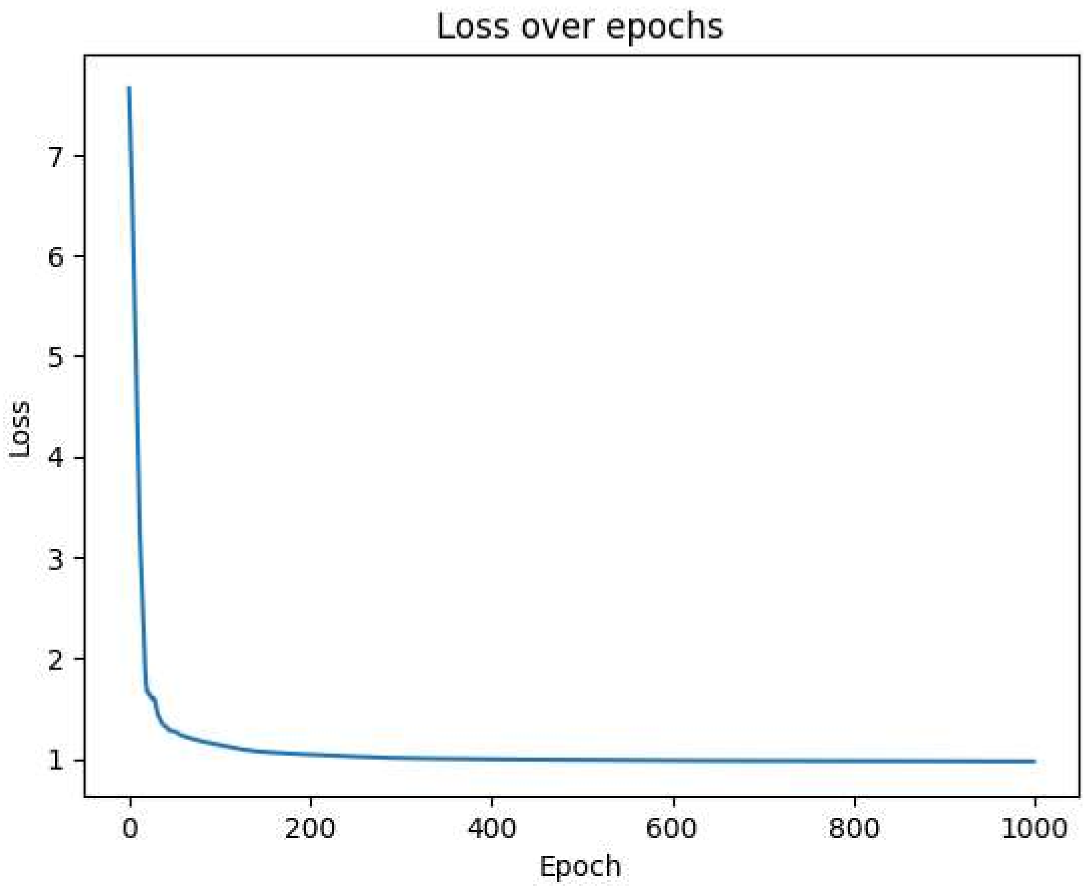
Connected to Python 3.11.7

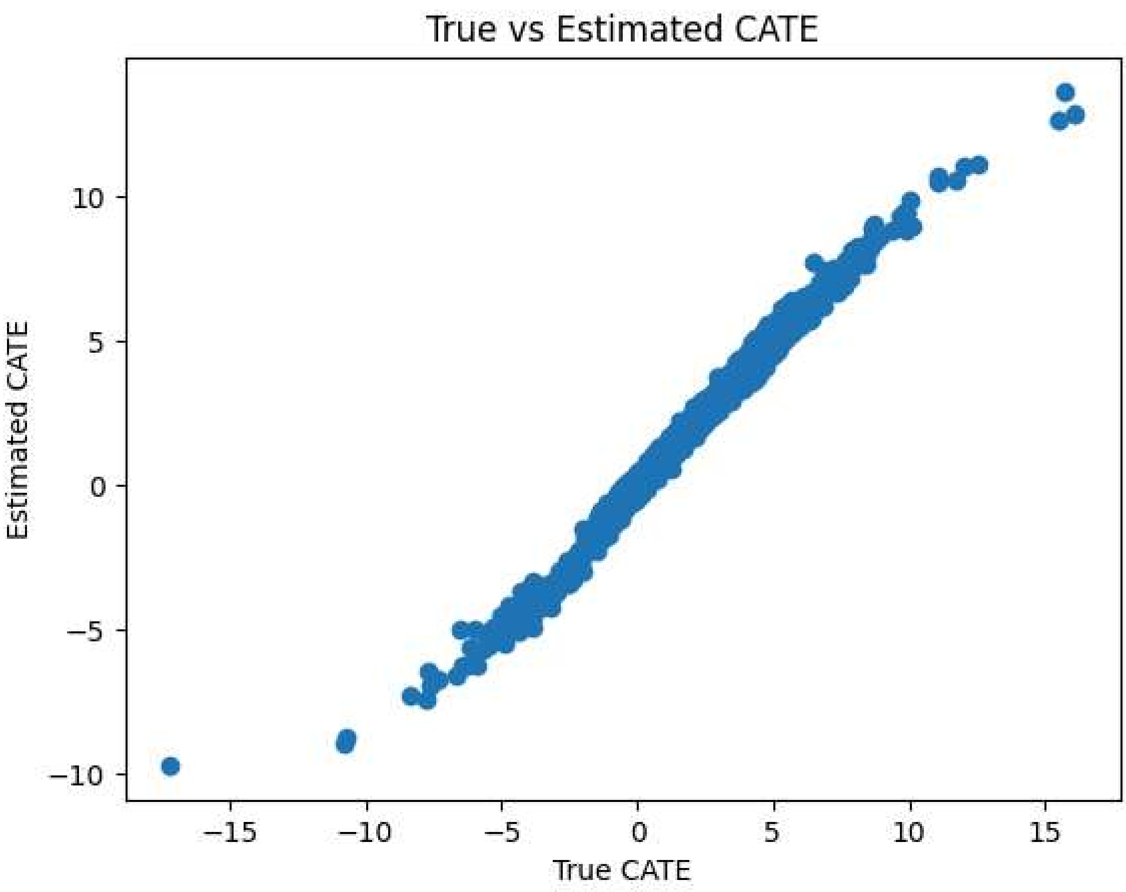
In [ ]:
```python
import random
import numpy as np
import torch
from torch import nn
from torch.nn import functional as F
from torch.utils.data import random_split
import matplotlib.pyplot as plt
from tqdm import tqdm
from causal_dnn import CausalDNN
from linear_dnn import LinearDNN
from helper_utils 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)
    # Reshape variables
    trub = torch.reshape(trub,(n,1))
    trua = torch.reshape(trua,(n,1))
    z = torch.reshape(z,(n,1))
    # Outcomes are linear in treatments
    y = trua + torch.mul(trub,z) + torch.randn(n,1)
    # 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')
```

```python
    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.cat([X, y, z],dim=1), (split1,split2
    # dat1 = {"X": samp1.dataset[:,0:d_in], "y": samp1.dataset[:,d_in:d_in+d_out],
    # dat2 = {"X": samp2.dataset[:,0:d_in], "y": samp2.dataset[:,d_in:d_in+d_out],
    # dat3 = {"X": samp3.dataset[:,0:d_in], "y": samp3.dataset[:,d_in:d_in+d_out],
    samp1, samp2, samp3 = random_split(torch.cat([y],dim=1), (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()/split1 + if2.var()/split2 + if3.var()/split3)).sqrt(

    # Report results
    print(f'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}')
```

```
100%|██████████| 1000/1000 [00:04<00:00, 229.37it/s]
```



Loss over epochs
Loss
Epoch

True vs Estimated CATE
Estimated CATE
True CATE

```
100%|██████████| 1000/1000 [00:02<00:00, 453.64it/s]
100%|██████████| 1000/1000 [00:02<00:00, 449.90it/s]
100%|██████████| 1000/1000 [00:02<00:00, 449.64it/s]
 93%|█████████▎| 186/200 [00:00<00:00, 569.25it/s]
Training stopped at epoch 187, Loss: 0.0009968930389732122
100%|██████████| 200/200 [00:00<00:00, 615.50it/s]
100%|██████████| 200/200 [00:00<00:00, 624.09it/s]
100%|██████████| 200/200 [00:00<00:00, 624.56it/s]
 56%|█████▋    | 113/200 [00:00<00:00, 560.38it/s]
Training stopped at epoch 114, Loss: 0.0009835874661803246
100%|██████████| 200/200 [00:00<00:00, 574.47it/s]
100%|██████████| 200/200 [00:00<00:00, 603.41it/s]
100%|██████████| 200/200 [00:00<00:00, 611.88it/s]
 58%|█████▊    | 117/200 [00:00<00:00, 657.45it/s]
Training stopped at epoch 118, Loss: 0.0009990849066525698
100%|██████████| 200/200 [00:00<00:00, 635.45it/s]
100%|██████████| 200/200 [00:00<00:00, 619.04it/s]
100%|██████████| 200/200 [00:00<00:00, 607.53it/s]
ATE: 1.980
95% CI: [1.854, 2.105]
True ATE: 2.000
```