1. orb-slam的单目帧间匹配利用的是匀速模型，在下一帧图像中对上一帧图像的特征点划定一块固定的区域进行特征点关联。如果对上一帧的特征点利用语义分割贴上标签之后，在下一帧中进行特征点的语义分割，与上一帧进行匹配，速度更快，也不容易丢帧

mask-rcnn语义分割何凯明，把每个特征点标上语义信息  
两帧之间进行单帧匹配（优先级高），重定位

现在的语义slam

1. 将float描述符转换为二进制，直接的转换精度下降很快，可以试试神经网络
2. 如果用语义slam的话，得学习比较多概率的东西，有很多的公式，可能最后比较难，如果用二进制描述符的话，可能需要的相关知识少一些
3. 描述子的trick：

光照不敏感：灰度梯度（SIFT）

噪声不敏感：直方图（SIFT）

亮度影响：特征归一化（SIFT）

描述子占用空间大：二进制描述子（BRIEF、ORB）

尺度不变：图像金字塔

视角不变性：特征点的主方向（ORB）

1. 应该花两三天对所有出现的特征点和描述子进行总结归纳，了解每个的优缺点，实现方法
2. 孪生网络训练方式借鉴，可以考虑结合语义之类，用Harris或其他角点检测方法，提取的还是低维度信息，而且通篇都是利用3D投影变换进行的学习，不是单纯的像Superpoint着眼于特征点提取上，尽管在outdoor数据上也取得了较好的效果，但其他泛化效果有待考察（利用RGB-D局限于室内尺度），但任务驱动型思想很好。

GCNv2实现了在TX2上的ORB-SLAM架构的实时slam

1. GNN注意力机制
2. 需要学习的东西：

传统的特征提取（Harris、SIFT、SURF），

图像预处理的方法，包括图像的滤波、分裂、归并、分割以及形态学处理等等，也可以与深度学习方法相结合，例如dilated CNN就可以看做形态学操作与CNN的结合

相关的深度学习架构：LSTM、RNN、GNN、GAN

ORB-SLAM2的代码

马太原学长的代码跑起来、理解

思考的方向：基于马太原学长的论文，写一下深度学习生成二进制描述符，最后最好能达到在TX2上运行，也不一定用马太原学长论文里面的损失函数，自己可以结合其他的论文进行损失函数的设计

1. 马太原学长论文：论文里面已经突出的优点在我的论文里面就一笔带过，主要实验就比较二进制，突出实时性，内存占用，帧率，最好能在tx2上跑。

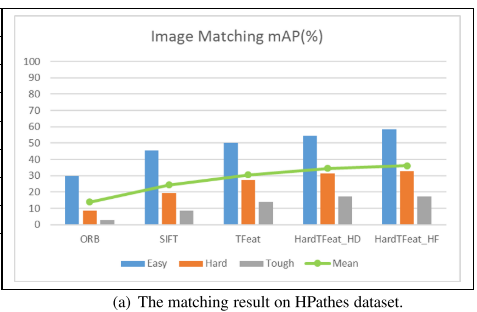
有篇论文实验显示深度学习描述符不如传统描述符，但是只描述了现象，没有分析原因，学长给出的原因是slam的匹配机制显示深度学习方法，在代价函数上有理论缺陷

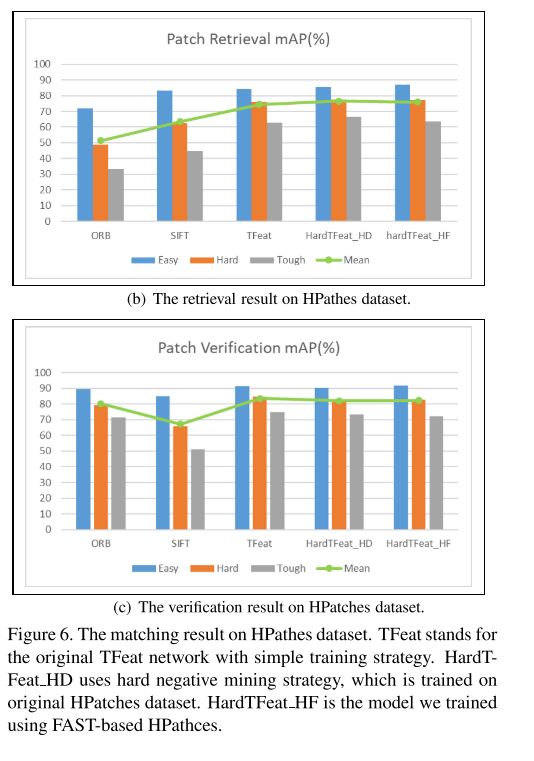
广读论文，看看人家的二进制有没有什么缺点，能不能有什么方法去解决，以及能不能有一些理论

最终的写的东西：

1.比较FPR95，用一个表格对不同描述子在Brown dataset上做对比，即单纯地比较深度学习描述子。

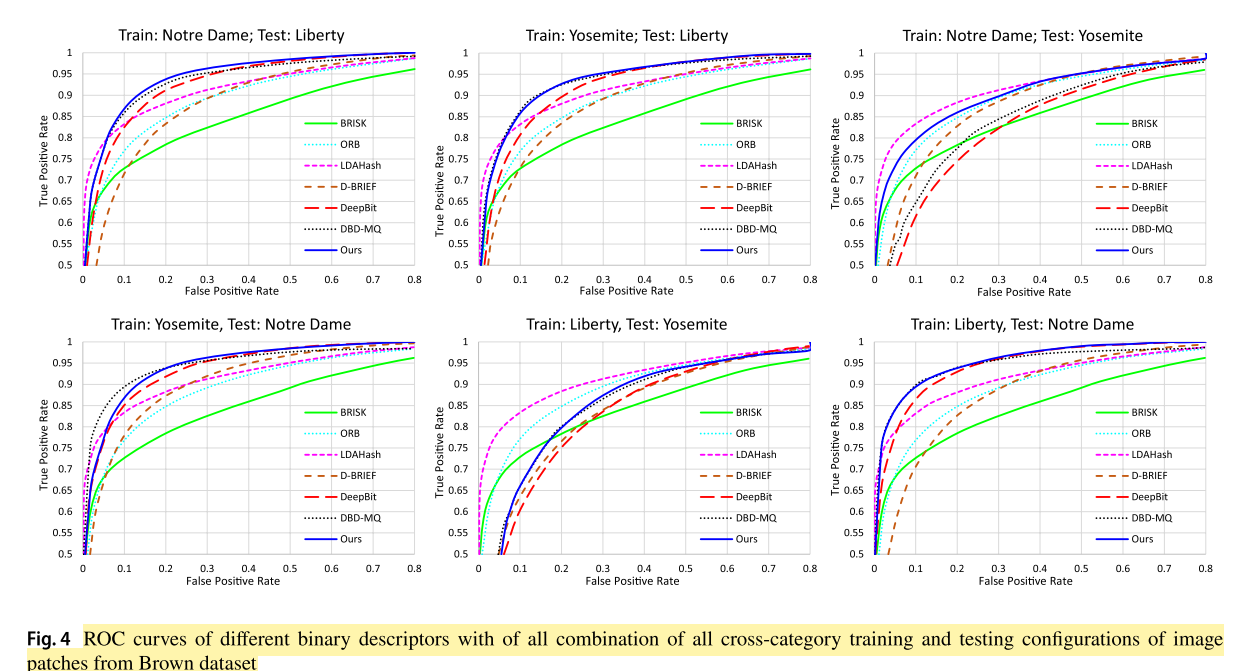
2.用HPatches进行比较（如果自己不去实现这些描述子的结果，则直接取出）（





（DF-SLAM: A Deep-Learning Enhanced Visual SLAM System based on Deep Local Features）

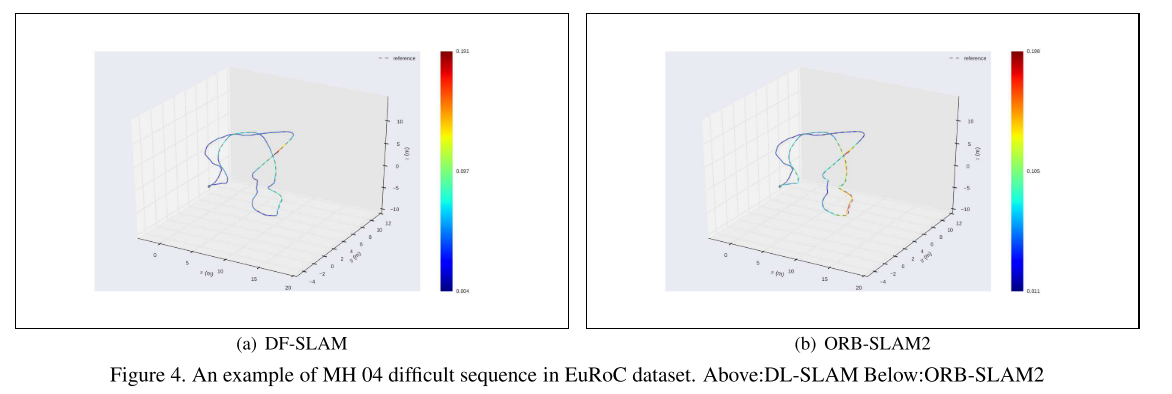
3.不同的FPR，画一个ROC图，但前提是需要实现其他不同的描述子，工作量可能有点大（Deep learned compact binary descriptor with a lightweight network‑in‑network architecture for visual description）

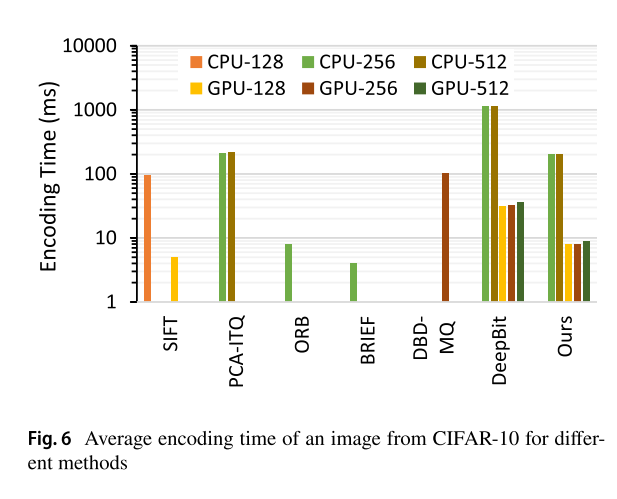


The Notre Dame and Liberty datasets consist of weak visual similarity over different patches, whereas Yosemite（数据集中不同的patches有很强的相似性，所以更难） consists of different patches that have a strong visual resemblance, making it the most challenging of the three datasets. Hence, most of the approaches, including our method, show a higher error rate for Yosemite than for the other two datasets. Figure 4 shows that most of the learning-based methods, including our approach, suffer from a high false positive rate when the Hamming radius is increased

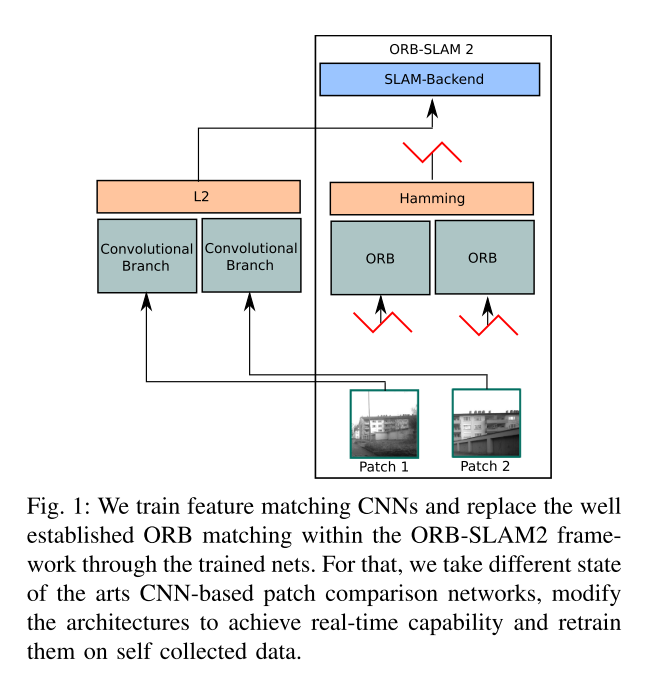
4.与ASD、ORB的在KITTI数据集上的回环时间的比较，对于一个数据集中有多个回环的可以取平均值，可以画一个柱状图（引自Deep learned compact binary descriptor with a lightweight network‑in‑network architecture for visual description），但是ORB的是在CPU上的，所以跟ORB可能没有可比性，不对，回环检测都是在CPU上跑词袋得到的，所以有一定的可比性

5.两种SLAM在数据集上的对比图（DF-SLAM: A Deep-Learning Enhanced Visual SLAM System based on Deep Local Features），用轨迹的颜色来代表与groundTruth的偏差，用evo绘制

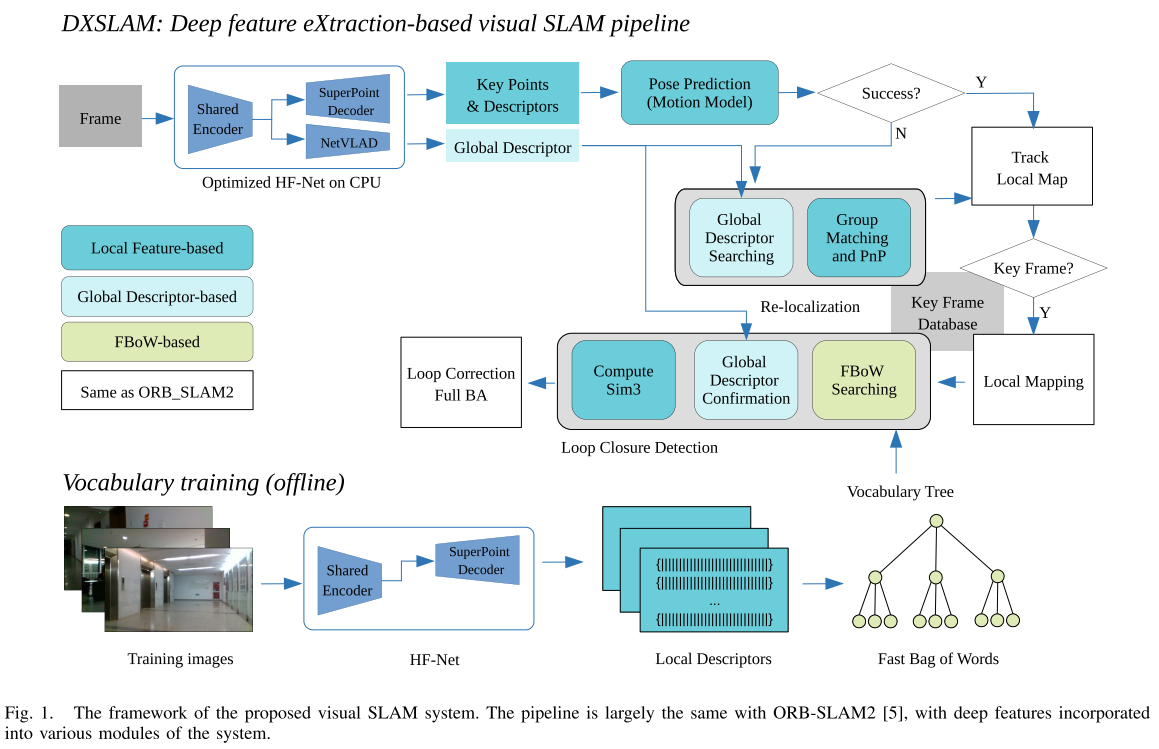




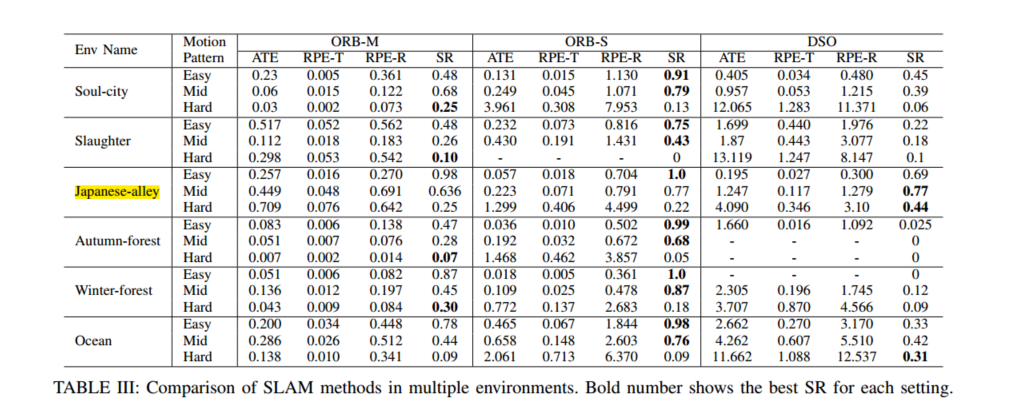
6.还有一种表示自己结构的画法（An Approach for CNN-Based Feature Matching Towards Real-Time SLAM ITSC2019）



DXSLAM: A Robust and Efficient Visual SLAM System with Deep Features



7.用TartanAir数据集来比较ORB-Monocular和ORB-Stereo（反正我的描述子用的BASD就行了，也可以说我的可移植性）（TartanAir: A Dataset to Push the Limits of Visual SLAM  
）



对于Brown dataset的介绍：Working hard to know your neighbor’s margins: Local descriptor learning loss

Contribution：

1. ~~对于ASD描述子框架进行了改进~~ 自己提出了一个描述符的方法和网络（将ReLU改为Tanh，可以），在克服尺度问题（可以提一下，只是不是自己做的）的同时，并且进行了二进制化（如何二进制化的，要不要自己提出一个新的改进的二进制化的方法，或者一个新的训练方法），得到了一个新的描述子，并且保证了描述子的准确度（在brown dataset上精度可以，可以考虑在HPatches上比较一下）。
2. 根据BASD设计了BASD-SLAM框架，在KITTI数据集上回环检测时间比ASD-SLAM高，轨迹估计比ORB精度高，在tartanAir数据集上BASD-SLAM的双目和单目都比原始的ORB好

目的：内存、速度、尺度性（最终实验用精度、准确性来体现，如果说了尺度性的话还能不能说双目SLAM）。二进制为了前两个，后面一个也能同时保证。灵感来于ASD，现在的二进制不能保证尺度。代价函数对描述符的学习是起决定性作用的，这是公认的。

Tanh作为激活函数（逼近sign函数），~~比ReLU更能保证尺度性~~（尺度性不是由激活函数来保证的，而是由损失函数来保证的）比ReLU更能更好地得到二进制描述符串。如果理论证明不了，也可以用实验现象说明，可以从过程上满足这个过程。可以将梯度图画出来证明。可以尝试从理论上去解释这个文章。（找一下sigmoid，relu，tanh相关的文章，看看他的好处坏处）每天看两篇。

ASD没搞二进制，我改进的网络的float比ASD好，并且同时兼顾了二进制的效果（比CDbin更好，放一个表格），比ASD扩展性好。实用性上：对于实时性要求高的用二进制，对于精度要求高的用float型。

实验证明：

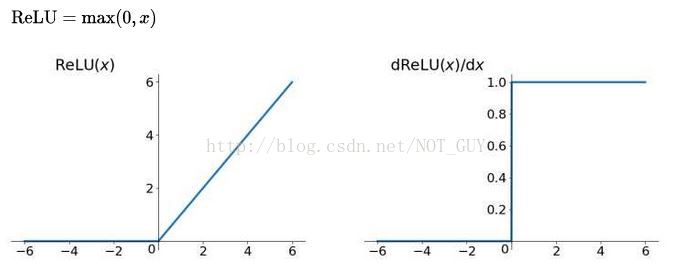
内存：同样的2000张图片，ASD和我的训练的词袋速度和词袋的大小

速度：一张图片描述子提取所用的时间（float和二进制的对比），单帧匹配的时间。二进制肯定要更好，因为是位运算

深度学习二进制描述符的slam对比也是需要的，得做一下实验。因为没有现成的二进制描述符的开源代码，就用2017年NIPS的文章的hardnet进行二进制来进行SLAM的比较。用今年的新的数据集tartanair进行对比

Tanh->ReLU之后为什么二进制效果更好，找理论依据

ReLU函数：0附近使用的是次梯度。



优点：（1）ReL的收敛速度比 sigmoid 和 tanh 快；（梯度不会饱和，解决了梯度消失问题）

    （2）计算复杂度低，不需要进行指数运算；

    （3）适合用于后向传播。

缺点：（1）ReLU的输出不是zero-centered；

    （2）Dead  ReLU  Problem（神经元坏死现象）：某些神经元可能永远不会被激活，导致相应参数永远不会被更新（在负数部分，梯度为0）。产生这种现象的两个原因：参数初始化问题；learning  rate太高导致在训练过程中参数更新太大。 解决方法：采用Xavier初始化方法，以及避免将learning  rate设置太大或使用adagrad等自动调节learning  rate的算法。

（3）ReLU不会对数据做幅度压缩，所以数据的幅度会随着模型层数的增加不断扩张。（BN）可以解决

**ReLU在x>0下，导数为常数1的特点：**

导数为常数1的好处就是在“链式反应”中不会出现梯度消失，但梯度下降的强度就完全取决于权值的乘积，这样就可能会出现梯度爆炸问题。解决这类问题：一是控制权值，让它们在（0，1）范围内；二是做梯度裁剪，控制梯度下降强度，如ReLU(x)=min(6, max(0,x))

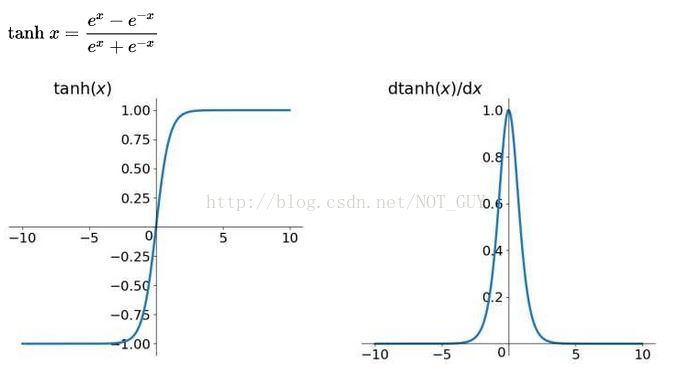
**ReLU在x<0下，输出置为0的特点：**

描述该特征前，需要明确深度学习的目标：深度学习是根据大批量样本数据，从错综复杂的数据关系中，找到关键信息（关键特征）。换句话说，就是把密集矩阵转化为稀疏矩阵，保留数据的关键信息，去除噪音，这样的模型就有了鲁棒性。ReLU将x<0的输出置为0，就是一个去噪音，稀疏矩阵的过程。而且在训练过程中，这种稀疏性是动态调节的，网络会自动调整稀疏比例，保证矩阵有最优的有效特征。

但是ReLU 强制将x<0部分的输出置为0（置为0就是屏蔽该特征），可能会导致模型无法学习到有效特征，所以如果学习率设置的太大，就可能会导致网络的大部分神经元处于‘dead’状态，所以使用ReLU的网络，学习率不能设置太大。

有一篇论文提到图像信息稠密，所以用ReLU可以进行图像信息的稀疏，可以去除噪音，并且可以防止梯度消失。但是因为我们使用的是low-level信息，只是从image-patch里面提取信息，并且网络shallow，所以ReLU丢失的很多信息就会对最终的结果产生影响，导致最终无法正确的分类。并且二进制最后一层要用0做阈值做二分类，所以隐藏层用ReLU会丢失很多负区间的信息。

2、tanh函数



tanh函数将输入值压缩到 -1~1 的范围，因此它是0均值的，解决了Sigmoid函数的非zero-centered问题，但是它也存在梯度消失和幂运算的问题。

其实 tanh(x)=2sigmoid(2x)-1

需要激活函数是零均值，因为我们最终的目的就是要用0对浮点型描述子进行二进制化。而且tanh函数更能逼近sign函数。ReLU函数当输入值为负的时候，他的输出为0，而我们用0做阈值进行二进制的时候，这是一个比较致命的影响，它会降低负数在网络结果中出现的频率。而在一个大数据的情况下，每一个bit出现0和1的概率应该是一样的。而且我们会在Tanh前使用BN，所以整个数值不会很大，Tanh的梯度消失不会有多大影响。在浅层网络中表达能力上升。Tanh能够将0附近的一个分部映射到另一个0附近。

可以从数据集预测结果的浮点数在正负区间的点的数量来比较，如果描述子的值的正负数量差别比较大，则表示ReLU的信息量更少

论文：Bi-Real Net--Enhancing the Performance of 1-bit CNNs\_ECCV\_2018\_paper.pdf

ReLU激活函数是非负的，而Sign是-1和+1.因为这个不同，实数型CNNs用ReLU可能不能提供一个好的初始值来训练1-bit CNNs。所以我们用clip(-1,x,1)来代替ReLU预训练实数CNN模型，因为clip激活函数比ReLU更接近sign函数。这个替换效果后面会用实验来评估

我可以用ReLU和Tanh作为激活函数进行对比实验，做一个表格

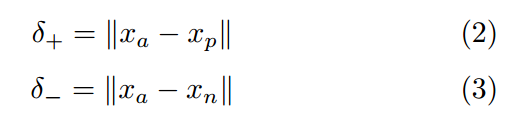
Tanh比ReLU好的原因：

1.我们最终的目的是用0做阈值对浮点型描述子进行二进制化，Tanh是零均值的，更能逼近sign函数，而ReLU不是0均值的，最终用0做阈值不合理。

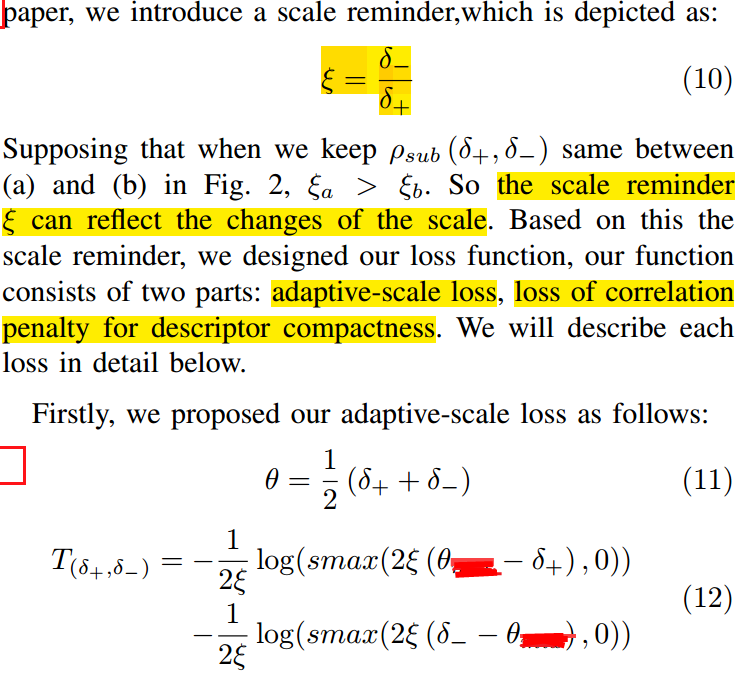
2.网络比较浅层，当输入值为负的时候，ReLU的输出为0，他会丢失大量在负数区间的信息，降低负数在网络结果中出现的概率，导致最终的二进制结果不理想

尺度不变性的讨论：

学长论文的尺度不变性是对生成的128维二进制描述子进行了归一化，得到xa,xp,xn，得到，然后再计算L2距离



然后因为利用triplet loss的时候是用的δ+-δ-，所以当两者同时增大或者减小同样的数量，而损失函数不变，导致了尺度空间不连续。所以用了一个尺度因子去引导网络注意尺度的变化

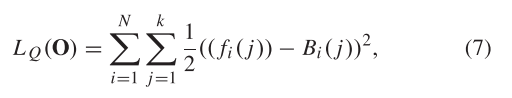


我如何在二进制中说明也有尺度问题，或者如何保证尺度问题的解决方法有用。如何说明Tanh的优越性（没法说明，早就有论文说了这个问题）

二进制的尺度问题：因为sign函数是不可导的，所以反向传播时有问题的，因此只能利用浮点型描述子来进行训练，然后最后

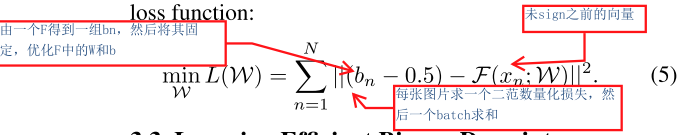
Quantization loss

CDbin是用的如下公式，但是二进制sign函数不可导，没办法反向传播。不知道他们怎么做的，还可以得到比较好的结果。



CDbin引用的CVPR2016的文章Learning Compact Binary Descriptors with Unsupervised Deep Neural Networks。人家也用了这个量化损失，不过人家是迭代优化，每次用网络得到一个bn之后，固定这个bn，然后用这个式子去优化W，并没有直接去优化。这样就比较麻烦。

F:\Users\gxf\AppData\Local\Temp\1607559306(1).png



而量化损失的目的仅仅是计算与±1的距离，则可以用下式来代替量化损失，也能达到让xa接近±1，并且不会存在不可导问题。

（并且描述子二进制化与一般的深度学习网络的二进制化不一样，只要符号相同，最后的二进制结果就是一样的。而网络的二进制会产生量化损失所以需要量化后的值无限接近于原值，才能有比较好的精度。但是描述符的二进制化只需要让正的更大一点，负的更多一点，只要能保证符号正确即可）



现在SLAM的结果不怎么好，跟其他的差不多：不放SLAM；为啥有的场景好，有的场景不好；回避SLAM。

如果要带SLAM的话那就必须从光照（搞不了，人家都是从光流法 方向搞的）和视角方向加一些特性才行。

自己想一下创新点：提出二进制的方法；二进制解决尺度问题。提出二进制的尺度问题（mty学长的论文也没多少人知道，所以也不要太担心），然后自己的网络能够保证尺度问题，需要画一幅图：δ+和δ-和Loss图，要能看出趋势是向δ+小，δ-大的方向落。这个图必须出。（这个感觉意义不大，）

最后可以画一幅mAP的来进行比较，画一个Brown dataset的来进行比较。

可以找找其他的二进制方法，然后扯上尺度问题