

## 27 Domain Adaptation

到目前为止,我们已经训练了很多,Machine Learning 的 Model,所以对大家来说,训练一个 Classifier 完全不是一个问题

所以假设要你训练一个数字的 Classifier,你只要有给你训练资料,你训练好一个模型,然后 Apply 在用在测试资料上就结束了,那像数字辨识这麼简单的问题,你在 Benchmark Corpus,在MNIST Benchmark Corpus 上,随便做一做,可能都会做到 99.5% 的正确率

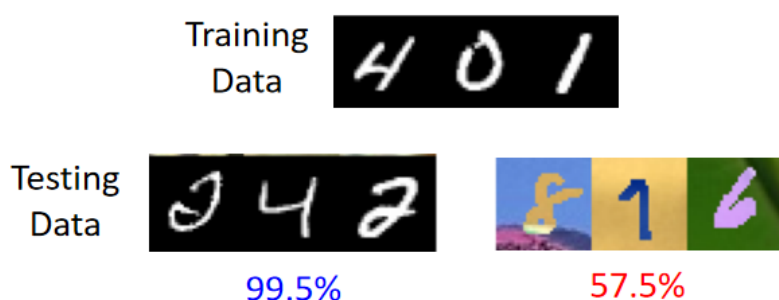
但是假设今天测试资料,跟训练资料的分布不一样,怎么办呢

我们举一个简单的例子,假设训练的时候,你的数字是黑白的,但测试的时候,你的数字是彩色的,会发生什么样的事情呢

你可能会觉得说,一边是黑白的 一边是,虽然一边是黑白的 一边是彩色的,但是对 Model 来说,数字的形状是一样的,它能够在黑白的图片上认出数字来,在彩色的图片上,会不会应该也可以认出数字来呢

但实际上不是,如果你今天在这样子黑白的数字上面,训练一个模型,直接用到彩色的数字上,你得到的正确率会非常地低,会低到只有 57%,不能算是一个及格的分

You have learned a lot about ML. Training a classifier is not a big deal for you. 😊



The results are from: <http://proceedings.mlr.press/v37/ganin15.pdf>

Domain shift: Training and testing data have different distributions.



Domain adaptation

Transfer learning: <https://youtu.be/qD6iD4TFsdQ>

所以我们今天知道说,一旦训练资料跟测试资料,它中间有一些差异,它们中间的分布是不一样的,你在训练资料上训练出来的模型,在测试资料上面可能就会坏掉,那这种问题 叫做 Domain Shift,也就是当你的训练资料跟测试资料,它的分布有些不同的时候,这种状况叫做 Domain Shift

在多数的这种作业裡面,或者是 Benchmark Corpus 裡面,我们都假设无视 Domain Shift 这个问题,我们的训练资料跟测试资料,往往有著一样的分布,那这样都会给大家一个错误的印象就是,哇 这个今天的什麼人工智慧,真的是很厉害,都超越人类啦,在很多任务上面都有极高的正确率,但是实际上用在真实的应用上,当你的训练资料跟测试资料的中间,有一点差异的时候,机器能不能够做得好,就是一个未知数了

那我们今天就是要来讲说,假设训练资料跟测试资料,有一点差异的时候,有没有什麼方法可以让我们,能够做得比什麼都不做结果还要好

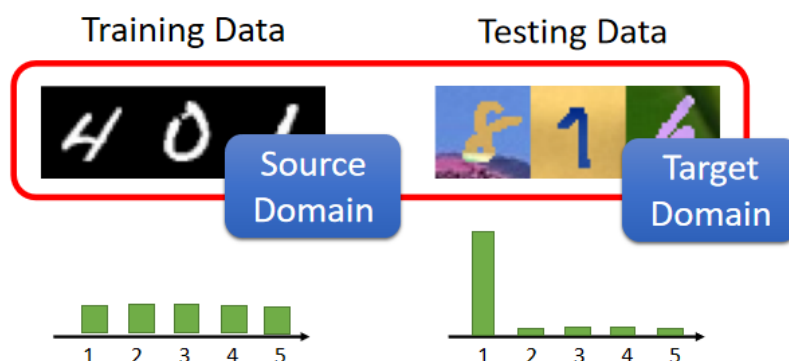
那今天就是要讲 Domain Adaptation 的技术,那 Domain Adaptation 的技术,也可以看做是 Transfer Learning 的一种

Transfer Learning 就是,你在 A 任务上学到的技能,可以被用在 B 任务上,那对于 Domain Adaptation 来说,你的训练资料是一个 Domain,你的测试资料是另外一个 Domain,你在训练资料上面,某一个 Domain 上学到的资讯,你要把它用到另外一个 Domain,用到测试资料上面,所以你是把一个 Domain 学到的知识,用在另外一个 Domain 上,所以它可以看做是,Transfer Learning 的其中一个环节,那在过去上课的录影裡面,有完整的讲了 Transfer Learning 相关的技术,那因为今天时间有限,我们就只 Focus 在,Domain Adaptation 的部分就好,如果你有兴趣的话,你可以再看一下过去上课的录影

## Domain Shift

Domain Shift,其实有很多**多种不同的类型**,我们刚才看到的,只是 Domain Shift 的其中一种可能,是模型输入的资料分布有变化的状况,那输入分布有变化是一种可能性

其实还有另外一种可能性是,**输出的分布也可能有变化**,举例来说 在你的训练资料上面,可能每一个数字它出现的机率都是一样的,但是在测试资料上面,可能每一个输出的机率是不一样的,有可能某一个数字它输出的机率特别大,有没有可能有这种事情发生呢,这也是有可能的



那这也是一种 Domain Shift,还有一种更罕 比较罕见,但也不是完全不可能发生的状况是,**输入跟输出虽然分布可能是一样的,但它们之间的关系变了**



也许在你的训练资料裡面,这种东西叫做 0,但是在你的测试资料裡面,这种东西叫做 1,这也不是不可能的,也是有可能发生这种状况嘛,也是有可能发生说,输入跟输出它们的关系不一样,在训练跟测试资料不一样的状况,那这又是另外一种 Domain Shift

那我们今天呢 只专注在,输入资料不同的 Domain Shift 的上面,好 那在等一下的课程裡面,这个**测试的资料**,我们说它来自 **Target Domain** **训练的资料**,我们说它来自 **Source Domain**,所以 **Source Domain** 是我们的训练资料,Target Domain 是我们的测试资料

## Domain Adaptation

在 Domain Adaptation 裡面,我们的这个情境是这个样子的,我们有一堆训练资料,那这边我们就直接拿手写数字辨识,来当作我们的例子啦

我们有一堆训练资料,那这些资料来自 Source Domain,而且这些训练资料是有标注的,我们知道每张图片对应的数字是什麽,但是我们希望再用这些资料,训练出一个模型,这个模型可以用在不一样的 Domain 上

## Source Domain (with labeled data)



"4" "0" "1"

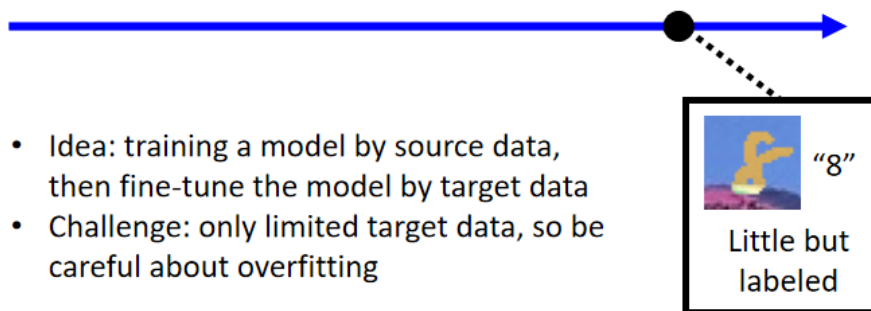
那要把这个模型用在不一样的 Domain 上,在训练的时候,我们就必须要对另外一个 Domain,也就是现在测试资料所在的 Target Domain,有一些了解,那随著了解的程度不同,我们就有不同的 Domain Adaptation 的方法

- 那了解最多的是,假设我们在 Target Domain 上,我们有一点资料,而且这些资料居然还有 Label,那这是一种情况
- 但还有另外一种更好的状况是,也许你根本在 Target Domain 上,就有一大堆的资料,那些资料也都有 Label,那你其实就不需要做 Domain Adaptation,你直接拿 Target Domain 的资料来训练就好了

所以如果你在 Target Domain 上,已经有一大堆的资料,而且它们还有标注,那你不需要做 Domain Adaptation

那要做 Domain Adaptation 的情境可能是,你有 Target Domain 的资料 也有标注,但是量非常地少,那在这种状况下怎么办呢

### Knowledge of target domain



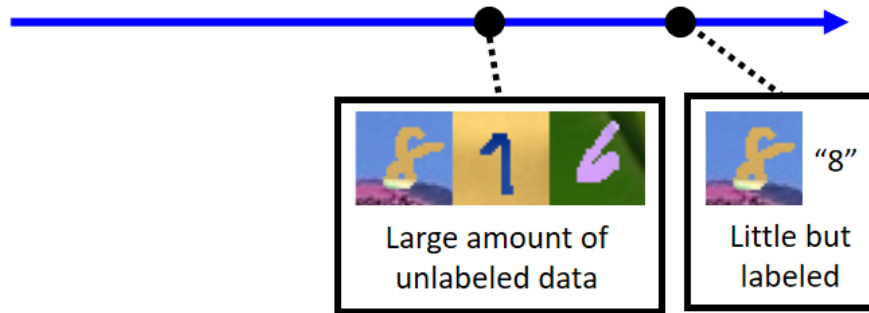
那这种状况还算是 Domain Adaptation 裡面,比较容易处理的状况,如果你今天遇到的是,有标注资料只是资料量很少的情况下,你可以用这些有标注的资料,来微调你在 Source Domain 上训练出来的模型,那这边所谓的微调,就跟你在做 BERT 的时候的行为很像,就是你已经有一个在 Source Domain 上,训练好的 Model,那你拿 Target Domain 的 Data,只稍微跑个两三个 (Epoch) 就足够了

那在这一种情境下,你需要注意的问题就是,因为你的 Target Domain 的资料量非常少,所以你要小心 不要 Overfit,也就是说 你不要在 Target Domain 上的资料上,跑太多的 Iteration,那如果你跑太多的 Iteration,可能会 Overfit 到 Target 的这些少量的资料上,然后呢 你在你真正的 Testing Set 上就做不好,这是有可能的

那为了避免 Overfitting 的情况,过去就有很多的 Solution,比如说 把 Learning Rate 调小一点,举例来说你要让 (fine tune) 前,跟 (fine tune) 后的模型的参数,不要差很多,或者是让 (fine tune) 前,跟 (fine tune) 后的模型,它的输入跟输出的关系,不要差很多 等等,那有很多不同的方法,那这边呢 我们就不细讲

那今天主要想要跟大家分享的情境,也是我们作业要处理的情境是,我们在 Target Domain 上有大量的资料,但是这些资料是没有标注的,你的 Target Domain 是有颜色的数字,你也蒐集到了一大堆有颜色的数字的图片,但是没有人标注说,每一张图片裡面的数字是什麽

## Knowledge of target domain



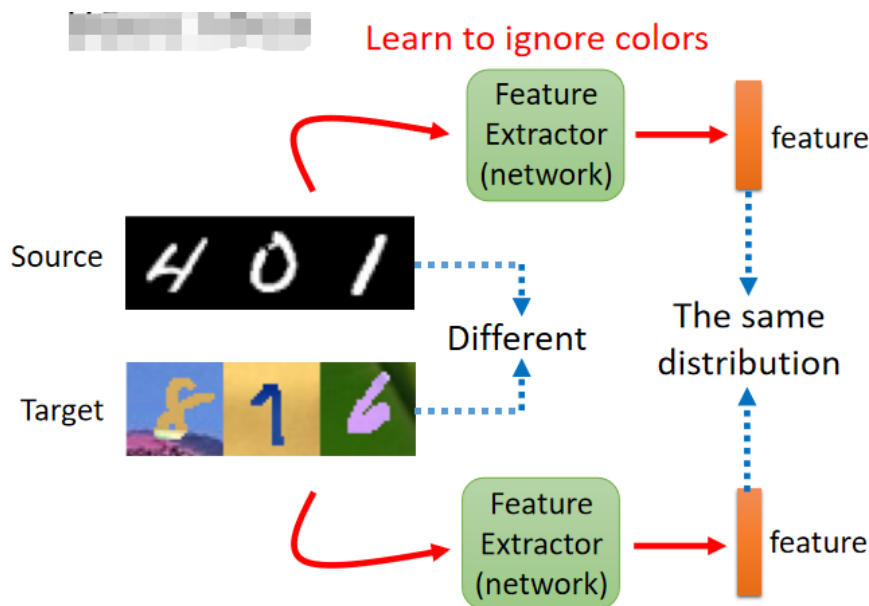
那我们在作业裡面要处理的就是,遇到这种状况的时候,到底应该要怎麼解呢,好 到底应该要怎麼解呢

那像这种情境,其实是蛮符合你在真实的系统上,有可能会发生的情境,举例来说 你在实验室裡面训练了一个模型,你想要把它用在真实的场域裡面,你就把你的模型上线,那确实有一些人来用,但是你发现你得到的 Feedback 很差,大家都嫌弃你的系统正确率很低,那怎麽办,但是你这个时候,你也许就可以用 Domain Adaptation 的技术,那因为你的系统已经上线,也真的有人使用,所以你可以蒐集到一大堆的资料,只是这些资料它们是没有标注的

那现在要问的问题是,怎麽用这些没有标注的资料,来帮助我们在 Source Domain 上,训练出一个模型,它可以用在 Targe Domain 上呢

## Basic Idea

那这边最 Basic 的想法是这个样子的,这边基本的概念是这个样子,我们想要找一个 Feature Extractor

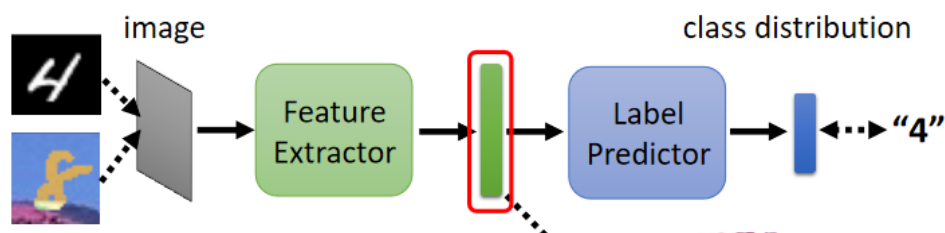


这个 Feature Extractor,它其实也是一个 Network,这个 Network 呢 吃一张图片作为输入,它吐出一个 vector,吐出一个 Feature,虽然 Source Domain 跟 Target Domain,它们的 Image 表面上看起来不一样,但是 Feature Extractor,会把它们不一样的部分拿掉,只抽取它们共同的部分

所以虽然从图片上看起来,这两组图片一个有颜色 一个没有颜色,它本来就很不一样,但是我们期待说,这个 Feature Extractor 可以学到,就无视颜色这件事情,把颜色的资讯滤掉,那今天不管是来自 Source Domain 的图片,还是来自 Target Domain 的图片,只要通过这个 Feature Extractor 以后,它得到的 Feature 看起来是没有差异的,它们看起来有一样的分布,那这样你就可以用这些 Feature,训练一个模型,在 Source Domain 上训练一个模型,直接用在 Target Domain 上,那接下来的问题就是,怎麽找出这样一个 Feature Extractor

# Domain Adversarial Training

怎麼找出这样的一个 Feature Extractor 呢,那其实我们可以把一个一般的 Classifier,就分成 Feature Extractor,跟 Label Predictor 两个部分



我们知道一个 Image 的 Classifier,就是输入一张 Image Output,就是分类的结果,那假设这个 Image 的 Classifier 有 10 层,那我们就说,前 5 层算是 Feature Extractor,后 5 层算是 Label Predictor,因为前 5 层,你一个 Image 通过前 5 层,它输出就是一个 vector 嘛,那如果你上 CNN 的话,它输出其实是 Feature Map 啦,但 Feature Map 拉直,也可以看做是一个 vector 嘛,那这个 vector,再丢到 Label Predictor 的后面 5 层,它会產生 Class,那所以我们可以把前 5 层,看做是 Feature Extractor

那你可能会问说,那為什麼是前 5 层呢,為什麼不是前 4 层 前 3 层 前 2 层 前 1 层呢,可以是前 4 前 3 前 2 前 1,这个是你自己决定的,一个 Classifier 裡面,哪些部分算 Feature Extractor,哪些部分算是 Label Predictor,这个是你自己决定的,那这个也算是一个 Hyper Parameter 啦,就跟 Network 架构要调一下一样,要调一样

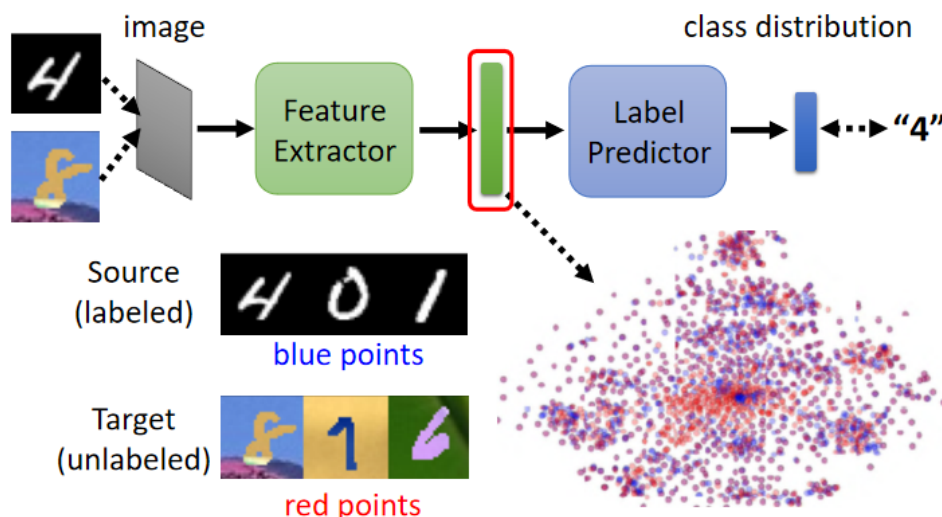
那如果你今天呢,用 Domain Adaptation 的方法,那这边等一下要用的方法叫,Domain Adversarial Training

用 Domain Adversarial Training,你要把 Classifier 裡面的哪个部分 哪几层,当做 Feature Extractor,这个也要问你自己,那这个也是你自己要决定的

那我们现在要怎麼来训练这个,Feature Extractor 跟 Label Predictor 呢,今天對於 Source Domain 上的资料,Source Domain 上的资料是有标注的,我们就期待把 Source Domain 的资料丢进去,那就去跟训练一个一般的分类器一样,它通过 Feature Extractor,再通过 Label Predictor,可以產生正确的答案

但不一样的地方是,Target Domain 的这些资料,我们有一堆 Target Domain 的资料,但这些资料是没有任何的标注的,这些资料是没有任何的标注的,所以我们不能说把这些资料丢进去以后,期待 Label Predictor 会 Output 什麼数字,因为我们根本不知道 Label Predictor,要 Output 什麼数字才是对的

但是这些资料可以怎麼被使用呢,这些资料的使用方式就是,我们把这些图片丢到这个 Image,丢进这个 Image Classifier,然后我们把,Feature Extractor 的 Output 拿出来看,拿出来看以后,我们希望 Source Domain 的图片,丢进去的 Feature,跟 Target Domain 的图片丢进去的 Feature,它们看起来要分不出差异

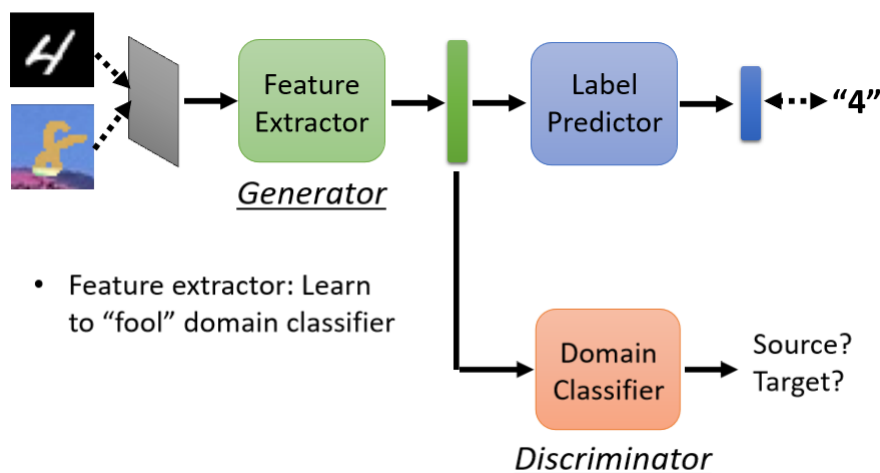




就是这个 Source Domain 的图片,我们用它的 Feature,我们用蓝色的点来表示,Target Domain 的图片,它的 Feature 我们用红色的点来表示,我们要这些蓝色的点跟这些红色的点,分不出差异,那怎麼让蓝色的点跟红色的点,分不出差异呢,那这个就要藉由,Domain Adversarial Training 的技术

那我们现在要做的事情就是,训练一个 Domain 的 Classifier,这个 Domain 的 Classifier,它就是一个**二元的分类器**,它吃这个 vector 当作输入,它要做的事情就是判断说,**这个 vector 是来自於 Source Domain,还是来自於 Target Domain**

而 Feature Extractor 它学习的目标,就是要去想办法骗过这个 Domain Classifier,那听到骗过这件事情

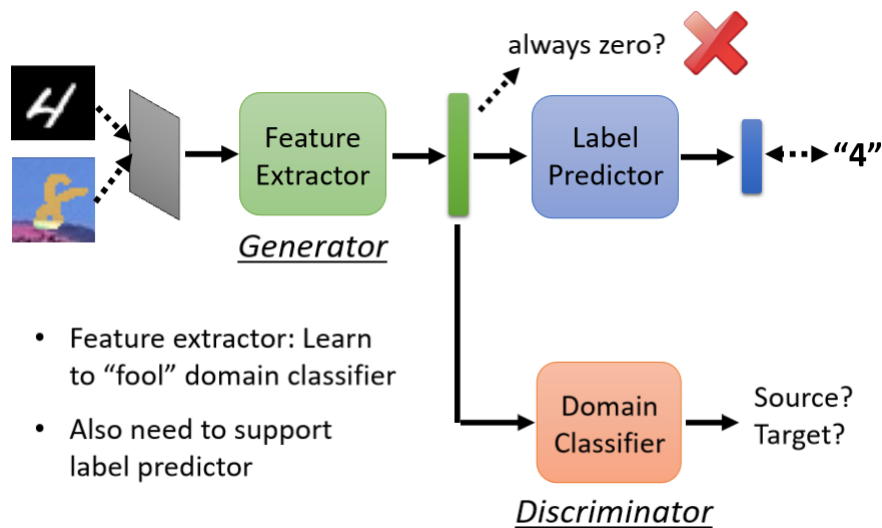


是不是让你脑中就浮现了,Gan 这个东西,是不是就浮现了,Generative Adversaria Network 这个东西呢,没错 Domain Adversarial Training,就非常像是 Gan,你可以把 Feature Extractor,想成是 Generator,把 Domain Classifier,想成是 Discriminator,那其实 Domain Adversarial Training,最早的 Paper,我记得是发表在 2015 年的 ICML 上面,比那个 Gan 还要稍微晚一点点啦,不过它们几乎可以说是同时期的作品,在 Domain Adversarial Training 那篇 Paper 裡面,是有引用到 Gan 那篇 Paper,但那时候 Gan 那篇 Paper,还没有上[NeurIPS](#),所以它只说,欸 有一篇 有另外一篇 Paper,它提了一个叫 Gan 的想法,然后它是 Technical report 放在网路上的,还跟我的想法有点像,所以它们算是一个同时期的作品

但是讲到这边,这个 Domain Adversarial Training,跟 Gan 还是有一点不一样,那在这个游戏裡面,对 **Generator 好像优势太大了**,那对 Generator 来说,它要骗过 Discriminator,完全不需要花什麼力气,有一个非常无脑的做法,就是你的 Feature Extractor,也就是你的 **Generator**,不管看到什麼输入,永远都输出 0 就好了

看到什麼输入我都输出一个 Zero vector,那对 Domain Classifier 来说,它完全不知道 Input 的 Image 是什麼,它永远都看到 Zero vector,它就完全无法分辨这个 vector,来自於哪一个 Domain,但是这显然不是我们要的状况,如果 Feature Extractor 只会输出 Zero vector,那这样子等於根本就什麼事都没有做是一样的

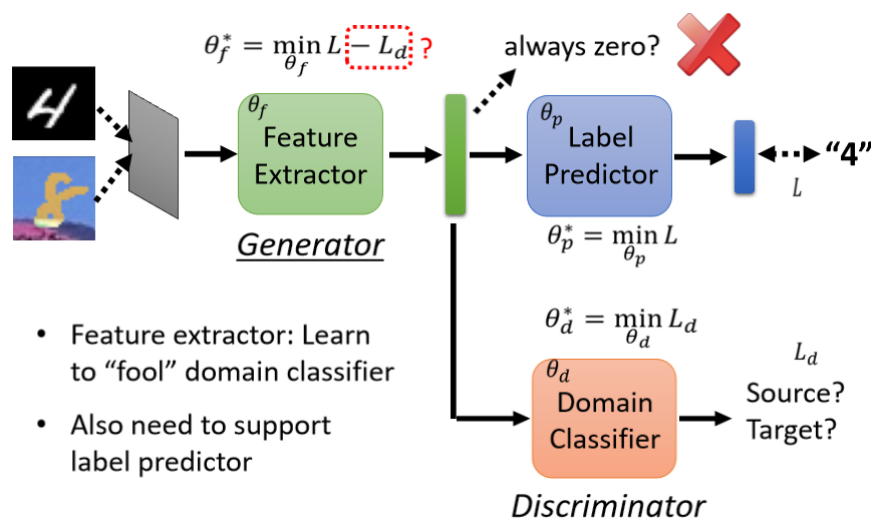
那这件事情会发生吗,其实**这件事情是不会发生的**,為什麼,因為 **Label Predictor 也需要这个 Feature**



Label Predictor 它也需要这个 Feature,让它可以去判断,输入的图片属于哪一个类别,所以假设 Generator 它就直接放一个大绝说,今天不管输入什么样的 Image,我输出都是 Zero vector,那对于 Label Predictor 来说,它就没有办法判断是哪一张图片,那在这个情况下,因为 Feature Extractor,它还是需要产生这个 vector,让 Label Predictor 可以产生正确的图片,所以 Feature Extractor 它就不能放大绝,它就不能看到什么东西,永远都输出 Zero vector

那这边呢,我们用符号再稍微把我们刚才讲过的事情,再重新说 说得更清楚一点

- 假设 Label Predictor 的参数,我们就叫它  $\theta_p$
- Domain Classifier 的参数 叫做  $\theta_d$
- 然后 Feature Extractor 的参数 叫做  $\theta_f$



然后呢 这个 Source Domain 上的这些 Image,它的 Classification 的这个 Cross Entropy,就 Source Domain 这些 Image,它是有 Label 的,所以你可以算它们的 Cross Entropy,你根据它们的 Cross Entropy,订出一个 Loss

这边是有一个  $L$ ,  $L$  是这个 Domain 的这个,这个 Source Domain 上的那些 Image,它有 Label,你可以算出 Cross Entropy,然后对于这个 Domain 的 Classifier 而言,它要去想办法分辨,Source 跟 Target Domain 的差距,它要去做一个, Binary 的 Classification 的问题,它要去做一个二元分类的问题,这个分类的问题有一个 Loss 叫做  $L_d$

那我们现在要去找一个  $\theta_p$ ,它可以让这个  $L$  越小越好

我们要去找一个  $\theta_d$ ,它可以让这个  $L_d$  越小越好

你说 Label Predictor 它要做的事情,就是让这个 Source Domain 的 Image,分类越正确越好,Domain Classifier 要做的事情,就是让 Domain 的分类越正确越好

而 Feature Extractor 呢, Feature Extractor 它要做的事情是,它站在 Label Predictor 这边,然后呢 它要去捅 Domain Classifier 一刀,它要去做 Domain Classifier 相反的事情,所以这个 Feature Extractor,它的 Loss 是 Label Predictor 的 Loss  $L$ ,去减掉 Domain Classifier 的 Loss,叫做  $L_d$

所以 Feature Extractor 它的 Loss,就是大  $L$  减掉  $L_d$ ,你要去找一个参数 找一组参数  $\theta_f$ ,它可以让大  $L$  减  $L_d$  的值越小越好

这个是最原始的, Domain Adversarial Training 的做法,但这真的是最好的做法吗,你可以想想看喔,那这个详情我们就不细讲,这个留给大家自己思考 自己发觉,你想想看喔,假设 Domain Classifier 它的工作,是要把 Source Domain 跟 Target Domain 分开,是要看到这种图片,它知道它的 Feature 来自 Source Domain,看到这种图片,它知道它的 Feature 来自 Target Domain,它要把这两组 Feature 分开



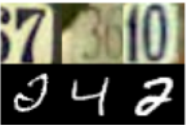


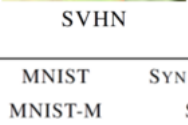
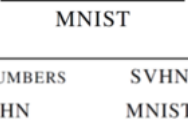

而 **Feature Extractor 如果它的 Loss 是 Domain Classifier 直接加一个负号**,那意味著什麼,意味著说它要做的事情,就是跟 Domain Classifier 相反,本来 Domain Classifier 看到这张图片的 Feature,它说是 Source,那现在 Feature Extractor,是要让 Domain Classifier 看到这个图片以后,它说是 Target,看到这个图片 反过来要是 Source,如果你这么做,不是也把两组 Feature 分开来了吗

我们说我们现在要做的事情,是要让 Domain 跟,这个 Source 跟 Target Domain 没有差别,但是你今天不管是用  $L_d$ ,还是负的  $L_d$ ,其实都是要把 Source 跟 Target Domain 分开呀,你去让  $L_d$  的值,本来 Domain Classifier,是要让  $L_d$  的值越小越好,你现在是负的  $L_d$ , Feature Extractor 要让  $L_d$  的值越大越好,其实也是把 Source 跟 Target Domain 分开,所以这未必是最好的做法,至於怎麽做,当然这招是有用的,这招是有用的,那但是怎麽做可以做得更好,欸 这个留给大家慢慢思考

好 那我们来看一下, Domain Adversarial Training, 最原始的 Paper, 它做的结果怎麽样呢,当年看到这个 Paper 的时候,真的觉得结果非常地惊人

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

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

那这边呢 它做了四个任务,那上半部 是 Source Domain 的图片,这边其实都是数字辨识啦,那下半部呢 都是 Target Domain 的图片,好 如果今天呢,我们是拿 Target Domain 的图片来做 Training, Target Domain 的图片来做 Testing,那结果像是这个样子,每一个任务正确率都是 90% 以上,但如果说,我们今天是 Source Domain Training, Target Domain Testing, Train 在黑白的数字上,测试在彩色的数字上,结果直接惨掉,哇 这直接惨掉 没办法做啦,结果直接惨掉

那如果加上 Domain Adversarial Training 的话,结果怎麽样呢,你会发现说本来如果只 Train 在黑白的图片上,测试在彩色的图片上,正确率 57.5

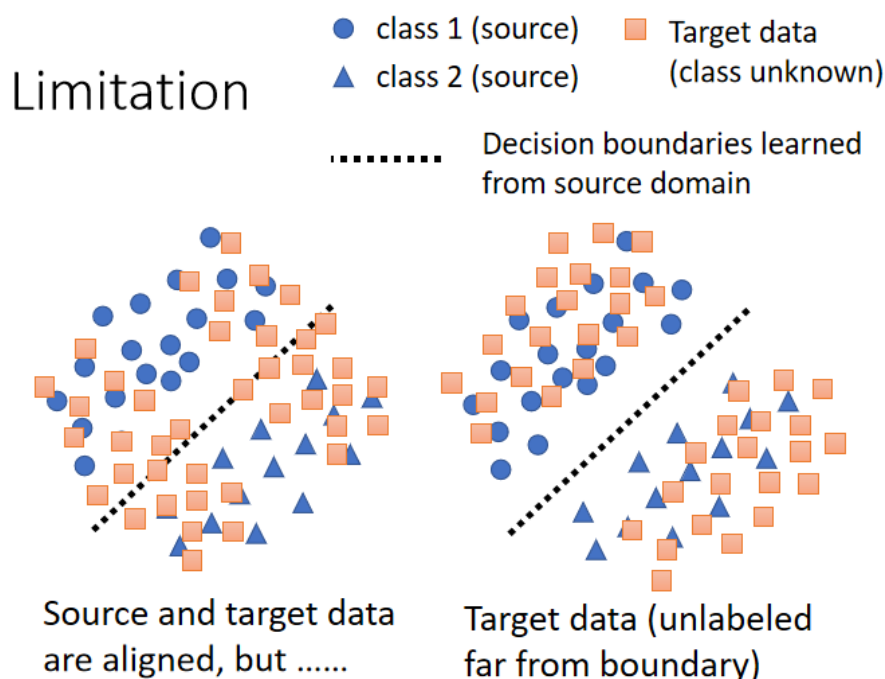


METHOD	SOURCE	MNIST	SYN NUMBERS	SVHN	SYN SIGNS
	TARGET	MNIST-M	SVHN	MNIST	GTSRB
SOURCE ONLY		.5749	.8665	.5919	.7400
PROPOSED APPROACH		<b>.8149</b> (57.9%)	<b>.9048</b> (66.1%)	<b>.7107</b> (29.3%)	<b>.8866</b> (56.7%)
TRAIN ON TARGET		.9891	.9244	.9951	.9987

那如果今天有做 Domain Adversarial Training,正确率就飙升到 81%,在很多其他任务上进步量都挺明显的,59 到 71,74 到 88.7,这个进步量都是挺明显的,那这个就是 Domain Adversarial Training

## Limitation

那刚才这整套想法,还是有一个限制,有一个小小的问题,什么样小小的问题呢



我们来看看哦,今天 蓝色的圈圈跟蓝色的三角形,代表 Source Domain 上的两个 Class,那我们当然可以找一个 Boundary,去把这两组 Class 把它分开来,对于 Target Domain 上的 Data,我们没有任何的 Class 的 Label,我们就只能说所有 Target Domain 的 Data,我们都用这个正方形来表示它

那我们今天训练的目标,就是要让这些正方形它的分布,跟这个圈圈三角形合起来的分布越接近越好,但是什麼叫做越接近越好呢,左边这个 Case,红色的点跟蓝色的点,它们也算是蛮 Align 在一起的,也算是分布蛮接近的,右边这个 Case,红色的点跟蓝色的点,它们也算是分布蛮接近的,但是你觉得左边比较好,还是右边比较好呢,

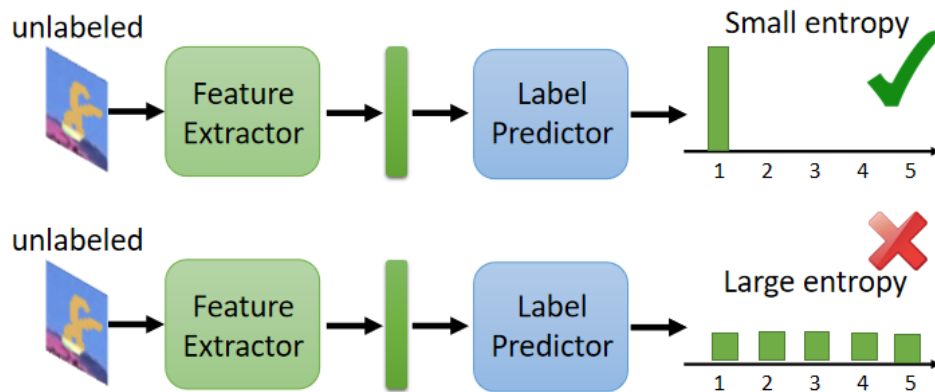
没错 很多人也觉得是右边,所以我们是不是**应该要让右边的状况发生**,而避免让左边的状况发生呢,好 怎麽做呢,怎麽让右边的状况发生,左边的状况不要发生呢

也许一个可能的想法是,我们既然知道蓝色的圈圈跟蓝色的三角形,它们的**分界点**在哪裡,这个分界点我们是知道的,那我们应该要让这些方形,虽然我们不知道它是哪一个类别,但我们让这些方形远离这一个分界点,怎麽让方形远离这个分界点呢,那在文献上就有很多不同的做法啦,你可以参考一下文献,看看你觉得怎麽做比较好

## Considering Decision Boundary

举例来说一个最简单的做法是说,呃 今天我有很多 Unlabeled 的图片,丢到 Feature Extractor,再丢到 Label Predictor 以后,我不知道它是哪一个类别,但是我希望它离 **Boundary 越远越好**,那什麼叫做离 Boundary 越远越好呢

- 如果今天输出的结果非常地集中,叫做离 Boundary 远
- 如果今天输出的结果每一个类别都非常地接近,叫做离 Boundary 近



Used in Decision-boundary Iterative Refinement Training with a Teacher (DIRT-T) <https://arxiv.org/abs/1802.08735>

Maximum Classifier Discrepancy <https://arxiv.org/abs/1712.02560>

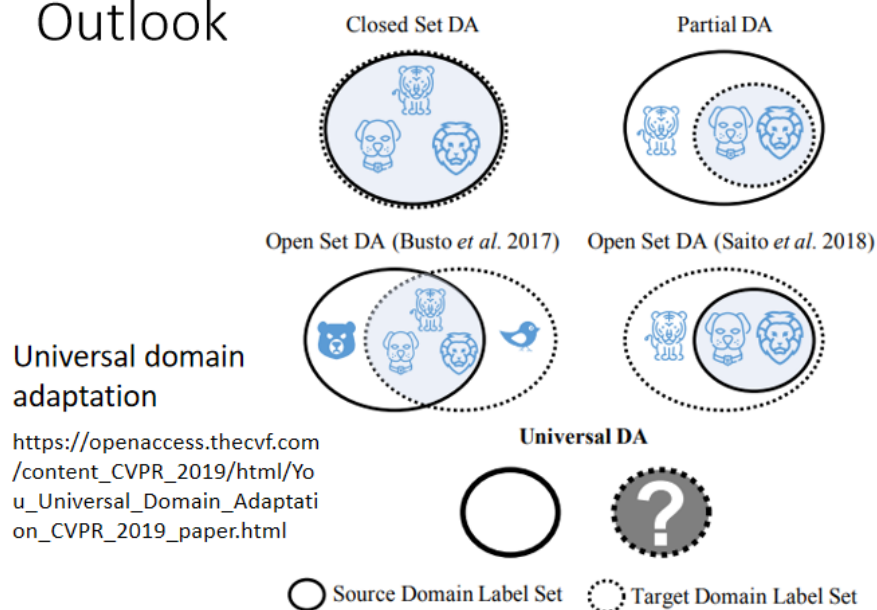
所以我希望说把 unlabeled 的 Image,丢进 Feature Extractor,再丢进 Label Predictor,输出的结果,它离 Boundary 越远越好,也就是说这集中在某一个类别上,我们虽然不知道它应该算是哪一个类别,但至少应该集中在某一个类别上

那这只是一种招数,并不是全部,那你可以参考一下文献,比如说有一个知名的方法叫做 DIRT-T,DIRT-T,这个 DIRT-T 它其实,它 Paper 裡面还特别告诉你说,这个要唸 Dirty,要唸 Dirty,这个大家都是这个模型命名大师,都会命名一些很有创意的名字,好 这个 DIRT-T 是一个招数,还有另外一个招数叫这个,Maximum Classifier Discrepancy,如果你要在这个 Domain Adaptation 座位裡面,得到最好的结果的话,那这些招数是不可或缺的,那实际上这些招数怎麽进行,还挺复杂,这个就留给大家自己研究

那这边还有一个问题,什麽样的问题呢,我们到目前為止,好像都假设说,Source Domain 跟 Target Domain,它的类别都要是一模一样,Source Domain 假设是影像分类的问题,Source Domain 有老虎 狮子跟狗,Target Domain 也应该要有老虎 狮子跟狗,但是真的一定会这样吗

Target Domain 是没有 Label 的,我们根本不知道 Target Domain 裡面,有什麽样的类别

## Outlook



而在这个图示裡面 这个实心的,实线的圈圈代表,Source Domain 裡面有的东西,这个虚实线的圈圈,代表 Target Domain 裡面有的东西,所以呢 有没有可能是,这个 Source Domain 裡面的东西比较多,Target Domain 裡面的东西比较少呢,有没有可能是,Source Domain 裡面的东西比较少,Target Domain 的东西比较多呢,有没有可能两者虽然有交集,但是各自都有独特的类别呢,这都是有可能发生的

所以在这个前提之下,你说 Source Domain 跟 Target Domain,你硬要把它们完全 Align 在一起,听起来有点问题呀,因为举例来说在这个 Case 裡面,哦 你说你要让 Source Domain 的 Data,跟 Target Domain 的 Data,它们的 Feature 完全 Match 在一起,那意味著说,你硬是要让老虎去变得跟狗像,或者是老虎硬是要变得跟狮子像,到时候你就分不出老虎这个类别了

听起来就是有问题的方法,那怎麽解决这个问题,怎麽解决 Source Domain 跟 Target Domain,它可能有不一样的 Label 的问题,那你可以参见这个,Universal Domain Adaptation 这篇文章

好 那我们来看看有没有同学有问题要问的,好 有同学问说

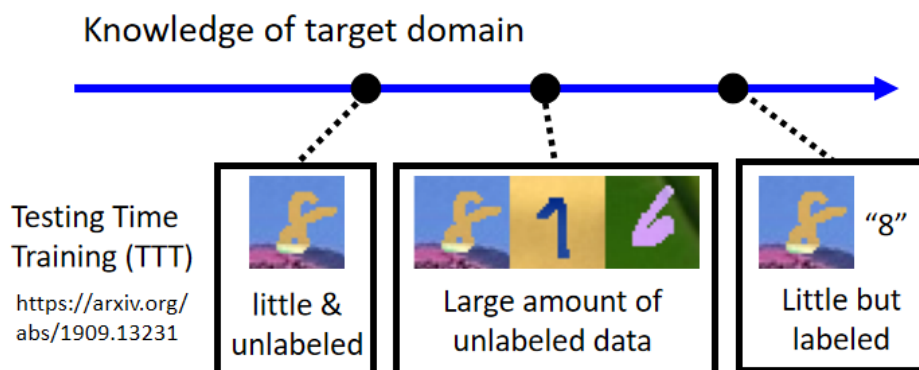
**如果 Feature Extractor 是 CNN,而不是 Linear Layer,那 Domain Classifier Input,就是 Feature Map,拉直的 Latent Embedding,这样 Latent Space 学到的东西,把两个 Domain 分部弥平会不会有影响,因为 Feature Map 本来就有 Space 的关系,现在却硬是被拉直**

答:

你说得非常对,真的 你说得非常对,就是 Feature Extractor,它是一个复杂的 Network,然后我们硬是要,把两个 Domain 的东西拉在一起,会不会变成它只是為了拉在一起而拉在一起,它根本就没有学到,我们本来希望,这个 Feature Space 学到的东西呢,简单的回答就是 会,会,所以 Domain Adaptation,没有大家想像得那麽容易 Train 起来,虽然好像刚才讲得都非常地顺利,那作业裡面可以自己体验一下啦,这个 Domain Adaptation,算是偏难做的一个作业,所以 呃 这一个,你知道我们在 Train 的时候,我们有两件事情,有两件事互相结抗,一个是要去骗过 Domain Classifier,另外一个是要让分类变正确,那我们期待说这两件事情都可以同时做好,也就是说一方面既骗过 Domain Classifier,一方面又分类分得好,那就同时把两个 Domain Align 在一起,同时 Latent Space,我们又希望它的分布是正确的,比如说我们觉得 1 跟 7 比较像,那他们為了要让 Classifier 做好,那今天你的 Feature Extractor,就会让 1 跟 7 比较像,然后 1 跟,比如说 1 跟 4 比较不像,它就让 1 跟 4 拉得比较远一点,我们期待说,藉由需要把 Label Predictor 的 Performance,冲高这件事情,latent representation 裡面的这个 Space,仍然是保留一个比较好的 Latent Space,但是不一定 这件事不一定总是会成功了,如果你今天你给 Domain Classifier,就是要骗过 Domain Classifier,这件事情的权重太大,你的 Model 就会学到说,它都只想骗过 Domain Classifier,它就不会產生好的 Latent Space,所以刚才同学问的问题,确实是有可能发生的,所以大家在实做的时候,这个也是有些参数要调的,好好 希望这样有回答到同学的问题

那接下来 还有一个更严峻的状况,刚才我们是假设说没有 Labeled Data,但至少有一大堆,这个时候你还可以说,我要把两个 Space 呢 把它拉在一起

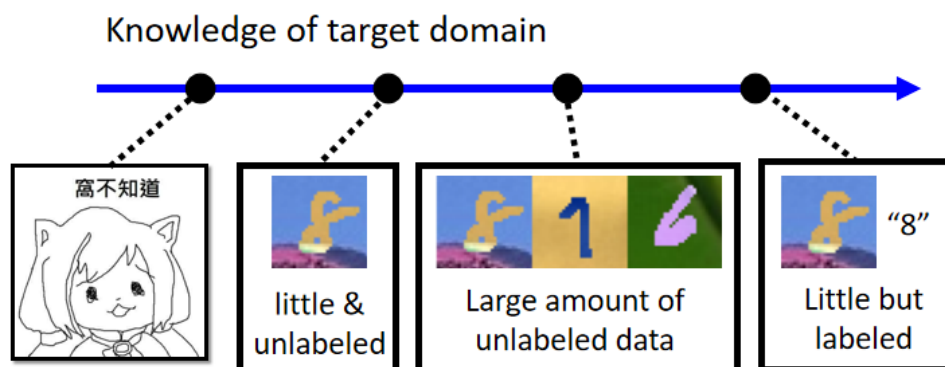
但是有一个可能是,假设不只没有 Label,而且 Data 还很少,比如说我就只有一张而已



这个时候你只有一张,你的这个 Target Domain 只有一张,只有一个点,你根本没有办法跟那个 Source Domain,把它 Align 在一起,这个时候怎麽办呢,假设 Target Domain 的 Data 非常少的时候,怎麽办呢,也不是没有方法啦

有一个方法叫做 **Testing Time Training**, 它的缩写是 TTT, 这个我们就把连结附在这边给大家参考, Testing Time Training 就是想要处理, 假设我的 Target Domain 没有 Label, 而且还只有一张的时候, 到底应该要怎么办

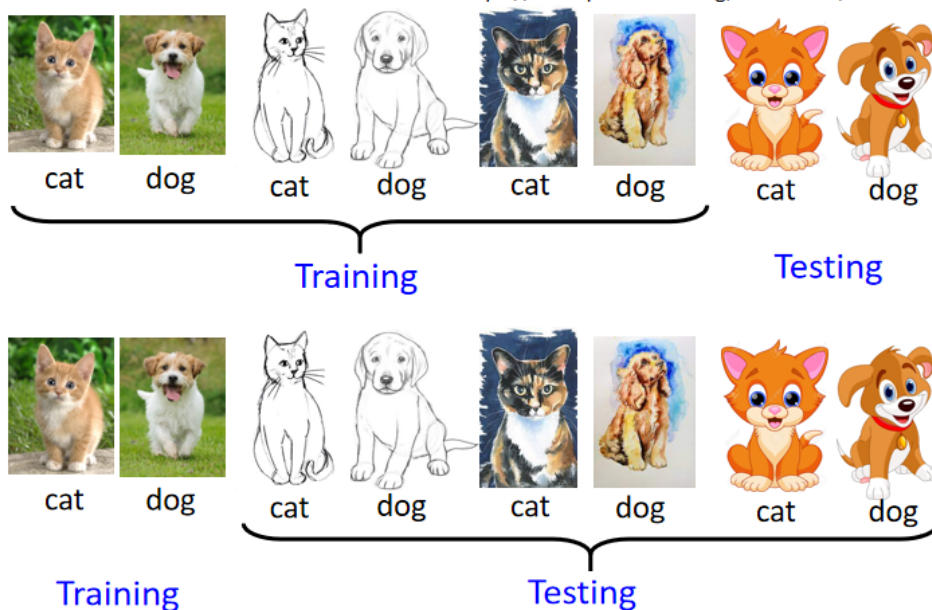
但其实还有一个更严峻的状况, 这个状况是, 如果我什麼都不知道怎么办呢, 如果我们对 Target Domain 一无所知的话, 怎么办呢



这个时候又分成两种情形, 对 Target Domain 一无所知的这种问题, 这个时候我们就不叫 Domain 的 Adaptation, 通常就叫 **Domain Generalization**

### **Domain Generalization**

<https://ieeexplore.ieee.org/document/8578664>



<https://arxiv.org/abs/2003.13216>

因为我们并不是要, Adapt 到某一个特定的 Domain 上, 我们对那个特定的 Domain 已经一无所知了, 我们是期待今天机器学到, Domain Generalization, 在 Testing 的时候, 不管来什麼神奇的 Domain, 它都可以处理, 那 Domain Generalization, 又分成两种状况

- 一种状况是我的训练资料非常地丰富, 本来就包含了各式各样不同的 Domain, 假设你要做猫狗的分类器, 那你现在呢 在训练资料裡面, 有真实的猫跟狗的照片, 有素描的猫跟狗的照片, 然后有这个水彩画的猫跟狗的照片, 期待因为训练资料有多个 Domain, 模型可以学到如何弥平 Domain 间的差异, 今天有测试资料是卡通的猫跟狗, 它也可以处理, 这是一种状况, 那这种状况你还比较能够想像要怎麼处理, 那我们这边就不细讲, 我们都只各放一些有代表性的论文, 给大家参考
- 但还有另外一种, 你会觉得真的不知如何下手的状况是, 假设训练资料只有一个 Domain 呢, 假设你的训练资料只有一个 Domain, 而测试资料有多种不同的 Domain 的话, 怎麼处理呢, 在文献上也不是没有人试, 也是有人试著去解惑这种问题的, 那他怎麼做呢, 这细节我们就不讲啦, 在概念上就是有点像是 Data Augmentation, 虽然你只有一个 Domain 的资料, 想个 Data Augmentation 的方法, 去產生多

个 Domain 的资料,然后你就可以套上面这个 Scenario 来做做看,看能不能够在测试的时候,新的 Domain 都可以做好,好 这个是 Domain Generalization

那这个部分就是很简短的跟大家带过,这个 Domain Adaptation 的种种技术,更多的细节,在下一堂课助教的说明裡面