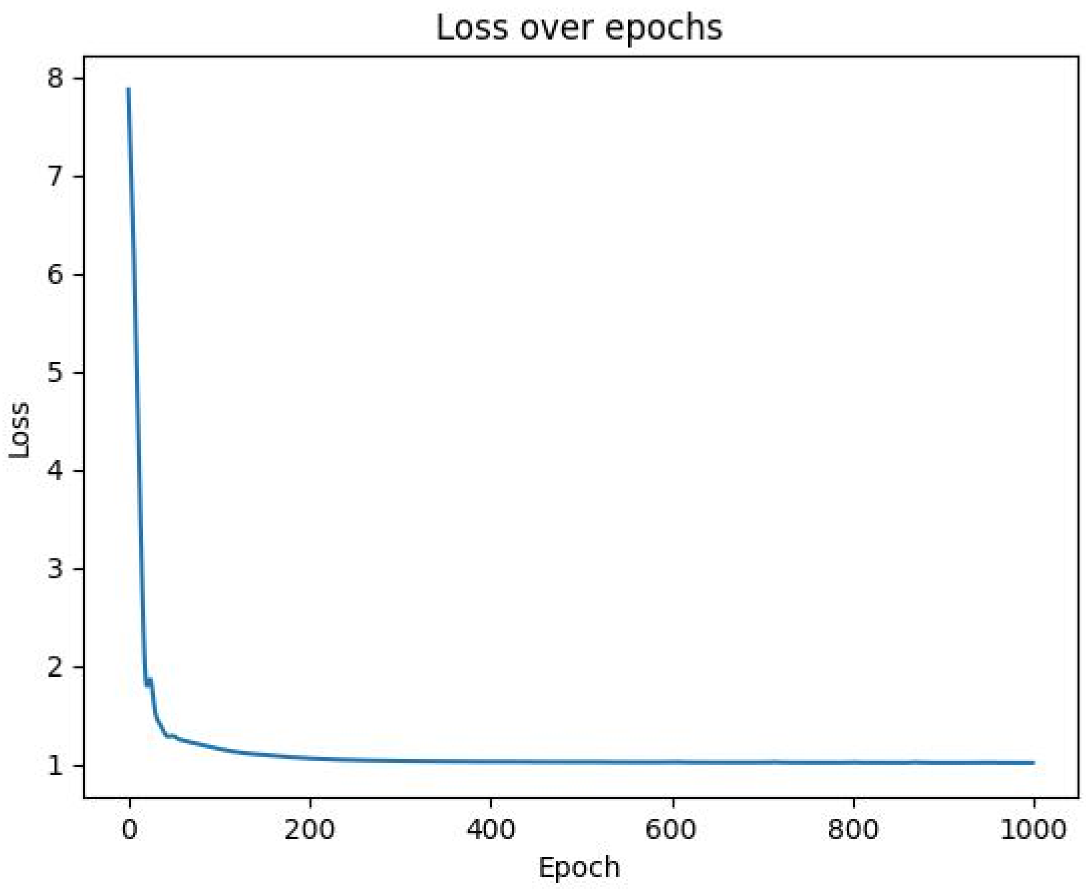
In [ ]:

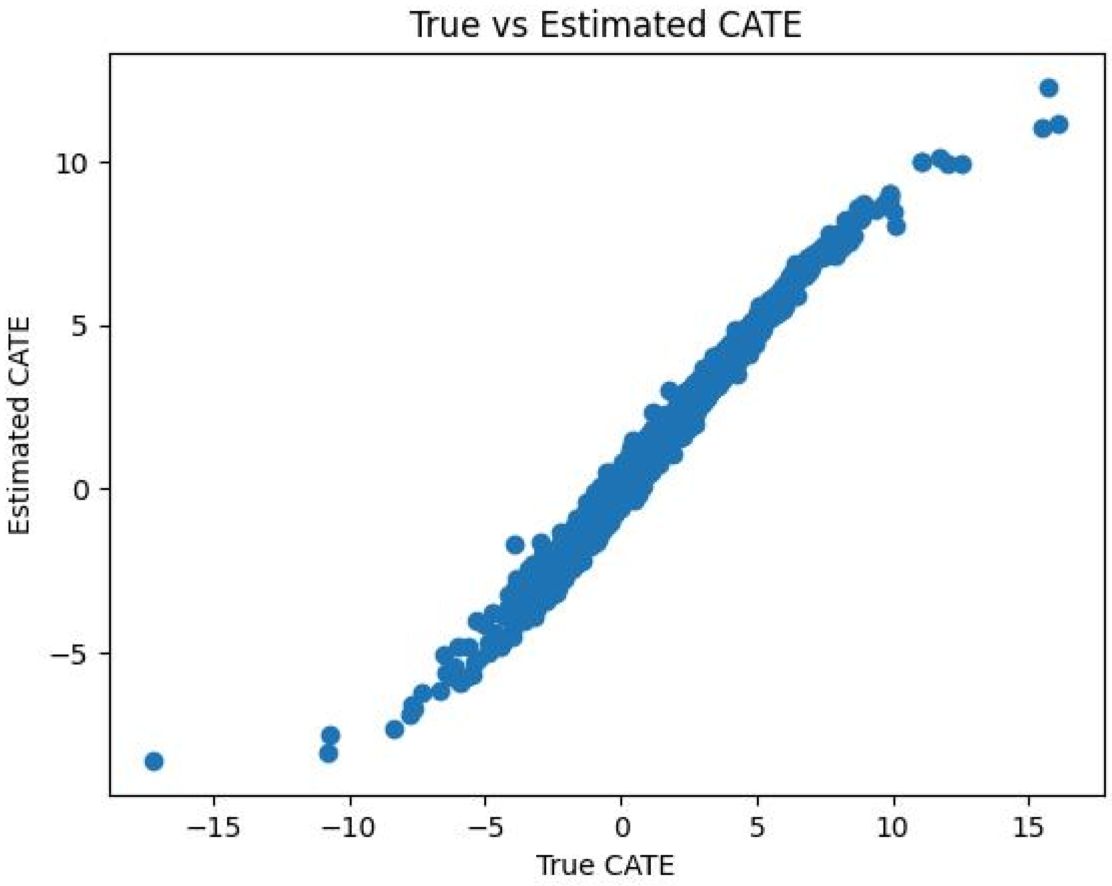
```python
import random
import numpy as np
import torch
from torch import nn
from torch.nn import functional as F
from torch.utils.data import random_split
import matplotlib.pyplot as plt
from tqdm import tqdm
from causal_dnn import CausalDNN
from linear_dnn import LinearDNN
from helper_utils import *

if __name__ == "__main__":
    # Set the seed for reproducibility
    torch.manual_seed(12345)
    # Example usage:
    n = 15000
    d_in = 5
    d_out = 1
    d_param = 2
    dim_z = 1
    h_arch = [20, 20]
    X = torch.randn(n, d_in)
    #  True CATE
    #  beta(x) = 2 - x_2 + 0.25 * x_1^3
    trub = 2-X[:,1] + .25*torch.pow(X[:,0], 3)
    # True Baseline
    # alpha(x) = 0.2 * x_1 - 1.3 * x_2 - 0.5 * x_3
    trua = X[:,0]*0.2 - X[:,1]*1.3 - X[:,2]*.5
    # Treatment (constant propensity)
    z = (torch.rand(n, dim_z)>.5)
    zalt = (torch.rand(n, dim_z)>.5) # Alternative treatment
    zopt = trub > 0 # 'Optimal' treatment
    # Reshape variables
    trub = torch.reshape(trub,(n,1))
    trua = torch.reshape(trua,(n,1))
    z = torch.reshape(z,(n,1))
    zalt = torch.reshape(zalt,(n,1))
    zopt = torch.reshape(zopt,(n,1))
    # Outcomes are linear in treatments
    y = trua + torch.mul(trub,z) + torch.randn(n,1)
    # Compute uplift in average outcome for alternative treatments
    ydiffa = (torch.mul(trub,zalt) - torch.mul(trub,z))
    ydiffb = (torch.mul(trub,zopt) - torch.mul(trub,z))
    # Collect data
    dat = {"X": X, "y": y, "z": z}

    # Estimate CATEs over Entire Dataset
    model = CausalDNN(num_output=d_param, num_input=d_in, hidden_arch=h_arch, lr=0.
    loss_values = model.train(X, y, z, epochs=1000, tol = 1e-3)

    het_alpha = model.alpha_vec.detach().numpy()
    het_beta = model.beta_vec.detach().numpy()
```

```
# Generate the graph
plt.plot(loss_values)
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Loss over epochs')
plt.show()

# Generate comparison graph
plt.scatter(trub, het_beta)
plt.xlabel('True CATE')
plt.ylabel('Estimated CATE')
plt.title('True vs Estimated CATE')
plt.show()


# Split data for cross-fitting
split1 = int(n/3)
split2 = int(n/3)
split3 = n-split1-split2
samp1, samp2, samp3 = random_split(torch.arange(n), (split1,split2,split3))
dat1 = {"X": X[samp1.indices,:], "y": y[samp1.indices,:], "z": z[samp1.indices,
dat2 = {"X": X[samp2.indices,:], "y": y[samp2.indices,:], "z": z[samp2.indices,
dat3 = {"X": X[samp3.indices,:], "y": y[samp3.indices,:], "z": z[samp3.indices,

# Create models
model1 = CausalDNN(num_output=d_param, num_input=d_in, hidden_arch=h_arch, lr=0
model2 = CausalDNN(num_output=d_param, num_input=d_in, hidden_arch=h_arch, lr=0
model3 = CausalDNN(num_output=d_param, num_input=d_in, hidden_arch=h_arch, lr=0

# Train models
model1.train(dat1["X"], dat1["y"], dat1["z"], epochs=1000, tol = 1e-3)
model2.train(dat2["X"], dat2["y"], dat2["z"], epochs=1000, tol = 1e-3)
model3.train(dat3["X"], dat3["y"], dat3["z"], epochs=1000, tol = 1e-3)

# Make Lambda
lproj1 = make_lam(dat1, model3)
lproj2 = make_lam(dat2, model1)
lproj3 = make_lam(dat3, model2)

# Define statistic
H_func=lambda x,y: y[:,1]

# Compute influence functions
if1 = proc_res(dat1, model2, lproj3, H_func)
if2 = proc_res(dat2, model3, lproj1, H_func)
if3 = proc_res(dat3, model1, lproj2, H_func)

# Compute ATE and Confidence Intervals
ate_beta = torch.cat((if1, if2, if3), dim=0).mean(dim=0)
ate_se = ((1/3)*(if1.var() + if2.var() + if3.var())/n).sqrt()

# Report results
print('\n')
print(f'Estimated ATE: {ate_beta.item():.3f}')
print(f'95% CI: [{ate_beta.item()-1.96*ate_se.item():.3f}, {ate_beta.item()+1.9
print(f'True ATE: {trub.mean().item():.3f}')
```

```python
# Define statistic for alternative treatment
def C_func0(x,b):
    new_target = x['zalt'].view(-1,1)
    new_outcome = new_target*(b[:,1].view(-1,1))
    prev_outcome = x['z']*b[:,1].view(-1,1)
    return new_outcome - prev_outcome

# Compute influence functions for alternative treatment
if1a = proc_res(dat1, model2, lproj3, C_func0)
if2a = proc_res(dat2, model3, lproj1, C_func0)
if3a = proc_res(dat3, model1, lproj2, C_func0)

# Compute ATE and Confidence Intervals for alternative treatment
uplift_a_beta = torch.cat((if1a, if2a, if3a), dim=0).mean(dim=0)
uplift_a_se = ((1/3)*(if1a.var() + if2a.var() + if3a.var())/n).sqrt()

# Report results
print('\n')
print(f'Estimated Uplift for Alternative Treatment: {uplift_a_beta.item():.3f}'
print(f'95% CI: [{uplift_a_beta.item()-1.96*uplift_a_se.item():.3f}, {uplift_a_
print(f'True Uplift for Alternative Treatment: {ydiffa.mean().item():.3f}')

# Define statistic for another alternative treatment
def C_func1(x,b):
    new_target = (b[:,1]>0).view(-1,1)
    new_outcome = new_target*(b[:,1].view(-1,1))
    prev_outcome = x['z']*b[:,1].view(-1,1)
    return new_outcome - prev_outcome

# Compute influence functions for another alternative treatment
if1b = proc_res(dat1, model2, lproj3, C_func1)
if2b = proc_res(dat2, model3, lproj1, C_func1)
if3b = proc_res(dat3, model1, lproj2, C_func1)

# Compute ATE and Confidence Intervals for another alternative treatment
uplift_b_beta = torch.cat((if1b, if2b, if3b), dim=0).mean(dim=0)
uplift_b_se = ((1/3)*(if1b.var() + if2b.var() + if3b.var())/n).sqrt()

# Report results
print('\n')
print(f'Estimated Uplift for Optimal Treatment: {uplift_b_beta.item():.3f}')
print(f'95% CI: [{uplift_b_beta.item()-1.96*uplift_b_se.item():.3f}, {uplift_b_
print(f'True Uplift for Optimal Treatment: {ydiffb.mean().item():.3f}')
```

```
100%|██████████| 1000/1000 [00:02<00:00, 351.19it/s]
```


Loss over epochs
Loss
Epoch
8
7
6
5
4
3
2
1
0
200
400
600
800
1000


True vs Estimated CATE
Estimated CATE
True CATE
10
5
0
−5
−15
−10
−5
0
5
10
15

```
100%|██████████| 1000/1000 [00:01<00:00, 635.54it/s]
100%|██████████| 1000/1000 [00:01<00:00, 669.44it/s]
100%|██████████| 1000/1000 [00:01<00:00, 671.52it/s]
 32%|███▎      | 65/200 [00:00<00:00, 872.28it/s]
Training stopped at epoch 66, Loss: 0.0009963539196178317
100%|██████████| 200/200 [00:00<00:00, 885.93it/s]
100%|██████████| 200/200 [00:00<00:00, 964.31it/s]
100%|██████████| 200/200 [00:00<00:00, 931.93it/s]
 47%|████▋     | 94/200 [00:00<00:00, 899.80it/s]
Training stopped at epoch 95, Loss: 0.0009982577757909894
100%|██████████| 200/200 [00:00<00:00, 817.42it/s]
100%|██████████| 200/200 [00:00<00:00, 826.34it/s]
100%|██████████| 200/200 [00:00<00:00, 861.91it/s]
 80%|████████  | 160/200 [00:00<00:00, 893.76it/s]
Training stopped at epoch 161, Loss: 0.0009843397419899702
100%|██████████| 200/200 [00:00<00:00, 951.79it/s]
100%|██████████| 200/200 [00:00<00:00, 913.06it/s]
100%|██████████| 200/200 [00:00<00:00, 918.76it/s]

Estimated ATE: 2.049
95% CI: [1.943, 2.154]
True ATE: 2.000


Estimated Uplift for Alternative Treatment: -0.024
95% CI: [-0.081, 0.033]
True Uplift for Alternative Treatment: -0.047


Estimated Uplift for Optimal Treatment: 1.078
95% CI: [0.986, 1.171]
True Uplift for Optimal Treatment: 1.036
```