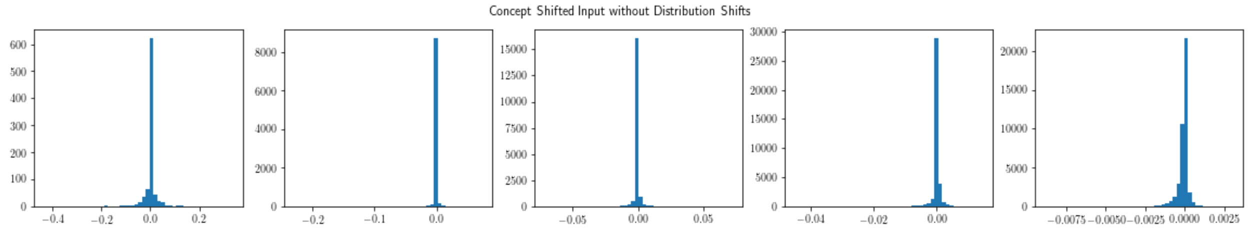
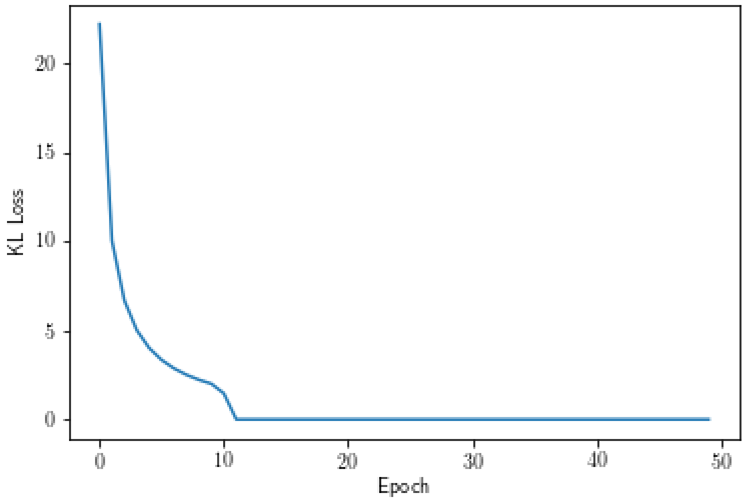
    - Concept shift

```
mu/std: -0.002108/0.041871
max/min: 0.339081/-0.437909
mu/std: -0.000252/0.006205
max/min: 0.075581/-0.230498
mu/std: -0.000123/0.003158
max/min: 0.073064/-0.072005
mu/std: -0.000091/0.001543
max/min: 0.014728/-0.045182
mu/std: -0.000094/0.000390
max/min: 0.003099/-0.008912
```



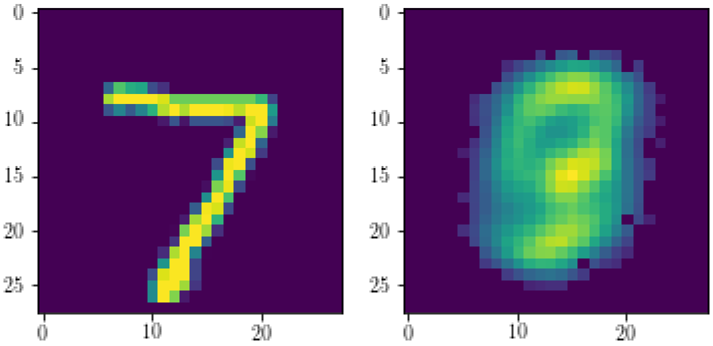
    - Conclusion: It seems that no matter which shift happens, the bottom layer is always affected the most.
        - An interesting thing is that even if concept shift happens, the top/output layer is not affected much. But the bottom/input layer is affected the most.
- 06/05/2021
    - Investigate the denoising phenomenon noted in the Deep Kalman Filter paper
    - Learn a VAE for MNIST
        - **Set the weight of KL divergence to about 30**; single layer 100-size fully-connected neural network for both recognition and generative nets; 10 latent size for *z*
            - Parameters:
                - mini_batchsize = 64
                - no_epochs = 50
                - lr = 0.001
                - z_dim = 10
            - “Posterior collapse” occurs; see plots and logs below



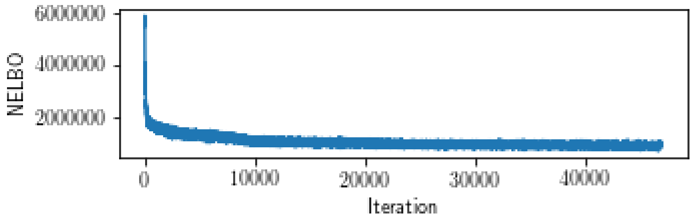
```
[2021-06-05 17:44:43.165083] Epoch: 0002 total
cost= 86.518294632 log-likelihood term=
64.310820505 kl/regluarization term=
22.207474127
[2021-06-05 17:44:47.038463] Epoch: 0003 total
cost= 41.999615222 log-likelihood term=
31.961637134 kl/regluarization term=
10.037978088
[2021-06-05 17:44:50.988228] Epoch: 0004 total
cost= 27.986521870 log-likelihood term=
21.306938014 kl/regluarization term=
6.679583856
[2021-06-05 17:44:54.885333] Epoch: 0005 total
cost= 20.987773813 log-likelihood term=
15.979468402 kl/regluarization term=
5.008305411
[2021-06-05 17:44:58.795692] Epoch: 0006 total
cost= 16.790315138 log-likelihood term=
12.784674775 kl/regluarization term=
4.005640363
[2021-06-05 17:45:02.813752] Epoch: 0007 total
cost= 13.990387380 log-likelihood term=
10.652728521 kl/regluarization term=
3.337658859
[2021-06-05 17:45:06.785798] Epoch: 0008 total
cost= 11.992053274 log-likelihood term=
9.131286625 kl/regluarization term= 2.860766649
[2021-06-05 17:45:10.722516] Epoch: 0009 total
cost= 10.492452305 log-likelihood term=
7.989354366 kl/regluarization term= 2.503097939
```

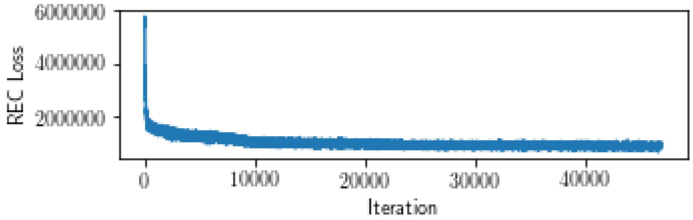
```
[2021-06-05 17:45:14.813366] Epoch: 0010 total
cost= 9.326128521 log-likelihood term=
7.101178546 kl/regluarization term= 2.224949975
[2021-06-05 17:45:18.948932] Epoch: 0011 total
cost= 8.393613520 log-likelihood term=
6.391175764 kl/regluarization term= 2.002437756
[2021-06-05 17:45:22.963609] Epoch: 0012 total
cost= 7.269937655 log-likelihood term=
5.810364966 kl/regluarization term= 1.459572689
[2021-06-05 17:45:27.084931] Epoch: 0013 total
cost= 5.326592045 log-likelihood term=
5.326589155 kl/regluarization term= 0.000002890
[2021-06-05 17:45:30.935306] Epoch: 0014 total
cost= 4.916507074 log-likelihood term=
4.916503670 kl/regluarization term= 0.000003404
```

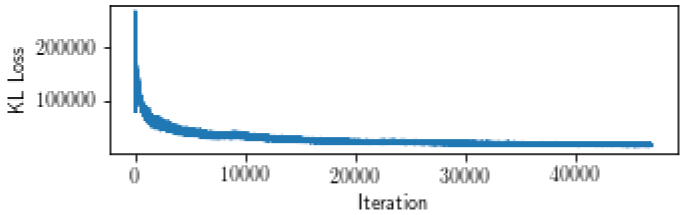
- Thus the decoder/recognition model takes all the responsibility of generating images but not rely on the encoder. As a result, see the reconstruction plot

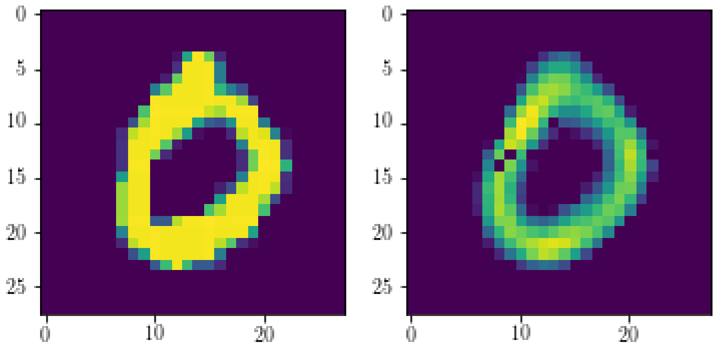


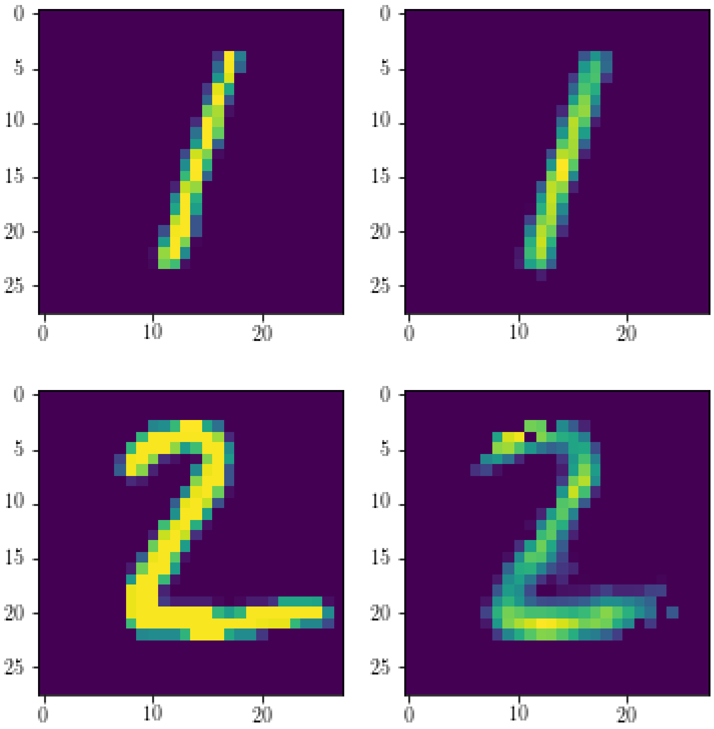
- ■ Experiment setup:
  Single layer network
  mini_batchsize = 64
  no_epochs = 50
  lr = 0.001
  z_dim = 20
  net_size = 100
  - Empirical caveat: Images are at the original scale: {0, …, 255} in integers are easier to learn than the 0-1 scaling, i.e., divide all dimensions by 255.
  - See results below for images at the original scale


6000000
4000000
2000000
NELBO
0
10000
20000
30000
40000
Iteration

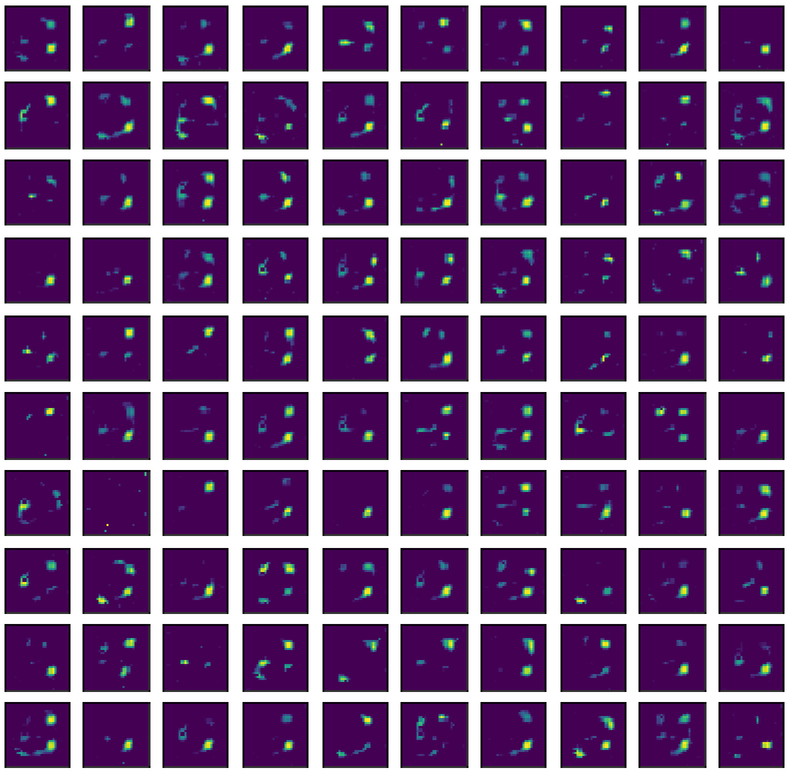

6000000
4000000
2000000
REC Loss
0
10000
20000
30000
40000
Iteration


200000
100000
KL Loss
0
10000
20000
30000
40000
Iteration


0
5
10
15
20
25
0
10
20
0
5
10
15
20
25
0
10
20



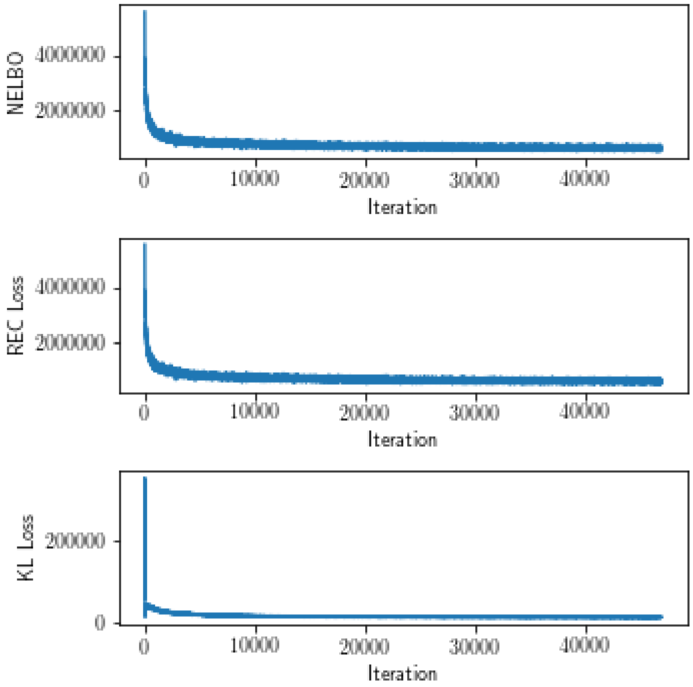
- Random samples do not look good.



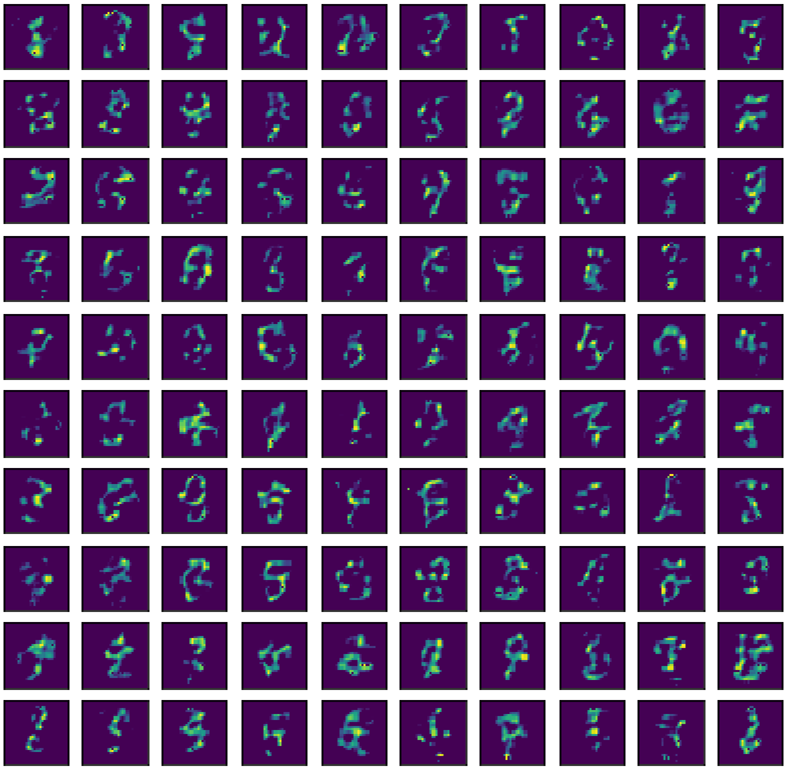
- Experiment setup:
  three-layer network
  mini_batchsize = 64

no_epochs = 50
lr = 0.001
z_dim = 100
net_size = 200

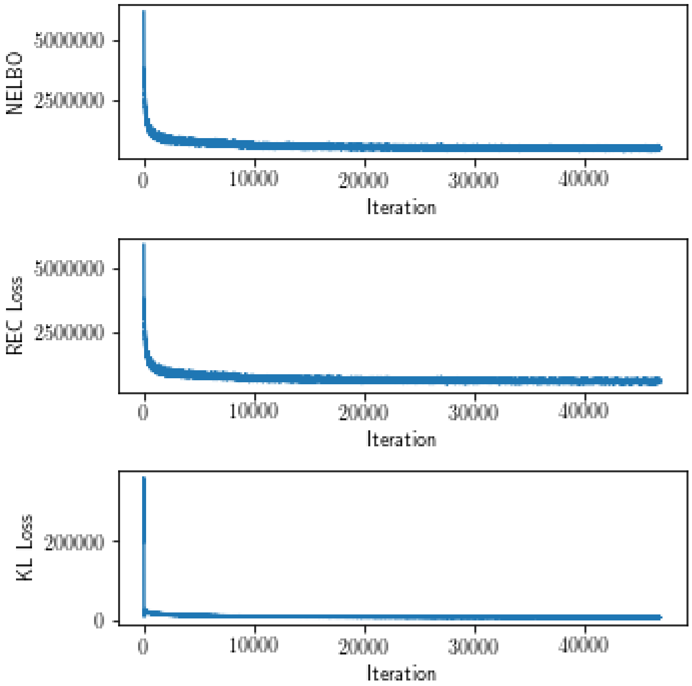
- Empirical caveat: Images are at the original scale: {0, …, 255} in integers are easier to learn than the 0-1 scaling, i.e., divide all dimensions by 255.
- See results below for images at the original scale



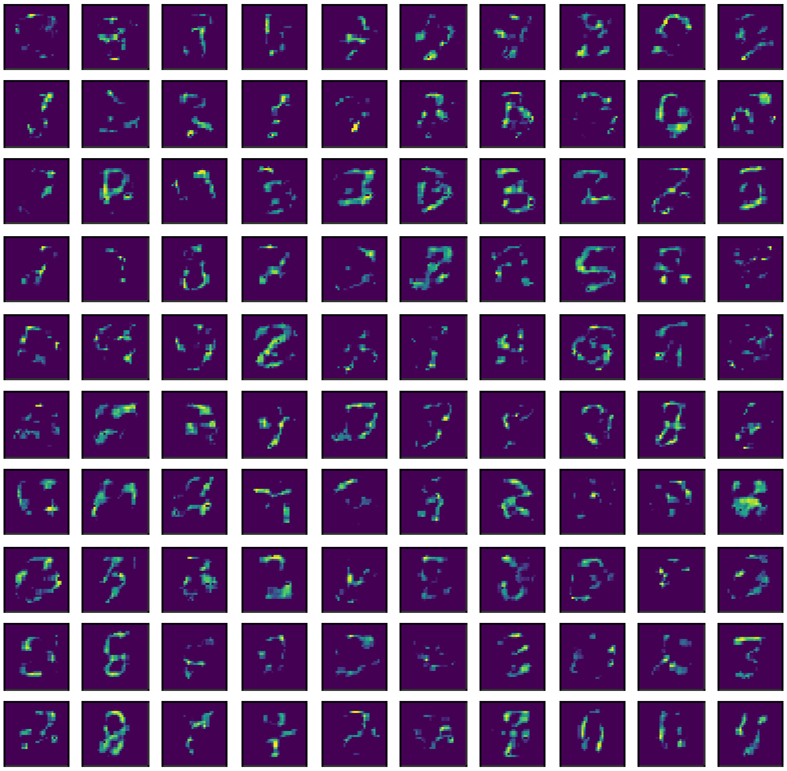
- Random samples do not look good.



- Experiment setup:
  three-layer network
  mini_batchsize = 64
  no_epochs = 50
  lr = 0.001
  z_dim = 500
  net_size = 1000
  - See results below for images at the original scale



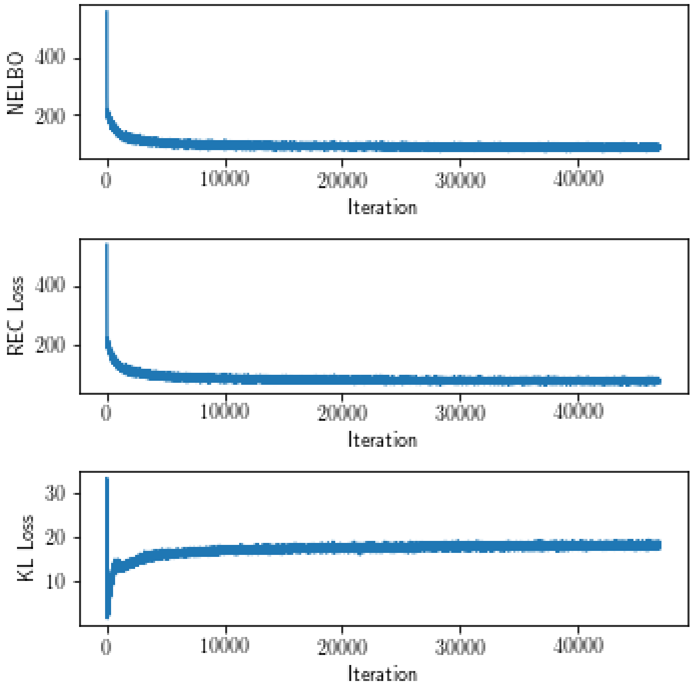
- Random samples do not look good.



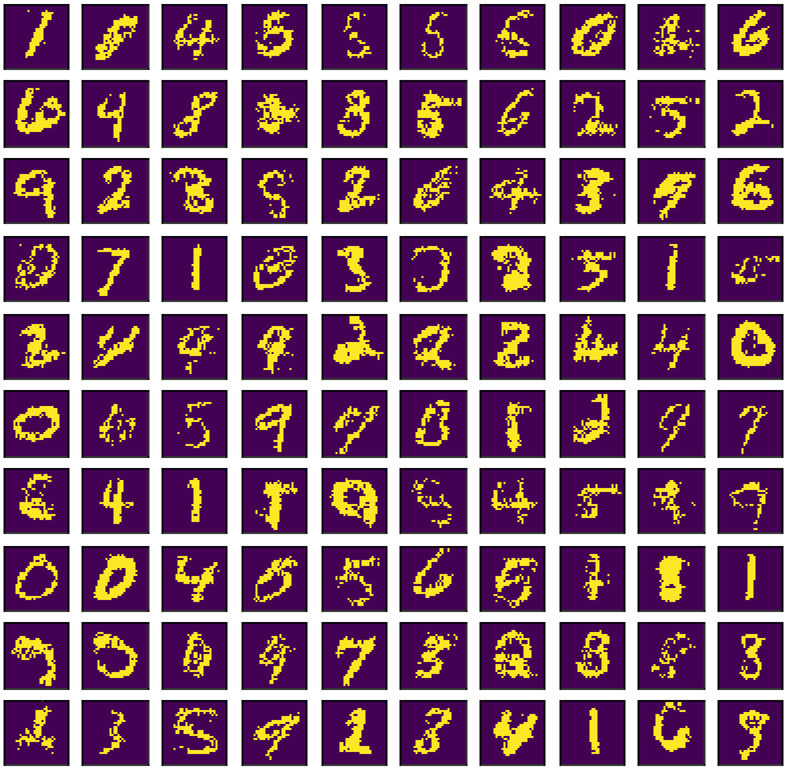
I just found the above results are generated from the neural network whose final layer has a RELU activation function; usually, the final layer should not come with an activation function. If I replace the RELU function with None, then the

training seems to become unstable. A plausible reason is that RELU >=0, which corresponds to 0~255 input values.

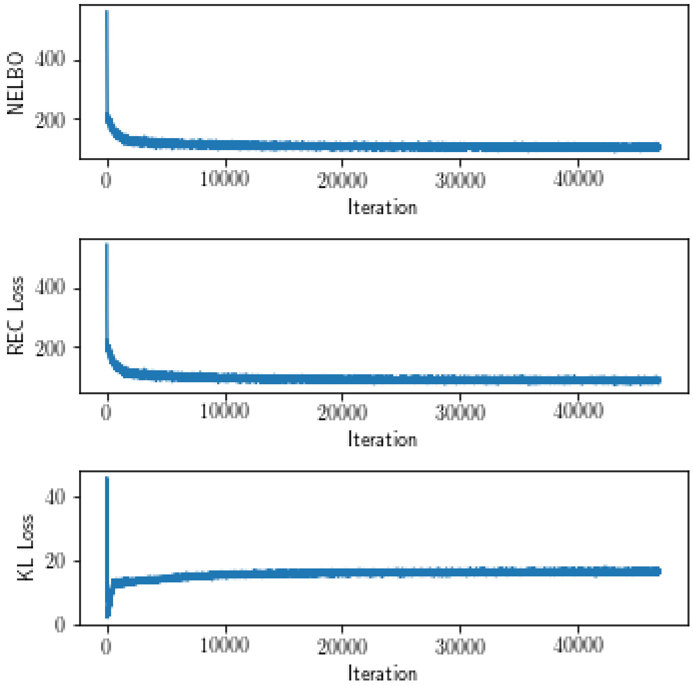
- Experiment setup:
- Binarized MNIST with Bernoulli output
  three-layer network
  mini_batchsize = 64
  no_epochs = 50
  lr = 0.001
  z_dim = 100
  net_size = 200
    - See results below for images at the original scale



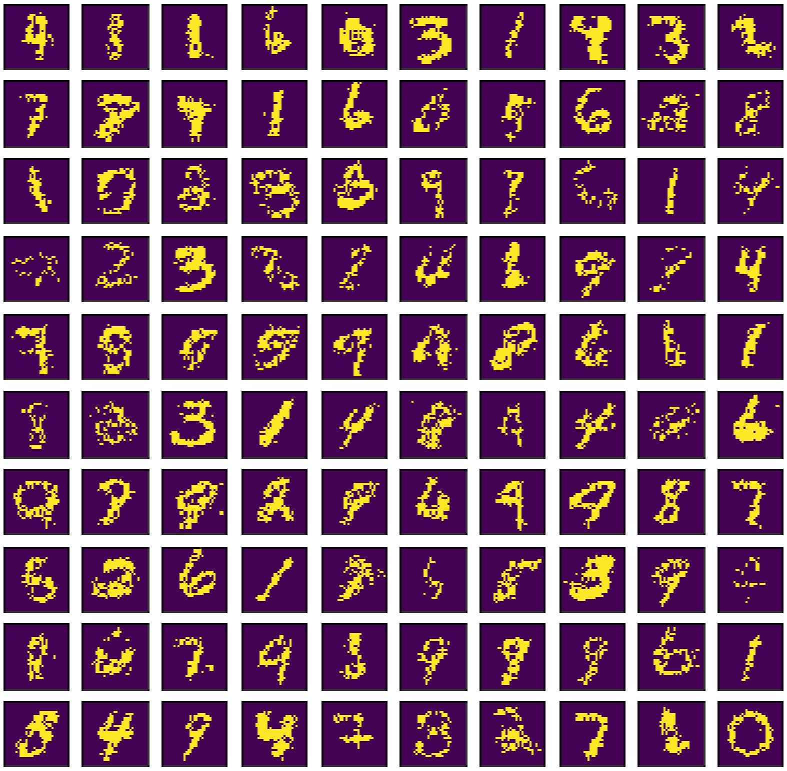
    - Random sample.



- Experiment setup:
- Scaling MNIST [0,1] with Bernoulli output
  three-layer network
  mini_batchsize = 64
  no_epochs = 50
  lr = 0.001
  z_dim = 100
  net_size = 200
  - See results below for images at the original scale

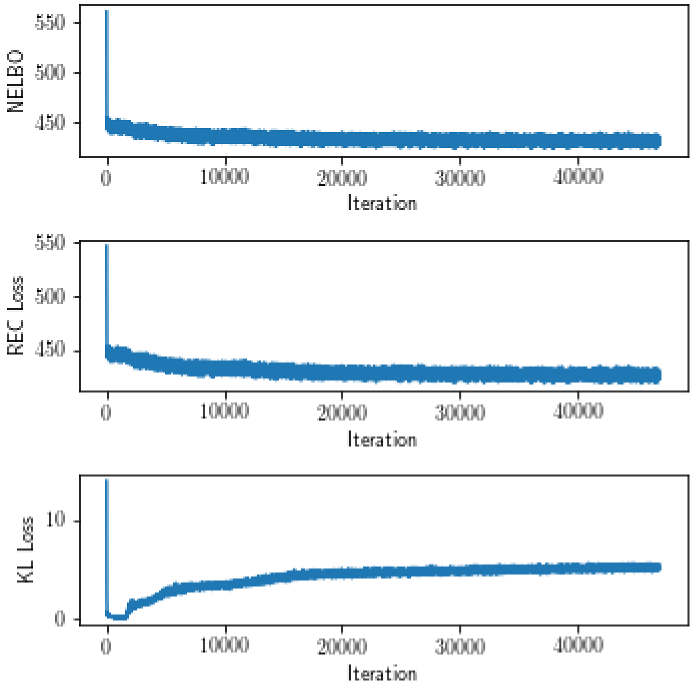


    - Random samples.

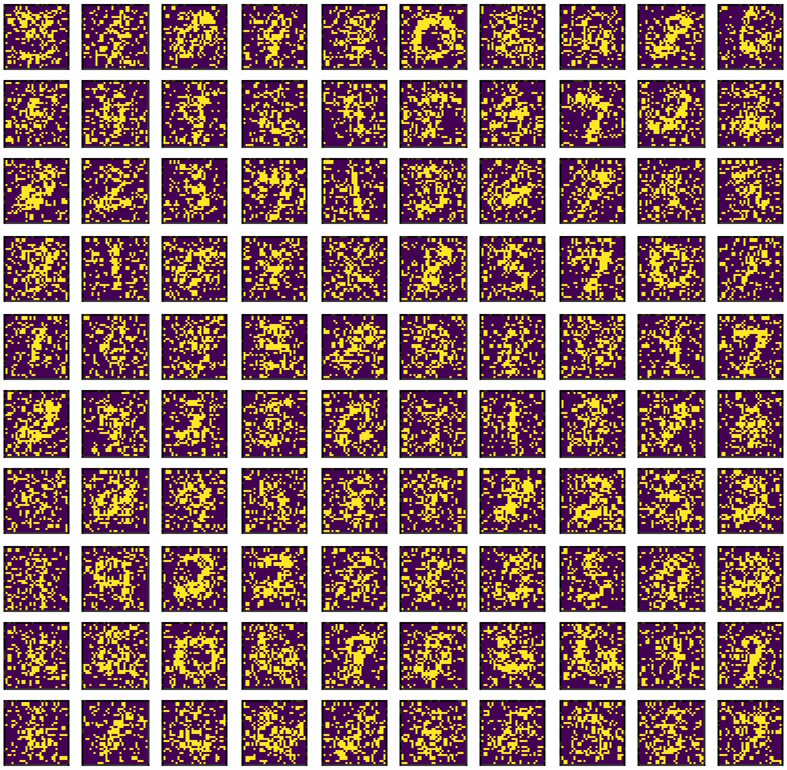


- 06/06/2021
    - Now we investigate the *denoising phenomenon* presented in the paper “Deep Kalman Filter”

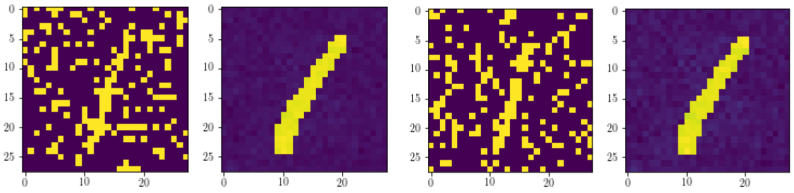
- We use **binarized MNIST** and randomly flip the pixel with probability 0.2. The reconstruction function is binary cross entropy.
    - The denoising phenomenon does not occur; it learns the noisy structure.
    - Experiment setup:
    - Scaling MNIST [0,1] with Bernoulli output
      three-layer network
      mini_batchsize = 64
      no_epochs = 50
      lr = 0.001
      z_dim = 100
      net_size = 200
        - See results below for images at the original scale

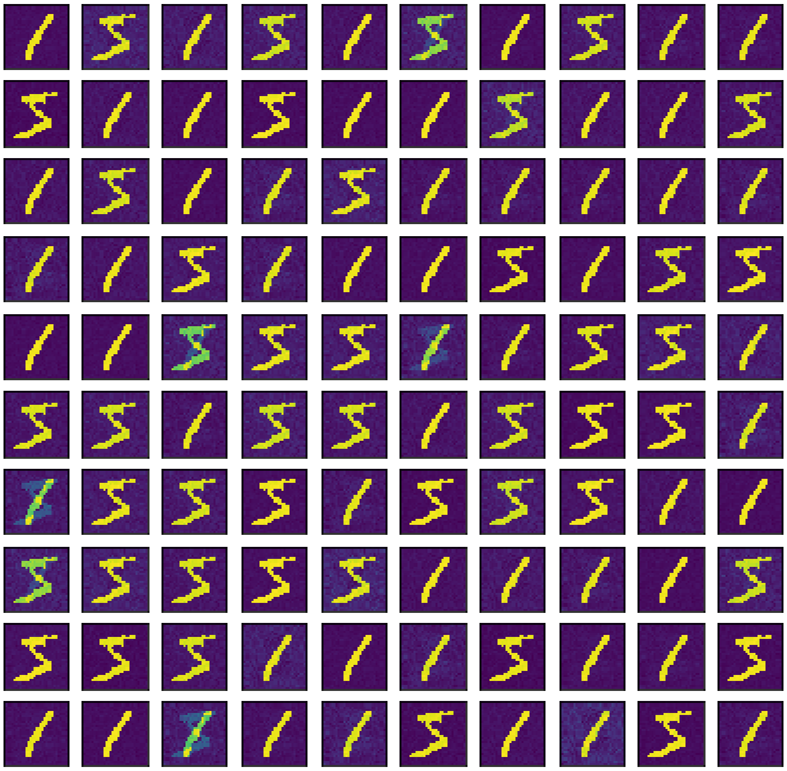


        - Random samples.



- Small Healing MNIST experiment
  - training data

        - Sample data (**mean value** of the Bernoulli variable): it has a denoise phenomenon. ***But it is hard to say whether this is what we want.*** If the noise has interesting details then we lose information; if the noise is pure noise, then we can say we succeed in denoising.
            - The reconstruction can not reproduce the detailed noise in the input image.
            - 



    - Hypothesis: variational autoencoder learns an average representation of images but overlooks the details. A reason is that the generative model shares parameters among the whole dataset. So the learned representation is affected by other data samples.
      Thus there should be some network dedicated to consider sample-specific details.