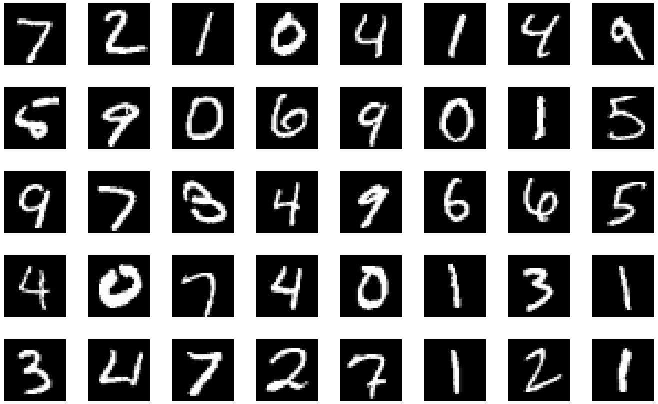
Report of Digital Recognizer

XiaoLong Luo

Useful Methods & Interesting Models

Update: December 16, 2020



Contents

1 Basic Introduction

1.1 Discription of Dataset:

- Training set: 42000 pic, stored in train.csv
- 28*28 pixels per pic, range from (0,255)
  – Thus we need to normalize them first.
- each pic has a label range from 0 to 9
- csv file, use pandas to load them

- I know this is the third time you here them...

1.2 Challenges based on CNN

Basic Steps of ML problems:

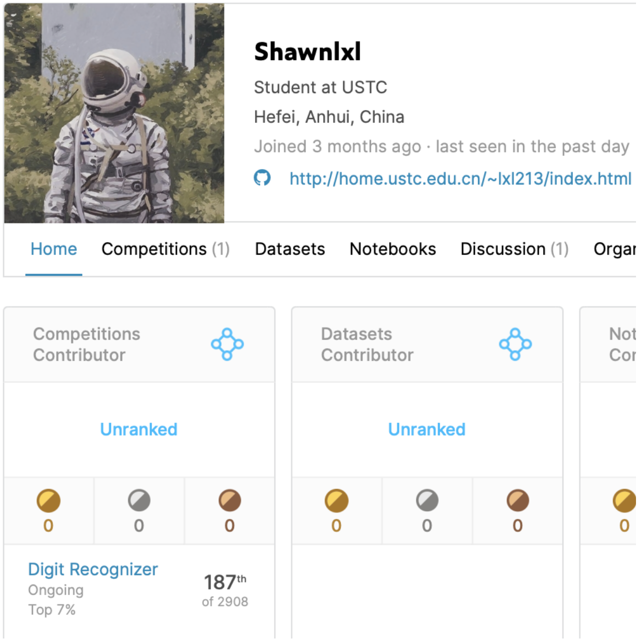
1. Pre-process of the data
2. Choose the right models, appropriate loss function, optimization algorithm. etc.
3. Training process, evaluation and optimization

Challenges:

1. Balance the model complication and the overfitting problems of Neural networks
2. Long Training time. Always combine with the first problem.
3. Compare with test set (28000), the training set is roughly small. – **Want more data.**

2 My Contribution

2.1 Rank and Code

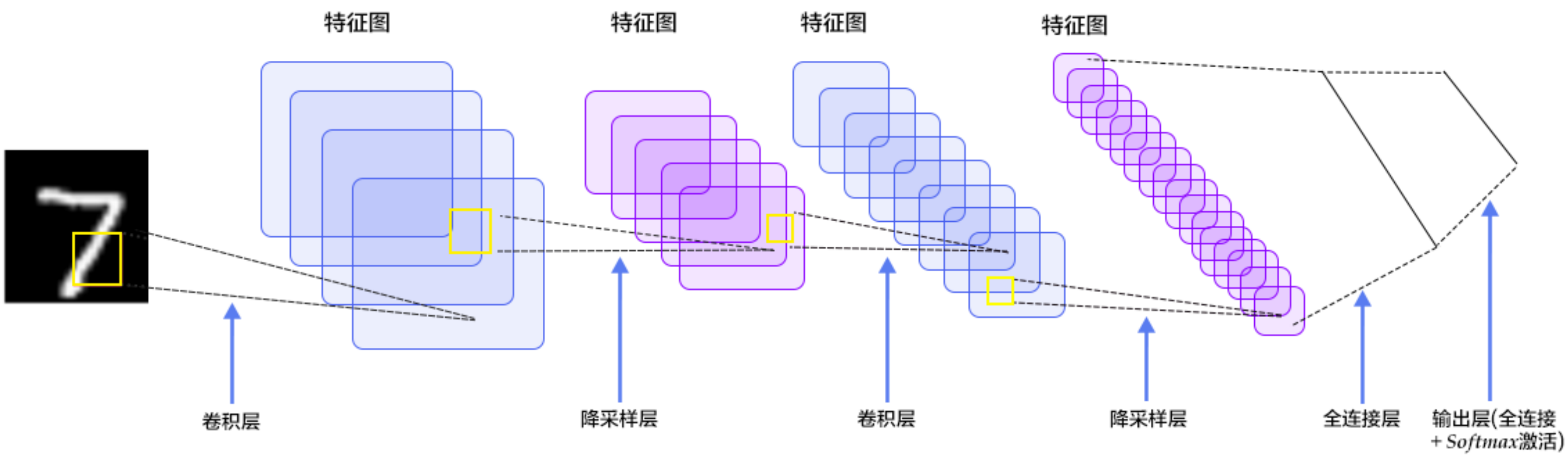


Code[1]

[1]The code is posted on:https://github.com/lxl213

2.2 More Complicated Models

Initial one:



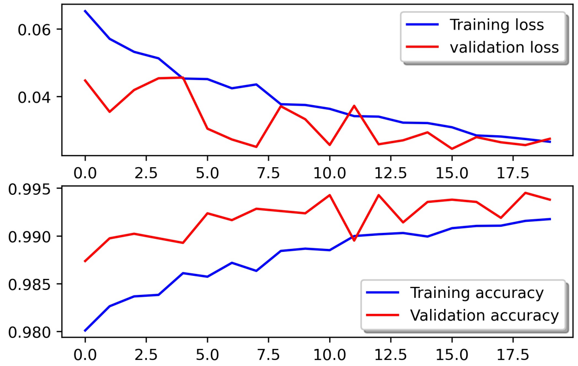
Final one: Too long to be listed here...

- 7 Conv2d layer, 7 BatchNormalization layer,
- 3 Dropout layer, 2 FC layer for prediction

2.3 Train test split and Multi-Models

Idea:

1. Multi-Models to "Vote " for a prediction – Useful skills in many models
2. We randomly split up the Training set to make a validation set(8:2).
   — (why not cross-validation?)
3. For the kaggle competition, we set a certain validation accuracy level, if the model's performance surpass it, we stop train and save the model .
   — Compare with a fixed epoch, can help to save a lot time.



2.4 Data augmentation

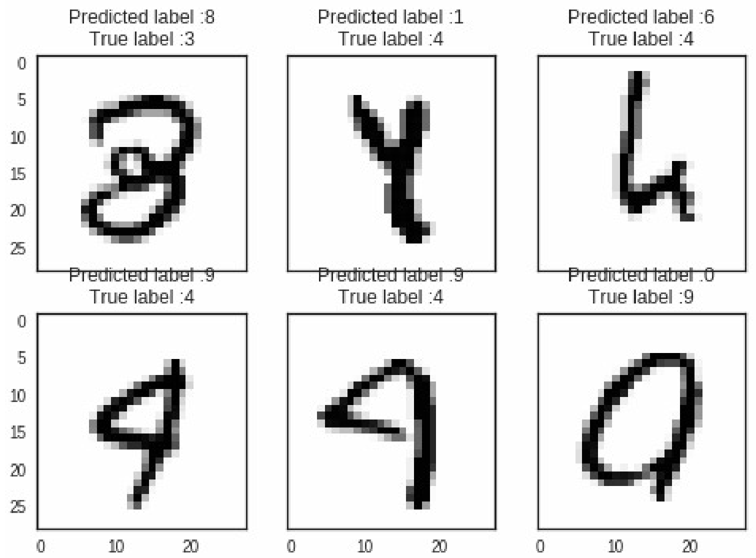
Idea:

1. From the training set:

   such as rotation, flip, width- shift and Gauss noise etc.

2. Generate **more data!**
3. A basic datagen by keras:

```
datagen = ImageDataGenerator(
    featurewise_center=True,
    featurewise_std_normalization=True,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True)
```



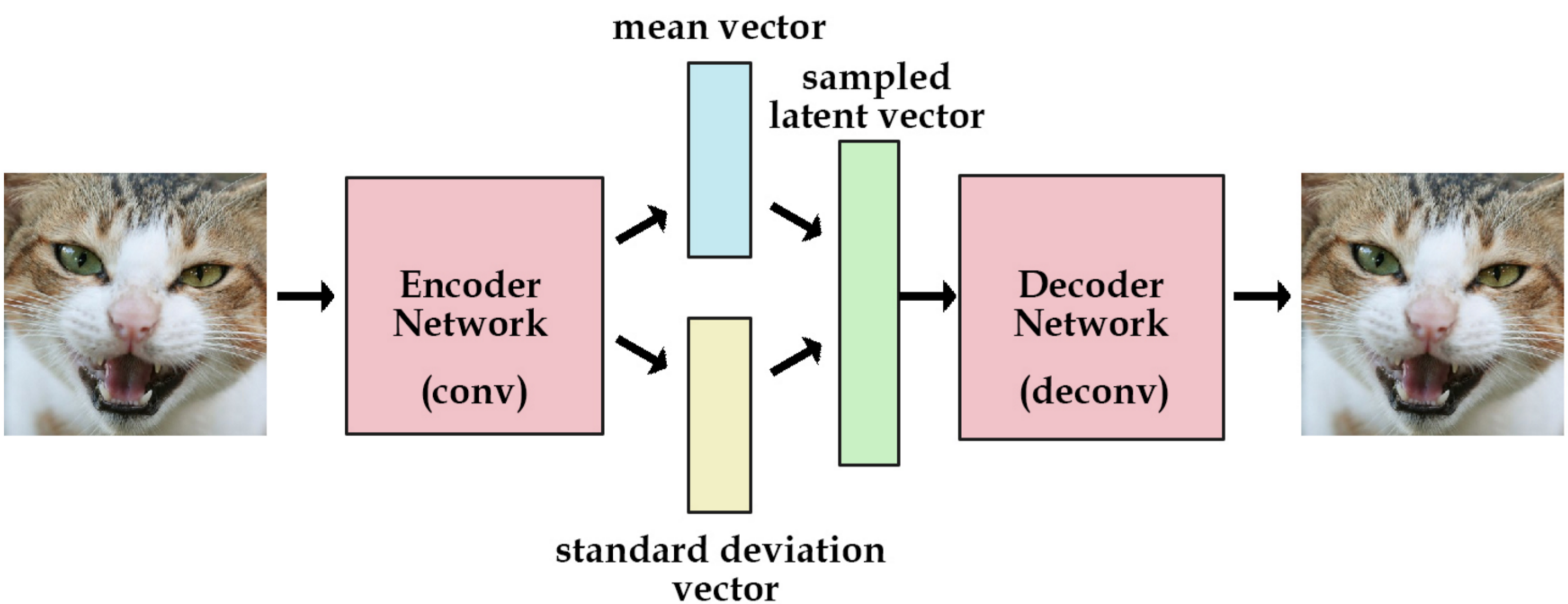
2.5 Other methods:

- Dropout
- Tuning skills: LR, Batch-size, Epoch ,etc.
- CNN‘s structure..

3 Interest Models

3.1 Variational Auto-Encoder(VAE)

What it is?



How it works?

Implementation:

```
VAE(
  (conv1): Sequential(
    (0): Conv2d(1, 16, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (conv2): Sequential(
    (0): Conv2d(16, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (conv3): Sequential(
    (0): Conv2d(32, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (fc_encode1): Linear(in_features=784, out_features=10, bias=True)
  (fc_encode2): Linear(in_features=784, out_features=10, bias=True)
  (fc_decode): Linear(in_features=10, out_features=784, bias=True)
  (deconv1): Sequential(
    (0): ConvTranspose2d(16, 16, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
  )
  (deconv2): Sequential(
    (0): ConvTranspose2d(16, 1, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): Sigmoid()
  )
)
```

Key of this method:

$$\log p_\theta\left(x^{(i)}\right) = \mathbf{E}_{z\sim q_\phi\left(z|x^{(i)}\right)}\left[\log p_\theta\left(x^{(i)}\right)\right] \qquad \left(p_\theta\left(x^{(i)}\right) \text{ Does not depend on } z\right)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)} \mid z\right) p_\theta(z)}{p_\theta\left(z \mid x^{(i)}\right)}\right] \qquad (\text{ Bayes' Rule })$$

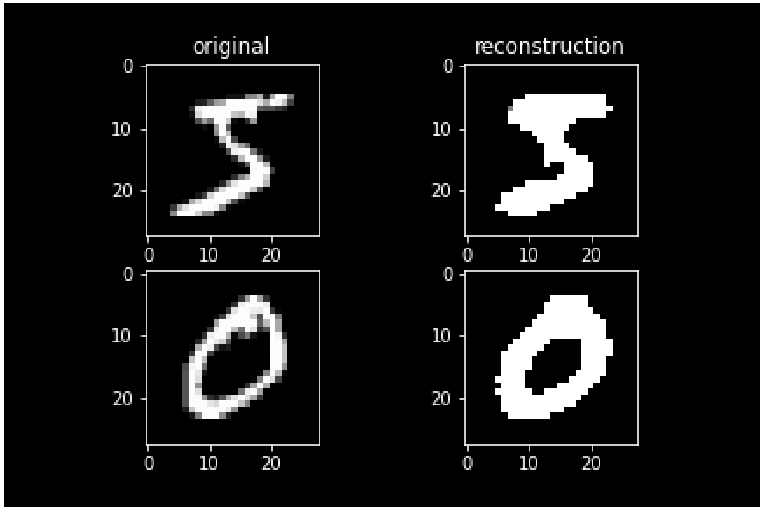
$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)} \mid z\right) p_\theta(z)}{p_\theta\left(z \mid x^{(i)}\right)} \frac{q_\phi\left(z \mid x^{(i)}\right)}{q_\phi\left(z \mid x^{(i)}\right)}\right] \qquad (\text{ Multiply by constant })$$

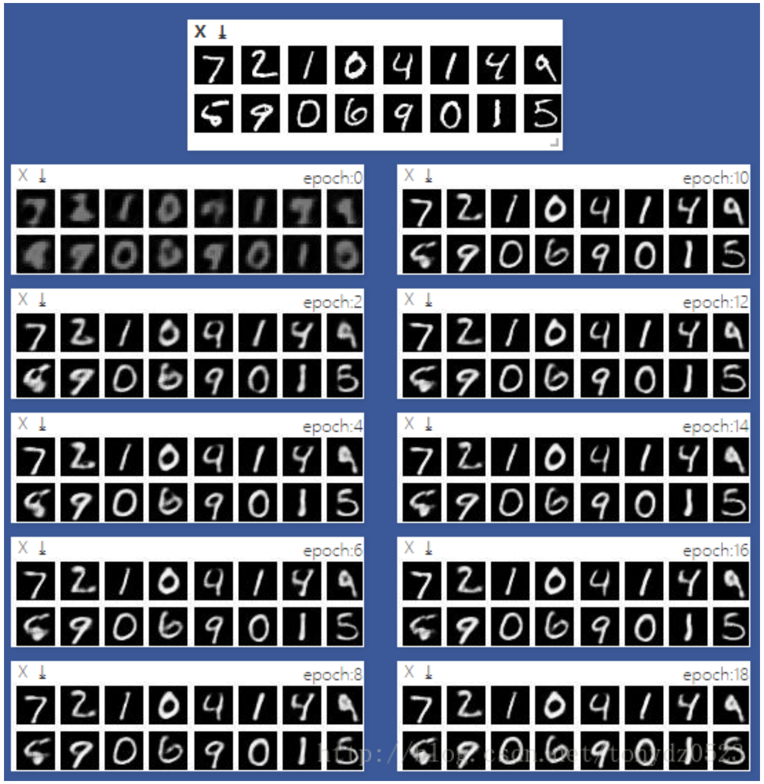
$$= \mathbf{E}_z\left[\log p_\theta\left(x^{(i)} \mid z\right)\right] - \mathbf{E}_z\left[\log \frac{q_\phi\left(z \mid x^{(i)}\right)}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi\left(z \mid x^{(i)}\right)}{p_\theta\left(z \mid x^{(i)}\right)}\right]$$

$$= \mathbf{E}_z\left[\log p_\theta\left(x^{(i)} \mid z\right)\right] - D_{KL}\left(q_\phi\left(z \mid x^{(i)}\right) \| p_\theta(z)\right) +$$

$$D_{KL}\left(q_\phi\left(z \mid x^{(i)}\right) \| p_\theta\left(z \mid x^{(i)}\right)\right)$$

Performance: Train process and reconstruction



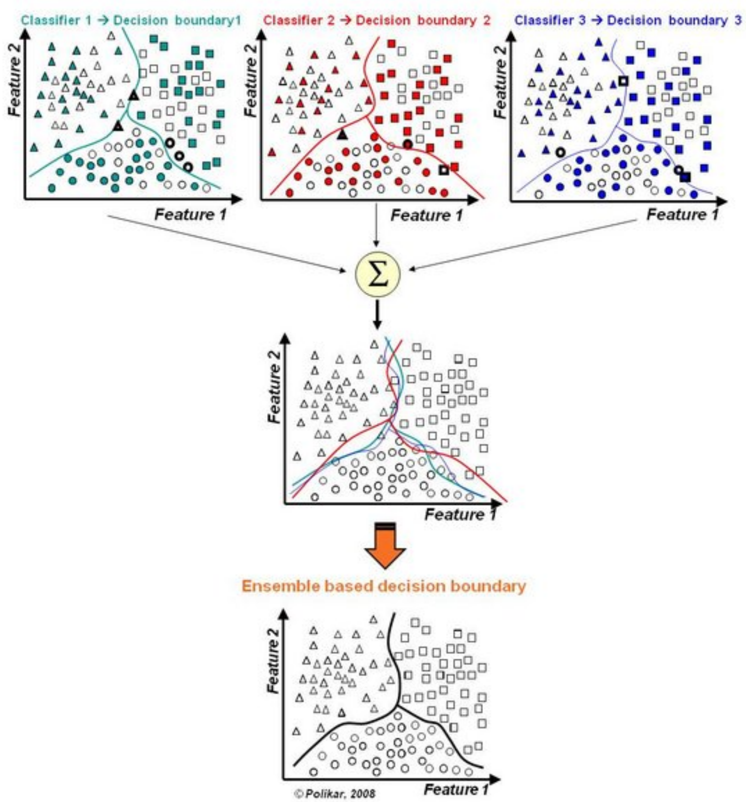


Remark The right plot is generated by module: visdom.[2]

[2]See:https://www.cnblogs.com/fanghao/p/10256287.html

3.2 Random Forest

A easy Implemented model with short training time




```python
rawData=pd.read_csv(trainPath).values
trainData=rawData[:,1:]
trainLabel=rawData[:,0]
testData= pd.read_csv(testPath).values
X=trainData
Y=trainLabel

clf=RandomForestClassifier(n_estimators=100)
clf=clf.fit(X,Y)
testLabel=clf.predict(testData)

#训练结果保存
df=pd.DataFrame(testLabel,columns=['label'])
df.to_csv('testLabel.csv',header=True,index=False)
```

executed in 25.2s, finished 23:26:33 2020–12–16

4 Futher Discussion

1. The choice of loss(cross-) and can we pose some penalty in this case?
2. The explanation theory of the VAE, to help us identify the latent variable —Disentangled Variational Auto-Encoder[3]
3. GAN and DCGAN(Deep Convolutional Generative Adversarial Networks)[4] can achieve a higher performance than VAE in some cases. What about in MNIST?
4. Other Dataset.

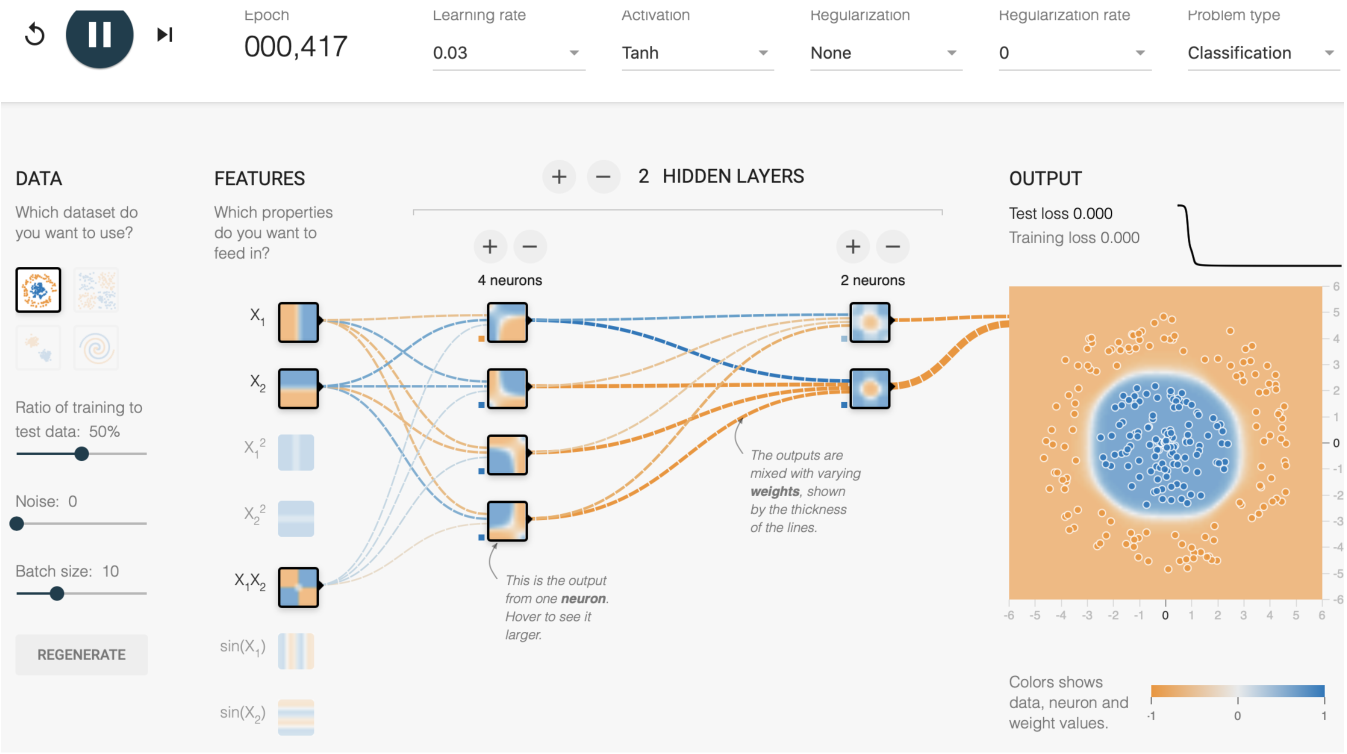
---

[3]Disentangled Variational Auto-Encoder for semi-supervised learning,2018

[4]Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,2016

4.1 Interesting Stuff

A Playground for CNN[5]



[5]See And Try! http://playground.tensorflow.org