For the same data in Prob 1, find a bi-linear system which maps the two view inputs $x_k^m = F_k(:,m)$ and $x_k^n = F_k(:,n)$ where $n \neq m$ to its labels such that

$$\min_W \max_{v_k, u_k} \quad \frac{1}{K}\sum_{k=1}^{K}(y_k - (x_k^m + v_k)^T W (x_k^n + u_k))^2$$
$$s.t. \quad ||u_k||_2^2 \leq \delta^2, \quad ||v_k||_2^2 \leq \delta^2, \quad k = 1, \ldots, K.$$

where W is a weight matrix. Derive update rules to estimate W and report classification accuracy for 10-fold cross-validation

let's denote x_k^m as x_k and x_k^n as z_k.

solve the problem with respect to v_k.

$$\max_{v_k}(y_k - (x_k + v_k)^T W (z_k + u_k))^2 = \max_{v_k}(y_k - (x_k + v_k)^T w_R)^2$$

There is the same problem as in the first task, so

$$v_k^* = \lambda \frac{w_R}{||w_R||_2}\delta = -sign(y_k - x_k^T w_R)\delta\frac{w_R}{||w_R||_2}$$

The same for u_k

$$\max_{u_k}(y_k - (x_k + v_k)^T W (z_k + u_k))^2 = \max_{u_k}(y_k - w_L^T (z_k + u_k))^2 =$$
$$\max_{u_k}(y_k - (z_k + u_k)^T w_L)^2$$

$$u_k^* = -sign(y_k - z_k^T w_L)\delta\frac{w_L}{||w_L||_2}$$

For W

$$\min_W \max_{v_k, u_k} \quad \frac{1}{K}\sum_{k=1}^{K}(y_k - (x_k + v_k)^T W (z_k + u_k))^2 =$$
$$\min_W \quad \frac{1}{K}\sum_{k=1}^{K}(y_k - (x_k + v_k^*)^T W (z_k + u_k^*))^2$$

denote X as $[x_1 + v_1^*, \ldots, x_K + v_K^*]$ and Z as $[z_1 + u_1^*, \ldots, z_K + u_K^*]$ $\sum_{k=1}^{K}(y_k - (x_k + v_k^*)^T W (z_k + u_k^*))^2 = ||y - diag(X^T W Z)||_2^2$

I did not find out how to differentiate the diag function or more simpler matrix form. So:

$$W = \text{argmin}_W ||y - diag(X^T W Z)||_2^2$$

result:

1. $$w_L = W^{tT}(x_k + v_k^t)$$
$$u_k^{t+1} = -sign(y_k - z_k^T w_L)\delta\frac{w_L}{||w_L||_2}$$
2. $$w_R = W^t(z_k + u_k)$$
$$v_k^{t+1} = -sign(y_k - x_k^T w_R)\delta\frac{w_R}{||w_R||_2}$$
3. $$X = [x_1 + v_1^{t+1}, \ldots, x_K + v_K^{t+1}] \quad Z = [z_1 + u_1^{t+1}, \ldots, z_K + u_K^{t+1}]$$
$$W^{t+1} = \text{argmin}_W ||y - diag(X^T W Z)||_2^2$$

In [ ]:
```
import cv2
import os
import glob
import numpy as np
from numpy.linalg import norm
import matplotlib.pyplot as plt
from keras.datasets import mnist
from numpy.linalg import inv
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score, precision_score
from scipy.optimize import minimize
import cvxpy as cp
```

```
2023-12-17 13:25:30.329116: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered
2023-12-17 13:25:30.329185: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered
2023-12-17 13:25:30.385157: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
2023-12-17 13:25:30.510448: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-12-17 13:25:31.762328: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
```

In [ ]:
```
def build_Gaborfilters():
    filters = []
    ksize = 28
    for theta in np.linspace(0, np.pi, 32):
        kern = cv2.getGaborKernel(
            (ksize, ksize), 4.0, theta, 10.0, 0.5, 0, ktype=cv2.CV_32F
        )
        kern /= 1.5 * kern.sum()
        filters.append(kern)
    return filters


def process(img, filters):
    accum = []
    for kern in filters:
        fimg = cv2.filter2D(img, cv2.CV_8UC3, kern)
        accum.append(fimg.reshape(-1))
    return np.vstack(accum).T


(images, labels), (test_X, test_y) = mnist.load_data()
is_1_2 = (labels == 0) | (labels == 2)
labels = labels[is_1_2][:1000]
images = images[is_1_2][:1000]


F_list = []
for img in images:
    F = process(img, build_Gaborfilters())
    F_list.append(F)
```

In [ ]:
```
def get_U(Y, X, Z, V, W, delta):
    K = Y.shape[0]
    U_next = []
    for k in range(K):
        x_k = X[:, k]
        z_k = Z[:, k]
        v_k = V[:, k]
        y_k = Y[k]

        w_1 = W @ (x_k + v_k)
        u_next_k = -np.sign(y_k - z_k.T @ w_1) * delta * w_1 / norm(w_1, 2)
        U_next.append(u_next_k)
    return np.vstack(U_next).T


def get_V(Y, X, Z, U, W, delta):
    K = Y.shape[0]
    V_next = []
    for k in range(K):
        x_k = X[:, k]
        z_k = Z[:, k]
        u_k = U[:, k]
        y_k = Y[k]

        w_r = W.T @ (z_k + u_k)
        v_next_k = -np.sign(y_k - x_k.T @ w_r) * delta * w_r / norm(w_r, 2)
        V_next.append(v_next_k)
    return np.vstack(V_next).T


def get_cost_func(Y, X, Z, W, V, U):
    K = Y.shape[0]
    cost = [
        (Y[k] - (X[:, k] + V[:, k]).T @ W @ (Z[:, k] + U[:, k])) ** 2 for k in range(K)
    ]
    return sum(cost)


# def get_cost_func(Y, X, Z, W, V, U):
#     # Y_predict = predict(X+X + V, Z+Z + U, W+W)
#     Y_predict = cp.diag((X + V).T @ W @ (Z + U))
#     cost = cp.norm(Y - Y_predict, 2)
#     return cost


def get_W(Y, X, Z, V, U, rank=1):
    w1 = cp.Variable((784, rank))
    w2 = np.random.random((784, rank)).astype(np.float32)
    W = w1 @ w2.T
    cost = get_cost_func(Y, X, Z, W, V, U)
    prob = cp.Problem(cp.Minimize(cost))
    prob.solve("SCS")

    w1 = w1.value
    w2 = cp.Variable((784, rank))
    W = w1 @ w2.T
    cost = get_cost_func(Y, X, Z, W, V, U)
    prob = cp.Problem(cp.Minimize(cost))
    prob.solve("SCS")
    w2 = w2.value
    W = w1 @ w2.T
    return W


def optimize(Y, X, Z, delta, num_iter, rank):
    W = np.random.random(size=(784, 784)).astype(np.float32)
    V = np.random.random(size=(784, Y.shape[0])).astype(np.float32)
    for _ in range(num_iter):
        U = get_U(Y, X, Z, V, W, delta)
        V = get_V(Y, X, Z, U, W, delta)
        W = get_W(Y, X, Z, V, U, rank=rank)
    return W


def predict(X, Z, W):
    return np.diag(X.T @ W @ Z)
```

It was too slow to update W So I substituted $W = w_1 w_2^T$

Here You may see I use rank 1 approximation. (I tried to use a lot of things like diff. methods, GPU, max_num_iter option, reformulate the problem but it didn't work or was too slow. It is still too slow so I used 1st rank approximation and only 1 iteration)

In [ ]:
```
kf = KFold(n_splits=10)
m = 3
n = 15
for i, (train, test) in enumerate(kf.split(labels)):
    X_train = [F_list[img_indx][:, m] for img_indx in train]
    X_train = np.vstack(X_train).astype(np.float32).T / 255

    Z_train = [F_list[img_indx][:, n] for img_indx in train]
    Z_train = np.vstack(Z_train).astype(np.float32).T / 255

    X_test = [F_list[img_indx][:, m] for img_indx in test]
    X_test = np.vstack(X_test).astype(float).T / 255

    Z_test = [F_list[img_indx][:, n] for img_indx in test]
    Z_test = np.vstack(Z_test).astype(np.float32).T / 255

    y_test = labels[test].astype(np.float32) / 2
    y_train = labels[train].astype(np.float32) / 2

    W_star = optimize(y_train, X_train, Z_train, delta=0.01, num_iter=1, rank=1)

    y_predict_test = np.round(predict(X_test, Z_test, W_star))
    score_test = accuracy_score(y_test, y_predict_test)

    y_predict_train = np.round(predict(X_train, Z_train, W_star))
    score_train = accuracy_score(y_train, y_predict_train)

    print(f"fold #{i} accuracy test {score_test} | accuracy train {score_train}")
```

```
/home/sun/.venv/lib/python3.10/site-packages/cvxpy/problems/problem.py:158: UserWarning: Objective contains too many subexpressions. Consider vectorizing your CVXPY code to speed up compilation.
  warnings.warn("Objective contains too many subexpressions. "
fold #0 accuracy test 0.61 | accuracy train 1.0
/home/sun/.venv/lib/python3.10/site-packages/cvxpy/problems/problem.py:158: UserWarning: Objective contains too many subexpressions. Consider vectorizing your CVXPY code to speed up compilation.
  warnings.warn("Objective contains too many subexpressions. "
fold #1 accuracy test 0.58 | accuracy train 1.0
/home/sun/.venv/lib/python3.10/site-packages/cvxpy/problems/problem.py:158: UserWarning: Objective contains too many subexpressions. Consider vectorizing your CVXPY code to speed up compilation.
  warnings.warn("Objective contains too many subexpressions. "
fold #2 accuracy test 0.53 | accuracy train 1.0
/home/sun/.venv/lib/python3.10/site-packages/cvxpy/problems/problem.py:158: UserWarning: Objective contains too many subexpressions. Consider vectorizing your CVXPY code to speed up compilation.
  warnings.warn("Objective contains too many subexpressions. "
fold #3 accuracy test 0.59 | accuracy train 1.0
/home/sun/.venv/lib/python3.10/site-packages/cvxpy/problems/problem.py:158: UserWarning: Objective contains too many subexpressions. Consider vectorizing your CVXPY code to speed up compilation.
  warnings.warn("Objective contains too many subexpressions. "
fold #4 accuracy test 0.64 | accuracy train 1.0
/home/sun/.venv/lib/python3.10/site-packages/cvxpy/problems/problem.py:158: UserWarning: Objective contains too many subexpressions. Consider vectorizing your CVXPY code to speed up compilation.
  warnings.warn("Objective contains too many subexpressions. "
fold #5 accuracy test 0.59 | accuracy train 1.0
/home/sun/.venv/lib/python3.10/site-packages/cvxpy/problems/problem.py:158: UserWarning: Objective contains too many subexpressions. Consider vectorizing your CVXPY code to speed up compilation.
  warnings.warn("Objective contains too many subexpressions. "
fold #6 accuracy test 0.58 | accuracy train 1.0
/home/sun/.venv/lib/python3.10/site-packages/cvxpy/problems/problem.py:158: UserWarning: Objective contains too many subexpressions. Consider vectorizing your CVXPY code to speed up compilation.
  warnings.warn("Objective contains too many subexpressions. "
fold #7 accuracy test 0.54 | accuracy train 1.0
/home/sun/.venv/lib/python3.10/site-packages/cvxpy/problems/problem.py:158: UserWarning: Objective contains too many subexpressions. Consider vectorizing your CVXPY code to speed up compilation.
  warnings.warn("Objective contains too many subexpressions. "
fold #8 accuracy test 0.63 | accuracy train 1.0
/home/sun/.venv/lib/python3.10/site-packages/cvxpy/problems/problem.py:158: UserWarning: Objective contains too many subexpressions. Consider vectorizing your CVXPY code to speed up compilation.
  warnings.warn("Objective contains too many subexpressions. "
fold #9 accuracy test 0.6 | accuracy train 1.0
```

In [ ]:
```
```