

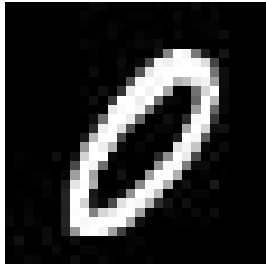
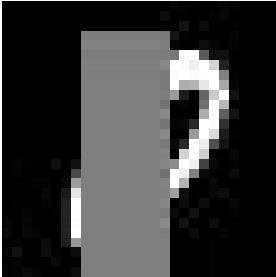
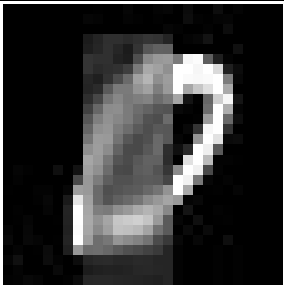
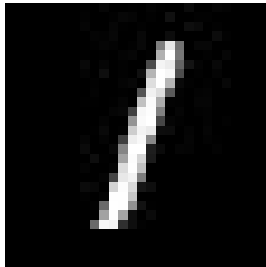
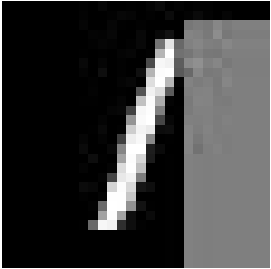
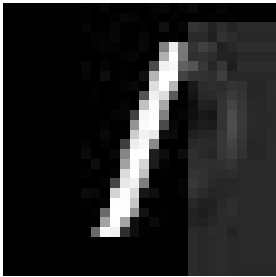
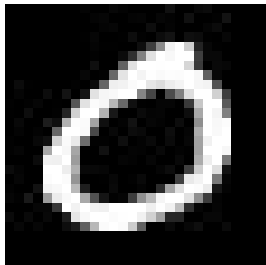
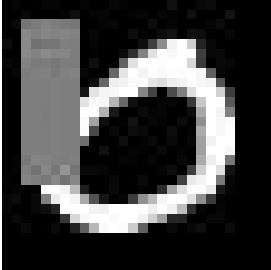
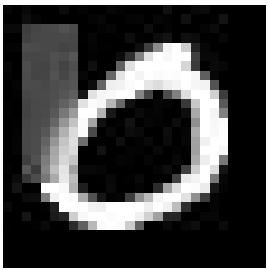
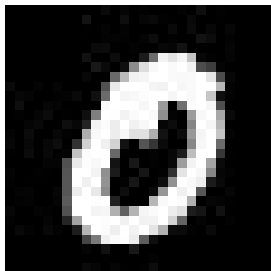
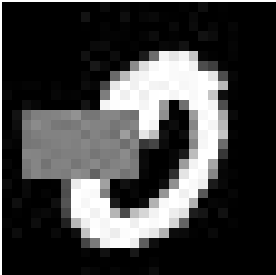
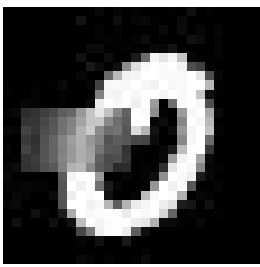
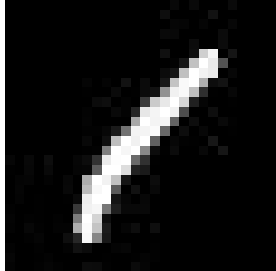
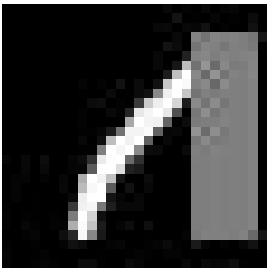
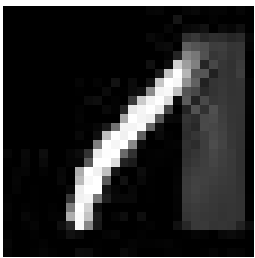
作業環境:

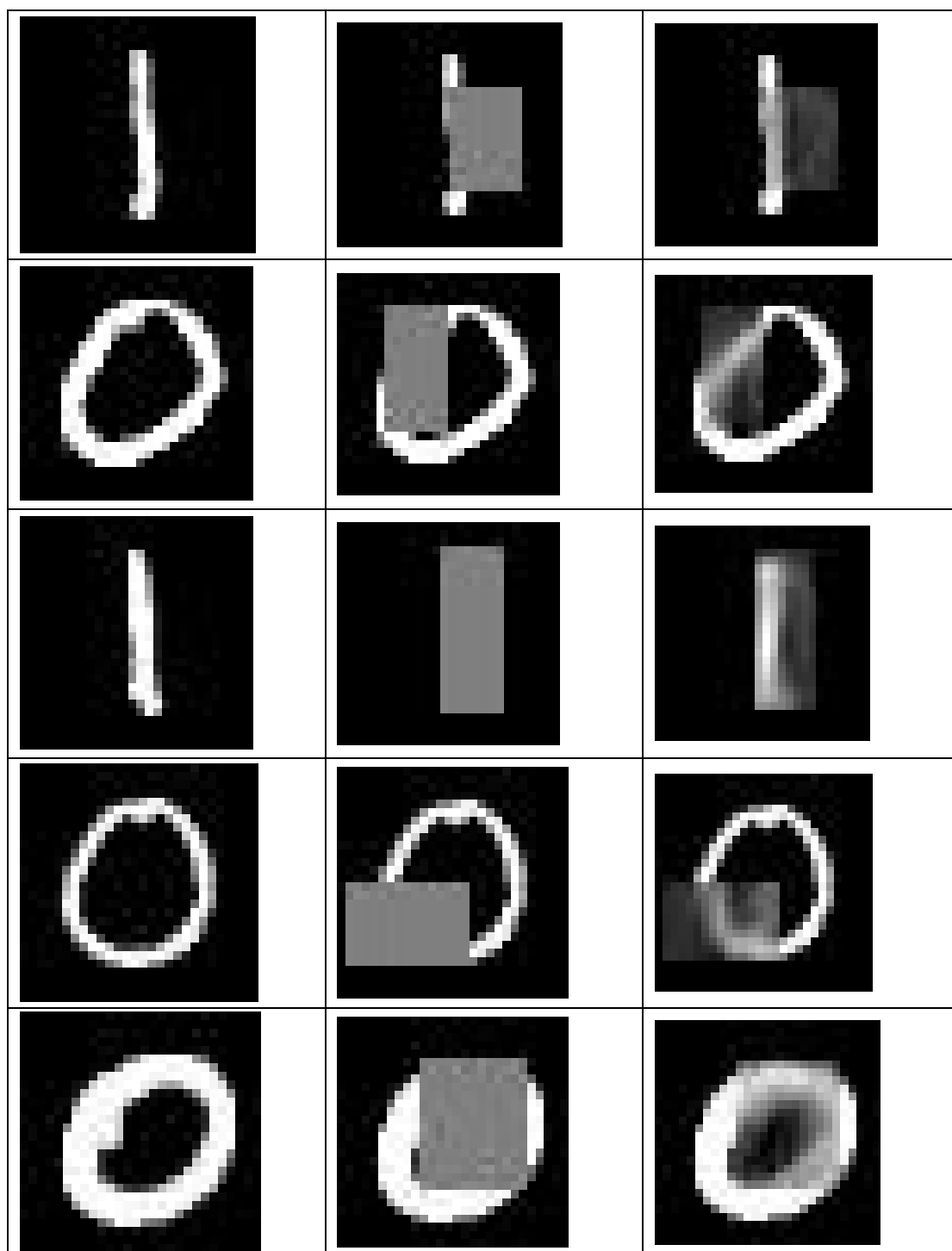
Ubuntu 20.04.1 LTS x86_64

Intel Xeon E5-2620 v4

NVIDIA GeForce RTX 2080

1. Visualization

原始圖片	被汙染的圖片	修復後的圖片
		
		
		
		
		



2. Model setting

Encoder 的 input 是 28×28 ，然後接 linear 到 512，再接 linear 到 128

Decoder 的 input 是 128，然後接 linear 到 512，最後是 linear 到 28×28

基本上完全沒有使用 activation，也就是 linear autoencoder，然後 encoder 和 decoder 的架構完全對稱。

Optimizer 使用 Adam，所有的參數都是使用預設值。

Batch size 是用 128 去訓練，訓練前使用 RandomErasing 去修改圖片再去訓練。

基本上都是用 20 個 epoch 去 train model。

3. Quantitative study

我只有使用 0,1 的 class 來做訓練

並且計算 L2-loss 都是在圖片正歸化到 0~1 之後所計算的

(如果圖片的灰階值範圍用 0~255 會太大，所以我用 0~1)

直接拿 model predict 出來的圖片和 origin 來算 loss，這樣每張圖片大概是

0.020457544089041775

但是如果我有做 postprocessing，也就是沒有被汙染的地方用原始的圖片貼上，

也就是投影片上面的方法，再算一次 loss 的話，大概是 0.01827332191554531

其實可以看到 loss 是有下降的。

4. What you have learned

這個作業其實做起來是滿容易的，但是一開始看會發現效果很差，原因是因為 predict 出來的圖片的 value，他的數值範圍很大，使用 ToTensor 會把圖片的灰階值縮成 0~1，不過預測出來的數值範圍基本上有負數，也會有遠遠超過 1 的值，所以當我把預測出來的 value 減掉最小值，就是讓最小值為 0，再把整張圖片除以最大值，也就是把圖片的數值範圍變成 0~1 之間，這樣還原成原本的灰階範圍時，效果就很不錯。

再使用講義上的 postprocessing，結果就是上面的樣子。我是直接使用 torchvision 裡面的 RandomErasing，就是直接在圖片中使用一塊矩形遮掉，被遮掉的地方數值是 0.5，也就是灰色。還原出來的圖片其實大多都是成功的，不過還原出來的那一塊看起來會有一點髒髒的，圖片中有一張 1 比較特別，因為 1 全部都被遮掉了，但是居然有成功的還原出一個 1，雖然看起來有點模糊，不過這代表方法是成功的。

另外做作業的時候我有嘗試過有很多不同的 hidden size 來做，因為 input size 只有 784 而已，所以通常 hidden size 都要比較小，我有嘗試過使用 4096，也就是 784-4096-784，這樣效果其實會更糟，loss 會在訓練中突然多一大堆，所以最後還是簡單採用了 128。

同場加映:

其實除了 linear autoencoder，也就是不能加任何的 non-linear activation，我有嘗試了一些 non-linear 的方法，其實我把 sigmoid 接在 decoder output 那邊，這作業就做完了，因為效果好得很驚人。