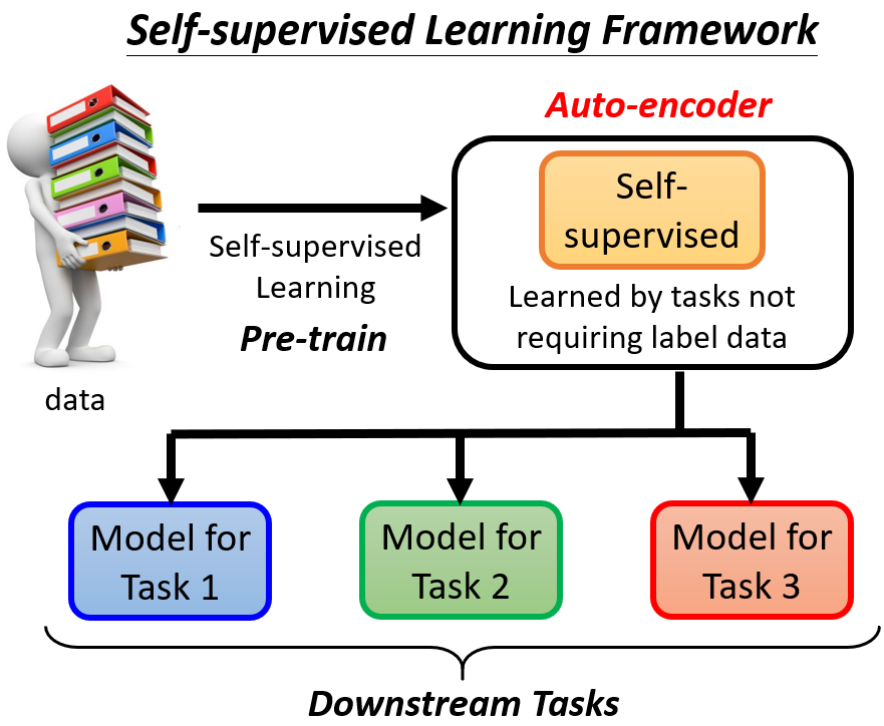
Auto-Encoder P1

Self-supervised Learning Framework

在讲 Auto-Encoder 之前,其实 Auto-Encoder 也可以算是,Self-Supervised Learning 的一环,所以再让我们用非常短的时间,来看一下Self-Supervised Learning 的 Framework



首先你有大量的没有标注的资料,用这些没有标注的资料,你可以去训练一个模型,你必须发明一些不需要标注资料的任务,比如说做填空题,比如说预测下一个 Token

这个不用标注资料的学习叫做,Self-Supervised Learning,或者是也有人叫 Pre-Training,那用这些不用标注资料的任务,学完一个模型以后,它本身没有什麼用,BERT 只能做填空题,GPT 只能够把一句话补完,但是你可以把它用在其他下游的任务裡面

你可以把 Self-Supervised Learning 的 Model,做一点点的微微的调整,就可以用在下游的任务裡面

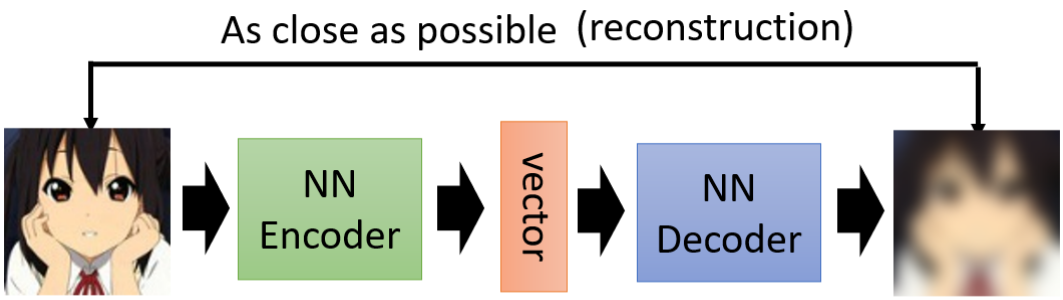
在有 BERT 在有 GPT 之前,其实有一个更古老的任务,更古老的不需要用标注资料的任务,就叫做 Auto-Encoder,所以你也可以**把 Auto-Encoder,看作是 Self-Supervised Learning 的,一种 Pre-Train 的方法**

当然可能不是所有人都会同意这个观点,有人可能会说这个 Auto-Encoder,不算是 Self-Supervised Learning,这个 Auto-Encoder 很早就有了嘛,2006 年 15 年前就有了嘛,然后 Self-Supervised Learning 是,19 年才有这个词彙嘛,所以 Auto-Encoder,不算 Self-Supervised Learning 的一环

那这个都是见仁见智的问题,这种名词定义的问题,真的我们就不用太纠结在这个地方,从 Self-Supervised Learning,它是不需要用 Label Data 来训练,这个观点来看,Auto-Encoder 我认为它可以算是,Self-Supervised Learning 的其中的一种方法,它就跟填空 预测,接下来的 Token 是很类似的概念,只是用的是另外不一样的想法

Auto-encoder

Auto-Encoder 是怎麼运作的呢,那现在我们,因为刚才在讲 Self-Supervised Learning 的时候,都是用文字做例子,那现在我们换成用影像来做例子



假设你有非常大量的图片,在 Auto-Encoder 裡面你有两个 Network,一个叫做 Encoder,一个叫做 Decoder,他们就是两个 Network

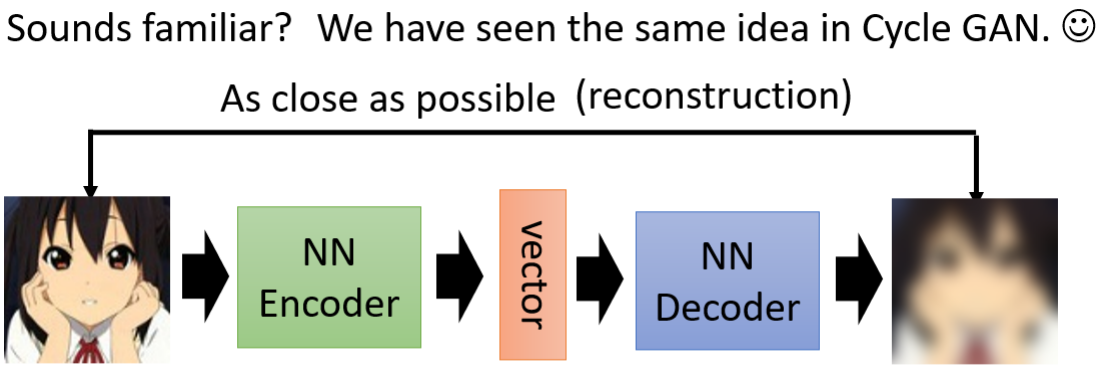
- Encoder 把一张图片读进来,它把这张图片变成一个向量,就 Encoder 它可能是很多层的 CNN,把一张图片读进来,它的输出是一个向量,接下来这个向量会变成 Decoder 的输入
- Decoder 会产生一张图片,所以 Decoder 的 Network 的架构,可能会像是 GAN 裡面的 Generator,它是 11 个向量输出一张图片

训练的目标是希望,**Encoder 的输入**跟 **Decoder 的输出,越接近越好**

假设你把图片看作是一个很长的向量的话,我们就希望这个向量跟 Decoder 的输出,这个向量,这两个向量他们的距离越接近越好,也有人把这件事情叫做 Reconstruction,叫做重建

因为我们就是把一张图片,压缩成一个向量,接下来 Decoder 要根据这个向量,重建原来的图片,那我们希望原输入的结果,跟重建后的结果越接近越好

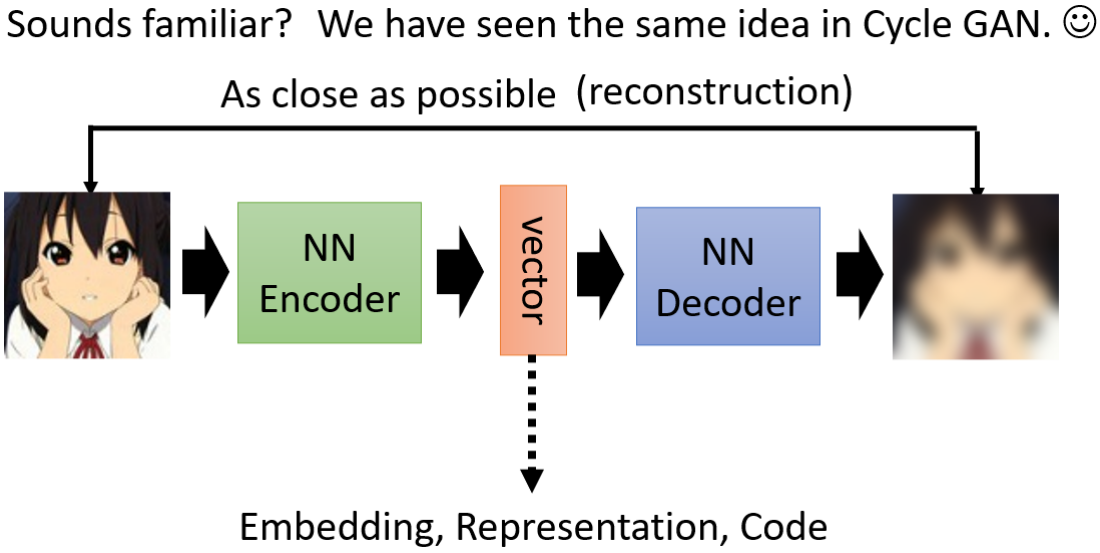
讲到这边你可能会发现说,这个东西 这个概念似曾相似,没错 我们在讲 Cycle GAN 的时候,已经讲过了这个概念



我们说在做 Cycle GAN 的时候,我们会需要两个 Generator,第一个 Generator,把 X Domain 的图片转到 Y Domain,另外一个 Generator,把 Y Domain 的图片转回来,希望最原先的图片,跟转完两次后的图片越接近越好

那这边 Encoder 和 Decoder,这个 Auto-Encoder 的概念,跟 Cycle GAN 其实是一模一样的,都是希望所有的图片经过两次转换以后,要跟原来的输出越接近越好,而这个训练的过程,完全不需要任何的标注资料,你只需要蒐集到大量的图片,你就可以做这个训练

所以它是一个 **Unsupervised Learning 的方法**,跟 Self-Supervised 那一系列,Pre-Training 的做法一样,你**完全不需要任何的标注资料**

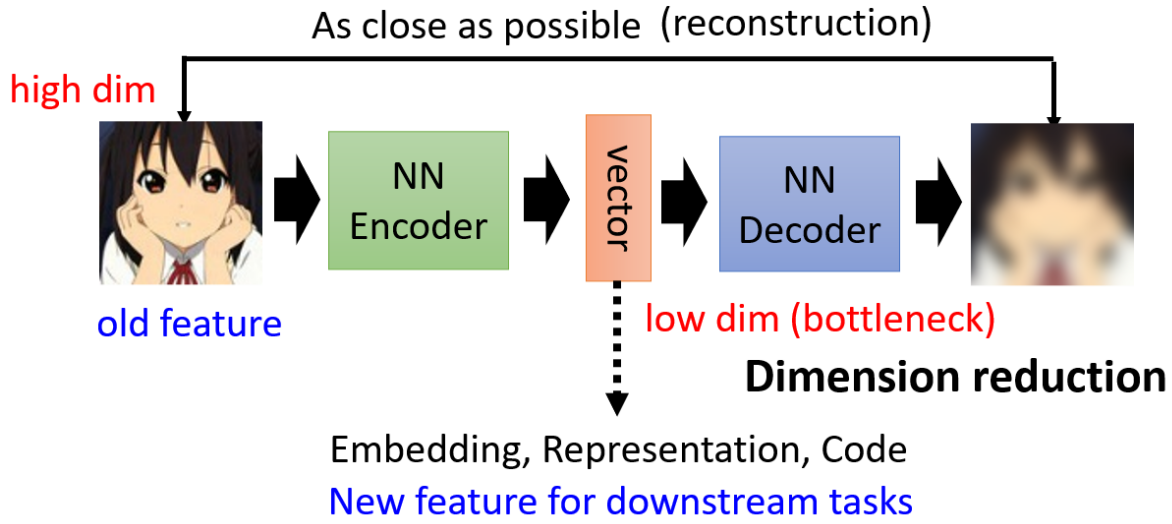


那像这样子这个 Encoder 的输出,有时候我们叫它 Embedding,我们在讲 BERT 的时候,也提过 Embedding 这个词彙了,那有的人叫它 Representation,有的人叫它 Code,因为 Encoder 是一个编码嘛,所以这个有人把这个 Vector 叫做 Code,那其实指的都是同一件事情

怎麼把 Train 的 Auto-Encoder,用在 Downstream 的任务裡面呢

常见的用法就是,原来的图片,你也可以把它看作是一个很长的向量,但这个**向量太长了 不好处理**,那怎麼办呢

你把这个图片丢到 **Encoder 以后,输出另外一个向量,这个向量你会让它比较短**,比如说只有 10 维 只有 100 维,那你拿这个新的向量来做你接下来的任务,也就是图片不再是一个很高维度的向量,它通过 Encoder 的压缩以后,变成了一个低维度的向量,你再拿这个低维度的向量,来做接下来想做的事情,这就是常见的,Auto-Encoder用在 Downstream 的任务,用在下游任务的方法

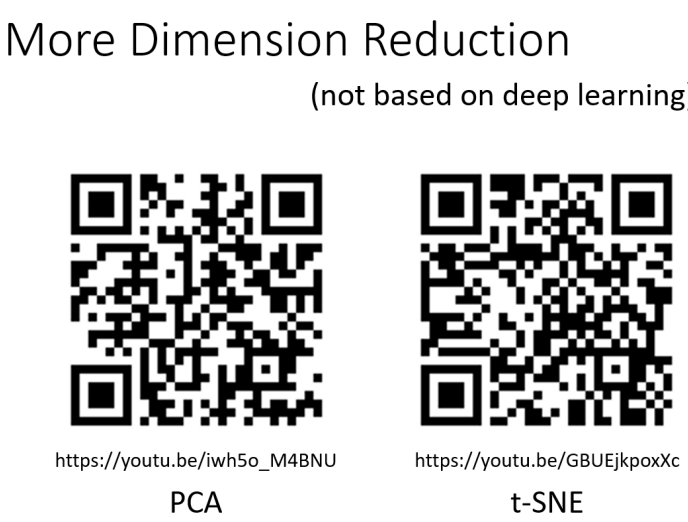


那因为通常 Encoder 的输入,是一个维度非常高的向量,而 Encoder 的输出,也就是我们的 Embedding,Representation 或者 Code,它是一个非常低维度的向量,比如说输入是 100×100 的图片,那 100×100 那就是 1 万维的向量了,如果是 RGB 那就是 3 万维的向量

但是通常 Encoder 的 Output 你会设得很小,比如说 10，100 这样的等级,所以这个这边会有一个特别窄的地方,所以这个部分,这个 Encoder 的输出,有时候又叫做 Bottleneck,叫做瓶颈,就本来输入是很宽的,输出也是很宽的 中间特别窄,所以这一段就叫做 Bottleneck

而 Encoder 做的事情,是把本来很高维度的东西,转成低维度的东西,把高维度的东西转成低维度的东西又叫做 Dimension Reduction

Dimension Reduction 这个技术,我相信你在 Machine Learning 相关的应用上,应该常常听到这个名词,那有关 Dimension Reduction 的技术,它其实牵涉的非常非常地广,所以我们这边就不再细讲,因为这门课,我们只专注在深度学习相关的技术,你可以把 Auto-Encoder 的 Encoder,当作拿来做 Dimension Reduction,那其他还有很多不是 Deep Learning Base的,不是以深度学习为基础的,Dimension Reduction的技术,我就把录影的连接留在这边



比如说 PCA 比如说 T-SNE,我就把录影的连结留在这边给大家参考

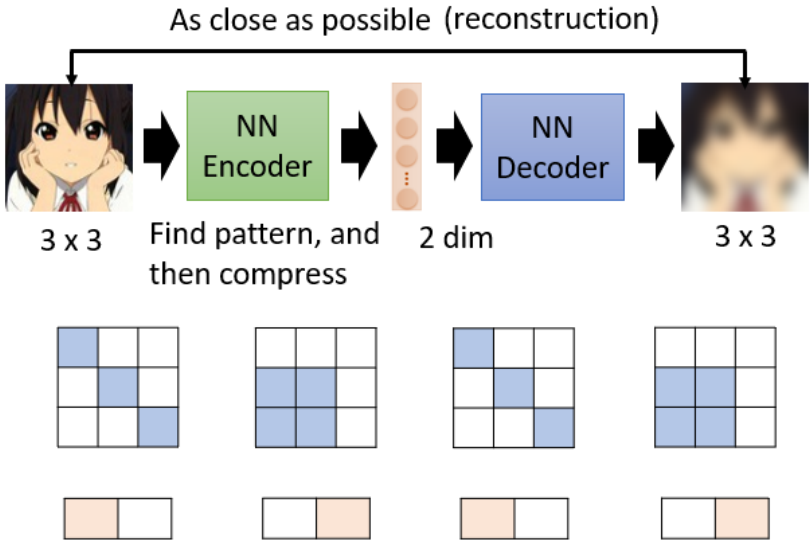
Why Auto-encoder?

好 那 Auto-Encoder 到底好在哪裡,当我们把一个高维度的图片,变成一个低维度的向量的时候,到底带来什麼样的帮助,这让我想到神鵰侠侶的其中一段

神鵰侠侶裡面有一段,就是杨过进去那个绝情谷,遇到这个绝情谷谷主公孙止的弟子,就是樊一翁,樊一翁就是这个人,那樊一翁的武器是什麼,他的武器除了一根钢杖以外,还有他的鬍子,他可以去甩动他的鬍子当做一个软鞭来使用,他的鬍子甩起来有两丈那麼长,可以是一个很厉害的武器,杨过跟他打了很久都难分上下

突然呢杨过说,我在三招之内一定要剪掉你的鬍子,大家突然都很诧异,想说杨过虽然武功可能比樊一翁还高一点,但是也没有高太多,怎麼有办法三招就剪掉他的鬍子,后来杨过真的在三招内剪掉他的鬍子,为什麼呢,因为杨过发现说,这个鬍子是由头所操控的,虽然鬍子甩开来有两丈那麼长,但是头能够做的变化还是有限的,所以虽然表面鬍子的鞭法非常地厉害,但是只要直接去打他的头,就直接去打他脸,就会逼著他不得不闪避,就会逼著他这个鬍子能够动的路线变得有限,然后就打败了樊一翁,就把他的鬍子剪掉了,故事结束,那这个跟 Auto-Encoder 有什麼关係呢

好 我们来想一下,Auto-Encoder 这件事情它要做的,是**把一张图片压缩又还原回来,**但是还原这件事情为什麽能成功呢



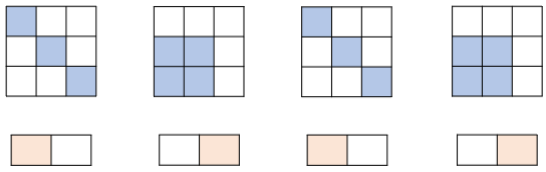
你想想看假设本来图片是 3×3,3×3 很小,但我们就假设 3×3 好了,本来的图片是 3×3,你要用 9 个数值来描述一张 3×3 的图片,假设 Encoder 输出的这个向量是二维的,我们怎麽有可能从二维的向量,去还原 3×3 的图片,还原9个数值呢

我们怎麽有办法把 9 个数值变成 2 个数值,又还原成 3,又还原回 9 个数值呢

能够做到这件事情是因为,**对于影像来说,并不是所有 3×3 的矩阵都是图片,图片的变化其实是有限的**,你随便 Sample 一个 Random 的 Noise,随便 Sample 一个矩阵,出来它通常都不是你会看到的图片

举例来说,假设图片是 3×3 的,那它的变化,虽然表面上应该要有 3×3 个数值,才能够描述 3×3 的图片,但是也许它的变化实际上是有限的

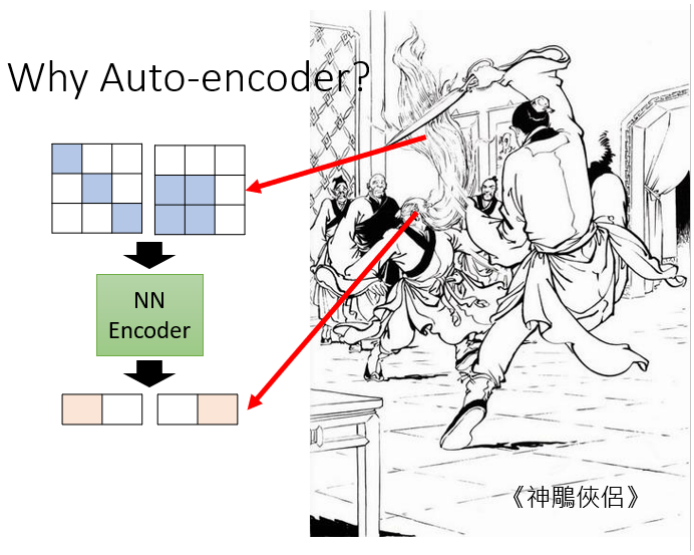
也许你把图片蒐集起来发现说



它只有这样子的类型,跟这样子的类型,其他类型根本就不是,你一般在训练的时候会看到的状况,就是因为说图片的变化还是有限的

所以你在做这个 Encoder 的时候,Encoder 可以说,我就只用两个维度就可以描述一张图片,虽然图片是 3×3,应该用 9 个数值才能够储存,但是实际上它的变化也许只有两种类型,那你就可以说看到这种类型,我就左边这个维度是 1 右边是 0,看到这种类型就左边这个维度是 0,右边这个维度是 1

那所以对应到刚才这个樊一翁的例子



就是这个鬍子是图片複杂的状态,是原来图片的 Pixel,是原来图片的像素

而 Encoder 做的事情就是化繁为简,本来比较複杂的东西,它只是**表面上比较複杂,事实上它的变化其实是有限的**,你只要找出它有限的变化,你就可以把本来複杂的东西,把它变得用比较简单的方法来表示它

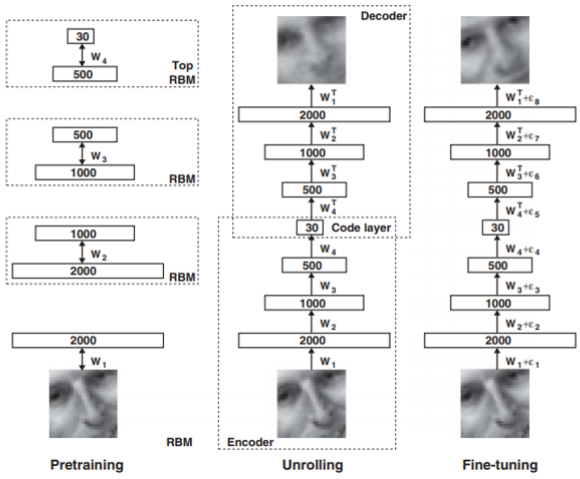
如果我们可以把複杂的图片,用比较简单的方法来表示它,那我们就只需要比较少的训练资料,在下游的任务裡面,我们可能就只需要比较少的训练资料,就可以让机器学到,我们本来要它学的事情,这个就是 Auto-Encoder 的概念

Auto-encoder is not a new idea

那 Auto-Encoder,它从来都不是一个新的想法,它真的是非常非常地有历史,举例来说在这个 Hinton,Hinton 就是 Deep Learning 之父

Hinton 在 06 年的 Science 的 Paper 裡面,就有提到 Auto-Encoder 这个概念,只是那个时候用的 Network,跟今天用的 Network,当然还是有很多不一样的地方,我们讲 2006 年是 15 年前,15 年前的 Auto-Encoder 长什麼样子

Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507



那个时候人们**不觉得,Deep 的 Network 是 Train 得起来的**,那时候觉得说这个把 Network 叠很多很多层,然后每一层一起 Train 不太可能成功,所以那时候的信念是,**每一层应该分开训练**,所以 Hinton 用的是一个叫做,Restricted Boltzmann Machine 的技术,缩写是 RBM

我们特别把 Hinton 15 年前的文章,把它的裡面的这个,Paper 裡面的图拿出来给大家看一下,过去 15 年前,人们是怎麼看待深度学习这个问题,那个时候觉得说,要 Train 一个这个很深的 Network 不太可能,每一层分开要 Train,虽然这个说很深也没有很深,只是三层,这个跟你作业 2 做得还要更shallow,但是在15年前这个已经是,哇 很深啊 它有三层太可怕了

那这个三层要分开来训练才可以,那这边说分开来训练这件事情叫做 Pretraining,但它跟 Self-Supervised Learning 的 Pre-Train,又不一样

假设你说 Auto-Encoder 这个东西是 Pre-Train,那现在这个 Pre-Train 是,Pre-Train 的 Pre-Train,它是要 Pre-Train 那个 Auto-Encoder,而且每一层用一个叫做 RBM 的技术,分开来训练

先把每一层都训练好,再全部接起来做微调这件事情,那这边的微调并不是 BERT 的微调,它是微调那个 Pre-Train 的 Model

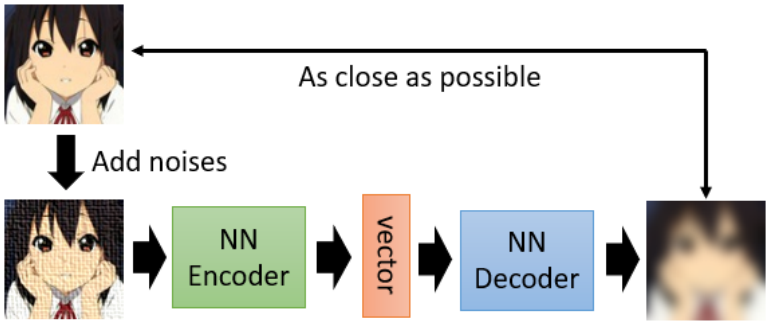
那这个 Restricted Boltzmann Machine,你会发现今天很少有人在提到它了,它其实不是一个 Deep Learning 的技术,它有点複杂,我们在这门课裡面也没有打算要深入细讲,什麼是 Restricted Boltzmann Machine,那为什麼现在都没有什麼人用它呢,就是因为**它没有什麼用**

在10年前呢,都相信这个 Deep 的 Network,一定要用 Restricted Boltzmann Machine,然后其实 Hinton 后来在 2012 年的时候,有一篇 Paper 偷偷在结尾下一个结论说,**其实 Restricted Boltzmann Machine,也没有什麼必要** ,所以后来就没有什麼人再用Restricted Boltzmann Machine

而且那时候还有一个神奇的信念,是觉得说那个 Encoder Decoder,它必须是对称,所以 Encoder 的第一层,跟 Encoder 的最后,跟 Decoder 的最后一层,他们必须互为 Transfers,不过现在已经没有,比较少有人在使用这样子的限制,好 这张投影片只想告诉你说,**Auto-Encoder 不是新的概念,它是一个非常有历史的概念**

De-noising Auto-encoder

那 Auto-Encoder 还有一个常见的变形,叫做 De-Noising 的 Auto-Encoder



The idea sounds familiar? ☺

Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *ICML,* 2008.

De-Noising 的 Auto-Encoder 是说,我们把原来要输进去给 Encoder 的图片,**加上一些杂讯**,就自己随便找一个杂讯把它加进去,然后一样通过 Encoder,一样再通过 Decoder,试图还原原来的图片

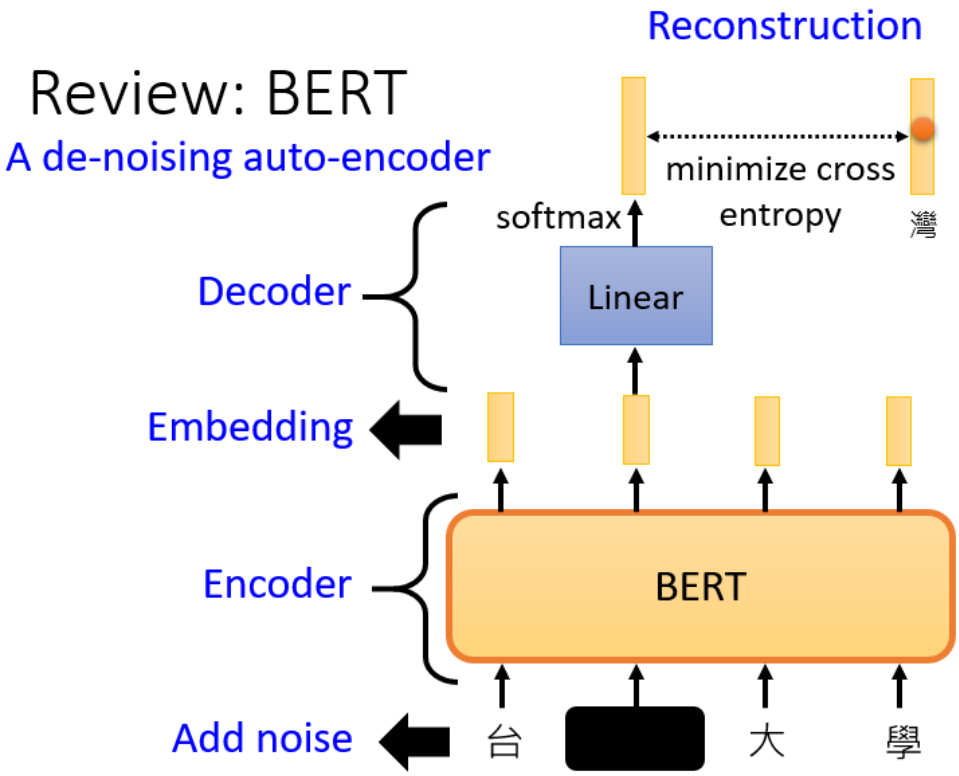
那我们现在还原的,不是 Encoder 的输入,Encoder 的输入的图片是有加杂讯的,我们要还原的不是 Encoder 的输入,我们**要还原的是加入杂讯之前的结果**

所以你会发现说,现在 Encoder 跟 Decoder,除了还原原来的图片这个任务以外,它还**多了一个任务**,这个任务是什麼,这个任务就是,它必须要**自己学会把杂讯去掉**

Encoder 看到的是没有杂讯的图片,但 Decode要还原的目标是,Encoder 看到的是有加杂讯的图片,但 Decoder 要还原的目标是,没有加杂讯的图片,所以 Encoder 加 Decoder,他们合起来必须要联手能够把杂讯去掉,这样你才能够把,De-Noising 的 Auto-Encoder 训练起来

那说到 De-Noising 的 Auto-Encoder,有没有发现这个概念,其实也一点都不陌生呢,De-Noising 的 Auto-Encoder,也不算是太新的技术,至少在 2008 年的时候,就已经有相关的论文了

但是如果你看今天的 BERT 的话,其实你也可以把它看作就是一个,De-Noising 的 Auto-Encoder



输入我们会加 Masking,那些 **Masking 其实就是 Noise**,BERT 的模型就是 Encoder,它的输出就是 Embedding

在讲 BERT 的技术的时候,我们就告诉你说这个输出就叫做 Embedding,接下来有一个 Linear 的模型,就是 Decoder,Decoder 要做的事情,就是还原原来的句子,也就是把填空题被盖住的地方,把它还原回来,所以我们可以说,BERT 其实就是一个,De-Noising 的 Auto-Encoder

有同学可能会问说,为什麼这个 Decoder 一定要 Linear 的呢,它**不一定要是 Linear**,它可以不是 Linear

或者是我们换一个说法,这个 BERT 它有 12 层,最小的那个 BERT 有 12 层,比较大的有 24 层或者是 48 层,好 那最小的 BERT 是 12 层,如果我们说这个 12 层中间,第 6 层的输出是 Embedding,那你其实也可以说剩下的 6 层,就是 Decoder,你可以说 BERT,就假设你在用 BERT 的时候,你用的不是第 12 层的输出,而是第 6 层的输出,那你完全可以说,BERT 的前 6 层就是 Encoder,后面 6 层就是 Decoder,总之这个 Decoder,没有一定要是 Linear