DOCUMENTATION FOR SECURE MULTI PARTY COMPUTATION

## **Introduction**

There are 2 types of architecture implemented - Server Model and Non-Servermodel.

Server Model Architecture:

* Consists of separate components: Image Provider (client) , Model Provider (client), and Server Code.
* Image Provider: Provides images to the server for processing.
* Model Provider: Provides trained models (simple neural networks and CNNs) to the server for inference.
* Server Code: Receives images from the Image Provider, performs inference using the provided models from the Model Provider, and returns the results.

Non-Server Model Architecture:

* Does not have separate client and server components; everything runs within the same script.
* Both image and model provisioning and inferencing occur within the same script.
* Utilises the same trained models (simple neural networks and CNNs) for inference as in the Server Model Architecture.

Overall, the key difference lies in the separation of client and server in the Server Model Architecture, where image and model provisioning are handled separately from the server code, whereas in the Non-Server Model Architecture, all components run within the same script without this separation.

## 

## 

## **Introduction to CONFIG FILES**

This is a guide to distinguish among the different config files required for running different kinds of configurations like with split in NN, with vertical split in CNN , with vertical and horizontal split in CNN , with helpernode in NN and CNN.

(To see the split and helpernode meaning , check section I. )

Server Model Architecture

1. Split
   * Simple neural network (any number of layers possible)

smpc-split-config.json

model\_split\_config.json- copy it from example-model\_split\_config.json

* + CNN
    - Vertical Split only ( any number of layers possible )

smpc-cnn-config.json

model\_cnn\_config.json - copy it from ex-1l\_mnist-model\_cnn\_config or ex-2l\_mnist-model\_cnn\_config or model\_cnn\_config.json.

Depending on the user's requirement.

* + - Horizontal split and vertical split ( any number of layers possible )

smpc-cifar-config.json , smpc-mnist-config.json

model\_cnn\_config.json - copy it from ex-1l\_mnist-model\_cnn\_config(3 layers) or ex-2l\_mnist-model\_cnn\_config(4 layers) or model\_cnn\_config.json(6 layers)

Depending on the user's requirement.

1. Helpernode
   * Simple neural network ( any number of layers possible )

smpc-helpernode-config.json

model\_helpernode\_config.json - copy it from example-model\_helpernode\_config.json(5 layers) or example-model\_split\_config(2 layers)

* + CNN - MNIST and CIFAR ( any number of layers possible )

smpc-helpernode-config.json

model\_helpernode\_config.json -copy it from ex-1l\_mnist-model\_cnn\_config(3 layers) or ex-2l\_mnist-model\_cnn\_config(4 layers) or model\_cnn\_config.json(6 layers)

Note that helpernode using CNN , works with 3 layer , 4 layer and 6 layers.(Any number of layers).

Image\_config.json is used in all of the configurations.

1. ConstantModel
   * Simple neural network

smpc-constant-config.json

image\_config\_constmodel.json

model\_split\_config.json- - copy it from example-model\_split\_config.json

Non-Server Model Architecture (same workflow as server model)

* Split
  + Simple neural network (any number of layers possible)
  + CNN
    - Vertical Split only ( any number of layers possible )
    - Horizontal split and vertical split ( any number of layers possible )
* Helpernode
  + Simple neural network (any number of layers possible)
  + CNN - MNIST and CIFAR ( any number of layers possible )
* ConstantModel
  + Simple neural network

Same corresponding config files used in non server model.Only difference is that non server model does not use image\_config.json.Image index to be inferenced can be given from the smpc\_config.json

## 

## **III. Implementing Secure Multi-Party Computation (SMPC) on Server Model Architecture**

## ***To implement without docker and locally***

This setup outlines the steps for setting up and testing secure multi-party computation (MPC) scripts locally without Docker. This setup is primarily intended for testing purposes and allows developers to run MPC scripts on their local machines for debugging and validation.

#### **General modifications in config files which are applicable to all.**

* Navigate to the config\_files directory within the iudx-MOTION2NX repository.
* Open the relevant configuration files(see section II).
* Modify the IP address to "127.0.0.1" and set the "dns\_resolve" parameter to false. These changes ensure that the system operates locally and does not attempt DNS resolution.

#### 

##### **Updating the image\_config.json**

In image\_config.json update the following :

* Change the input dimensions - rows , columns and channels according to the image to be sent to the servers for inferencing.
* Change the cs0\_dns\_resolve from true to false.(other dns\_resolve also)

Eg:

|  |
| --- |
| {  "cs0\_host": "127.0.0.1",  "cs1\_host": "127.0.0.1",  "cs0\_dns\_resolve": false,  "cs1\_dns\_resolve": false,  "cs0\_port\_cs0\_output\_receiver": 4007,  "cs0\_port\_cs1\_output\_receiver": 4008,  "cs0\_port\_image\_receiver": 4014,  "cs1\_port\_image\_receiver": 4015,  "fractional\_bits": 13,  "image\_id": 270,  "image\_rows": 32,  "image\_cols": 32,  "channels": 3 } |

The above config file is an example for sending cifar data to the servers (run locally).

-Give the appropriate image\_id.(0-199 for mnist and 200-299 for cifar data.)

-Give correct dimensions for image\_rows , image\_cols , channels according to the image\_id.

Note:

[To do inferencing using the images given in iudx-MOTION2NX/data/ImageProvider/images , the images from X1.csv-X199.csv are MNIST images and X200.csv-X299.csv are CIFAR10 images.

**While running the server model architecture image dimensions are taken from image\_config.json**

**While running non-server model architecture image dimensions are taken from smpc-config.json. ]**

#### 

#### **Steps to run “*Split - Simple neural network*” (can run with any number of layers - NN only.)**

This guide outlines the steps to run the "Split - Simple Neural Network" locally without Docker, utilising specific configuration settings and scripts.

1. Go to base directory iudx-MOTION2NX/config\_files
2. Change the config files appropriately for the above configuration.See section II for getting the correct config\_file to change and model\_config also.

|  |
| --- |
| {  "cs0\_host": "127.0.0.1",  "cs1\_host": "127.0.0.1",  "reverse\_ssh\_host": "127.0.0.1",  "cs0\_dns\_resolve": false,  "cs1\_dns\_resolve": false,  "reverse\_ssh\_dns\_resolve": false,  "cs0\_port\_model\_receiver": 4005,  "cs1\_port\_model\_receiver": 4006,  "cs0\_port\_cs0\_output\_receiver": 4007,  "cs0\_port\_cs1\_output\_receiver": 4008,  "cs0\_port\_inference": 4009,  "cs1\_port\_inference":4010,  "relu0\_port\_inference": 4012,  "relu1\_port\_inference": 4013,  "cs0\_port\_image\_receiver": 4014,  "cs1\_port\_image\_receiver": 4015,  "number\_of\_layers": 2,  "fractional\_bits": 13,  "image\_id": 2,  "splits": 4,  "split\_layers\_genr": [{"rows":256, "splits":4}] } |

The above config file is an example for running NN(with mnist) locally.

- Change the number\_of\_layers and splits according to the model and required memory optimization.

- Update split\_layers\_genr also if the “splits” is updated.In case of no splits , change the “splits” to 1.

1. cd iudx-MOTION2NX/scripts/ServerModel\_Architecture/Split/
2. Open four separate terminal windows or tabs to run the necessary scripts concurrently.In four terminals , enter the following commands :

bash remote\_server0\_split.sh

bash remote\_server1\_split.sh

bash ImageProvider.sh

bash ModelProvider.sh

#### **Steps to run “*Convolutional Neural Network with options of vertical and horizontal splits*”.**

This guide outlines the steps to run Convolutional Neural Network (CNN) with options for vertical and horizontal splits locally without Docker, utilising specific configuration settings and scripts.

1. Go to base directory iudx-MOTION2NX/config\_files
2. Change the config files appropriately for the above configuration.See section II for getting the correct config\_file to change and model\_config also.

|  |
| --- |
| {  "cs0\_host": "127.0.0.1",  "cs1\_host": "127.0.0.1",  "reverse\_ssh\_host": "127.0.0.1",  "cs0\_dns\_resolve": false,  "cs1\_dns\_resolve": false,  "reverse\_ssh\_dns\_resolve": false,  "cs0\_port\_model\_receiver": 4005,  "cs1\_port\_model\_receiver": 4006,  "cs0\_port\_cs0\_output\_receiver": 4007,  "cs0\_port\_cs1\_output\_receiver": 4008,  "cs0\_port\_inference": 4009,  "cs1\_port\_inference":4010,  "relu0\_port\_inference": 4012,  "relu1\_port\_inference": 4013,  "cs0\_port\_image\_receiver": 4014,  "cs1\_port\_image\_receiver": 4015,  "fractional\_bits": 13,  "image\_id": 1,  "image\_rows": 28,  "image\_cols": 28,  "channels": 1,  "split\_layers\_genr": [{"layer\_id":1, "kernels":5, "splits":5, "kernel\_rows":5, "kernel\_cols":5, "pads":[1,1,0,0], "strides": [2,2], "row\_split": 1},  {"layer\_id":2, "kernels":3, "splits":3, "kernel\_rows":4, "kernel\_cols":4, "pads":[1,1,0,0], "strides": [2,2], "row\_split": 1},  {"layer\_id":3, "rows":100, "splits":1},  {"layer\_id":4, "rows":10, "splits":1}]   } |

The above config file is an example for running cnn on mnist data locally.

- Update split\_layers\_genr according to the model and required memory optimisation.

* *“splits”* - is the number of vertical splits. Its maximum value is the number of kernels in case of the CNN layer. In the case of the NN layer(layer id 3 and 4) , “splits” is the number of splits done in rows of a model.Its maximum value is the number of rows of model in that layer.
* *“row\_split” -* is the number of horizontal splits of an image. Its maximum is output\_rows (after CNN) of each layer.

1. cd iudx-MOTION2NX/scripts/ServerModel\_Architecture/Split/CNN
2. In ModelProvider\_CNN.sh , change the existing smpc\_config\_path in script

To run cnn with cifar data, change the

smpc\_config\_path=${BASE\_DIR}/config\_files/**cifar**-smpc-cnn-config.json

To run cnn with mnist data, change the

smpc\_config\_path=${BASE\_DIR}/config\_files/**mnist**-smpc-cnn-config.json

1. Open four separate terminal windows or tabs to run the necessary scripts concurrently.In four terminals , enter the following commands :

bash Hsplit\_remote\_server0\_cifar.sh/Hsplit\_remote\_server0\_mnist.sh

bash Hsplit\_remote\_server1\_cifar.sh/Hsplit\_remote\_server1\_mnist.sh

bash ImageProvider.sh

bash ModelProvider\_CNN.sh

[ Note: Here Hsplit\_remote\_server0\_cifar.sh takes model parameters from **cifar**-smpc-cnn-config.json and Hsplit\_remote\_server0\_mnist.sh takes model parameters from **mnist**-smpc-cnn-config.json ]

#### 

#### **Steps to run “*Convolutional Neural Network with options of vertical splits only.*”**

This guide outlines the steps to run Convolutional Neural Network (CNN) with vertical splits only , locally without Docker, utilising specific configuration settings and scripts.

1. Go to base directory iudx-MOTION2NX/config\_files
2. Change the config files appropriately for the above configuration.See section II for getting the correct config\_file to change and model\_config file also.

|  |
| --- |
| {  "cs0\_host": "127.0.0.1",  "cs1\_host": "127.0.0.1",  "reverse\_ssh\_host": "127.0.0.1",  "cs0\_dns\_resolve": false,  "cs1\_dns\_resolve": false,  "reverse\_ssh\_dns\_resolve": false,  "cs0\_port\_model\_receiver": 4005,  "cs1\_port\_model\_receiver": 4006,  "cs0\_port\_cs0\_output\_receiver": 4007,  "cs0\_port\_cs1\_output\_receiver": 4008,  "cs0\_port\_inference": 4009,  "cs1\_port\_inference":4010,  "relu0\_port\_inference": 4012,  "relu1\_port\_inference": 4013,  "cs0\_port\_image\_receiver": 4014,  "cs1\_port\_image\_receiver": 4015,  "fractional\_bits": 13,  "image\_id": 1,  "image\_rows": 28,  "image\_cols": 28,  "channels": 1,  "split\_layers\_genr": [{"layer\_id":1, "kernels":5, "splits":5},  {"layer\_id":2, "kernels":3, "splits"3},  {"layer\_id":3, "rows":108, "splits":1},  {"layer\_id":4, "rows":10, "splits":2}] } |

The above config file is an example for running cnn on mnist data locally.This config file is for doing memory optimizations according to the vertical splits of kernel in CNN layer.

To change for cifar data add more layers according to the model of cifar.(refer section II for model\_configuration)

-Update split\_layers\_genr according to the model and required memory optimisation.

* *“splits”* - is the number of vertical splits. Its maximum value is the number of kernels in case of CNN layer. In case of the NN layer(layer id 3 and 4) , “splits” is the number of splits done in rows of a model.Its maximum value is the number of rows of model in that layer.

1. cd iudx-MOTION2NX/scripts/ServerModel\_Architecture/Split/CNN
2. In ModelProvider\_CNN.sh , change the existing smpc\_config\_path in script to

smpc\_config\_path=${BASE\_DIR}/config\_files/smpc-cnn-config.json

1. Open four separate terminal windows or tabs to run the necessary scripts concurrently.In four terminals , enter the following commands :

bash remote\_server0\_split.sh

bash remote\_server1\_split.sh

bash ImageProvider.sh

bash ModelProvider\_CNN.sh

#### **Steps to run “*Simple Neural Network using helpernode.*” (Any number of layers.)**

This guide outlines the steps to run a simple neural network using helpernode , locally without Docker, utilising specific configuration settings and scripts.

1. Go to base directory iudx-MOTION2NX/config\_files
2. Change the config files appropriately for the above configuration.See section II for getting the correct config\_file to change and model\_config also.

|  |
| --- |
| {  "cs0\_host": "127.0.0.1",  "cs1\_host": "127.0.0.1",  "helpernode\_host": "127.0.0.1",  "reverse\_ssh\_host": "127.0.0.1",  "cs0\_dns\_resolve": false,  "cs1\_dns\_resolve": false,  "helpernode\_dns\_resolve": false,  "reverse\_ssh\_dns\_resolve": false,  "cs0\_port\_model\_receiver": 4005,  "cs1\_port\_model\_receiver": 4006,  "cs0\_port\_cs0\_output\_receiver": 4007,  "cs0\_port\_cs1\_output\_receiver": 4008,  "cs0\_port\_inference": 4009,  "cs1\_port\_inference":4010,  "helpernode\_port\_inference":4011,  "relu0\_port\_inference": 4012,  "relu1\_port\_inference": 4013,  "cs0\_port\_image\_receiver": 4014,  "cs1\_port\_image\_receiver": 4015,  "number\_of\_layers": 5,  "fractional\_bits": 13,  "image\_id": 1,  "layer\_types": [1,1,0,0] } |

The above config file is an example for running Neural Network on mnist data locally using helpernode.

* The “number\_of\_layers” is 2 or 5 , for the given model for doing inference on nn(mnist data).
* No need to change “layer\_types”.

1. cd iudx-MOTION2NX/scripts/ServerModel\_Architecture/HelperNode
2. Open five separate terminal windows or tabs to run the necessary scripts concurrently.In five terminals , enter the following commands :

bash helpernode\_server0.sh

bash helpernode\_server1.sh

bash helpernode\_server2.sh

bash ImageProvider.sh

bash ModelProvider.sh

#### **Steps to run “*CNN + NN using helper node.*” (Any number of layers.)**

This guide outlines the steps to run CNN using helper node , locally without Docker, utilising specific configuration settings and scripts.

1. Go to base directory iudx-MOTION2NX/config\_files
2. Change the config files appropriately for the above configuration.See section II for getting the correct config\_file to change and model\_config also.

|  |
| --- |
| {  "cs0\_host": "127.0.0.1",  "cs1\_host": "127.0.0.1",  "helpernode\_host": "127.0.0.1",  "reverse\_ssh\_host": "127.0.0.1",  "cs0\_dns\_resolve": false,  "cs1\_dns\_resolve": false,  "helpernode\_dns\_resolve": false,  "reverse\_ssh\_dns\_resolve": false,  "cs0\_port\_model\_receiver": 4005,  "cs1\_port\_model\_receiver": 4006,  "cs0\_port\_cs0\_output\_receiver": 4007,  "cs0\_port\_cs1\_output\_receiver": 4008,  "cs0\_port\_inference": 4009,  "cs1\_port\_inference":4010,  "helpernode\_port\_inference":4011,  "relu0\_port\_inference": 4012,  "relu1\_port\_inference": 4013,  "cs0\_port\_image\_receiver": 4014,  "cs1\_port\_image\_receiver": 4015,  "number\_of\_layers": 4,  "fractional\_bits": 13,  "image\_id": 250,  "layer\_types": [1,1,0,0] } |

The above config file is an example for running CNN and NN on mnist/cifar data locally using helpernode.

* The “number\_of\_layers” is according to the number of layers in the given model for doing inference on nn(mnist data).
* In “layer\_types , 1 denotes CNN layer and 0 denotes NN layer.

Example , if first 2 layers are CNN and next 2 are NN , then the layer\_type is [1,1,0,0]

If first 4 layers are CNN and next 2 are NN , then the layer\_types is [1,1,1,1,0,0]

1. cd iudx-MOTION2NX/scripts/ServerModel\_Architecture/HelperNode/CNN
2. Open five separate terminal windows or tabs to run the necessary scripts concurrently.In five terminals , enter the following commands :

bash helpernode\_server0\_cnn.sh

bash helpernode\_server1\_cnn.sh

bash helpernode\_server2\_cnn.sh

bash ImageProvider.sh

bash ModelProvider.sh

## 

## ***To implement with docker on cloud***

To run Secure Multi Party Computation code on cloud vm , we have deployed it using docker.Docker provides a lightweight, portable environment for running applications, while cloud VMs offer scalability, reliability, and easy management.

The setup is that server code is running on cloud VMs while the image provider and model provider are running locally, the configuration ensures efficient and flexible operation.

1. **Cloud-Based Server Code:**
   * The server code, responsible for handling computation tasks and coordinating communication within the secure multi-party computation (SMPC) network, runs on cloud VMs (Server 1 and Helper Node in AWS, Server 0 in Azure).
2. **Local Image Provider:**
   * The image provider, which supplies input images to the SMPC network, runs locally.
3. **Local Model Provider:**
   * Similarly, the model provider, responsible for delivering the trained neural network model for inference, also runs locally.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cloud VM name | Cloud | Public IP | Private IP | Spec | username |
| ap-south-1-test-smpc-server-1 | AWS | 13.200.145.195  (old 13.232.226.177) | 10.2.1.25 | 2vcpu, 4GiB, 30 GiB SSD | ubuntu |
| ap-south-1-test-smpc-server-helpernode | AWS | 15.206.80.191 | 10.2.1.171 | 1vcpu, 0.5GiB, 30 GiB SSD | ubuntu |
| smpc-server0 | Azure | 20.197.27.117 | 10.139.0.4 | 2vcpu, 4 GiB, 30 GiB SSD | ubuntu |

reverse ssh host ip : 52.172.238.250

These details provide the necessary information for accessing and managing each cloud VM instance. Ensure you have the appropriate credentials and access permissions to interact with these instances.

#### **General modifications in config files (local on system) which are applicable to all.**

* In the config files , change the existing ip address to the public ip address of the cloud vm according to the above table **locally on the system.**
* Change the cs0\_dns\_resolve from true to false.
* change the existing ip address of reverse\_ssh\_host to “52.172.238.250”.
* Change the ip addresses in image\_config.json
* **The above changes are applied to any config file (locally on the system) before sending shares to the cloud VM.**
* smpc-split-config.json
* smpc-cnn-config.json
* smpc-cifar-config.json
* smpc-mnist-config.json
* smpc-helpernode-config.json

Eg: In cifar-smpc-cnn-config.json

|  |
| --- |
| {  "cs0\_host": "20.197.27.117",  "cs1\_host": "13.200.145.195",  "reverse\_ssh\_host": "52.172.238.250",  "cs0\_dns\_resolve": false,  "cs1\_dns\_resolve": false,  "reverse\_ssh\_dns\_resolve": false,  "cs0\_port\_model\_receiver": 4005,  "cs1\_port\_model\_receiver": 4006,  "cs0\_port\_cs0\_output\_receiver": 4007,  "cs0\_port\_cs1\_output\_receiver": 4008,  "cs0\_port\_inference": 4009,  "cs1\_port\_inference":4010,  "relu0\_port\_inference": 4012,  "relu1\_port\_inference": 4013,  "cs0\_port\_image\_receiver": 4014,  "cs1\_port\_image\_receiver": 4015,  "fractional\_bits": 13,  "image\_id": 205,  "image\_rows": 32,  "image\_cols": 32,  "channels": 3,  "split\_layers\_genr": [{"layer\_id":1, "kernels":32, "splits":32, "kernel\_rows":3, "kernel\_cols":3, "pads":[1,1,1,1], "strides": [1,1], "row\_split": 1},  {"layer\_id":2, "kernels":32, "splits":32, "kernel\_rows":5, "kernel\_cols":5, "pads":[1,1,1,1], "strides": [2,2], "row\_split": 4},  {"layer\_id":3, "kernels":64, "splits":64, "kernel\_rows":5, "kernel\_cols":5, "pads":[2,2,2,2], "strides": [2,2], "row\_split": 1},  {"layer\_id":4, "kernels":64, "splits":64, "kernel\_rows":3, "kernel\_cols":3, "pads":[0,0,0,0], "strides": [1,1], "row\_split": 1},  {"layer\_id":5, "rows":512, "splits":256},  {"layer\_id":6, "rows":10, "splits":1}] } |

##### **Updating the image\_config.json**

In image\_config.json update the following :

* Update the existing ip addresses to public ip address of the 2 servers.
* Change the input dimensions - rows , columns and channels according to the image to be sent to the servers for inferencing.
* Change the cs0\_dns\_resolve from true to false.(other dns\_resolve also)

Eg:

|  |
| --- |
| {  "cs0\_host": "20.197.27.117",  "cs1\_host": "13.200.145.195",  "cs0\_dns\_resolve": false,  "cs1\_dns\_resolve": false,  "cs0\_port\_cs0\_output\_receiver": 4007,  "cs0\_port\_cs1\_output\_receiver": 4008,  "cs0\_port\_image\_receiver": 4014,  "cs1\_port\_image\_receiver": 4015,  "fractional\_bits": 13,  "image\_id": 270,  "image\_rows": 32,  "image\_cols": 32,  "channels": 3 } |

The above config file is an example for sending cifar data to the servers (run locally).

-Give the appropriate image\_id.(0-199 for mnist and 200-299 for cifar data.)

-Give correct dimensions for image\_rows , image\_cols , channels according to the image\_id.

Note:

[To do inferencing using the images given in iudx-MOTION2NX/data/ImageProvider/images , the images from X1.csv-X199.csv are MNIST images and X200.csv-X299.csv are CIFAR10 images.

**While running the server model architecture image dimensions are taken from image\_config.json , therefore need to update image\_config.json before sending image shares.**]

##### **Update the model\_config files**

Update the model\_config files according to user requirements. Refer section II for copying correct model details for implementing inference using different configurations

**Getting access to servers on cloud VM**

Generate ssh public private key

* 1. ssh-keygen -b 4096 -f ~/.ssh/iudx\_cloud
  2. passphrase refers to encrypting a private key using a passphrase based key. For convenience, can be ignored and press enter, i.e. without passphrase and hence key would be unencrypted
  3. send the public key to add access to VMs, the file with ‘.pub’ extension, i.e the file at ~/.ssh/iudx\_cloud.pub
  4. **DO NOT SEND the private key**

Once the keys are added to the servers , the user can access the servers on cloud using the below commands:

ssh ubuntu@20.197.27.117

ssh ubuntu@13.200.145.195

ssh ubuntu@15.206.80.191

#### **Reverse tunnelling setup**

{description on reverse tunnelling}

1. First, check if possible to ssh using below command, u will be see some stats of server and will exit. this is expected as u dont have shell access for autossh machine

|  |
| --- |
| ssh -i ~/.ssh/iudx\_cloud sshtunuser@analytics.iudx.org.in |

1. Open one terminal and reverse tunnel to receive image output shares from server 0

|  |
| --- |
| autossh -M 23323 -N -q -o "ExitOnForwardFailure yes" -o "ServerAliveInterval 10" -o "ServerAliveCountMax 3" -R 4007:localhost:4007 sshtunuser@analytics.iudx.org.in -i ~/.ssh/iudx\_cloud |

1. Open another terminal and reverse tunnel to receive image output shares from server 1

|  |
| --- |
| autossh -M 23325 -N -q -o "ExitOnForwardFailure yes" -o "ServerAliveInterval 10" -o "ServerAliveCountMax 3" -R 4008:localhost:4008 sshtunuser@analytics.iudx.org.in -i ~/.ssh/iudx\_cloud |

#### 

#### **Ip addresses in cloud**

The ip addresses are in the following format :

In server0 , the ip address of cs0\_host is "10.139.0.4" (its private ip) and the ip address of cs1\_host is "13.200.145.195" (public ip of server 1).

Similar cases are for server 1 and server 2 (helpernode).

The ip addresses are set up correctly in all the config\_files. The changes the user can do is the number of layers and splits according to the requirements. User doesnt have to change the ip addresses in cloud.

#### **Steps to run “*Split - Simple neural network*” (can run with any number of layers - NN only.)**

This guide outlines the steps to run the "Split - Simple Neural Network" on cloud VM using docker.

1. Two servers(server0 and server1) are needed to run this setup. After getting access to the clouds , cd iudx-MOTION2NX/config\_files. To get correct configuration file to change , refer to section II.

|  |
| --- |
| {  "cs0\_host": "10.139.0.4",  "cs1\_host": "13.232.226.177",  "reverse\_ssh\_host": "52.172.238.250",  "cs0\_dns\_resolve": false,  "cs1\_dns\_resolve": false,  "reverse\_ssh\_dns\_resolve": false,  "cs0\_port\_model\_receiver": 4005,  "cs1\_port\_model\_receiver": 4006,  "cs0\_port\_cs0\_output\_receiver": 4007,  "cs0\_port\_cs1\_output\_receiver": 4008,  "cs0\_port\_inference": 4009,  "cs1\_port\_inference":4010,  "relu0\_port\_inference": 4012,  "relu1\_port\_inference": 4013,  "cs0\_port\_image\_receiver": 4014,  "cs1\_port\_image\_receiver": 4015,  "number\_of\_layers": 2,  "fractional\_bits": 13,  "image\_id": 2,  "splits": 8,  "split\_layers\_genr": [{"rows":256, "splits":64}]  } |

The above config file is an example for running NN(with mnist input data).

- Change the number\_of\_layers and splits according to the model and required memory optimization.

- Update split\_layers\_genr also if the “splits” is updated.In case of no splits , change the “splits” to 1.

**If any of the server config file is changed , then the other one should also be changed (same changes) in other server.**

1. cd iudx-MOTION2NX/scripts/ServerModel\_Architecture/Split
2. Enter the below cmd lines to execute scripts in docker :

Run the below commands in the 2 servers simultaneously.

For server0:

The below command starts the Docker container for Secure Multi-Party Computation (SMPC) on Server0 using the specified Docker Compose files. The -d flag runs the container in detached mode, allowing it to run in the background.

|  |
| --- |
| docker compose -f docker-compose.remote\_split\_servermodel.yaml  -f ../../../docker-compose.remote-registry.yaml up -d smpc-server0 |

For server1:

|  |
| --- |
| docker compose -f docker-compose.remote\_split\_servermodel.yaml  -f ../../../docker-compose.remote-registry.yaml up -d smpc-server1 |

The below commands are common to both servers.

This command displays the logs of the running Docker containers. The -f flag follows the output in real-time, allowing you to continuously monitor the logs.

|  |
| --- |
| docker compose -f docker-compose.remote\_split\_servermodel.yaml  -f ../../../docker-compose.remote-registry.yaml logs -f |

This command stops and removes the Docker containers after the execution is completed. It performs a cleanup operation to ensure resources are released appropriately.

|  |
| --- |
| docker compose -f docker-compose.remote\_split\_servermodel.yaml  -f ../../../docker-compose.remote-registry.yaml down |

1. The image provider and model provider send image and mode shares from their respective systems(locally).

cd iudx-MOTION2NX/scripts/ServerModel\_Architecture/Split/

bash ImageProvider.sh

bash ModelProvider.sh

#### **Steps to run “*Convolutional Neural Network with options of vertical and horizontal splits*”.**

This guide outlines the steps to run Convolutional Neural Network (CNN) with options for vertical and horizontal splits on cloud VMs using docker.

1. Two servers(server0 and server1) are needed to run this setup. After getting access to the clouds , cd iudx-MOTION2NX/config\_files. To get the correct configuration file to change , refer to section II.

|  |
| --- |
| {  "cs0\_host": "10.139.0.4",  "cs1\_host": "13.232.226.177",  "reverse\_ssh\_host": "52.172.238.250",  "cs0\_dns\_resolve": false,  "cs1\_dns\_resolve": false,  "reverse\_ssh\_dns\_resolve": false,  "cs0\_port\_model\_receiver": 4005,  "cs1\_port\_model\_receiver": 4006,  "cs0\_port\_cs0\_output\_receiver": 4007,  "cs0\_port\_cs1\_output\_receiver": 4008,  "cs0\_port\_inference": 4009,  "cs1\_port\_inference":4010,  "relu0\_port\_inference": 4012,  "relu1\_port\_inference": 4013,  "cs0\_port\_image\_receiver": 4014,  "cs1\_port\_image\_receiver": 4015,  "fractional\_bits": 13,  "image\_id": 1,  "image\_rows": 28,  "image\_cols": 28,  "channels": 1,  "split\_layers\_genr": [{"layer\_id":1, "kernels":5, "splits":5, "kernel\_rows":5, "kernel\_cols":5, "pads":[1,1,0,0], "strides": [2,2], "row\_split": 1},  {"layer\_id":2, "kernels":3, "splits":3, "kernel\_rows":4, "kernel\_cols":4, "pads":[1,1,0,0], "strides": [2,2], "row\_split": 1},  {"layer\_id":3, "rows":100, "splits":1},  {"layer\_id":4, "rows":10, "splits":1}]   } |

The above config file(smpc-mnist-config.json) is an example for running cnn on mnist data.

- Update split\_layers\_genr according to the model and required memory optimisation.

* *“Splits”* (in 1st 2 layers) - is the number of vertical splits. Its maximum value is the number of kernels in case of the CNN layer. In the case of the NN layer(layer id 3 and 4) , “splits”(in NN layer) is the number of splits done in rows of a model.Its maximum value is the number of rows of model in that layer.
* *“row\_split” -* is the number of horizontal splits of an image. Its maximum is output\_rows (after CNN) of each layer.

**If any of the server config file is changed , then the other one should also be changed (same changes)in other server.**

1. cd iudx-MOTION2NX/scripts/ServerModel\_Architecture/Split/CNN/
2. List the files inside the folder and select the docker compose file user wants to run.

1. Enter the below cmd lines to execute scripts in docker :

Run the below commands in the 2 servers simultaneously , to run on mnist data.For cifar data , replace mnist with cifar.

For server0:

|  |
| --- |
| docker compose -f docker-compose.mnist\_Hsplit\_servermodel.yaml -f ../../../../docker-compose.remote-registry.yaml up -d smpc-server0 |

(or)

|  |
| --- |
| docker compose -f docker-compose.cifar\_Hsplit\_servermodel.yaml -f ../../../../docker-compose.remote-registry.yaml up -d smpc-server0 |

For server1:

|  |
| --- |
| docker compose -f docker-compose.mnist\_Hsplit\_servermodel.yaml -f ../../../../docker-compose.remote-registry.yaml up -d smpc-server1 |

(or)

|  |
| --- |
| docker compose -f docker-compose.cifar\_Hsplit\_servermodel.yaml  -f ../../../../docker-compose.remote-registry.yaml up -d smpc-server1 |

The below commands are common to both servers.

To check the logs

|  |
| --- |
| docker compose -f docker-compose.mnist\_Hsplit\_servermodel.yaml  -f ../../../../docker-compose.remote-registry.yaml logs -f |

To bring the docker container down

|  |
| --- |
| docker compose -f docker-compose.mnist\_Hsplit\_servermodel.yaml  -f ../../../docker-compose.remote-registry.yaml down |

1. The image provider and model provider send image and mode shares from their respective systems(locally).

bash ImageProvider.sh

1. In ModelProvider\_CNN.sh , change the existing smpc\_config\_path in script to

To run cnn with cifar data, change the

smpc\_config\_path=${BASE\_DIR}/config\_files/**cifar**-smpc-cnn-config.json

To run cnn with mnist data, change the

smpc\_config\_path=${BASE\_DIR}/config\_files/**mnist**-smpc-cnn-config.json

bash ModelProvider\_CNN.sh

#### **Steps to run “*Convolutional Neural Network with options of vertical splits only.*”**

This guide outlines the steps to run Convolutional Neural Network (CNN) with vertical splits on cloud VMs using docker.

1. Two servers(server0 and server1) are needed to run this setup. After getting access to the clouds , cd iudx-MOTION2NX/config\_files. To get the correct configuration file to change , refer to section II.

|  |
| --- |
| {  "cs0\_host": "10.139.0.4",  "cs1\_host": "13.232.226.177",  "reverse\_ssh\_host": "52.172.238.250",  "cs0\_dns\_resolve": false,  "cs1\_dns\_resolve": false,  "reverse\_ssh\_dns\_resolve": false,  "cs0\_port\_model\_receiver": 4005,  "cs1\_port\_model\_receiver": 4006,  "cs0\_port\_cs0\_output\_receiver": 4007,  "cs0\_port\_cs1\_output\_receiver": 4008,  "cs0\_port\_inference": 4009,  "cs1\_port\_inference":4010,  "relu0\_port\_inference": 4012,  "relu1\_port\_inference": 4013,  "cs0\_port\_image\_receiver": 4014,  "cs1\_port\_image\_receiver": 4015,  "fractional\_bits": 13,  "image\_id": 1,  "image\_rows": 28,  "image\_cols": 28,  "channels": 1,  "split\_layers\_genr": [{"layer\_id":1, "kernels":5, "splits":5},  {"layer\_id":2, "kernels":3, "splits"3},  {"layer\_id":3, "rows":108, "splits":1},  {"layer\_id":4, "rows":10, "splits":2}] } |

The above config file is an example for running cnn on mnist data in cloud VM.This config file is for doing memory optimizations according to the vertical splits of kernel in CNN layer.

To change for cifar data add more layers according to the model of cifar.(refer section II for model\_configuration)

-Update split\_layers\_genr according to the model and required memory optimisation.

* *“splits”(in 1st and 2nd layer)* - is the number of vertical splits. Its maximum value is the number of kernels in case of CNN layer.
* “splits” (in 3rd and 4th layer) is the number of splits done in rows of a model.Its maximum value is the number of rows of model in that layer.

1. cd iudx-MOTION2NX/scripts/ServerModel\_Architecture/Split/CNN/
2. List the files inside the folder and select the docker compose file user wants to run.

Enter the below cmd lines to execute scripts in docker :

For server0:

|  |
| --- |
| docker compose -f docker-compose.cnn\_split\_servermodel.yaml  -f ../../../../docker-compose.remote-registry.yaml up -d smpc-server0 |

For server1:

|  |
| --- |
| docker compose -f docker-compose.cnn\_split\_servermodel.yaml  -f ../../../../docker-compose.remote-registry.yaml up -d smpc-server1 |

The below commands are common to both servers.

To check the logs

|  |
| --- |
| docker compose -f docker-compose.mnist\_Hsplit\_servermodel.yaml  -f ../../../../docker-compose.remote-registry.yaml logs -f |

To bring the docker container down

|  |
| --- |
| docker compose -f docker-compose.mnist\_Hsplit\_servermodel.yaml  -f ../../../../docker-compose.remote-registry.yaml down |

1. In ModelProvider\_CNN.sh , change the existing smpc\_config\_path in script to

smpc\_config\_path=${BASE\_DIR}/config\_files/smpc-cnn-config.json

1. The image provider and model provider send image and mode shares from their respective systems(locally).

bash ImageProvider.sh

bash ModelProvider\_CNN.sh

#### **Steps to run “*Simple Neural Network using helpernode.*” (Any number of layers.)**

This guide outlines the steps to run a simple neural network using helpernode on cloud VMs using docker.

1. Three servers(server0 ,server1,server2) are needed to run this setup.

ssh ubuntu@20.197.27.117

ssh ubuntu@13.200.145.195

ssh ubuntu@15.206.80.191

After getting access to the clouds , cd iudx-MOTION2NX/config\_files. To get the correct configuration file to change , refer to section II.

|  |
| --- |
| {  "cs0\_host": "20.197.27.117",  "cs1\_host": "13.200.145.195",  "helpernode\_host": "15.206.80.191",  "reverse\_ssh\_host": "52.172.238.250",  "cs0\_dns\_resolve": false,  "cs1\_dns\_resolve": false,  "helpernode\_dns\_resolve": false,  "reverse\_ssh\_dns\_resolve": false,  "cs0\_port\_model\_receiver": 4005,  "cs1\_port\_model\_receiver": 4006,  "cs0\_port\_cs0\_output\_receiver": 4007,  "cs0\_port\_cs1\_output\_receiver": 4008,  "cs0\_port\_inference": 4009,  "cs1\_port\_inference":4010,  "helpernode\_port\_inference":4011,  "relu0\_port\_inference": 4012,  "relu1\_port\_inference": 4013,  "cs0\_port\_image\_receiver": 4014,  "cs1\_port\_image\_receiver": 4015,  "number\_of\_layers": 5,  "fractional\_bits": 13,  "image\_id": 0,  "layer\_types": [1,1,1,1,0,0] } |

The above config file is an example for running Neural Network on mnist data on cloud VMS

* The “number\_of\_layers” is 2 or 5 , for the given model for doing inference on nn(mnist data).
* No need to change “layer\_types”.

1. cd iudx-MOTION2NX/scripts/ServerModel\_Architecture/HelperNode
2. List the files inside the folder and select the docker compose file user wants to run.

Enter the below cmd lines to execute scripts in docker :

For server0:

|  |
| --- |
| docker compose -f docker-compose.remote\_helpernode\_servermodel.yaml -f ../../../docker-compose.remote-registry.yaml up -d smpc-server0 |

For server1:

|  |
| --- |
| docker compose -f docker-compose.remote\_helpernode\_servermodel.yaml  -f ../../../docker-compose.remote-registry.yaml up -d smpc-server1 |

For server2:

|  |
| --- |
| docker compose -f docker-compose.remote\_helpernode\_servermodel.yaml  -f ../../../docker-compose.remote-registry.yaml up -d smpc-server2 |

The below commands are common to both servers.

To check the logs

|  |
| --- |
| docker compose -f docker-compose.remote\_helpernode\_servermodel.yaml  -f ../../../docker-compose.remote-registry.yaml logs -f |

To bring the docker container down

|  |
| --- |
| docker compose -f docker-compose.mnist\_Hsplit\_servermodel.yaml  -f ../../../docker-compose.remote-registry.yaml down |

1. The image provider and model provider send image and mode shares from their respective systems(locally).

bash ImageProvider.sh

bash ModelProvider.sh

#### **Steps to run “*CNN + NN using helper node.*” (Any number of layers.)**

This guide outlines the steps to run CNN using helper node on cloud VMs using docker.

1. Three servers(server0 ,server1,server2) are needed to run this setup.

ssh ubuntu@20.197.27.117

ssh ubuntu@13.200.145.195

ssh ubuntu@15.206.80.191

After getting access to the clouds , cd iudx-MOTION2NX/config\_files. To get the correct configuration file to change , refer to section II.

|  |
| --- |
| {  "cs0\_host": "20.197.27.117",  "cs1\_host": "13.200.145.195",  "helpernode\_host": "15.206.80.191",  "reverse\_ssh\_host": "52.172.238.250",  "cs0\_dns\_resolve": false,  "cs1\_dns\_resolve": false,  "helpernode\_dns\_resolve": false,  "reverse\_ssh\_dns\_resolve": false,  "cs0\_port\_model\_receiver": 4005,  "cs1\_port\_model\_receiver": 4006,  "cs0\_port\_cs0\_output\_receiver": 4007,  "cs0\_port\_cs1\_output\_receiver": 4008,  "cs0\_port\_inference": 4009,  "cs1\_port\_inference":4010,  "helpernode\_port\_inference":4011,  "relu0\_port\_inference": 4012,  "relu1\_port\_inference": 4013,  "cs0\_port\_image\_receiver": 4014,  "cs1\_port\_image\_receiver": 4015,  "number\_of\_layers": 6,  "fractional\_bits": 13,  "image\_id": 0,  "layer\_types": [1,1,1,1,0,0] } |

The above config file is an example for running CNN and NN on cifar data using helpernode.

* The “number\_of\_layers” is according to the number of layers in the given model for doing inference on nn(mnist data).
* In “layer\_types , 1 denotes CNN layer and 0 denotes NN layer.

Example , if first 2 layers are CNN and next 2 are NN , then the layer\_type is [1,1,0,0]

If first 4 layers are CNN and next 2 are NN , then the layer\_types is [1,1,1,1,0,0]

1. cd iudx-MOTION2NX/scripts/ServerModel\_Architecture/HelperNode/CNN
2. List the files inside the folder and select the docker compose file user wants to run.

Enter the below cmd lines to execute scripts in docker :

For server0:

|  |
| --- |
| docker compose -f docker-compose.remote\_helpernode\_cnnservermodel.yaml -f ../../../../docker-compose.remote-registry.yaml up -d smpc-server0 |

For server1:

|  |
| --- |
| docker compose -f docker-compose.remote\_helpernode\_cnnservermodel.yaml -f ../../../../docker-compose.remote-registry.yaml up -d smpc-server1 |

For server2:

|  |
| --- |
| docker compose -f docker-compose.remote\_helpernode\_cnnservermodel.yaml -f ../../../../docker-compose.remote-registry.yaml up -d smpc-server2 |

The below commands are common to both servers.

To check the logs

|  |
| --- |
| docker compose -f docker-compose.remote\_helpernode\_cnnservermodel.yaml  -f ../../../../docker-compose.remote-registry.yaml logs -f |

To bring the docker container down

|  |
| --- |
| docker compose -f docker-compose.remote\_helpernode\_cnnservermodel.yaml  -f ../../../../docker-compose.remote-registry.yaml down |

1. The image provider and model provider send image and mode shares from their respective systems(locally).

bash ImageProvider.sh

bash ModelProvider.sh