Replicating Ian Goodfellow GAN

By Ang Ming Liang

About Me

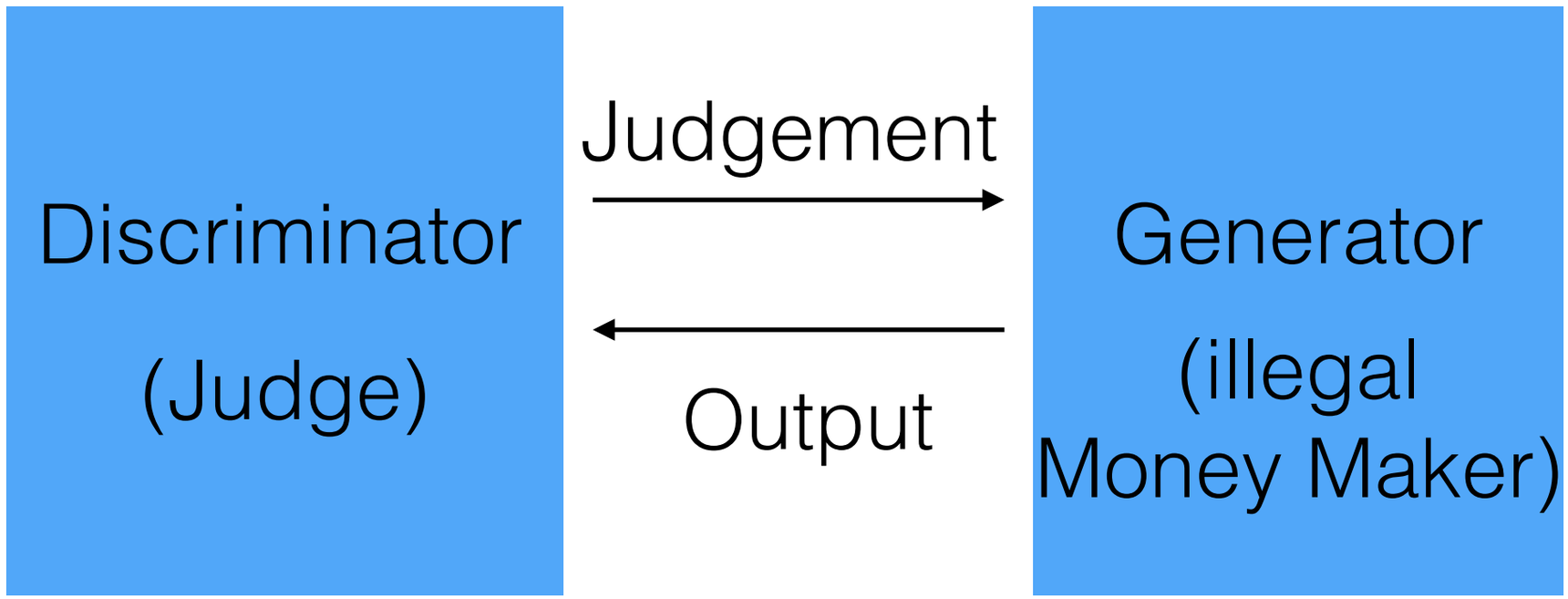
I am a machine learning enthusiast

Currently serving my national service

Graduated Junior College (Equivalent to high school) in 2017

Will enter NUS Computational Biology program in 2019

The idea behind GANs



Pseudo Code

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Source : Goodfellow 2014

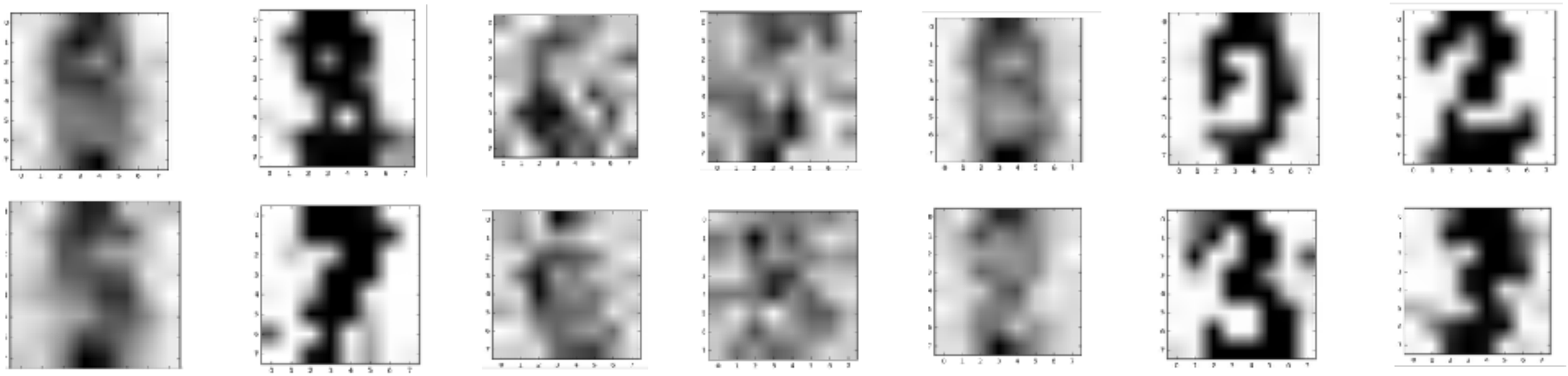
In practise

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \quad (1)$$

In practice, equation 1 may not provide sufficient gradient for G to learn well. Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data. In this case, $\log(1 - D(G(\boldsymbol{z})))$ saturates. Rather than training G to minimize $\log(1 - D(G(\boldsymbol{z})))$ we can train G to maximize $\log D(G(\boldsymbol{z}))$. This objective function results in the same fixed point of the dynamics of G and D but provides much stronger gradients early in learning.

Source : Goodfellow 2014

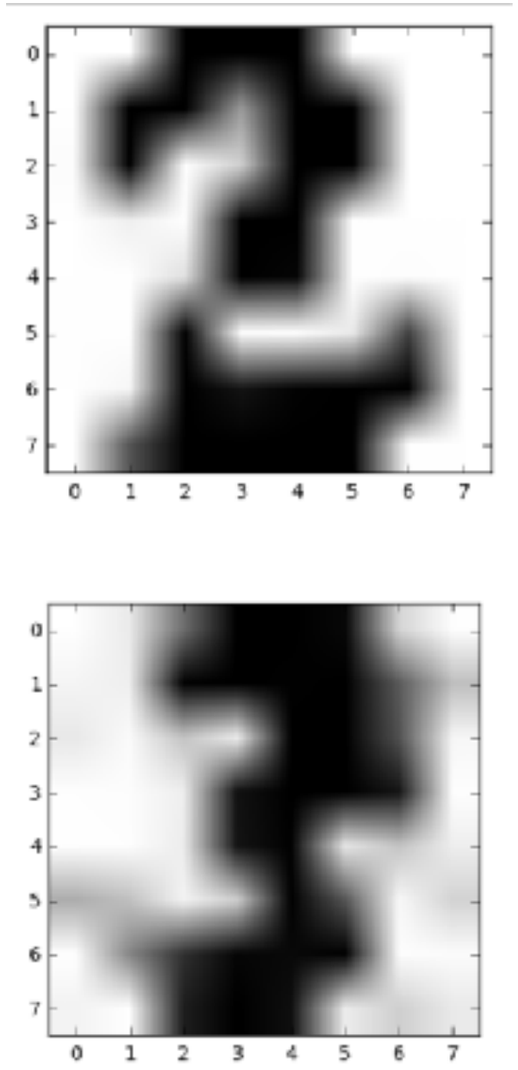
Result



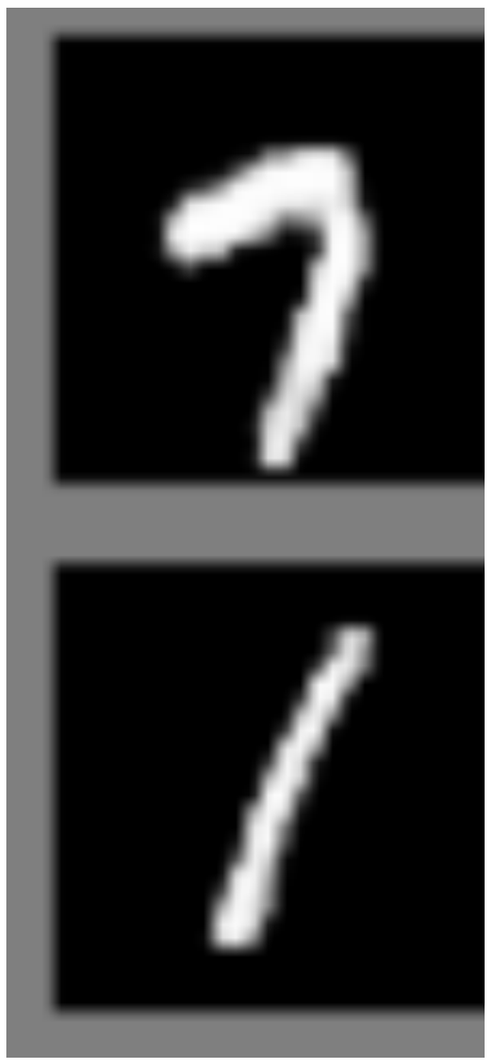
Vanilla | Tanh | Dropout (p = 0.5) | Leaky _Relu | Adam | Adam + Tanh | Adam + Tanh + Dropout (p = 0.5)

This diagram doesn’t show u how many times I took to get these images

Result



Adam + Tanh
+ Dropout
(p = 0.5)



MNIST result in
GAN paper

Conclusion

1. I have successfully replicated Ian Goodefellow original GAN
2. When improving your GAN try Tanh or Adam first before trying anything else.
3. The best results come from adding Tanh + Adam + Dropout. But the most reliable and consistent results come from Tanh + Adam

Lose ends

1. "Weird GAN"

2. Why is tanh so important ?

“Weird” GAN

```
w1.data += learning_rate*w1.grad.data
b.data += learning_rate*b.grad.data

w2.data += learning_rate*w2.grad.data
b2.data += learning_rate*b2.grad.data

w3.data += learning_rate*w3.grad.data
b3.data += learning_rate*b3.grad.data
```

Yet Weird GAN
leads to faster convergence !

```
vx_w1 = rho*vx_w1 + w1.grad.data
vx_b1 = rho*vx_b1 + b.grad.data
w1.data += learning_rate*vx_w1  # Gr
b.data += learning_rate*vx_b1

vx_w2 = rho*vx_w2 + w2.grad.data
vx_b2 = rho*vx_b2 + b2.grad.data
w2.data += learning_rate*vx_w2
b2.data += learning_rate*vx_b2

vx_w3 = rho*vx_w3 + w3.grad.data
vx_b3 = rho*vx_b3 + b3.grad.data
w3.data += learning_rate*vx_w3
b3.data += learning_rate*vx_b3

#print(w1.data)

w1.grad.data.zero_()
b.grad.data.zero_()

w2.grad.data.zero_()
b2.grad.data.zero_()

w3.grad.data.zero_()
b3.grad.data.zero_()
```

This is missing

Weird GAN

Vanilla GAN

What does grad.data.zero do ?

```
input = Variable(data)
# Get the features
features = feature_extractor(input)

# Compute first loss and get the gradients for it
loss1 = task1(features)
loss1.backward(retain_graph=True)
# This add the gradients wrt loss1 in both the "task1" net and the "feature_extra
# So each parameter "w" in "feature_extractor" has it gradient d(loss1)/dw

# Perform the second task and get the gradients for it as well
loss2 = task2(features)
loss2.backward()
# This will add gradients in "task2" and accumulate in "feature_extractor"
# Now each parameter in "feature_extractor" contains d(loss1)/dw + d(loss2)/dw
```

Source : https://discuss.pytorch.org/t/why-do-we-need-to-set-the-gradients-manually-to-zero-in-pytorch/4903/9

So what does this mean for “Weird GAN” ?

w1.grad.data = d(D loss t=1)/dw1 + d(G loss t=1)/dw1+
d(D loss t=2)/dw1 + d(G loss t=2)/dw1+
d(D loss t=3)/dw1 + d(G loss t=3)/dw1+
d(D loss t=3)/dw1 + d(G loss t=3)/dw1 + ….

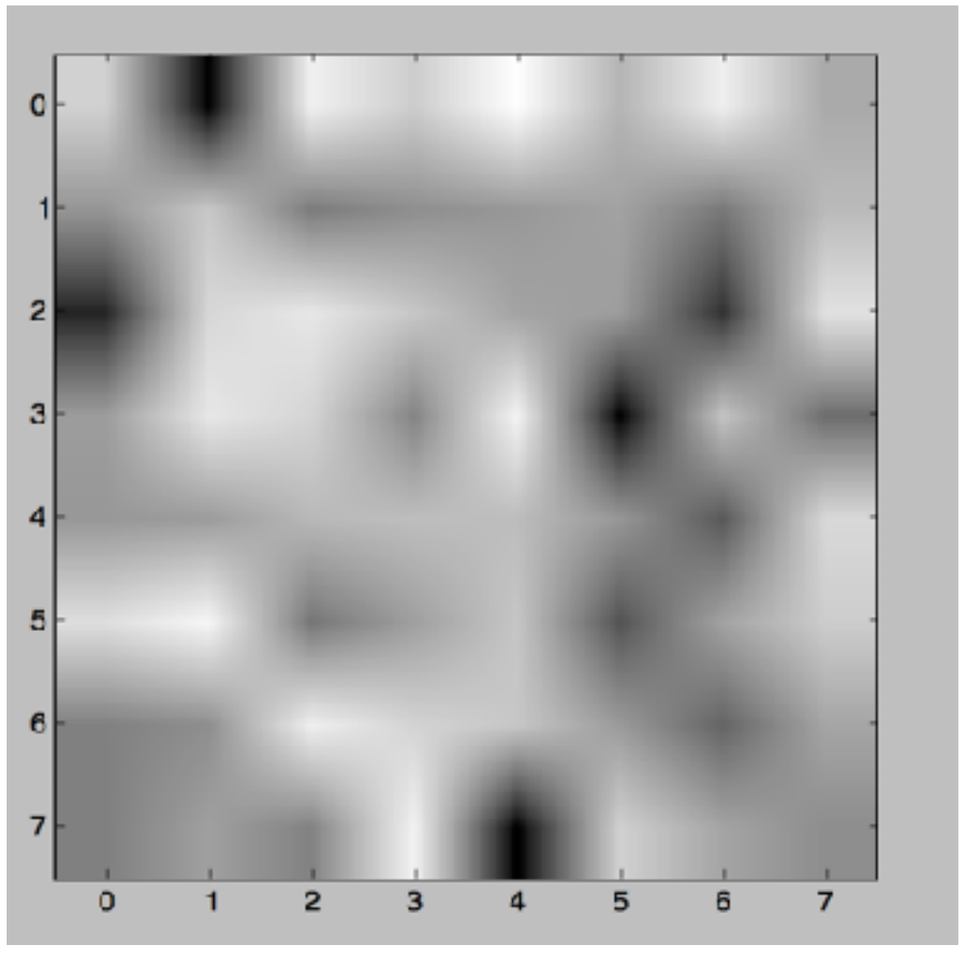
Does it work because ?

1. Include d(G loss t=n)/dw1

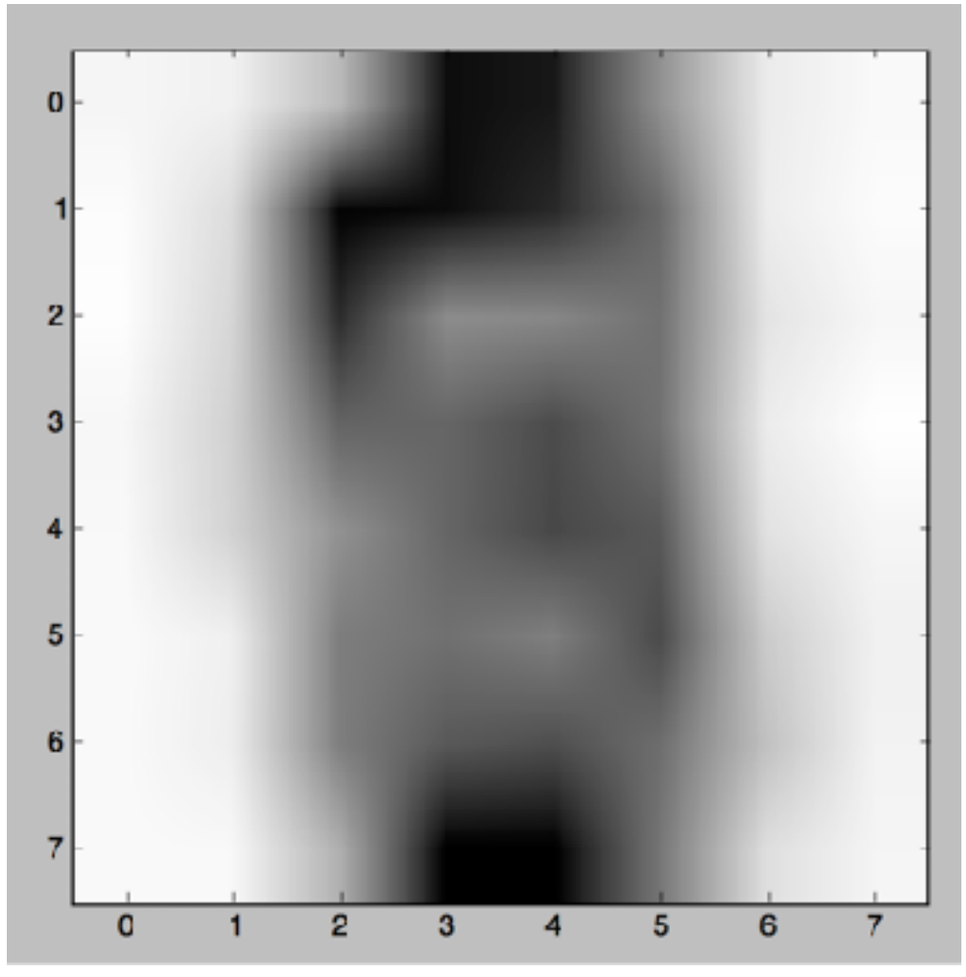
2. d(D loss t=1)/dw1 + d(D loss t=2)/dw1 + ..

Option 1

Epochs = 1000



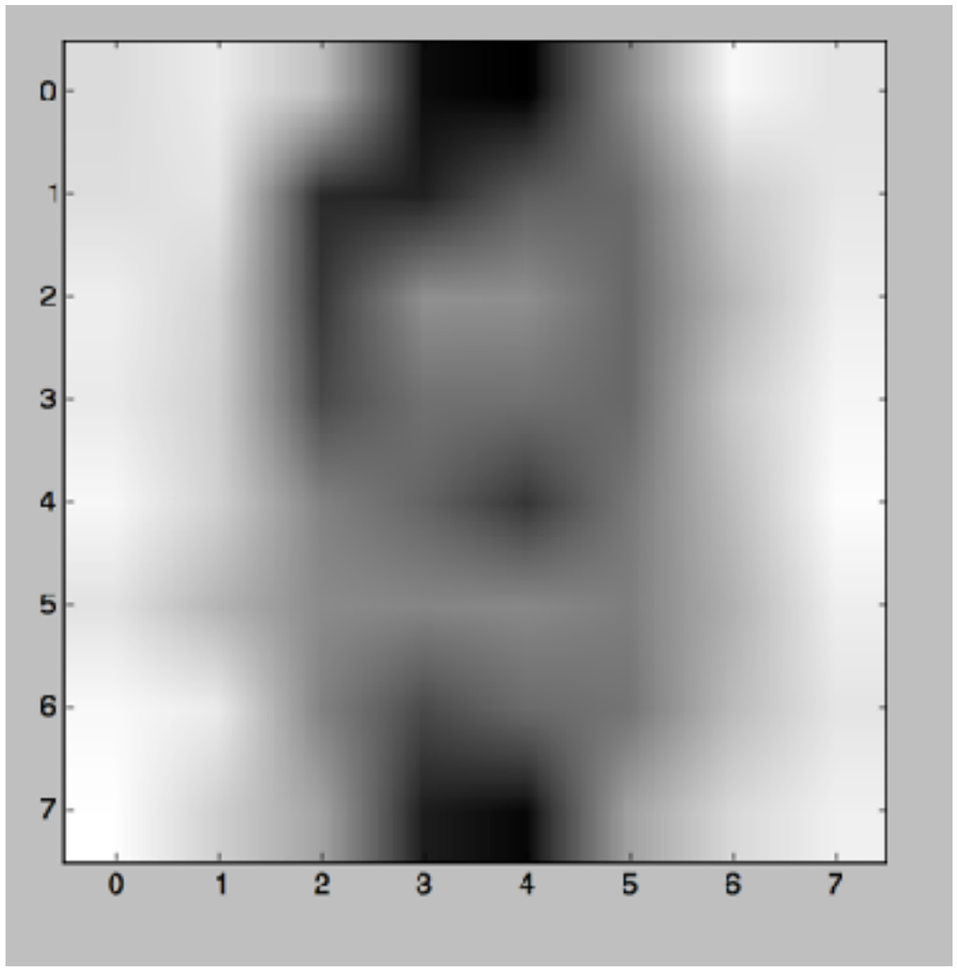
Cross talk



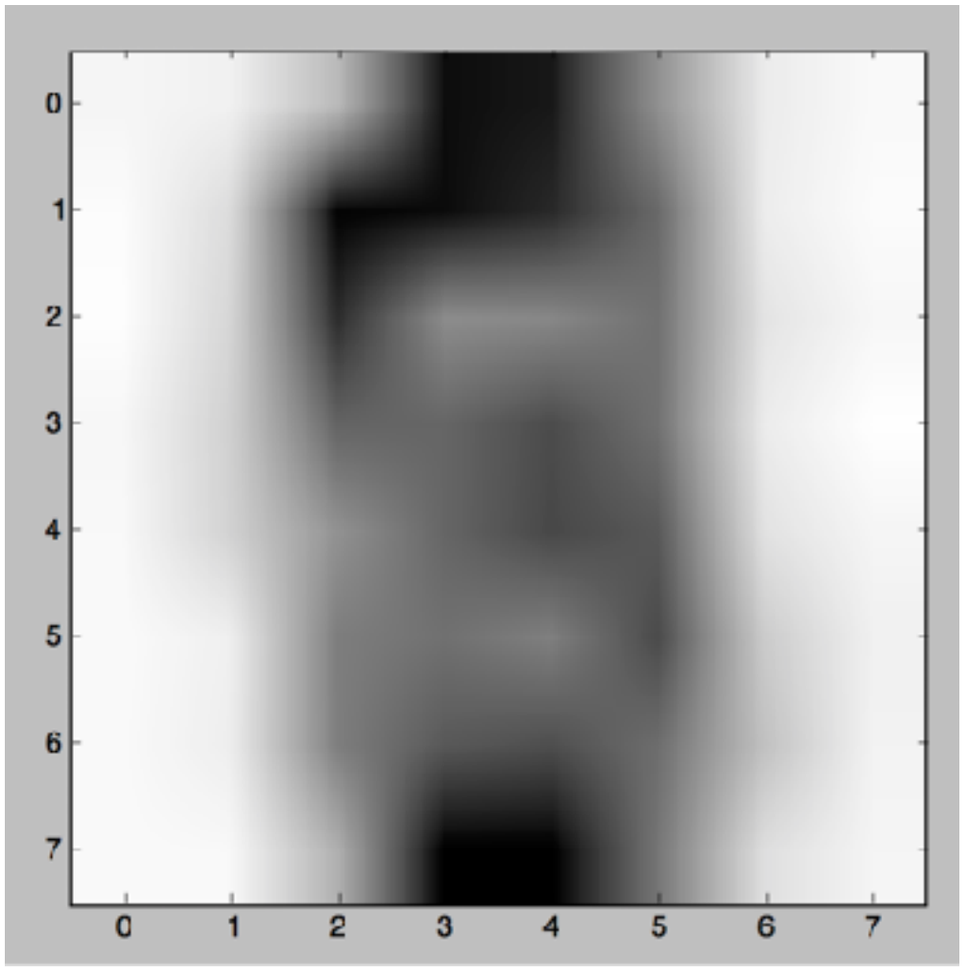
Weird

Option 2

Epochs = 1000



Momentum
rho=0.999

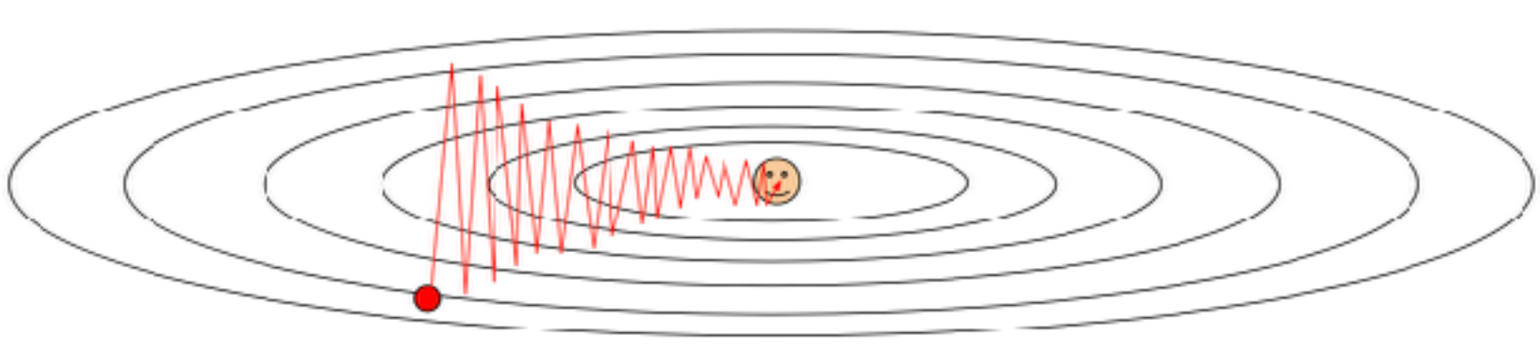


Weird

What does this mean

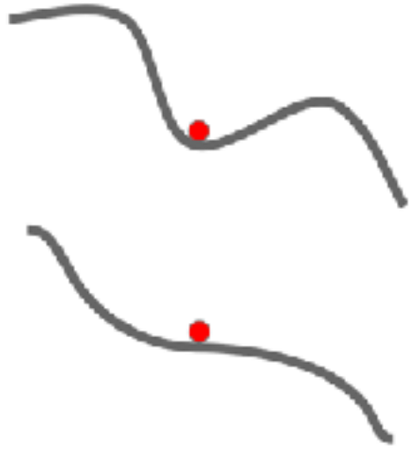
1. Poor conditioning

Very slow progress along shallow dimension, jitter along steep direction



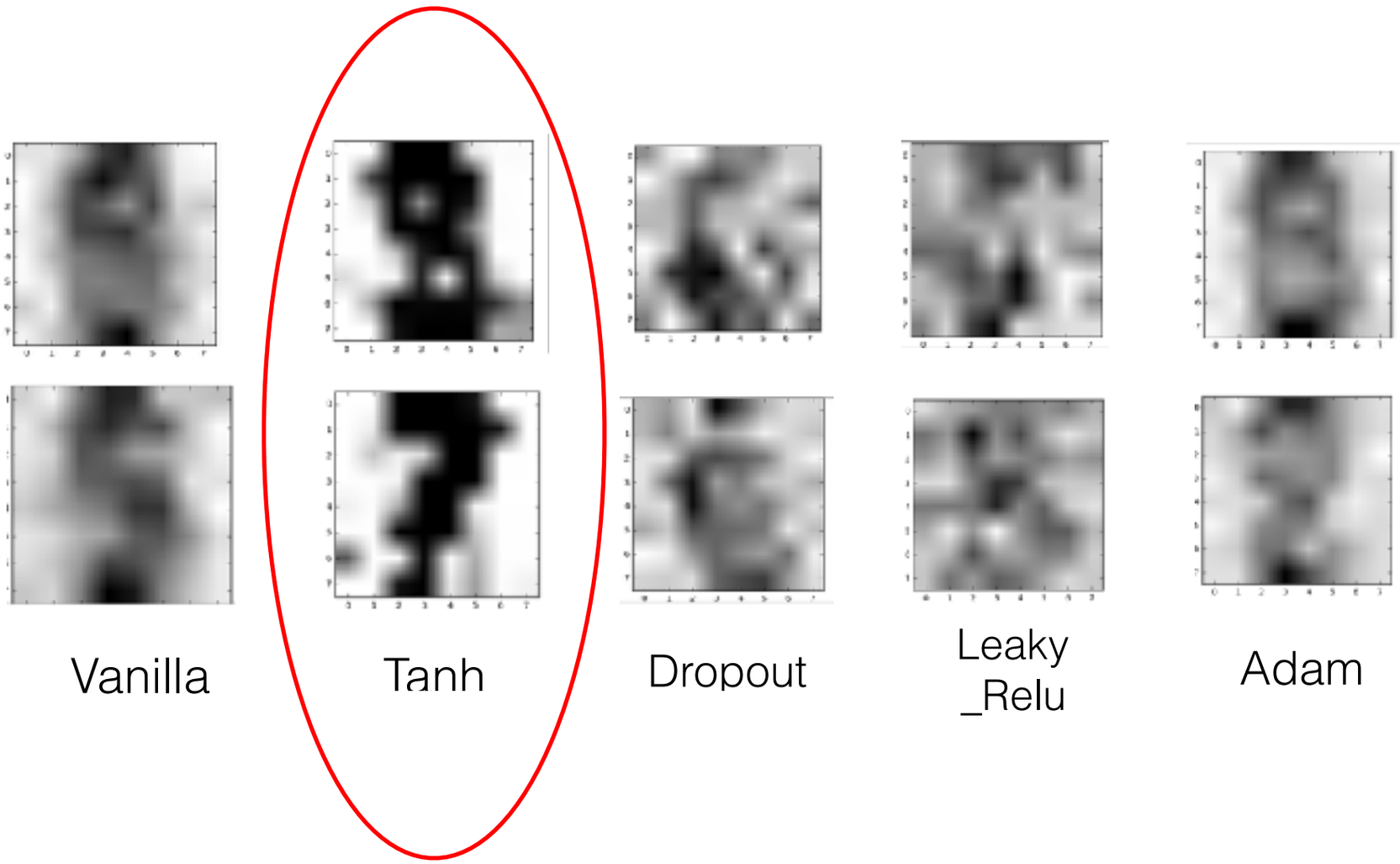
Loss function has high **condition number**: ratio of largest to smallest singular value of the Hessian matrix is large

2. Saddle points and local minima



Source : CS231n Stanford 2017

Tanh mystery



Best improvement

Why ?

The accepted answer

1. Normalize the inputs

- normalize the images between -1 and 1
- Tanh as the last layer of the generator output

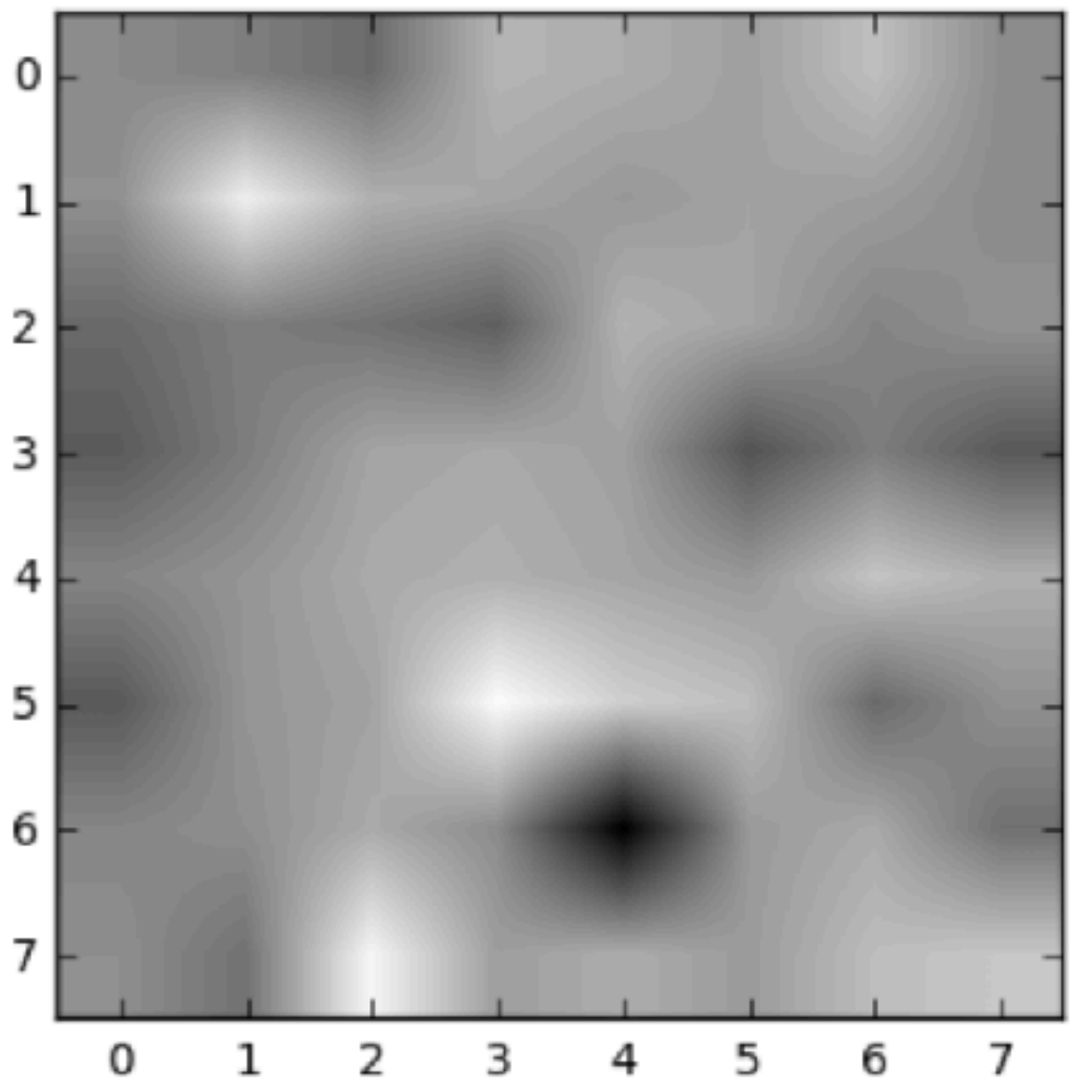
source : https://github.com/soumith/ganhacks

Because **normalisation**

Incomplete at best

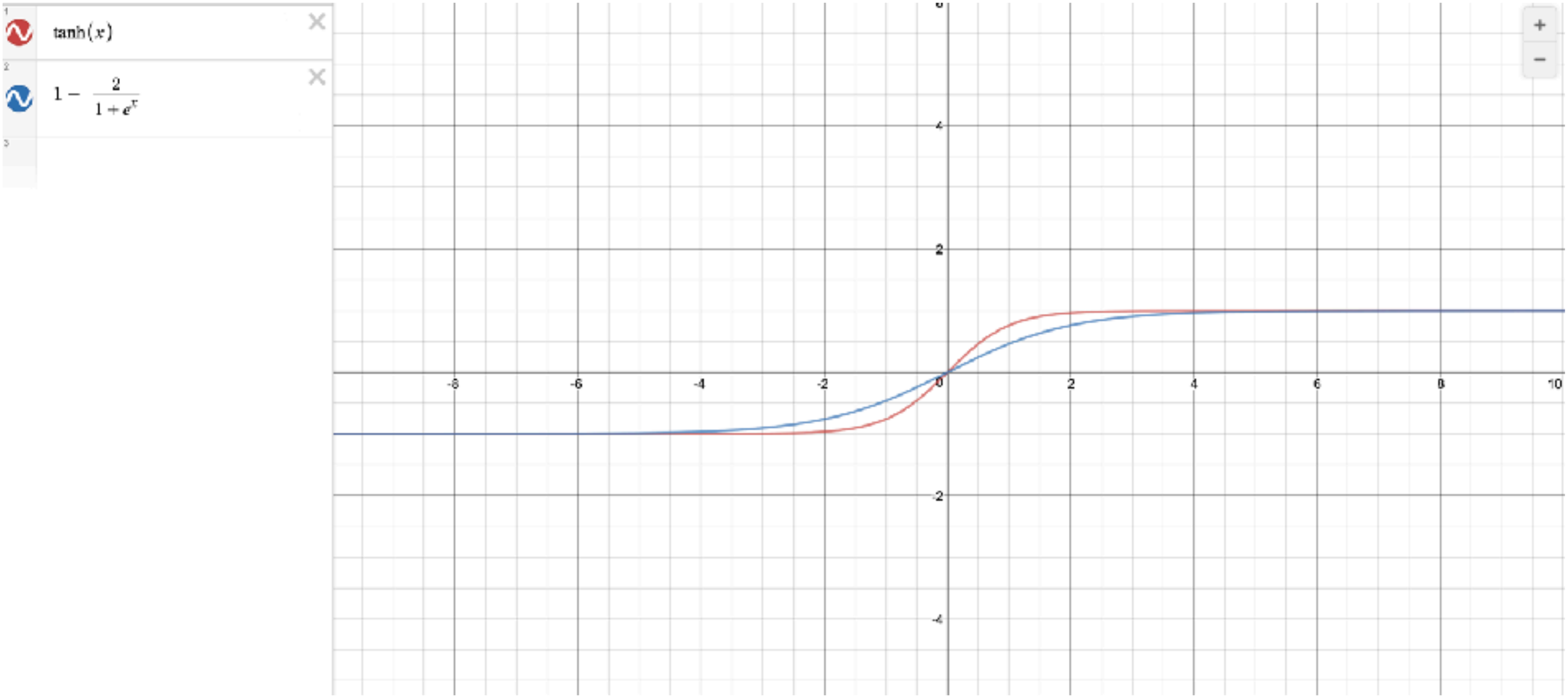
Experiment 1

We normalise the values to ~N(0,1)

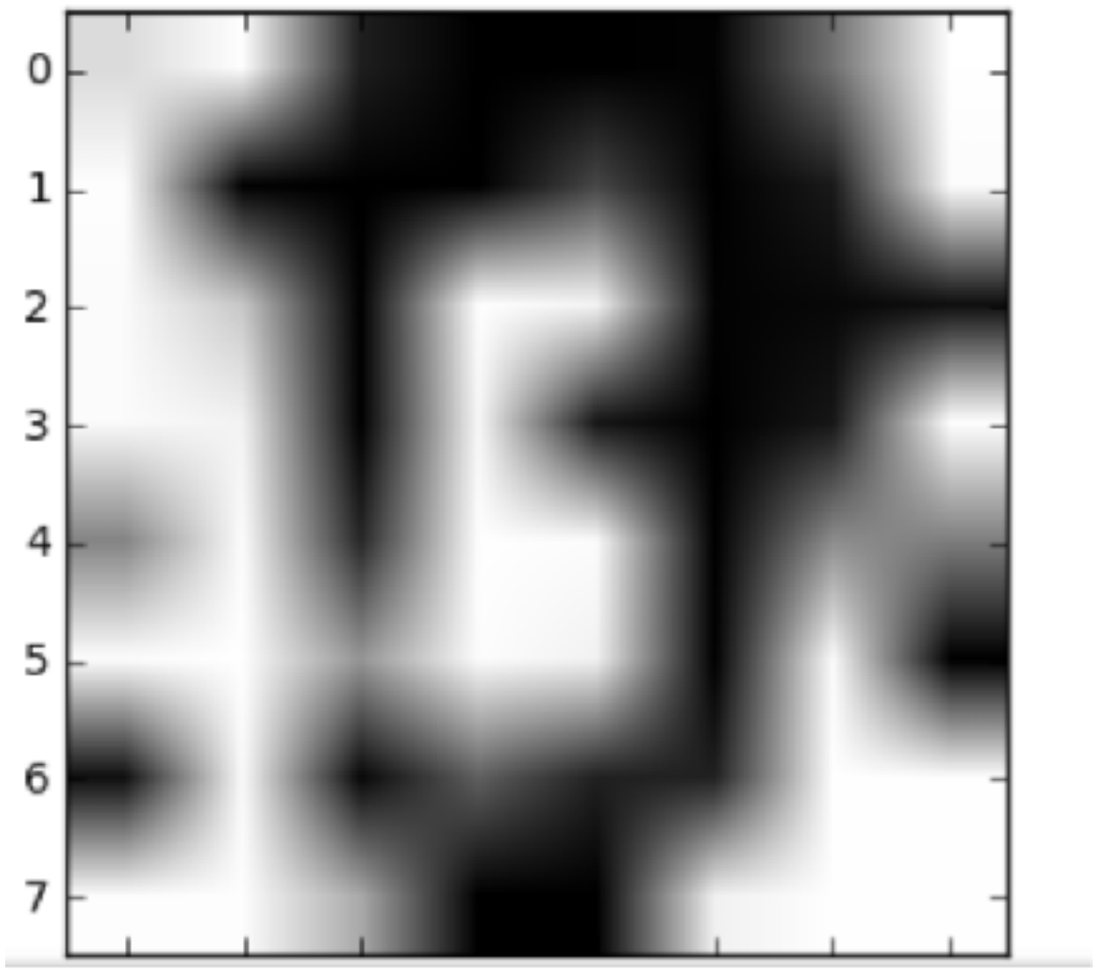


Experiment 2

We try sigmoid but multiply by 2 and min 1

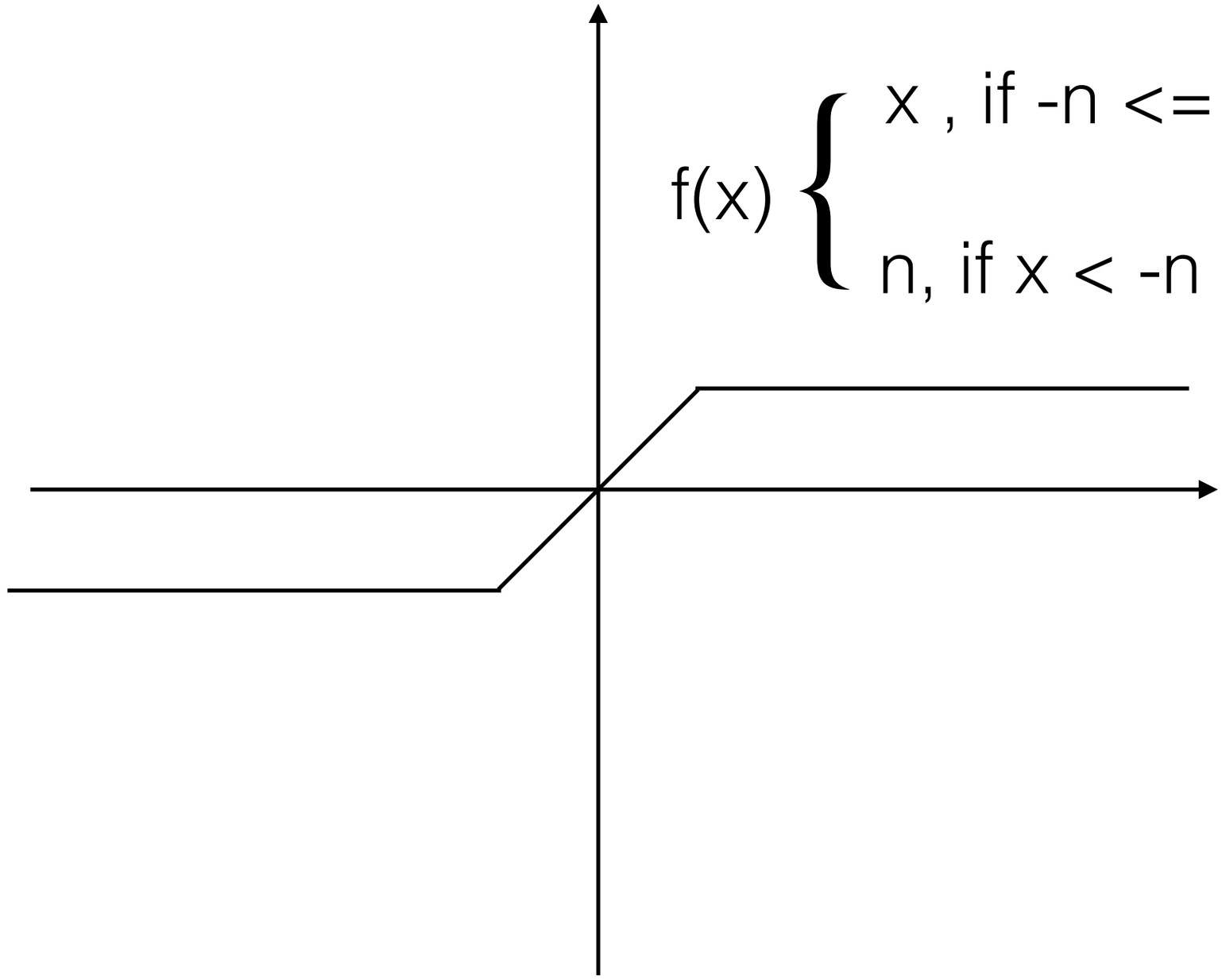


Experiment 2

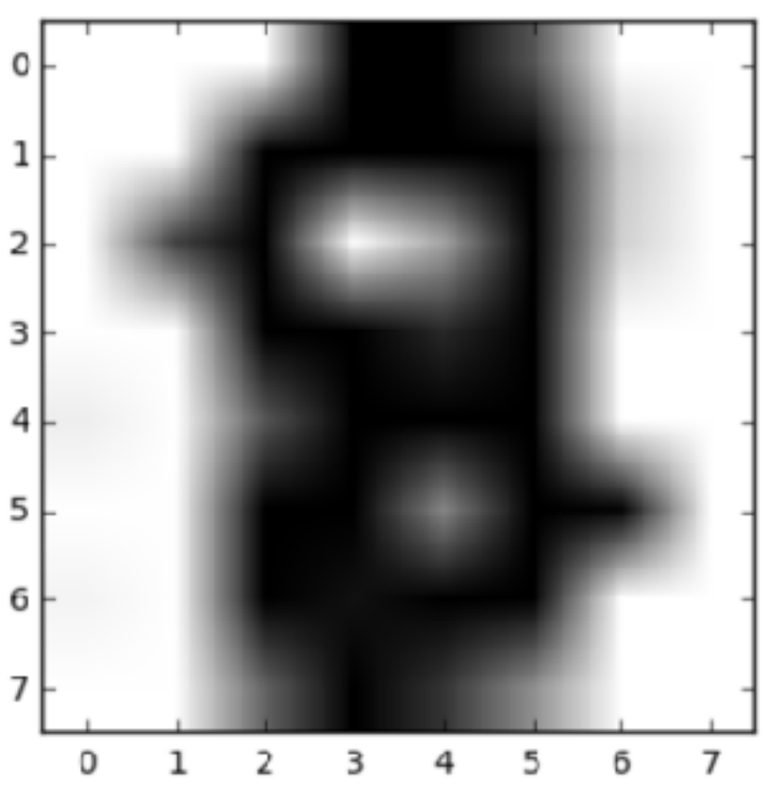


Experiment 3

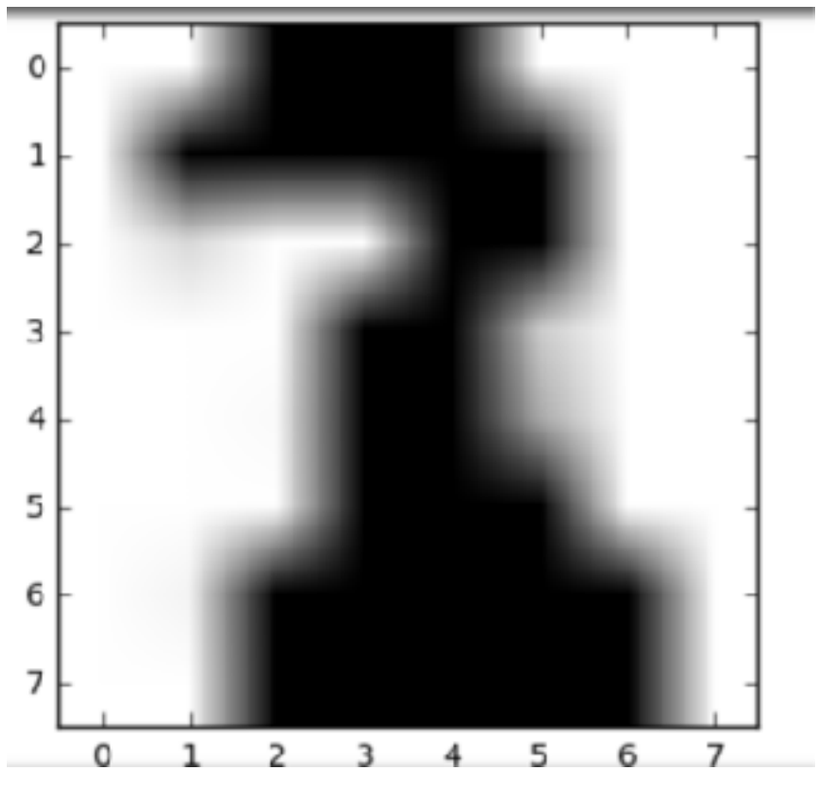
$$f(x) \begin{cases} x \text{ , if } -n <= x <= x \\ n, \text{ if } x < -n \text{ or } x > n \end{cases}$$



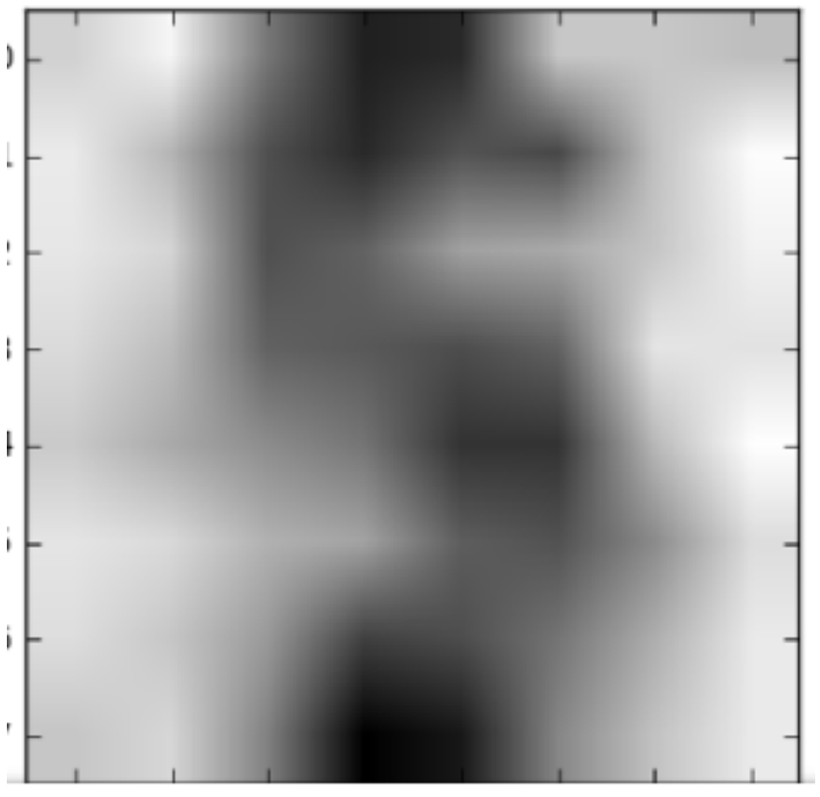
Experiment 3



Clamped
min=-1.0,
max = 1.0



Clamped
min=-1.2,
max = 1.2



Vanilla

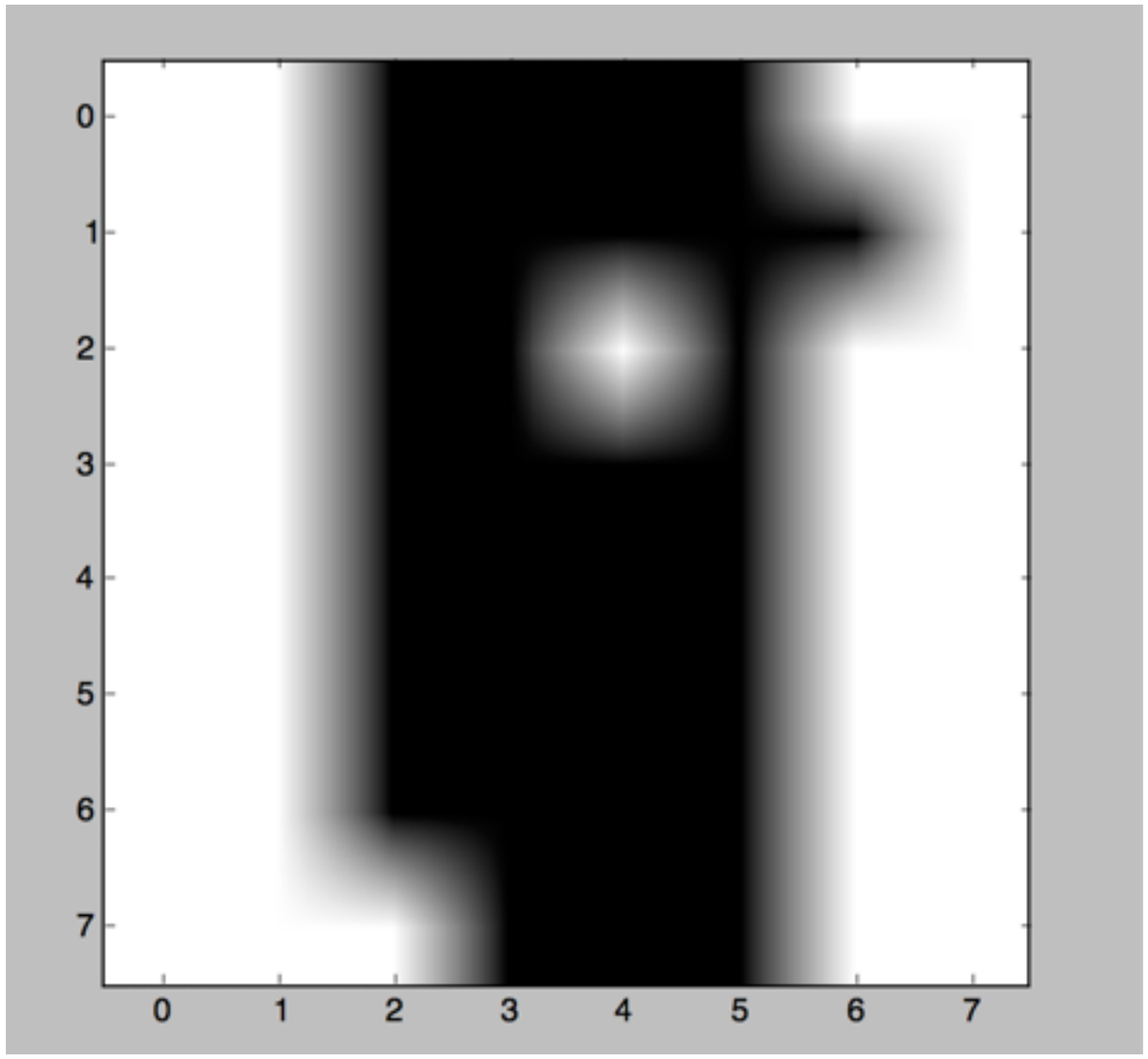
What experiment 3 suggest ?

1. Gradient saturation not normalisation is responsible for the improvement

2. Gradient saturation results in clear images, but it is not responsible for learning the image structure.

3. -1 to 1 is an arbitrary range

Experiment 3 extended



Mode collapse !

Clamped
min=-0.3,
max = 0.3

Thank you

Email : angmingliang4017ic@gmail.com
Facebook: Ang Ming Liang