Final Project Demo

Group1:陳維仁、曾靖渝、袁維澤

Outline

- Implementation of classifier

- Implementation of GAN

Implementation of classifier

Methodology & Evaluation

. Basically based on sample code

. Modify some parameters to improve performance

. We focus on what we’ve tried to increase accuracy

Methodology & Evaluation

First try:

increase the number of hidden layers

Result:

Little improvement(still not good enough , with accuracy 0.7~0.8)

Methodology & Evaluation

Second try:

increase training epochs from 20 to 25 , and increase the initial learning rate of Adam optimizer(to speed up)

Result:

At around 22nd training epoch , early stopping terminates the training process

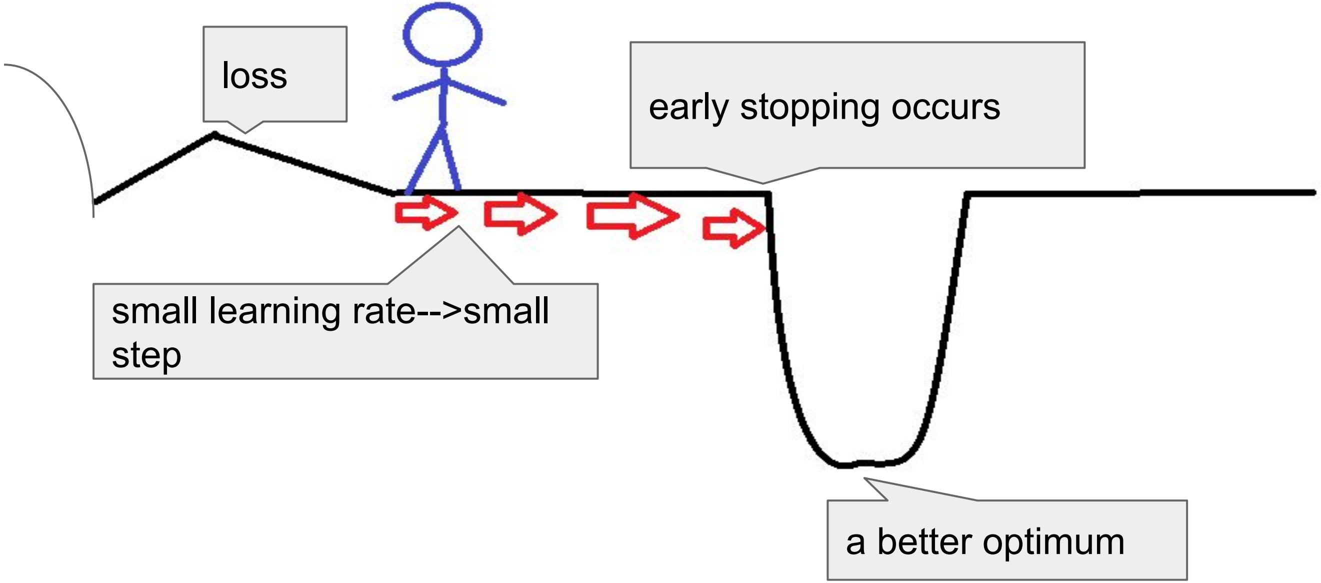
Remove Early stopping

Purpose of Early stopping:

avoid overfitting

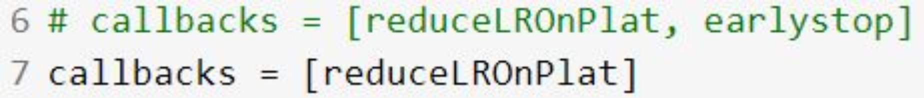
But what if …...

Remove Early stopping

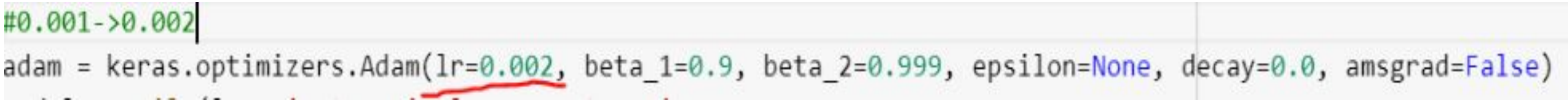


Remove Early stopping

After removing the Early stopping , and increasing the initial learning rate of Adam optimizer(we change the default setting)……



```
# callbacks = [reduceLROnPlat, earlystop]
callbacks = [reduceLROnPlat]
```



```
#0.001->0.002
adam = keras.optimizers.Adam(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
```

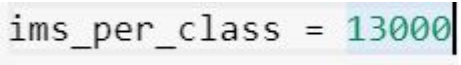
Result:

With accuracy 0.79~0.85

Can the classifier be better?

Yes!

The total number of images per class is over 60000 , we only take 2000!

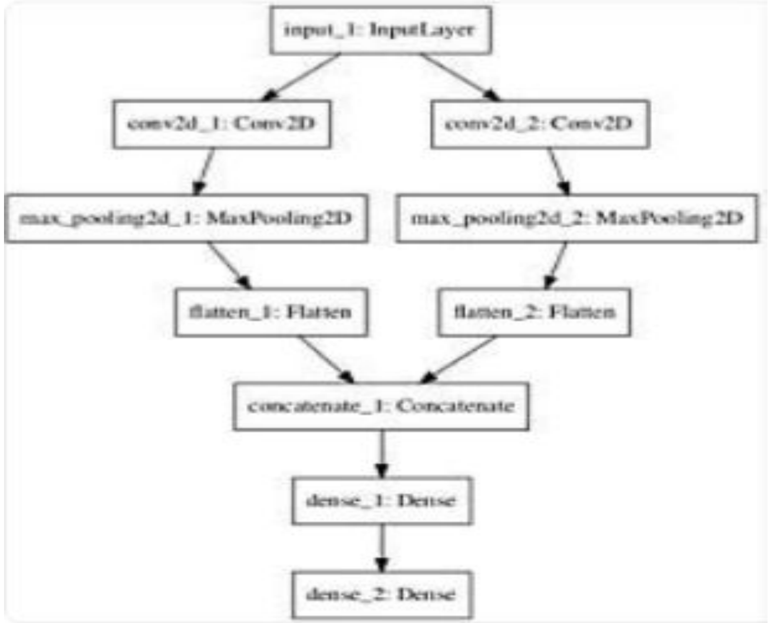


```
ims_per_class = 13000
```

Result:

With accuracy 0.95~0.97

Can the classifier be better?



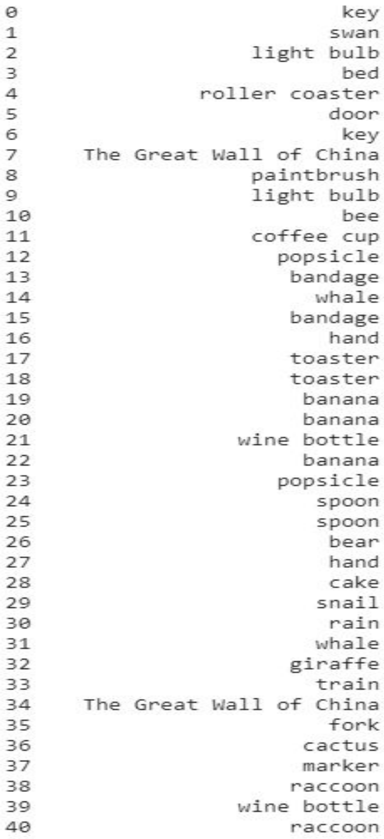
change the sequential model to non-linear mode

Result:

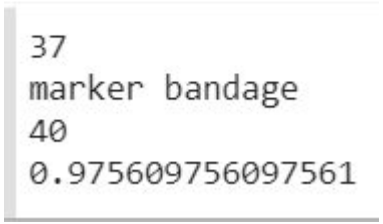
With accuracy only 0.56~0.6

-->We use the classifier of previous page as our final classifier

Test result



```
0                     key
1                    swan
2              light bulb
3                     bed
4          roller coaster
5                    door
6                     key
7   The Great Wall of China
8              paintbrush
9              light bulb
10                    bee
11             coffee cup
12               popsicle
13                bandage
14                  whale
15                bandage
16                   hand
17                toaster
18                toaster
19                 banana
20                 banana
21            wine bottle
22                 banana
23               popsicle
24                  spoon
25                  spoon
26                   bear
27                   hand
28                   cake
29                  snail
30                   rain
31                  whale
32                giraffe
33                  train
34  The Great Wall of China
35                   fork
36                 cactus
37                 marker
38                raccoon
39            wine bottle
40                raccoon
```



```
37
marker bandage
40
0.975609756097561
```

Demo result

Accuracy : 36/40

Implementation of GAN

Methodology

. Use DCGAN , one GAN for each class label

. Using some tips to better the performance of DCGAN

Methodology

1. Use Strided Convolutions

It is common to use pooling layers such as max-pooling layers for downsampling in convolutional neural networks.

In GANs, the recommendation is not to use pooling layers, and instead use the stride in convolutional layers to perform downsampling in the discriminator model.

Methodology

2. Remove Fully-Connected Layers

It is common to use fully-connected layers after feature extraction layers in convolutional layers as an interpretation of the extracted features prior to the output layers of the model.

Methodology

2. Remove Fully-Connected Layers

Instead, in GANs, fully-connected layers are not used, in the discriminator and the convolutional layers are flattened and passed directly to the output layer.

Methodology

3. Use Batch Normalization

Batch normalization standardizes the activations from a prior layer to have a zero mean and unit variance. This has the effect of stabilizing the training process.

Batch normalization has become a staple when training deep convolutional neural networks, and GANs are no different.

Methodology

4. Use ReLU, Leaky ReLU, and Tanh

Activation functions such as RELU are used to address the vanishing gradient problem in deep convolutional neural networks and promote sparse activations (e.g. lots of zero values).

Methodology

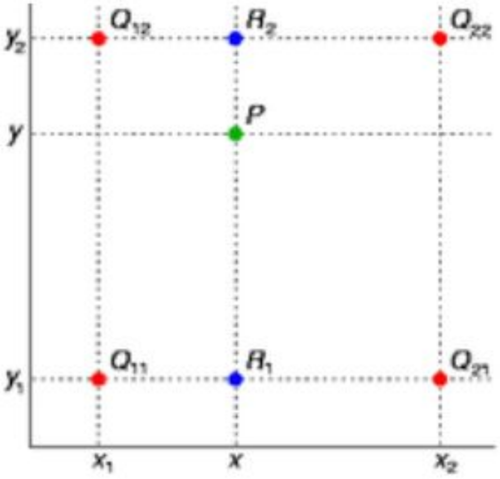
4. Use ReLU, Leaky ReLU, and Tanh

ReLU is recommended for the generator, but not for the discriminator model. Instead, a variation of ReLU that allows values less than zero, called Leaky ReLU, is preferred in the discriminator.

Image.BILINEAR

Can use the red points to get the green points

-->increase the resolution of the image



Methodology

Increase/Decrease the training data size based on class label

- 100000 for some labels

-200000 for others

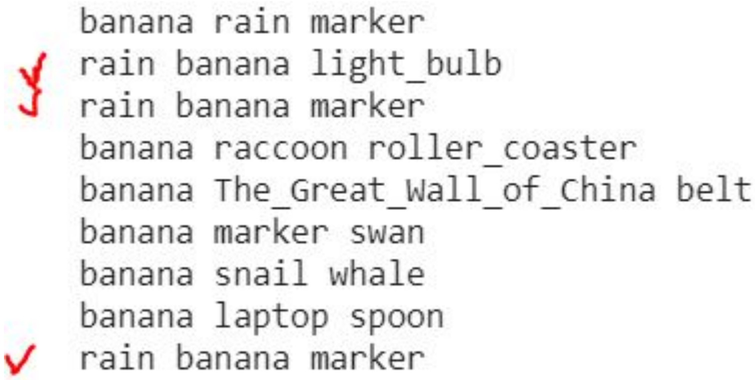
-The reason will be given in the Discussion & Conclusion

Evaluation & Test result

-During training process of GAN , we save the generator every 1 ~ 3 epochs , and feed the generator into the classifier to perceive how many epochs the GAN has the best performance.

Evaluation & Test result

-Our classifier will list three most likely labels , we will use this to determine whether further training does better.

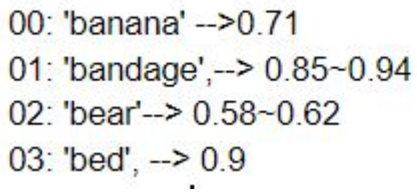


```
  banana rain marker
✓ rain banana light_bulb
✓ rain banana marker
  banana raccoon roller_coaster
  banana The_Great_Wall_of_China belt
  banana marker swan
  banana snail whale
  banana laptop spoon
✓ rain banana marker
```

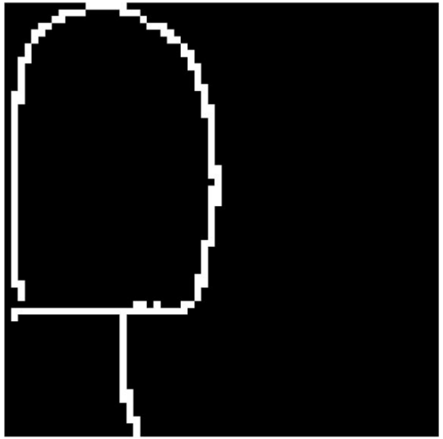
-Red pen represents further training may be better

Evaluation & Test result

Accuracy returned by oue classifier



```
00: 'banana' -->0.71
01: 'bandage',--> 0.85~0.94
02: 'bear'--> 0.58~0.62
03: 'bed', --> 0.9
```



Demo result

```
Accuracy : 100/100 (0 subjective)
```

Discussion & Conclusion

-Easy image label(e.g. hand) → Every doodle drawing game players can draw perfectly(high quality data)

-->increase data size to 200000

-Difficult label(e.g. raccoon) →Many low quality drawings

-->decrease data size to 100000