In [ ]:
```
import itertools

from tqdm import tqdm
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

import torch
import torch.nn as nn
```

Question 1

(a)

In [ ]:
```
import pickle
from torch.utils.data import Dataset, DataLoader


class MnistDataset():
    def __init__(self, X, y, transform_X=lambda x: x):
        assert X.shape[0] == y.shape[0]
        self.X = transform_X(X)
        self.y = y

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        return self.X[idx], self.y[idx]


def load_mnist(path):
    with open(path, 'rb') as f:
        train_data, test_data = pickle.load(f)

    X_train = torch.tensor(train_data[0], dtype=torch.float).unsqueeze(1)
    y_train = torch.tensor(train_data[1], dtype=torch.long)
    X_test = torch.tensor(test_data[0], dtype=torch.float).unsqueeze(1)
    y_test = torch.tensor(test_data[1], dtype=torch.long)
    return X_train, y_train, X_test, y_test
```

In [ ]:
```
# load data and normalize by dividing the maximum value
X_train, y_train, X_test, y_test = load_mnist('mnist.pkl')
X_train /= torch.max(X_train)
X_test /= torch.max(X_test)
```

In [ ]:
```
# further split the train to train/validation with 80/20
X_further_train, X_val, y_further_train, y_val = train_test_split(X_train, y

# use MnistDataset class to handle the data
```

```
train_data = MnistDataset(X_further_train, y_further_train)
val_data = MnistDataset(X_val, y_val)
test_data = MnistDataset(X_test, y_test)
```

(b)

In [ ]:

```
class VAE(nn.Module):
    def __init__(self, in_channels=1, z_dim=32):
        super().__init__()
        self.encoder = nn.Sequential(
            # conv1, input_channel -> 4
            nn.Conv2d(in_channels, 4, kernel_size=4, padding=1, stride=2),
            nn.ReLU(), # relu
            # conv2, channel 4 -> 8
            nn.Conv2d(4, 8, kernel_size=4, padding=1, stride=2),
            nn.ReLU(), # relu
            nn.Conv2d(8, 16, kernel_size=4, padding=1, stride=2),  # conv3,
            nn.ReLU(), # relu
            nn.Conv2d(16, 32, kernel_size=4, padding=1, stride=2), # conv4,
            nn.ReLU(), # relu
            nn.Flatten(), # flatten
        )

        # manually calculate the dimension after all convolutions
        dim_after_conv = 2
        hidden_dim = 32 * dim_after_conv * dim_after_conv

        # Readout layer is mu
        self.readout_mu = nn.Linear(hidden_dim, z_dim)
        # Readout layer
        self.readout_sigma = nn.Linear(hidden_dim, z_dim)

        # You can use nn.ConvTranspose2d to decode
        self.decoder = nn.Sequential(
            nn.Linear(z_dim, hidden_dim),
            nn.Unflatten(1, (32, dim_after_conv, dim_after_conv)),
            nn.ConvTranspose2d(32, 16, kernel_size=4, stride=2, padding=1),
            nn.ReLU(), # relu
            nn.ConvTranspose2d(16, 8, kernel_size=4, stride=2, padding=1), #
            nn.ReLU(), # relu
            nn.ConvTranspose2d(8, 4, kernel_size=4, stride=2, padding=1), #
            nn.ReLU(), # relu
            nn.ConvTranspose2d(4, in_channels, kernel_size=4, stride=2, padd
            nn.Sigmoid(), # use a sigmoid activation to squeeze the outputs
        )

    def reparameterize(self, mu, sigma):
        """
        Reparameterize, i.e. generate a z ~ N(\mu, \sigma)
        """
        # generate epsilon ~ N(0, I)
        # hint: use torch.randn or torch.randn_like
        epsilon = torch.randn_like(sigma)
        # z = \mu + \sigma * \epsilon
```

```
        z = mu + sigma * epsilon
        return z

    def encode(self, x):
        # call the encoder to map input to a hidden state vector
        h = self.encoder(x)
        # use the "readout" layer to get \mu and \sigma
        mu = self.readout_mu(h)
        sigma = self.readout_sigma(h)
        return mu, sigma

    def decode(self, z):
        # call the decoder to map z back to x
        return self.decoder(z)

    def forward(self, x):
        mu, sigma = self.encode(x)
        z = self.reparameterize(mu, sigma)
        x_recon = self.decode(z)
        return x_recon, mu, sigma
```

(c)

For debugging: The `test_kld_loss_func` should output 1.3863

In [ ]:

```
def kld_loss_func(mu, sigma):
    """
    KL-Divergence: KLD = 0.5 * sum(\mu^2 + \sigma^2 - ln(\sigma^2) - 1)

    Parameters
    ----------
    mu: torch.Tensor
        Mean vector in the VAE bottleneck region
    sigma: torch.Tensor
        Standard Deviation vector in the VAE bottleneck region

    Return
    ------
    kld: torch.Tensor
        KL-Divegence loss (a scalar)
    """
    return 0.5 * torch.sum(mu.pow(2) + sigma.pow(2) - torch.log(sigma.pow(2)


def vae_loss_func(recon_x, x, mu, sigma):

    bce_loss = nn.BCELoss(reduction='sum')(recon_x, x)
    kld_loss = kld_loss_func(mu, sigma)

    return bce_loss + kld_loss


def test_kld_loss_func():
    mu = torch.tensor([0.5, 0.5, 1.0])
```

```
    sigma = torch.tensor([1.0, 0.5, 0.5])
    print(kld_loss_func(mu, sigma))

test_kld_loss_func()
```

```
tensor(1.3863)
```

In [ ]:
```
import torch.nn.functional as F

class VAETrainer:

    def __init__(self, model, learning_rate, batch_size, epoch, l2):
        self.model = model
        num_params = sum(item.numel() for item in model.parameters())
        print(f"{model.__class__.__name__} - Number of parameters: {num_para

        self.optimizer = torch.optim.Adam(model.parameters(), learning_rate,

        self.epoch = epoch
        self.batch_size = batch_size



    def train(self, train_data, val_data, early_stop=True, verbose=True, dra
        train_loader = DataLoader(train_data, batch_size=self.batch_size, sh

        train_loss_list = []
        val_loss_list = []

        weights = self.model.state_dict()
        lowest_val_loss = np.inf

        for n in tqdm(range(self.epoch), leave=False):
            self.model.train()
            epoch_loss = 0.0
            for X_batch, y_batch in train_loader:
                batch_importance = 1 / len(train_data)
                # call the model
                X_batch_recon, mu, sigma = self.model(X_batch)
                batch_loss = vae_loss_func(X_batch_recon, X_batch, mu, sigma

                self.optimizer.zero_grad()
                batch_loss.backward()
                self.optimizer.step()

                epoch_loss += batch_loss.detach().cpu().item() * batch_impor

            train_loss_list.append(epoch_loss)

            val_loss = self.evaluate(val_data, print_loss=False)
            val_loss_list.append(val_loss)

            if early_stop:
                if val_loss < lowest_val_loss:
                    lowest_val_loss = val_loss
                    weights = self.model.state_dict()
```

```
        if draw_curve:
            x_axis = np.arange(self.epoch)
            fig, ax = plt.subplots(1, 1, figsize=(5, 4))
            ax.plot(x_axis, train_loss_list, label="Train")
            ax.plot(x_axis, val_loss_list, label="Validation")
            ax.set_title("Total Loss")
            ax.set_xlabel("# Epoch")

        if early_stop:
            self.model.load_state_dict(weights)

        return {
            "train_loss_list": train_loss_list,
            "val_loss_list": val_loss_list,
        }

    def evaluate(self, data, print_loss=True):
        self.model.eval()
        loader = DataLoader(data, batch_size=self.batch_size)
        total_loss = 0.0
        for X_batch, y_batch in loader:
            with torch.no_grad():
                batch_importance = 1 / len(data)
                X_batch_recon, mu, sigma = self.model(X_batch)
                batch_loss = vae_loss_func(X_batch_recon, X_batch, mu, sigma
                total_loss += batch_loss.detach().cpu().item() * batch_impor
        if print_loss:
            print(f"Total Loss: {total_loss}")
        return total_loss
```

In [ ]:

```
train_loader = DataLoader(train_data, batch_size=128, shuffle=True)
for X_batch, y_batch in train_loader:
    print(X_batch.shape)
```

```
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
```

```
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
```

```
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
```

```
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
```

```
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
```

```
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
```

```
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
torch.Size([128, 1, 32, 32])
```

In [ ]:
```
x, y = next(iter(train_loader))
```

In [ ]:
```
y.shape
```

Out[ ]:
```
torch.Size([128])
```

In [ ]:
```
x.shape
```

Out[ ]:
```
torch.Size([128, 1, 32, 32])
```

In [ ]:
```
vae = VAE()
```

In [ ]:
```
vae(x)
```

Out[ ]:
```
(tensor([[[[0.5465, 0.5466, 0.5461,  ..., 0.5467, 0.5459, 0.5462],
           [0.5431, 0.5441, 0.5418,  ..., 0.5430, 0.5411, 0.5471],
           [0.5462, 0.5461, 0.5440,  ..., 0.5463, 0.5431, 0.5472],
           ...,
           [0.5431, 0.5445, 0.5424,  ..., 0.5439, 0.5422, 0.5468],
           [0.5462, 0.5462, 0.5440,  ..., 0.5469, 0.5421, 0.5468],
           [0.5446, 0.5432, 0.5440,  ..., 0.5433, 0.5442, 0.5450]]],


         [[[0.5465, 0.5465, 0.5461,  ..., 0.5467, 0.5459, 0.5462],
           [0.5431, 0.5441, 0.5418,  ..., 0.5430, 0.5411, 0.5471],
           [0.5462, 0.5462, 0.5440,  ..., 0.5463, 0.5431, 0.5472],
           ...,
           [0.5431, 0.5445, 0.5423,  ..., 0.5439, 0.5422, 0.5468],
           [0.5462, 0.5462, 0.5440,  ..., 0.5469, 0.5421, 0.5468],
           [0.5446, 0.5432, 0.5440,  ..., 0.5434, 0.5442, 0.5450]]],


         [[[0.5465, 0.5465, 0.5461,  ..., 0.5467, 0.5459, 0.5462],
           [0.5430, 0.5442, 0.5418,  ..., 0.5430, 0.5411, 0.5471],
           [0.5462, 0.5462, 0.5440,  ..., 0.5463, 0.5431, 0.5472],
           ...,
           [0.5431, 0.5446, 0.5424,  ..., 0.5439, 0.5422, 0.5468],
           [0.5462, 0.5462, 0.5440,  ..., 0.5469, 0.5421, 0.5468],
           [0.5446, 0.5432, 0.5440,  ..., 0.5433, 0.5442, 0.5450]]],


         ...,


         [[[0.5465, 0.5465, 0.5461,  ..., 0.5467, 0.5459, 0.5462],
           [0.5430, 0.5442, 0.5418,  ..., 0.5430, 0.5411, 0.5471],
           [0.5462, 0.5462, 0.5440,  ..., 0.5463, 0.5431, 0.5472],
           ...,
           [0.5431, 0.5445, 0.5424,  ..., 0.5439, 0.5422, 0.5468],
           [0.5462, 0.5462, 0.5439,  ..., 0.5469, 0.5421, 0.5468],
           [0.5446, 0.5432, 0.5440,  ..., 0.5433, 0.5442, 0.5450]]],


         [[[0.5465, 0.5466, 0.5461,  ..., 0.5467, 0.5459, 0.5462],
           [0.5430, 0.5442, 0.5418,  ..., 0.5430, 0.5411, 0.5471],
           [0.5462, 0.5462, 0.5440,  ..., 0.5463, 0.5430, 0.5472],
           ...,
           [0.5431, 0.5445, 0.5423,  ..., 0.5438, 0.5422, 0.5468],
           [0.5462, 0.5462, 0.5439,  ..., 0.5469, 0.5420, 0.5468],
           [0.5446, 0.5432, 0.5440,  ..., 0.5434, 0.5442, 0.5450]]],


         [[[0.5465, 0.5465, 0.5461,  ..., 0.5467, 0.5459, 0.5462],
           [0.5430, 0.5442, 0.5418,  ..., 0.5430, 0.5411, 0.5471],
           [0.5462, 0.5462, 0.5440,  ..., 0.5463, 0.5431, 0.5472],
           ...,
           [0.5431, 0.5445, 0.5424,  ..., 0.5439, 0.5422, 0.5468],
           [0.5462, 0.5462, 0.5440,  ..., 0.5469, 0.5421, 0.5468],
           [0.5446, 0.5432, 0.5440,  ..., 0.5434, 0.5442, 0.5450]]]],
        grad_fn=<SigmoidBackward0>),
```

```
tensor([[-0.0063,  0.0882,  0.0155,  ..., -0.0328,  0.0110, -0.0878],
        [-0.0067,  0.0876,  0.0164,  ..., -0.0327,  0.0111, -0.0903],
        [-0.0060,  0.0866,  0.0179,  ..., -0.0334,  0.0124, -0.0885],
        ...,
        [-0.0091,  0.0878,  0.0183,  ..., -0.0336,  0.0114, -0.0815],
        [-0.0089,  0.0875,  0.0205,  ..., -0.0341,  0.0110, -0.0853],
        [-0.0068,  0.0873,  0.0178,  ..., -0.0346,  0.0101, -0.0884]],
       grad_fn=<AddmmBackward0>),
tensor([[ 0.0068,  0.0253,  0.0311,  ...,  0.0323,  0.0757, -0.0555],
        [ 0.0057,  0.0251,  0.0289,  ...,  0.0334,  0.0748, -0.0554],
        [ 0.0035,  0.0267,  0.0292,  ...,  0.0345,  0.0760, -0.0548],
        ...,
        [ 0.0020,  0.0261,  0.0306,  ...,  0.0363,  0.0740, -0.0537],
        [ 0.0047,  0.0252,  0.0276,  ...,  0.0314,  0.0759, -0.0552],
        [ 0.0063,  0.0269,  0.0267,  ...,  0.0357,  0.0750, -0.0543]],
       grad_fn=<AddmmBackward0>))
```

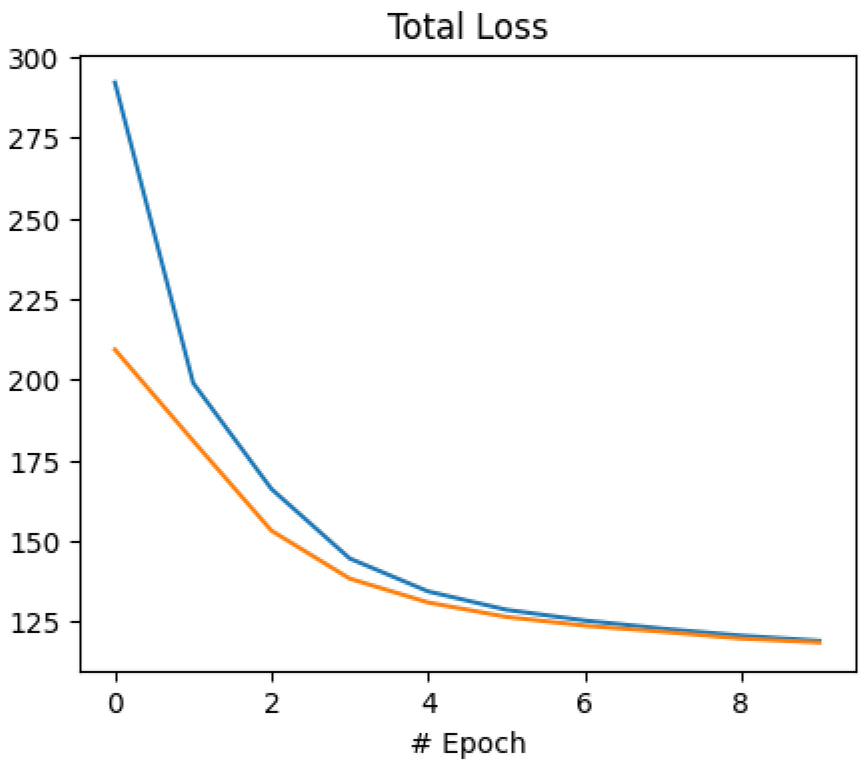
In [ ]:

```python
vae = VAE()
trainer = VAETrainer(vae, 1e-3, 128, 10, 1e-5)
# train
trainer.train(train_data, val_data)
```

```
VAE - Number of parameters: 34201
```

Out[ ]:

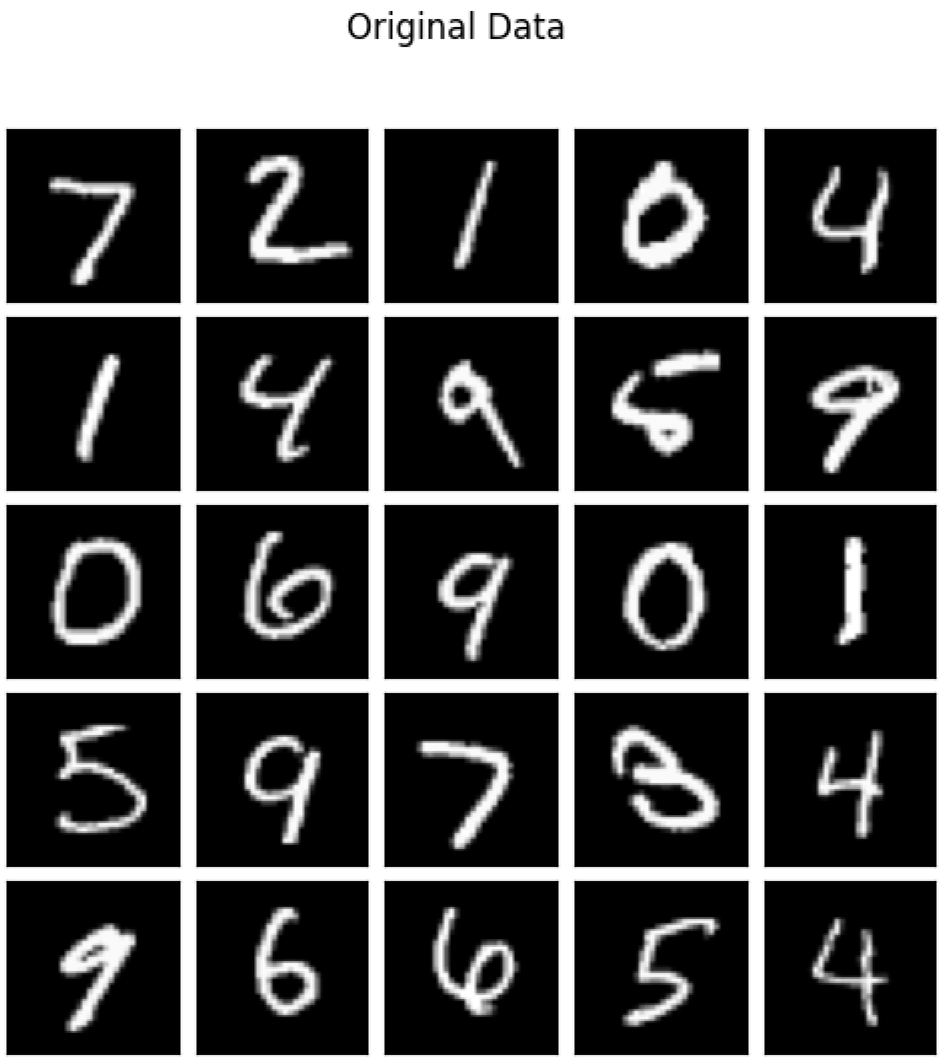
```
{'train_loss_list': [292.1656652832029,
  198.88523193359373,
  166.01494095865885,
  144.44174377441396,
  134.2405961710612,
  128.53765653483072,
  125.20061944580074,
  122.6587432657878,
  120.50614562988284,
  118.90336954752603],
 'val_loss_list': [209.27612076822913,
  180.93946402994783,
  153.07922412109372,
  138.20236848958334,
  130.80785856119792,
  126.32972094726566,
  123.6293977050781,
  121.72244881184905,
  119.62093229166669,
  118.39482324218758]}
```

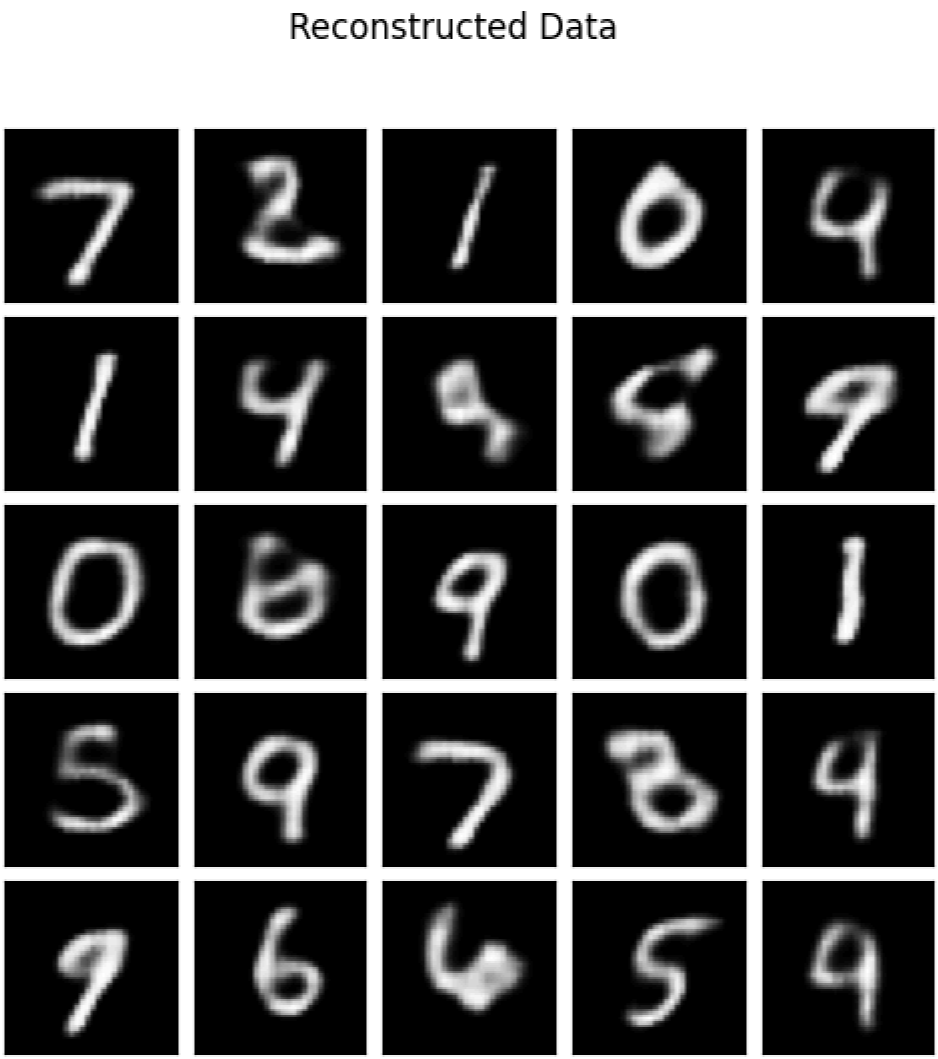


In [ ]:
```
# evaulate the qualitiy of reconstruction
def plot_digits(data, title):
    fig, axes = plt.subplots(5, 5, figsize=(6, 6))
    fig.subplots_adjust(hspace=0.1, wspace=0.1)
    fig.suptitle(title)
    for i, ax in enumerate(axes.flatten()):
        im = ax.imshow(data[i].reshape(32, 32), cmap='gray')
        ax.set_xticks([])
        ax.set_yticks([])

def compare_reconstruct(model, X):
    plot_digits(X, "Original Data")
    with torch.no_grad():
        X_recon, _, _ = model(X)
    plot_digits(X_recon, "Reconstructed Data")


compare_reconstruct(trainer.model, X_test[:100])
```


Original Data



Reconstruction worked well! If I had more time to train on additional epochs, then reconstruction could potentially look even better!

Question 2

In [ ]:
```
from torch_geometric.datasets import QM9
from torch_geometric.loader import DataLoader as GraphDataLoader
from torch_geometric.utils import scatter
```

(a)

In [ ]:
```
def load_qm9(path="./QM9"):
    def transform(data):
        edge_index = torch.tensor(
            list(itertools.permutations(range(data.x.shape[0]), 2)),
            dtype=torch.long
        ).T
        edge_feature = 1 / torch.sqrt(
            torch.sum(
```

```
                (data.pos[edge_index[0]] - data.pos[edge_index[1]]) ** 2,
                axis=1, keepdim=True
            )
        )
        data.edge_index = edge_index
        data.edge_attr = edge_feature
        data.y = data.y[:, [-7]]
        return data

    qm9 = QM9(path, transform=transform)
    return qm9

qm9 = load_qm9("./QM9")
train_data, test_data = train_test_split(qm9, train_size=0.8, test_size=0.2)
```

In [ ]:
```
qm9[0].x.shape
```

Out[ ]: torch.Size([5, 11])

In [ ]:
```
qm9[0].edge_attr.shape
```

Out[ ]: torch.Size([20, 1])

In [ ]:
```
# find out the dimension of node input features
node_input_dim = 11
edge_input_dim = 1
```

(b)

In [ ]:
```
class Layer(nn.Module):
    """
    Basic layer, a linear layer with a ReLU activation
    """
    def __init__(self, in_dim, out_dim):
        super().__init__()
        self.layers = nn.Sequential(
            nn.Linear(in_dim, out_dim), # linear layer
            nn.ReLU() # relu
        )

    def forward(self, x):
        return self.layers(x)


class MessagePassingLayer(nn.Module):
    """
    A message passing layer that updates nodes/edge features
    """
    def __init__(self, node_hidden_dim, edge_hidden_dim):
        super().__init__()
        # figure out the input/output dimension
        self.edge_net = Layer(2*node_hidden_dim + edge_hidden_dim, edge_hidd
        # figure out the input/output dimension
```

```
        self.node_net = Layer(node_hidden_dim + edge_hidden_dim, node_hidder

    def forward(self, node_features, edge_features, edge_index):
        """
        Update node and edge features

        Parameters
        ----------
        node_features: torch.Tensor
            Node features from the previous layer
        edge_features: torch.Tensor
            Edge features from the previous layer
        edge_index: torch.Tensor
            A sparse matrix (n_edge, 2) in which each column denotes node ir
        """
        # concatnate previous edge features with node features forming the e
        # hint: use edge_features[edge_index[0(or 1)]] to get node features
        concate_edge_features = torch.cat([
            node_features[edge_index[0]], # features of one node
            node_features[edge_index[1]], # features of the other node
            edge_features # previous edge features
        ], dim=1)

        # pass through the "edge_net" to map it back to the original dimensi
        updated_edge_features = self.edge_net(concate_edge_features)


        # use scatter to aggrate the edge features to nodes
        aggr_edge_features = scatter(updated_edge_features, edge_index[0])
        # concatenate it with previous node features
        concate_node_features = torch.cat([aggr_edge_features, node_features
        # pass through the "node_net" to map it back to the original dimensi
        updated_node_features = self.node_net(concate_node_features)

        return updated_node_features, updated_edge_features


class GraphNet(nn.Module):
    def __init__(self, node_input_dim, edge_input_dim, node_hidden_dim, edge
        super().__init__()
        # embed the input node features
        self.node_embed = Layer(node_input_dim, node_hidden_dim)
        # embed the input edge features
        self.edge_embed = Layer(edge_input_dim, edge_hidden_dim)
        # message passing layer
        self.message_passing = MessagePassingLayer(node_hidden_dim, edge_hic
        # use a linear layer as readout to get the "atomic" energy contribut
        self.readout = Layer(node_hidden_dim, 1)


    def forward(self, node_features, edge_features, edge_index, batch):
        """
        Update node and edge features

        Parameters
        ----------
```

```
        node_features: torch.Tensor
            Node features from the previous layer
        edge_features: torch.Tensor
            Edge features from the previous layer
        edge_index: torch.Tensor
            A sparse matrix (n_edges, 2) in which each column denotes node i
        batch: torch.Tensor
            A 1-D tensor (n_nodes,) that tells you each node belongs to whic
        """
        node_hidden = self.node_embed(node_features) # call the node embeddi
        edge_hidden = self.edge_embed(edge_features) # call the edge embeddi
        updated_node_hidden, updated_edge_hidden = self.message_passing(node
        readout = self.readout(updated_node_hidden) # use the readout layer
        out = scatter(readout, batch) # use the scatter function to aggregat
        return out
```

In [ ]:

```
class GNNTrainer:
    def __init__(self, model, batch_size, learning_rate, epoch, l2):
        self.model = model

        num_params = sum(item.numel() for item in model.parameters())
        print(f"{model.__class__.__name__} - Number of parameters: {num_para

        self.batch_size = batch_size
        self.optimizer = torch.optim.Adam(model.parameters(), learning_rate,
        self.epoch = epoch

    def train(self, dataset, draw_curve=True):
        self.model.train()
        loader = GraphDataLoader(dataset, batch_size=self.batch_size, shuffl

        loss_func = nn.MSELoss()
        batch_loss_list = []
        for i in range(self.epoch):
            print(f"Epoch: {i}")
            for batch_data in tqdm(loader, leave=False):
                batch_size = batch_data.y.shape[0]
                batch_pred = self.model(batch_data.x, batch_data.edge_attr,
                batch_loss = loss_func(batch_pred, batch_data.y)

                self.optimizer.zero_grad()
                batch_loss.backward()
                self.optimizer.step()

                batch_loss_list.append(batch_loss.detach().numpy())

        if draw_curve:
            fig, ax = plt.subplots(1, 1, figsize=(5, 4), constrained_layout=
            ax.set_yscale("log")
            ax.plot(np.arange(len(batch_loss_list)), batch_loss_list)
            ax.set_xlabel("# Batch")
            ax.set_ylabel("Loss")

        return batch_loss_list

    def evaluate(self, dataset, draw_curve=True):
```

```
        self.model.eval()
        loader = GraphDataLoader(dataset, batch_size=self.batch_size)
        y_true, y_pred = [], []
        with torch.no_grad():
            for batch_data in tqdm(loader, leave=False):
                batch_pred = self.model(batch_data.x, batch_data.edge_attr,
                y_pred.append(batch_pred.detach().numpy().flatten())
                y_true.append(batch_data.y.detach().numpy().flatten())

        y_true = np.concatenate(y_true)
        y_pred = np.concatenate(y_pred)
        mse = np.mean((y_true - y_pred) ** 2)

        if draw_curve:
            fig, ax = plt.subplots(1, 1, figsize=(5, 4), constrained_layout=
            ax.scatter(y_true, y_pred, label=f"MSE: {mse:.2f}", s=2)
            ax.set_xlabel("Ground Truth")
            ax.set_ylabel("Predicted")
            xmin, xmax = ax.get_xlim()
            ymin, ymax = ax.get_ylim()
            vmin, vmax = min(xmin, ymin), max(xmax, ymax)
            ax.set_xlim(vmin, vmax)
            ax.set_ylim(vmin, vmax)
            ax.plot([vmin, vmax], [vmin, vmax], color='red')
            ax.legend()

        return mse
```

(c)

In [ ]:
```
node_hidden_dim = 64
edge_hidden_dim = 64

net = GraphNet(node_input_dim, edge_input_dim, node_hidden_dim, edge_hidden_
```

In [ ]:
```
loader = GraphDataLoader(test_data, batch_size=512)
loader_iter = iter(loader)
```

In [ ]:
```
batch_data = next(loader_iter)
batch_data.y
```

Out[ ]:
```
tensor([[ -70.4864],
        [ -75.7974],
        [ -79.6356],
        [ -93.8650],
        [ -74.7377],
        [ -87.3666],
        [ -83.9652],
        [ -80.5365],
        [ -72.0014],
        [ -60.1266],
        [ -75.0548],
        [ -75.2168],
        [ -89.5242],
        [ -61.2288],
        [ -78.6524],
        [ -75.2447],
        [ -86.2708],
        [ -75.7737],
        [ -73.5429],
        [ -68.3996],
        [ -66.4798],
        [ -88.5515],
        [ -73.5441],
        [ -66.5256],
        [ -75.8396],
        [ -87.3322],
        [ -75.0089],
        [ -67.5386],
        [ -75.9805],
        [ -81.8090],
        [ -77.1845],
        [ -89.2092],
        [ -78.3847],
        [ -74.4109],
        [ -87.7866],
        [ -77.0461],
        [ -71.6219],
        [ -82.8159],
        [ -58.8180],
        [ -74.6424],
        [ -87.5544],
        [ -77.9297],
        [ -57.6477],
        [ -93.9125],
        [ -65.6367],
        [ -97.0705],
        [ -85.6746],
        [ -95.1741],
        [ -67.1238],
        [ -94.0203],
        [ -81.1124],
        [ -73.9630],
        [ -78.8615],
        [ -62.6135],
        [ -81.5175],
        [ -80.6055],
```

```
[ -76.4693],
[ -85.1082],
[ -70.9081],
[ -94.4560],
[ -55.3420],
[ -58.8297],
[ -73.8042],
[ -66.8616],
[ -37.9489],
[ -76.8431],
[ -61.0027],
[ -72.1446],
[ -70.8205],
[ -80.5579],
[ -87.1917],
[ -69.9095],
[ -81.1122],
[ -88.5486],
[ -77.9882],
[ -60.6686],
[ -92.0746],
[ -66.9259],
[ -63.9113],
[ -70.5046],
[ -67.2877],
[ -41.0315],
[-101.0941],
[ -96.5148],
[ -81.7576],
[ -57.0562],
[ -79.1949],
[ -83.3271],
[ -72.2305],
[ -73.4217],
[ -51.0731],
[ -78.9340],
[ -74.5987],
[ -76.5236],
[ -79.3786],
[ -75.7702],
[ -95.0754],
[ -68.4121],
[ -73.6670],
[ -75.3127],
[ -68.0103],
[ -71.3626],
[ -76.5123],
[ -81.7318],
[ -73.4120],
[ -67.5974],
[ -85.8803],
[ -80.2097],
[ -70.0820],
[ -82.3549],
[ -63.0495],
[ -83.4827],
```

```
[ -75.8810],
[ -66.8625],
[ -81.3980],
[ -91.2998],
[ -76.0617],
[ -75.0103],
[ -66.5512],
[ -78.7859],
[ -81.2091],
[ -82.2257],
[ -90.4977],
[ -91.3698],
[ -63.2929],
[ -83.5035],
[ -83.2906],
[ -88.4732],
[ -74.7991],
[ -77.8216],
[ -68.1019],
[ -67.4237],
[ -70.4183],
[ -73.6437],
[ -80.8961],
[ -78.3066],
[ -80.2407],
[ -71.9250],
[ -57.7383],
[ -76.5072],
[ -75.6422],
[ -79.1316],
[ -93.7921],
[ -78.9215],
[ -83.6718],
[ -80.3796],
[ -74.9084],
[ -60.7959],
[ -80.2645],
[ -87.2560],
[ -70.9590],
[ -71.5249],
[ -73.8898],
[ -76.9050],
[ -53.7568],
[ -79.6581],
[ -72.2228],
[ -76.0661],
[ -76.5177],
[ -75.5088],
[ -81.4862],
[ -75.9152],
[ -74.9022],
[ -81.7600],
[ -73.6292],
[ -61.2251],
[ -82.5225],
[ -79.0467],
```

```
[ -82.1079],
[ -82.3408],
[ -96.2954],
[ -89.4135],
[ -63.7231],
[ -76.1637],
[ -74.3476],
[ -66.7407],
[ -64.2249],
[ -87.0001],
[ -79.3765],
[ -83.6915],
[ -88.0992],
[ -86.9751],
[ -58.3500],
[ -72.0177],
[ -73.4034],
[ -68.7962],
[ -70.3934],
[ -66.8600],
[ -56.3114],
[ -71.7804],
[ -74.1163],
[ -80.4022],
[ -85.3989],
[ -70.6258],
[ -73.0060],
[ -75.0191],
[ -80.5904],
[ -70.7717],
[ -73.1086],
[ -75.3841],
[ -71.6425],
[ -84.9651],
[ -93.8405],
[ -62.6765],
[ -76.6470],
[ -57.8895],
[ -88.4952],
[ -59.6330],
[ -70.7936],
[ -79.5274],
[ -82.3046],
[ -67.9861],
[ -70.4074],
[ -75.5867],
[ -73.0943],
[ -74.5136],
[ -81.1540],
[ -86.7441],
[ -79.2224],
[ -90.6063],
[ -70.1166],
[ -90.1102],
[ -67.2248],
[ -79.9050],
```

```
        [ -63.6127],
        [ -73.4543],
        [ -69.7469],
        [ -70.8926],
        [ -76.6120],
        [ -83.7805],
        [ -61.8384],
        [ -70.8758],
        [ -65.0438],
        [ -72.8134],
        [ -95.1231],
        [ -74.9355],
        [ -70.6625],
        [ -86.0061],
        [ -91.7056],
        [ -66.4922],
        [ -72.5621],
        [ -54.6563],
        [-102.2002],
        [ -82.7756],
        [ -78.8943],
        [ -74.1533],
        [ -75.9885],
        [ -86.4212],
        [ -73.8273],
        [ -63.9799],
        [ -84.4209],
        [ -78.4436],
        [ -86.5860],
        [ -72.2286],
        [ -51.2812],
        [ -87.7859],
        [ -85.8625],
        [ -67.7601],
        [ -76.0333],
        [ -81.2016],
        [ -72.1317],
        [ -73.2932],
        [ -58.9422],
        [ -76.4288],
        [ -78.3884],
        [ -78.6968],
        [ -70.5991],
        [ -79.5566],
        [ -66.9264],
        [ -74.1060],
        [ -77.1276],
        [ -85.0765],
        [ -83.5052],
        [ -82.8198],
        [ -65.2110],
        [ -63.6381],
        [ -76.8797],
        [ -83.3036],
        [ -71.5233],
        [ -81.1289],
```

```
[ -75.6200],
[ -79.6771],
[ -73.7961],
[ -71.7426],
[ -75.6720],
[ -51.2072],
[ -88.1074],
[ -87.0754],
[ -75.6804],
[ -90.6549],
[ -74.0448],
[ -80.3714],
[ -67.6168],
[ -59.6295],
[ -96.4873],
[ -81.0142],
[ -82.9289],
[ -81.8366],
[ -76.7733],
[ -75.3229],
[ -88.5846],
[ -78.2418],
[ -68.1004],
[ -87.5152],
[ -96.5140],
[ -65.1612],
[ -75.0258],
[ -67.0584],
[ -57.5403],
[ -75.0545],
[ -73.1014],
[ -82.7343],
[ -82.6280],
[ -67.4128],
[ -72.0509],
[ -82.6509],
[ -58.7723],
[ -66.1950],
[ -83.9626],
[ -83.3127],
[-107.8617],
[ -93.6045],
[ -74.3767],
[ -65.8263],
[ -64.7234],
[ -60.4570],
[ -71.6384],
[ -83.2223],
[ -71.7960],
[ -61.9289],
[ -69.0919],
[ -73.2464],
[ -68.6373],
[ -81.4486],
[ -69.0684],
[ -85.4613],
```

```
        [ -78.3028],
        [ -65.0102],
        [ -79.9280],
        [ -72.8375],
        [ -79.6869],
        [ -85.3697],
        [ -81.8741],
        [ -70.3284],
        [ -63.9028],
        [ -84.6850],
        [ -53.0928],
        [ -56.8389],
        [ -82.3751],
        [ -69.2758],
        [-101.1958],
        [ -78.7151],
        [ -75.9877],
        [ -72.1837],
        [ -65.2828],
        [ -89.8850],
        [ -76.0609],
        [ -68.6102],
        [ -90.8762],
        [ -64.1556],
        [ -78.2033],
        [ -86.9983],
        [ -88.4489],
        [ -86.9338],
        [ -70.4767],
        [ -64.8573],
        [ -87.4995],
        [ -73.9009],
        [ -84.7698],
        [ -80.9867],
        [ -42.6205],
        [ -79.2680],
        [ -73.8050],
        [ -58.6043],
        [ -86.1403],
        [ -89.7575],
        [ -63.0927],
        [ -80.8930],
        [ -74.2156],
        [ -73.6562],
        [ -89.3239],
        [ -73.5798],
        [ -93.0783],
        [ -81.8353],
        [ -85.6103],
        [ -76.1923],
        [ -81.7916],
        [ -70.9214],
        [ -94.7765],
        [ -78.1407],
        [ -79.1235],
        [ -79.6129],
```

```
[ -65.6965],
[ -74.2840],
[ -69.2655],
[ -74.1983],
[ -79.6958],
[ -66.7160],
[ -74.9008],
[ -76.8486],
[ -68.1188],
[ -67.9364],
[ -92.7917],
[ -63.7563],
[ -73.4181],
[ -78.4418],
[ -59.9365],
[ -69.0717],
[ -70.9760],
[ -69.3785],
[ -88.5632],
[ -71.7750],
[ -87.4986],
[ -83.7122],
[ -72.4035],
[ -66.9430],
[ -74.2785],
[ -68.9441],
[ -75.6072],
[ -74.9635],
[ -75.7605],
[ -93.8173],
[ -78.7987],
[ -96.4589],
[ -79.9341],
[ -82.6664],
[ -85.3342],
[ -80.2659],
[ -70.5505],
[ -81.7613],
[ -96.0561],
[ -63.3860],
[ -82.5181],
[ -86.3968],
[ -62.9638],
[ -67.3688],
[ -65.3137],
[ -83.8509],
[ -82.3700],
[ -59.4710],
[ -87.8020],
[ -78.5563],
[ -67.3657],
[ -68.2649],
[ -63.7901],
[ -67.2378],
[ -65.5137],
[ -65.4720],
```

```
        [ -93.9035],
        [ -75.3406],
        [ -60.1730],
        [ -76.2883],
        [ -80.4947],
        [ -81.6021],
        [ -68.6204],
        [ -69.7101],
        [ -77.3100],
        [ -65.9950],
        [ -89.2519],
        [ -94.0955],
        [ -61.6324],
        [ -75.6938],
        [ -99.8534],
        [ -89.2810],
        [ -70.8631],
        [ -74.6044],
        [ -81.4417],
        [ -86.8994],
        [ -76.2152],
        [ -68.7798],
        [ -62.4219],
        [ -74.9229],
        [ -65.6250],
        [ -62.1877],
        [ -73.3145],
        [ -87.1040],
        [ -79.0401],
        [ -62.7523],
        [ -83.9756],
        [ -73.1909],
        [ -58.7205],
        [ -75.1858],
        [ -64.5377],
        [ -70.9141],
        [ -69.2055],
        [ -80.5774],
        [ -69.2533],
        [ -60.7993],
        [ -81.9226],
        [ -81.6709],
        [ -45.6235],
        [ -79.0195],
        [ -77.0576],
        [ -81.3962],
        [ -73.0852],
        [ -75.5317],
        [ -88.1486],
        [ -74.0096],
        [ -68.7264],
        [ -87.5594],
        [ -63.7013],
        [ -56.6853],
        [ -72.2785],
        [ -91.8890],
```

```
        [ -76.9250],
        [ -80.9735],
        [-107.0260],
        [ -79.2465],
        [ -90.1927],
        [ -71.6768],
        [ -81.8217],
        [ -85.5043]])
```

In [ ]:
```
batch_pred = net(batch_data.x, batch_data.edge_attr, batch_data.edge_index,
batch_pred
```

Out[ ]:
```
tensor([[19.1684],
        [17.9427],
        [23.0643],
        [30.7166],
        [18.7136],
        [26.9141],
        [24.4820],
        [25.4380],
        [19.4917],
        [12.3360],
        [22.1620],
        [17.9877],
        [32.6588],
        [12.9678],
        [19.7037],
        [21.3127],
        [26.9698],
        [20.8378],
        [21.3007],
        [17.8954],
        [16.2375],
        [25.9707],
        [17.1623],
        [16.9777],
        [21.2631],
        [29.0107],
        [18.7519],
        [18.6176],
        [20.4353],
        [26.0143],
        [19.1760],
        [29.0234],
        [22.1520],
        [22.1093],
        [29.0131],
        [23.5389],
        [18.3152],
        [24.5618],
        [13.6811],
        [19.9675],
        [26.4476],
        [23.6245],
        [11.6114],
        [30.6554],
        [15.9336],
        [36.7241],
        [26.9584],
        [30.6050],
        [16.2216],
        [30.6699],
        [21.7765],
        [21.2820],
        [23.1113],
        [13.3928],
        [24.5382],
        [25.5137],
```

```
[19.1880],
[25.9862],
[17.5159],
[30.6102],
[10.8966],
[12.2515],
[18.7518],
[14.8584],
[ 7.6301],
[20.8581],
[13.3215],
[19.9305],
[19.5303],
[22.1793],
[25.9798],
[17.8741],
[23.1816],
[25.9842],
[18.0423],
[13.2241],
[31.6089],
[14.1366],
[17.0108],
[17.5361],
[16.2338],
[ 6.1476],
[34.6560],
[29.6163],
[24.4904],
[12.5351],
[22.6107],
[21.7598],
[19.9262],
[19.1204],
[ 8.5680],
[20.4188],
[20.4358],
[20.8914],
[23.0899],
[21.3107],
[31.3127],
[15.2134],
[18.7456],
[21.2608],
[17.8389],
[17.1668],
[23.5114],
[24.4548],
[17.1906],
[16.2859],
[26.8605],
[21.8062],
[17.9491],
[23.1440],
[12.9811],
[24.4689],
```

```
        [21.2804],
        [18.6146],
        [25.4516],
        [31.6538],
        [20.9602],
        [18.3554],
        [15.5985],
        [20.5334],
        [25.4843],
        [20.9539],
        [27.9715],
        [31.5940],
        [14.0912],
        [24.5074],
        [20.9211],
        [26.0113],
        [21.8292],
        [20.4333],
        [15.5674],
        [18.2852],
        [16.7839],
        [19.6134],
        [25.9508],
        [23.5519],
        [25.4544],
        [19.5783],
        [12.2702],
        [17.9938],
        [21.7263],
        [23.5033],
        [33.8753],
        [22.6226],
        [24.4650],
        [22.5842],
        [22.2172],
        [13.6178],
        [22.6751],
        [26.0244],
        [20.3200],
        [19.4903],
        [16.8182],
        [20.4666],
        [10.9650],
        [22.5976],
        [19.9664],
        [24.0932],
        [21.2719],
        [21.2792],
        [25.4243],
        [19.1846],
        [16.8143],
        [21.7818],
        [21.2992],
        [16.5617],
        [24.5601],
        [23.0256],
```

```
[24.4693],
[21.8046],
[32.7845],
[25.0396],
[14.4940],
[21.2315],
[18.3622],
[16.2461],
[16.2080],
[27.5449],
[22.5603],
[23.5991],
[26.0403],
[26.4534],
[12.3029],
[19.1024],
[19.4798],
[18.6514],
[15.5259],
[16.2010],
[13.3257],
[20.0025],
[21.3234],
[22.6641],
[23.5676],
[19.9180],
[19.2525],
[19.2207],
[22.6078],
[19.2445],
[19.5366],
[21.2771],
[16.7573],
[24.5071],
[30.6198],
[12.6881],
[19.6851],
[12.8878],
[26.0030],
[11.3613],
[16.6952],
[19.6642],
[21.8315],
[15.8994],
[14.9164],
[20.9251],
[18.7807],
[21.2749],
[21.8294],
[26.1206],
[23.1467],
[25.1870],
[16.8143],
[28.0632],
[18.6043],
[23.0483],
```

```
        [15.1163],
        [19.6002],
        [17.8911],
        [19.5065],
        [20.4114],
        [24.5813],
        [13.6045],
        [17.4702],
        [15.4789],
        [19.4905],
        [29.6348],
        [17.9361],
        [17.5268],
        [26.9385],
        [31.6033],
        [15.5916],
        [17.4364],
        [11.2050],
        [34.6513],
        [21.7796],
        [23.4661],
        [21.2639],
        [20.9020],
        [26.9919],
        [16.7211],
        [15.2084],
        [27.4023],
        [20.4509],
        [26.9040],
        [17.4841],
        [ 8.8135],
        [25.9699],
        [26.9344],
        [18.6368],
        [20.4754],
        [22.6358],
        [19.1774],
        [19.2276],
        [13.3305],
        [24.0024],
        [17.9579],
        [20.4189],
        [17.4376],
        [22.6277],
        [16.5904],
        [18.8059],
        [20.4210],
        [26.8881],
        [24.4759],
        [27.9251],
        [14.4179],
        [14.4432],
        [23.5107],
        [22.2815],
        [19.1885],
        [24.4675],
```

```
        [20.4309],
        [25.4610],
        [19.1665],
        [20.2485],
        [20.4149],
        [ 9.6889],
        [25.0784],
        [25.9347],
        [19.1890],
        [25.1498],
        [19.9512],
        [25.4379],
        [16.6571],
        [13.6581],
        [29.6115],
        [25.0069],
        [24.4524],
        [25.4618],
        [24.0449],
        [21.3163],
        [25.0351],
        [20.4562],
        [15.6011],
        [26.0326],
        [36.7629],
        [15.5735],
        [19.1455],
        [18.2332],
        [12.4594],
        [21.2550],
        [19.5375],
        [27.9908],
        [25.4435],
        [16.6509],
        [19.4902],
        [23.1181],
        [11.5855],
        [18.5716],
        [21.7617],
        [24.5472],
        [39.9525],
        [30.6579],
        [19.1482],
        [17.0090],
        [14.4413],
        [11.5917],
        [19.9805],
        [24.4960],
        [19.1553],
        [13.0283],
        [15.8915],
        [19.4491],
        [17.8429],
        [25.5668],
        [17.5605],
        [23.6110],
```

```
[23.5443],
[15.6211],
[22.6703],
[16.7589],
[21.7667],
[26.8808],
[21.7924],
[15.2177],
[12.7120],
[26.9758],
[ 9.8926],
[10.3473],
[21.8656],
[17.8773],
[34.5362],
[23.9972],
[21.2857],
[15.9953],
[16.6834],
[29.0024],
[21.2780],
[17.8973],
[31.6380],
[16.2821],
[23.6020],
[27.5190],
[26.0220],
[26.0287],
[17.4473],
[14.4633],
[30.0196],
[21.2756],
[24.5046],
[19.6796],
[ 8.6751],
[25.4670],
[19.0888],
[12.1793],
[26.9115],
[25.0824],
[14.0699],
[25.4162],
[19.2143],
[22.1506],
[26.0069],
[17.1016],
[30.6633],
[21.7910],
[26.9551],
[21.7537],
[24.5175],
[16.0151],
[32.7671],
[23.5783],
[22.6292],
[24.4730],
```

```
        [16.9279],
        [19.9340],
        [17.1944],
        [17.1387],
        [22.1850],
        [17.0028],
        [18.3616],
        [23.4821],
        [16.2898],
        [18.2759],
        [30.6285],
        [14.7513],
        [19.4924],
        [23.5149],
        [13.3387],
        [17.5256],
        [17.5682],
        [15.5073],
        [26.0327],
        [20.0137],
        [26.0030],
        [23.5980],
        [17.1283],
        [18.6625],
        [18.7486],
        [15.5914],
        [21.2844],
        [19.0620],
        [22.6480],
        [33.8875],
        [22.5812],
        [29.6309],
        [23.4687],
        [24.5734],
        [24.5060],
        [25.9054],
        [17.8470],
        [25.9290],
        [29.6670],
        [14.7768],
        [23.0997],
        [27.0083],
        [14.8286],
        [18.2033],
        [14.3836],
        [23.5625],
        [24.5106],
        [11.6363],
        [26.4002],
        [17.9336],
        [16.3296],
        [17.5260],
        [14.4279],
        [16.2319],
        [14.4743],
        [14.4385],
```

```
[30.6900],
[21.2987],
[12.2853],
[20.8901],
[22.6249],
[19.6610],
[17.8928],
[15.5200],
[20.4803],
[14.4447],
[28.9384],
[30.6255],
[14.3556],
[18.4190],
[35.6916],
[26.5502],
[14.8149],
[21.2717],
[22.1250],
[25.9973],
[20.4693],
[17.5368],
[14.0637],
[22.2173],
[14.4345],
[15.4635],
[19.9048],
[25.9893],
[25.4108],
[17.8626],
[27.4638],
[19.1835],
[11.6858],
[18.7686],
[15.1368],
[19.4800],
[17.9637],
[21.7715],
[18.2344],
[12.9657],
[22.6301],
[19.6371],
[ 9.9112],
[22.6796],
[19.5958],
[25.4663],
[19.1495],
[21.3365],
[26.0094],
[23.9258],
[15.5971],
[28.9958],
[14.1474],
[13.2570],
[19.9056],
[31.6052],
```

```
        [24.9780],
        [22.6175],
        [39.9239],
        [20.4983],
        [28.0160],
        [19.1934],
        [21.7209],
        [26.9410]], grad_fn=<ScatterAddBackward0>)
```

In [ ]:

```
loss_func = nn.MSELoss()
loss_func(batch_pred, batch_data.y)
```

Out[ ]:

```
tensor(9634.5879, grad_fn=<MseLossBackward0>)
```

In [ ]:

```
# 1e-3, 128, 10, 1e-5

# train
learning_rate = 1e-3
n_epoch = 3
batch_size = 512
l2 = 1e-5

trainer = GNNTrainer(net, batch_size, learning_rate, n_epoch, l2)
trainer.train(train_data)
```

```
GraphNet - Number of parameters: 21569
Epoch: 0

Epoch: 1

Epoch: 2
```

```
Out[ ]: [array(9736.492, dtype=float32),
         array(6316.7534, dtype=float32),
         array(5924.873, dtype=float32),
         array(5907.2515, dtype=float32),
         array(5884.923, dtype=float32),
         array(5787.246, dtype=float32),
         array(5792.818, dtype=float32),
         array(5887.7734, dtype=float32),
         array(5865.004, dtype=float32),
         array(5804.1816, dtype=float32),
         array(5869.4536, dtype=float32),
         array(5805.659, dtype=float32),
         array(6018.964, dtype=float32),
         array(5907.6255, dtype=float32),
         array(5914.883, dtype=float32),
         array(5850.363, dtype=float32),
         array(5946.798, dtype=float32),
         array(5932.56, dtype=float32),
         array(5940.7495, dtype=float32),
         array(5983.2334, dtype=float32),
         array(5883.205, dtype=float32),
         array(5845.581, dtype=float32),
         array(5790.1074, dtype=float32),
         array(5913.377, dtype=float32),
         array(5867.625, dtype=float32),
         array(5912.2925, dtype=float32),
         array(5891.485, dtype=float32),
         array(5914.533, dtype=float32),
         array(5881.984, dtype=float32),
         array(5894.752, dtype=float32),
         array(5884.404, dtype=float32),
         array(5965.698, dtype=float32),
         array(5909.1494, dtype=float32),
         array(5861.289, dtype=float32),
         array(5891.264, dtype=float32),
         array(6045.749, dtype=float32),
         array(5838.654, dtype=float32),
         array(5911.449, dtype=float32),
         array(5834.392, dtype=float32),
         array(5966.5957, dtype=float32),
         array(5922.8955, dtype=float32),
         array(5876.157, dtype=float32),
         array(5973.8945, dtype=float32),
         array(5878.112, dtype=float32),
         array(5714.804, dtype=float32),
         array(5830.4824, dtype=float32),
         array(5774.551, dtype=float32),
         array(5921.841, dtype=float32),
         array(5959.5615, dtype=float32),
         array(5851.961, dtype=float32),
         array(5941.869, dtype=float32),
         array(5781.862, dtype=float32),
         array(5904.33, dtype=float32),
         array(5844.794, dtype=float32),
         array(5934.712, dtype=float32),
         array(5970.752, dtype=float32),
```

```
              array(5896.122, dtype=float32),
              array(5910.3877, dtype=float32),
              array(6057.898, dtype=float32),
              array(5913.632, dtype=float32),
              array(5874.0205, dtype=float32),
              array(5851.3066, dtype=float32),
              array(5811.8906, dtype=float32),
              array(5927.3545, dtype=float32),
              array(5899.5625, dtype=float32),
              array(5812.3564, dtype=float32),
              array(5926.455, dtype=float32),
              array(5719.8335, dtype=float32),
              array(5831.518, dtype=float32),
              array(5922.9424, dtype=float32),
              array(5889.759, dtype=float32),
              array(5981.812, dtype=float32),
              array(5915.8916, dtype=float32),
              array(5852.4536, dtype=float32),
              array(5969.623, dtype=float32),
              array(5995.0503, dtype=float32),
              array(5933.6177, dtype=float32),
              array(5948.2554, dtype=float32),
              array(5947.469, dtype=float32),
              array(6038.062, dtype=float32),
              array(5774.287, dtype=float32),
              array(5930.373, dtype=float32),
              array(5895.2163, dtype=float32),
              array(5921.6143, dtype=float32),
              array(5862.067, dtype=float32),
              array(5962.9717, dtype=float32),
              array(5908.6484, dtype=float32),
              array(5942.5547, dtype=float32),
              array(5876.2803, dtype=float32),
              array(5878.5264, dtype=float32),
              array(5924.4985, dtype=float32),
              array(6003.955, dtype=float32),
              array(5924.5894, dtype=float32),
              array(5868.7627, dtype=float32),
              array(5800.94, dtype=float32),
              array(5894.1035, dtype=float32),
              array(5825.203, dtype=float32),
              array(5917.772, dtype=float32),
              array(5857.3926, dtype=float32),
              array(6020.645, dtype=float32),
              array(5848.888, dtype=float32),
              array(5914.2812, dtype=float32),
              array(6006.1465, dtype=float32),
              array(5922.177, dtype=float32),
              array(6059.56, dtype=float32),
              array(5941.948, dtype=float32),
              array(5873.7812, dtype=float32),
              array(5970.5005, dtype=float32),
              array(5949.282, dtype=float32),
              array(5822.5874, dtype=float32),
              array(5871.0015, dtype=float32),
              array(5934.9097, dtype=float32),
```

```
array(5930.656, dtype=float32),
array(5900.956, dtype=float32),
array(6106.0156, dtype=float32),
array(5979.9155, dtype=float32),
array(5912.449, dtype=float32),
array(6075.28, dtype=float32),
array(5841.0176, dtype=float32),
array(5893.985, dtype=float32),
array(5951.366, dtype=float32),
array(5852.457, dtype=float32),
array(6004.15, dtype=float32),
array(5912.547, dtype=float32),
array(5976.2417, dtype=float32),
array(5911.945, dtype=float32),
array(5977.284, dtype=float32),
array(5966.512, dtype=float32),
array(5961.4434, dtype=float32),
array(5841.331, dtype=float32),
array(5957.8413, dtype=float32),
array(5808.592, dtype=float32),
array(5827.7373, dtype=float32),
array(5895.808, dtype=float32),
array(5921.927, dtype=float32),
array(5881.007, dtype=float32),
array(5853.5596, dtype=float32),
array(5919.9395, dtype=float32),
array(5957.0327, dtype=float32),
array(5936.836, dtype=float32),
array(5857.021, dtype=float32),
array(6014.938, dtype=float32),
array(5843.7637, dtype=float32),
array(5857.4893, dtype=float32),
array(5856.342, dtype=float32),
array(5901.4844, dtype=float32),
array(5879.2866, dtype=float32),
array(5990.755, dtype=float32),
array(5883.3687, dtype=float32),
array(5961.7275, dtype=float32),
array(5903.1685, dtype=float32),
array(5892.7173, dtype=float32),
array(5839.69, dtype=float32),
array(5864.99, dtype=float32),
array(5925.6714, dtype=float32),
array(5940.9395, dtype=float32),
array(5891.408, dtype=float32),
array(5771.519, dtype=float32),
array(5934.248, dtype=float32),
array(5834.1304, dtype=float32),
array(5831.2495, dtype=float32),
array(5899.7, dtype=float32),
array(5914.12, dtype=float32),
array(5865.602, dtype=float32),
array(5875.216, dtype=float32),
array(5875.7803, dtype=float32),
array(5938.8613, dtype=float32),
array(5893.06, dtype=float32),
```

```
array(5889.529, dtype=float32),
array(5920.957, dtype=float32),
array(5910.1753, dtype=float32),
array(5968.265, dtype=float32),
array(6000.1816, dtype=float32),
array(5767.964, dtype=float32),
array(5903.6797, dtype=float32),
array(5772.2227, dtype=float32),
array(5926.929, dtype=float32),
array(5861.7183, dtype=float32),
array(5838.0864, dtype=float32),
array(5894.212, dtype=float32),
array(5931.498, dtype=float32),
array(5991.4604, dtype=float32),
array(5874.81, dtype=float32),
array(5936.85, dtype=float32),
array(5876.2, dtype=float32),
array(5836.0234, dtype=float32),
array(5934.538, dtype=float32),
array(5837.826, dtype=float32),
array(5986.9087, dtype=float32),
array(5926.631, dtype=float32),
array(5836.594, dtype=float32),
array(5893.277, dtype=float32),
array(6008.651, dtype=float32),
array(5867.5376, dtype=float32),
array(5924.232, dtype=float32),
array(6000.62, dtype=float32),
array(5889.3467, dtype=float32),
array(5626.2524, dtype=float32),
array(5927.215, dtype=float32),
array(5952.5923, dtype=float32),
array(6025.4277, dtype=float32),
array(5780.54, dtype=float32),
array(5916.662, dtype=float32),
array(5931.632, dtype=float32),
array(5938.825, dtype=float32),
array(5912.3433, dtype=float32),
array(5963.041, dtype=float32),
array(5845.812, dtype=float32),
array(5910.8584, dtype=float32),
array(5873.3525, dtype=float32),
array(5874.176, dtype=float32),
array(5873.1504, dtype=float32),
array(5938.654, dtype=float32),
array(5922.1123, dtype=float32),
array(5813.879, dtype=float32),
array(5883.422, dtype=float32),
array(5858.8457, dtype=float32),
array(5860.183, dtype=float32),
array(5926.2686, dtype=float32),
array(5854.402, dtype=float32),
array(6034.5654, dtype=float32),
array(5904.8013, dtype=float32),
array(5923.1777, dtype=float32),
array(5820.384, dtype=float32),
```

```
              array(5831.329, dtype=float32),
              array(5898.8496, dtype=float32),
              array(5908.1367, dtype=float32),
              array(5880.578, dtype=float32),
              array(5930.2656, dtype=float32),
              array(5922.466, dtype=float32),
              array(5922.619, dtype=float32),
              array(5892.242, dtype=float32),
              array(5944.957, dtype=float32),
              array(5915.02, dtype=float32),
              array(5908.9775, dtype=float32),
              array(5924.048, dtype=float32),
              array(5840.3457, dtype=float32),
              array(5933.174, dtype=float32),
              array(5945.4863, dtype=float32),
              array(5972.847, dtype=float32),
              array(5925.498, dtype=float32),
              array(5948.1387, dtype=float32),
              array(6011.8706, dtype=float32),
              array(5861.589, dtype=float32),
              array(5869.655, dtype=float32),
              array(5931.479, dtype=float32),
              array(5924.7466, dtype=float32),
              array(5783.2583, dtype=float32),
              array(5970.614, dtype=float32),
              array(5860.381, dtype=float32),
              array(5746.1562, dtype=float32),
              array(5898.455, dtype=float32),
              array(5892.7305, dtype=float32),
              array(5954.994, dtype=float32),
              array(5856.945, dtype=float32),
              array(5930.4443, dtype=float32),
              array(5938.4844, dtype=float32),
              array(5926.8096, dtype=float32),
              array(5917.0566, dtype=float32),
              array(5844.2617, dtype=float32),
              array(5961.282, dtype=float32),
              array(6018.7935, dtype=float32),
              array(5852.73, dtype=float32),
              array(5938.3574, dtype=float32),
              array(5899.396, dtype=float32),
              array(5884.0234, dtype=float32),
              array(6063.626, dtype=float32),
              array(5883.73, dtype=float32),
              array(5861.615, dtype=float32),
              array(5857.012, dtype=float32),
              array(5891.9595, dtype=float32),
              array(5843.2246, dtype=float32),
              array(6015.5273, dtype=float32),
              array(5837.045, dtype=float32),
              array(5906.604, dtype=float32),
              array(5928.3667, dtype=float32),
              array(5881.8994, dtype=float32),
              array(5872.0977, dtype=float32),
              array(5927.6733, dtype=float32),
              array(5805.4634, dtype=float32),
```

```
array(5931.6665, dtype=float32),
array(5820.02, dtype=float32),
array(5918.332, dtype=float32),
array(5835.9297, dtype=float32),
array(5882.3345, dtype=float32),
array(5846.2827, dtype=float32),
array(5942.509, dtype=float32),
array(5861.367, dtype=float32),
array(5852.1836, dtype=float32),
array(5958.8774, dtype=float32),
array(5787.399, dtype=float32),
array(5852.4404, dtype=float32),
array(5953.5234, dtype=float32),
array(5826.5596, dtype=float32),
array(5996.2334, dtype=float32),
array(5894.821, dtype=float32),
array(5853.197, dtype=float32),
array(5919.124, dtype=float32),
array(5853.71, dtype=float32),
array(5929.9336, dtype=float32),
array(5947.1826, dtype=float32),
array(5951.7866, dtype=float32),
array(5919.6387, dtype=float32),
array(5810.324, dtype=float32),
array(5840.6987, dtype=float32),
array(5851.2803, dtype=float32),
array(5950.6836, dtype=float32),
array(5981.9634, dtype=float32),
array(5911.707, dtype=float32),
array(5916.8955, dtype=float32),
array(5889.4487, dtype=float32),
array(5891.88, dtype=float32),
array(5970.462, dtype=float32),
array(5889.372, dtype=float32),
array(5982.1147, dtype=float32),
array(5854.2783, dtype=float32),
array(5868.535, dtype=float32),
array(5967.034, dtype=float32),
array(5810.3413, dtype=float32),
array(5843.019, dtype=float32),
array(5926.248, dtype=float32),
array(5954.6357, dtype=float32),
array(5905.9023, dtype=float32),
array(5949.6396, dtype=float32),
array(5846.2334, dtype=float32),
array(5941.911, dtype=float32),
array(5897.5596, dtype=float32),
array(5954.0557, dtype=float32),
array(5828.4565, dtype=float32),
array(5958.738, dtype=float32),
array(5815.2188, dtype=float32),
array(6043.4614, dtype=float32),
array(5908.337, dtype=float32),
array(5838.8354, dtype=float32),
array(5792.3047, dtype=float32),
array(5873.5513, dtype=float32),
```

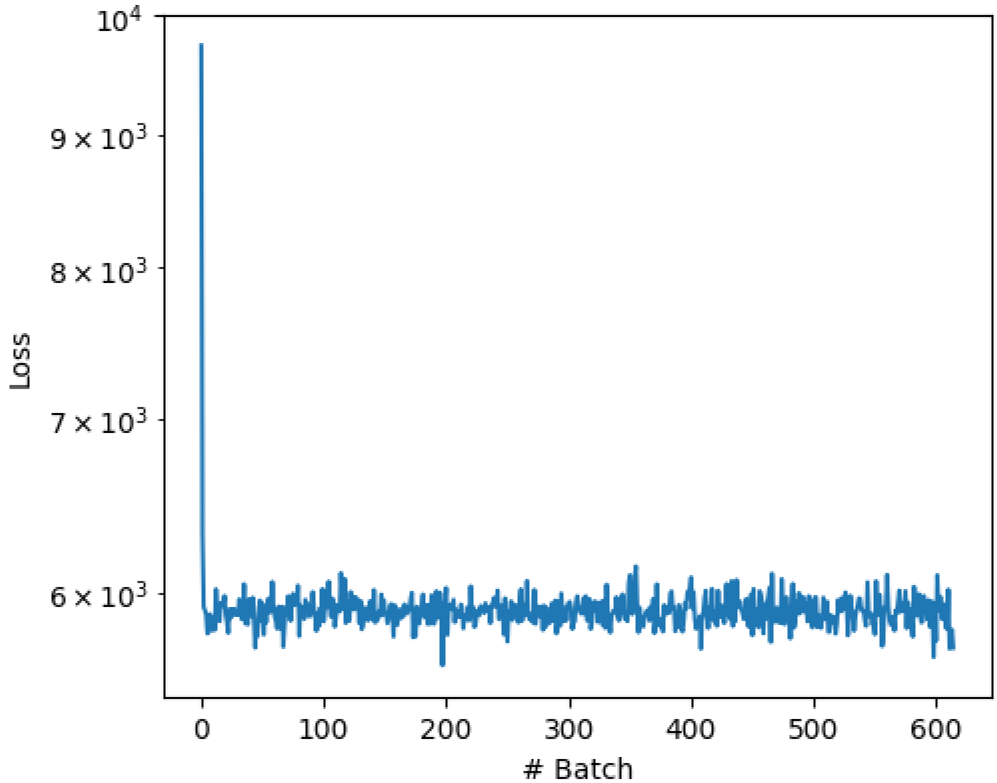
```
array(5892.8677, dtype=float32),
array(5907.1304, dtype=float32),
array(5899.263, dtype=float32),
array(5976.704, dtype=float32),
array(5882.5605, dtype=float32),
array(5853.99, dtype=float32),
array(5915.2456, dtype=float32),
array(5910.6455, dtype=float32),
array(5991.776, dtype=float32),
array(5886.0776, dtype=float32),
array(5860.787, dtype=float32),
array(5878.437, dtype=float32),
array(5850.383, dtype=float32),
array(6049.117, dtype=float32),
array(6094.3506, dtype=float32),
array(6007.5386, dtype=float32),
array(5887.134, dtype=float32),
array(5973.0728, dtype=float32),
array(6094.576, dtype=float32),
array(6141.8867, dtype=float32),
array(5922.017, dtype=float32),
array(5793.963, dtype=float32),
array(5921.1377, dtype=float32),
array(5855.9146, dtype=float32),
array(5834.985, dtype=float32),
array(5822.6226, dtype=float32),
array(5877.5024, dtype=float32),
array(5894.713, dtype=float32),
array(5907.2993, dtype=float32),
array(5894.034, dtype=float32),
array(5853.4336, dtype=float32),
array(5927.243, dtype=float32),
array(5862.5586, dtype=float32),
array(5967.788, dtype=float32),
array(5788.9707, dtype=float32),
array(5844.003, dtype=float32),
array(6045.5356, dtype=float32),
array(5844.827, dtype=float32),
array(5843.4907, dtype=float32),
array(5822.385, dtype=float32),
array(5865.374, dtype=float32),
array(5797.451, dtype=float32),
array(5935.1514, dtype=float32),
array(5886.9824, dtype=float32),
array(5918.4844, dtype=float32),
array(5920.615, dtype=float32),
array(5892.6416, dtype=float32),
array(5996.204, dtype=float32),
array(6007.784, dtype=float32),
array(5915.9014, dtype=float32),
array(5924.5933, dtype=float32),
array(5798.8223, dtype=float32),
array(5825.958, dtype=float32),
array(5820.4517, dtype=float32),
array(5935.459, dtype=float32),
array(5911.8193, dtype=float32),
```

```
            array(5814.635, dtype=float32),
            array(5796.791, dtype=float32),
            array(5852.491, dtype=float32),
            array(5947.165, dtype=float32),
            array(5915.1577, dtype=float32),
            array(6005.8135, dtype=float32),
            array(5975.208, dtype=float32),
            array(6030.122, dtype=float32),
            array(6082.6084, dtype=float32),
            array(5919.109, dtype=float32),
            array(6002.8154, dtype=float32),
            array(5864.903, dtype=float32),
            array(5811.834, dtype=float32),
            array(5846.0234, dtype=float32),
            array(5838.08, dtype=float32),
            array(5876.9106, dtype=float32),
            array(5709.3696, dtype=float32),
            array(5855.496, dtype=float32),
            array(5900.7705, dtype=float32),
            array(5973.8506, dtype=float32),
            array(6011.5786, dtype=float32),
            array(5875.2, dtype=float32),
            array(5815.4307, dtype=float32),
            array(5919.6387, dtype=float32),
            array(5862.7734, dtype=float32),
            array(6021.0093, dtype=float32),
            array(5889.587, dtype=float32),
            array(5848.0405, dtype=float32),
            array(5903.004, dtype=float32),
            array(5826.592, dtype=float32),
            array(6011.169, dtype=float32),
            array(5872.0996, dtype=float32),
            array(5881.896, dtype=float32),
            array(5932.9844, dtype=float32),
            array(5870.676, dtype=float32),
            array(5840.4805, dtype=float32),
            array(5895.482, dtype=float32),
            array(6046.6807, dtype=float32),
            array(5886.7217, dtype=float32),
            array(5926.7812, dtype=float32),
            array(5860.3906, dtype=float32),
            array(6062.728, dtype=float32),
            array(5825.547, dtype=float32),
            array(5883.6406, dtype=float32),
            array(5849.5063, dtype=float32),
            array(6068.493, dtype=float32),
            array(5921.506, dtype=float32),
            array(5942.548, dtype=float32),
            array(5954.1675, dtype=float32),
            array(5913.69, dtype=float32),
            array(5893.1577, dtype=float32),
            array(5860.6616, dtype=float32),
            array(5884.624, dtype=float32),
            array(5967.6294, dtype=float32),
            array(5871.1284, dtype=float32),
            array(5821.631, dtype=float32),
```

```
            array(5941.6045, dtype=float32),
            array(5993.083, dtype=float32),
            array(5778.1533, dtype=float32),
            array(6020.439, dtype=float32),
            array(5840.869, dtype=float32),
            array(5909.8257, dtype=float32),
            array(5910.772, dtype=float32),
            array(5818.5186, dtype=float32),
            array(6008.5366, dtype=float32),
            array(5986.6626, dtype=float32),
            array(5949.008, dtype=float32),
            array(5787.8027, dtype=float32),
            array(5937.971, dtype=float32),
            array(5864.3765, dtype=float32),
            array(5936.7334, dtype=float32),
            array(5983.462, dtype=float32),
            array(6027.7354, dtype=float32),
            array(5744.067, dtype=float32),
            array(6102.3794, dtype=float32),
            array(5865.8066, dtype=float32),
            array(5846.7817, dtype=float32),
            array(5847.5713, dtype=float32),
            array(5921.994, dtype=float32),
            array(5886.9297, dtype=float32),
            array(5843.536, dtype=float32),
            array(5841.3813, dtype=float32),
            array(6073.879, dtype=float32),
            array(5936.5176, dtype=float32),
            array(5835.6406, dtype=float32),
            array(5967.9263, dtype=float32),
            array(5976.0264, dtype=float32),
            array(5894.9697, dtype=float32),
            array(5836.995, dtype=float32),
            array(5762.739, dtype=float32),
            array(5906.419, dtype=float32),
            array(6047.458, dtype=float32),
            array(5816.941, dtype=float32),
            array(5852.1304, dtype=float32),
            array(5950.306, dtype=float32),
            array(5888.8574, dtype=float32),
            array(5999.121, dtype=float32),
            array(5891.5796, dtype=float32),
            array(5936.744, dtype=float32),
            array(5979.767, dtype=float32),
            array(5962.4204, dtype=float32),
            array(5899.944, dtype=float32),
            array(5941.856, dtype=float32),
            array(5849.2915, dtype=float32),
            array(5982.235, dtype=float32),
            array(5957.905, dtype=float32),
            array(5885.828, dtype=float32),
            array(5818.0522, dtype=float32),
            array(5956.3193, dtype=float32),
            array(5976.04, dtype=float32),
            array(5976.729, dtype=float32),
            array(5819.692, dtype=float32),
```

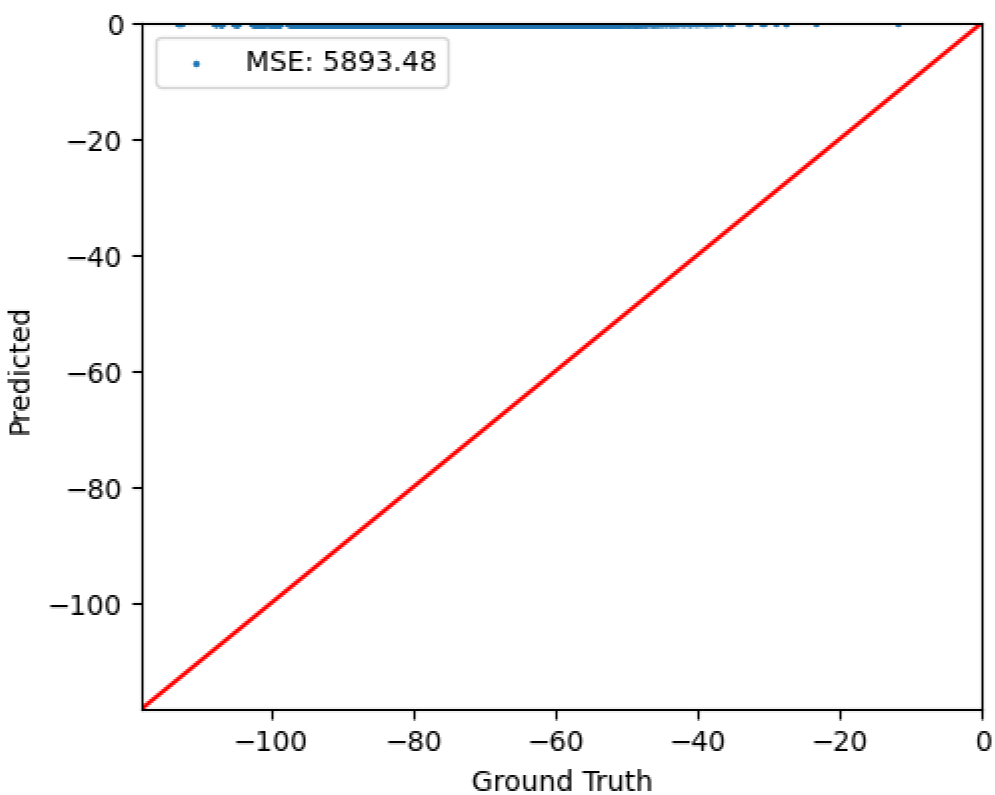
```
array(5882.9663, dtype=float32),
array(5837.867, dtype=float32),
array(5815.5244, dtype=float32),
array(5857.093, dtype=float32),
array(5974.3086, dtype=float32),
array(5862.879, dtype=float32),
array(5908.2197, dtype=float32),
array(5952.8193, dtype=float32),
array(5795.3584, dtype=float32),
array(5796.15, dtype=float32),
array(5883.5635, dtype=float32),
array(5862.265, dtype=float32),
array(5844.225, dtype=float32),
array(5873.0645, dtype=float32),
array(5786.312, dtype=float32),
array(5836.0254, dtype=float32),
array(5961.745, dtype=float32),
array(5934.4927, dtype=float32),
array(5900.5527, dtype=float32),
array(5971.455, dtype=float32),
array(5909.792, dtype=float32),
array(5965.931, dtype=float32),
array(6001.664, dtype=float32),
array(5934.0176, dtype=float32),
array(5777.41, dtype=float32),
array(5874.375, dtype=float32),
array(5984.223, dtype=float32),
array(5898.204, dtype=float32),
array(5900.6777, dtype=float32),
array(5801.228, dtype=float32),
array(5868.77, dtype=float32),
array(5906.6963, dtype=float32),
array(5933.922, dtype=float32),
array(5971.515, dtype=float32),
array(5968.745, dtype=float32),
array(5908.792, dtype=float32),
array(5909.5884, dtype=float32),
array(5895.338, dtype=float32),
array(5846.8535, dtype=float32),
array(5826.178, dtype=float32),
array(6055.9873, dtype=float32),
array(5878.679, dtype=float32),
array(5883.8677, dtype=float32),
array(5889.755, dtype=float32),
array(5833.3, dtype=float32),
array(5846.3926, dtype=float32),
array(5783.246, dtype=float32),
array(6062.243, dtype=float32),
array(5882.513, dtype=float32),
array(5943.8735, dtype=float32),
array(5989.129, dtype=float32),
array(5885.348, dtype=float32),
array(5723.286, dtype=float32),
array(5776.2144, dtype=float32),
array(5924.982, dtype=float32),
array(5913.584, dtype=float32),
```

```
                 array(6096.5894, dtype=float32),
                 array(5930.8936, dtype=float32),
                 array(5855.2666, dtype=float32),
                 array(5832.375, dtype=float32),
                 array(5837.949, dtype=float32),
                 array(5875.1904, dtype=float32),
                 array(5802.514, dtype=float32),
                 array(5907.1357, dtype=float32),
                 array(5890.121, dtype=float32),
                 array(5956.7847, dtype=float32),
                 array(6011.944, dtype=float32),
                 array(5899.332, dtype=float32),
                 array(5940.1323, dtype=float32),
                 array(5903.009, dtype=float32),
                 array(5911.8706, dtype=float32),
                 array(6009.192, dtype=float32),
                 array(5814.593, dtype=float32),
                 array(5949.674, dtype=float32),
                 array(5847.0234, dtype=float32),
                 array(5909.2573, dtype=float32),
                 array(5836.9365, dtype=float32),
                 array(5771.5215, dtype=float32),
                 array(6032.544, dtype=float32),
                 array(5949.9043, dtype=float32),
                 array(5879.146, dtype=float32),
                 array(5944.947, dtype=float32),
                 array(5992.7407, dtype=float32),
                 array(5880.576, dtype=float32),
                 array(5844.6533, dtype=float32),
                 array(5848.64, dtype=float32),
                 array(5983.8257, dtype=float32),
                 array(5906.478, dtype=float32),
                 array(5952.6504, dtype=float32),
                 array(5860.211, dtype=float32),
                 array(5960.6523, dtype=float32),
                 array(5820.9834, dtype=float32),
                 array(5855.7983, dtype=float32),
                 array(5952.442, dtype=float32),
                 array(5667.8994, dtype=float32),
                 array(5918.335, dtype=float32),
                 array(5750.4834, dtype=float32),
                 array(6094.6396, dtype=float32),
                 array(5907.0166, dtype=float32),
                 array(5980.665, dtype=float32),
                 array(5972.9824, dtype=float32),
                 array(5859.949, dtype=float32),
                 array(5911.221, dtype=float32),
                 array(5939.78, dtype=float32),
                 array(5818.5376, dtype=float32),
                 array(5810.804, dtype=float32),
                 array(6017.117, dtype=float32),
                 array(5710.3623, dtype=float32),
                 array(5783.8438, dtype=float32),
                 array(5806.9355, dtype=float32),
                 array(5714.5273, dtype=float32)]
```



In [ ]:
```
# evaulate
trainer.evaluate(test_data)
```

Out[ ]: 5893.482



It appears that the MSE is quite large, as we are always predicting a slightly positive enthalpy of formation, versus the true enthalpy of formations being negative.