Question 13.4

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt

datapath = "datasets/universal_autoencoder_samples.csv"

# Import data
X = np.loadtxt(datapath, delimiter=",")

X= X.T

# Normalize X to be between 0 and 1
X_min = X.min(axis=0)
X_max = X.max(axis=0)
X_normalized = (X - X_min) / (X_max - X_min)


# Print the shape of X
print("Shape of X:", X_normalized.shape)
```

```
Shape of X: (100, 2)
```

In [ ]:
```
def sigmoid(z, d=False):
    """Compute the sigmoid function or its derivative."""
    sig = 1 / (1 + np.exp(-z))
    if d:
        return (
            sig * (1 - sig) + 1e-12
        )  # Add epsilon to avoid division by zero in gradients
    return sig


def relu(z, d=False):
    """Compute the ReLU function or its derivative."""
    if d:
        return (
            np.where(z > 0, 1, 0) + 1e-12
        )  # Use np.where for better handling with arrays
```

```
    return np.maximum(0, z)  # np.maximum is more idiomatic and clear

layers = [
    # activation, shape:(out,in)
    {"act": relu, "shape": (1024, 2)},
    {"act": relu, "shape": (50, 1024)},
    {"act": relu, "shape": (1024, 50)},
    {"act": sigmoid, "shape": (2, 1024)},
]
```

In [ ]:

```
# Global properties
l, errors, epochs = len(layers), [], 1000
# Adam optimizer properties
lr, b1, b2 = 0.002, 0.9, 0.999
rw, mw, rb, mb = {}, {}, {}, {}

# Layer properties
a, w, b, f = {}, {}, {}, {}

# Initialize each layer's parameters
for i, layer in enumerate(layers, 1):
    n_out, n_in = layer["shape"]
    f[i] = layer["act"]

    # Xavier Initialization for weights
    w[i] = np.random.randn(n_out, n_in) / np.sqrt(n_in)

    # Initialize biases and Adam parameters (momentum and velocity) for weights and biases
    b[i], rb[i], mb[i] = (
        np.zeros((n_out, 1)),
        np.zeros((n_out, 1)),
        np.zeros((n_out, 1)),
    )
    rw[i], mw[i] = np.zeros((n_out, n_in)), np.zeros((n_out, n_in))
```

In [ ]:

```
def forward_pass(x):
    a[0] = x.T
    for i in range(1, l + 1):
        a[i] = f[i]((w[i] @ a[i - 1]) + b[i])
```

```python
def backward_pass():
    dz, dw, db = {}, {}, {}
    for i in range(l, 0, -1):
        if i == l:
            d = 0.5 * (a[l] - a[0])  # Loss derivative
        else:
            d = w[i + 1].T @ dz[i + 1]
        dz[i] = d * f[i](a[i], d=True)
        dw[i] = dz[i] @ a[i - 1].T
        db[i] = np.sum(dz[i], axis=1, keepdims=True)
    return dz, dw, db


def adam_update(m, r, z, dz):
    for i in range(1, l + 1):
        m[i] = b1 * m[i] + (1 - b1) * dz[i]
        r[i] = b2 * r[i] + (1 - b2) * (dz[i] ** 2)
        m_hat = m[i] / (1.0 - b1**t)
        r_hat = r[i] / (1.0 - b2**t)
        z[i] -= lr * m_hat / (np.sqrt(r_hat) + 1e-12)


# Main training loop
for t in range(1, epochs + 1):
    # Training
    for batch in np.split(X_normalized, 1):  # Assuming batching might need revision
        forward_pass(batch)
        dz, dw, db = backward_pass()
        adam_update(mw, rw, w, dw)
        adam_update(mb, rb, b, db)

    # Validation
    forward_pass(X_normalized)
    errors.append(np.mean((a[l] - a[0]) ** 2))
    print("Val loss - ", errors[-1])
```

```
Val loss -  0.06624673810235165
Val loss -  0.04686015767558502
Val loss -  0.039029754231910824
Val loss -  0.03792728012404295
Val loss -  0.034214800051776295
Val loss -  0.029888142696734298
Val loss -  0.025513056597087833
Val loss -  0.021977026020561466
Val loss -  0.021558577736221246
Val loss -  0.020298676274953697
Val loss -  0.019440568673512335
Val loss -  0.016871629591723
Val loss -  0.015454330900009668
Val loss -  0.01453371140972331
Val loss -  0.014763304498849598
Val loss -  0.013058417764673052
Val loss -  0.011404158731931853
Val loss -  0.01130009550253512
Val loss -  0.009829565434845216
Val loss -  0.009372690620116062
Val loss -  0.008446760148457192
Val loss -  0.007343469802949423
Val loss -  0.006474432430739179
Val loss -  0.0053924581278162135
Val loss -  0.0047500847053012964
Val loss -  0.004222049495693897
Val loss -  0.0037293031080225724
Val loss -  0.003858310887428792
Val loss -  0.003688395137119691
Val loss -  0.003538872715569996
Val loss -  0.00220316529875948
Val loss -  0.0015266455824757522
Val loss -  0.0017440421120280861
Val loss -  0.001767602996927613
Val loss -  0.0020747515778906044
Val loss -  0.000985800638854173
Val loss -  0.0006122618422780596
Val loss -  0.0007928282672170227
Val loss -  0.0011349316028844458
Val loss -  0.0015377880408856165
Val loss -  0.0009054829070704848
Val loss -  0.00047661099394303864
```

```
Val loss -  0.0005023046155864568
Val loss -  0.0007316995658554324
Val loss -  0.0011457977512834639
Val loss -  0.0007002912860613762
Val loss -  0.000446282006573656
Val loss -  0.0004706704088165742
Val loss -  0.0006429366978123677
Val loss -  0.0007301887801311888
Val loss -  0.0003698892122605904
Val loss -  0.00025768428895950356
Val loss -  0.000421470545688015
Val loss -  0.00041027384286735296
Val loss -  0.0003964678783376234
Val loss -  0.00020127829874631452
Val loss -  0.0002595453330966675
Val loss -  0.0003639045948886291
Val loss -  0.00027908688146658836
Val loss -  0.00021059415650275976
Val loss -  0.00019826601365416442
Val loss -  0.00023182089024514178
Val loss -  0.00030542977059691556
Val loss -  0.00022427267347806886
Val loss -  0.0001793825476344894
Val loss -  0.00020259172271324777
Val loss -  0.0002144202179468973
Val loss -  0.0002412594011184044
Val loss -  0.00017439468505440773
Val loss -  0.00015718301309143858
Val loss -  0.00016792714415939187
Val loss -  0.0001819208974492089
Val loss -  0.00019992178938860652
Val loss -  0.00016427213514033507
Val loss -  0.00015427513077949587
Val loss -  0.00013569031073775036
Val loss -  0.0001355812669039101
Val loss -  0.00015613394477162042
Val loss -  0.00016070009493055376
Val loss -  0.0001806778199336397
Val loss -  0.00016185564961829948
Val loss -  0.00017751278112163976
Val loss -  0.00015383818732652343
Val loss -  0.00016532374443924927
```

```
Val loss -  0.0001453052040685919
Val loss -  0.00015485656645499428
Val loss -  0.0001388215819364104
Val loss -  0.00015816965139401078
Val loss -  0.0001528964568827083
Val loss -  0.00019139234944191198
Val loss -  0.00020503217436062792
Val loss -  0.00032309943478498306
Val loss -  0.00044135281310675803
Val loss -  0.0008491533266041179
Val loss -  0.0013031205016468242
Val loss -  0.002271452686440023
Val loss -  0.002376195365280993
Val loss -  0.001975927105761075
Val loss -  0.00046904213809634076
Val loss -  0.00019414489222318126
Val loss -  0.0011064361687997861
Val loss -  0.0011894105609061335
Val loss -  0.0004782614429649229
Val loss -  0.0001645822666160586
Val loss -  0.0006895610035982971
Val loss -  0.0008048306643112662
Val loss -  0.0001789478937394392
Val loss -  0.00026093742915646496
Val loss -  0.0006705583905748658
Val loss -  0.00031630934250429204
Val loss -  0.00011599848930305352
Val loss -  0.0004185283624826333
Val loss -  0.0003658944259046594
Val loss -  0.00013362505118726394
Val loss -  0.00022026093406380377
Val loss -  0.0003245126507383279
Val loss -  0.0001986984601143673
Val loss -  0.00012268550598123094
Val loss -  0.00023893605550816336
Val loss -  0.0002451981764818981
Val loss -  0.00010775070698269415
Val loss -  0.00015285345295463083
Val loss -  0.00023806259575653423
Val loss -  0.00013509323301028658
Val loss -  0.00010743798514888012
Val loss -  0.0001876553601054995
```

```
Val loss -  0.00016049874940377454
Val loss -  0.00011153746218970267
Val loss -  0.00012877228547020128
Val loss -  0.00015320555146131355
Val loss -  0.00014166563772282587
Val loss -  0.00010257737457291308
Val loss -  0.00012066856843925655
Val loss -  0.00015088045871008816
Val loss -  0.00011155613906362664
Val loss -  0.00010083859600486798
Val loss -  0.0001245263028618425
Val loss -  0.00012066163072668841
Val loss -  0.00011409368165349154
Val loss -  9.999761802241764e-05
Val loss -  0.00010585815734734822
Val loss -  0.0001231566119277735
Val loss -  0.00010388527927093652
Val loss -  9.802379915061342e-05
Val loss -  0.0001047797145626403
Val loss -  0.00010571832664655672
Val loss -  0.0001109648737091444
Val loss -  9.664933848691116e-05
Val loss -  9.544639540689235e-05
Val loss -  0.00010313196941063994
Val loss -  0.00010023235608785645
Val loss -  0.00010393142871118754
Val loss -  9.419901180596812e-05
Val loss -  9.367312107593459e-05
Val loss -  9.83941868353251e-05
Val loss -  9.567853241824907e-05
Val loss -  0.00010079730892884616
Val loss -  9.357602493953293e-05
Val loss -  9.333044359538645e-05
Val loss -  9.312604953612932e-05
Val loss -  9.25843309195895e-05
Val loss -  9.731182969653093e-05
Val loss -  9.32407530530487e-05
Val loss -  9.587380486841453e-05
Val loss -  9.085276741817347e-05
Val loss -  9.157672364712718e-05
Val loss -  9.06739785577264e-05
Val loss -  8.992351515018648e-05
```

```
Val loss -  9.291979511535056e-05
Val loss -  9.028288145767755e-05
Val loss -  9.438732570128242e-05
Val loss -  9.042928957883077e-05
Val loss -  9.373019744950603e-05
Val loss -  8.940428375514686e-05
Val loss -  9.215037758076085e-05
Val loss -  8.815651403134966e-05
Val loss -  9.077277067726284e-05
Val loss -  8.750601466732206e-05
Val loss -  8.990819459158576e-05
Val loss -  8.727534800913147e-05
Val loss -  8.999415745911135e-05
Val loss -  8.728395439278187e-05
Val loss -  9.207281300338604e-05
Val loss -  8.925539417192562e-05
Val loss -  9.972803459951839e-05
Val loss -  9.919498919705e-05
Val loss -  0.00012657872339980926
Val loss -  0.00014248870143100098
Val loss -  0.00022918653687277072
Val loss -  0.00032593098499415627
Val loss -  0.0006424741617356163
Val loss -  0.0010608290109704818
Val loss -  0.002055006909110878
Val loss -  0.002750750681285258
Val loss -  0.003358052699033583
Val loss -  0.0016860312065717859
Val loss -  0.0002535708159488064
Val loss -  0.0004737468702782444
Val loss -  0.0013952950588302945
Val loss -  0.0011737227692891377
Val loss -  0.0001305796190330806
Val loss -  0.0005268557550381729
Val loss -  0.0010424994434199645
Val loss -  0.00024369174819731477
Val loss -  0.0002594758707474897
Val loss -  0.0007791714576651528
Val loss -  0.0002446351668744806
Val loss -  0.00017748912495902324
Val loss -  0.0005661837683084061
Val loss -  0.0001942173385637148
```

```
Val loss -  0.0001582824516952445
Val loss -  0.0004299669611050696
Val loss -  0.00014626022821794213
Val loss -  0.00015223799678984953
Val loss -  0.0003357436216870552
Val loss -  0.00011662098593272906
Val loss -  0.00015048264603448834
Val loss -  0.00026823561863133656
Val loss -  9.802027700759663e-05
Val loss -  0.0001423362052251937
Val loss -  0.00021885216810506268
Val loss -  8.92648243837913e-05
Val loss -  0.00013387997920175361
Val loss -  0.00018164697640716602
Val loss -  8.513141268372081e-05
Val loss -  0.00012446191969418789
Val loss -  0.00015428315696010492
Val loss -  8.393936264489585e-05
Val loss -  0.00011573104414824838
Val loss -  0.00013620135555042593
Val loss -  8.34314053377847e-05
Val loss -  0.00010660610105530429
Val loss -  0.0001223291378777353
Val loss -  8.21375752228857e-05
Val loss -  9.833058857227391e-05
Val loss -  0.00011242765913344471
Val loss -  8.129717030798469e-05
Val loss -  9.181238528238222e-05
Val loss -  0.00010461322869726314
Val loss -  8.021943693635073e-05
Val loss -  8.675390009616275e-05
Val loss -  9.894835429737039e-05
Val loss -  8.004917648521715e-05
Val loss -  8.262321353898462e-05
Val loss -  9.380687051414844e-05
Val loss -  7.99770173846305e-05
Val loss -  7.889630475981021e-05
Val loss -  8.859801516626773e-05
Val loss -  8.038302434462811e-05
Val loss -  7.70990548844829e-05
Val loss -  8.314879931155848e-05
Val loss -  8.052888192793987e-05
```

```
Val loss -  7.711326029250561e-05
Val loss -  7.812223798870856e-05
Val loss -  7.911007230107056e-05
Val loss -  7.848158081110632e-05
Val loss -  7.51781050830937e-05
Val loss -  7.609074393007009e-05
Val loss -  7.862710164051755e-05
Val loss -  7.426753599141224e-05
Val loss -  7.359707452874058e-05
Val loss -  7.71619540155415e-05
Val loss -  7.414920623524665e-05
Val loss -  7.311625643132673e-05
Val loss -  7.453270804319433e-05
Val loss -  7.325112486563777e-05
Val loss -  7.397567046596433e-05
Val loss -  7.232329696988966e-05
Val loss -  7.174369536246944e-05
Val loss -  7.378914015233015e-05
Val loss -  7.106023927703935e-05
Val loss -  7.106562186124582e-05
Val loss -  7.191316953889059e-05
Val loss -  7.048833231747266e-05
Val loss -  7.140917955265907e-05
Val loss -  6.989549472931048e-05
Val loss -  7.001499401367994e-05
Val loss -  7.080242790192108e-05
Val loss -  6.943095429779529e-05
Val loss -  6.956141117651956e-05
Val loss -  6.927157826550899e-05
Val loss -  6.908077970511166e-05
Val loss -  6.933597511238591e-05
Val loss -  6.856654617796429e-05
Val loss -  6.791950635188643e-05
Val loss -  6.838621004266865e-05
Val loss -  6.740719141896727e-05
Val loss -  6.811289217653479e-05
Val loss -  6.719495328828553e-05
Val loss -  6.706117916145752e-05
Val loss -  6.734783942370235e-05
Val loss -  6.652000019881708e-05
Val loss -  6.725063899480631e-05
Val loss -  6.616391099149297e-05
```

```
Val loss -  6.637695952850657e-05
Val loss -  6.603579260554385e-05
Val loss -  6.590469825987569e-05
Val loss -  6.596806225109478e-05
Val loss -  6.604969380121546e-05
Val loss -  6.566671610697841e-05
Val loss -  6.71586447598484e-05
Val loss -  6.610039095565501e-05
Val loss -  7.032374203916283e-05
Val loss -  6.899283840090458e-05
Val loss -  7.760541840545948e-05
Val loss -  7.799398866426498e-05
Val loss -  9.496622605082075e-05
Val loss -  9.994714763431792e-05
Val loss -  0.00013248565693152164
Val loss -  0.00014513802161529002
Val loss -  0.00019850384732963099
Val loss -  0.00020741159310206086
Val loss -  0.000252764093222867
Val loss -  0.00020666702245398385
Val loss -  0.00018449469151993748
Val loss -  0.00010726139022690334
Val loss -  8.10242019027031e-05
Val loss -  7.266324209359245e-05
Val loss -  8.814546901358177e-05
Val loss -  0.00012087086784806751
Val loss -  0.00011276428197771485
Val loss -  0.00012297583631410238
Val loss -  9.36493202254195e-05
Val loss -  9.155787225178269e-05
Val loss -  7.781463939077657e-05
Val loss -  7.712310128448236e-05
Val loss -  7.787495316562463e-05
Val loss -  7.463422794873945e-05
Val loss -  8.731584566770104e-05
Val loss -  8.467350050627494e-05
Val loss -  0.00010079790221934786
Val loss -  9.163780852608428e-05
Val loss -  9.764045712818708e-05
Val loss -  7.956512925071645e-05
Val loss -  7.686634052679576e-05
Val loss -  6.297877203770668e-05
```

```
Val loss -  6.22422263832658e-05
Val loss -  6.018272428503852e-05
Val loss -  6.245900653584063e-05
Val loss -  6.592946666443536e-05
Val loss -  6.543376808796053e-05
Val loss -  6.866471471208931e-05
Val loss -  6.461631943888769e-05
Val loss -  6.87001354000123e-05
Val loss -  6.396576984241467e-05
Val loss -  7.352353228024463e-05
Val loss -  7.238319379314017e-05
Val loss -  9.652475927699692e-05
Val loss -  0.00011018839048394705
Val loss -  0.00017597248459933236
Val loss -  0.00023884086727440007
Val loss -  0.00042012234369084313
Val loss -  0.0005678275907733194
Val loss -  0.0008948349331194277
Val loss -  0.0009284569867999292
Val loss -  0.0010545284292685165
Val loss -  0.0008754243888005738
Val loss -  0.0008796615841195602
Val loss -  0.0007080405258367131
Val loss -  0.0003895204371841689
Val loss -  0.00014105239123103766
Val loss -  0.00024041122518744803
Val loss -  0.0005255772009961842
Val loss -  0.00042792286741054325
Val loss -  0.00012244938699499582
Val loss -  0.00010876380535798771
Val loss -  0.00031058577813290083
Val loss -  0.00028650110271020133
Val loss -  0.00010930056233762674
Val loss -  0.00012479720986782234
Val loss -  0.00019497793537419982
Val loss -  0.0001445250289366428
Val loss -  0.00010568570392385336
Val loss -  0.00012756314611814108
Val loss -  0.00012780044541360987
Val loss -  8.03891609889587e-05
Val loss -  9.276727108588359e-05
Val loss -  0.0001381180480923657
```

```
Val loss -  8.446892915966977e-05
Val loss -  5.976112071967698e-05
Val loss -  9.853190638045014e-05
Val loss -  9.511165435984157e-05
Val loss -  7.675900135473151e-05
Val loss -  7.094780052239525e-05
Val loss -  8.103525643282651e-05
Val loss -  7.435467922533367e-05
Val loss -  6.686019218411815e-05
Val loss -  8.540904989811162e-05
Val loss -  7.411977875338479e-05
Val loss -  5.65872382877617e-05
Val loss -  6.567897527990755e-05
Val loss -  7.622605259331215e-05
Val loss -  7.275145063335262e-05
Val loss -  5.7638901487014526e-05
Val loss -  6.334141740186194e-05
Val loss -  6.295550639730711e-05
Val loss -  5.733968038105336e-05
Val loss -  5.834297296712537e-05
Val loss -  6.007396772423689e-05
Val loss -  6.405360037925013e-05
Val loss -  5.514584464006873e-05
Val loss -  5.665198150055352e-05
Val loss -  5.802784177130942e-05
Val loss -  5.591154882256311e-05
Val loss -  5.389780252488257e-05
Val loss -  5.310115694023823e-05
Val loss -  5.774960608747605e-05
Val loss -  5.604859905255146e-05
Val loss -  5.5901748887134176e-05
Val loss -  5.259834781813852e-05
Val loss -  5.615527522534165e-05
Val loss -  5.34619736457582e-05
Val loss -  5.4706702449740995e-05
Val loss -  5.1465966853932246e-05
Val loss -  5.443437184381032e-05
Val loss -  5.3015343257978824e-05
Val loss -  5.480471867443705e-05
Val loss -  5.169323740801543e-05
Val loss -  5.561757495619646e-05
Val loss -  5.349819061136426e-05
```

```
Val loss -  6.1009782249610375e-05
Val loss -  5.94301442238042e-05
Val loss -  7.381865720767515e-05
Val loss -  7.795920719975076e-05
Val loss -  0.00010900864243641326
Val loss -  0.0001263186690694547
Val loss -  0.00019325905522922567
Val loss -  0.00023859586608221606
Val loss -  0.0003731390051306917
Val loss -  0.0004452731160192047
Val loss -  0.0006159729143856339
Val loss -  0.0006107897867604607
Val loss -  0.0006392063040708134
Val loss -  0.0004181437390142459
Val loss -  0.00024100189901607062
Val loss -  7.202184592356582e-05
Val loss -  6.510845846133006e-05
Val loss -  0.00018061708076947604
Val loss -  0.00026043277096216987
Val loss -  0.00026989248013246274
Val loss -  0.00014612588113244705
Val loss -  6.303591897707591e-05
Val loss -  6.910394691527913e-05
Val loss -  0.00012616817269778718
Val loss -  0.00018063330716192486
Val loss -  0.00012522177337359773
Val loss -  7.734235571161749e-05
Val loss -  5.402648830844717e-05
Val loss -  7.574040360487447e-05
Val loss -  0.0001132974695817194
Val loss -  9.956315851980095e-05
Val loss -  8.804134941608184e-05
Val loss -  5.4872431179112155e-05
Val loss -  5.20840793828122e-05
Val loss -  6.837186768946162e-05
Val loss -  7.349539756616795e-05
Val loss -  8.415536815072143e-05
Val loss -  6.422891086873434e-05
Val loss -  5.5255554228569084e-05
Val loss -  4.729812588900433e-05
Val loss -  4.9479621283540603e-05
Val loss -  6.35364956800775e-05
```

```
Val loss -  6.21133471289937e-05
Val loss -  6.893869629167477e-05
Val loss -  5.518397476631265e-05
Val loss -  5.2394960145416834e-05
Val loss -  4.576112024207663e-05
Val loss -  4.5757236746481674e-05
Val loss -  5.0825213759953235e-05
Val loss -  5.0549436748890295e-05
Val loss -  5.900015889860082e-05
Val loss -  5.510098160624417e-05
Val loss -  6.217707725673309e-05
Val loss -  5.5140022983823514e-05
Val loss -  5.784646284019714e-05
Val loss -  4.941315191693446e-05
Val loss -  4.960278464118579e-05
Val loss -  4.446763366501667e-05
Val loss -  4.4735309345604204e-05
Val loss -  4.365390985859002e-05
Val loss -  4.377691247581855e-05
Val loss -  4.4938362042646916e-05
Val loss -  4.4055787536667645e-05
Val loss -  4.705512723875661e-05
Val loss -  4.580409036505309e-05
Val loss -  5.268595407753877e-05
Val loss -  5.284004490557138e-05
Val loss -  7.051352944962864e-05
Val loss -  7.815722205182157e-05
Val loss -  0.00012458418159948004
Val loss -  0.0001627686439695205
Val loss -  0.00028232269186570324
Val loss -  0.0003914593036365221
Val loss -  0.000641553381913526
Val loss -  0.0007707049633063847
Val loss -  0.0009832954965095164
Val loss -  0.0007698431142254677
Val loss -  0.0005339681108865857
Val loss -  0.00018210965173712408
Val loss -  9.54369396599376e-05
Val loss -  0.000245356473666162
Val loss -  0.00033300143790385714
Val loss -  0.0002887409375812913
Val loss -  0.0001299907980157686
```

```
Val loss -  9.711681238987026e-05
Val loss -  0.00019885550839899159
Val loss -  0.0001981807248706463
Val loss -  0.0001617661568407252
Val loss -  0.00010202471863161449
Val loss -  0.00010325677431388049
Val loss -  0.00014232086723552407
Val loss -  0.00011544182088332866
Val loss -  0.0001040758263617467
Val loss -  9.988229780106398e-05
Val loss -  9.381855869210063e-05
Val loss -  9.014328647712594e-05
Val loss -  6.425973777909388e-05
Val loss -  8.007282658554018e-05
Val loss -  9.747192543223005e-05
Val loss -  7.73573423231605e-05
Val loss -  5.6569518656789256e-05
Val loss -  5.1083322750437e-05
Val loss -  7.146048358178911e-05
Val loss -  7.963133136598924e-05
Val loss -  5.839469067360542e-05
Val loss -  4.442380656095304e-05
Val loss -  4.940529121417573e-05
Val loss -  6.275428511588463e-05
Val loss -  6.706059931734812e-05
Val loss -  5.0387982240609545e-05
Val loss -  4.1589344001837295e-05
Val loss -  4.431671658306446e-05
Val loss -  5.237087026238183e-05
Val loss -  5.869903754821595e-05
Val loss -  5.100648000136116e-05
Val loss -  4.402208042260266e-05
Val loss -  3.970388577814387e-05
Val loss -  4.2247033583987205e-05
Val loss -  4.8249909160256626e-05
Val loss -  4.7575939609052176e-05
Val loss -  4.511214932493918e-05
Val loss -  4.005677309783203e-05
Val loss -  3.871964139970353e-05
Val loss -  4.1210566531776346e-05
Val loss -  4.292439527622078e-05
Val loss -  4.510525601745299e-05
```

```
Val loss -  4.245324614487931e-05
Val loss -  4.0655442010669415e-05
Val loss -  3.7804557747790105e-05
Val loss -  3.7671709376315276e-05
Val loss -  3.884613276615669e-05
Val loss -  4.0354734711679754e-05
Val loss -  4.173863812947847e-05
Val loss -  4.12151401033262e-05
Val loss -  4.024734294808426e-05
Val loss -  3.8039017025904e-05
Val loss -  3.70446201327392e-05
Val loss -  3.680839371123529e-05
Val loss -  3.684388497900227e-05
Val loss -  3.8070500808356285e-05
Val loss -  3.797856769116917e-05
Val loss -  4.0339763799666196e-05
Val loss -  3.933235150400669e-05
Val loss -  4.3627468572479504e-05
Val loss -  4.269128012749428e-05
Val loss -  5.477600047603727e-05
Val loss -  6.284123977944722e-05
Val loss -  0.00010398088291482298
Val loss -  0.0001371643576899029
Val loss -  0.00023843011815098916
Val loss -  0.0003392682129307606
Val loss -  0.0005451635480989728
Val loss -  0.0006463883811970859
Val loss -  0.0007719912210590194
Val loss -  0.0005703408160793619
Val loss -  0.0003535853964612835
Val loss -  8.531855689159379e-05
Val loss -  6.295165085525458e-05
Val loss -  0.0002193069874859031
Val loss -  0.0003164755508080698
Val loss -  0.0002903901490340341
Val loss -  0.00010055752748354172
Val loss -  4.149469585357574e-05
Val loss -  0.0001262654170668253
Val loss -  0.00018799076250977187
Val loss -  0.00017432569918338209
Val loss -  7.557600853521658e-05
Val loss -  4.259290613820814e-05
```

```
Val loss -  9.627836197361489e-05
Val loss -  0.00012810199517244454
Val loss -  0.00012775510434784342
Val loss -  5.797523601509185e-05
Val loss -  3.767765260622822e-05
Val loss -  6.442757178665238e-05
Val loss -  8.715396448008794e-05
Val loss -  9.24954346156026e-05
Val loss -  5.789507334996646e-05
Val loss -  3.962439846189457e-05
Val loss -  4.759326297967086e-05
Val loss -  6.038280361270041e-05
Val loss -  7.410017464606821e-05
Val loss -  5.1385273612825305e-05
Val loss -  4.022657176273359e-05
Val loss -  3.73124939014928e-05
Val loss -  4.658823378548428e-05
Val loss -  5.932585658382283e-05
Val loss -  5.096806174252356e-05
Val loss -  4.4288982892083296e-05
Val loss -  3.369042707558267e-05
Val loss -  3.502030260989498e-05
Val loss -  4.3900306633233036e-05
Val loss -  4.4925627279036013e-05
Val loss -  4.748363186124953e-05
Val loss -  3.854534184442724e-05
Val loss -  3.489697892858932e-05
Val loss -  3.296240297279295e-05
Val loss -  3.385847285652128e-05
Val loss -  3.9767019878868646e-05
Val loss -  3.802952147996149e-05
Val loss -  3.995318620378025e-05
Val loss -  3.4060507911943314e-05
Val loss -  3.290863318219645e-05
Val loss -  3.162752626573589e-05
Val loss -  3.165431553425107e-05
Val loss -  3.518232044869242e-05
Val loss -  3.420854857394598e-05
Val loss -  3.792524741062821e-05
Val loss -  3.467416048177012e-05
Val loss -  3.807477997821014e-05
Val loss -  3.5037881334855685e-05
```

```
Val loss -  4.253555860602064e-05
Val loss -  4.446959178496907e-05
Val loss -  6.278818780233293e-05
Val loss -  7.602899507945349e-05
Val loss -  0.00011846179387880288
Val loss -  0.00015054234325433832
Val loss -  0.00022068522709858227
Val loss -  0.00025123668546342545
Val loss -  0.00030713915227379937
Val loss -  0.00026207011774015453
Val loss -  0.000204148208768943
Val loss -  8.598719634185043e-05
Val loss -  3.334627647302772e-05
Val loss -  5.863099235307583e-05
Val loss -  0.00010154211379254712
Val loss -  0.00013435184075918385
Val loss -  7.943331227383549e-05
Val loss -  3.947183302700303e-05
Val loss -  3.7255122737856054e-05
Val loss -  6.63636177638582e-05
Val loss -  8.603596927278669e-05
Val loss -  5.637693485195387e-05
Val loss -  3.476549314105396e-05
Val loss -  3.80180935043132e-05
Val loss -  6.0507546952902215e-05
Val loss -  6.971407164781622e-05
Val loss -  5.4100090727927775e-05
Val loss -  3.698454697370785e-05
Val loss -  4.464257309108902e-05
Val loss -  5.087533544126987e-05
Val loss -  6.385292069093717e-05
Val loss -  4.9324816312844466e-05
Val loss -  5.57683892854784e-05
Val loss -  5.9177700072384414e-05
Val loss -  8.763742309111167e-05
Val loss -  9.976701420386892e-05
Val loss -  0.000120301214547093
Val loss -  0.00012956238227024473
Val loss -  0.00018759010132766984
Val loss -  0.00020923464246465302
Val loss -  0.00028952455445082633
Val loss -  0.0003107908407163292
```

```
Val loss -  0.000367588040983791
Val loss -  0.0003040490233870544
Val loss -  0.00024436235555389836
Val loss -  0.00011601299184651146
Val loss -  4.540901020007605e-05
Val loss -  3.504524738780123e-05
Val loss -  7.8111113121389e-05
Val loss -  0.00013758385498054251
Val loss -  0.0001622410754767358
Val loss -  0.00018608472834602045
Val loss -  9.49831441545981e-05
Val loss -  5.8989804077315724e-05
Val loss -  3.335183321902885e-05
Val loss -  5.247052208345277e-05
Val loss -  8.7447849346492e-05
Val loss -  9.576965255970153e-05
Val loss -  0.00010204881643958453
Val loss -  5.718103599936151e-05
Val loss -  4.200502103138751e-05
Val loss -  2.669176551557286e-05
Val loss -  3.39482303659751e-05
Val loss -  5.4445531615587155e-05
Val loss -  6.073300677641787e-05
Val loss -  7.164909313413489e-05
Val loss -  4.671379288310129e-05
Val loss -  4.1475926338014466e-05
Val loss -  2.6643044127572136e-05
Val loss -  2.5610922201989642e-05
Val loss -  3.362293620085595e-05
Val loss -  3.667664599523284e-05
Val loss -  5.0095718952429655e-05
Val loss -  4.4391220844615994e-05
Val loss -  5.184635604057595e-05
Val loss -  4.0816787975212833e-05
Val loss -  4.026051222278366e-05
Val loss -  3.059839739343883e-05
Val loss -  2.7980183465041766e-05
Val loss -  2.3292051264799866e-05
Val loss -  2.3613740668548018e-05
Val loss -  2.6092247590111235e-05
Val loss -  2.6093655270740066e-05
Val loss -  3.080703697298736e-05
```

```
Val loss -  2.871079314652915e-05
Val loss -  3.712135755839498e-05
Val loss -  3.4130913968576576e-05
Val loss -  4.645050275440194e-05
Val loss -  4.733981881817255e-05
Val loss -  6.304230309148803e-05
Val loss -  6.391599941952186e-05
Val loss -  9.388398820279924e-05
Val loss -  9.72018750160342e-05
Val loss -  0.00014004021621167535
Val loss -  0.00015608735354355715
Val loss -  0.00021532167399150663
Val loss -  0.00023371446742835166
Val loss -  0.00028634736123241264
Val loss -  0.0002367609213926356
Val loss -  0.0002160993912476107
Val loss -  0.00011726352518241464
Val loss -  5.931821894104421e-05
Val loss -  3.199077061631706e-05
Val loss -  4.3174074366136604e-05
Val loss -  8.110682115074898e-05
Val loss -  9.691691536318002e-05
Val loss -  0.00013749179629933398
Val loss -  8.553328597561779e-05
Val loss -  6.76797147735906e-05
Val loss -  3.321447439894606e-05
Val loss -  2.4213544483085853e-05
Val loss -  3.377713267480139e-05
Val loss -  4.305004588625622e-05
Val loss -  7.001371206893639e-05
Val loss -  5.680276048830003e-05
Val loss -  6.453006549390909e-05
Val loss -  4.5204823918329724e-05
Val loss -  3.949153309907245e-05
Val loss -  2.971621931269226e-05
Val loss -  2.950424134908816e-05
Val loss -  2.857181299723134e-05
Val loss -  2.6557772728707613e-05
Val loss -  2.6829491593327745e-05
Val loss -  2.356339087293156e-05
Val loss -  3.1259355573617554e-05
Val loss -  3.433861882075721e-05
```

```
Val loss -  4.829496020663461e-05
Val loss -  5.3909760980169214e-05
Val loss -  7.25324146423094e-05
Val loss -  7.362316145463601e-05
Val loss -  9.46449443141572e-05
Val loss -  8.98248182947817e-05
Val loss -  0.00010699256690577667
Val loss -  9.493413296771896e-05
Val loss -  0.00010266257181078123
Val loss -  8.422209967451138e-05
Val loss -  8.978369972937071e-05
Val loss -  6.62168255779694e-05
Val loss -  7.242725803680846e-05
Val loss -  5.5649047811153225e-05
Val loss -  5.9460948783464015e-05
Val loss -  4.4856848515468275e-05
Val loss -  4.848754645067791e-05
Val loss -  3.470812862700642e-05
Val loss -  3.816115486193186e-05
Val loss -  2.8412105890612002e-05
Val loss -  3.61934025071303e-05
Val loss -  3.598286336197448e-05
Val loss -  5.587885355289838e-05
Val loss -  6.106927516614621e-05
Val loss -  8.831028690824364e-05
Val loss -  9.43651770064573e-05
Val loss -  0.00012095885012246674
Val loss -  0.00012665046622406715
Val loss -  0.0001547084927953688
Val loss -  0.00013174033314302302
Val loss -  0.00014293177588163203
Val loss -  0.00012325926389162244
Val loss -  0.00014197820481080565
Val loss -  0.0001288347603822426
Val loss -  0.0001396422959487462
Val loss -  0.00010882862535096576
Val loss -  8.198799744960437e-05
Val loss -  4.2016291970191736e-05
Val loss -  2.415140521383059e-05
Val loss -  2.2701487598592662e-05
Val loss -  3.0886505577671994e-05
Val loss -  5.750438314326927e-05
```

```
Val loss -  6.018237036491933e-05
Val loss -  7.589162511688085e-05
Val loss -  6.377703543819437e-05
Val loss -  7.851535100593195e-05
Val loss -  8.082274125713443e-05
Val loss -  0.00010614924992385894
Val loss -  0.00010864281103023128
Val loss -  0.00011513808835717271
Val loss -  7.907045448606867e-05
Val loss -  5.1538555270511964e-05
Val loss -  2.380106132358238e-05
Val loss -  1.882577075215771e-05
Val loss -  2.7044474146705805e-05
Val loss -  4.031682301543803e-05
Val loss -  5.786185007626162e-05
Val loss -  6.510891460445784e-05
Val loss -  8.390941304305236e-05
Val loss -  7.732354738527768e-05
Val loss -  9.604966962629247e-05
Val loss -  9.143851002499793e-05
Val loss -  0.00010373794522919763
Val loss -  9.428189290863433e-05
Val loss -  9.61399343507054e-05
Val loss -  6.299892624992146e-05
Val loss -  4.822125105008999e-05
Val loss -  2.7259905621092292e-05
Val loss -  2.8108642209385852e-05
Val loss -  2.642450269749044e-05
Val loss -  3.427866316586506e-05
Val loss -  3.573319672628077e-05
Val loss -  3.637647866664783e-05
Val loss -  2.9971959194514796e-05
Val loss -  2.608119870513423e-05
Val loss -  2.001164999970783e-05
Val loss -  2.5171220407757387e-05
Val loss -  2.7420874539380104e-05
Val loss -  4.0083386896434977e-05
Val loss -  4.374000938488796e-05
Val loss -  7.348630740521007e-05
Val loss -  8.359191912501649e-05
Val loss -  0.00013466286222192265
Val loss -  0.0001667633570543013
```

```
Val loss -  0.00023721792422425343
Val loss -  0.0002515755302497584
Val loss -  0.0002747712950479883
Val loss -  0.00022761110172092082
Val loss -  0.00018678350851379766
Val loss -  8.310564497234203e-05
Val loss -  3.598167978764923e-05
Val loss -  2.4713236534971665e-05
Val loss -  3.703626167775993e-05
Val loss -  7.73443259045165e-05
Val loss -  7.409852717368252e-05
Val loss -  7.822404378199955e-05
Val loss -  7.307560549744976e-05
Val loss -  7.687958427482714e-05
Val loss -  4.3487813199999484e-05
Val loss -  3.082802749103625e-05
Val loss -  2.2296711315357413e-05
Val loss -  2.943235798501655e-05
Val loss -  3.9177037238883677e-05
Val loss -  4.9821812130139786e-05
Val loss -  8.957821492402838e-05
Val loss -  5.2510413122223764e-05
Val loss -  4.629399243044982e-05
Val loss -  3.345245520694514e-05
Val loss -  2.21524684375685e-05
Val loss -  1.5684840985326748e-05
Val loss -  1.5307763295112115e-05
Val loss -  2.4543787660742797e-05
Val loss -  3.4342270336547714e-05
Val loss -  6.351565096697073e-05
Val loss -  8.581829447974313e-05
Val loss -  0.00013840052708488136
Val loss -  0.00010225433160476832
Val loss -  7.681191829595948e-05
Val loss -  4.237187049771151e-05
Val loss -  2.2468550268520503e-05
Val loss -  1.6166711924207013e-05
Val loss -  2.0814715412332112e-05
Val loss -  3.522773055386905e-05
Val loss -  4.6148054916095406e-05
Val loss -  7.070522175518206e-05
Val loss -  6.642223751126542e-05
```

```
Val loss -  6.999997550243143e-05
Val loss -  4.97358252052198e-05
Val loss -  4.2189343252577565e-05
Val loss -  2.927021977934073e-05
Val loss -  3.34827368114176e-05
Val loss -  3.4567099308507465e-05
Val loss -  3.842577149557127e-05
Val loss -  3.8642572440349456e-05
Val loss -  4.012217544861699e-05
Val loss -  4.9054365541896884e-05
Val loss -  3.617317542843557e-05
Val loss -  3.4121091940882426e-05
Val loss -  3.818175812758455e-05
Val loss -  6.376006380580427e-05
Val loss -  5.1241747755765394e-05
Val loss -  6.44253561066312e-05
Val loss -  7.459278344060218e-05
Val loss -  0.00010905864530727761
Val loss -  0.00010342720302391427
Val loss -  0.00011246411481910357
Val loss -  0.00010517429459475149
Val loss -  0.0001323081745292796
Val loss -  8.158643379522332e-05
Val loss -  7.003947067678972e-05
Val loss -  3.52857134674292e-05
Val loss -  3.0747741955649374e-05
Val loss -  1.7565605077227165e-05
Val loss -  2.990346138948545e-05
Val loss -  3.160114442015781e-05
Val loss -  3.350087948843573e-05
Val loss -  3.573995302458915e-05
Val loss -  3.258887260097258e-05
Val loss -  2.744249011691433e-05
Val loss -  1.942858961315874e-05
Val loss -  2.133828530167572e-05
Val loss -  1.7664300942815927e-05
Val loss -  2.780931704738724e-05
Val loss -  3.83365530187068e-05
Val loss -  7.12550535855972e-05
Val loss -  7.416197922529999e-05
Val loss -  9.170150490410072e-05
Val loss -  0.00012818143850243762
```

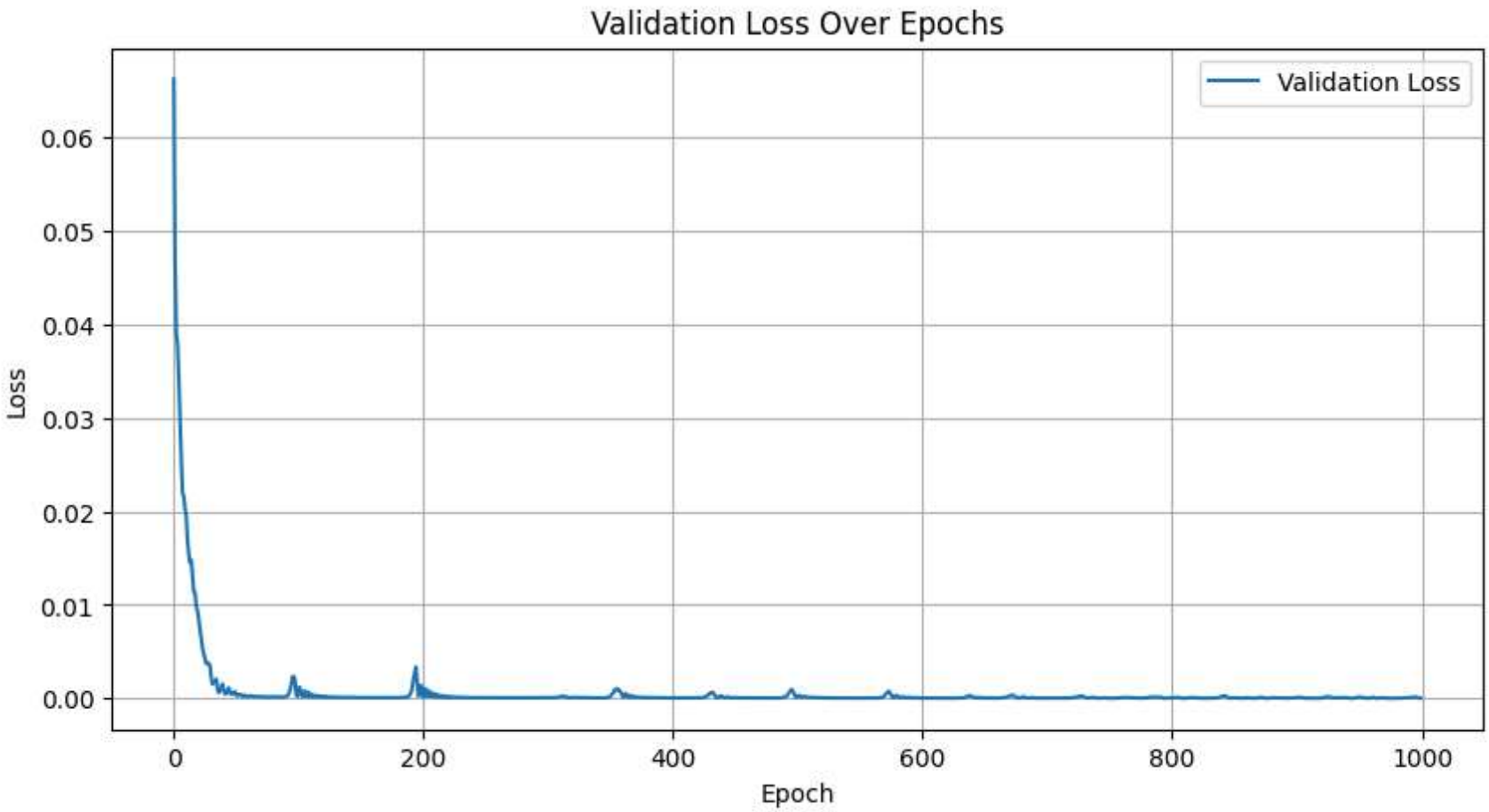
```
Val loss -  0.0002143783161501656
Val loss -  0.0001629983283502016
Val loss -  0.00014697435546482456
Val loss -  0.00011550498059848358
Val loss -  0.00010122980027564534
Val loss -  7.056835048746909e-05
Val loss -  7.675386135309068e-05
Val loss -  6.206123262095708e-05
Val loss -  5.9713399153866056e-05
Val loss -  5.38636264969005e-05
Val loss -  4.471971166711472e-05
Val loss -  5.6297609481280494e-05
Val loss -  6.0639847063552086e-05
Val loss -  0.00010569105971686966
Val loss -  8.091659624201469e-05
Val loss -  8.261437721386017e-05
Val loss -  6.913855560013526e-05
Val loss -  5.0087203531401914e-05
Val loss -  2.3294487982745833e-05
Val loss -  1.6164836444764393e-05
Val loss -  1.5716699783997462e-05
Val loss -  2.073848945088215e-05
Val loss -  4.408760695085919e-05
Val loss -  5.5892570100871316e-05
Val loss -  8.176561804650356e-05
Val loss -  0.00010191612494296565
Val loss -  0.00015555607997825717
Val loss -  0.00011232189271000531
Val loss -  8.958908394286973e-05
Val loss -  8.136928623139422e-05
Val loss -  7.702852621787138e-05
Val loss -  3.591112146848649e-05
Val loss -  2.073255596107852e-05
Val loss -  1.9888494390984623e-05
Val loss -  4.800133916941604e-05
Val loss -  9.184435350073516e-05
Val loss -  0.00010476588482708105
Val loss -  0.00011066718819492
Val loss -  6.427022837608866e-05
Val loss -  3.761242647195669e-05
Val loss -  1.5179532335813336e-05
Val loss -  2.3117595590859498e-05
```

```
Val loss -  4.445187763077148e-05
Val loss -  6.059814131483816e-05
Val loss -  8.466423313985946e-05
Val loss -  5.6389737187949924e-05
Val loss -  4.2757123178549863e-05
Val loss -  3.494187601062149e-05
Val loss -  3.6014450192332616e-05
Val loss -  2.4033238438684587e-05
Val loss -  2.7182027164829483e-05
Val loss -  1.833317815808084e-05
Val loss -  1.9205416608681147e-05
Val loss -  1.2377906086110696e-05
Val loss -  1.596751016448582e-05
Val loss -  1.3718939315196588e-05
Val loss -  2.1094898892234608e-05
Val loss -  1.9171943736443898e-05
Val loss -  2.4778141176067143e-05
Val loss -  3.164365112339812e-05
Val loss -  5.5897349141236976e-05
Val loss -  4.737950347488938e-05
Val loss -  5.048377757546158e-05
Val loss -  5.883718604343016e-05
Val loss -  8.179894916834453e-05
Val loss -  6.97441463847295e-05
Val loss -  8.315797889679288e-05
Val loss -  8.430169572705009e-05
Val loss -  0.00010256509640569136
Val loss -  0.00011640592758260835
Val loss -  0.00017773532722420603
Val loss -  0.0001320772133022645
Val loss -  0.00011319762560459284
Val loss -  7.467116416856815e-05
Val loss -  3.6221124518868074e-05
Val loss -  2.5490895537533644e-05
```

In [ ]:

```python
# Import matplotlib
import matplotlib.pyplot as plt

# Plot the validation errors
plt.figure(figsize=(10, 5))
plt.plot(errors, label="Validation Loss")
plt.title("Validation Loss Over Epochs")
```

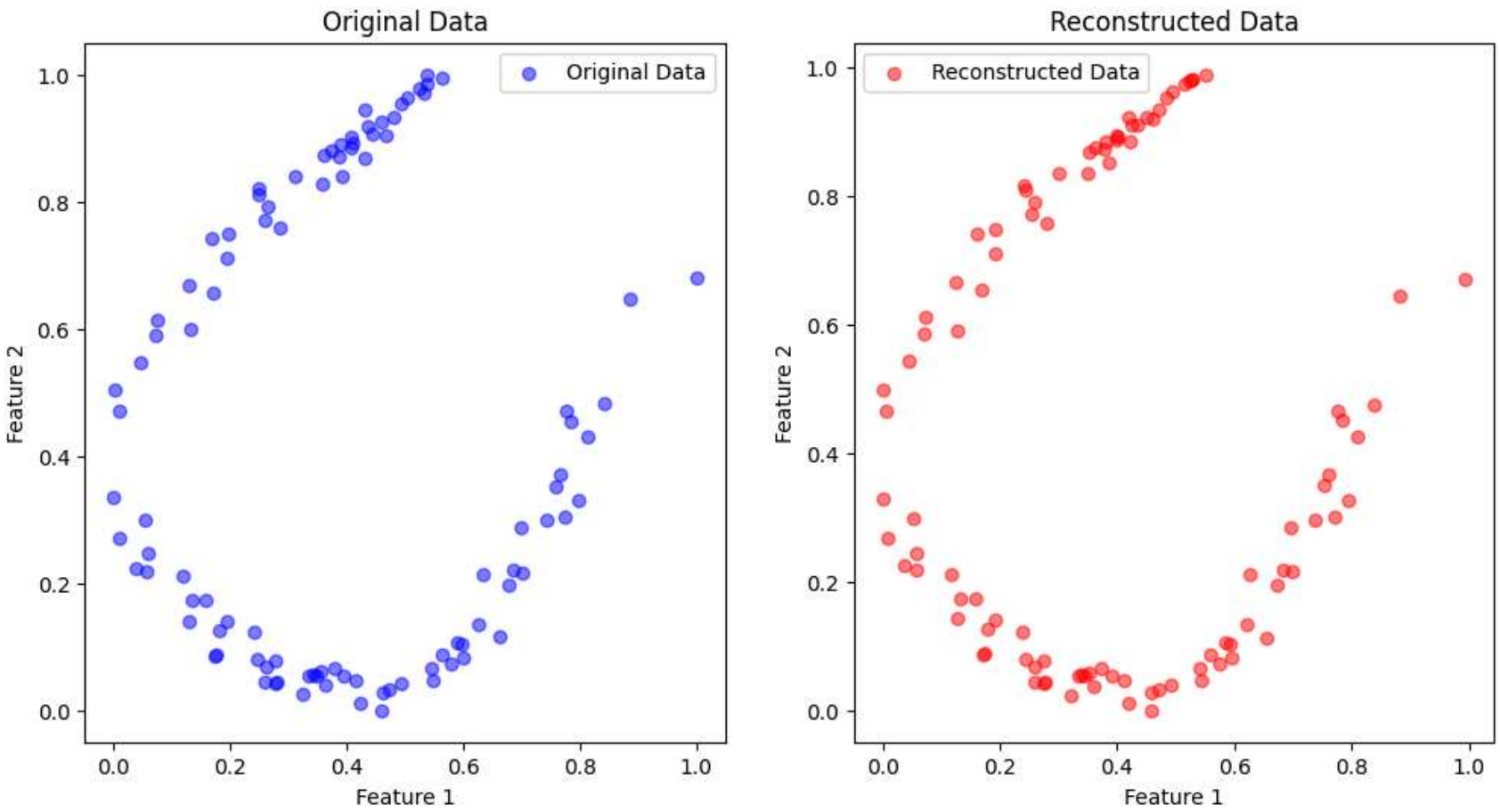
```
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:
```
import matplotlib.pyplot as plt
print(a[l].shape)

# 'a[l]' contains the reconstructed output from the last layer
# Since 'a[l]' has shape (2, number of samples), we transpose it to align with the original data
reconstructed_X = a[l].T

# Plot original data
```

```python
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.scatter(
    X_normalized[:, 0], X_normalized[:, 1], c="blue", alpha=0.5, label="Original Data"
)
plt.title("Original Data")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()

# Plot reconstructed data
plt.subplot(1, 2, 2)
plt.scatter(
    reconstructed_X[:, 0],
    reconstructed_X[:, 1],
    c="red",
    alpha=0.5,
    label="Reconstructed Data",
)
plt.title("Reconstructed Data")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()

plt.show()
```

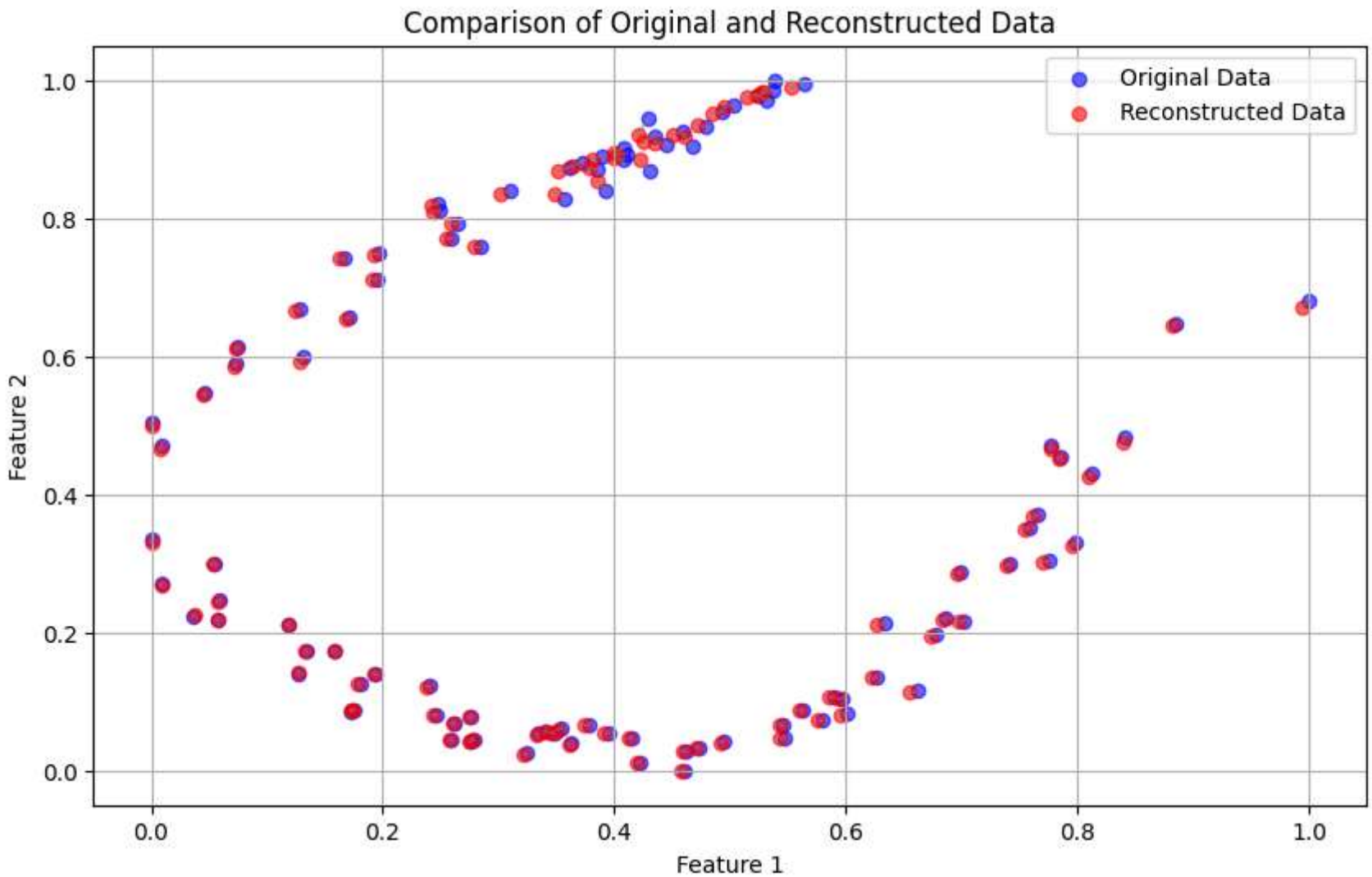
```
(2, 100)
```



In [ ]:

```
import matplotlib.pyplot as plt

# 计算重建数据
# 假设 a[L] 是网络的最后一层输出，形状为 (2, 样本数)，转置以便和原始数据对齐
reconstructed_X = a[l].T

# 创建一个图形和坐标轴
plt.figure(figsize=(10, 6))
plt.scatter(
    X_normalized[:, 0], X_normalized[:, 1], c="blue", alpha=0.6, label="Original Data"
)
plt.scatter(
    reconstructed_X[:, 0],
```

```python
    reconstructed_X[:, 1],
    c="red",
    alpha=0.6,
    label="Reconstructed Data",
)

# 为每个点绘制连接线
for i in range(len(X_normalized)):
    plt.plot(
        [X_normalized[i, 0], reconstructed_X[i, 0]],
        [X_normalized[i, 1], reconstructed_X[i, 1]],
        "gray",
        alpha=0.5,
    )

# 添加图例和标签
plt.title("Comparison of Original and Reconstructed Data")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()
plt.grid(True)
plt.show()
```



Question 13.8

In [ ]:
```
# get MNIST data from online repository
from sklearn.datasets import fetch_openml

x, y = fetch_openml("mnist_784", version=1, return_X_y=True)
```

```
x = x.T

# convert string labels to integers
y = np.array([int(v) for v in y])[:, np.newaxis]
y = y.T

print("input shape = ", x.shape)
print("output shape = ", y.shape)
```

```
input shape =  (784, 70000)
output shape =  (1, 70000)
```

In [ ]:

```
# 设置抽样数量
num_samples = 50000

# 生成随机的样本索引
indices = np.random.choice(x.shape[1], num_samples, replace=False)

# 使用这些索引来选择样本
# Sample the data
x_sample = x[indices]
y_sample = y[:, indices]

# 输出抽样后的数据形状
print("Sampled input shape = ", x_sample.shape)
print("Sampled output shape = ", y_sample.shape)
```

```
Sampled input shape =  (784, 50000)
Sampled output shape =  (1, 50000)
```

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler


# Activation functions and their derivatives
def relu(x):
    return np.maximum(0, x)


def relu_derivative(x):
```

```python
    return (x > 0).astype(x.dtype)


def softmax(x):
    exp_x = np.exp(x - np.max(x, axis=1, keepdims=True))
    return exp_x / np.sum(exp_x, axis=1, keepdims=True)


# Loss function
def compute_loss(y, y_pred):
    m = y.shape[0]
    log_likelihood = -np.log(y_pred[range(m), y])
    loss = np.mean(log_likelihood)
    return loss


# Batch normalization forward and backward
def batch_norm_forward(x, gamma, beta, eps=1e-8):
    mean = np.mean(x, axis=0)
    variance = np.var(x, axis=0)
    x_norm = (x - mean) / np.sqrt(variance + eps)
    out = gamma * x_norm + beta
    cache = (x, x_norm, mean, variance, gamma, beta)
    return out, cache


def batch_norm_backward(dout, cache):
    x, x_norm, mean, variance, gamma, beta = cache
    N = x.shape[0]
    x_mu = x - mean
    std_inv = 1.0 / np.sqrt(variance + 1e-8)

    dx_norm = dout * gamma
    dvar = np.sum(dx_norm * x_mu, axis=0) * -0.5 * std_inv**3
    dmu = np.sum(dx_norm * -std_inv, axis=0) + dvar * np.mean(-2.0 * x_mu, axis=0)

    dx = (dx_norm * std_inv) + (dvar * 2 * x_mu / N) + (dmu / N)
    dgamma = np.sum(dout * x_norm, axis=0)
    dbeta = np.sum(dout, axis=0)

    return dx, dgamma, dbeta
```

```python
# Initialize parameters
def initialize_parameters(layer_sizes):
    parameters = []
    for i in range(len(layer_sizes) - 1):
        W = np.random.randn(layer_sizes[i], layer_sizes[i + 1]) * np.sqrt(
            2.0 / layer_sizes[i]
        )  # He initialization
        b = np.zeros((1, layer_sizes[i + 1]))
        gamma = np.ones((1, layer_sizes[i + 1]))
        beta = np.zeros((1, layer_sizes[i + 1]))
        parameters.append({"W": W, "b": b, "gamma": gamma, "beta": beta})
    return parameters


# Forward and backward passes
def forward_backward_propagation(X, y, parameters):
    caches = []
    activations = [X]
    A = X

    # Forward pass with batch normalization and ReLU
    for param in parameters[:-1]:  # Last layer without batch normalization
        Z = np.dot(A, param["W"]) + param["b"]
        BN, cache_bn = batch_norm_forward(Z, param["gamma"], param["beta"])
        A = relu(BN)
        activations.append(A)
        caches.append((param, cache_bn))

    # Output layer
    final_Z = np.dot(A, parameters[-1]["W"]) + parameters[-1]["b"]
    final_A = softmax(final_Z)
    activations.append(final_A)
    caches.append((parameters[-1], None))

    # Compute loss
    loss = compute_loss(y, final_A)

    # Backward pass
    grads = []
    dA = final_A
    dA[range(y.shape[0]), y] -= 1  # Derivative of cross-entropy loss with softmax
```

```python
    dA /= y.shape[0]

    for i in reversed(range(len(caches))):
        param, cache_bn = caches[i]
        if cache_bn:
            dZ = relu_derivative(cache_bn[0]) * dA  # Apply relu derivative
            dA, dgamma, dbeta = batch_norm_backward(dZ, cache_bn)
            grads.append(
                {
                    "dW": np.dot(activations[i].T, dA),
                    "db": np.sum(dA, axis=0),
                    "dgamma": dgamma,
                    "dbeta": dbeta,
                }
            )
        else:
            dW = np.dot(activations[i].T, dA)
            db = np.sum(dA, axis=0)
            grads.append({"dW": dW, "db": db})
        dA = np.dot(dA, param["W"].T)

    grads.reverse()
    return loss, grads


# Update parameters
def update_parameters(parameters, grads, learning_rate):
    for param, grad in zip(parameters, grads):
        param["W"] -= learning_rate * grad["dW"]
        param["b"] -= learning_rate * grad["db"]
        if "dgamma" in grad and "dbeta" in grad:
            param["gamma"] -= learning_rate * grad["dgamma"]
            param["beta"] -= learning_rate * grad["dbeta"]


# Main training function
def train_with_norm(X, y, layer_sizes, epochs, learning_rate, init_para):
    parameters = init_para
    losses = []  # 存储损失历史
    for epoch in range(epochs):
        loss, grads = forward_backward_propagation(X, y, parameters)
        update_parameters(parameters, grads, learning_rate)
```

```python
        losses.append(loss)
        if epoch % 10 == 0:
            print(f"Epoch {epoch}, Loss: {loss}")
    return parameters, losses


# Load data
X, y = fetch_openml("mnist_784", version=1, return_X_y=True, as_frame=False)
y = y.astype(int)

# Preprocess data
scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)

# Split data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(
    X_normalized, y, test_size=0.2, random_state=42
)


# Define layer sizes and train the network
layer_sizes = [784, 128, 64, 10]  # Including input layer and output layer
init_parameter = initialize_parameters(layer_sizes)

import copy
init_parameter_wo_batch_norm = copy.deepcopy(init_parameter)

trained_parameters, lossess_with_batch_norm = train_with_norm(
    X_train,
    y_train,
    layer_sizes,
    epochs=10,
    learning_rate=0.01,
    init_para=init_parameter,
)
```

```
Epoch 0, Loss: 2.695709478303017
```

In [ ]:

```python
def no_batch_norm_forward(x, gamma, beta):
    out = gamma * x + beta
    cache = (x, gamma, beta)
    return out, cache
```

```python
def no_batch_norm_backward(dout, cache):
    x, gamma, beta = cache
    dx = dout * gamma
    dgamma = np.sum(dout * x, axis=0)
    dbeta = np.sum(dout, axis=0)
    return dx, dgamma, dbeta




# Forward and backward passes
def forward_backward_propagation_wo_norm(X, y, parameters):
    caches = []
    activations = [X]
    A = X

    # Forward pass with batch normalization and ReLU
    for param in parameters[:-1]:  # Last layer without batch normalization
        Z = np.dot(A, param["W"]) + param["b"]
        BN, cache_bn = no_batch_norm_forward(Z, param["gamma"], param["beta"])
        A = relu(BN)
        activations.append(A)
        caches.append((param, cache_bn))

    # Output layer
    final_Z = np.dot(A, parameters[-1]["W"]) + parameters[-1]["b"]
    final_A = softmax(final_Z)
    activations.append(final_A)
    caches.append((parameters[-1], None))

    # Compute loss
    loss = compute_loss(y, final_A)

    # Backward pass
    grads = []
    dA = final_A
    dA[range(y.shape[0]), y] -= 1  # Derivative of cross-entropy loss with softmax
    dA /= y.shape[0]

    for i in reversed(range(len(caches))):
```

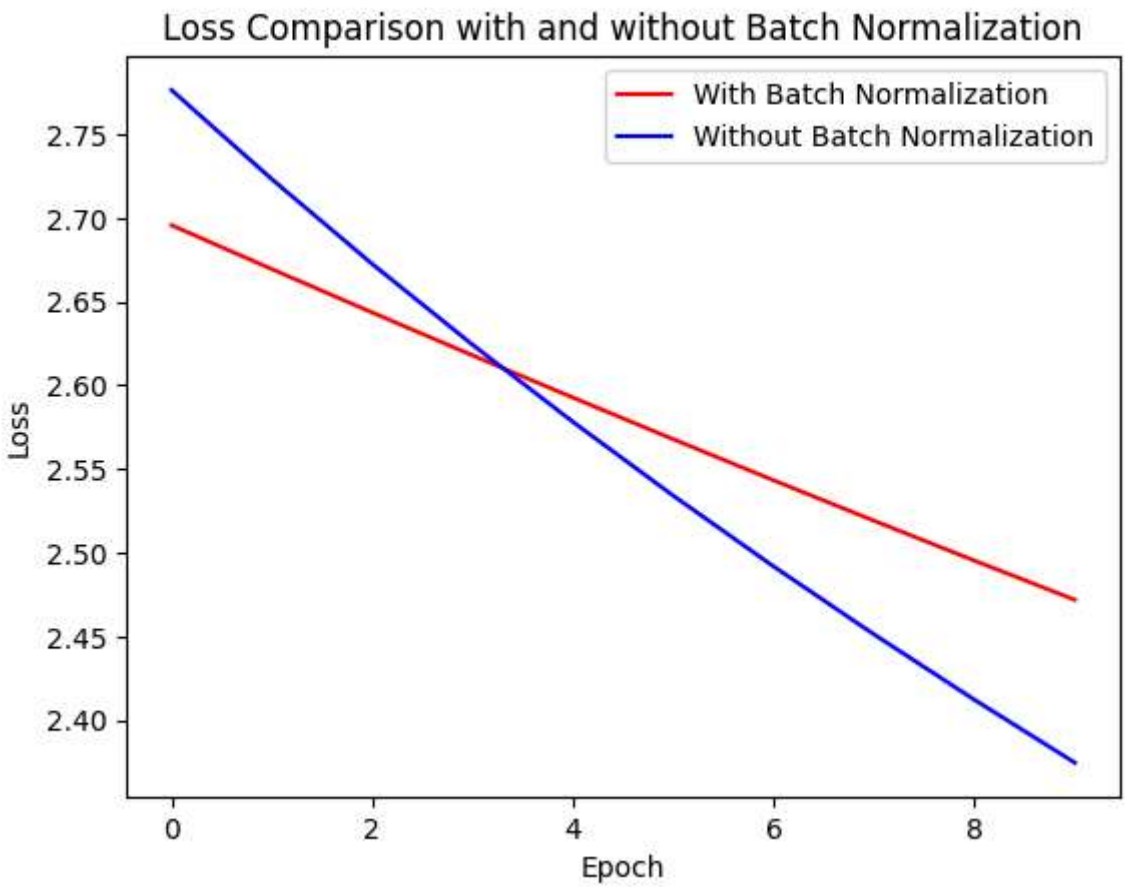
```python
        param, cache_bn = caches[i]
        if cache_bn:
            dZ = relu_derivative(cache_bn[0]) * dA  # Apply relu derivative
            dA, dgamma, dbeta = no_batch_norm_backward(dZ, cache_bn)
            grads.append(
                {
                    "dW": np.dot(activations[i].T, dA),
                    "db": np.sum(dA, axis=0),
                    "dgamma": dgamma,
                    "dbeta": dbeta,
                }
            )
        else:
            dW = np.dot(activations[i].T, dA)
            db = np.sum(dA, axis=0)
            grads.append({"dW": dW, "db": db})
        dA = np.dot(dA, param["W"].T)

    grads.reverse()
    return loss, grads


# Main training function
def train_without_norm(X, y, layer_sizes, epochs, learning_rate, init_para):
    parameters = init_para
    losses = []  # 存储损失历史
    for epoch in range(epochs):
        loss, grads = forward_backward_propagation_wo_norm(X, y, parameters)
        update_parameters(parameters, grads, learning_rate)
        losses.append(loss)
        if epoch % 10 == 0:
            print(f"Epoch {epoch}, Loss: {loss}")
    return parameters, losses


layer_sizes = [784, 128, 64, 10]  # Including input layer and output layer
trained_parameters, lossess_without_batch_norm = train_without_norm(
    X_train,
    y_train,
    layer_sizes,
    epochs=10,
    learning_rate=0.01,
```

```
    init_para=init_parameter_wo_batch_norm,
)
```

```
Epoch 0, Loss: 2.776456197184883
```

In [ ]:

```
plt.plot(
    range(10), lossess_with_batch_norm, label="Without Batch Normalization", color="red"
)
plt.plot(
    range(10),
    lossess_without_batch_norm,
    label="With Batch Normalization",
    color="blue",
)


plt.xlabel("Epoch")

plt.ylabel("Loss")

plt.title("Loss Comparison with and without Batch Normalization")

plt.legend()
plt.show()
```



Question 13.9

In [ ]:
```
import numpy as np

data_path = "./datasets/noisy_sin_sample.csv"

# Load the data using numpy
data = np.loadtxt(data_path, delimiter=",")

# Separate the features and target
X = data[:-1, :].T  # Transpose to make each row an observation
y = data[-1, :]
```

```python
# Split the data into training and validation sets (2/3 training, 1/3 validation)
num_samples = X.shape[0]
train_size = int(2 / 3 * num_samples)

# Training data
X_train = X[:train_size]
y_train = y[:train_size]

# Validation data
X_val = X[train_size:]
y_val = y[train_size:]

X_train.shape, y_train.shape, X_val.shape, y_val.shape
```

Out[ ]: ((14, 1), (14,), (7, 1), (7,))

In [ ]:

```python
import numpy as np


def initialize_weights(layers):
    weights = {}
    for i in range(1, len(layers)):
        weights[f"W{i}"] = np.random.randn(layers[i], layers[i - 1]) * np.sqrt(
            1.0 / layers[i - 1]
        )
        weights[f"b{i}"] = np.zeros((layers[i], 1))
    return weights


def tanh(x):
    return np.tanh(x)


def tanh_derivative(x):
    return 1 - np.tanh(x) ** 2


def forward_pass(X, weights, layers):
    activations = {"A0": X}
    for i in range(1, len(layers)):
        z = np.dot(weights[f"W{i}"], activations[f"A{i-1}"]) + weights[f"b{i}"]
        a = tanh(z)
```

```python
        activations[f"Z{i}"] = z
        activations[f"A{i}"] = a
    return activations


def compute_loss(y, y_hat):
    return np.mean((y - y_hat) ** 2)


def backward_pass(activations, weights, layers, X, y):
    m = X.shape[1]
    y = y.reshape(1, -1)  # Reshape y to match output dimensions
    gradients = {}
    # Output layer error
    delta = (activations[f"A{len(layers)-1}"] - y) * tanh_derivative(
        activations[f"Z{len(layers)-1}"]
    )
    gradients[f"W{len(layers)-1}"] = (
        1 / m * np.dot(delta, activations[f"A{len(layers)-2}"].T)
    )
    gradients[f"b{len(layers)-1}"] = 1 / m * np.sum(delta, axis=1, keepdims=True)

    # Hidden layers error
    for i in range(len(layers) - 2, 0, -1):
        delta = np.dot(weights[f"W{i+1}"].T, delta) * tanh_derivative(
            activations[f"Z{i}"]
        )
        gradients[f"W{i}"] = 1 / m * np.dot(delta, activations[f"A{i-1}"].T)
        gradients[f"b{i}"] = 1 / m * np.sum(delta, axis=1, keepdims=True)

    return gradients


def update_weights(weights, gradients, learning_rate):
    for key in weights.keys():
        weights[key] -= learning_rate * gradients[key]
    return weights


# Neural network architecture
layers = [
    1,
```

```
    10,
    10,
    10,
    1,
]  # Three hidden layers with 10 units each, 1 input and 1 output

# Initialize weights
weights = initialize_weights(layers)

# Learning rate and epochs
learning_rate = 0.01
epochs = 10000
patience = 10  # Patience for early stopping

# Early stopping variables
best_loss = np.inf
patience_counter = 0

train_losses, val_losses = [], []
bast_epoch = 0

# Training loop
for epoch in range(epochs):
    # Forward pass on training data
    activations_train = forward_pass(X_train.T, weights, layers)
    train_loss = compute_loss(y_train, activations_train[f"A{len(layers)-1}"])
    train_losses.append(train_loss)

    # Forward pass on validation data
    activations_val = forward_pass(X_val.T, weights, layers)
    val_loss = compute_loss(y_val, activations_val[f"A{len(layers)-1}"])
    val_losses.append(val_loss)

    # Backpropagation and weight update
    gradients = backward_pass(activations_train, weights, layers, X_train.T, y_train)
    weights = update_weights(weights, gradients, learning_rate)

    # Check for early stopping
    if val_loss < best_loss:
        best_loss = val_loss
        best_weights = weights
        patience_counter = 0
```

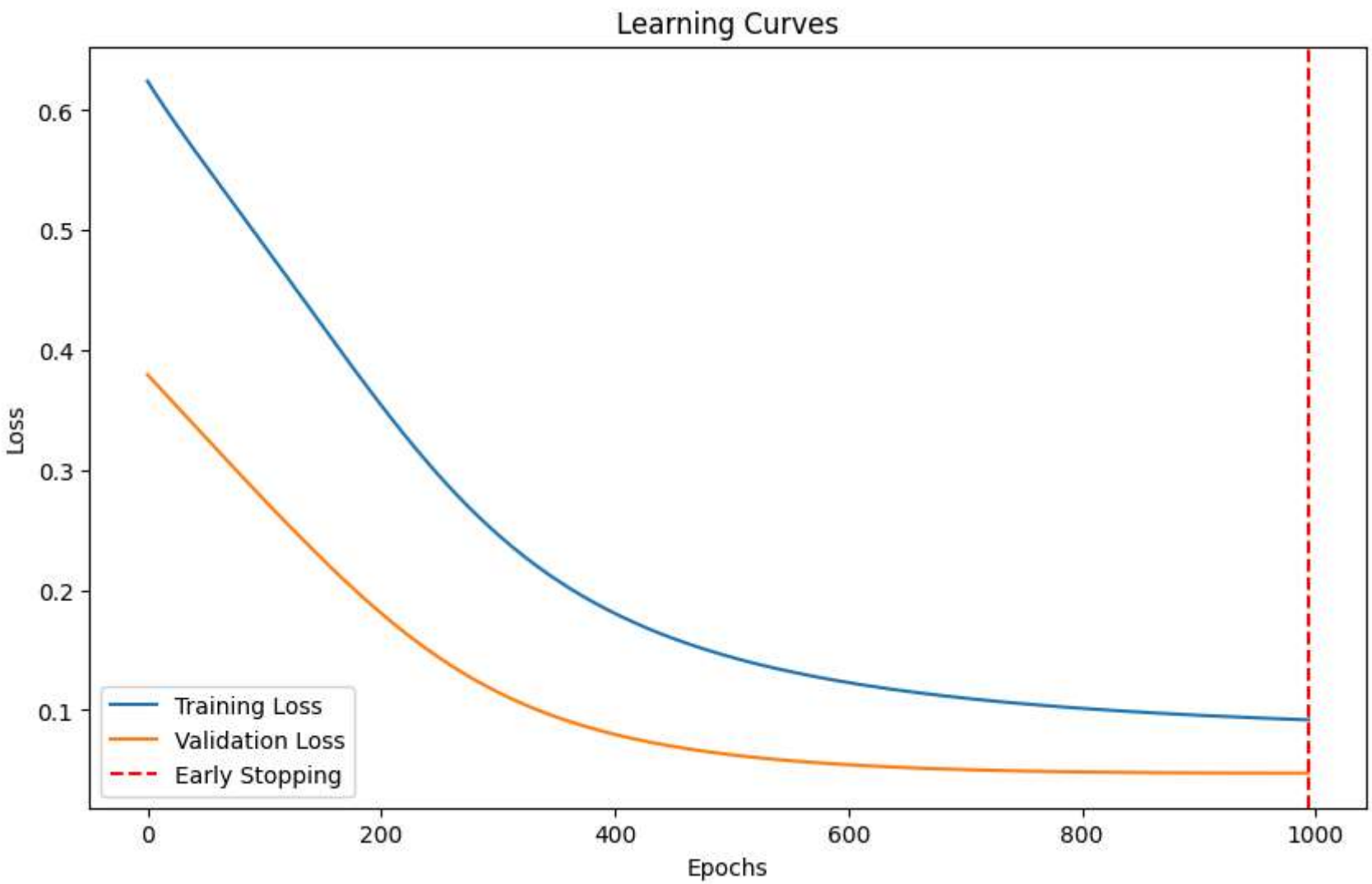
```
    else:
        patience_counter += 1
        if patience_counter >= patience:
            print(f"Early stopping at epoch {epoch} with validation loss: {best_loss}")
            best_epoch = epoch
            break

    # Print training information every 50 epochs
    if epoch % 50 == 0:
        print(
            f"Epoch {epoch}, Training Loss: {train_loss:.4f}, Validation Loss: {val_loss:.4f}"
        )

# The final best weights are stored in `best_weights`
```

```
Epoch 0, Training Loss: 0.6243, Validation Loss: 0.3796
Epoch 50, Training Loss: 0.5536, Validation Loss: 0.3270
Epoch 100, Training Loss: 0.4871, Validation Loss: 0.2751
Epoch 150, Training Loss: 0.4198, Validation Loss: 0.2255
Epoch 200, Training Loss: 0.3541, Validation Loss: 0.1806
Epoch 250, Training Loss: 0.2949, Validation Loss: 0.1433
Epoch 300, Training Loss: 0.2462, Validation Loss: 0.1148
Epoch 350, Training Loss: 0.2086, Validation Loss: 0.0941
Epoch 400, Training Loss: 0.1805, Validation Loss: 0.0796
Epoch 450, Training Loss: 0.1596, Validation Loss: 0.0696
Epoch 500, Training Loss: 0.1440, Validation Loss: 0.0626
Epoch 550, Training Loss: 0.1320, Validation Loss: 0.0577
Epoch 600, Training Loss: 0.1228, Validation Loss: 0.0543
Epoch 650, Training Loss: 0.1156, Validation Loss: 0.0519
Epoch 700, Training Loss: 0.1098, Validation Loss: 0.0502
Epoch 750, Training Loss: 0.1051, Validation Loss: 0.0491
Epoch 800, Training Loss: 0.1013, Validation Loss: 0.0483
Epoch 850, Training Loss: 0.0982, Validation Loss: 0.0478
Epoch 900, Training Loss: 0.0956, Validation Loss: 0.0475
Epoch 950, Training Loss: 0.0934, Validation Loss: 0.0474
Early stopping at epoch 994 with validation loss: 0.047342225003456286
```

In [ ]:

```
import matplotlib.pyplot as plt

# Plotting the learning curves
plt.figure(figsize=(10, 6))
plt.plot(train_losses, label="Training Loss")
```

```python
plt.plot(val_losses, label="Validation Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Learning Curves")
plt.axvline(x=best_epoch, color="r", linestyle="--", label="Early Stopping")
plt.legend()
plt.show()
```



Question 13.10

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import accuracy_score
import time


# Load data
x, y = fetch_openml("mnist_784", version=1, return_X_y=True)
x = x.astype(np.float32)
```

In [ ]:
```
# Standardize data
scaler = StandardScaler()
x = scaler.fit_transform(x)

# Convert labels to integers and then one-hot encode
y = np.array([int(v) for v in y])[:, np.newaxis]
encoder = OneHotEncoder()
y_onehot = encoder.fit_transform(y).toarray()

# Split the data
x_train, x_test, y_train, y_test = train_test_split(
    x, y_onehot, test_size=0.2, random_state=42
)

# Neural network parameters
input_size = 784  # Number of input neurons (28x28 pixels)
hidden_size = 50  # Number of hidden neurons
output_size = 10  # Number of output neurons (digits 0-9)
learning_rate = 0.1
epochs = 500

# Weight initialization
np.random.seed(42)
```

```python
W1 = np.random.randn(input_size, hidden_size) * 0.01
b1 = np.zeros((1, hidden_size))
W2 = np.random.randn(hidden_size, output_size) * 0.01
b2 = np.zeros((1, output_size))


# Activation functions
def sigmoid(x):
    return 1 / (1 + np.exp(-x))


def softmax(x):
    exp_x = np.exp(x - np.max(x, axis=1, keepdims=True))
    return exp_x / np.sum(exp_x, axis=1, keepdims=True)


# Storing loss and accuracy history
loss_history = []
train_accuracy_history = []
valid_accuracy_history = []

# Training the network
for epoch in range(epochs):
    start_time = time.time()

    # Forward pass
    z1 = np.dot(x_train, W1) + b1
    a1 = sigmoid(z1)
    z2 = np.dot(a1, W2) + b2
    a2 = softmax(z2)

    # Compute the loss
    loss = -np.mean(np.log(a2[np.arange(len(y_train)), y_train.argmax(axis=1)]))
    loss_history.append(loss)

    # Backpropagation
    dz2 = a2 - y_train
    dW2 = np.dot(a1.T, dz2)
    db2 = np.sum(dz2, axis=0, keepdims=True)
    dz1 = np.dot(dz2, W2.T) * a1 * (1 - a1)
    dW1 = np.dot(x_train.T, dz1)
    db1 = np.sum(dz1, axis=0, keepdims=True)
```

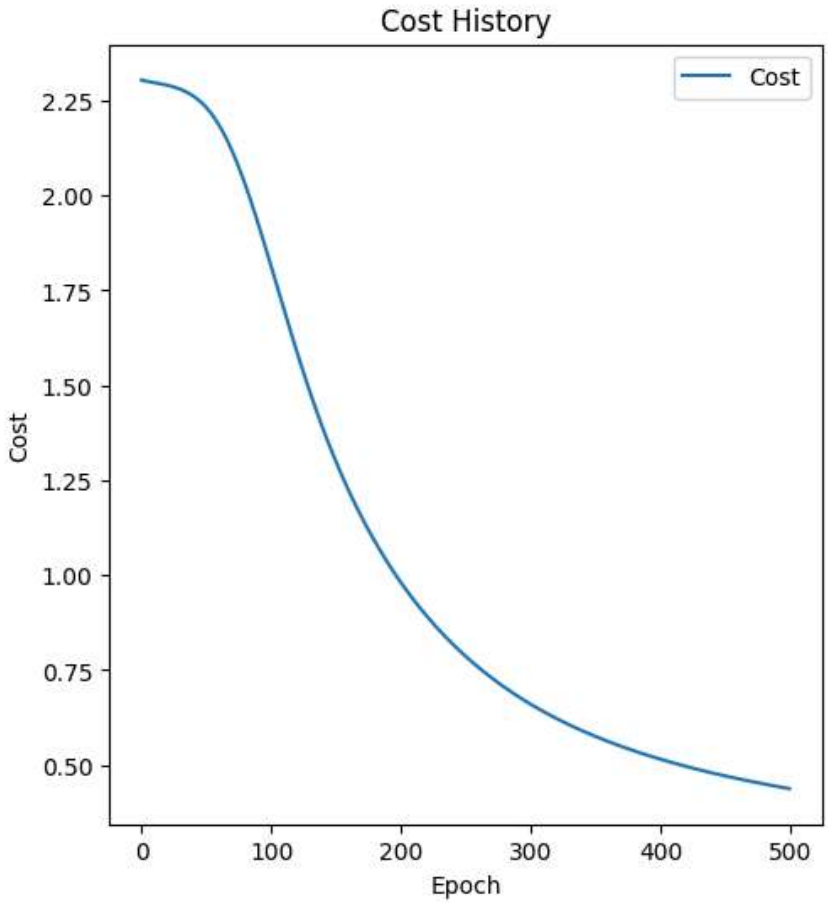
```
    # Update weights
    W2 -= learning_rate * dW2 / len(x_train)
    b2 -= learning_rate * db2 / len(x_train)
    W1 -= learning_rate * dW1 / len(x_train)
    b1 -= learning_rate * db1 / len(x_train)

    # Training accuracy
    train_predictions = np.argmax(a2, axis=1)
    train_accuracy = accuracy_score(y_train.argmax(axis=1), train_predictions)
    train_accuracy_history.append(train_accuracy)

    # Validation accuracy
    z1_valid = np.dot(x_test, W1) + b1
    a1_valid = sigmoid(z1_valid)
    z2_valid = np.dot(a1_valid, W2) + b2
    a2_valid = softmax(z2_valid)
    valid_predictions = np.argmax(a2_valid, axis=1)
    valid_accuracy = accuracy_score(y_test.argmax(axis=1), valid_predictions)
    valid_accuracy_history.append(valid_accuracy)

    # Timing
    end_time = time.time()
    elapsed_time = end_time - start_time

    # Print training progress
    if epoch % 10 == 0 or epoch == epochs - 1:
        print(
            f"step {epoch} done in {elapsed_time:.1f} secs, train acc = {train_accuracy:.4f}, valid acc = {valid_accu
        )
```

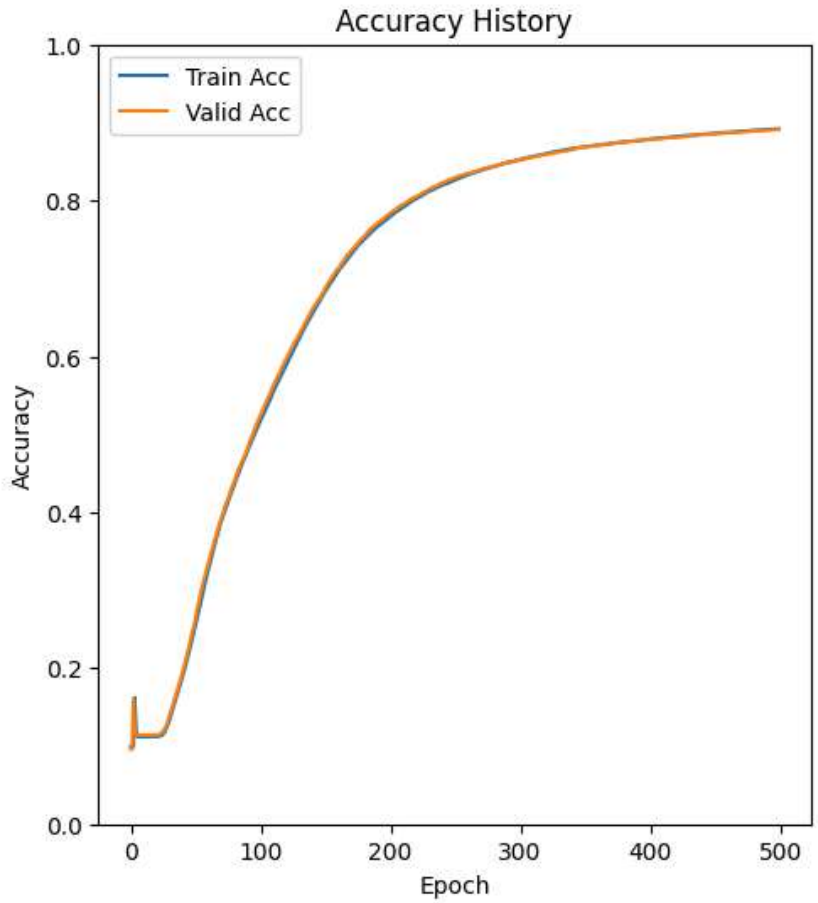
```
step 0 done in 0.4 secs, train acc = 0.0993, valid acc = 0.0959
step 10 done in 0.4 secs, train acc = 0.1121, valid acc = 0.1143
step 20 done in 0.4 secs, train acc = 0.1122, valid acc = 0.1145
step 30 done in 0.4 secs, train acc = 0.1348, valid acc = 0.1395
step 40 done in 0.4 secs, train acc = 0.1904, valid acc = 0.1960
step 50 done in 0.4 secs, train acc = 0.2564, valid acc = 0.2664
step 60 done in 0.4 secs, train acc = 0.3287, valid acc = 0.3360
step 70 done in 0.4 secs, train acc = 0.3902, valid acc = 0.3937
step 80 done in 0.4 secs, train acc = 0.4371, valid acc = 0.4411
step 90 done in 0.4 secs, train acc = 0.4792, valid acc = 0.4821
step 100 done in 0.4 secs, train acc = 0.5178, valid acc = 0.5249
step 110 done in 0.4 secs, train acc = 0.5556, valid acc = 0.5636
step 120 done in 0.4 secs, train acc = 0.5898, valid acc = 0.5977
step 130 done in 0.4 secs, train acc = 0.6240, valid acc = 0.6297
step 140 done in 0.4 secs, train acc = 0.6553, valid acc = 0.6611
step 150 done in 0.4 secs, train acc = 0.6844, valid acc = 0.6884
step 160 done in 0.4 secs, train acc = 0.7098, valid acc = 0.7129
step 170 done in 0.5 secs, train acc = 0.7319, valid acc = 0.7360
step 180 done in 0.4 secs, train acc = 0.7505, valid acc = 0.7541
step 190 done in 0.4 secs, train acc = 0.7664, valid acc = 0.7709
step 200 done in 0.4 secs, train acc = 0.7794, valid acc = 0.7836
step 210 done in 0.4 secs, train acc = 0.7916, valid acc = 0.7954
step 220 done in 0.4 secs, train acc = 0.8025, valid acc = 0.8052
step 230 done in 0.4 secs, train acc = 0.8118, valid acc = 0.8147
step 240 done in 0.4 secs, train acc = 0.8197, valid acc = 0.8224
step 250 done in 0.4 secs, train acc = 0.8263, valid acc = 0.8295
step 260 done in 0.4 secs, train acc = 0.8331, valid acc = 0.8349
step 270 done in 0.4 secs, train acc = 0.8389, valid acc = 0.8397
step 280 done in 0.4 secs, train acc = 0.8439, valid acc = 0.8439
step 290 done in 0.4 secs, train acc = 0.8485, valid acc = 0.8481
step 300 done in 0.4 secs, train acc = 0.8525, valid acc = 0.8519
step 310 done in 0.4 secs, train acc = 0.8566, valid acc = 0.8560
step 320 done in 0.4 secs, train acc = 0.8601, valid acc = 0.8591
step 330 done in 0.4 secs, train acc = 0.8635, valid acc = 0.8624
step 340 done in 0.4 secs, train acc = 0.8666, valid acc = 0.8655
step 350 done in 0.4 secs, train acc = 0.8689, valid acc = 0.8687
step 360 done in 0.4 secs, train acc = 0.8710, valid acc = 0.8706
step 370 done in 0.4 secs, train acc = 0.8730, valid acc = 0.8729
step 380 done in 0.4 secs, train acc = 0.8752, valid acc = 0.8749
step 390 done in 0.4 secs, train acc = 0.8766, valid acc = 0.8769
step 400 done in 0.4 secs, train acc = 0.8785, valid acc = 0.8784
step 410 done in 0.4 secs, train acc = 0.8805, valid acc = 0.8796
```

```
step 420 done in 0.4 secs, train acc = 0.8819, valid acc = 0.8813
step 430 done in 0.4 secs, train acc = 0.8834, valid acc = 0.8828
step 440 done in 0.5 secs, train acc = 0.8848, valid acc = 0.8844
step 450 done in 0.5 secs, train acc = 0.8863, valid acc = 0.8855
step 460 done in 0.4 secs, train acc = 0.8874, valid acc = 0.8868
step 470 done in 0.5 secs, train acc = 0.8888, valid acc = 0.8877
step 480 done in 0.5 secs, train acc = 0.8897, valid acc = 0.8889
step 490 done in 0.5 secs, train acc = 0.8911, valid acc = 0.8900
step 499 done in 0.4 secs, train acc = 0.8918, valid acc = 0.8910
```

In [ ]:

```python
# Plotting cost and accuracy history
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.plot(loss_history, label="Cost")
plt.title("Cost History")
plt.xlabel("Epoch")
plt.ylabel("Cost")
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(train_accuracy_history, label="Train Acc")
plt.plot(valid_accuracy_history, label="Valid Acc")
plt.title("Accuracy History")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.ylim(0, 1.0)  # Set the y-axis to be from 0 to 1.0
plt.legend()

plt.show()
```


Cost History
Cost
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0
100
200
300
400
500
Epoch


Accuracy History
Train Acc
Valid Acc
1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
0
100
200
300
400
500
Epoch