3 months ago |
| plot_predictions.ipynb | Updating license | 3 months ago |
| quantize_model.ipynb | Updating license | 3 months ago |
| run_unet_horovod.sh | Updating to new data loader | 4 months ago |
| script_slurm | Updating Horovod script | 4 months ago |
| settings.py | Adding 3D inference notebook | 3 months ago |
| train.py | Updating 3D print | 3 months ago |
| train_horovod.py | Updating 2D to TensorFlow 2 and OpenVINO 2021 | 3 months ago |



BKC/BKM FOR HPC AI

WHITE PAPER



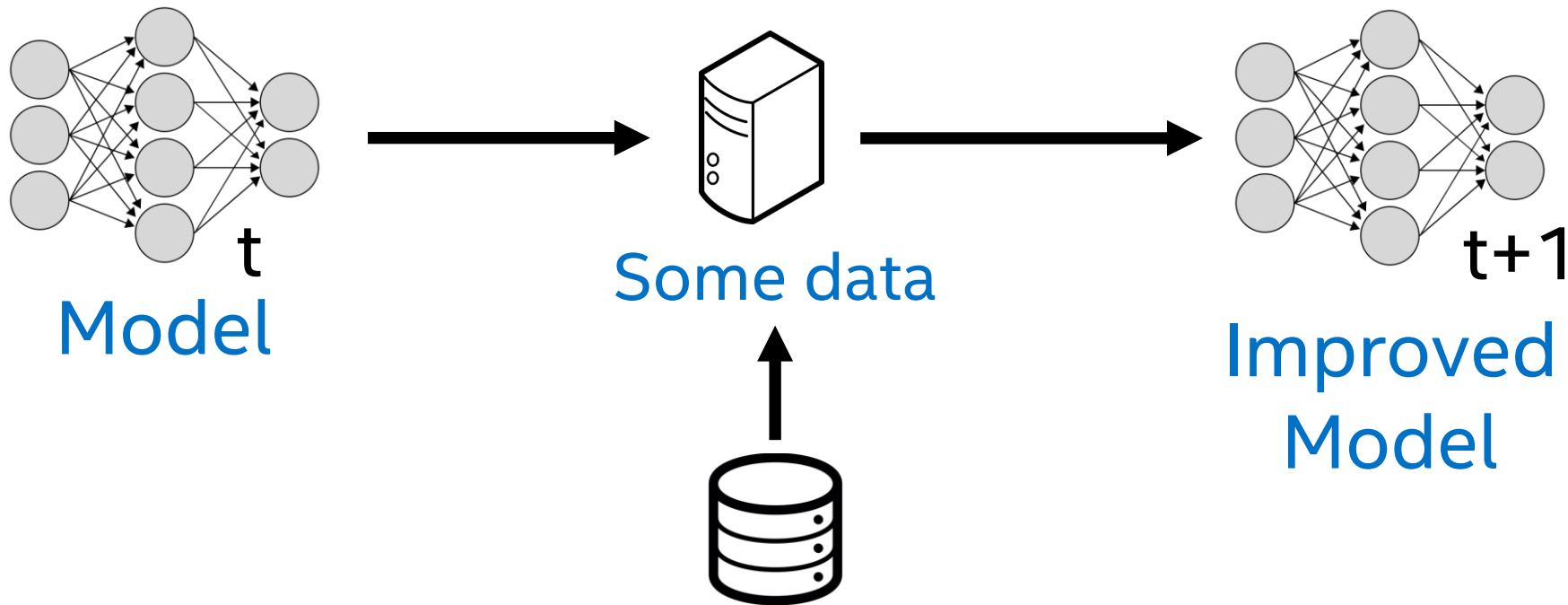
Best Practices for Scaling Deep Learning
Training and Inference with TensorFlow* On
Intel® Xeon® Processor-Based HPC
Infrastructures

Version: 1.1
Date of Issue: January 2019
Prepared By: Aishwarya Bhandare[¶], Deepthi Karkada[¶], Kushal Datta[¶], Anupama Kurpad[¶], Vamsi Sripathi[¶], Sun Choi[¶], Vikram Saletore[¶]
[¶]Connectivity Group & [¶]AI Products Group, Data Center Group
Customer Solutions Technical Enabling, Intel Corporation

- Docker
- SLURM
- Singularity
- NFS
- Lustre

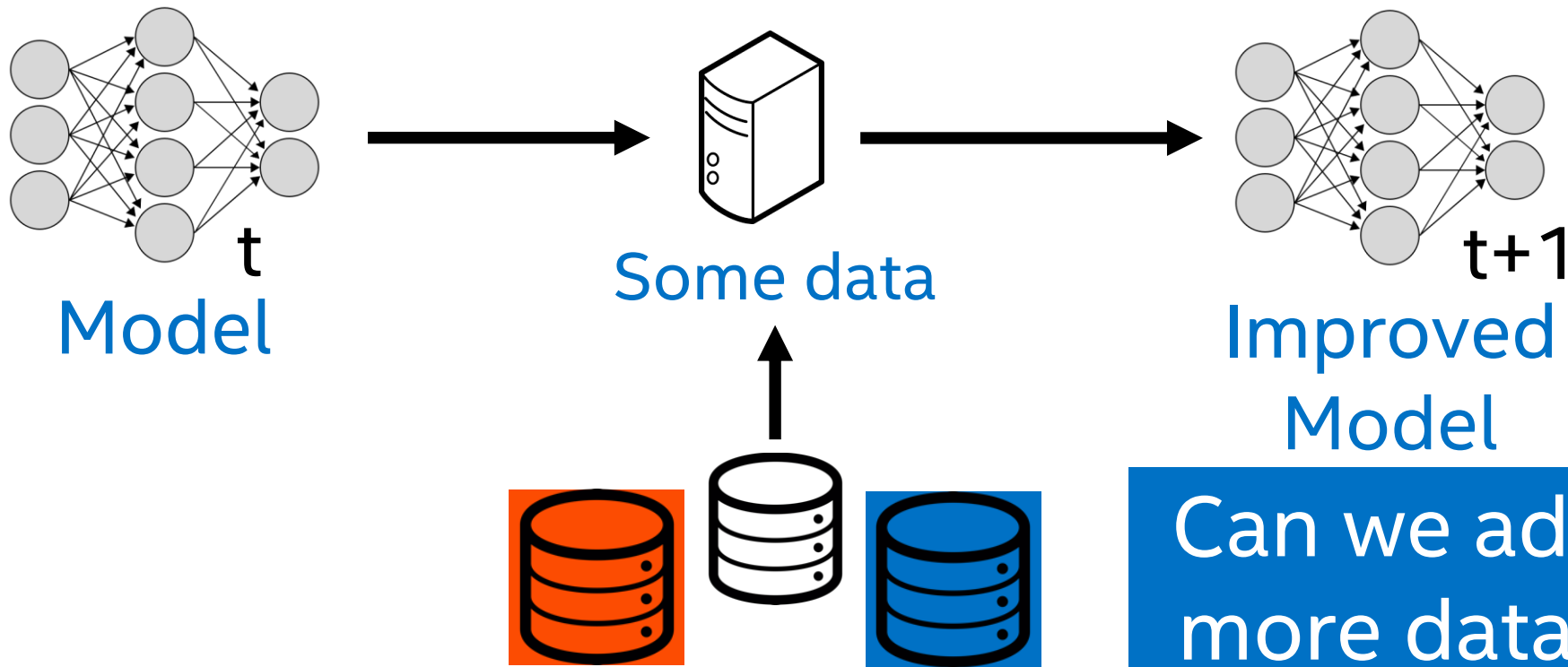
FEDERATED LEARNING

THE DATA SILO PROBLEM



Eventually, we hit the limit of our dataset.

THE DATA SILO PROBLEM



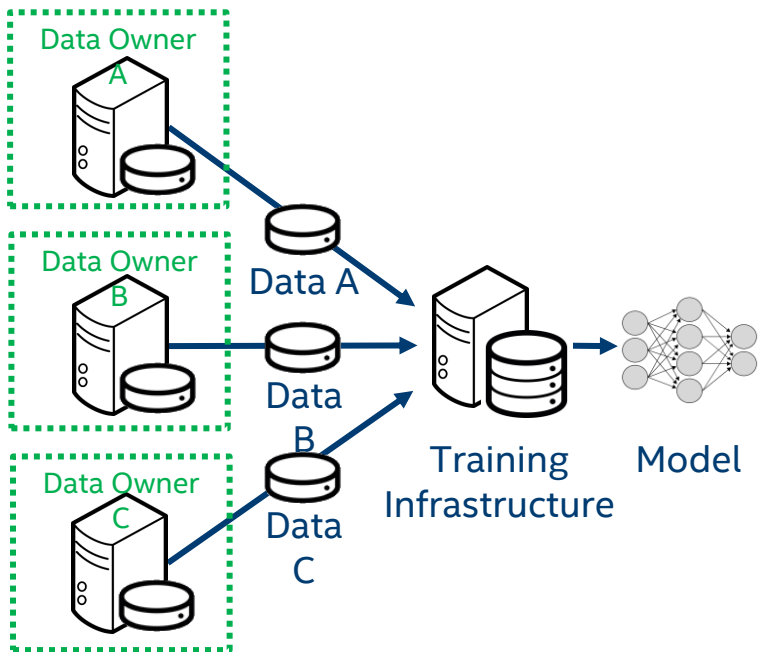
Challenges for Training AI Models?



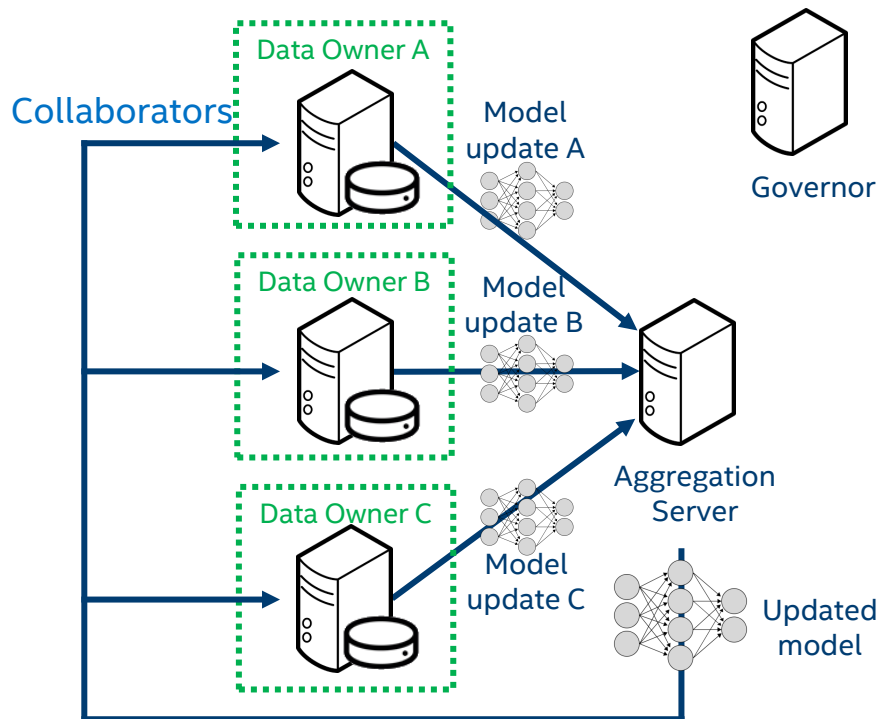
- Data is legally protected (HIPAA, GDPR)
- Data is sensitive
- Data too valuable or value unknown
- Data too large to transmit

Centralized Learning vs Federated Learning

Centralized Learning



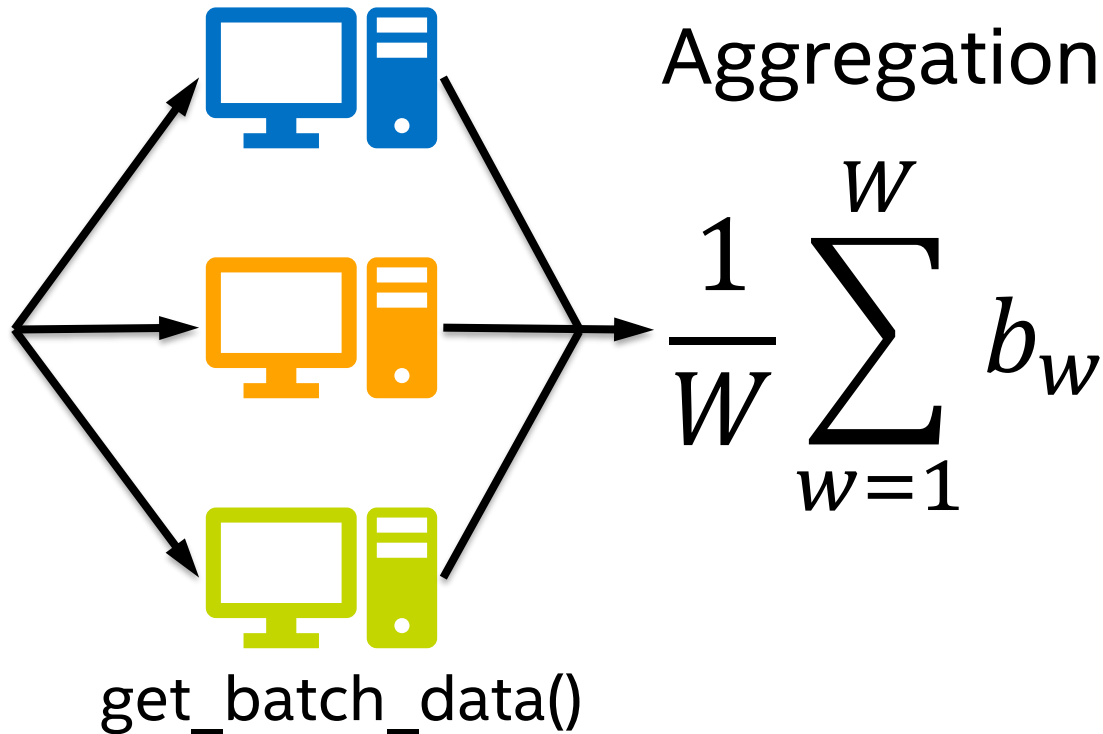
Federated Learning



50 Foot View of Federated Learning



```
169 prediction = K.layers.Conv3D(name="PredictionMask",  
170                               filters=number_output_classes,  
171                               kernel_size=(1, 1, 1),  
172                               activation="sigmoid")(convOut)  
173  
174 model = K.models.Model(inputs=[inputs], outputs=[prediction],  
175                        name=model_name)
```





scientific reports

[Explore our content](#) ▾ [Journal information](#) ▾

[nature](#) > [scientific reports](#) > [articles](#) > [article](#)

Article | [Open Access](#) | Published: 28 July 2020

Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data

Micah J. Sheller, Brandon Edwards, G. Anthony Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka R. Colen & Spyridon Bakas [✉](#)

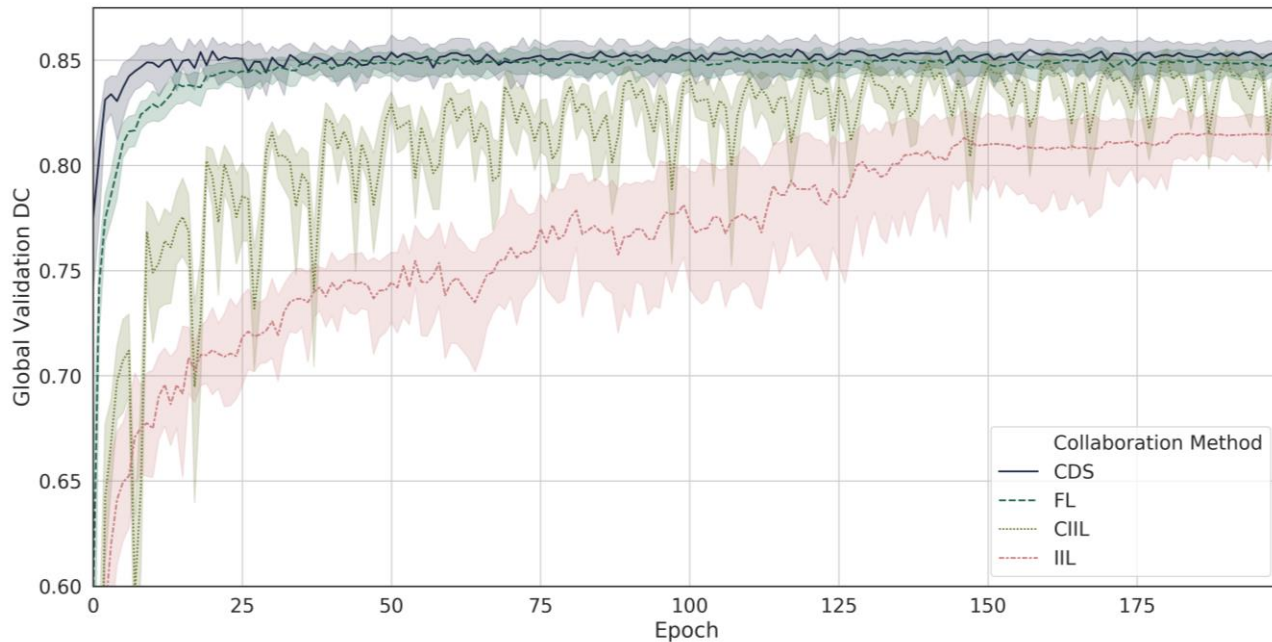
Scientific Reports **10**, Article number: 12598 (2020) | [Cite this article](#)

4738 Accesses | **2** Citations | **121** Altmetric | [Metrics](#)

Abstract

Several studies underscore the potential of deep learning in identifying complex patterns, leading to diagnostic and prognostic biomarkers. Identifying sufficiently large and diverse datasets, required for training, is a significant challenge in medicine and can rarely be found in individual institutions. Multi-institutional collaborations based on centrally-shared patient data

FEDERATING THE U-NET TRAINING [ORIGINAL INSTITUTIONS]*



How much better does each institution do when training on the full data vs. just their own data?

- ~ **17%** better on the hold-out BraTS data
- ~ **2.6%** better on their own validation data

Navigation

Project description

Release history

Download files

Project links

Homepage

Source Code

Documentation

Bug Tracker

Statistics

GitHub statistics:

★ Stars: 74

🔗 Forks: 15

📄 Open issues/PRs: 7

View statistics for this project via [Libraries.io](#), or by using our [public dataset](#) on [Google BigQuery](#)

Meta

License: Apache Software License

Project description

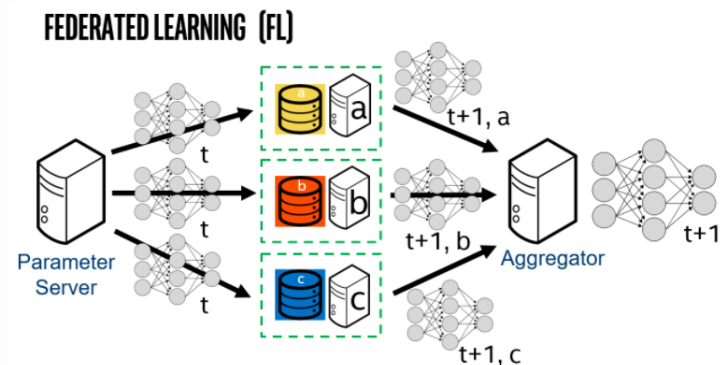
Welcome to Intel® Open Federated Learning

python 3.6 | 3.7 | 3.8 build passing docs passing pypi v1.0.1 slack @openfl License Apache 2.0

[Federated learning](#) is a distributed machine learning approach that enables organizations to collaborate on machine learning projects without sharing sensitive data, such as, patient records, financial data, or classified secrets ([Sheller MJ, et al., 2020](#); [Sheller MJ, et al., 2019](#); [Yang Y, et al., 2019](#); [McMahan HB, et al., 2016](#)).

The basic premise behind federated learning is that the model moves to meet the data rather than the data moving to meet the model. Therefore, the minimum data movement needed across the federation is solely the model parameters and their updates.

Open Federated Learning (OpenFL) is a Python 3 project developed by Intel Labs and Intel Internet of Things Group.



github.com/intel/openfl
openfl.readthedocs.io/



OpenFL Interfaces

Python API

Experimentation, Single Node

Federated Keras MNIST Tutorial

```
In [ ]: #Install dependencies if not already installed
!pip install tensorflow mnist

In [ ]: import numpy as np
import mnist
import tensorflow as tf
import tensorflow.keras as keras
from tensorflow.keras import backend as K
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, Dense
from tensorflow.keras.utils import to_categorical

import openfl.native as fx
from openfl.federated import FederatedModel, FederatedDataSet

After importing the required packages, the next step is setting up our openfl workspace. To do this, simply run the fx.init() command as follows:

In [ ]: #Setup default workspace, logging, etc.
fx.init('keras_cnn_mnist')

Now we are ready to define our dataset and model to perform federated learning on. The dataset should be composed of a numpy array. We start with a simple fully connected model that is trained on the MNIST dataset.

In [ ]: #Import training and validation images/labels
train_images = mnist.train_images()
train_labels = to_categorical(mnist.train_labels())
valid_images = mnist.test_images()
valid_labels = to_categorical(mnist.test_labels())

def preprocess(images):
    #normalize
    images = (images / 255) - 0.5
    #Flatten
    images = images.reshape((-1, 784))
    return images

# Preprocess the images.
train_images = preprocess(train_images)
valid_images = preprocess(valid_images)

feature_shape = train_images.shape[1]
classes = 10

fl_data = FederatedDataSet(train_images, train_labels, valid_images, valid_labels, batch_size=32, num_classes=classes)

def build_model(feature_shape, classes):
    #Defines the MNIST model
    model = Sequential()
    model.add(Dense(64, input_shape=feature_shape, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(classes, activation='softmax'))

    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model

In [ ]: #Create a federated model using the build_model function and dataset
fl_model = FederatedModel(build_model, data_loader=fl_data)
```

fx CLI

Production, Multi-node

```
Intel OpenFL - Secure Federated Learning at the Edge™

CORRECT USAGE

fx [options] [command] [subcommand] [args]

GLOBAL OPTIONS

-l, --log-level TEXT Logging verbosity level.
--help Show this message and exit.

AVAILABLE COMMANDS

tutorial      Manage Jupyter notebooks.

  > start      Start the Jupyter notebook from the tutorials...

plan         Manage Federated Learning Plans.

  > freeze     Finalize the Data Science plan.
  > initialize Initialize Data Science plan.
  > print      Print the current plan.
  > remove     Remove this plan.
  > save       Save the current plan to this plan and...
  > switch     Switch the current plan to this plan.

workspace    Manage Federated Learning Workspaces.

  > certify    Create certificate authority for federation.
  > create     Create the workspace.
  > dockerize  Pack FL.Edge and the workspace as a Docker...
  > export     Export federated learning workspace.
  > import     Import federated learning workspace.

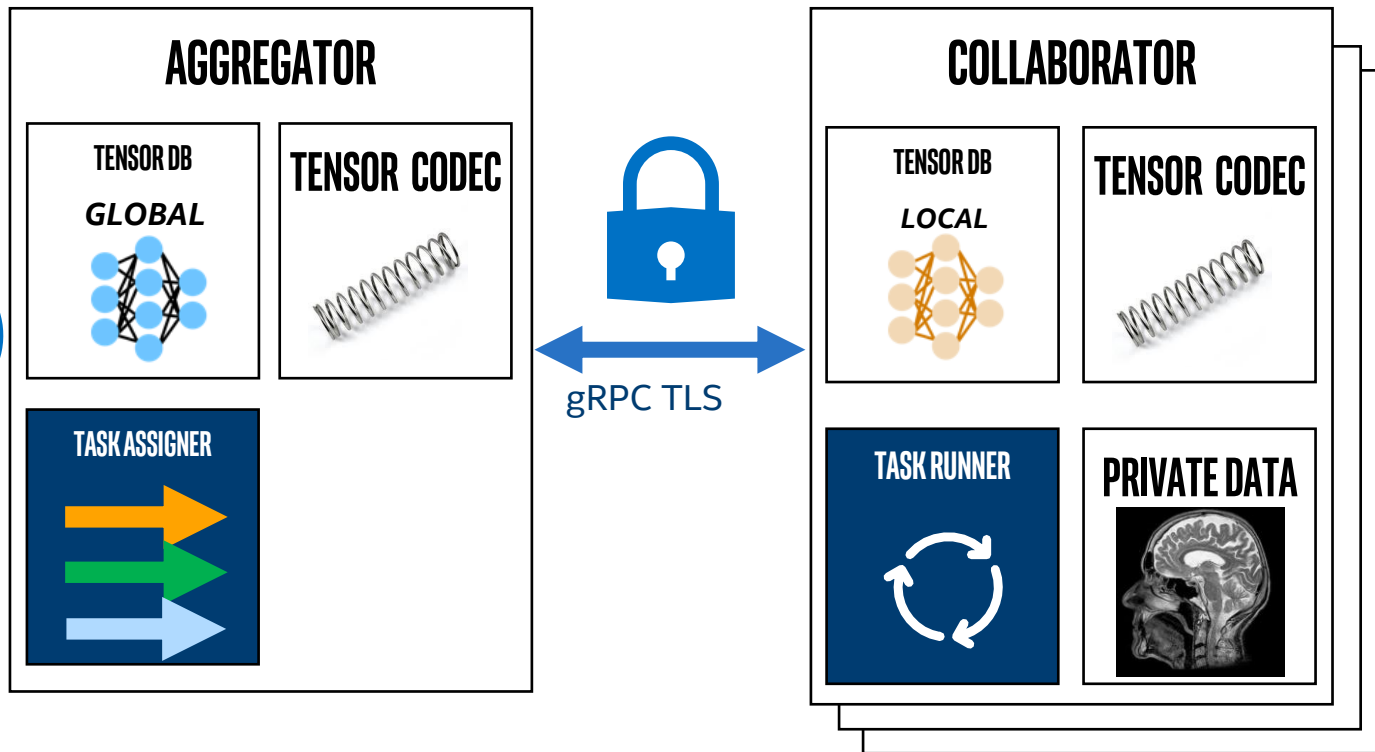
collaborator Manage Federated Learning Collaborators.

  > certify    Certify the collaborator.
  > generate-cert-request Generate certificate request for the...
  > start      Start a collaborator service.

aggregator   Manage Federated Learning Aggregator.
```

50 Foot View of OpenFL Architecture

Use the federation for other tasks?



FL Plan

```
2 aggregator :
3   defaults : plan/defaults/aggregator.yaml
4   template : openfl.component.Aggregator
5   settings :
6     init_state_path : save/keras_cnn_mnist_init.pbuf
7     best_state_path : save/keras_cnn_mnist_best.pbuf
8     last_state_path : save/keras_cnn_mnist_last.pbuf
9     rounds_to_train : 10
10 collaborator :
11   defaults : plan/defaults/collaborator.yaml
12   template : openfl.component.Collaborator
13   settings :
14     delta_updates : false
15     opt_treatment : RESET
16 data_loader :
17   defaults : plan/defaults/data_loader.yaml
18   template : code.tfmnist_inmemory.TensorFlowMnistInMemory
19   settings :
20     collaborator_count : 2
21     data_group_name : mnist
22     batch_size : 256
23 task_runner :
24   defaults : plan/defaults/task_runner.yaml
25   template : code.keras_cnn.KerasCNN
26 network :
27   defaults : plan/defaults/network.yaml
28 assigner :
29   defaults : plan/defaults/assigner.yaml
30   template : openfl.component.RandomGroupedAssigner
31   settings :
32     task_groups :
33       - name : validation_only
34         percentage : 0.5
35         tasks :
36           - aggregated_model_validation
37       - name : train_and_validate
38         percentage : 0.5
39         tasks :
40           - aggregated_model_validation
41           - train
42           - locally_tuned_model_validation
43 tasks :
44   defaults : plan/defaults/tasks_keras.yaml
```



Aggregator Plan



Collaborator Plan



Data Loader



Task Runner

OpenFL Python API

Federated Keras MNIST Tutorial

```
In [ ]: #Install dependencies if not already installed
!pip install tensorflow mnist
```

```
In [ ]: import numpy as np
import mnist
import tensorflow as tf
import tensorflow.keras as keras
from tensorflow.keras import backend as K
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, Dense
from tensorflow.keras.utils import to_categorical

import openfl.native as fx
from openfl.federated import FederatedModel, FederatedDataSet
```

After importing the required packages, the next step is setting up our openfl workspace. To do this, simply run the `fx.init()` command as follows:

```
In [ ]: #Setup default workspace, Logging, etc.
fx.init('keras_cnn_mnist')
```

Now we are ready to define our dataset and model to perform federated learning on. The dataset should be composed of a numpy array. We start with a simple fully connected model that is trained on the MNIST dataset.

```
In [ ]: #Import training and validation images/Labels
train_images = mnist.train_images()
train_labels = to_categorical(mnist.train_labels())
valid_images = mnist.test_images()
valid_labels = to_categorical(mnist.test_labels())

def preprocess(images):
    #Normalize
    images = (images / 255) - 0.5
    #Flatten
    images = images.reshape((-1, 784))
    return images

# Preprocess the images.
train_images = preprocess(train_images)
valid_images = preprocess(valid_images)

feature_shape = train_images.shape[1]
classes = 10

fl_data = FederatedDataSet(train_images, train_labels, valid_images, valid_labels, batch_size=32, num_classes=classes)

def build_model(feature_shape, classes):
    #Defines the MNIST model
    model = Sequential()
    model.add(Dense(64, input_shape=feature_shape, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(classes, activation='softmax'))

    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'],)
    return model
```

```
In [ ]: #Create a federated model using the build model function and dataset
fl_model = FederatedModel(build_model, data_loader=fl_data)
```

OpenFL Python API

```
In [ ]: #Create a federated model using the build_model function and dataset
final_model = FederatedModel(build_model,data_loader=f1_data)
```

The `FederatedModel` object is a wrapper around your Keras, Tensorflow or PyTorch model that makes it compatible with openfl. It provides built in federated training and validation functions that we will see used below. Using its `setup` function, collaborator models and datasets can be automatically defined for the experiment.

```
In [ ]: collaborator_models = f1_model.setup(num_collaborators=2)
collaborators = {'one':collaborator_models[0], 'two':collaborator_models[1]#, 'three':collaborator_models[2]}
```

```
In [ ]: #Original MNIST dataset
print(f'Original training data size: {len(train_images)}')
print(f'Original validation data size: {len(valid_images)}\n')

#Collaborator one's data
print(f'Collaborator one\'s training data size: {len(collaborator_models[0].data_loader.X_train)}')
print(f'Collaborator one\'s validation data size: {len(collaborator_models[0].data_loader.X_valid)}\n')

#Collaborator two's data
print(f'Collaborator two\'s training data size: {len(collaborator_models[1].data_loader.X_train)}')
print(f'Collaborator two\'s validation data size: {len(collaborator_models[1].data_loader.X_valid)}\n')

#Collaborator three's data
#print(f'Collaborator three\'s training data size: {len(collaborator_models[2].data_loader.X_train)}')
#print(f'Collaborator three\'s validation data size: {len(collaborator_models[2].data_loader.X_valid)}')
```

We can see the current plan values by running the `fx.get_plan()` function

```
In [ ]: #Get the current values of the plan. Each of these can be overridden
import json
print(json.dumps(fx.get_plan(), indent=4, sort_keys=True))
```

Now we are ready to run our experiment. If we want to pass in custom plan settings, we can easily do that with the `override_config` parameter

```
In [ ]: #Run experiment, return trained FederatedModel
final_f1_model = fx.run_experiment(collaborators,override_config={'aggregator.settings.rounds_to_train':5})
```

```
In [ ]: #Save final model
final_f1_model.save_native('final_model')
```

OpenFL Flow

1. Setup PKIs (recommend commercial CA)
2. Define TF/PyTorch/Other model
3. Define data loader
4. Define FL Plan
5. Create OpenFL workspace
6. Distribute OpenFL workspace
7. Start OpenFL task runner

OpenFL fx CLI

```
Usage: fx workspace create [OPTIONS]
```

```
Try 'fx workspace create --help' for help.
```

```
Error: Missing option '--template'. Choose from:
```

```
keras_cnn_mnist,  
tf_2dunet,  
tf_cnn_histology,  
torch_cnn_mnist.
```

```
(fledge_env) tony@b4969173b450:~$ fx workspace create --prefix ~/work1 --template keras_cnn_mnist
```

Fully-functional demos as templates:

- Keras CNN for MNIST
- TensorFlow 2D U-Net
- TensorFlow CNN for Histology
- PyTorch CNN for MNIST

```
New workspace directory structure:
```

```
work1  
├── cert  
│   └── config  
│       ├── signing-ca.conf  
│       ├── root-ca.conf  
│       ├── server.conf  
│       └── client.conf  
├── save  
├── logs  
├── data  
├── requirements.txt  
├── plan  
│   └── defaults  
│       ├── collaborator.yaml  
│       ├── assigner.yaml  
│       ├── data_loader.yaml  
│       ├── task_runner.yaml  
│       ├── tasks_torch.yaml  
│       ├── network.yaml  
│       ├── tasks_tensorflow.yaml  
│       ├── tasks_keras.yaml  
│       └── aggregator.yaml  
├── data.yaml  
├── cols.yaml  
├── plan.yaml  
└── code  
    ├── tfmnist_inmemory.py  
    ├── mnist_utils.py  
    ├── keras_cnn.py  
    └── __init__.py
```

```
8 directories, 21 files
```

```
✓OK
```

OpenFL fx CLI

```
task_runner:
  defaults: plan/defaults/task_runner.yaml
  settings: {}
  template: code.keras_cnn.KerasCNN
tasks:
  aggregated_model_validation:
    function: validate
    kwargs:
      apply: global
      batch_size: 32
      metrics:
        - acc
  defaults: plan/defaults/tasks_keras.yaml
  locally_tuned_model_validation:
    function: validate
    kwargs:
      apply: local
      batch_size: 32
      metrics:
        - acc
  settings: {}
  train:
    function: train
    kwargs:
      batch_size: 32
      epochs: 1
      metrics:
        - loss
```

```
INFO      • Starting the Aggregator Service. aggregator.py:28
INFO      Building • Object RandomGroupedAssigner from fledge.component Module. plan.py:149
INFO      Settings • {'task_groups': [{'name': 'train_and_validate', 'percentage': 1.0, 'tasks': ['aggregated_model_validation', 'train', 'locally_tuned_model_validation']}], 'authorized_cols': ['alpha123', 'bravo456'], 'rounds_to_train': 10} plan.py:150
INFO      Override • {'defaults': 'plan/defaults/assigner.yaml'} plan.py:151
INFO      Building • Object Aggregator from fledge.component Module. plan.py:149
INFO      Settings • {'best_state_path': 'save/keras_cnn_mnist_best.pbuf', 'init_state_path': 'save/keras_cnn_mnist_init.pbuf', 'last_state_path': 'save/keras_cnn_mnist_last.pbuf', 'rounds_to_train': 10, 'aggregator_uuid': 'aggregator_plan.yaml_33f2ee1e', 'federation_uuid': 'plan.yaml_33f2ee1e', 'authorized_cols': ['alpha123', 'bravo456'], 'assigner': <fledge.component.assigner.random_grouped_assigner.RandomGroupedAssigner object at 0x7f8c20d072b0>} plan.py:150
INFO      Override • {'defaults': 'plan/defaults/aggregator.yaml'} plan.py:151
INFO      Starting Aggregator gRPC Server server.py:145
```


OpenFL fx CLI

```
for aggregated_model_validation, round 2
agggregator.py:366
[21:28:40] INFO Collaborator bravo456 is sending task results
for train, round 2
agggregator.py:366
INFO Collaborator alpha123 is sending task results
for train, round 2
agggregator.py:366
[21:28:41] INFO Collaborator bravo456 is sending task results
for locally_tuned_model_validation, round 2
agggregator.py:366
INFO Collaborator alpha123 is sending task results
for locally_tuned_model_validation, round 2
agggregator.py:366
INFO train task metrics...

agggregator.py:502
INFO loss: 0.1192

agggregator.py:531
INFO aggregated_model_validation task metrics...

agggregator.py:502
INFO acc: 0.9600

agggregator.py:531
INFO Saved the best model with score 0.959996

agggregator.py:536
INFO locally_tuned_model_validation task metrics...

agggregator.py:502
INFO acc: 0.9695

agggregator.py:531
INFO Saving round 3 model...

agggregator.py:592
INFO Starting round 3...

agggregator.py:600
rou

nd 3
[21:28:40] INFO Collaborator bravo456 is sending task results
for train, round 2
agggregator.py:366
INFO Collaborator alpha123 is sending task results
for train, round 2
agggregator.py:366
[21:28:41] INFO Collaborator bravo456 is sending task results
for locally_tuned_model_validation, round 2
agggregator.py:366
INFO Collaborator alpha123 is sending task results
for locally_tuned_model_validation, round 2
agggregator.py:366
INFO train task metrics...

agggregator.py:502
INFO loss: 0.1192

agggregator.py:531
INFO aggregated_model_validation task metrics...

agggregator.py:502
INFO acc: 0.9600

agggregator.py:531
INFO Saved the best model with score 0.959996

agggregator.py:536
INFO locally_tuned_model_validation task metrics...

agggregator.py:502
INFO acc: 0.9695

agggregator.py:531
INFO Saving round 3 model...

agggregator.py:592
INFO Starting round 3...

agggregator.py:600
collaborator.py:163
[21:28:08] INFO Waiting for tasks...

collaborator.py:163
[21:28:18] INFO Waiting for tasks...

collaborator.py:163
[21:28:28] INFO Waiting for tasks...

collaborator.py:163
INFO Received the following tasks: ['aggregated_model_validation', 'train', 'locally_tuned_model_validation']
collaborator.py:136
[21:28:29] INFO Sending metric for task aggregated_model_validation, round number 1: acc 0.9204000234603882
collaborator.py:319
[21:28:33] INFO Sending metric for task train, round number 1: loss 0.13767806259791057
collaborator.py:319
[21:28:34] INFO Sending metric for task locally_tuned_model_validation, round number 1: acc 0.9430000185966492
collaborator.py:319
[21:28:35] INFO Waiting for tasks...

collaborator.py:163
INFO Received the following tasks: ['aggregated_model_validation', 'train', 'locally_tuned_model_validation']
collaborator.py:136
[21:28:36] INFO Sending metric for task aggregated_model_validation, round number 2: acc 0.96119998555908
collaborator.py:319
[21:28:40] INFO Sending metric for task train, round number 2: loss 0.0793122284034888
collaborator.py:319
[21:28:41] INFO Sending metric for task locally_tuned_model_validation, round number 2: acc 0.9652000069618225
collaborator.py:319
INFO Waiting for tasks...

collaborator.py:163
2.0, as updates are applied automatically.

collaborator.py:163
[21:28:16] INFO Sending metric for task aggregated_model_validation, round number 0: acc 0.10142028331756592
collaborator.py:319
[21:28:19] INFO Sending metric for task train, round number 0: loss 0.6948366242356688
collaborator.py:319
[21:28:20] INFO Sending metric for task locally_tuned_model_validation, round number 0: acc 0.9183836579322815
collaborator.py:319
INFO Waiting for tasks...

collaborator.py:163
INFO Received the following tasks: ['aggregated_model_validation', 'train', 'locally_tuned_model_validation']
collaborator.py:136
[21:28:21] INFO Sending metric for task aggregated_model_validation, round number 1: acc 0.915783166885376
collaborator.py:319
[21:28:25] INFO Sending metric for task train, round number 1: loss 0.24818640858776397
collaborator.py:319
[21:28:26] INFO Sending metric for task locally_tuned_model_validation, round number 1: acc 0.9519904255867004
collaborator.py:319
INFO Waiting for tasks...

collaborator.py:163
[21:28:36] INFO Waiting for tasks...

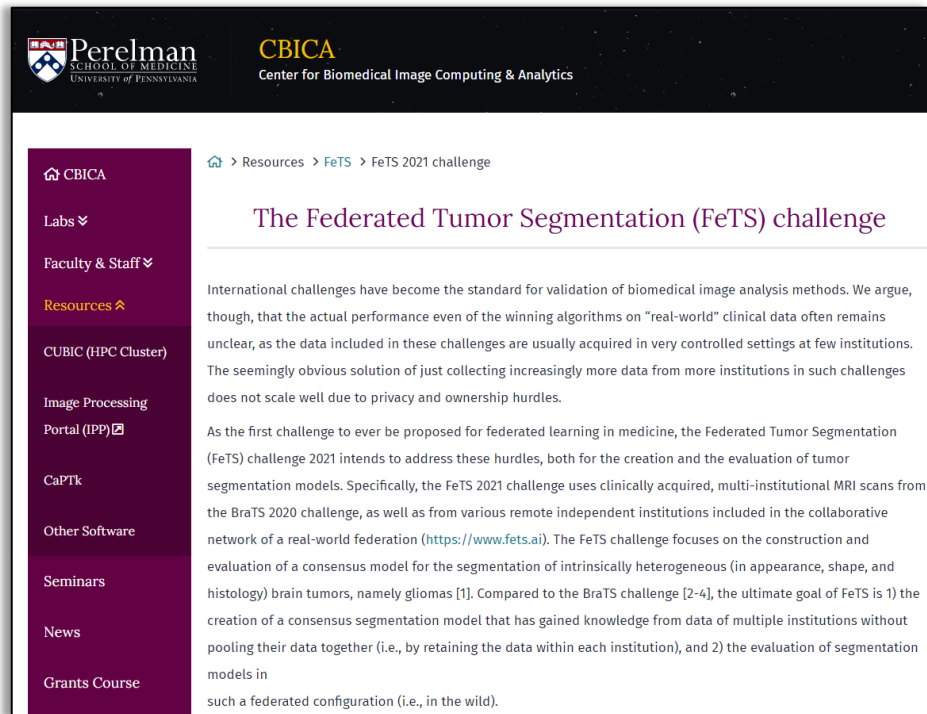
collaborator.py:163
INFO Received the following tasks: ['aggregated_model_validation', 'train', 'locally_tuned_model_validation']
collaborator.py:136
INFO Sending metric for task aggregated_model_validation, round number 2: acc 0.9587917327880859
collaborator.py:319
[21:28:40] INFO Sending metric for task train, round number 2: loss 0.15912549474000207
collaborator.py:319
[21:28:40] INFO Sending metric for task locally_tuned_model_validation, round number 2: acc 0.9652000069618225
collaborator.py:319
INFO Waiting for tasks...
```

Aggregator

Collaborator A

Collaborator B

Federated Tumor Segmentation (FeTS) Challenge



The screenshot shows the website for the Federated Tumor Segmentation (FeTS) challenge. At the top, there are logos for Perelman School of Medicine and CBICA (Center for Biomedical Image Computing & Analytics). The main heading is "The Federated Tumor Segmentation (FeTS) challenge". The text describes the challenge as a standard for validation of biomedical image analysis methods, noting that performance on "real-world" clinical data often remains unclear. It mentions that the challenge aims to address privacy and ownership hurdles by using clinically acquired, multi-institutional MRI scans from the BraTS 2020 challenge and other remote independent institutions. The ultimate goal is the creation of a consensus segmentation model that has gained knowledge from data of multiple institutions without pooling their data together, and the evaluation of segmentation models in a federated configuration.

- Task 1 ("Federated Training")
- Task 2 ("Federated Evaluation")

Starts May 2021

<https://www.med.upenn.edu/cbica/fets/miccai2021>

Quiz Time