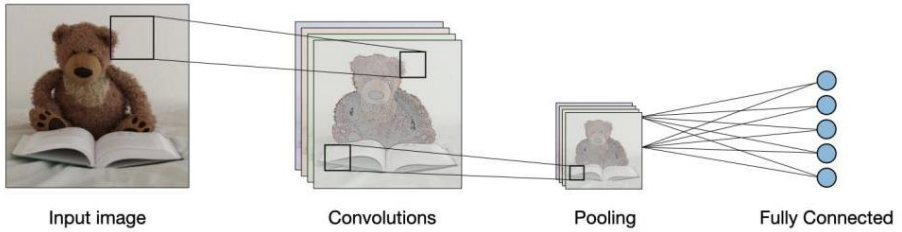
第一部分 卷积神经网络 / Convolutional Neural Networks

翻译&校正 | 韩信子@ShowMeAI 编辑 | 南乔@ShowMeAI 原文作者 | https://stanford.edu/~shervine 本节原文超链

[1] 概述 / Overview

传统 CNN 架构 Architecture of a traditional CNN

卷积神经网络（Convolutional Neural Networks，CNNs），是一种特定类型的神经网络，通常由以下几种网络层组成：



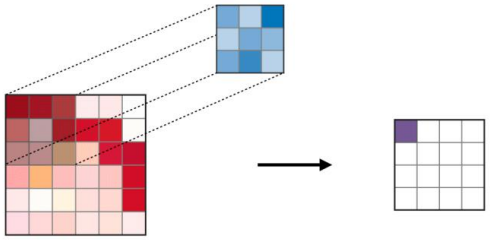
备注 1：Convolution layer [卷积层]，Pooling layer[池化层]，Fully connected layer[全连接层]。

备注 2：卷积层（Convolution layer，CONV）和池化层（Pooling layer，POOL）可以通过超参数来进行微调（详见下文）。

[2] 网络层类别 / Types of layer

卷积层 Convolution layer (CONV)

卷积层（CONV）利用卷积核在相应的维度上扫描输入 I，来实现卷积操作。它的超参数包括卷积核的大小 F 和步长 S。得到的输出结果 O 即为特征图或激活图。



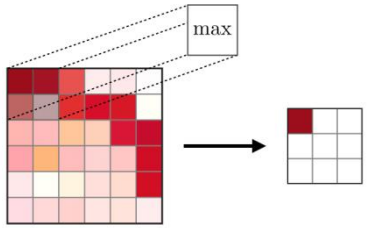
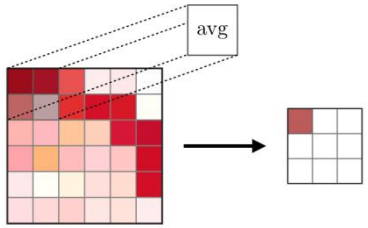
备注：卷积操作也可以推广到一维 (1D) 和三维 (3D) 的情况。

GIF 图→ 点击 ShowMeAI GitHub

池化层 Pooling (POOL)

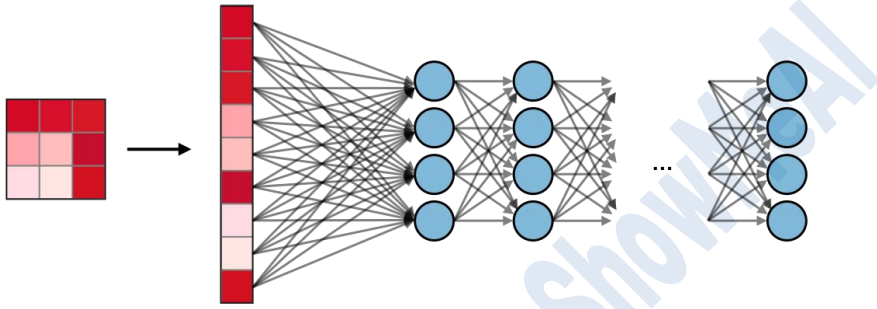
池化层（POOL）是一种下采样操作，通常接在卷积层之后，具有空间不变性。

最大池化（Max pooling）和平均池化（Average pooling）是两种特殊的池化方式，分别取邻域内的最大值和平均值作为单元输出。

类型	最大池化 Max pooling	平均池化 Average pooling
目的	每个池化操作选择当前窗口中的最大值	每个池化操作计算当前窗口内的平均值
图例		
注释	保留检测特征 最常用	特征图的下采样 在 LeNet 中使用

全连接层 Fully Connected (FC)

全连接层（FC）是在展开的输入上操作的，其中每个输入都会和所有神经元连接。如果网络中存在全连结层的话，它通常出现在 CNN 网络结构的末尾处，可以用于目标函数的优化，例如类别评分等。GIF 图→ 点击 ShowMeAI GitHub

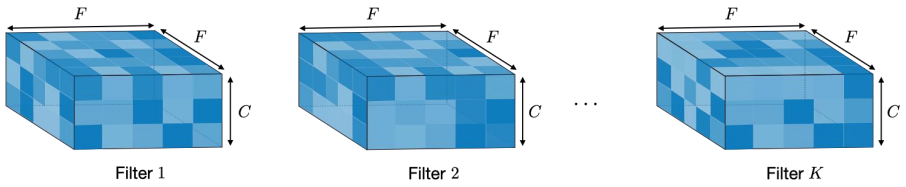


[3] 卷积核超参数 / Filter Hyperparameters

卷积层中包含卷积核。对于这些卷积核，了解其超参数的含义是很重要的。

卷积核的维度 Dimensions of a filter

一个大小为 F × F 的卷积核应用于包含 C 个通道的输入得到的卷积核体为 F × F × C，在输入大小为 I × I × C 上做卷积操作，生成大小为 O × O × 1 的特征图（激活图）。



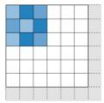
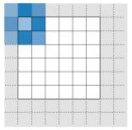
备注：用 K 个 F×F 大小的卷积核作用后得到大小为 O×O×K 的特征图

步长 Stride

对于卷积或者池化操作，步长 S 表示的是卷积窗口在每次操作完成后移动的像素数。

零填充 Zero-padding

零填充表示在输入的边界增加 P 个 0 。这个值可以手动指定，也可以通过下面三种模式自动设置如下：

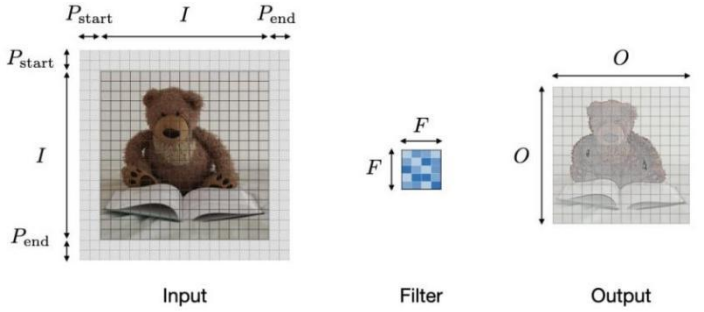
模式	Valid	Same	Full
值	$P = 0$	$P_{start} = \left\lceil \frac{S\left\lceil \frac{I}{S} \right\rceil - I + F - S}{2} \right\rceil$ $P_{end} = \left\lceil \frac{S\left\lceil \frac{I}{S} \right\rceil - I + F - S}{2} \right\rceil$	$P_{start} \in [[0, F-1]]$ $P_{end} = F - 1$
图例			
目的	无填充 如果维度不匹配丢弃最后的卷积操作	填充使得特征图大小为 $\left\lceil \frac{I}{S} \right\rceil$ 输出大小在数学计算上很方便 也称之为“半”填充	最大填充使得末端卷积应用于输入的限制 卷积核“看到”端到端的输入

[4] 调整超参数 / Tuning Hyperparameters

卷积层中的参数兼容性 Parameter compatibility in convolution layer

用 I 来表示输入大小，F 表示卷积核的长度，P 表示零填充的大小，S表示步长，那么沿着维度的输出特征图的大小 O 公式为：

$$O = \frac{I - F + P_{start} + P_{end}}{S} + 1$$

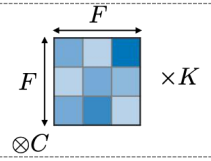
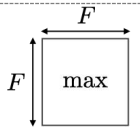
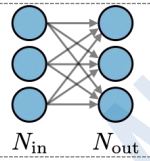


备注 1：输入 [Input]，卷积核[Filter]，输出[Output]。

备注 2：通常情况下，$P_{start} = P_{end} \triangleq P$，这个场景下就可以将上面公式中的 $P_{start} + P_{end}$ 替换为 $2P$ 。

理解模型的复杂度 Understanding the complexity of the model

为了确定一个模型的复杂度，确定模型框架中包含的参数量通常很有帮助。在卷积神经网络的给顶层中，其操作如下：

	CONV	POOL	FC
图例			
输入大小	$I \times I \times C$	$I \times I \times C$	N_{in}
输出大小	$O \times O \times K$	$O \times O \times C$	N_{out}
参数量	$(F \times F \times C + 1) \cdot K$	0	$(N_{in} + 1) \times N_{out}$
备注	每个卷积核一个偏置参数 在大多数情况下 $S < F$ 常见的 K 为 $2C$	池化操作是按通道进行的 大多数情况下，$S = F$	输入层展开 每个神经元一个偏置参数 FC 层的神经元数量不受结构限制

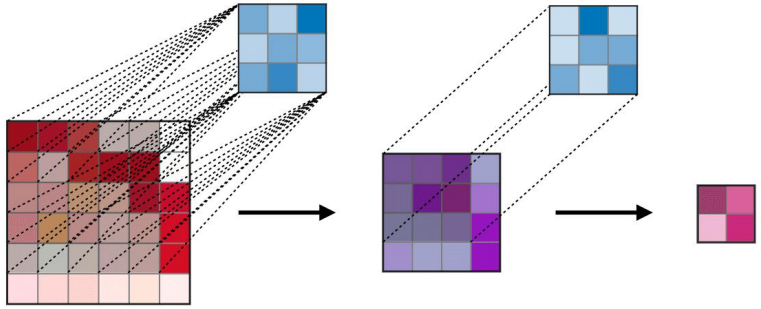
感受野 Receptive field

第 k 层的感受野是输入层中第 k 个激活图能够看到的像素，该区域记为 $R_k \times R_k$ 。

F_j 表示第 j 层的卷积核大小， S_i 表示第 i 层的步长值，约定 $S_0 = 1$，第 k 层的感受野可以用一下的公式来计算：

$$R_k = 1 + \sum_{j=1}^{k} (F_j - 1) \prod_{i=0}^{j-1} S_i$$

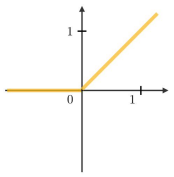
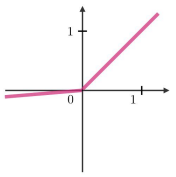
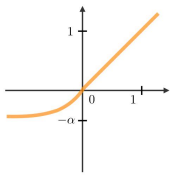
在下面这个例子中，$F_1 = F_2 = 3$ 以及 $S_1 = S_2 = 1$，得到 $R_2 = 1 + 2 \cdot 1 + 2 \cdot 1 = 5$。



[5] 常用的激活函数 / Commonly Used Activation Functions

修正线性单元 Rectified Linear Unit

修正线性单元（ ReLU ）是一个激活函数，在所有的输入所有元素上使用。它是为了将非线性引入到网络中。下表总结了其变体：

ReLU	Leaky ReLU	ELU
$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$	$g(z) = \max(\alpha(e^z - 1), z)$ with $\alpha \ll 1$
		
生物学上可以解释的非线性复杂度	解决 ReLU 负值死亡的问题	全局可导

Softmax

在网络末尾的 softmax 操作，可以看成是适用性更广的逻辑(logistic)函数，输入一个分数向量，通过 softmax 函数输出概率向量。

$$p = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} \quad \text{其中} \quad p_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

[6] 目标检测 / Object Detection

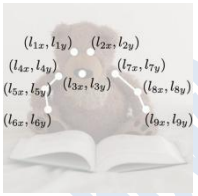
模型类别 Types of models

有 3 中主要的目标检测算法，因此预测的目标是不同的。如下表：

图像分类	分类和定位	目标检测
		
分类一张图片 预测物体的概率	在图片中检测一个物体 预测物体的概率和所处位置	在图片中检测多个物体 预测每个物体的概率和所处位置
传统 CNN	简化的 YOLO，R-CNN	YOLO，R-CNN

检测 Detection

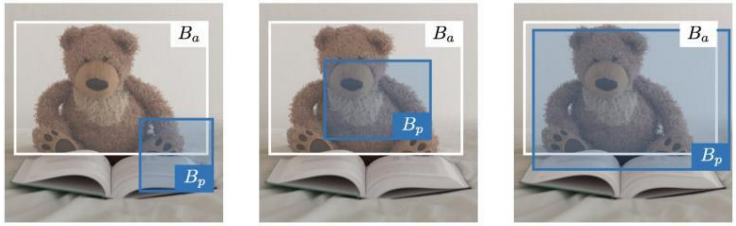
在目标检测的任务中，根据我们是指向定位物体的位置或者是检测图像中更复杂的形状，采用不同的方法。下表中总结了两个主要的方法：

边框检测 Bounding box detection	关键点检测 Landmark detection
检测物体所在的图片区域	检测一个形状或者物体的特征（比如眼睛），更精细
	
边框中心 (b_x, b_y)，高度 b_h 和宽度 b_w	参考点 $(l_{1x}, l_{1y}), \ldots, (l_{nx}, l_{ny})$

交并比 Intersection over Union

交并比，也称之为 IoU，是一个用于衡量预测框 B_p 与实际边框 B_a 相交正确率的函数。

$$\text{IoU}(B_p, B_a) = \frac{B_p \cap B_a}{B_p \cup B_a}$$



备注： 总有 $\text{IoU} \in [0,1]$。通常情况下，如果预测框 B_p 的 $\text{IoU}(B_p, B_a) \geq 0.5$，表明该预测结果相当不错了。

锚框 Anchor boxes

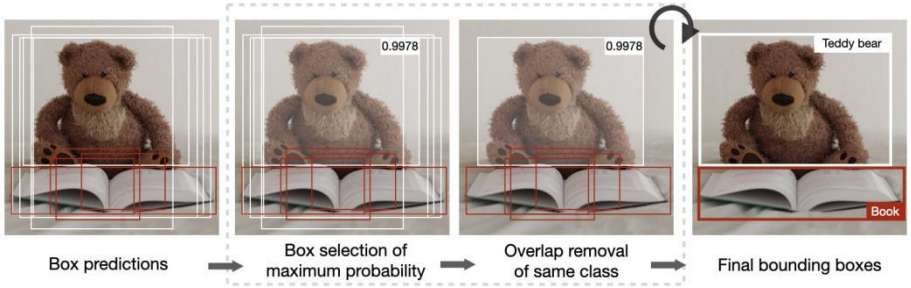
锚框是一种用于预测重叠边框的技术。在实际中，允许神经网络同时预测多个框，每个预测框具有特定的几何参数（比如长宽比）。比如说，第一个预测可能是某个长宽比的矩形框，而第二个可能是另一个长宽比的矩形框。

非极大值抑制 Non-max suppression

非极大值抑制是一种用于移除相同物体的重叠边界框，通过选择最有代表性的那个边框。删除小于 **0.6** 预测概率的边框之后，重复执行以下步骤直至保留最终的边框：
对于一个给定的类型：

第一步：选择预测概率最大的边框。

第二步：去掉与上一个边框的 IoU≥0.5 的边框。



备注：边框预测[Box predictions]，选择最大概率边框[Box selection of maximum probability]，去除同一类别的重复边框[Overlap removal of same class]，最终边框[Final bounding boxes]。

YOLO

You Only Look Once (YOLO) 是一个目标检测算法，执行步骤：

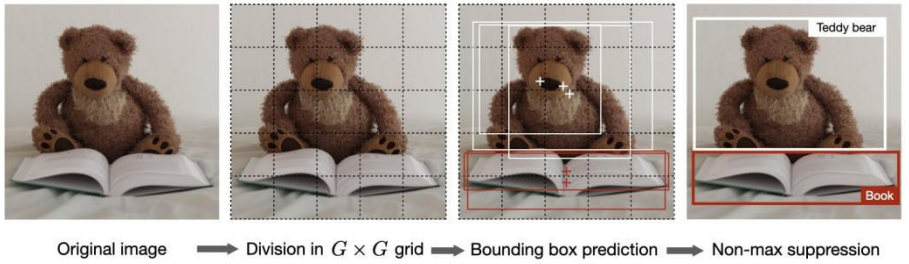
第一步：将输入图像分割为 $G \times G$ 的网格。

第二步：对于每个网格，运行 **CNN** 网络预测下面给定的 y，重复 k 次。

$$y = \left[p_c, b_x, b_y, b_k, b_w, c_1, c_2, \ldots, c_p, \ldots\right]^T \in \mathbb{R}^{GxGxkx(5+p)}$$

- p_c 是检测物体的概率
- b_x, b_y, b_h, b_w 是检测边框的属性
- $c_1, \ldots, c_p$ 是 p 个类别的独热向量表示
- k 是锚框的数量

第三步：运行非极大值抑制算法去除潜在的重叠边框。

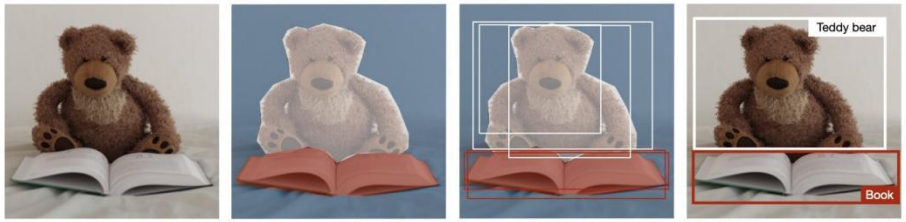


备注 1：原始图像[Original image]，分成 G×G 网格[Division in GxG grid]，边框预测[Bounding box prediction]，非极大值抑制[Non-max suppression]。

备注 2：当 $p_c = 0$，神经网络没有检测到任何物体。这个情况下，对应的预测结果 $b_x, \ldots, c_p$ 将被忽略。

R-CNN

R-CNN 是一个目标检测算法，首先对图像进行分割以找到候选边界框，然后运行检测算法以在候选边界框中找到最可能的目标对象。

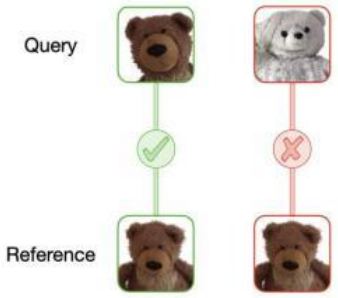
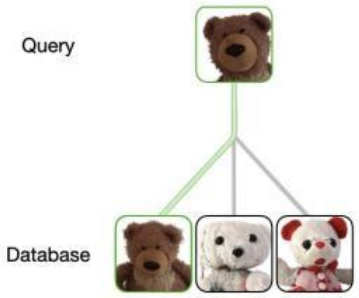


备注：尽管原始的 R-CNN 算法计算资源消耗大且速度慢，但是后面在它的基础上优化的算法，例如 Fast R-CNN 和 Faster R-CNN，有着更快的速度和更好的效果。

6.1 人脸验证和识别 Face Verification and Recognition

模型类别 Types of models

下表中总结两个主要的模型类别：

人脸验证 Face verification	人脸识别 Face recognition
这是正确的人吗？一对一查询	这是数据库中 K 个人的其中一个吗？一对多查询
	

单样本学习 One Shot Learning

单样本学习是一种人脸验证算法，使用有限的训练数据集来学习一个能够衡量两张给定图片的差异的相似度函数。通常将两张图片的相似度函数记作 **d(image 1，image 2)**。

孪生神经网络 Siamese Network

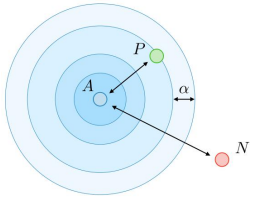
Siamese Networks（孪生神经网络）这类神经网络可以学习如何对图像进行编码，然后量化计算两个图像的不同程度。对于给定的输入图像 $x^{(i)}$，编码输出通常记作 $f(x^{(i)})$。

三元损失 Triplet loss

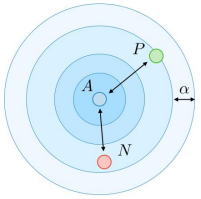
Triplet loss（三元损失）ℓ 是一个损失函数，根据图像三元组 **A**（锚点），**P**（正样本），**N**（负样本）的嵌入表示进行计算。

锚点和正样本属于同一个类别，负样本属于另外一个类别。如果我们定义边界距离为 $\alpha \in \mathbb{R}^+$，三元损失可以进行定义：

$$\ell(A,P,N) = \max(d(A,P) - d(A,N) + \alpha, 0)$$



$\ell(A,P,N) = 0$



$\ell(A,P,N) > 0$

6.2 神经风格迁移 Neural Style Transfer

动机 Motivation

神经风格迁移的目标是基于给定的图片内容 C 和图片风格 S，生成具有 C 内容和 S 风格的图片 G。

Content C + Style S = Generated image G

激活输出 Activation

在给定的网络层 l，激活函数输出记作 $a^{[l]}$，它的维度为 $n_H \times n_w \times n_c$。

内容代价函数 Content cost function

我们会用内容代价函数来衡量生成图片 G 和原始内容图片 C 的内容差异度。定义：

$$J_{\text{content}}(C,G) = \frac{1}{2}||a^{[l](C)} - a^{[l](G)}||^2$$

风格矩阵 Style matrix

同样第l层网络的风格矩阵$G^{[l]}$是一个 **Gram** 矩阵，矩阵中的每个元素$G^{[l]}_{kk'}$衡量通道k和k'之间的相关程度。它是基于激活输出$a^{[l]}$计算得到的，计算方式：

$$G^{[l]}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a^{[l]}_{ijk} a^{[l]}_{ijk'}$$

备注：风格图片的风格矩阵和生成图片分别记作$G^{[l](S)}$和$G^{[l](G)}$。

风格代价函数 Style cost function

风格代价函数$J_{style}(S,G)$用于衡量生成图片G和原始风格图片S之间的风格差异。

$$J^{[l]}_{style}(S,G) = \frac{1}{(2n_H n_W n_C)^2} ||G^{[l](S)} - G^{[l](G)}||^2_F = \frac{1}{(2n_H n_W n_C)^2} \sum_{k,k'=1}^{n_C} (G^{[l](S)}_{kk'} - G^{[l](G)}_{kk'})^2$$

总的代价函数 Overall cost function

最终代价函数定义为内容代价函数和风格代价函数的加权和，权重参数为α, β。

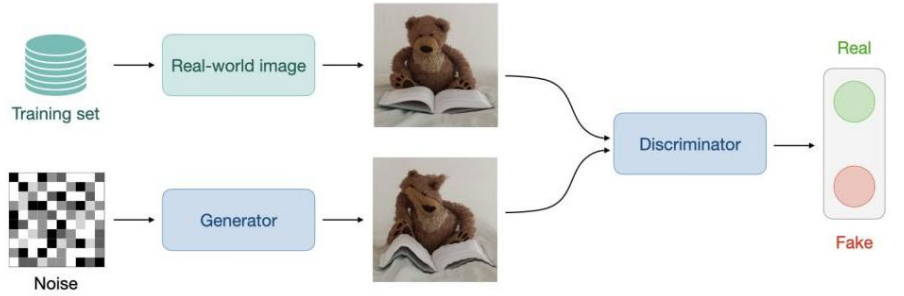
$$J(G) = \alpha J_{content}(C,G) + \beta J_{style}(S,G)$$

备注：较高的 α 值会使模型更加关注内容，而较高的 β 值会使模型更加关注风格。

6.3 使用计算技巧的架构 Architectures Using Computational Tricks

生成对抗网络 Generative Adversarial Network

生成对抗网络，也称之为 **GANs**，是由一个生成模型和判别模型构成，生成模型的目标是生成最真实的结果，这个结果输入到判别模型中，判别模型需要努力区分生成的图片和真实的图片。



备注：使用 **GANs** 变体的用例包括文本图像转换，音乐生成和语音合成。

残差网络架构 ResNet

残差网络架构（也称为 **ResNet**）使用具有大量网络层的残差单元来减小训练误差。残差单元具有以下特征方程式：

$$a^{[l+2]} = g(a^{[l]} + z^{[l+2]})$$

Inception 网络 Inception Network

inception 模块结构的特点在于，尝试不同的卷积，以便通过功能多样化来提高其性能。其中 1×1 是很特殊的卷积结构，可以降低计算量。

Awesome AI Courses Notes Cheat Sheets

Machine Learning CS229	Deep Learning CS230	Natural Language Processing CS224n	Computer Vision CS231n	Deep Reinforcement Learning CS285	Neural Networks for NLP CS11-747	DL for Self-Driving Cars 6.S094	...
Stanford	Stanford	Stanford	Stanford	UC Berkeley	CMU	MIT	...

是 **ShowMeAI** 资料库的分支系列，覆盖最具知名度的 TOP20+门 **AI** 课程，旨在为读者和学习者提供一整套高品质中文速查表，可以点击【这里】查看。

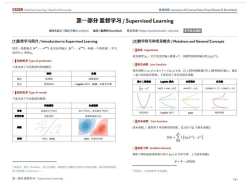
斯坦福大学（ **Stanford University** ）的 **Machine Learning**（ **CS229** ）和 **Deep Learning**（ **CS230** ）课程，是本系列的第一批产出。

本批两门课程的速查表由斯坦福大学计算机专业学生 **Shervine Amidi** 总结整理。原速查表为英文，可点击【这里】查看，**ShowMeAI** 对内容进行了翻译、校对与编辑排版，整理为当前的中文版本。

有任何建议和反馈，也欢迎通过下方渠道和我们联络 (*￣3￣)

CS229 | Machine Learning @ Stanford University

监督学习
Supervised Learning



中文速查表链接

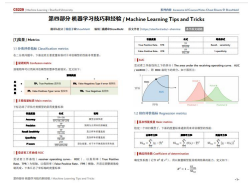
无监督学习
Unsupervised Learning

中文速查表链接

深度学习
Deep Learning

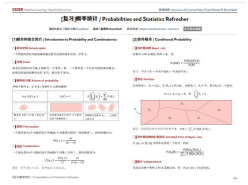
中文速查表链接

机器学习技巧和经验
Tips and Tricks



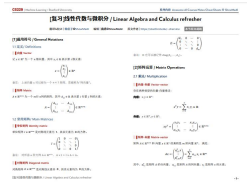
中文速查表链接

概率统计
Probabilities /Statistics



中文速查表链接

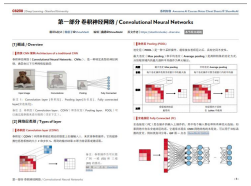
线性代数与微积分
Linear Algebra and Calculus



中文速查表链接

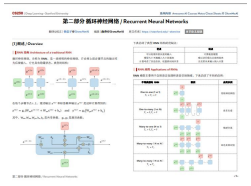
CS230 | Deep Learning @ Stanford University

卷积神经网络
CNN



中文速查表链接

循环神经网络
RNN



中文速查表链接

深度学习技巧与建议
Tips and Tricks

中文速查表链接

GitHub

ShowMeAI

https://github.com/ShowMeAI-Hub/

ShowMeAI 研究中心

扫码回复"**速查表**"

下载**最新**全套资料