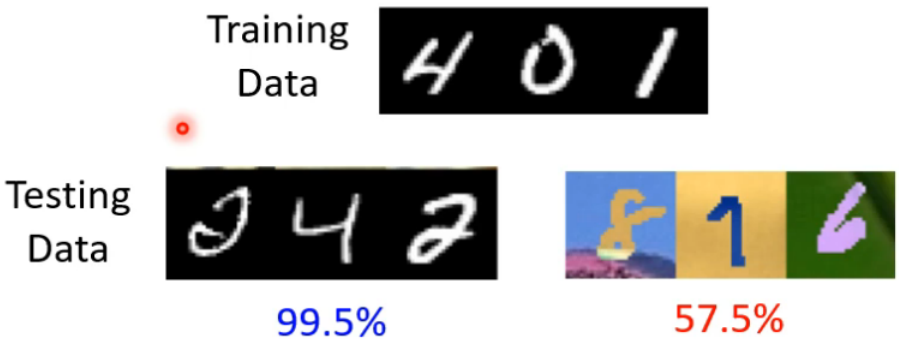
Lecture 12：领域自适应（Domain Adaptation）概述

> Lectured by HUNG-YI LEE (李宏毅)
>
> Recorded by Yusheng zhao（yszhao0717@gmail.com）

在完成一个分类器的训练的过程中，会发生训练资料和测试资料差异过大的现象；从而导致未知资料上的泛化误差过大。



这个问题叫做**Domain Shift**: Training and testing data have different distributions

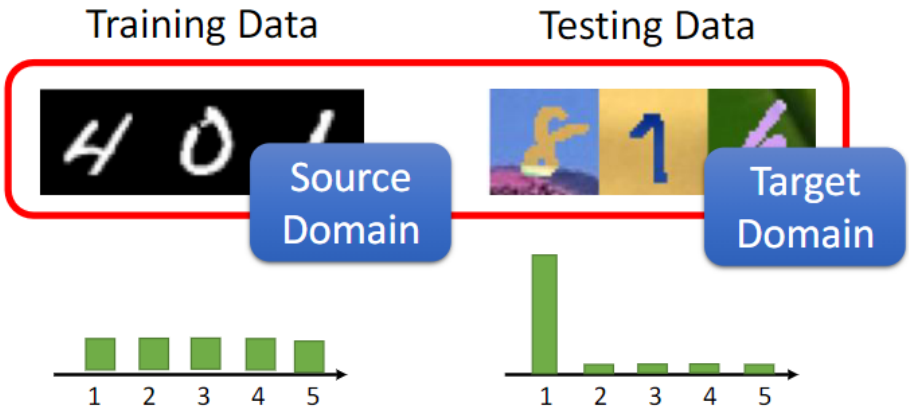
为了克服/削弱Domain Shift，我们提出了**Domain Adaptation**这个技术（也可以看作是迁移学习transfer learning的一种/环节）

> A任务上学到的技能可以用在B任务上
>
> 所谓**Domain Adaptation**：就是训练集上一个domain，测试集上另一个domain，你要把前者的domain学到的资讯用到另一个domain上

Domain Shift

两者可能性：

- 如上所说的，**输入资料**的分布不一致
- **输出**的分布有可能有变化

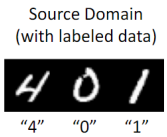
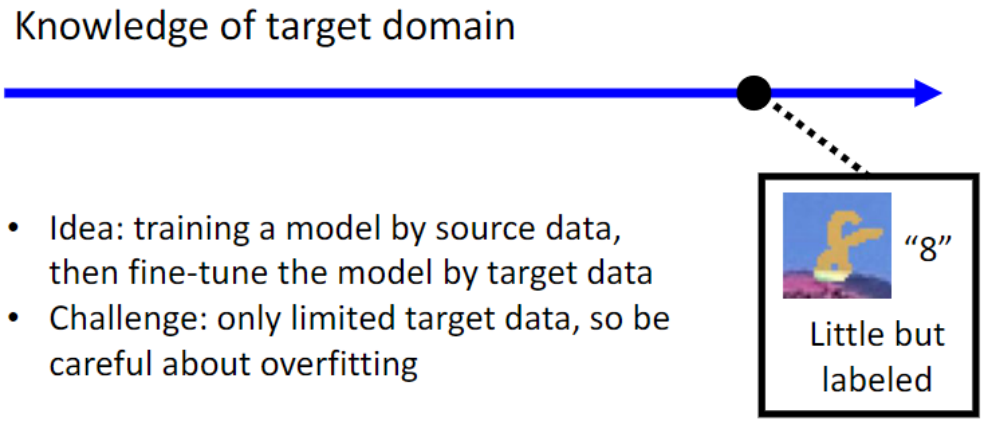
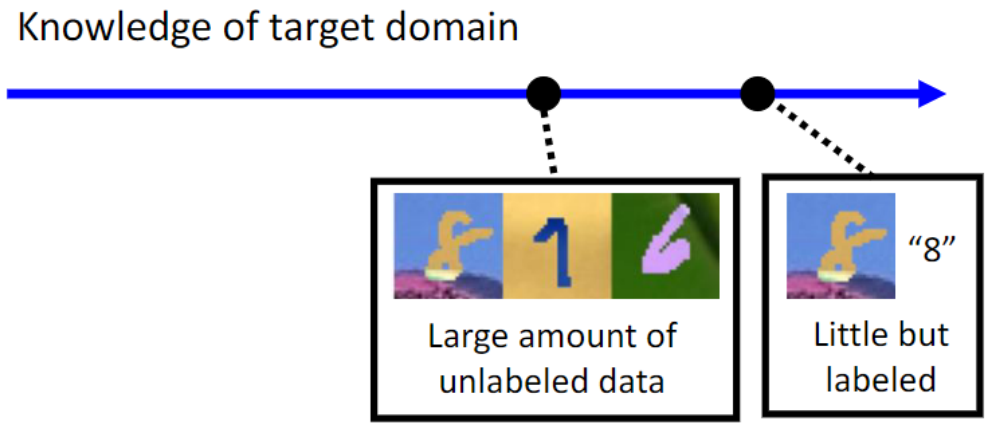


- 更罕见的一种：输入和输出的分布是一致的，但是“认知”（测试集和训练集的关系）变了

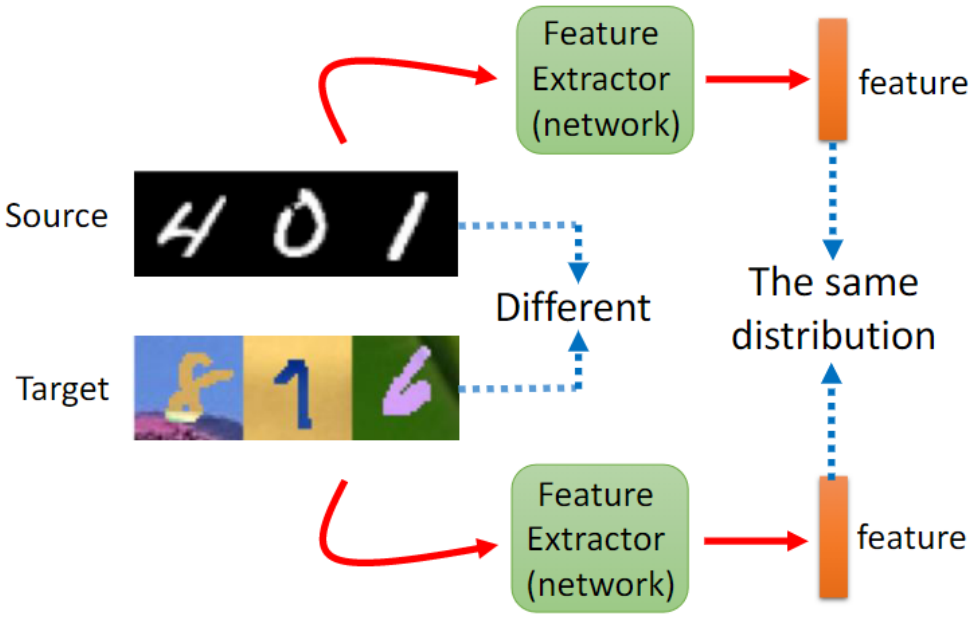
对同一个图案，训练集觉得是“0”，测试集上认知为“1”

以下的内容我们默认训练集来自Source Domain，测试集来自Target Domain

Domain Adaptation

> 情景描述如下：
>
> 我们有一堆训练资料，来自Source Domain，且资料是有标注的（labeled）
>
> 
>
>
> 为了把在训练资料上得到的domain用在测试资料上，我们必须要对测试资料上的即target domain有一些了解——随着了解程度不同，我们有不同的Domain Adaptation的方法。
>
> - 如果Target Domain上大部分资料被标注了，那就不需要做Domain Adaptation，直接在这个资料上面train就好了。
>
> 
>
>
> - 如上图，这个需要做Domain Adaptation的情形基本就是Target Domain有标注资料，但数量少；在这种情况下（比较容易处理）：用Target Domain上少量的标注资料去微调（fine-tune）Source Domain上train出来的模型（稍微多跑两三个epoch）。另外还要注意到不要在Target Domain上过拟合（注意不要过多的iteration）
>   - 关于削弱过拟合的方法：调节learning rate、让fine-tune前和fine-tune后的参数不要差太多、或者规定输入输出的关系不要差太多
> - **（重点）**Target Domain有大量的资料是没有标注的。这个情景是比较符合真实场景。
>
> 
>

第三个情景的Basic Idea：

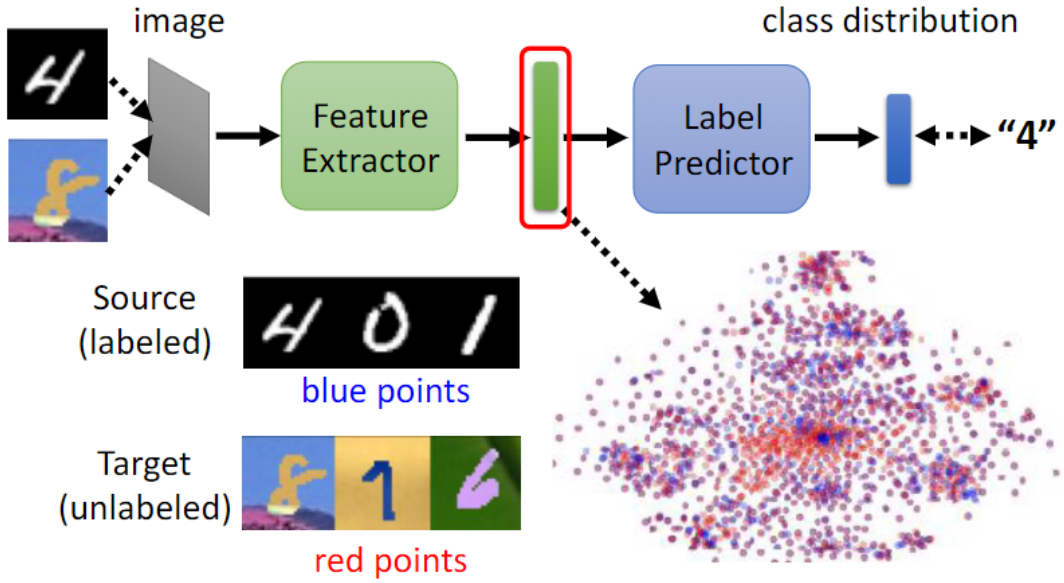


我们想要找一个Feature Extractor，这个也是一个network，吃一张图片为输入，输出一个vector（feature）。虽然Source Domain和Target Domain表面上看起来不一样，而Feature Extractor作用就是丢掉不一样的部分，保留两个domain相似的部分。以上图为例，Feature Extractor需要学会*忽视颜色（ignore colors）*，即把颜色的资讯滤掉。然后，我们就可以在Source Domain上用feature训练一个模型，就可以直接用在Target Domain上。

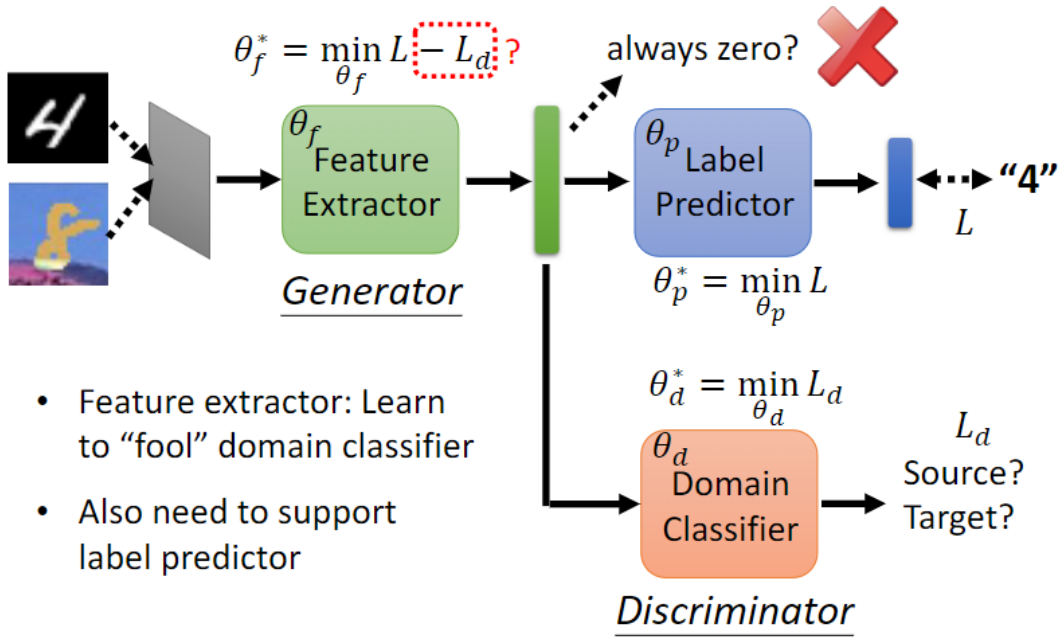
Domain Adversarial Training

找到Feature Extractor的方法：（假s设一个classifier有10层）——最basic的想法

- 把这个classifier分成两部分：Feature Extractor & Label Predictor（两个都是network）。至于怎么分就是个超参数
- 对于Source Domain由于大部分是标注的，所以和训练一般的分类器一样
- 对于Target Domain，由于大部分数据是未标注的，我们需要把这些unlabeled data丢进这个Feature Extractor，把它的output拿出来看其分布，我们的目标就是让这个分布和上一条Feature Extractor的输出分布没有差异



- 要做到上图中分布一致，需要用到Domain Adversarial Training的技术。训练一个Domain的classifier（二元的分类器），输入一个factor，输出判断这个factor是来自哪个Domain。而Feature Extractor的目标就是想办法"骗"过这个Domain Classifier。~~——嗯？？这尼玛不是GAN嘛？~~



和GAN的区别：FE（Generator）的输出还会受到Label Predictor的限制，所以不会输出零向量。

- 明确目标：一方面Source Domain的数据集（labeled）可以算出交叉熵，定出loss。三个网络任务分别是

$$\theta_f^* = \min_{\theta_f} L - L_d \quad (1)$$
$$\theta_p^* = \min_{\theta_p} L$$
$$\theta_d^* = \min_{\theta_d} L_d$$

对于Label Predictor图像分类的越正确越好，对于Domain Classifier就是Domain分类的越正确越好；对于Feature Extractor，其任务是背刺Domain Classifier，让feature难以分辨。

注意到Domain Classifier起到辅助训练的作用，我们需要的是提炼feature的Feature Extractor

- 事实上，这个$\theta_f^* = \min_{\theta_f} L - L_d$是有缺陷的，仅仅让FE做DC相反的事情，最极致的情况：把Source和target Domain反过来（让DC的loss最大），但这依然分出来feature，背离了设计的初衷。~~思考题：怎么做可以做得更好？~~

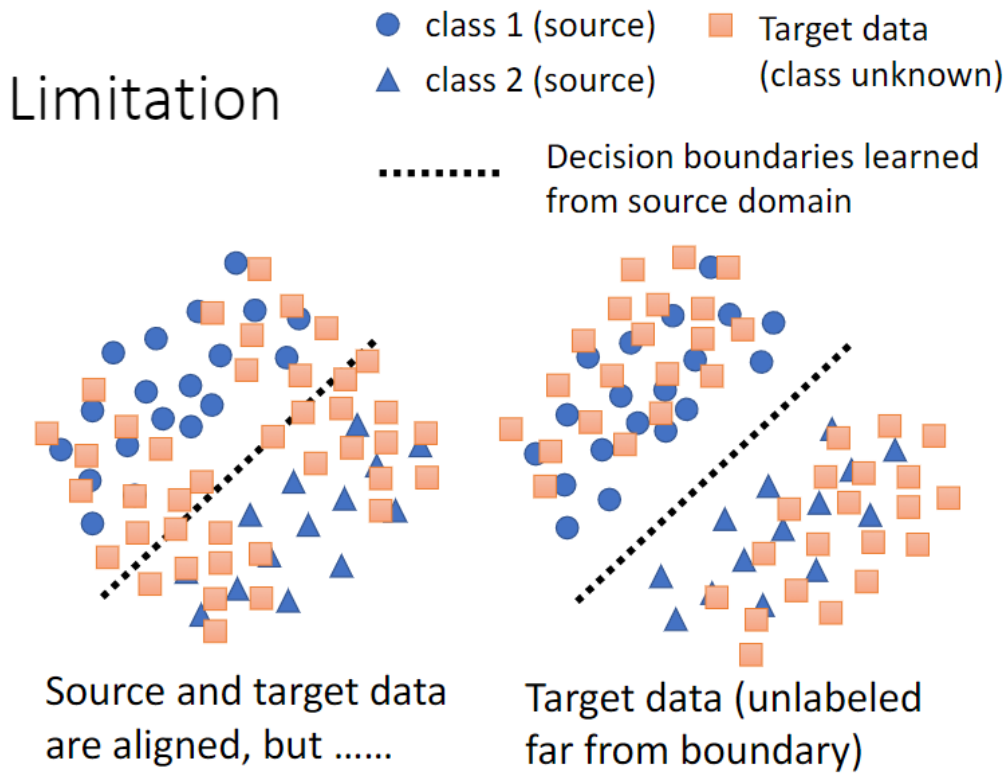
结果：

Yaroslav Ganin, Victor Lempitsky, Unsupervised Domain Adaptation by Backpropagation, ICML, 2015

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Domain-Adversarial Training of Neural Networks, JMLR, 2016

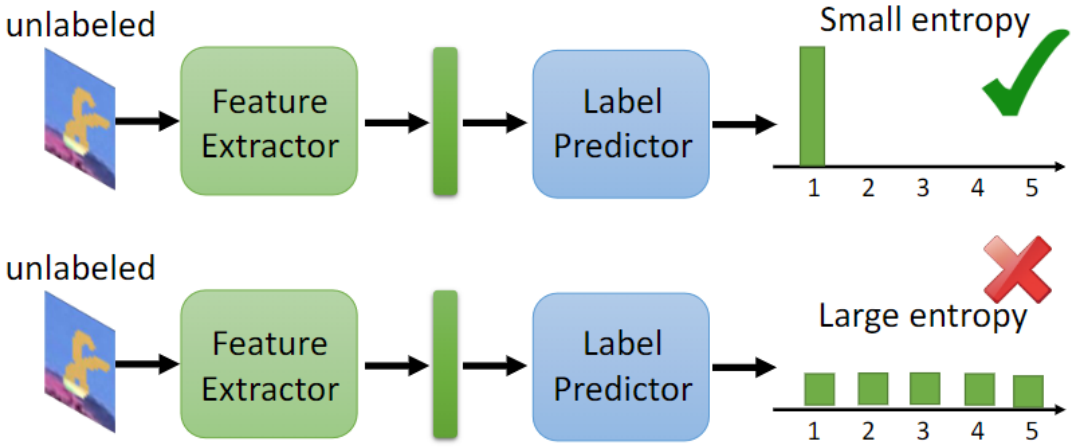
METHOD	SOURCE / TARGET	MNIST / MNIST-M	SYN NUMBERS / SVHN	SVHN / MNIST	SYN SIGNS / GTSRB
SOURCE ONLY		.5749	.8665	.5919	.7400
TRAIN ON TARGET		.9891	.9244	.9951	.9987

Limitation



我们更希望右边的状况而避免左边这个.....怎么做👇

- 一个可能的想法：在这个boundary上（算是一个hyperplane？？），有一些边界上的point（有点像support vector），我们要让方形远离这些分界点。



如果输出的结果非常集中：离boundary远。

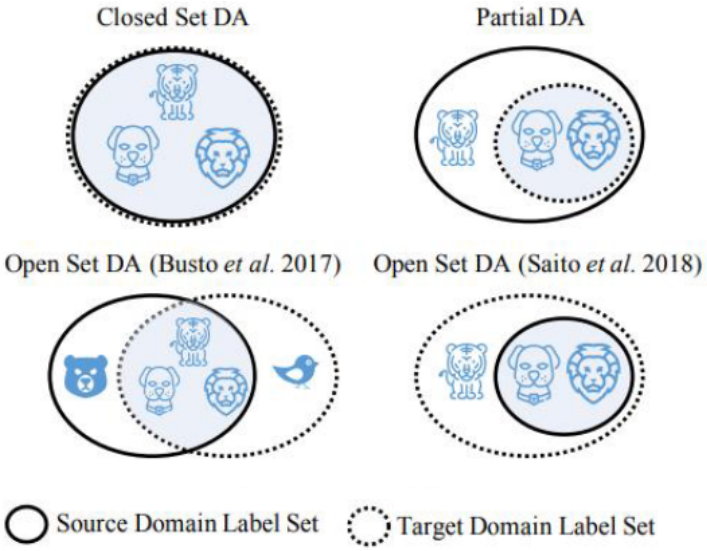
其他的一些方法：

Used in Decision-boundary Iterative Refinement Training with a Teacher (DIRT-T)

Maximum Classifier Discrepancy

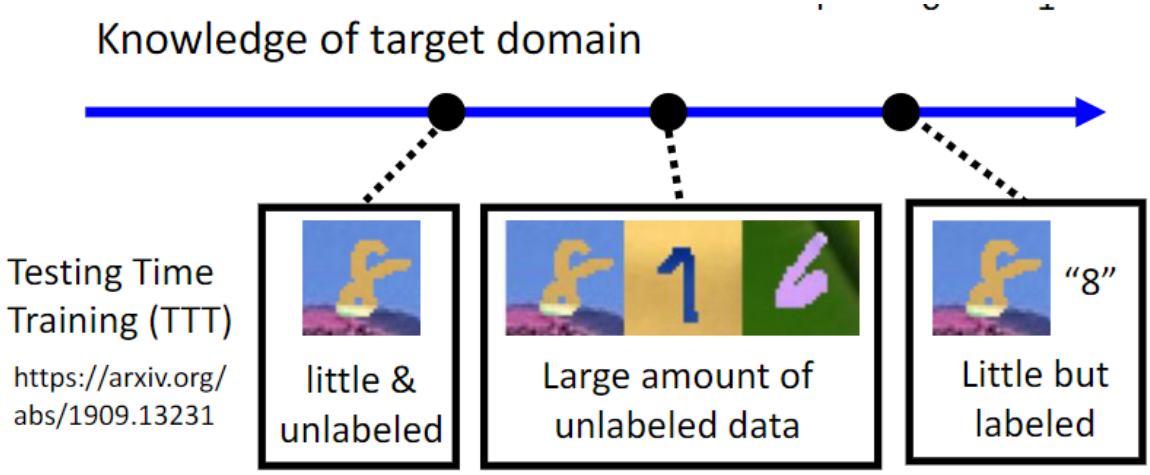
Outlook

- 以上我们都假设Source Domain和Target Domain类别都是一模一样的，实际上可能并不是这样的（以下图4中可能额）



关于这个问题，我们可以参见Universal domain adaptation这篇文章。

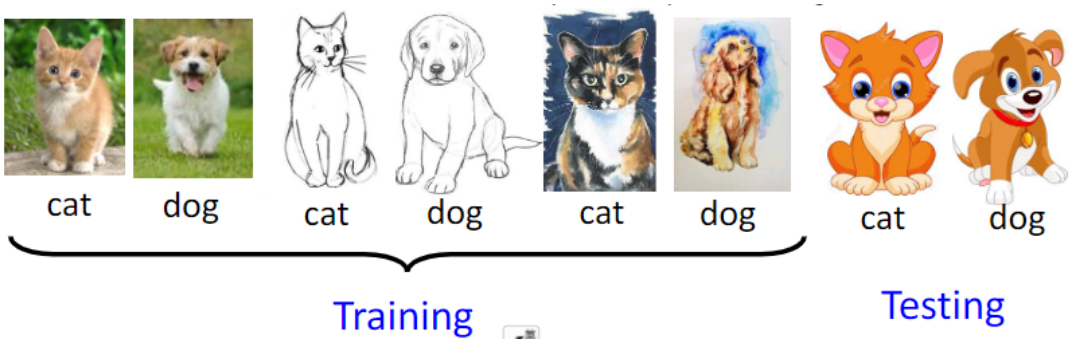
Domain Adaptation另外几种情景



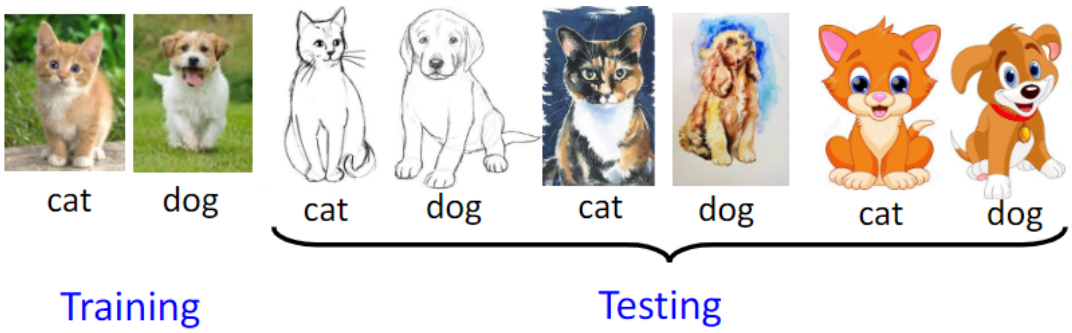
你的Target Domain不仅没有label而且量少，如此将Target和Source Domain去align一起非常困难。一种解决这个情景的方法就是Testing Time Training

更严峻的情况——对Target Domain一无所知；这时候我们的任务称之为Domain Generalization。分两种情况：

- 训练资料分成丰富，包含了各式各样的不同的Domain，做到了领域泛化，模型可以磨平不同domain的差异。文章：https://ieeexplore.ieee.org/document/8578664s



- 训练资料贫瘠（可能就一种domain），而测试资料是其他多种不同的domain



尝试的做法：https://arxiv.org/abs/2003.13216。有点像Data Augmentation，去生成多个domain资料，然后套第一种情况的做法