Please submit a single document for this assignment on Canvas by **Monday, Nov. 14th**. Post both **your code** and **a screenshot of its output.**

Suppose we wish to train a model to distinguish just the zeros and eights in this hand-drawn digit dataset. Those data consist of 1000 20x20 greyscale images — i.e., each example in the data set has 400 features, all told.

As a first pass, let us find a 399-dimensional hyperplane optimal in the since that it well separates the zeros from the eights in 400-dimensional space. To do this, let us use linear regression — that is, let us train a linear model with `MSELoss` as in previous projects.

This is Exercise 3 on this DL@DU project page. Please use the provided boilerplate code on that page.

Notes:

- Since we are using a linear model, there is no need for test data.
- For many of the 400 features of the data (such as any pixel position close to a edge), the standard deviation over all examples will be zero; you don't want to divide by that.

  So either forego normalizing or consider using `dulib.normalize` (which avoids division by zero); do something like this (to both center and normalize):

```
import du.lib as dulib

...

xss, xss_means = dulib.center(xss)
xss, xss_stds = dulib.normalize(xss)

...
```

- Feel free to use `dulib.train` if you so wish.
- (Optional) If you wish to show the percentage correct on your real-time graph, you can do something like this:

```
def pct_correct(yhatss, yss):
  zero = torch.min(yss).item()
  eight = torch.max(yss).item()
  th = 1e-3  # threshold
  cutoff = (zero+eight)/2
  count = 0
  for yhats, ys in zip(yhatss, yss):
    yhat = yhats.item()
    y = ys.item()
    if (yhat>cutoff and abs(y-eight)<th) or (yhat<cutoff and abs(y-zero)<th):
      count += 1
  return 100*count/len(yss)

model = train(
    model = model,
    crit = criterion,
    train_data = (xss, yss),
    valid_metric = pct_correct,
    #
    # put your other parameters here
    #
)

print("Percentage correct:", pct_correct(model(xss), yss))
```