```
import pandas as pd
import numpy as np
import torch
from torch import nn
from torch.utils.data import dataloader
from torchvision import datasets
from torchvision.transforms import ToTensor
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.metrics import classification_report, confusion_matrix
from scipy.stats import loguniform
import warnings
warnings.simplefilter('ignore')
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", force_remou
```

```
path="/content/drive/MyDrive/data set/clintox.csv"
path="/content/drive/MyDrive/data set/clintox_global_cdf_rdkit.csv"
clintox_dataset=pd.read_csv("/content/drive/MyDrive/data set/clintox.csv")
clintox_features=pd.read_csv("/content/drive/MyDrive/data set/clintox_global_cdf_rdkit.csv")
```

```
clintox_features=clintox_features.loc[:,clintox_features.apply(pd.Series.nunique) !=1]
clintox_dataset=clintox_dataset.iloc[clintox_features.dropna().index]
clintox_dataset=clintox_dataset.reset_index()
clintox_features=clintox_features.dropna()
clintox_features=clintox_features.reset_index()
index_array=[]
for i in np.arange(1,3):
  index_array.append(clintox_dataset.iloc[:,i+1].dropna().index)
```

```
def label_ith(i):
```

```
  return pd.DataFrame(data=clintox_dataset.iloc[index_array[i]].iloc[:,i+2])
def feature_ith(i):
  return clintox_features.iloc[index_array[i]].drop('index',axis=1)


## train_test_split with sklearn library.
X_training_data=[]
X_test=[]
y_training_data=[]
y_test=[]
for i in np.arange(0,2):
  X_training_data_tmp,X_test_tmp,y_training_data_tmp,y_test_tmp=train_test_split(feature_ith(i),
                                                label_ith(i),
                                                stratify=label_ith(i),
                                                test_size=0.20,
                                                random_state=1234)
  X_training_data.append(X_training_data_tmp)
  X_test.append(X_test_tmp)
  y_training_data.append(y_training_data_tmp)
  y_test.append(y_test_tmp)


X_training_data_tmp.shape,X_test_tmp.shape,y_training_data_tmp.shape,y_test_tmp.shape
```

```
     ((1168, 186), (293, 186), (1168, 1), (293, 1))
```

```
## principal components
from sklearn.decomposition import PCA
X_training_data_pca=[]
X_test_pca=[]
for i in np.arange(0,2):
  pca=PCA(n_components=67)
  principalComponents=pca.fit_transform(X_training_data[i])
  X_training_data_PCA_tmp=pd.DataFrame(data=principalComponents)
  X_training_data_pca.append(X_training_data_PCA_tmp)
  X_test_pca_tmp=pd.DataFrame(data=pca.transform(X_test[i]))
  X_test_pca.append(X_test_pca_tmp)
```

```
  print('PCA with 67 principal components retains',
        np.sum(pca.explained_variance_ratio_)*100,'% of data VAR.(it is related for',i,'th label)')
```

```
PCA with 67 principal components retains 95.0738154170295 % of data VAR.(it is related for 0 th label)
PCA with 67 principal components retains 95.05301828482887 % of data VAR.(it is related for 1 th label)
```

```
##Hyperparameter Optimization with random search
### Do not implement due to time cost.
# define model
model = LogisticRegression()
# define evaluation
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
# define search space
space = dict()
space['solver'] = ['newton-cg', 'lbfgs', 'liblinear']
space['penalty'] = ['none', 'l1', 'l2', 'elasticnet']
space['C'] = loguniform(1e-5,1e-4, 1e-3,100)
from sklearn.model_selection import RandomizedSearchCV
# define search
search = RandomizedSearchCV(model, space, n_iter=500, scoring='accuracy',
                            n_jobs=-1, cv=cv, random_state=1)
# execute search
result = search.fit(X_training_data_tmp, y_training_data_tmp)
# summarize result
print('Best Score: %s' % result.best_score_)
print('Best Hyperparameters: %s' % result.best_params_)


##Hyperparameter Optimization with Gridsearch
### Do not implement due to time cost.
# define model
model = LogisticRegression()
# define evaluation
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
# define search space
space = dict()
space['solver'] = ['newton-cg', 'lbfgs', 'liblinear']
space['penalty'] = ['none', 'l1', 'l2', 'elasticnet']
space['C'] = [1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100]
from sklearn.model_selection import GridSearchCV
```

```
# define search
search = GridSearchCV(model, space, scoring='accuracy', n_jobs=-1, cv=cv)
# execute search
result = search.fit(X_training_data_tmp, y_training_data_tmp)
# summarize result
print('Best Score: %s' % result.best_score_)
print('Best Hyperparameters: %s' % result.best_params_)
```

```
Best Score: 0.9246588017580382
Best Hyperparameters: {'C': 1e-05, 'penalty': 'l1', 'solver': 'liblinear'}
```

```
from sklearn import preprocessing

scaler_data = preprocessing.MinMaxScaler()
train_data = scaler_data.fit_transform(X_training_data_tmp)
test_data = scaler_data.transform(X_test_tmp)

scaler_labels = preprocessing.MinMaxScaler()
train_labels_before = y_training_data_tmp.values.reshape(-1, 1)
train_labels = scaler_labels.fit_transform(y_training_data_tmp.values.reshape(-1, 1))
test_labels = scaler_labels.transform(y_test_tmp.values.reshape(-1, 1))

print(train_data.shape, train_labels.shape, test_data.shape, test_labels.shape)
print("Train labels before scaling: {} {} {}Train labels after scaling: {} {}".format('\n',train_labels_before,'\n', '\n', tr
```

```
(1168, 186) (1168, 1) (293, 186) (293, 1)
Train labels before scaling:
 [[1]
 [0]
 [0]
 ...
 [0]
 [0]
 [0]]
Train labels after scaling:
 [[1.]
 [0.]
 [0.]
 ...
```

```
      [0.]
      [0.]
      [0.]]
```

```
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
```

```
# transform to torch tensor

tensor_x = torch.tensor(train_data, dtype=torch.float).to(device)
tensor_x2 = torch.tensor(test_data, dtype=torch.float).to(device)

tensor_y = torch.tensor(train_labels, dtype=torch.float).to(device)
tensor_y2 = torch.tensor(test_labels, dtype=torch.float).to(device)
```

```
# create your dataset
from torch.utils.data import TensorDataset
trainset = TensorDataset(tensor_x, tensor_y)
testset = TensorDataset(tensor_x2,tensor_y2)
```

```
trainset[0]
```

```
     (tensor([0.9556, 0.1962, 0.4677, 0.5947, 0.5463, 0.4044, 0.5613, 0.4802, 0.5996,
              0.4911, 0.6461, 0.5312, 0.6462, 0.5241, 0.0000, 0.2649, 0.0000, 0.5283,
              0.1279, 0.3328, 0.8861, 0.0000, 0.6019, 0.9856, 0.3497, 0.3599, 0.2946,
              0.0882, 0.0344, 0.6316, 0.7318, 0.4163, 0.3115, 0.5841, 0.6270, 0.5460,
              0.4471, 0.2024, 0.5522, 0.2024, 0.7530, 0.7755, 0.7528, 0.5567, 0.4358,
              0.4558, 0.5062, 0.3551, 0.3994, 0.0938, 0.0000, 0.0000, 0.0000, 0.6222,
              0.0000, 0.3789, 0.0670, 0.4438, 0.0334, 0.0000, 0.6922, 0.0000, 0.0000,
              0.0000, 0.4435, 0.3306, 0.4853, 0.0000, 0.0000, 0.0000, 0.8986, 0.2684,
              0.0000, 0.0000, 0.0000, 0.9105, 0.1689, 0.2289, 0.8420, 0.1159, 0.5146,
              0.0678, 0.0000, 0.0000, 0.0000, 0.8202, 0.4625, 0.7236, 0.0000, 0.2591,
              0.0000, 0.0000, 0.0000, 0.4362, 0.6534, 0.0000, 0.7810, 0.7543, 0.0000,
              0.0000, 0.0416, 0.4999, 0.0000, 0.0755, 0.5006, 0.0000, 0.0000, 0.0000,
              0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.3728, 0.4628,
              0.0000, 0.0000, 0.0000, 0.0000, 0.5978, 0.0000, 0.0000, 0.0000, 0.0000,
              0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
              0.0000, 0.0000, 0.0000, 0.0000, 0.6222, 0.0000, 0.0000, 0.0000, 0.0000,
              1.0000, 0.7894, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
```

```
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000,
        0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.7393]), tensor([1.]))
```

```
!pip install -U "ray[default]"
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: ray[default] in /usr/local/lib/python3.7/dist-packages (1.13.0)
Requirement already satisfied: frozenlist in /usr/local/lib/python3.7/dist-packages (from ray[default]) (1.3.0)
Requirement already satisfied: numpy>=1.16 in /usr/local/lib/python3.7/dist-packages (from ray[default]) (1.21.6)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from ray[default]) (2.23.0)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.7/dist-packages (from ray[default]) (3.13)
Requirement already satisfied: jsonschema in /usr/local/lib/python3.7/dist-packages (from ray[default]) (4.3.3)
Requirement already satisfied: aiosignal in /usr/local/lib/python3.7/dist-packages (from ray[default]) (1.2.0)
Requirement already satisfied: protobuf<4.0.0,>=3.15.3 in /usr/local/lib/python3.7/dist-packages (from ray[default])
Requirement already satisfied: grpcio<=1.43.0,>=1.28.1 in /usr/local/lib/python3.7/dist-packages (from ray[default])
Requirement already satisfied: msgpack<2.0.0,>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from ray[default]) (
Requirement already satisfied: attrs in /usr/local/lib/python3.7/dist-packages (from ray[default]) (21.4.0)
Requirement already satisfied: click<=8.0.4,>=7.0 in /usr/local/lib/python3.7/dist-packages (from ray[default]) (7.1
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from ray[default]) (3.7.1)
Requirement already satisfied: virtualenv in /usr/local/lib/python3.7/dist-packages (from ray[default]) (20.15.1)
Requirement already satisfied: smart-open in /usr/local/lib/python3.7/dist-packages (from ray[default]) (5.2.1)
Requirement already satisfied: prometheus-client<0.14.0,>=0.7.1 in /usr/local/lib/python3.7/dist-packages (from ray[
Requirement already satisfied: py-spy>=0.2.0 in /usr/local/lib/python3.7/dist-packages (from ray[default]) (0.3.12)
Requirement already satisfied: opencensus in /usr/local/lib/python3.7/dist-packages (from ray[default]) (0.10.0)
Requirement already satisfied: aiohttp>=3.7 in /usr/local/lib/python3.7/dist-packages (from ray[default]) (3.8.1)
Requirement already satisfied: aiohttp-cors in /usr/local/lib/python3.7/dist-packages (from ray[default]) (0.7.0)
Requirement already satisfied: gpustat>=1.0.0b1 in /usr/local/lib/python3.7/dist-packages (from ray[default]) (1.0.0
Requirement already satisfied: colorful in /usr/local/lib/python3.7/dist-packages (from ray[default]) (0.5.4)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.7/dist-packages (from aiohttp>=3.7->ray[defa
Requirement already satisfied: asynctest==0.13.0 in /usr/local/lib/python3.7/dist-packages (from aiohttp>=3.7->ray[d
Requirement already satisfied: typing-extensions>=3.7.4 in /usr/local/lib/python3.7/dist-packages (from aiohttp>=3.7
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.7/dist-packages (from aiohttp>=3.7->ray
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.7/dist-packages (from aiohttp>=
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in /usr/local/lib/python3.7/dist-packages (from aiohttp>
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.7/dist-packages (from gpustat>=1.0.0b1->ray[defaul
Requirement already satisfied: blessed>=1.17.1 in /usr/local/lib/python3.7/dist-packages (from gpustat>=1.0.0b1->ray
Requirement already satisfied: psutil>=5.6.0 in /usr/local/lib/python3.7/dist-packages (from gpustat>=1.0.0b1->ray[d
Requirement already satisfied: nvidia-ml-py<=11.495.46,>=11.450.129 in /usr/local/lib/python3.7/dist-packages (from
```

```
Requirement already satisfied: wcwidth>=0.1.4 in /usr/local/lib/python3.7/dist-packages (from blessed>=1.17.1->gpust
Requirement already satisfied: idna>=2.0 in /usr/local/lib/python3.7/dist-packages (from yarl<2.0,>=1.0->aiohttp>=3.
Requirement already satisfied: importlib-resources>=1.4.0 in /usr/local/lib/python3.7/dist-packages (from jsonschema
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from jsonschema->ray[de
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.7/dist-packages (from importlib-resources>=1.4.
Requirement already satisfied: google-api-core<3.0.0,>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from opencen
Requirement already satisfied: opencensus-context>=0.1.2 in /usr/local/lib/python3.7/dist-packages (from opencensus-
Requirement already satisfied: pytz in /usr/local/lib/python3.7/dist-packages (from google-api-core<3.0.0,>=1.0.0->o
Requirement already satisfied: setuptools>=40.3.0 in /usr/local/lib/python3.7/dist-packages (from google-api-core<3.
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.6.0 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: packaging>=14.3 in /usr/local/lib/python3.7/dist-packages (from google-api-core<3.0.0
Requirement already satisfied: google-auth<2.0dev,>=1.25.0 in /usr/local/lib/python3.7/dist-packages (from google-ap
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from google-auth<2.0
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from google-auth<2.
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (from google-auth<2.0dev,>=1.
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging>=1
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->ray[defau
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->ray[defa
Requirement already satisfied: distlib<1,>=0.3.1 in /usr/local/lib/python3.7/dist-packages (from virtualenv->ray[def
Requirement already satisfied: platformdirs<3,>=2 in /usr/local/lib/python3.7/dist-packages (from virtualenv->ray[de
```

```
import ray
from functools import partial
import numpy as np
import os
import torch.nn.functional as F
import torch.optim as optim
from torch.utils.data import random_split
from torchsummary import summary

from ray import tune
from ray.tune import CLIReporter
from ray.tune.schedulers import ASHAScheduler


# Function is useful when we want to read the dataset from a file and to share a data directory
```

```
# between different trials (specially when we are working with a large dataset).

def load_data(data_dir=None):
    return trainset, testset

class Net(nn.Module):
    def __init__(self, config):
        super().__init__()

        self.config = config
        self.hidden_dim1 = int(self.config.get("hidden_dim1", 120))
        self.hidden_dim2 = int(self.config.get("hidden_dim2", 120))
        self.hidden_dim3 = int(self.config.get("hidden_dim3", 120))

        self.act1 = self.config.get("act1", "relu")
        self.act2 = self.config.get("act2", "relu")
        self.act3 = self.config.get("act3", "relu")

        self.linear1 = nn.Linear(186, self.hidden_dim1)
        self.linear2 = nn.Linear(self.hidden_dim1, self.hidden_dim2)
        self.linear3 = nn.Linear(self.hidden_dim2, self.hidden_dim3)
        self.linear4 = nn.Linear(self.hidden_dim3, 1)

    @staticmethod
    def activation_func(act_str):
        if act_str=="tanh":
            return eval("torch."+act_str)
        elif act_str=="selu" or act_str=="relu":
            return eval("torch.nn.functional."+act_str)

    def forward(self, x):
        output = self.linear1(x)
        output = self.activation_func(self.act1)(output)
        output = self.linear2(output)
        output = self.activation_func(self.act2)(output)
        output = self.linear3(output)
        output = self.activation_func(self.act3)(output)
```

```
        output = self.linear4(output)
```

```
model = Net({})
```

```
from prettytable import PrettyTable

def count_parameters(model):
    table = PrettyTable(["Modules", "Parameters"])
    total_params = 0
    for name, parameter in model.named_parameters():
        if not parameter.requires_grad: continue
        param = parameter.numel()
        table.add_row([name, param])
        total_params+=param
    print(table)
    print(f"Total Trainable Params: {total_params}")
    return total_params
```

```
tensor_x.shape
```

```
     torch.Size([1168, 186])
```

```
summary(model, (1,tensor_x.shape[1]))

count_parameters(model)
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1               [-1, 1, 120]          22,440
            Linear-2               [-1, 1, 120]          14,520
            Linear-3               [-1, 1, 120]          14,520
            Linear-4                 [-1, 1, 1]             121
================================================================
Total params: 51,601
Trainable params: 51,601
Non-trainable params: 0
----------------------------------------------------------------
```

```
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.20
Estimated Total Size (MB): 0.20
----------------------------------------------------------------
+---------------+-----------+
|    Modules    | Parameters |
+---------------+-----------+
| linear1.weight |   22320    |
|  linear1.bias  |    120     |
| linear2.weight |   14400    |
|  linear2.bias  |    120     |
| linear3.weight |   14400    |
|  linear3.bias  |    120     |
| linear4.weight |    120     |
|  linear4.bias  |     1      |
+---------------+-----------+
Total Trainable Params: 51601
51601
```

```
def trainable_func(config, checkpoint_dir=None, data_dir=None, epochs=10):

    net = Net(config)

    device = "cpu"
    if torch.cuda.is_available():
        device = "cuda:0"
        if torch.cuda.device_count() > 1:
            net = nn.DataParallel(net)
    net.to(device)

    '''
    Define a loss function
    '''
    ## Classification
    criterion = nn.CrossEntropyLoss()

    ## Regression
    #criterion = nn.MSELoss(reduction='sum')
```

```
# Define an optimizer
optimizer = optim.Adam(net.parameters(), lr=config.get("lr",0.0003))

if checkpoint_dir:
    model_state, optimizer_state = torch.load(
        os.path.join(checkpoint_dir, "checkpoint"))
    net.load_state_dict(model_state)
    optimizer.load_state_dict(optimizer_state)

# Load data
trainset, testset = load_data(data_dir)

# Split the dataset into training and validation sets
train_size = int(len(trainset) * 0.8)
train_subset, val_subset = random_split(trainset, [train_size, len(trainset) - train_size])

# Define data loaders (which combines a dataset and a sampler, and provides an iterable over the given dataset)
trainloader = torch.utils.data.DataLoader(
    train_subset,
    batch_size=int(config.get("batch_size",32)),
    shuffle=True,
    num_workers=2)
valloader = torch.utils.data.DataLoader(
    val_subset,
    batch_size=int(config.get("batch_size",32)),
    shuffle=True,
    num_workers=2)

for epoch in range(epochs):  # loop over the dataset multiple times
    epoch_train_loss = 0.0
    # epoch_steps = 0
    net.train() # Prepare model for training
    for i, data in enumerate(trainloader):
        # get the inputs; data is a list of [inputs, labels]
        inputs, labels = data
        inputs, labels = inputs.to(device), labels.to(device)
```

```
        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        '''
        Compute train loss without scaling to print
        '''
        # outputs = torch.tensor(scaler_labels.inverse_transform(outputs.detach().cpu())).to(device)
        # labels = torch.tensor(scaler_labels.inverse_transform(labels.cpu())).to(device)
        # loss_train = criterion(outputs, labels)
        # epoch_train_loss += loss_train.detach().item()
    # print("[%d] loss: %.3f" % (epoch + 1, epoch_train_loss / len(train_subset)))

    # Validation loss
    val_loss = 0.0
    net.eval() # Prepare model for evaluation
    for i, data in enumerate(valloader):
        with torch.no_grad():
            inputs, labels = data
            inputs, labels = inputs.to(device), labels.to(device)

            outputs = net(inputs)

            # Inverse transform of the labels' scaler
            outputs = torch.tensor(scaler_labels.inverse_transform(outputs.detach().cpu())).to(device)
            labels = torch.tensor(scaler_labels.inverse_transform(labels.cpu())).to(device)

            loss = criterion(outputs, labels)
            val_loss += loss.cpu().numpy()

    with tune.checkpoint_dir(epoch) as checkpoint_dir:
        path = os.path.join(checkpoint_dir, "checkpoint")
        torch.save((net.state_dict(), optimizer.state_dict()), path)
```

```
        tune.report(epoch = epoch, loss=(val_loss / len(val_subset)))
    print("Finished Training")


def test_score(config, net, device="cpu"):
    trainset, testset = load_data()

    testloader = torch.utils.data.DataLoader(
        testset, batch_size=int(config.get("batch_size",32)), shuffle=False, num_workers=2)

    ## Regression
    #criterion = nn.MSELoss(reduction='sum')
    criterion = nn.CrossEntropyLoss()

    # Test loss
    test_loss = 0.0
    net.eval() # Prepare model for evaluation
    for i, data in enumerate(testloader):
        with torch.no_grad():
            inputs, labels = data
            inputs, labels = inputs.to(device), labels.to(device)

            outputs = net(inputs)

            # Inverse transform of the labels' scaler
            outputs = torch.tensor(scaler_labels.inverse_transform(outputs.detach().cpu())).to(device)
            labels = torch.tensor(scaler_labels.inverse_transform(labels.cpu())).to(device)

            loss = criterion(outputs, labels)
            test_loss += loss.cpu().numpy()

    return test_loss / len(testset)


ray.init() # Here we use ray.init() to evaluate available_resources for Ray
print(ray.available_resources())
ray.shutdown()  # Restart Ray defensively in case the ray connection is lost.
```

```
# Start Ray runtime with specific resources (not nessesarily all resources)
# You can change this values based on your machine resources)
ray.init(num_cpus=4, num_gpus=0)

"""Check Ray Tune is working properly (for trainable class)"""
# from ray.tune.utils import validate_save_restore
# validate_save_restore(Trainable)
# validate_save_restore(Trainable, use_object_store=True)
# print("Success!")
""""""
```

```
2022-07-18 08:21:50,552 INFO services.py:1476 -- View the Ray dashboard at http://127.0.0.1:8265
{'CPU': 2.0, 'memory': 7940225435.0, 'object_store_memory': 3970112716.0, 'node:172.28.0.2': 1.0}
2022-07-18 08:21:59,862 INFO services.py:1476 -- View the Ray dashboard at http://127.0.0.1:8265
''
```

```
%%capture
try:
    import optuna
except:
    %pip install optuna
    import optuna
```

```
from sklearn.metrics import roc_auc_score

def compute_score(model, data_loader, device="cpu"):
    model.eval()
    metric = roc_auc_score
    with torch.no_grad():
        prediction_all= torch.empty(0, device=device)
        labels_all= torch.empty(0, device=device)
        for i, (feats, labels) in enumerate(data_loader):
            feats=feats.to(device)
            labels=labels.to(device)
            prediction = model(feats).to(device)
            prediction = torch.sigmoid(prediction).to(device)
            prediction_all = torch.cat((prediction_all, prediction), 0)
```

```
            labels_all = torch.cat((labels_all, labels), 0)
        try:
            t = metric(labels_all.int().cpu(), prediction_all.cpu()).item()
        except ValueError:
            t = 0
    return t


def main(num_samples=10, max_num_epochs=10, gpus_per_trial=2):

    # define data directory here if you want to load data from files
    data_dir = os.path.abspath("./data")
    load_data(data_dir)

    # define the search space of hyperparameters
    config = {
        "act1 ": tune.choice(["relu","tanh","selu"]),
        "act2" : tune.choice(["relu","tanh","selu"]),
        "act3" : tune.choice(["relu","tanh","selu"]),
        "lr": tune.quniform(0.0005, 0.001, 0.0001),
        "batch_size": tune.choice([8, 16, 32]),
        "hidden_dim1" : tune.quniform(50, 200, 10),
        "hidden_dim2" : tune.quniform(50, 200, 10),
        "hidden_dim3" : tune.quniform(50, 200, 10),
    }

    # Optuna search algorithm
    from ray.tune.suggest.optuna import OptunaSearch
    from ray.tune.suggest import ConcurrencyLimiter
    search_alg = OptunaSearch(
        metric="loss", #or accuracy, etc.
        mode="max", #or max
        # seed = 42,
        # points_to_evaluate=[
        # {'lr': 0.0005, 'hidden_size': 100.0, 'readout1_out': 200.0, 'readout2_out': 180.0}
        # ],
        )
    search_alg = ConcurrencyLimiter(search_alg, max_concurrent=10)
```

```
scheduler = ASHAScheduler(
    metric ="loss",
    mode="max",
    max_t=max_num_epochs,
    reduction_factor=2,
    grace_period=4,
    brackets=5
    )

reporter = CLIReporter(
    # parameter_columns=["l1", "l2", "lr", "batch_size"],
    metric_columns=["loss", "training_iteration"]
    )

# wrap data loading and training for tuning using `partial`
# (note that there exist other methods for this purpose)
result = tune.run(
    partial(trainable_func, data_dir=data_dir, epochs=max_num_epochs),
    scheduler=scheduler,
    search_alg=search_alg,
    num_samples=num_samples,
    config=config,
    verbose=2,
    checkpoint_score_attr="loss",
    checkpoint_freq=0,
    keep_checkpoints_num=1,
    # checkpoint_at_end=True,
    # reuse_actors=reuse_actors_status,
    progress_reporter=reporter,
    resources_per_trial={"cpu": 2, "gpu": gpus_per_trial},
    stop={"training_iteration": max_num_epochs},
    )

best_trial = result.get_best_trial("loss", "max", "last")
print("Best trial config: {}".format(best_trial.config))
print("Best trial final validation score: {}".format(
    best_trial.last_result["loss"]))
```

```
    best_trained_model = Net(best_trial.config)
    device = "cpu"
    if torch.cuda.is_available():
        device = "cuda:0"
        if gpus_per_trial > 1:
            best_trained_model = nn.DataParallel(best_trained_model)
    best_trained_model.to(device)

    best_checkpoint_dir = best_trial.checkpoint.value
    model_state, optimizer_state = torch.load(os.path.join(
        best_checkpoint_dir, "checkpoint"))
    best_trained_model.load_state_dict(model_state)

    test_score_value = test_score(best_trial.config, best_trained_model, device)
    print("Best trial test set score: {}".format(test_score_value))


if __name__ == "__main__":
    # You can change the number of GPUs per trial here:
    main(num_samples=50, max_num_epochs=50, gpus_per_trial=0)
```

```
2022-07-18 08:23:14,965 INFO logger.py:630 -- pip install "ray[tune]" to see TensorBoard files.
2022-07-18 08:23:14,967 WARNING callback.py:106 -- The TensorboardX logger cannot be instantiated because either Ten
Streaming output truncated to the last 5000 lines.
| trainable_func_9d681022 | TERMINATED | 172.28.0.2:20234 | tanh    | tanh   | relu   |            16 |            80
| trainable_func_b3657efa | TERMINATED | 172.28.0.2:20193 | tanh    | tanh   | relu   |            16 |            80
| trainable_func_dea9cda6 | TERMINATED | 172.28.0.2:20193 | tanh    | tanh   | relu   |            16 |           130
| trainable_func_e0ecaae8 | TERMINATED | 172.28.0.2:20234 | relu    | tanh   | selu   |            32 |            80
| trainable_func_e0f59e00 | TERMINATED | 172.28.0.2:20234 | selu    | tanh   | selu   |            16 |           160
+-------------------------+------------+------------------+---------+--------+--------+---------------+-------------
... 2 more trials not shown (2 TERMINATED)


Trial trainable_func_b689714a reported epoch=15,loss=0.0,should_checkpoint=True with parameters={'act1 ': 'tanh', 'a
Trial trainable_func_c5837402 reported epoch=10,loss=0.0,should_checkpoint=True with parameters={'act1 ': 'tanh', 'a
== Status ==
Current time: 2022-07-18 08:30:01 (running for 00:06:46.10)
Memory usage on this node: 2.5/12.7 GiB
```

```
Using AsyncHyperBand: num_stopped=0
Bracket: Iter 32.000: 0.0 | Iter 16.000: 0.0 | Iter 8.000: 0.0 | Iter 4.000: 0.0
Bracket: Iter 32.000: 0.0 | Iter 16.000: 0.0 | Iter 8.000: 0.0
Bracket: Iter 32.000: 0.0 | Iter 16.000: 0.0
Bracket: Iter 32.000: 0.0
Bracket:
Resources requested: 4.0/4 CPUs, 0/0 GPUs, 0.0/7.4 GiB heap, 0.0/3.7 GiB objects
Result logdir: /root/ray_results/trainable_func_2022-07-18_08-23-14
Number of trials: 22/50 (1 PENDING, 2 RUNNING, 19 TERMINATED)
+-------------------------+------------+-----------------+--------+--------+--------+--------------+-------------
| Trial name              | status     | loc             | act1   | act2   | act3   |   batch_size |   hidden_dim1
|-------------------------+------------+-----------------+--------+--------+--------+--------------+-------------
| trainable_func_b689714a | RUNNING    | 172.28.0.2:20234 | tanh   | selu   | relu   |            8 |           170
| trainable_func_c5837402 | RUNNING    | 172.28.0.2:20193 | tanh   | selu   | relu   |            8 |           140
| trainable_func_c9437e7a | PENDING    |                 | tanh   | selu   | tanh   |           16 |           140
| trainable_func_0f3f3b36 | TERMINATED | 172.28.0.2:20234 | tanh   | tanh   | relu   |           32 |           180
| trainable_func_26cc2868 | TERMINATED | 172.28.0.2:20193 | relu   | tanh   | relu   |           32 |           120
| trainable_func_27e7a02e | TERMINATED | 172.28.0.2:20234 | tanh   | selu   | selu   |           32 |           120
| trainable_func_376088a4 | TERMINATED | 172.28.0.2:20193 | tanh   | selu   | tanh   |           32 |           150
| trainable_func_3871a39a | TERMINATED | 172.28.0.2:20234 | selu   | relu   | relu   |           16 |            90
| trainable_func_47523384 | TERMINATED | 172.28.0.2:20193 | relu   | relu   | tanh   |           32 |           190
| trainable_func_48c36094 | TERMINATED | 172.28.0.2:20193 | selu   | relu   | tanh   |           32 |           110
| trainable_func_58a8e74a | TERMINATED | 172.28.0.2:20234 | selu   | tanh   | relu   |           16 |           100
| trainable_func_6062cdc0 | TERMINATED | 172.28.0.2:20193 | relu   | tanh   | selu   |            8 |            50
| trainable_func_6b0b18fe | TERMINATED | 172.28.0.2:20234 | relu   | tanh   | selu   |            8 |            50
| trainable_func_77b8c3da | TERMINATED | 172.28.0.2:20193 | relu   | tanh   | selu   |            8 |            50
| trainable_func_91ef0570 | TERMINATED | 172.28.0.2:20234 | relu   | tanh   | selu   |           16 |            70
| trainable_func_9d681022 | TERMINATED | 172.28.0.2:20234 | tanh   | tanh   | relu   |           16 |            80
| trainable_func_b3657efa | TERMINATED | 172.28.0.2:20193 | tanh   | tanh   | relu   |           16 |            80
| trainable_func_dea9cda6 | TERMINATED | 172.28.0.2:20193 | tanh   | tanh   | relu   |           16 |           130
| trainable_func_e0ecaae8 | TERMINATED | 172.28.0.2:20234 | relu   | tanh   | selu   |           32 |            80
| trainable_func_e0f59e00 | TERMINATED | 172.28.0.2:20234 | selu   | tanh   | selu   |           16 |           160
+-------------------------+------------+-----------------+--------+--------+--------+--------------+-------------
... 2 more trials not shown (2 TERMINATED)


Trial trainable_func_b689714a reported epoch=20,loss=0.0,should_checkpoint=True with parameters={'act1 ': 'tanh', 'a
Trial trainable_func_c5837402 reported epoch=15,loss=0.0,should_checkpoint=True with parameters={'act1 ': 'tanh', 'a
== Status ==
Current time: 2022-07-18 08:30:06 (running for 00:06:51.68)
```