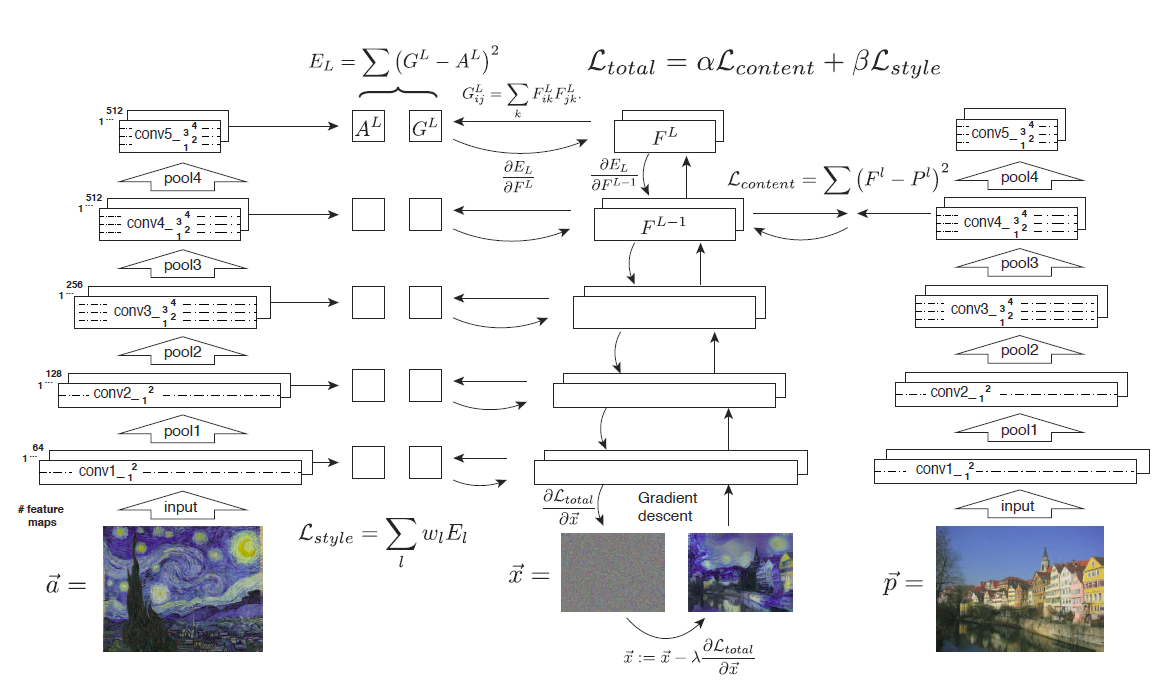
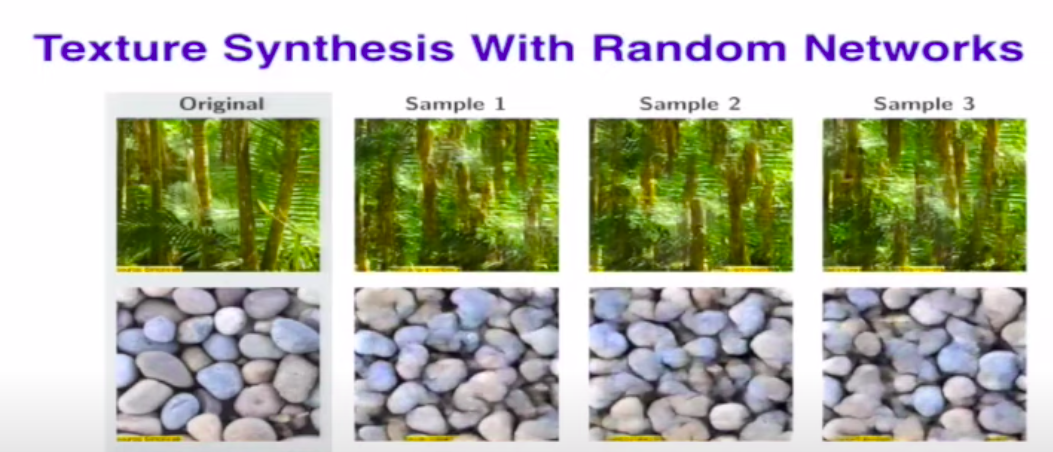
通常假设样本空间中全体样本服从一个未知的“分布”（distribution），我们获得的每个样本都是独立地从这个分布上采样获得的，即“独立同分布”（independent and identically distributed，简称 i.i.d.） ——《机器学习》 周志华

Here we use L-BFGS [32], which we found to work best for image synthesis. To extract image information on comparable scales, we always resized the style image to the same size as the content image before computing its feature representations. Finally, note that in difference to [24] we do not regularize our synthesis results with image priors. It could be argued, though, that the texture features from lower layers in the network act as a specific image prior for the style image. Additionally, some differences in the image synthesis are expected due to the different network architecture and optimization algorithm we use.



Style transfer algorithm. First content and style features are extracted and stored. The style image ~a is passed through the network and its style representation Al on all layers included are computed and stored (left). The content image ~p is passed through the network and the content representation Pl in one layer is stored (right). Then a random white noise image ~x is passed through the network and its style features Gl and content features Fl are computed. On each layer included in the style representation, the element-wise mean squared difference between Gl and Al is computed to give the style loss Lstyle (left). Also the mean squared difference between Fl and Pl is computed to give the content loss Lcontent (right). The total loss Ltotal is then a linear combination between the content and the style loss. Its derivative with respect to the pixel values can be computed using error back-propagation (middle). This gradient is used to iteratively update the image ~x until it simultaneously matches the style features of the style image ~a and the content features of the content image ~p (middle, bottom).

We find that matching the style representations up to higher layers in the network preserves local images structures an increasingly large scale, leading to a smoother and more continuous visual experience. Thus, the visually most appealing images are usually created by matching the style representation up to high layers in the network, which is why for all images shown we match the style features in layers ‘conv1\_1’, ‘conv2\_1’, ‘conv3\_1’, ‘conv4\_1’ and ‘conv5\_1’ of the network.



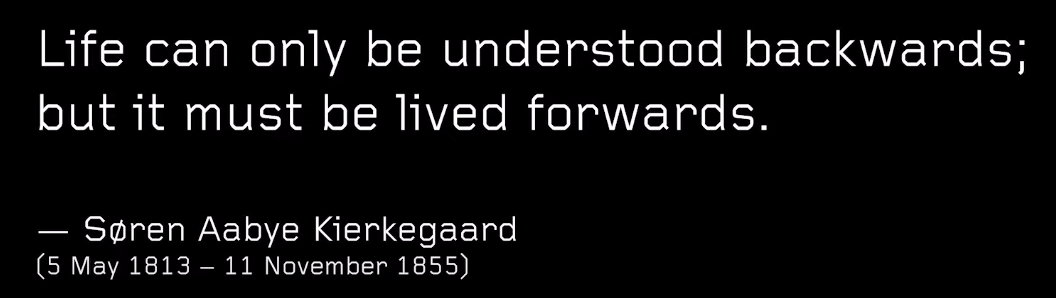
Try us use MSE loss function and

Optimizer = optim.LBFGS([input\_img.requires\_grad\_()])

What will a dream of a dream look like?

To impress the audience.

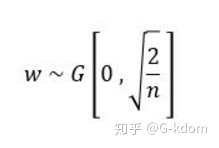




**He（何） initialization（MSRA）**

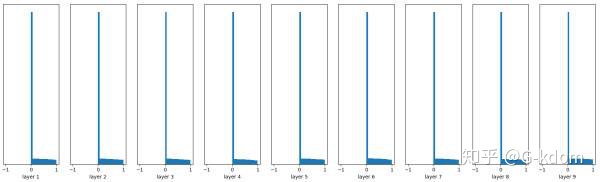
为了解决上面的问题，何恺明大神提出了一种针对ReLU的初始化方法，一般称作 He initialization 或 MSRA。论文链接：[Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification](https://link.zhihu.com/?target=https%3A//arxiv.org/pdf/1502.01852.pdf" \t "_blank)

主要想要解决的问题是由于经过ReLU后，方差会发生变化，因此我们初始化权值的方法也应该变化。只考虑输入个数时，MSRA初始化是一个均值为0，方差为2/n的高斯分布：

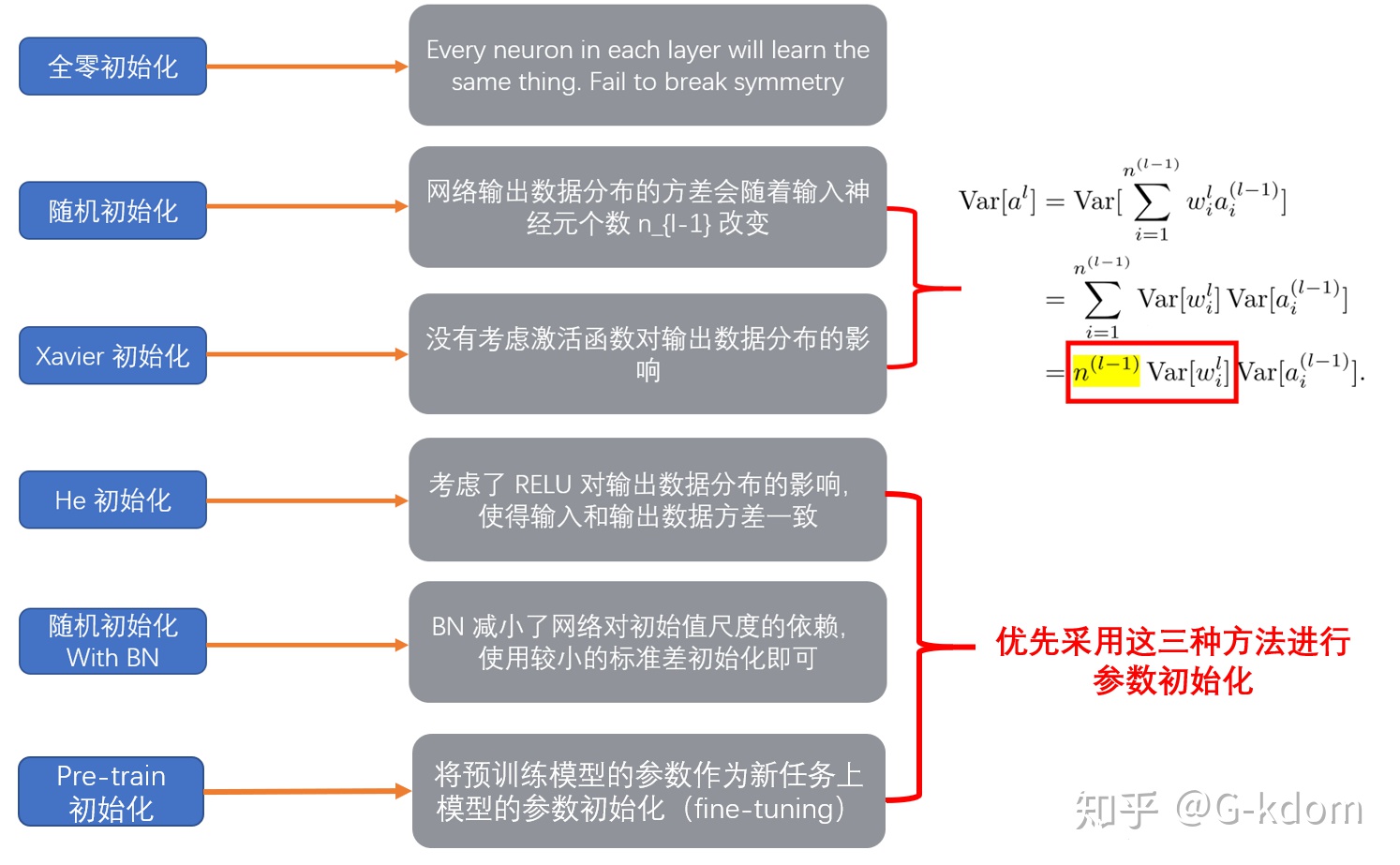


**He initialization的思想是：**在ReLU网络中，假定每一层有一半的神经元被激活，另一半为0（x负半轴中是不激活的），所以要保持variance不变，只需要在Xavier的基础上再除以2：

w = np.random.randn(node\_in, node\_out) / np.sqrt(node\_in/2)



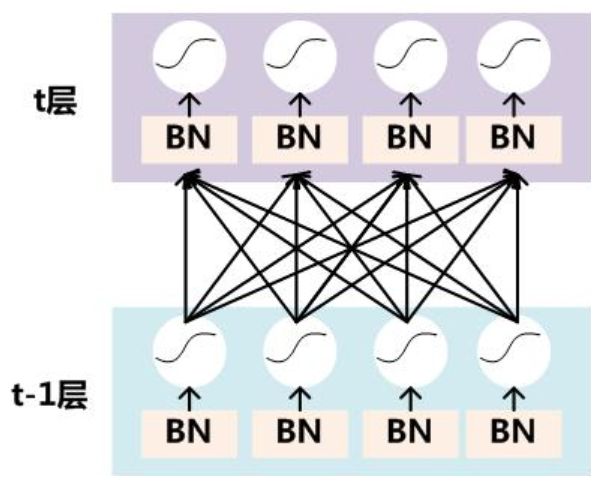
效果是比Xavier initialization好很多。现在神经网络中，隐藏层常使用ReLU，权重初始化常用He initialization这种方法。



在深度学习中，**神经网络的权重初始化方法对对模型的收敛速度和性能有着至关重要的影响**。在深度神经网络中，随着层数的增多，我们在梯度下降的过程中，极易出现梯度消失或者梯度爆炸。一个好的权重初始化虽然不能完全解决梯度消失和梯度爆炸的问题，但是对于处理这两个问题是有很大的帮助的，并且十分有利于模型性能和收敛速度，在某些网络结构中甚至能够提高准确率。因此，对权重w的初始化则显得至关重要。

BN(Batch Normalization)的想法简单粗暴，发生了没关系，我再加一层 layer，不厌其烦得把每次输出的分布都纠正回来。这样我们再权重初始化的时候，就可以放心大胆得随机了。

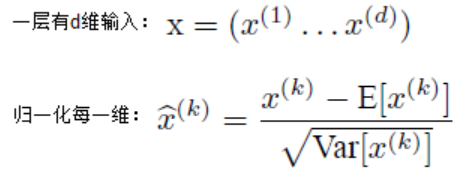
**BN的本质**是在网络的每一层输入的时候，又插入了一个**归一化层**，也就是先做一个归一化处理（均值0、方差为1），然后再进入网络的下一层。(如下图所示)



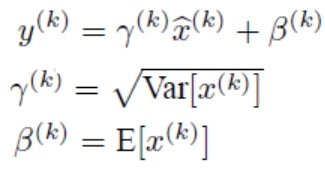
（图取自，[https://www.cnblogs.com/carlber/p/10752240.html](https://link.zhihu.com/?target=https%3A//www.cnblogs.com/carlber/p/10752240.html" \t "_blank)）

道理是比较简单，实际上BN层的运算包括了两个步骤，先“**归一化（Normalize）”，后“ 变换重构（Scale and shift）”**

**先归一**



**后变换重构**



为什么需要变换重构呢？因为强行归一化可能会损坏原始某一层学习到的特征分布，有公式可以证明通过**学习到**的重构参数γ、β，是可以恢复出原始的某一层所学到的特征的。

BN层也是需要学习的。

**BN优点**

(1) 减轻了对参数初始化的依赖  
(2) 训练更快，可以使用更高的学习率。  
(3) BN一定程度上增加了泛化能力，dropout等技术可以去掉。

Style Transfer

先黑白再上色效果好一些？

DenseNet对显存不是太友好，工业界也不爱用。好像也就是image to image的任务上稍微多见一些，因为需要对细节特征也有比较好的捕捉能力，比如semantic segmentation，super resolution之类。

# **2. 激活函数（Activation Functions）**

    总结：

     1）使用ReLU时，使Learning Rates尽量小

     2）尝试使用Leaky ReLU/Maxout/ELU

     3）可以使用tanh，但期望不要太高

     4）不要使用sigmoid

VGG-16网络

VGGNet是牛津大学计算机视觉组（Visual Geometry Group）和Google DeepMind公司的研究员一起研发的深度卷积神经网络。

VGGNet探索了卷积神经网络的深度与其性能之间的关系，通过反复堆叠3\*3的小型卷积核和2\*2的最大池化层，

VGGNet成功地构筑了16~19层深的卷积神经网络。VGGNet相比之前state-of-the-art的网络结构，错误率大幅下降，

VGGNet论文中全部使用了3\*3的小型卷积核和2\*2的最大池化核，通过不断加深网络结构来提升性能。

总结：

（1）VGG-16网络中的16代表的含义为：含有参数的有16个层，共包含参数约为1.38亿。

（2）VGG-16网络结构很规整，没有那么多的超参数，专注于构建简单的网络，都是几个卷积层后面跟一个可以压缩图像大小的池化层。即：全部使用3\*3的小型卷积核和2\*2的最大池化层。

卷积层：CONV=3\*3 filters, s = 1, padding = same convolution。

池化层：MAX\_POOL = 2\*2 , s = 2。

（3）优点：简化了卷积神经网络的结构；缺点：训练的特征数量非常大。

（4）随着网络加深，图像的宽度和高度都在以一定的规律不断减小，每次池化后刚好缩小一半，信道数目不断增加一倍。

为了比较不同模型的质量，我们收集了文献中的一些数据，分析发现不同的抽样方法得出的结论也不一样。比如，VGG-16和GoogleNet 的central-crop误差分别是8.7%和10.07%，表明VGG-16性能优于googleNet，而用10-crop抽样，则误差分别是9.33%和9.15%，VGG-16又比GoogleNet差了。于是，我们决定基于分析，对所有网络重新评估，使用单个central-crop抽样方法。

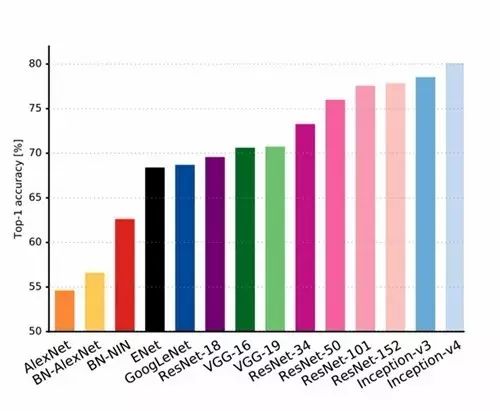
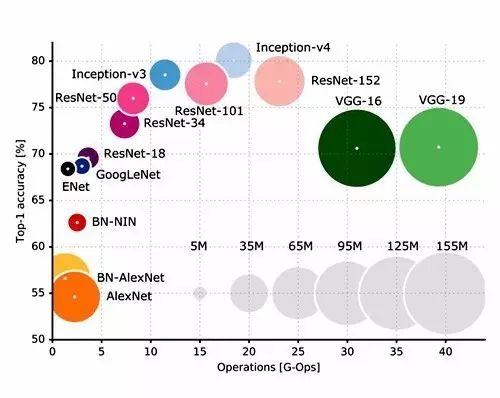
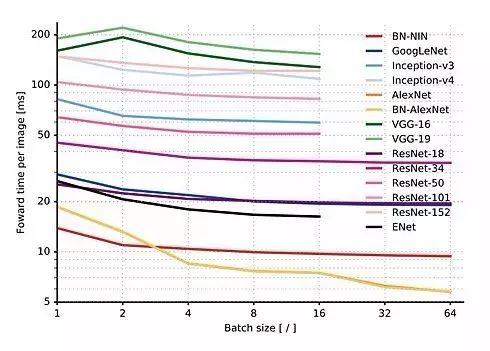


图1: Top1 vs. 网络.  Single-crop top-1 用最高评分体系检测准确度。上图中不同的配色方案表示不同的架构和作者。注意，同组网络共享相同的色相，比如所有的ResNet系列都是用粉色系表示的。

  
图 2: Top1 vs. 操作、数量大小、参数  Top-1 one-crop 准确度与单向前进传递所需操作次数的对比。图中气泡大小与网络参数数量成正比；右下角记录的是从5\*106 到155\*106参数值的历史最大值；所有数据都共享一个y轴，灰色点表示气泡中心的值。

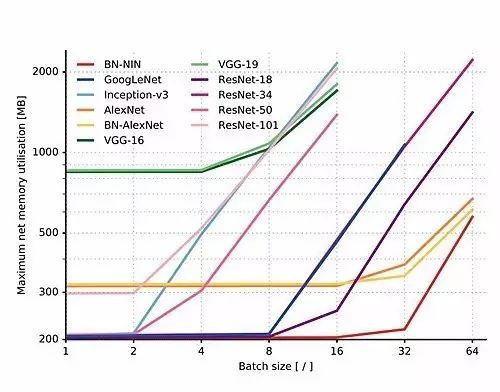
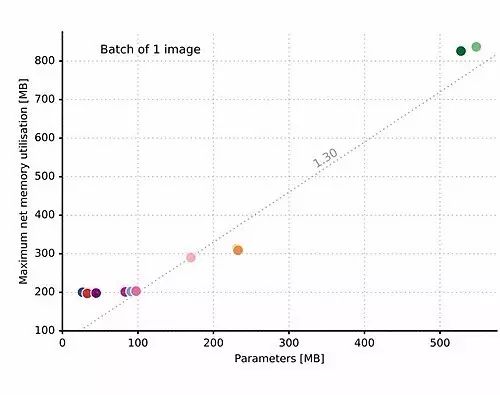
**推理时间（Inference Time）**

****

上图（图3）显示了各架构在每个图像上的推理时间，作为一个图像批大小（从1到64）函数。我们注意到 VGG 处理一张图像所需时间约1/5秒，这使它在 NVIDIA TX1 上实时应用的可能性较小。AlexNet 的批大小从1到64的变化中，处理速度提升了3倍，这是由于它的完全连接层的弱优化，这个发现令人惊讶。

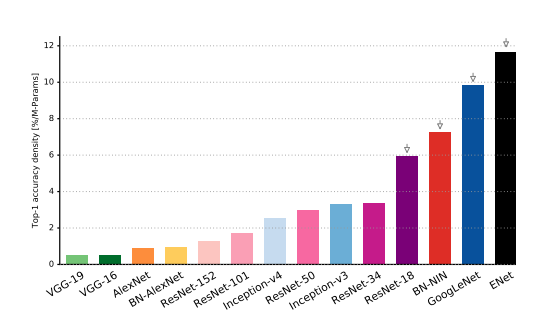
**内存（Memory）**

分析使用 CPU 和 GPU 共享内存的 TX1 设备的系统内存消耗得到的结果由下图可见，最初最大系统内存使用情况是不变的，随着批大小增加，内存消耗增大。这是由于网络模型的初始内存分配以及批处理时的内存需求随着图像数量的增加而成比例地增加。(有时候内存比核心频率更重要，频率低速度慢可以等，但是内存不够，有的模型没法装载，根本跑不起来)

****

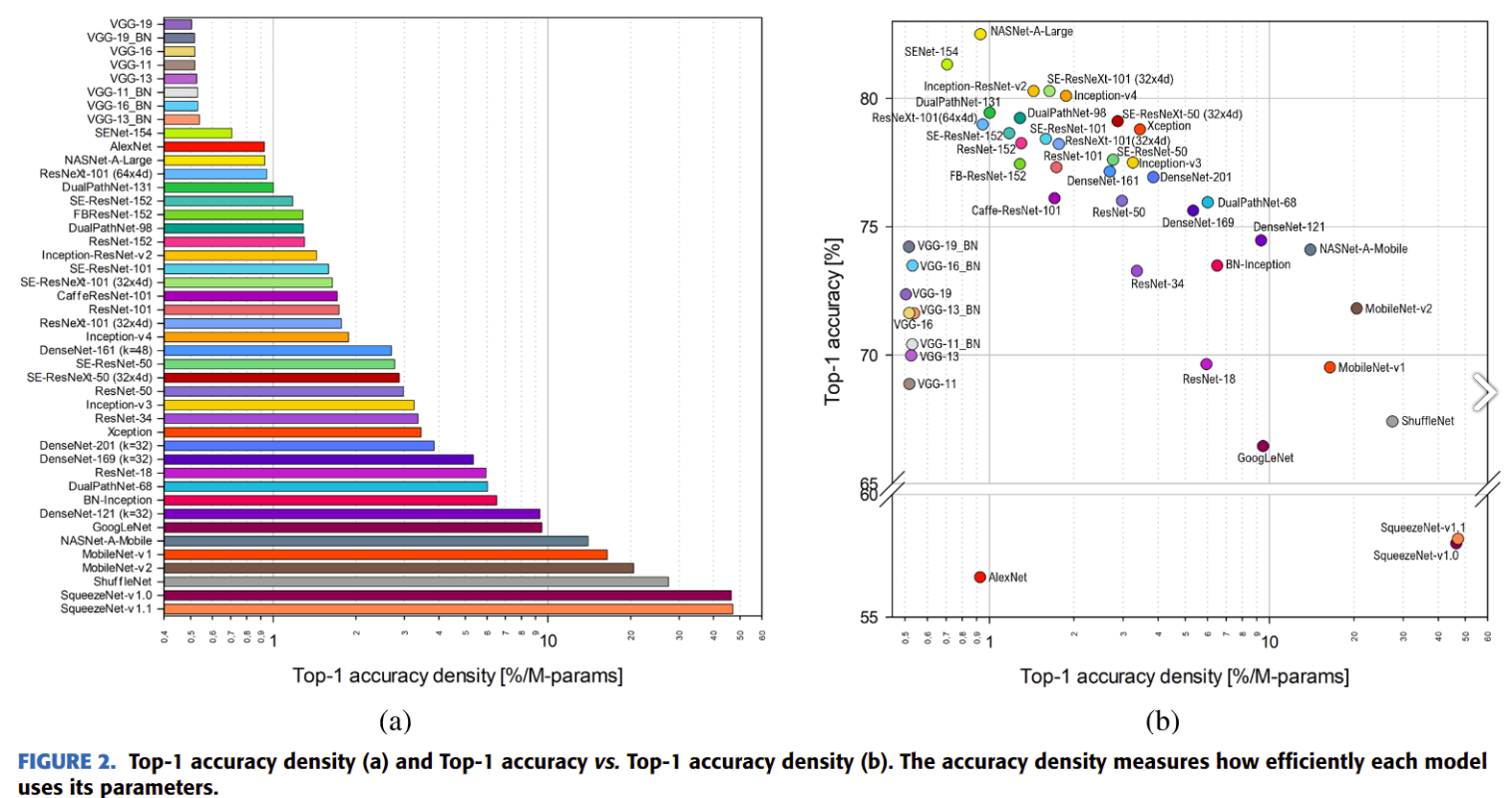
从上图中我们注意到，对规模小于 100 MB的网络，初始内存分配不会小于 200 MB，而且随后呈现为一条斜率为1.3的线性函数。

**参数使用**

****

我们已经知道，DNN 在利用全部学习能力（参数数量/自由度）方面非常低效。Han et al., 2015 的研究利用 DNN 的这个缺陷，使用权重剪枝（weights pruning）、量化（quantisation）和变长编码（variable-length symbol encoding）将网络规模减小了50倍。值得注意的是，使用更高效的架构能够产生更紧凑的呈现。如上图所示，虽然 VGG 比 AlexNet 的准确率更高（图1），但其信息密度不如 AlexNet。这意味着在 VGG 架构中引入的自由度带来的准确率上的提高不大。

ENet 是当前对参数空间利用率最好的架构。



When training a deep neural network, our biggest concern is nearly always overfitting — and in order to combat overfitting, we introduce regularization techniques. We apply regularization in the form of:

* Dropout
* L2 weight decay
* Reducing model capacity (i.e., a shallower model)

We also tend to be a bit more conservative with our learning rate to ensure our model does not overshoot areas of lower loss in the loss landscape.

That is all fine and good, but sometimes we end up over-regularizing our models.

If you go through all three reasons for validation loss being lower than training loss detailed above, you may have over-regularized your model. Start to relax your regularization constraints by:

* Lowering your L2 weight decay strength.
* Reducing the amount of dropout you’re applying.
* Increasing your model capacity (i.e., make it deeper).

You should also try training with a larger learning rate as you may have become too conservative with it.

When the computational resources are limited (usually in a DL project), over-regularizing or underfitting should also draw some attention.

