hyperparameter_tuning

November 17, 2023

1 Classification of a Signal that Produces Higgs Boson Particles and background signals

2 Convolutional Neural Network

2.0.1 Matthew Boyer and Jonah Goldfine

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import classification_report, accuracy_score, roc_curve,␣
 ↪auc, confusion_matrix
from sklearn.base import BaseEstimator,ClassifierMixin
from sklearn.decomposition import PCA
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset, random_split
from tqdm import tqdm
from skorch import NeuralNetBinaryClassifier
from skorch import callbacks as cb
import optuna
name_dtype=np.array([['class_label', np.float32], ['jet_1_b-tag', np.float64],
            ['jet_1_eta', np.float64], ['jet_1_phi', np.float64],
            ['jet_1_pt', np.float64], ['jet_2_b-tag', np.float64],
            ['jet_2_eta', np.float64], ['jet_2_phi', np.float64],
            ['jet_2_pt', np.float64], ['jet_3_b-tag', np.float64],
            ['jet_3_eta', np.float64], ['jet_3_phi', np.float64],
            ['jet_3_pt', np.float64], ['jet_4_b-tag', np.float64],
            ['jet_4_eta', np.float64], ['jet_4_phi', np.float64],
            ['jet_4_pt', np.float64], ['lepton_eta', np.float64],
            ['lepton_pT', np.float64], ['lepton_phi', np.float64],
            ['m_bb', np.float64], ['m_jj', np.float64],
            ['m_jjj', np.float64], ['m_jlv', np.float64],
```

```
            ['m_lv', np.float64], ['m_wbb', np.float64],
            ['m_wwbb', np.float64], ['missing_energy_magnitude', np.float64],
            ['missing_energy_phi', np.float64]])
fullData=pd.read_csv('HIGGS.csv',header=None,names=name_dtype[:,0])
unscaled_X=fullData.drop(['class_label'],axis=1)
scaler=StandardScaler()
full_X=pd.DataFrame(scaler.fit_transform(unscaled_X.values),index=unscaled_X.
 ↪index,columns=unscaled_X.columns)
full_y=fullData['class_label']
X_train_df,X_test_df,y_train_df,y_test_df=train_test_split(full_X,full_y,test_size=0.
 ↪2,random_state=0)
pca = PCA(n_components=0.95)  # retaining 95% of the variance
X_train_pca = pca.fit_transform(X_train_df)
X_test_pca = pca.transform(X_test_df)

# Checking the number of components selected and the amount of variance␣
 ↪explained
n_components = pca.n_components_
explained_variance = pca.explained_variance_ratio_.sum()

X_train=torch.tensor(X_train_pca).float()
X_test=torch.tensor(X_test_pca).float()
y_train=torch.tensor(y_train_df.values).float()
y_test=torch.tensor(y_test_df.values).float()
class DNN_NoDrop(nn.Module):
    def __init__(self, layer_sizes, activation):
        super(DNN_NoDrop, self).__init__()
        if layer_sizes == 'empty':
            layer_sizes = []
        else:
            layer_sizes = [int(size) for size in layer_sizes.split('_')]

        activation_functions = {'LeakyReLU': nn.LeakyReLU(), 'ReLU': nn.ReLU(),␣
 ↪'Tanh': nn.Tanh()}
        activation = activation_functions[activation]
        self.layers = nn.ModuleList()
        input_size = 23
        for hidden_size in layer_sizes:
            self.layers.append(nn.Linear(input_size, hidden_size))
            self.layers.append(activation)
            input_size = hidden_size

        self.layers.append(nn.Linear(input_size, 1))
        self.layers.append(nn.Sigmoid())

    def forward(self, x):
        for layer in self.layers:
```

```
            x = layer(x)
        return x.squeeze()
device = 'cuda' if torch.cuda.is_available() else 'cpu'
X_train, y_train = X_train.to(device), y_train.to(device)
dataset = TensorDataset(X_train, y_train)

# Split dataset into training and validation sets
train_size = int(0.8 * len(dataset))
val_size = len(dataset) - train_size
train_dataset, val_dataset = random_split(dataset, [train_size, val_size])
import time
from torch.cuda.amp import autocast, GradScaler

def objective(trial):
    start_time = time.time()
    layer_sizes = trial.suggest_categorical('layer_sizes', ['empty', '16',␣
 ↪'16_8', '64', '64_32', '64_32_16', '64_32_16_8', '128', '128_64',␣
 ↪'128_64_32', '128_64_32_16', '128_64_32_16_8'])
    activation = trial.suggest_categorical('activation', ['LeakyReLU', 'ReLU',␣
 ↪'Tanh'])
    max_epochs = trial.suggest_categorical('max_epochs', [10, 25, 50, 100, 250])
    batch_size = 704
    lr = trial.suggest_float('lr', 1e-5, 1e-1, log=True)

    model = DNN_NoDrop(layer_sizes=layer_sizes, activation=activation)
    criterion = nn.BCEWithLogitsLoss()
    optimizer = optim.Adam(model.parameters(), lr=lr)
    model.to(device)
    scaler = GradScaler()

    train_loader = DataLoader(train_dataset, batch_size=batch_size,␣
 ↪shuffle=True, num_workers=16)
    val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False,␣
 ↪num_workers=16)

    for epoch in range(max_epochs):
        model.train()
        for batch, (input, target) in enumerate(train_loader):
            input, target = input.to(device), target.to(device)
            optimizer.zero_grad()

            with autocast():
                output = model(input)
                loss = criterion(output, target)
            scaler.scale(loss).backward()
            scaler.step(optimizer)
            scaler.update()
```

```
    model.eval()
    val_loss = 0
    correct = 0
    with torch.no_grad(), autocast():
        for input, target in val_loader:
            input, target = input.to(device), target.to(device)
            output = model(input)
            val_loss += criterion(output, target).item()
            pred = torch.sigmoid(output).ge(0.5).view(-1)
            correct += pred.eq(target.view_as(pred)).sum().item()

    val_loss /= len(val_loader.dataset)
    accuracy = correct / len(val_loader.dataset)
    trial.report(accuracy, epoch)

    # Handle pruning based on the intermediate value
    if trial.should_prune():
        raise optuna.exceptions.TrialPruned()

    end_time = time.time()
    duration = end_time - start_time
    print(f"Trial {trial.number} took {duration:.2f} seconds.")

    return accuracy

study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=20)
```

```
c:\Python39\lib\site-packages\tqdm\auto.py:21: TqdmWarning: IProgress not found.
Please update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

[ ]:
```
best_trial = study.best_trial
best_params = best_trial.params

X_train, y_train = zip(*[data for data in train_dataset])
X_train, y_train = torch.stack(X_train), torch.tensor(y_train)
best_trial.fit(X_train, y_train)
```

```
Best trial parameters: {'layer_sizes': '112_56_28_14_7', 'activation':
'LeakyReLU', 'max_epochs': 250, 'batch_size': 200, 'lr': 0.00010022659726287869}
  epoch    accuracy    train_loss    valid_acc    valid_loss      dur
-------  ----------  ------------  -----------  ------------  -------
      1      0.6577        0.6135       0.6895
0.5854  33.3308
      2      0.7018        0.5711       0.7075
```

```
0.5629  33.2186
     3      0.7131        0.5548       0.7168
0.5498  33.4217
     4      0.7202        0.5449       0.7216
0.5423  33.1873
     5      0.7244        0.5386       0.7253
0.5371  33.1697
     6      0.7279        0.5335       0.7281
0.5327  33.2076
     7      0.7309        0.5290       0.7307
0.5290  33.4589
     8      0.7335        0.5251       0.7329
0.5257  33.0833
     9      0.7355        0.5218       0.7350
0.5229  33.2814
    10      0.7372        0.5190       0.7361
0.5206  33.3182
    11      0.7390        0.5166       0.7375
0.5185  33.0533
    12      0.7404        0.5145       0.7388
0.5168  33.0976
    13      0.7418        0.5126       0.7395
0.5153  33.3232
    14      0.7429        0.5109       0.7405
0.5138  33.0845
    15      0.7438        0.5094       0.7412
0.5125  33.3679
    16      0.7446        0.5081       0.7422
0.5114  33.2192
    17      0.7454        0.5069       0.7428
0.5105  33.3223
    18      0.7460        0.5059       0.7436
0.5096  33.0912
    19      0.7464        0.5049       0.7441
0.5088  33.4119
    20      0.7471        0.5040       0.7446
0.5080  33.1403
    21      0.7476        0.5032       0.7448
0.5073  33.5221
    22      0.7480        0.5024       0.7453
0.5066  32.9688
    23      0.7485        0.5017       0.7455
0.5062  33.3719
    24      0.7490        0.5011       0.7458
0.5056  33.4424
    25      0.7494        0.5005       0.7461
0.5052  33.2093
    26      0.7497        0.4999       0.7465
```

```
0.5047  33.3468
     27      0.7500          0.4993          0.7469
0.5041  33.4303
     28      0.7504          0.4988          0.7473
0.5038  33.2018
     29      0.7508          0.4983          0.7476
0.5033  33.5672
     30      0.7509          0.4979          0.7478
0.5030  33.3906
     31      0.7514          0.4974          0.7479
0.5028  33.2646
     32      0.7516          0.4970          0.7480
0.5025  33.2547
     33      0.7519          0.4967          0.7481
0.5024  33.6310
     34      0.7522          0.4963          0.7483
0.5023  34.8152
     35      0.7524          0.4959          0.7487
0.5019  33.9008
     36      0.7526          0.4956          0.7488
0.5017  34.2353
     37      0.7528          0.4952          0.7488
0.5015  33.4609
     38      0.7531          0.4949          0.7491
0.5013  33.3208
     39      0.7533          0.4946          0.7492
0.5011  33.4447
     40      0.7535          0.4943          0.7494
0.5009  33.3671
     41      0.7538          0.4940          0.7497
0.5008  33.7089
     42      0.7539          0.4937          0.7498
0.5004  33.5423
     43      0.7541          0.4934          0.7499
0.5003  33.4545
     44      0.7543          0.4932          0.7501
0.5001  33.3203
     45      0.7545          0.4929          0.7502
0.4999  33.5867
     46      0.7546          0.4927          0.7502
0.4997  33.8412
     47      0.7548          0.4924          0.7504
0.4995  33.6954
     48      0.7550          0.4922          0.7506
0.4992  33.7670
     49      0.7553          0.4919          0.7506
0.4991  33.6493
     50      0.7554          0.4917          0.7510
```

```
0.4988  33.8705
     51      0.7555        0.4915        0.7512
0.4986  33.7807
     52      0.7556        0.4913        0.7510
0.4984  33.8562
     53      0.7558        0.4910        0.7512
0.4983  33.8179
     54      0.7559        0.4908        0.7515
0.4982  33.9671
     55      0.7560        0.4906        0.7517
0.4978  34.7171
     56      0.7562        0.4904        0.7518
0.4977  34.0144
     57      0.7563        0.4902        0.7520
0.4976  34.0551
     58      0.7565        0.4900        0.7520
0.4974  33.2860
     59      0.7566        0.4898        0.7520
0.4974  33.2596
     60      0.7568        0.4896        0.7518
0.4973  33.3637
     61      0.7568        0.4895        0.7519
0.4971  33.9187
     62      0.7570        0.4893        0.7521
0.4971  33.5873
     63      0.7572        0.4891        0.7521
0.4970  34.0771
     64      0.7573        0.4889        0.7521
0.4969  33.6861
     65      0.7573        0.4887        0.7522
0.4967  33.5544
     66      0.7574        0.4886        0.7524
0.4966  33.4595
     67      0.7575        0.4884        0.7523
0.4964  33.7067
     68      0.7576        0.4882        0.7523
0.4964  33.6068
     69      0.7577        0.4881        0.7523
0.4963  33.7507
     70      0.7578        0.4879        0.7523
0.4961  33.4291
     71      0.7579        0.4877        0.7524        0.4961
33.5620
     72      0.7581        0.4876        0.7527
0.4960  32.8986
     73      0.7582        0.4874        0.7525
0.4959  32.9185
     74      0.7583        0.4873        0.7526
```

```
0.4959  33.2974
     75      0.7584         0.4872        0.7528
0.4958  33.1602
     76      0.7585         0.4870        0.7528
0.4957  32.7573
     77      0.7587         0.4869        0.7529
0.4957  33.2983
     78      0.7587         0.4868        0.7527
0.4957  33.6762
     79      0.7589         0.4866        0.7530
0.4955  33.0554
     80      0.7590         0.4865        0.7529
0.4954  33.4038
     81      0.7590         0.4864        0.7528
0.4953  33.0788
     82      0.7591         0.4862        0.7529
0.4952  33.1924
     83      0.7592         0.4861        0.7530
0.4951  33.2223
     84      0.7592         0.4860        0.7530         0.4952
33.1058
     85      0.7594         0.4858        0.7531
0.4949  33.0031
     86      0.7594         0.4857        0.7530         0.4950   33.1707
     87      0.7595         0.4856        0.7534
0.4948  33.2052
     88      0.7596         0.4854        0.7534
0.4947  33.2935
     89      0.7597         0.4853        0.7534
0.4946  33.1718
     90      0.7598         0.4852        0.7535
0.4944  32.9304
     91      0.7598         0.4851        0.7533
0.4944  32.9716
     92      0.7600         0.4849        0.7535
0.4943  33.4579
     93      0.7601         0.4848        0.7535
0.4943  32.8633
     94      0.7601         0.4847        0.7534
0.4943  33.0351
     95      0.7603         0.4846        0.7535
0.4942  33.0710
     96      0.7603         0.4845        0.7537
0.4941  33.2753
     97      0.7604         0.4843        0.7537
0.4940  32.7136
     98      0.7604         0.4842        0.7538
0.4939  33.3731
```

```
     99      0.7606        0.4841       0.7538
0.4938  32.7607
    100      0.7606        0.4840       0.7539
0.4939  33.3309
    101      0.7607        0.4839       0.7539
0.4937  33.0932
    102      0.7607        0.4838       0.7541
0.4937  33.2370
    103      0.7609        0.4837       0.7542
0.4936  33.5429
    104      0.7609        0.4836       0.7542       0.4935
33.3273
    105      0.7609        0.4835       0.7542
0.4934  33.0516
    106      0.7611        0.4834       0.7541       0.4934
33.6356
    107      0.7610        0.4833       0.7541       0.4934  32.9384
    108      0.7611        0.4832       0.7540
0.4932  33.1146
    109      0.7612        0.4831       0.7543
0.4930  33.2682
    110      0.7612        0.4830       0.7543
0.4930  33.1630
    111      0.7613        0.4829       0.7543
0.4929  33.2248
    112      0.7613        0.4828       0.7542
0.4928  33.3828
    113      0.7614        0.4827       0.7543       0.4928
33.0257
    114      0.7615        0.4826       0.7544
0.4927  33.2054
    115      0.7616        0.4825       0.7545
0.4927  33.1183
    116      0.7617        0.4824       0.7545
0.4926  32.9711
    117      0.7617        0.4823       0.7547
0.4926  33.3141
    118      0.7617        0.4822       0.7548
0.4924  33.2509
    119      0.7618        0.4821       0.7548       0.4926
33.1198
    120      0.7619        0.4820       0.7546       0.4925
33.2495
    121      0.7619        0.4820       0.7547       0.4924
33.1087
    122      0.7620        0.4819       0.7548
0.4924  32.8823
    123      0.7620        0.4818       0.7547       0.4925
```

```
33.2705
    124      0.7621        0.4817       0.7549
0.4923  33.1206
    125      0.7621        0.4816       0.7548        0.4923
33.3332
    126      0.7621        0.4815       0.7549
0.4922  33.1510
    127      0.7622        0.4814       0.7551
0.4921  33.5232
    128      0.7623        0.4814       0.7552
0.4921  32.9239
    129      0.7623        0.4813       0.7552        0.4921
33.5846
    130      0.7625        0.4812       0.7554
0.4919  33.0912
    131      0.7625        0.4811       0.7554
0.4919  33.1003
    132      0.7626        0.4811       0.7556
0.4918  33.0466
    133      0.7626        0.4810       0.7554        0.4918
33.2930
    134      0.7627        0.4809       0.7558
0.4916  32.9677
    135      0.7627        0.4808       0.7557
0.4916  33.5999
    136      0.7626        0.4808       0.7556        0.4917  33.0545
    137      0.7627        0.4807       0.7557
0.4915  33.3725
    138      0.7628        0.4806       0.7558
0.4914  33.4217
    139      0.7628        0.4805       0.7558        0.4915
33.1685
    140      0.7629        0.4805       0.7559
0.4915  33.1362
    141      0.7629        0.4804       0.7560
0.4914  33.5220
    142      0.7630        0.4803       0.7558
0.4914  33.0402
    143      0.7630        0.4803       0.7559
0.4913  33.6703
    144      0.7630        0.4802       0.7559        0.4913
33.4445
    145      0.7631        0.4801       0.7560
0.4913  33.4219
    146      0.7631        0.4801       0.7560
0.4912  33.3156
    147      0.7632        0.4800       0.7559
0.4911  33.1754
```

```
    148      0.7632        0.4799       0.7561
0.4911  33.4129
    149      0.7633        0.4799       0.7560        0.4912
33.5381
    150      0.7633        0.4798       0.7562
0.4911  33.1344
    151      0.7633        0.4797       0.7562
0.4910  33.2029
    152      0.7633        0.4797       0.7561        0.4910
33.0906
    153      0.7634        0.4796       0.7561
0.4910  33.1365
    154      0.7635        0.4796       0.7560
0.4910  33.5121
    155      0.7635        0.4795       0.7561        0.4910  33.3896
    156      0.7635        0.4795       0.7562        0.4910
33.0809
    157      0.7636        0.4794       0.7561        0.4910
33.2163
    158      0.7636        0.4794       0.7561        0.4910
33.3013
    159      0.7636        0.4793       0.7561        0.4909
33.0286
    160      0.7637        0.4792       0.7561
0.4908  33.5756
    161      0.7637        0.4792       0.7562
0.4909  33.0609
    162      0.7638        0.4791       0.7561
0.4908  34.2981
    163      0.7639        0.4791       0.7562
0.4907  34.2185
    164      0.7638        0.4790       0.7562        0.4908  34.2539
    165      0.7639        0.4790       0.7562        0.4908
34.7693
    166      0.7639        0.4789       0.7561        0.4908
36.0364
    167      0.7640        0.4789       0.7562        0.4908
36.1572
    168      0.7641        0.4788       0.7560        0.4908
36.1592
    169      0.7641        0.4788       0.7561
0.4907  35.9536
    170      0.7640        0.4787       0.7559        0.4908  35.9349
    171      0.7641        0.4786       0.7561        0.4907
36.0666
    172      0.7642        0.4786       0.7560        0.4907
35.9308
    173      0.7642        0.4785       0.7561
```

```
0.4907  36.2332
    174      0.7643        0.4785        0.7560
0.4906  36.2272
    175      0.7643        0.4784        0.7561        0.4906  35.8607
    176      0.7643        0.4784        0.7562        0.4906
36.3572
    177      0.7644        0.4783        0.7561
0.4906  36.0230
    178      0.7644        0.4783        0.7561        0.4905
36.1796
    179      0.7644        0.4782        0.7562        0.4906
36.1551
    180      0.7644        0.4782        0.7562        0.4905  35.8282
    181      0.7644        0.4781        0.7562        0.4906
36.2939
    182      0.7645        0.4781        0.7561
0.4905  36.0766
    183      0.7645        0.4781        0.7562        0.4905
36.0645
    184      0.7645        0.4780        0.7561
0.4903  36.1515
    185      0.7645        0.4780        0.7561        0.4904
35.7540
    186      0.7646        0.4779        0.7562
0.4903  36.0516
    187      0.7646        0.4779        0.7563        0.4903
36.1428
    188      0.7646        0.4778        0.7563
0.4903  35.7642
    189      0.7647        0.4778        0.7561        0.4903
36.3163
    190      0.7648        0.4778        0.7562        0.4903
36.0699
    191      0.7648        0.4777        0.7561        0.4903  36.0346
    192      0.7648        0.4777        0.7562        0.4904
36.2238
    193      0.7648        0.4776        0.7562        0.4904
35.9624
    194      0.7649        0.4776        0.7561        0.4905
36.2237
    195      0.7648        0.4775        0.7561        0.4905  36.0540
    196      0.7648        0.4775        0.7560        0.4905  35.8101
    197      0.7648        0.4775        0.7563        0.4904  36.1640
    198      0.7649        0.4774        0.7562        0.4904
36.0417
    199      0.7649        0.4774        0.7562        0.4905
36.1296
    200      0.7650        0.4773        0.7562        0.4904
```

```
36.2157
    201      0.7649        0.4773        0.7562        0.4903  35.7578
    202      0.7650        0.4773        0.7563        0.4903
36.1990
    203      0.7651        0.4772        0.7562        0.4903
36.2754
    204      0.7651        0.4772        0.7562        0.4904
36.0889
    205      0.7650        0.4771        0.7561        0.4904  36.1859
    206      0.7650        0.4771        0.7562        0.4904  35.9681
    207      0.7651        0.4771        0.7561        0.4904
36.1567
    208      0.7652        0.4770        0.7561        0.4904
36.4758
    209      0.7652        0.4770        0.7562        0.4903
35.7312
    210      0.7652        0.4770        0.7562        0.4904  36.1003
    211      0.7652        0.4769        0.7563        0.4903
35.9962
    212      0.7651        0.4769        0.7563
0.4902  36.0532
    213      0.7652        0.4769        0.7562        0.4903
36.3398
    214      0.7653        0.4768        0.7562        0.4903
35.8434
    215      0.7654        0.4768        0.7562        0.4903
36.0920
    216      0.7654        0.4768        0.7563        0.4903
36.3955
    217      0.7654        0.4767        0.7562        0.4903
34.9830
    218      0.7655        0.4767        0.7562        0.4904
35.1405
    219      0.7654        0.4767        0.7563        0.4903  34.1841
    220      0.7654        0.4766        0.7562        0.4903  33.9491
    221      0.7655        0.4766        0.7563
0.4902  34.6490
    222      0.7656        0.4765        0.7563
0.4902  34.6645
    223      0.7655        0.4765        0.7563        0.4902  34.7960
    224      0.7656        0.4765        0.7564
0.4902  34.8672
    225      0.7656        0.4764        0.7562        0.4902  34.8064
    226      0.7655        0.4764        0.7562        0.4902  34.7409
    227      0.7656        0.4764        0.7562        0.4903
34.7281
    228      0.7656        0.4764        0.7562        0.4902
34.6703
```

```
    229      0.7657        0.4763       0.7562        0.4902
34.6803
    230      0.7656        0.4763       0.7563        0.4902  34.2622
    231      0.7657        0.4763       0.7563        0.4902  34.9269
    232      0.7657        0.4762       0.7564
0.4902  33.9108
    233      0.7656        0.4762       0.7564
0.4901  34.8315
    234      0.7657        0.4762       0.7565
0.4901  34.8357
    235      0.7657        0.4761       0.7563        0.4902
34.4304
    236      0.7657        0.4761       0.7566
0.4900  34.5308
    237      0.7657        0.4761       0.7566
0.4899  34.7119
    238      0.7657        0.4760       0.7566
0.4899  34.9732
    239      0.7657        0.4760       0.7565        0.4900  34.6083
    240      0.7658        0.4760       0.7564        0.4900
34.7902
    241      0.7657        0.4760       0.7567        0.4900
34.6877
    242      0.7659        0.4759       0.7567
0.4899  34.5644
    243      0.7659        0.4759       0.7567
0.4899  34.0632
    244      0.7660        0.4759       0.7566        0.4899
33.8716
    245      0.7659        0.4758       0.7568        0.4899
34.5290
    246      0.7659        0.4758       0.7569
0.4898  35.1448
    247      0.7659        0.4758       0.7567        0.4898
34.6044
    248      0.7659        0.4758       0.7567        0.4897
34.4777
    249      0.7660        0.4757       0.7569
0.4897  34.8220
    250      0.7661        0.4757       0.7570
0.4897  34.9404
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
c:\Users\matth\Desktop\COMP 4531\final.ipynb Cell 12 line <cell line: 23>()
      <a href='vscode-notebook-cell:/c%3A/Users/matth/Desktop/COMP%204531/final.
 ↪ipynb#X24sZmlsZQ%3D%3D?line=20'>21</a> X_train, y_train = torch.
 ↪stack(X_train), torch.tensor(y_train)
```

```
    <a href='vscode-notebook-cell:/c%3A/Users/matth/Desktop/COMP%204531/final.
 ↪ipynb#X24sZmlsZQ%3D%3D?line=21'>22</a> best_model.fit(X_train, y_train)
---> <a href='vscode-notebook-cell:/c%3A/Users/matth/Desktop/COMP%204531/final.
 ↪ipynb#X24sZmlsZQ%3D%3D?line=22'>23</a> best_model.save_params(f='best_model.
 ↪pth')

File c:\Python39\lib\site-packages\skorch\net.py:2488, in NeuralNet.
 ↪save_params(self, f_params, f_optimizer, f_criterion, f_history,␣
 ↪use_safetensors, **kwargs)
   2485     def _save_state_dict(state_dict, f_name):
   2486         torch.save(module.state_dict(), f_name)
-> 2488 kwargs_module, kwargs_other = _check_f_arguments(
   2489     'save_params',
   2490     f_params=f_params,
   2491     f_optimizer=f_optimizer,
   2492     f_criterion=f_criterion,
   2493     f_history=f_history,
   2494     **kwargs)
   2496 if not kwargs_module and not kwargs_other:
   2497     if self.verbose:

File c:\Python39\lib\site-packages\skorch\utils.py:755, in␣
 ↪_check_f_arguments(caller_name, **kwargs)
    753 for key, val in kwargs.items():
    754     if not key.startswith('f_'):
--> 755         raise TypeError(
    756             "{name} got an unexpected argument '{key}', did you mean␣
 ↪'f_{key}'?"
    757             .format(name=caller_name, key=key))
    759     if val is None:
    760         continue

TypeError: save_params got an unexpected argument 'f', did you mean 'f_f'?
```

[ ]:
```
# Above error was fixed below, didn't want to re-run code above as it to 20␣
 ↪hours.
import pickle

with open("study.pkl", "wb") as f:
    pickle.dump(study, f)
#best_model.save_params(f='best_model.pth')
```