CS231n学习笔记

Assignment Git:

<https://github.com/CS231n-zju/CS231n>

视频地址：

[https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv\](https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv/)

课程作业：

<http://cs231n.github.io/>

Syllabus:

<http://cs231n.stanford.edu/syllabus.html>

Note翻译:

<http://www.52ml.net/17723.html>

**Week5**

[Understanding and Visualizing Convolutional Neural Networks 2](#_Toc353636346)

[Transfer Learning and Fine-tuning Convolutional Neural Networks 3](#_Toc193922701)

[Assignment3\_Q1: Image Captioning with Vanilla RNNs 3](#_Toc1712713083)

[1.参考材料 3](#_Toc1978126479)

[2. RNN BP传播的原理 3](#_Toc2028370660)

[3.Word embedding(Forward和BP) 3](#_Toc275096393)

[4.sigmoid和softmax的区别 3](#_Toc886419643)

[5. rnn训练过程 4](#_Toc594336038)

[6.rnn预测过程 5](#_Toc618817743)

[Assignment3\_Q2: Image Captioning with LSTMs 5](#_Toc155156198)

[1.step forward 5](#_Toc1010704239)

[2.lstm\_backward(dh, cache) 6](#_Toc1573230538)

[3.Extra Credit: Train a good captioning model! 6](#_Toc176478778)

[3.1一种机器翻译的评价准则——Bleu 6](#_Toc1165125998)

[3.2用前面的numpy版本实现 6](#_Toc1057956651)

[3.3用tensorflow实现(多层LSTM) 6](#_Toc135091250)

[Assignment3\_Q3: Network Visualization: Saliency maps, Class Visualization, and Fooling Images 7](#_Toc528735365)

[1.一些问题记录 7](#_Toc2123096032)

[2.Saliency Maps 8](#_Toc316932780)

[3.Fooling Images 8](#_Toc696111011)

[4.Class visualization 8](#_Toc1814320148)

[Assignment3\_Q4: Style Transfer 9](#_Toc419039943)

[1. 出现的问题记录 9](#_Toc720403163)

[2.loss定义 9](#_Toc1330549119)

[2.1 content loss 9](#_Toc195278128)

[2.2 style loss 9](#_Toc1393584249)

[2.3 Total-variation regularization 10](#_Toc248003113)

[2.4 Feature Inversion 10](#_Toc1780810552)

[Assignment3\_Q5: Generative Adversarial Networks 10](#_Toc2062368425)

[1.原理 10](#_Toc1129528004)

[2.GAN的训练tricks 11](#_Toc592994879)

[3.其他的生成模型 11](#_Toc268521123)

[4.实现GAN 11](#_Toc1323450705)

[4.1 LeakyReLU 11](#_Toc158224315)

[4.2 实现discriminator 11](#_Toc99163954)

[4.3 实现generator 11](#_Toc1204337717)

[4.4 ※GAN loss 11](#_Toc433320708)

[4.5 Optimizing our loss 12](#_Toc985583597)

[5.Least Squares GAN 12](#_Toc1798673756)

[6.Deep Convolutional GANs 12](#_Toc1052138452)

[7.※WGAN-GP 12](#_Toc1140739796)

# Understanding and Visualizing Convolutional Neural Networks

激活函数层可视化时可以发现的问题:

some activation maps may be all zero for many different inputs, which can indicate dead filters, and can be a symptom of high learning rates.

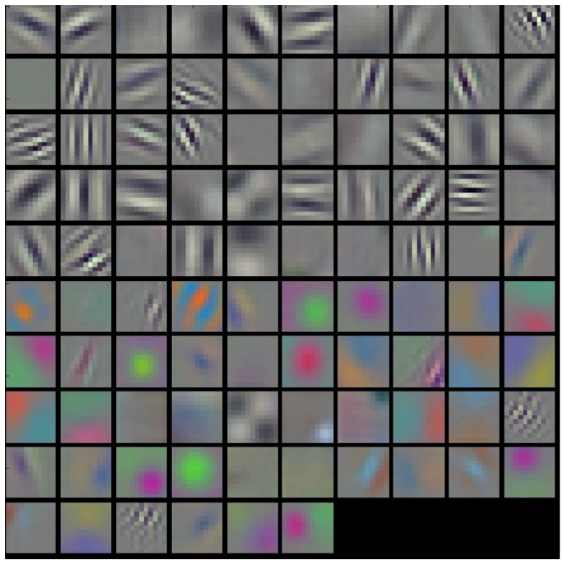
权重层可视化时可以发现的问题:

越平滑训练效果越好.

Noisy patterns can be an indicator of a network that hasn’t been trained for long enough, or possibly a very low regularization strength that may have led to overfitting.

输出AlexNet第一层:

发现灰度特征和彩色特征明显有聚类现象,主要是因为AlexNet有两个流去处理他们.



The color/grayscale features are clustered because the AlexNet contains two separate streams of processing, and an apparent consequence of this architecture is that one stream develops high-frequency grayscale features and the other low-frequency color features.

介绍把图像从高维空间映射到低维空间,但依旧保留图像像素点两两之间的距离的方法:t-sne,同时,经过t-sne之后,语义相近的图像会挨得比较近.

如何知道,识别图像中的狗的时候,我们得到的结果是来自于这只狗还是背景中的一些可能存在与狗的语义相关的信息,方法就是滑窗遮挡图像中的一部分,看概率会不会变化.

# Transfer Learning and Fine-tuning Convolutional Neural Networks

# Assignment3\_Q1: Image Captioning with Vanilla RNNs

## 1.参考材料

<http://cs224d.stanford.edu/>

## RNN BP传播的原理

dprev\_h=np.zeros((N,H))

for ti in range(T-1,-1,-1):

dx[:,ti,:], dprev\_h, dWx\_tmp, dWh\_tmp, db\_tmp=rnn\_step\_backward(

dh[:,ti,:]+dprev\_h, cache[ti])

dWx+=dWx\_tmp

dWh+=dWh\_tmp

db+=db\_tmp

dh0=dprev\_h

## 3.Word embedding(Forward和BP)

- x: Integer array of shape (N, T) giving indices of words. Each element idx

of x muxt be in the range 0 <= idx < V.

- W: Weight matrix of shape (V, D) giving word vectors for all words.

- out: Array of shape (N, T, D) giving word vectors for all input words.

前向:

out=W[x[:,]]

cache=(x,W)

后向:

np.add.at(dW,x[:],dout[:])

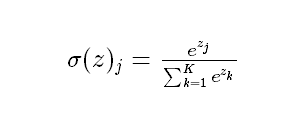
numpy.add.at用法:

<https://docs.scipy.org/doc/numpy/reference/generated/numpy.ufunc.at.html#numpy.ufunc.at>

## 4.sigmoid和softmax的区别

都在[0,1]之间

Sigmoid:2018-01-14 11-26-08屏幕截图

Softmax:

总结：

(1)sigmoid将一个real value映射到（0,1）的区间（当然也可以是（-1,1）），这样可以用来做二分类。

(2)softmax把一个k维的real value向量（a1,a2,a3,a4….）映射成一个（b1,b2,b3,b4….）其中bi是一个0-1的常数，然后可以根据bi的大小来进行多分类的任务，如取权重最大的一维。

## rnn训练过程

At training time, we have access to the ground-truth caption, so we feed ground-truth words as input to the RNN at each timestep.

在训练过程中,每次输入GT和上一时刻的输出,学习的是上下文关系,因此每次的输入就有GT,希望预测出的序列和目标序列接近(用loss衡量)

这里captions是GT

captions\_in = captions[:, :-1]

captions\_out = captions[:, 1:]

In the forward pass you will need to do the following:

(1) Use an affine transformation to compute the initial hidden state

from the image features. This should produce an array of shape (N, H)

(2) Use a word embedding layer to transform the words in captions\_in

from indices to vectors, giving an array of shape (N, T, W).

(3) Use either a vanilla RNN or LSTM (depending on self.cell\_type) to

process the sequence of input word vectors and produce hidden state

vectors for all timesteps, producing an array of shape (N, T, H).

(4) Use a (temporal) affine transformation to compute scores over the

vocabulary at every timestep using the hidden states, giving an

array of shape (N, T, V).

(5) Use (temporal) softmax to compute loss using captions\_out, ignoring

the points where the output word is <NULL> using the mask above.

①首层的输入hidden其实是图片特征\*W\_proj+b\_proj

h0=np.dot(features,W\_proj)+b\_proj

②把前N-1个GT做word embedding

③做前向rnn,生成的是(N, T, H)的word串,即预测值

④计算score(N, T, V)和loss

注意temporal\_softmax\_loss()里mask的作用,mask跳过所有对计算loss没有贡献的字符,比如<NULL>这样的标记

※注意到这里需要学习的几个参数:

①W\_proj和b\_proj:用来把图片转成第一层隐含层

②W\_embed(vocab\_size单词个数, wordvec\_dim每个单词向量的长度):把单词转成向量的矩阵

③Wx,Wh,b:注意到权值共享,处理当前单词,上一层,以及偏置

④W\_vocab(hidden\_dim, vocab\_size),b\_vocab:计算score的矩阵

## 6.rnn预测过程

我们会使用rnn\_step\_forward而不是rnn\_forward

①对前一个单词用训练好的embedding矩阵做embedding,其中第一个是<start>

②用①中处理好的词向量和前一个状态,去预测下一个状态,其中第一个h0是图像

③用②中的结果,用我们学好的W\_vocab和b\_vocab去得到现在整个词库里所有单词的得分,取最高的作为我们预测的这个时刻的单词

You will need to initialize the hidden state of the RNN by applying the learned affine transform to the input image features. The first word that you feed to

the RNN should be the <START> token; its value is stored in the variable self.\_start. At each timestep you will need to do to:

(1) Embed the previous word using the learned word embeddings

(2) Make an RNN step using the previous hidden state and the embedded

current word to get the next hidden state.

(3) Apply the learned affine transformation to the next hidden state to

get scores for all words in the vocabulary

(4) Select the word with the highest score as the next word, writing it

to the appropriate slot in the captions variable

For simplicity, you do not need to stop generating after an <END> token

is sampled, but you can if you want to.

HINT: You will not be able to use the rnn\_forward or lstm\_forward

functions; you'll need to call rnn\_step\_forward or lstm\_step\_forward in

a loop.

# Assignment3\_Q2: Image Captioning with LSTMs

## 1.step forward

- x: Input data, of shape (N, D)

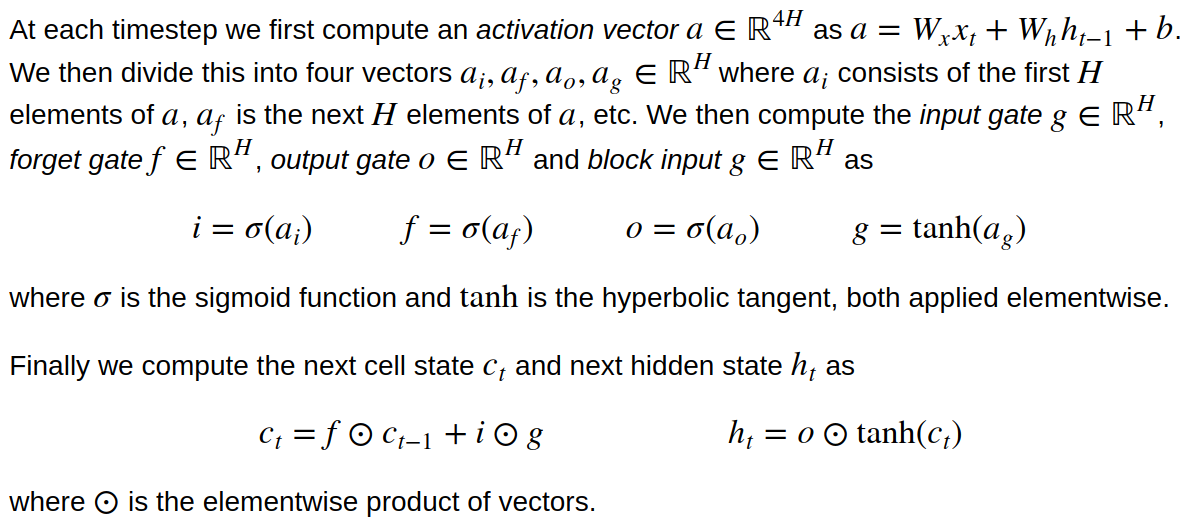
- prev\_h: Previous hidden state, of shape (N, H)

- prev\_c: previous cell state, of shape (N, H)

- Wx: Input-to-hidden weights, of shape (D, 4H)

- Wh: Hidden-to-hidden weights, of shape (H, 4H)

- b: Biases, of shape (4H,)



①a=np.dot(x,Wx)+np.dot(prev\_h,Wh)+b

②这时把a分成4份,ai,af,ao,ag

③分别用sigmoid和tanh激励函数得到:i,f,o,g

④2018-01-14 19-05-24屏幕截图

计算2个量:next\_h, next\_c

## 2.lstm\_backward(dh, cache)

为什么初始化:

dnext\_c=np.zeros((N,H))

for ti in range(T-1,-1,-1):

dx[:,ti,:], dprev\_h, dprev\_c, dWx\_tmp, dWh\_tmp, db\_tmp=

lstm\_step\_backward(dh[:,ti,:]+dprev\_h, dnext\_c, cache[ti])

dnext\_c= dprev\_c

dWx+=dWx\_tmp

dWh+=dWh\_tmp

db+=db\_tmp

## 3.Extra Credit: Train a good captioning model!

### 3.1一种机器翻译的评价准则——Bleu

### 3.2用前面的numpy版本实现

### 3.3用tensorflow实现(多层LSTM)

需要做几件事:

①定义跑模型并且阶段性输出结果的函数:

def run\_model(session, predict, loss\_val, Xd, yd,

epochs=1, batch\_size=64, print\_every=100,

training=None, plot\_losses=False):

②多层lstm的实现

③可以注意一下rnn里怎么做dropout

# Assignment3\_Q3: Network Visualization: Saliency maps, Class Visualization, and Fooling Images

GAN的预备操作:

In this notebook we will explore the use of image gradients for generating new images.

这一次,用整个CNN去定义一个loss,用bp去求解dx!然后合成图片去最小化这个loss.

这个part会探索三个问题:

* Saliency Maps(显著图): Saliency maps are a quick way to tell which part of the image influenced the classification decision made by the network.
* Fooling Images: We can perturb an input image so that it appears the same to humans, but will be misclassified by the pretrained network.一些生成的图片,会被分类器误分类.
* Class Visualization: We can synthesize an image to maximize the classification score of a particular class; this can give us some sense of what the network is looking for when it classifies images of that class. 生成一个图片去最大化某一类图的分类score,这会give us some sense of网络分类时所关注的关键点.

## 1.一些问题记录

(1)Squeezenet.ckpt找不到

解决方法:

<https://www.reddit.com/r/cs231n/comments/6fhz3z/assignment_3_squeezenet/?st=jcfq1gkp&sh=55497490>

|  |
| --- |
| cp squeezenet.ckpt.data-00000-of-00001 squeezenet.ckpt |

直接把squeezenet.ckpt.data-00000-of-00001复制一个squeezenet.ckpt

Tensorflow的模型的三个文件:

* model.ckpt.meta
* model.ckpt.index
* model.ckpt.data-00000-of-00001

其中:

Meta保存图结构,data保存里面所有的变量的具体值

具体解释:

①meta file: describes the saved graph structure, includes GraphDef, SaverDef, and so on; then apply tf.train.import\_meta\_graph('/tmp/model.ckpt.meta'), will restore Saver and Graph.

使用protocol buffer来保存整个tensorflow graph.例如所有的variables, operations, collections等等。这个文件使用.meta后缀

Checkpoint二进制文件2个,二进制文件包含所有的weights,biases,gradients和其他variables的值:

②index file: it is a string-string immutable table(tensorflow::table::Table). Each key is a name of a tensor and its value is a serialized BundleEntryProto. Each BundleEntryProto describes the metadata of a tensor: which of the "data" files contains the content of a tensor, the offset into that file, checksum, some auxiliary data, etc.

③data file: it is TensorBundle collection, save the values of all variables.

## 2.Saliency Maps

计算图片对分类结果的gradient : 图片产生一点点修改的时候,classification的结果会变化多少.

先求loss对于图像的梯度,再求绝对值,再求3个通道上的最大值.

we take the absolute value of this gradient, then take the maximum value over the 3 input channels

## 3.Fooling Images

沿着某一类的score的gradient上升的方向创造一张图片

直到:分类器认为这个图就是这一类

这张图长得会非常非常像猫,但是却会利用分类器的梯度漏洞,被分类成某一个特定的类别target\_y,比如狗,比如船

代码:

|  |
| --- |
| iters\_num=50    g=tf.gradients(model.classifier[0,target\_y],model.image)  dX = learning\_rate \* g / tf.norm(g)  for ti in range(iters\_num):  feed\_dict= {model.image:X\_fooling}  np\_dx=sess.run(dX,feed\_dict)[0]  X\_fooling+=np\_dx |

## 4.Class visualization

优化问题:

2018-01-15 22-39-38屏幕截图

2018-01-16 11-20-10屏幕截图

Graph设置:

|  |
| --- |
| loss = model.classifier[0,target\_y]-0.5\*l2\_reg\*(model.image\*model.image)  grad = tf.gradients(loss,model.image)[0]  dX=learning\_rate\*grad/tf.norm(grad) |

迭代内:

|  |
| --- |
| feed\_dict={model.image:X}  np\_dx=sess.run(dX,feed\_dict)  X+=np\_dx |

# Assignment3\_Q4: Style Transfer

用两个图去生成一个图,新图包含其中一个的content和另一个的style,定义loss function,在x上梯度下降

## 出现的问题记录

(1)找不到scipy

因为是用miniconda安装的py3.6,所以包管理工具不是pip而是conda,如果check\_scipy()不通过,应当:

|  |
| --- |
| conda install scipy |

## 2.loss定义

content loss + style loss + total variation loss

### 2.1 content loss

Fij - Feature map of current image

Pij - Feature map of source image

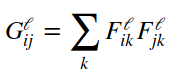
2018-01-16 12-45-58屏幕截图

### 2.2 style loss

We want the activation statistics of our generated image to match the activation statistics of our style image, and matching the (approximate) covariance is one way to do that.

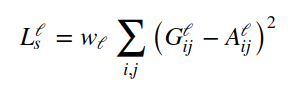
这里用gram matrix表示covariance

①计算gram matrix,假设一张图有c个filters,那么gram matrix应当是(C,C)的,其中每一个元素的公式是:



代码实现时注意:先做reshape再transpose

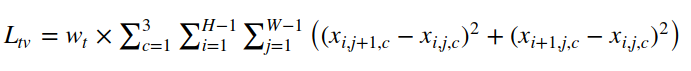
②计算current image和source image(for syle)的gram matrix 偏差,w\_l是标量,其中单层(一个filter)的loss是



③对所有filters求和

### 2.3 Total-variation regularization

最后一个loss,每个pixel和它右下相邻的两个pixel的偏差惩罚



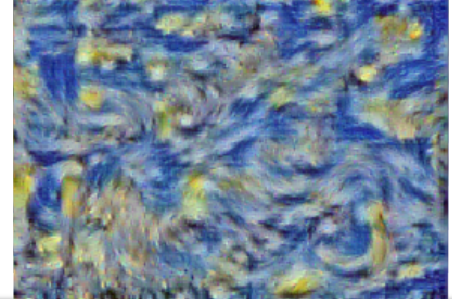
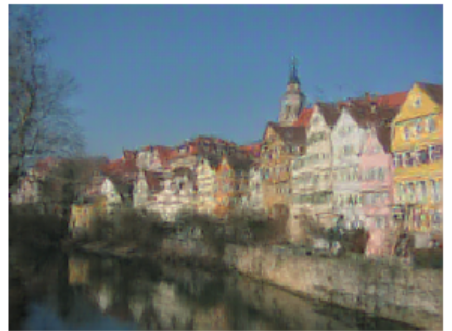
### 2.4 Feature Inversion

不考虑同时生成图和纹理,我们只考虑其中之一

通过初始化style weight为0,content weight为随机数,来从0开始生成整张图.

通过初始化Content weight为0,style weight为随机数,来模拟纹理.





# Assignment3\_Q5: Generative Adversarial Networks

## 1.原理

GAN包括两个网络: discriminator和generator

Discriminator:是一个二分类器,判别输入图像是来自training data还是fake.

Generator:一个生成器,输入是random noise,去生成非常可以欺骗discriminator的图片.

2018-01-16 17-53-08屏幕截图

我们的优化目的是G要让D错误分类的可能性升高,而D要让D的检测能力越来越好.

To optimize this minimax game, 交替对G做梯度下降,D做梯度上升,为了解决梯度消失,实践中会变成:

(1)Update the generator ( G ) to maximize the probability of the discriminator making the incorrect choice on generated data:

2018-01-16 17-59-11屏幕截图

(2)Update the discriminator ( D ), to maximize the probability of the discriminator making the correct choice on real and generated data:

2018-01-16 17-59-21屏幕截图

## 2.GAN的训练tricks

见:

<https://github.com/soumith/ganhacks>

<https://arxiv.org/abs/1606.03498>

## 3.其他的生成模型

VAE 变分自编码器

Deeper learning中生成模型的章节.

## 4.实现GAN

### 4.1 LeakyReLU

return tf.maximum(alpha\*x,x)

### 4.2 实现discriminator

注意tf.layer的使用:

文档:https://www.tensorflow.org/api\_docs/python/tf/layers

教程:https://www.tensorflow.org/tutorials/layers

激励函数用的是leakyrelu

### 4.3 实现generator

激励函数用relu

### 4.4 ※GAN loss

包括两个部分:

- D\_loss: discriminator loss scalar

- G\_loss: generator loss scalar

(1)对于给定的真实图片，辨别器要为其打上标签1 D\_real\_loss=tf.nn.sigmoid\_cross\_entropy\_with\_logits(labels=tf.ones\_like(logits\_real),logits=logits\_real)

(2)对于给定的生成图片，辨别器要为其打上标签0 D\_fake\_loss=tf.nn.sigmoid\_cross\_entropy\_with\_logits(labels=tf.zeros\_like(logits\_fake),logits=logits\_fake)

(3)对于生成器传给辨别器的生成图片，生成器希望辨别器打上标签 1

G\_loss=tf.nn.sigmoid\_cross\_entropy\_with\_logits(labels=tf.ones\_like(logits\_fake),logits=logits\_fake)

D\_loss=tf.reduce\_mean(D\_real\_loss+D\_fake\_loss)

G\_loss=tf.reduce\_mean(G\_loss)

PS:

(1)注意到label的shape要和logits一致

(2)注意到求D的loss的时候不要分别求均值,应当求和之后再取均值

### 4.5 Optimizing our loss

if your D(x) learns to be too fast (e.g. loss goes near zero), your G(z) is never able to learn.

Often D(x) is trained with SGD with Momentum or RMSProp instead of Adam, but here we'll use Adam for both D(x) and G(z).

## 5.Least Squares GAN

Loss变化了:

D\_loss = 0.5\*tf.square(score\_real-1)+0.5\*tf.square(score\_fake)

G\_loss = 0.5\*tf.square(score\_fake-1)

D\_loss=tf.reduce\_mean(D\_loss)

G\_loss=tf.reduce\_mean(G\_loss)

## 6.Deep Convolutional GANs

之前的GAN allows no real spatial reasoning. 因为缺乏卷积层.

因此,现在用卷积层来充当D和G.

注意generator里如何实现反卷积.

## 7.※WGAN-GP

Generator: DCGAN

Discriminator: InfoGAN Appendix C.1 MNIST

最重要的是loss的改变:

1. 首先,我们不再使用log损失.先前的log损失,本质是将score经过一个sigmoid归一化到[0,1]之间,来判断这个数和0/1之间的差距.

* 对于D:我们希望fake image的sigmoid score和0越接近越好,real image的score和1越接近越好
* 对于G:我们希望fake image被D评分后的score越接近1越好.

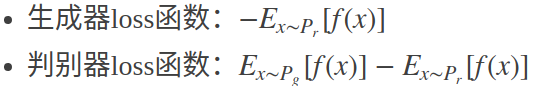
而在wGAN中,现在我们直接用score来计算loss,注意这里我们希望loss越低越好,因此加了负号:

* 对于D:我们希望fake image的score和0越接近越好,real image的score和1越接近越好,也就是

D\_loss = - ( tf.reduce\_mean(logits\_real) - tf.reduce\_mean(logits\_fake) )

* 对于G:我们希望fake image被D评分后的score越接近1越好,也就是

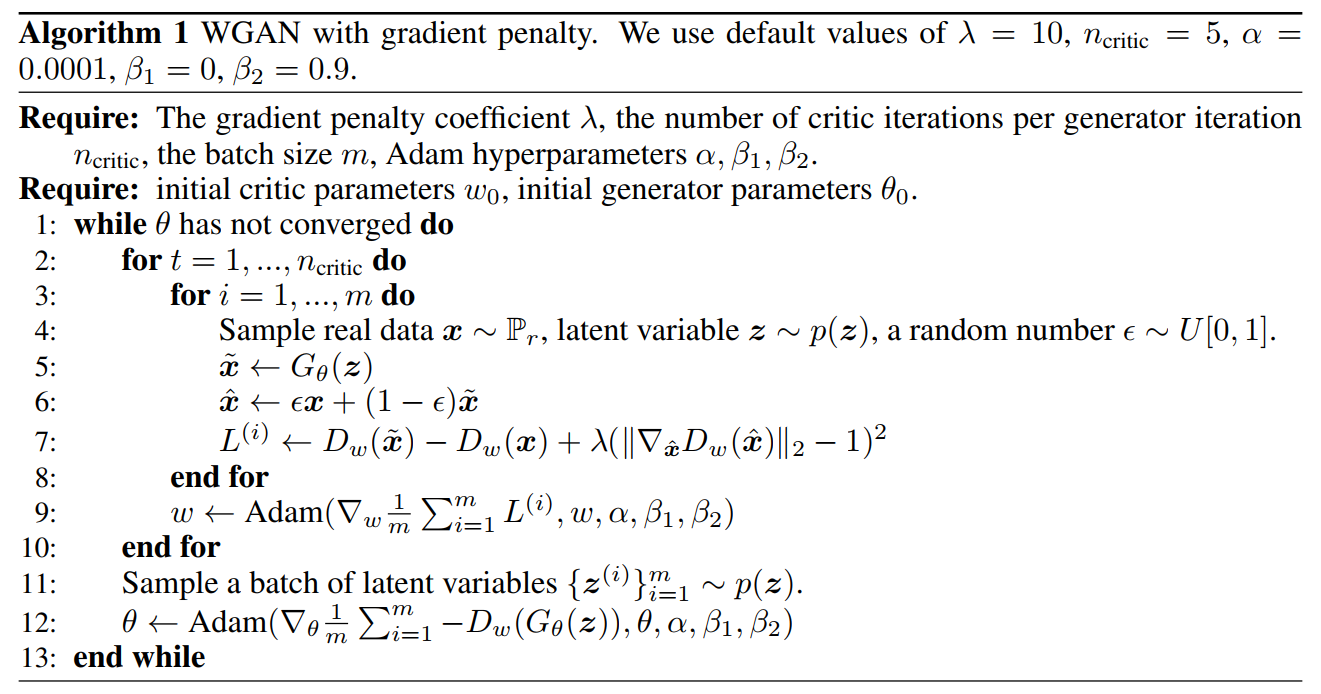
G\_loss = -tf.reduce\_mean(logits\_fake)



(2)在D的loss中添加了对x\_hat梯度的惩罚!

x\_hat是fake image和real image的一个重构图,在loss中添加梯度惩罚的算法如下:

来自这篇:<https://arxiv.org/pdf/1704.00028.pdf>



CS231n中的code:

|  |
| --- |
| lam = 10    eps = tf.random\_uniform(shape=[batch\_size,1],minval=0,maxval=1)  G\_sample\_falt=tf.reshape(G\_sample,[batch\_size,-1])  x\_hat = eps\*x+(1-eps)\*G\_sample\_falt  with tf.variable\_scope('',reuse=True) as scope:  grad\_D\_x\_hat = tf.gradients(discriminator(x\_hat),x\_hat)  grad\_norm = tf.norm(grad\_D\_x\_hat)  grad\_pen = lam\*tf.square(grad\_norm-1)  D\_loss+=grad\_pen |