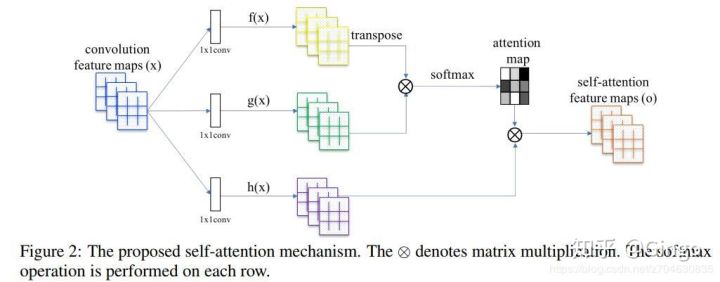
**SAGAN Ian Goodfellow**

由于卷积的局部感受野的限制，如果要生成大范围相关（Long-range dependency）的区域会出现问题，用更深的卷积网络参数量太大，于是采用将 Self Attention 引入到了生成器（以及判别器）中，使用来自所有特征位置的信息生成图像细节，同时保证判别器能鉴别距离较远的两个特征之间的一致性，获取全局信息。 IS从36.8提到了52.52，并把FID（Fréchet Inception Distance）从27.62降到了18.65。



SAGAN 使用注意力机制，高亮部位为注意力机制关注的位置。



传统的GAN也不是万能的，它有下面两个不足：

1. 没有用户控制（user control）能力 在传统的GAN里，输入一个随机噪声，就会输出一幅随机图像。但用户是有想法的，我们想输出的图像是我们想要的那种图像，和我们的输入是对应的、有关联的。比如输入一只猫的草图，输出同一形态的猫的真实图片（这里对形态的要求就是一种用户控制）。
2. 低分辨率（Low resolution）和低质量（Low quality）问题 尽管生成的图片看起来很不错，但如果你放大看，就会发现细节相当模糊。 ----------------朱俊彦（Jun-Yan Zhu） Games2018 Webinar 64期 ：[Siggraph 2018优秀博士论文报告](https://link.zhihu.com/?target=https%3A//games-cn.org/games-webinar-20180913-64/" \t "_blank)

### GauGAN（SPADE） Nvidia

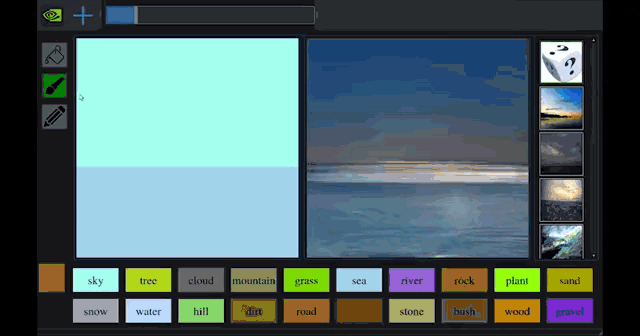
你画一幅涂鸦，用颜色区分每一块对应着什么物体，它就能照着你的大作，合成以假乱真的真实世界效果图。 通过语义布局进行图像的生成 Segmentation mask，算法是源于Pix2PixHD，生成自然的图像。

数据来源是成对的，通过自然场景的图像进行分割，就可以得到分割图像的布局，组成了对应的图像对。 但是区别在于，之前的Pix2PixHD，场景都很规律，如室内，街景，可以使用BN，但是后来发现Pix2PixHD在COCO这些无限制的数据集训练结果很差。如果整张天空或者整张的草地，则计算通过BN后，结果很相似，这样合成会出现问题。于是修改了BN，这种方法称为空间自适应归一化合成法SPADE。将原来的label信息代入到原来BN公式中的γ和β

Semantic Image Synthesis with Spatially-Adaptive Normalization--CVPR 2019。

这篇论文的一作，照例还是实习生。另外几位作者来自英伟达和MIT，CycleGAN的创造者朱俊彦是四作。

在基于语义合成图像这个领域里，这可是目前效果最强的方法。

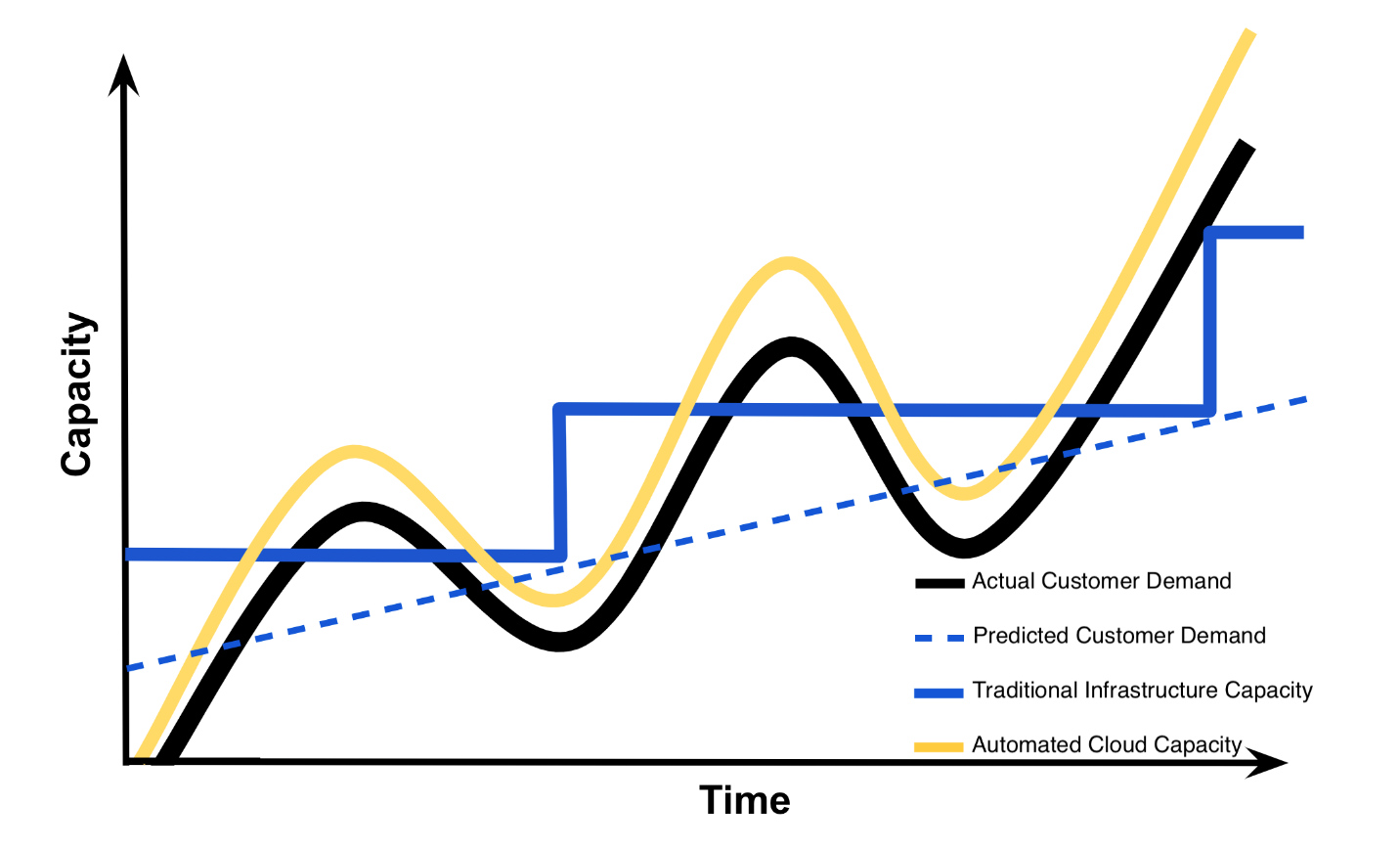


### Why would a business decide to use cloud computing?

Most of the factors related to choosing cloud computing services, instead of developing on-premise IT resources are related to **time** and **cost**. The **capacity utilization** graph below shows how cloud computing compares to traditional infrastructure (on-premise IT resources) in meeting customer demand.

**Capacity** in the graph below can be thought of as the IT resources like: compute capacity, storage, and networking, that's needed to meet customer demand for a business' products and the **costs** associated with those IT resources. In our vacation photos example, customer demand is for storing and sharing customer photos. The IT resources are the required software and hardware that enables photo storage and sharing in the cloud or on-premise (traditional infrastructure).

Looking at the graph, notice that traditional infrastructure doesn't scale when there are spikes in demand, and also leaves excess when preparing for future demand. This ability to easily meet unstable, fluctuating customer demand illustrates many of the benefits of cloud computing.



### Summary of Benefits of Risks Associated with Cloud Computing

The **capacity utilization** graph above was initially used by cloud providers like Amazon to illustrate the **benefits** of cloud computing. Summarized below are the **benefits** of cloud computing that are often what drives businesses to include cloud services in their IT infrastructure [**1**]. These same **benefits** are echoed in those provided by cloud providers Amazon ([**benefits**](https://aws.amazon.com/what-is-cloud-computing/)), Google ([**benefits**](https://cloud.google.com/what-is-cloud-computing/)), and Microsoft ([**benefits**](https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/)).

#### Benefits

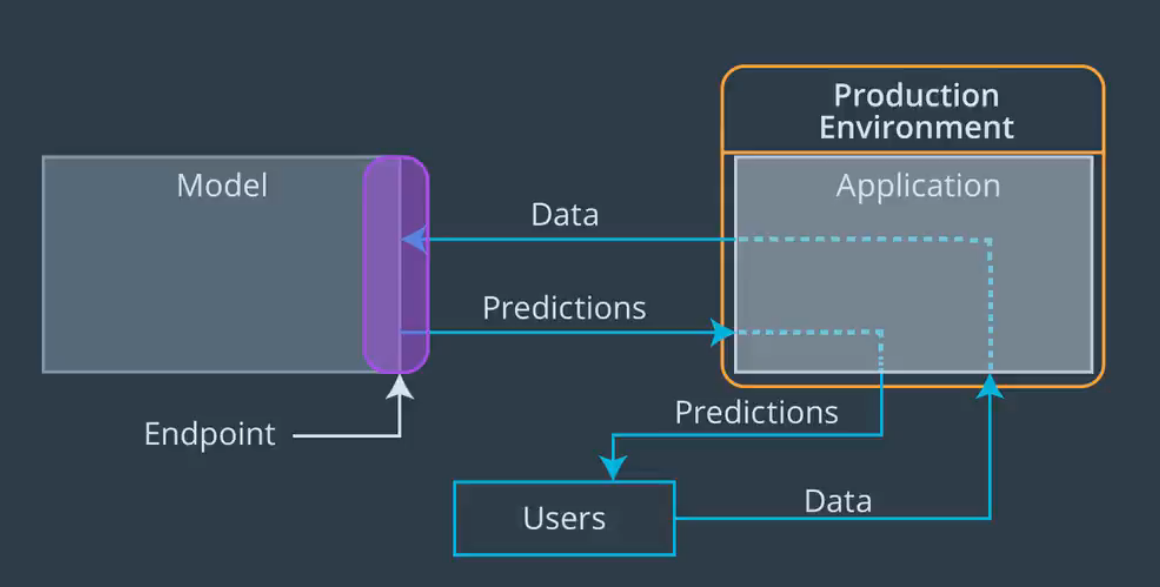
1. Reduced Investments and Proportional Costs (providing cost reduction)
2. Increased Scalability (providing simplified capacity planning)
3. Increased Availability and Reliability (providing organizational agility)

Below we have also summarized the **risks** associated with cloud computing. Cloud providers don't typically highlight the risks assumed when using their cloud services like they do with the benefits, but cloud providers like: Amazon ([**security**](https://aws.amazon.com/security/introduction-to-cloud-security/)), Google ([**security**](https://cloud.google.com/security/data-safety/)), and Microsoft ([**security**](https://www.microsoft.com/en-us/TrustCenter/CloudServices/Azure/default.aspx)) often provide details on security of their cloud services. It's up to the cloud user to understand the compliance and legal issues associated with housing data within a cloud provider's data center instead of on-premise. The service level agreements (SLA) provided for a cloud service often highlight security responsibilities of the cloud provider and those assumed by the cloud user.

#### Risks

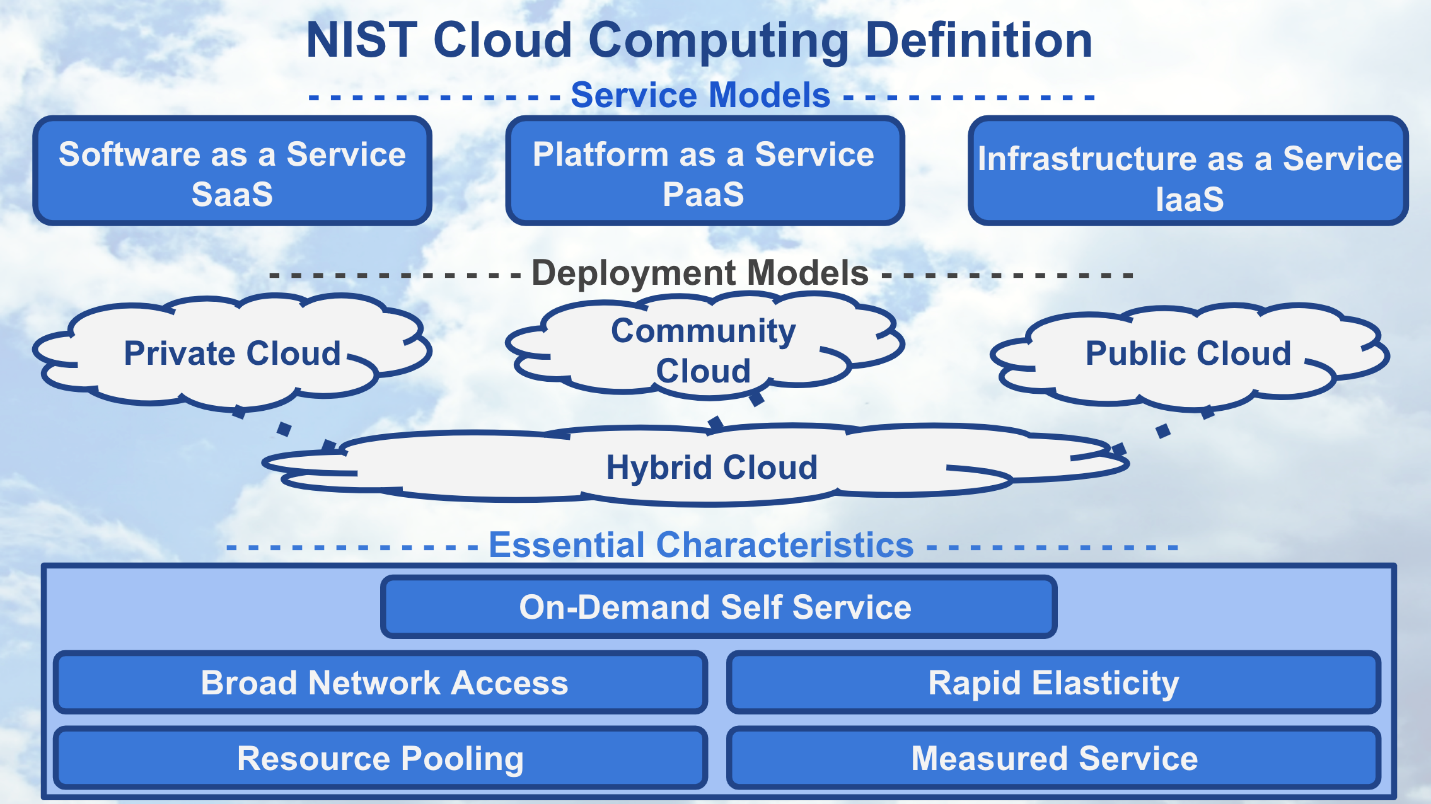
1. (Potential) Increase in Security Vulnerabilities
2. Reduced Operational Governance Control (over cloud resources)
3. Limited Portability Between Cloud Providers
4. Multi-regional Compliance and Legal Issues

**Model deployment**



## Paperspace

[**Paperspace**](https://www.paperspace.com/ml) simply provides GPU-backed virtual machines with industry standard software tools and frameworks like: [**TensorFlow**](https://www.tensorflow.org/), [**Keras**](https://keras.io/), [**Caffe**](http://caffe.berkeleyvision.org/), and [**Torch**](http://torch.ch/) for machine learning, deep learning, and data science. **Paperspace** claims to provide more powerful and less expensive virtual machines than are offered by **AWS**, **GCP**, or **Azure**.



## Shut Down SageMaker Instances, if not in use

#### Note: We recommend you shut down every resource (e.g., SageMaker instances, or any other hosted service) on the AWS cloud immediately after the usage; otherwise, you will be billed even if the resources are not in actual use.

Even if you are in the middle of the project and need to step away, **PLEASE SHUT DOWN YOUR SAGEMAKER INSTANCE**. You can re-instantiate later.

**S3** stands for Simple Storage Service (S3).

**S3** is a virtual storage solution that is mostly meant for data to be written to few times and read from many times. This is, in some sense, the main workhorse for data storage and transfer when using Amazon services. These are similar to file folders that contain data and metadata about that data, such as the data size, date of upload, author, and so on.

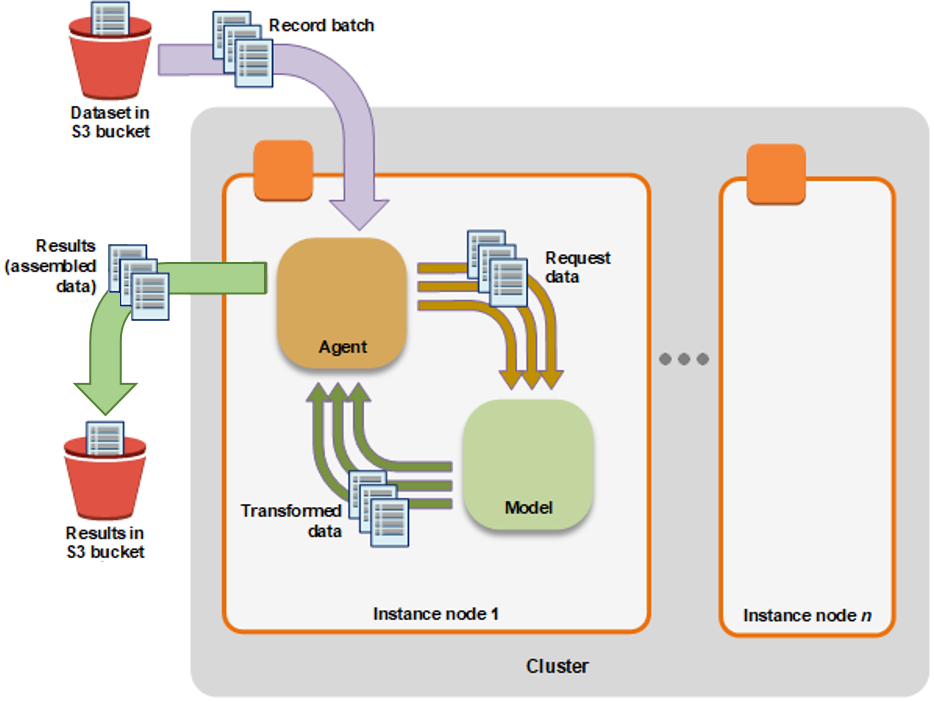
**To get inferences for an entire dataset, use batch transform.** With batch transform, you create a batch transform job using a trained model and the dataset, which must be stored in Amazon S3. Amazon SageMaker saves the inferences in an S3 bucket that you specify when you create the batch transform job. Batch transform manages all of the compute resources required to get inferences. This includes launching instances and deleting them after the batch transform job has completed. Batch transform manages interactions between the data and the model with an object within the instance node called an agent.

Use batch transform when you:

* Want to get inferences for an entire dataset and index them to serve inferences in real time
* Don't need a persistent endpoint that applications (for example, web or mobile apps) can call to get inferences
* Don't need the subsecond latency that SageMaker hosted endpoints provide

You can also use batch transform to preprocess your data before using it to train a new model or generate inferences.

The following diagram shows the workflow of a batch transform job:



To perform a batch transform, create a batch transform job using either the SageMaker console or the API. Provide the following:

* The path to the S3 bucket where you've stored the data that you want to transform.
* The compute resources that you want SageMaker to use for the transform job. *Compute resources* are machine learning (ML) compute instances that are managed by SageMaker.
* The path to the S3 bucket where you want to store the output of the job.
* The name of the SageMaker model that you want to use to create inferences. You must use a model that you have already created either with the [CreateModel](https://docs.aws.amazon.com/sagemaker/latest/APIReference/API_CreateModel.html) operation or the console.

Why not batch transfer during training?

### 什么是docker

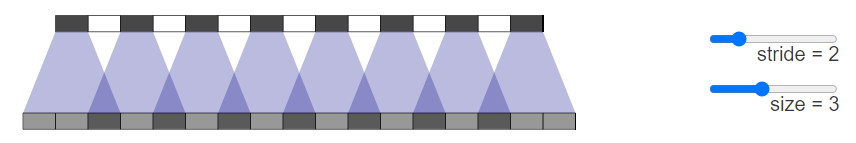
docker是一个开源的应用容器引擎，基于go语言开发并遵循了apache2.0协议开源。docker可以让开发者打包他们的应用以及依赖包到一个轻量级、可移植的容器中，然后发布到任何流行的linux服务器，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口（类iPhone的app），并且容器开销极其低。

棋盘效应？

不要直接用转置卷积，改用upsampling加上convolution可能会解决你的问题

可以看下这个

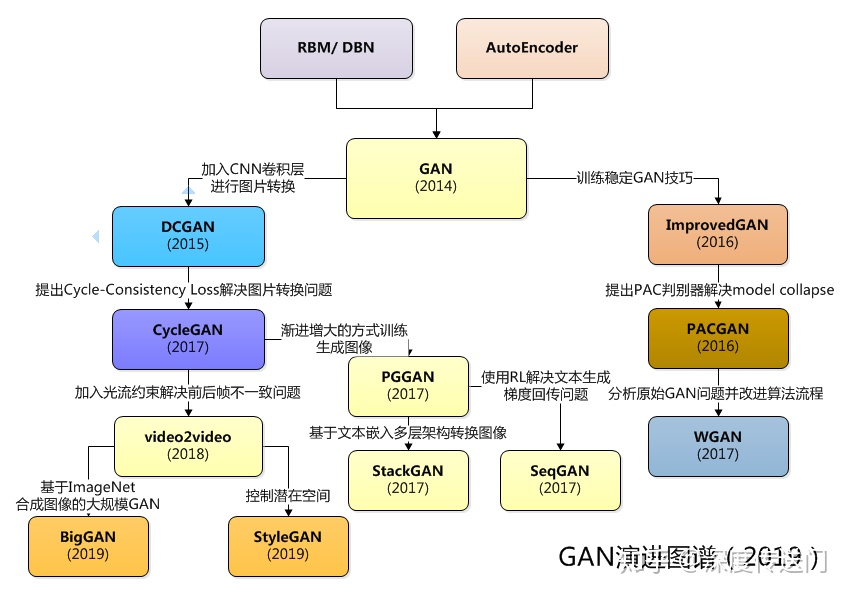
Unfortunately, deconvolution can easily have “uneven overlap,” putting more of the metaphorical paint in some places than others [7]. In particular, deconvolution has uneven overlap when the kernel size (the output window size) is not divisible by the stride (the spacing between points on the top). While the network could, in principle, carefully learn weights to avoid this  — as we’ll discuss in more detail later — in practice neural networks struggle to avoid it completely.

The overlap pattern also forms in two dimensions. The uneven overlaps on the two axes multiply together, creating a characteristic checkerboard-like pattern of varying magnitudes.

循环一致性损失其实是一个自编码损失。

常见的cyclegan通常是k层下采样，n层resblock，k层上采样。k的层数通常不会很大。如果可视化过每一层网络层的输出结果，可以知道，前k层可以极大程度上保留其原有的结构信息。后面的resblock具备较强的梯度传播能力，因此在优化过程中，循环一致性损失权重通常比对抗损失高，因此不会因为对抗损失造成中间层输出以及生成图像与原图像结构差异很大的情况。

我做过一些实验，发现如果对抗损失权重过高，就无法保证生成图像的结构一致性。如果训练数据太少，也无法保证结构一致性。



## Vid2Vid

Vid2Vid通过在生成器中加入光流约束，判别器中加入光流信息以及对前景和背景分别建模重点解决了视频转换过程中前后帧图像的不一致性问题。

## PGGAN

PGGAN[9]创造性地提出了以一种渐进增大（Progressive growing）的方式训练GAN，利用逐渐增大的PGGAN网络实现了效果令人惊叹的生成图像。“Progressive Growing” 指的是先训练 4x4 的网络，然后训练 8x8，不断增大，最终达到 1024x1024。这既加快了训练速度，又大大稳定了训练速度，并且生成的图像质量非常高。

### 用GAN实现油画风

例如MIT和IBM沃森联合实验室，发布的名为**AI Portraits Ars**的工具，只要给定一张图像，就能秒变「文艺复兴款」的油画风。



值得一提的是，这不是风格迁移，而是GAN自己画出来的。

不止能拿着照片作画，看着视频也没问题，例如1960年的老电影《惊魂记》。



这项技术里GAN的生成器就像是个画师，而判别器就充当鉴赏家角色，负责识破生成器的画不是人类作品。

生成器里有个潜在空间，里面有千千万万的AI画像，都是从人类画作里，用不同的向量修改而成。

然后，要按着照片生成画像，就把一个向量 (Latent Vector) ，映射到千万AI画像里和照片最接近的一幅。  
链接：https://www.zhihu.com/question/326698388/answer/700515502

介绍完背景之后，下面分享几点我在训练CycleGAN的过程中的感受。

* G/D默认都是用Adam优化器，初始参数默认都是相同的学习率，我听从身边的人的建议尝试调低过D的学习率，以及在前5个epoch freeze住D，后者在domain adaptation方面有些许提升，前者没有，对于生成图像质量两者效果不明显。
* 在CycleGAN源码中计算Ga的loss，也就是这个Generator，和Gb是同等权重的。但是在CyCADA的代码中是. 对于两个方向的Discriminator则都是同等权重的。我个人认为正向的S->T的生成器较大的loss会帮助我们获得想要的fake\_B（在A->B的cylcegan中，realA即左侧图通过Ga获得fakeB即右侧图）

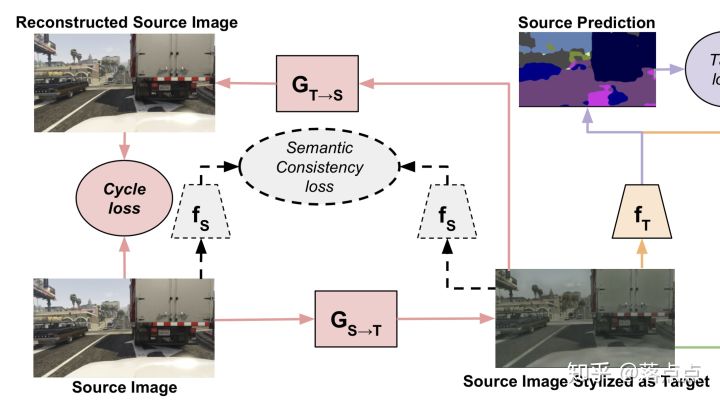
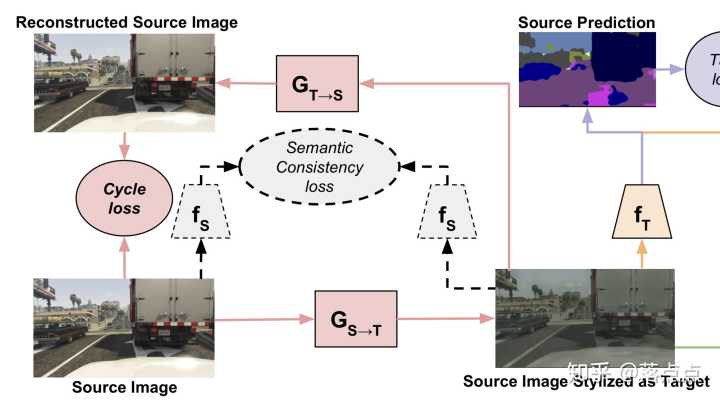
# GAN loss D\_A(G\_A(A))

self.loss\_G\_A = 2 \* self.criterionGAN(self.netD\_A(self.fake\_B), True)

# GAN loss D\_B(G\_B(B))

self.loss\_G\_B = self.criterionGAN(self.netD\_B(self.fake\_A), True)

* 有回答说道关于resize小图的问题，至少在我们从GTA->CityScapes的过程中，我们发现scale width或者是resize的方式，要比crop更好，因为如果我们resize到1024，然后再crop 400x400，也就是大家常用的配置，这种情况是不如我们scale width到600~800，然后再crop到400x400，甚至不如我们直接scale width到400的效果好。这其中有一点问题就是语义信息，对于大图crop之后可能会获得大量的misalignment的patches，大致意思就是我可能会crop出来一个草地，然后对应另外一个域的天空，虽然从统计意义上更matter的是两个域本身的在高维空间的分布，但是这可能算是一个纹理信息和语义的trade off，至少我们采用后者获得了更好的图片质量。
* 另外则是semantic的问题，也即是上个点涉及到的。CyCADA中提出使用单独的网络来监督CycleGAN训练过程中出现的语义不一致的问题，它提出了自己的Semantic Consistency Loss，**实实在在的取得了很好的效果！**即下图所示，我们可以采用这种方式获取更好的transfer的图片。



* 如果训练效果一直不好，可以尝试加入identity loss，CycleGAN论文中有提到，代码也有不过默认是关闭的。这个部分似乎会让训练变得更难收敛，在做domain adaptation这件事情上没有太好的收益，但是图像迁移的质量确实有所提升。
* 最后难以训练，这点是确切的，通常我们需要在GTA上训练20-30个epoch才能获得一个比较好的transfer的G模型，而也听一个跟作者熟悉的bdd大佬说过作者当时从想出idea做实验做通花了快一年的时间.

后面图片生成要去归一化

以前我在同时训练cycle gan 和dual gan 时发现cycle gan 生成的图片质量确实不如dual gan ，当时觉得很奇怪，毕竟这两个模型用的时一样的idea ，只是backbone network 不一样， 一个是用residule block ,一个是用U-net . 后来才发现是反卷积这块有问题。

[https://distill.pub/2016/deconv-checkerboard/​distill.pub](https://link.zhihu.com/?target=https%3A//distill.pub/2016/deconv-checkerboard/" \t "_blank)

cycle gan 是用转置卷积，会有问题，参考上面的blog，用upsampling + conv 或者采用U-net 的skip connection架构会缓和这种问题。具体的你可以看MUNIT 的代码。

1 . 训练之前一定要看自己的数据是否满足高度的结构一致性, 举个例子:

苹果 <=> 橘子: 都是球形, OK!  
苹果 <=> 香蕉: Mode Collapse!

2 . 想要获得高清大图的结果, 不要 resize 成小图训练. 可以crop 成小 patch 训练, 由于 G 是全卷积的, inference 时输入原图尺寸即可获得大图结果.

可以把bn换成instance norm. 然后，可以加一下domain identity loss

最后，记得上采样用可以替代的卷积来处理

训练CycleGAN要有耐心！

我训练的时候用了下面几个trick：

1. 学习率别太高！
2. 对抗损失权重不要太高，循环一致性损失权重为1的时候，对抗损失我一般设置为0.1
3. 判别器优化频率高于生成器
4. 使用最小二乘损失

How to make a model better?

Better model structure, more data, more features as input(like RNN, LSTM)

Has the market structure changed? To which direction will it change? How much will it changes?

conda create --no-default-packages -n myenv python=3.7