```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", f
```

```
!cp /content/drive/MyDrive/FL/FLARE\ sample/tf2_net.py /content
```

```
import tf2_net
```

```
!pip install nvflare
```

```
Collecting nvflare
  Downloading nvflare-2.0.12-py3-none-any.whl (799 kB)
     |████████████████████████████████| 799 kB 5.4 MB/s
Collecting cryptography
  Downloading cryptography-36.0.1-cp36-abi3-manylinux_2_24_x86_64.whl (3.6 MB)
     |████████████████████████████████| 3.6 MB 34.2 MB/s
Requirement already satisfied: google-api-python-client in /usr/local/lib/python3.7/dist-packages (from nvfla
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/dist-packages (from nvflare) (3.13)
Requirement already satisfied: grpcio in /usr/local/lib/python3.7/dist-packages (from nvflare) (1.43.0)
Requirement already satisfied: psutil in /usr/local/lib/python3.7/dist-packages (from nvflare) (5.4.8)
Collecting tenseal==0.3.0
  Downloading tenseal-0.3.0-cp37-cp37m-manylinux2014_x86_64.whl (4.4 MB)
     |████████████████████████████████| 4.4 MB 31.1 MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from nvflare) (1.19.5)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.7/dist-packages (from cryptography->nvfla
Requirement already satisfied: pycparser in /usr/local/lib/python3.7/dist-packages (from cffi>=1.12->cryptogr
Requirement already satisfied: uritemplate<4dev,>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from googl
Requirement already satisfied: httplib2<1dev,>=0.15.0 in /usr/local/lib/python3.7/dist-packages (from google-
Requirement already satisfied: google-auth<3dev,>=1.16.0 in /usr/local/lib/python3.7/dist-packages (from goog
Requirement already satisfied: google-api-core<3dev,>=1.21.0 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: six<2dev,>=1.13.0 in /usr/local/lib/python3.7/dist-packages (from google-api-p
Requirement already satisfied: google-auth-httplib2>=0.0.3 in /usr/local/lib/python3.7/dist-packages (from go
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.6.0 in /usr/local/lib/python3.7/dist-packa
```

```
Requirement already satisfied: setuptools>=40.3.0 in /usr/local/lib/python3.7/dist-packages (from google-api-
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in /usr/local/lib/python3.7/dist-packages (from goo
Requirement already satisfied: protobuf>=3.12.0 in /usr/local/lib/python3.7/dist-packages (from google-api-co
Requirement already satisfied: packaging>=14.3 in /usr/local/lib/python3.7/dist-packages (from google-api-cor
Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages (from google-api-core<3dev,>=1.
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from google-a
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (from google-auth<3dev
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from google-
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packa
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-packages (from pyasn1-mo
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests<3.
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests<3.0
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests<3.0.0dev
Installing collected packages: tenseal, cryptography, nvflare
Successfully installed cryptography-36.0.1 nvflare-2.0.12 tenseal-0.3.0
```

```
# Copyright (c) 2021, NVIDIA CORPORATION.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

import tensorflow as tf
import numpy as np

from nvflare.apis.dxo import DXO, DataKind, from_shareable
from nvflare.apis.fl_constant import ReturnCode
from nvflare.apis.event_type import EventType
from nvflare.apis.executor import Executor
from nvflare.apis.fl_context import FLContext
```

```
from nvflare.apis.shareable import Shareable, make_reply
from nvflare.apis.signal import Signal

from tf2_net import Net


class SimpleTrainer(Executor):
    def __init__(self, epochs_per_round):
        super().__init__()
        self.epochs_per_round = epochs_per_round
        self.train_images, self.train_labels = None, None
        self.test_images, self.test_labels = None, None
        self.model = None

    def handle_event(self, event_type: str, fl_ctx: FLContext):
        if event_type == EventType.START_RUN:
            self.setup(fl_ctx)

    def setup(self, fl_ctx: FLContext):
        (self.train_images, self.train_labels), (
            self.test_images,
            self.test_labels,
        ) = tf.keras.datasets.mnist.load_data()
        self.train_images, self.test_images = (
            self.train_images / 255.0,
            self.test_images / 255.0,
        )

        # simulate separate datasets for each client by dividing MNIST dataset in half
        client_name = fl_ctx.get_identity_name()
        if client_name == "site-1":
            self.train_images = self.train_images[: len(self.train_images) // 2]
            self.train_labels = self.train_labels[: len(self.train_labels) // 2]
            self.test_images = self.test_images[: len(self.test_images) // 2]
            self.test_labels = self.test_labels[: len(self.test_labels) // 2]
        elif client_name == "site-2":
            self.train_images = self.train_images[len(self.train_images) // 2 :]
            self.train_labels = self.train_labels[len(self.train_labels) // 2 :]
            self.test_images = self.test_images[len(self.test_images) // 2 :]
```

```
        self.test_labels = self.test_labels[len(self.test_labels) // 2 :]

    model = Net()

    loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
    model.compile(optimizer="adam", loss=loss_fn, metrics=["accuracy"])
    _ = model(tf.keras.Input(shape=(28, 28)))
    self.var_list = [model.get_layer(index=index).name for index in range(len(model.get_weights()))]
    self.model = model

def execute(
    self,
    task_name: str,
    shareable: Shareable,
    fl_ctx: FLContext,
    abort_signal: Signal,
) -> Shareable:
    """
    This function is an extended function from the super class.
    As a supervised learning based trainer, the train function will run
    evaluate and train engines based on model weights from `shareable`.
    After finishing training, a new `Shareable` object will be submitted
    to server for aggregation.
    Args:
        task_name: dispatched task
        shareable: the `Shareable` object acheived from server.
        fl_ctx: the `FLContext` object achieved from server.
        abort_signal: if triggered, the training will be aborted.
    Returns:
        a new `Shareable` object to be submitted to server for aggregation.
    """

    # retrieve model weights download from server's shareable
    if abort_signal.triggered:
        return make_reply(ReturnCode.TASK_ABORTED)

    if task_name != "train":
        return make_reply(ReturnCode.TASK_UNKNOWN)
```

```
        dxo = from_shareable(shareable)
        model_weights = dxo.data

        # use previous round's client weights to replace excluded layers from server
        prev_weights = {
            self.model.get_layer(index=key).name: value for key, value in enumerate(self.model.get_weights())
        }

        ordered_model_weights = {key: model_weights.get(key) for key in prev_weights}
        for key in self.var_list:
            value = ordered_model_weights.get(key)
            if np.all(value == 0):
                ordered_model_weights[key] = prev_weights[key]

        # update local model weights with received weights
        self.model.set_weights(list(ordered_model_weights.values()))

        # adjust LR or other training time info as needed
        # such as callback in the fit function
        self.model.fit(
            self.train_images,
            self.train_labels,
            epochs=self.epochs_per_round,
            validation_data=(self.test_images, self.test_labels),
        )

        # report updated weights in shareable
        weights = {self.model.get_layer(index=key).name: value for key, value in enumerate(self.model.get_weights(
        dxo = DXO(data_kind=DataKind.WEIGHTS, data=weights)

        self.log_info(fl_ctx, "Local epochs finished. Returning shareable")
        new_shareable = dxo.to_shareable()
        return new_shareable


# Copyright (c) 2021, NVIDIA CORPORATION.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
```

```
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

import os
import pickle
import json

import tensorflow as tf
from nvflare.apis.event_type import EventType
from nvflare.apis.fl_constant import FLContextKey
from nvflare.apis.fl_context import FLContext
from nvflare.app_common.abstract.model import ModelLearnable
from nvflare.app_common.abstract.model_persistor import ModelPersistor
from tf2_net import Net
from nvflare.app_common.app_constant import AppConstants
from nvflare.app_common.abstract.model import make_model_learnable


class TF2ModelPersistor(ModelPersistor):
    def __init__(self, save_name="tf2_model.pkl"):
        super().__init__()
        self.save_name = save_name

    def _initialize(self, fl_ctx: FLContext):
        # get save path from FLContext
        app_root = fl_ctx.get_prop(FLContextKey.APP_ROOT)
        env = None
        run_args = fl_ctx.get_prop(FLContextKey.ARGS)
        if run_args:
            env_config_file_name = os.path.join(app_root, run_args.env)
            if os.path.exists(env_config_file_name):
                try:
```

```
                with open(env_config_file_name) as file:
                    env = json.load(file)
            except:
                self.system_panic(
                    reason="error opening env config file {}".format(env_config_file_name), fl_ctx=fl_ctx
                )
                return

    if env is not None:
        if env.get("APP_CKPT_DIR", None):
            fl_ctx.set_prop(AppConstants.LOG_DIR, env["APP_CKPT_DIR"], private=True, sticky=True)
        if env.get("APP_CKPT") is not None:
            fl_ctx.set_prop(
                AppConstants.CKPT_PRELOAD_PATH,
                env["APP_CKPT"],
                private=True,
                sticky=True,
            )

    log_dir = fl_ctx.get_prop(AppConstants.LOG_DIR)
    if log_dir:
        self.log_dir = os.path.join(app_root, log_dir)
    else:
        self.log_dir = app_root
    self._pkl_save_path = os.path.join(self.log_dir, self.save_name)
    if not os.path.exists(self.log_dir):
        os.makedirs(self.log_dir)

    fl_ctx.sync_sticky()

def load_model(self, fl_ctx: FLContext) -> ModelLearnable:
    """
        initialize and load the Model.
    Args:
        fl_ctx: FLContext
    Returns:
        Model object
    """
```

```
        if os.path.exists(self._pkl_save_path):
            self.logger.info(f"Loading server weights")
            with open(self._pkl_save_path, "rb") as f:
                model_learnable = pickle.load(f)
        else:
            self.logger.info(f"Initializing server model")
            network = Net()
            loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
            network.compile(optimizer="adam", loss=loss_fn, metrics=["accuracy"])
            _ = network(tf.keras.Input(shape=(28, 28)))
            var_dict = {network.get_layer(index=key).name: value for key, value in enumerate(network.get_weights()
            model_learnable = make_model_learnable(var_dict, dict())
        return model_learnable

    def handle_event(self, event: str, fl_ctx: FLContext):
        if event == EventType.START_RUN:
            self._initialize(fl_ctx)

    def save_model(self, model_learnable: ModelLearnable, fl_ctx: FLContext):
        """
            persist the Model object
        Args:
            model: Model object
            fl_ctx: FLContext
        """
        model_learnable_info = {k: str(type(v)) for k, v in model_learnable.items()}
        self.logger.info(f"Saving aggregated server weights: \n {model_learnable_info}")
        with open(self._pkl_save_path, "wb") as f:
            pickle.dump(model_learnable, f)
```

```
!poc -n 2
```

```
    This will delete poc folder in current directory and create a new one. Is it OK to proceed? (y/N) y
    Successfully creating poc folder.  Please read poc/Readme.rst for user guide.
```

```
!mkdir -p poc/admin/transfer
```

```
!cp -rf /content/drive/MyDrive/FL/NVFlare-main/examples/* poc/admin/transfer
```

```
!nohup bash ./poc/server/startup/start.sh &
```

```
nohup: appending output to 'nohup.out'
```

```
!nohup bash ./poc/site-1/startup/start.sh &
```

```
nohup: appending output to 'nohup.out'
```

```
!nohup bash ./poc/site-2/startup/start.sh &
```

```
nohup: appending output to 'nohup.out'
```

```
!./poc/admin/startup/fl_admin.sh localhost
```

```
/content/poc/admin/startup
Admin Server: localhost on port 8003
User Name: admin
Password:
Type ? to list commands; type "? cmdName" to show usage of a command.
> check_status server
FL_app name: ?
Engine status: stopped
Run number has not been set.
Registered clients: 2
--------------------------------------------------------------------------------
| CLIENT | TOKEN                                | LAST CONNECT TIME        |
--------------------------------------------------------------------------------
| site-1 | d2a8e53a-78a9-4e48-b8d1-83dcfc39d8be | Sun Feb 13 18:38:43 2022 |
| site-2 | 9094ec01-1dad-4fdf-92b4-7f922de61b50 | Sun Feb 13 18:38:46 2022 |
--------------------------------------------------------------------------------
Done [1532 usecs] 2022-02-13 18:39:38.174908
> upload_app hello-tf2
Created folder /content/poc/server/startup/../transfer/hello-tf2
```

```
> set_run_number 1
Created a new run folder: run_1
---------------------
| CLIENT | RESPONSE |
---------------------
| site-1 | OK       |
| site-2 | OK       |
---------------------
Done [502768 usecs] 2022-02-13 18:43:29.389927
> deploy_app hello-tf2 all
deployed app "hello-tf2" to Server
---------------------
| CLIENT | RESPONSE |
---------------------
| site-1 | OK       |
| site-2 | OK       |
---------------------
Done [406576 usecs] 2022-02-13 18:43:48.774886
> start_app all
Server app is starting....
---------------------
| CLIENT | RESPONSE |
---------------------
| site-1 |          |
| site-2 |          |
---------------------
Done [15383932 usecs] 2022-02-13 18:44:22.332943
> check_status server
FL_app name: hello-tf2
Engine status: started
Current run number: 1
Registered clients: 2
------------------------------------------------------------------------
| CLIENT | TOKEN                                | LAST CONNECT TIME        |
------------------------------------------------------------------------
| site-1 | d2a8e53a-78a9-4e48-b8d1-83dcfc39d8be | Sun Feb 13 18:44:43 2022 |
| site-2 | 9094ec01-1dad-4fdf-92b4-7f922de61b50 | Sun Feb 13 18:44:46 2022 |
------------------------------------------------------------------------
Done [1600 usecs] 2022-02-13 18:44:53.130865
```

```
from google.colab import drive
drive.mount('/content/drive')
```