



Chapter1&2

Chapter1

人工智能

人工智能定义

- “人工智能”的概念最早在1956年的美国达特茅斯会议（Dartmouth Conference）上提出，当时会议的主题是“用机器来模仿人类学习以及其它方面的智能”。因此，1956年被认为是人工智能的元年。
- 一般认为，人工智能是研究、开发用于模拟、延伸和扩展人的智能的理论、方法、技术及应用系统的一门新兴学科。

人工智能分类

计算智能（已实现）

感知智能（基本实现）

认知智能（未来热点）

人工智能技术

搜索理论、知识表示与推理方法、机器学习算法、感知、决策与控制技术等。

机器学习

机器学习定义

让计算机具有像人一样的学习和思考能力的技术的总称。具体来说是从已知数据中获得规律，并利用规律对未知数据进行预测的技术。

机器学习分类

- 有监督学习（跟学师评）：有老师（环境）的情况下，学生（计算机）从老师（环境）那里获得对错指示、最终答案的学习方法。包含线性回归、多项式回归、决策树和随机森林等回归算法，以及KNN、逻辑回归、贝叶斯和支持向量机等分类算法。
- 无监督学习（自学标评）：没有老师（环境）的情况下，学生（计算机）自学的过程，一般使用一些既定标准进行评价，或无评价。包含K-Means聚类、主成分分析、关联分析和密度估计等算法。
- 弱监督学习：仅有少量环境提示（教师反馈）或者少量数据（试题）标签（答案）的情况下，机器（学生）不断进行学习的方法。包含强化学习、半监督学习和多示例学习等算法。

算法

Machine Learning Algorithms (sample)

		<u>Unsupervised</u>	<u>Supervised</u>
<u>Continuous</u>	<ul style="list-style-type: none">• Clustering & Dimensionality Reduction<ul style="list-style-type: none">◦ SVD◦ PCA◦ K-means	<ul style="list-style-type: none">• Regression<ul style="list-style-type: none">◦ Linear◦ Polynomial• Decision Trees• Random Forests	
<u>Categorical</u>	<ul style="list-style-type: none">• Association Analysis<ul style="list-style-type: none">◦ Apriori◦ FP-Growth• Hidden Markov Model	<ul style="list-style-type: none">• Classification<ul style="list-style-type: none">◦ KNN◦ Trees◦ Logistic Regression◦ Naive-Bayes◦ SVM	

- 人工智能旨在为机器赋予人的智能，并使得机器在某些方面超越人类。
- 机器学习是人工智能的重要组成部分，让机器具有像人一样的学习能力。
- 深度学习是机器学习的一个重要分支，它突破了传统机器学习算法的瓶颈，在多个研究与应用领域取得了巨大的进展。

深度学习

深度学习定义

- 深度学习是指通过构建多层神经网络结构来学习数据的特征，以便于进行数据分类、回归与生成。
- 深度学习与浅层学习相比，神经网络结构的层数更多（一般大于或等于4层），通过多层神经网络结构可以学习得到更丰富的数据特征。
- 浅层前馈神经网络和深度前馈神经网络（区别在于隐藏层1层/多层）

深度学习算法

- 有监督学习的浅层学习算法：决策树、支持向量机、感知机和 Boosting 等。
- 无监督学习的浅层学习算法：自编码器、受限玻尔兹曼机、高斯混合模型和稀疏自编码器等。
- 有监督学习的深度学习算法：深度前馈神经网络、卷积神经网络、循环神经网络、Transformer、胶囊网络和深度森林等。
- 无监督学习的深度学习算法：深度自编码器、生成对抗网络、深度玻尔兹曼机和深度信念网络等。

1. 深度前馈神经网络 (Deep Feedforward Neural Network, DNN)

计算流程：

- 输入 → 多层线性加权求和（仿射变换）→ 非线性激活函数（如ReLU、Sigmoid）→ 输出。

- 所有层之间无循环或共享权重，信息从输入单向传播到输出。

理解：

最基本的神经网络结构，适用于静态输入数据（如图像、表格数据）。缺乏对时间或空间局部特征的建模能力。

2. 卷积神经网络 (Convolutional Neural Network, CNN)

计算流程：

- 输入图像 → 卷积层（提取局部特征）→ 激活函数 → 池化层（降采样）→ 多层堆叠 → 全连接层 → 输出。

理解：

特别适合处理具有局部结构的数据（如图像、视频）。利用卷积核共享权重，显著减少参数量并捕捉局部空间结构。

3. 循环神经网络 (Recurrent Neural Network, RNN)

计算流程：

- 输入序列逐步输入 → 每一步：当前输入与上一步隐藏状态结合 → 生成当前隐藏状态 → 最终输出。

理解：

适用于

序列数据（如时间序列、语音、文本）。能记住前面步骤的状态信息，但易受梯度消失/爆炸影响，记忆能力有限。

4. Transformer

计算流程：

- 输入嵌入 + 位置编码 → 多头自注意力机制（Self-Attention）→ 前馈网络 → 层归一化和残差连接 → 堆叠多个编码器/解码器。

理解：

完全基于注意力机制，无需循环结构，支持并行计算，适合处理长距离依赖的序列任务（如机器翻译、文本生成）。是当前NLP和多模态模型的核心架构。

5. 生成对抗网络 (Generative Adversarial Network, GAN)

计算流程：

- 生成器（Generator）：随机噪声 → 生成假数据。
- 判别器（Discriminator）：判断输入是真实数据或生成数据。
- 通过对抗训练，生成器逐渐学会欺骗判别器。

理解：

一个博弈框架中的双网络结构，适合生成高质量数据（如图像合成、风格迁移）。训练不稳定但生成效果出色。

6. 深度信念网络 (Deep Belief Network, DBN) 与深度玻尔兹曼机 (Deep Boltzmann Machine, DBM)

计算流程：

- DBN：由多个受限玻尔兹曼机（RBM）堆叠组成，逐层无监督训练 → 全网络微调。
- DBM：多层RBM组成，但具有对称连接，每层节点之间可以上下交流（比DBN更复杂）。

理解：

是早期深度学习的重要模型，强调无监督预训练。适用于特征学习与生成任务。相比DBN，DBM具有更强的表达能力但训练更复杂。

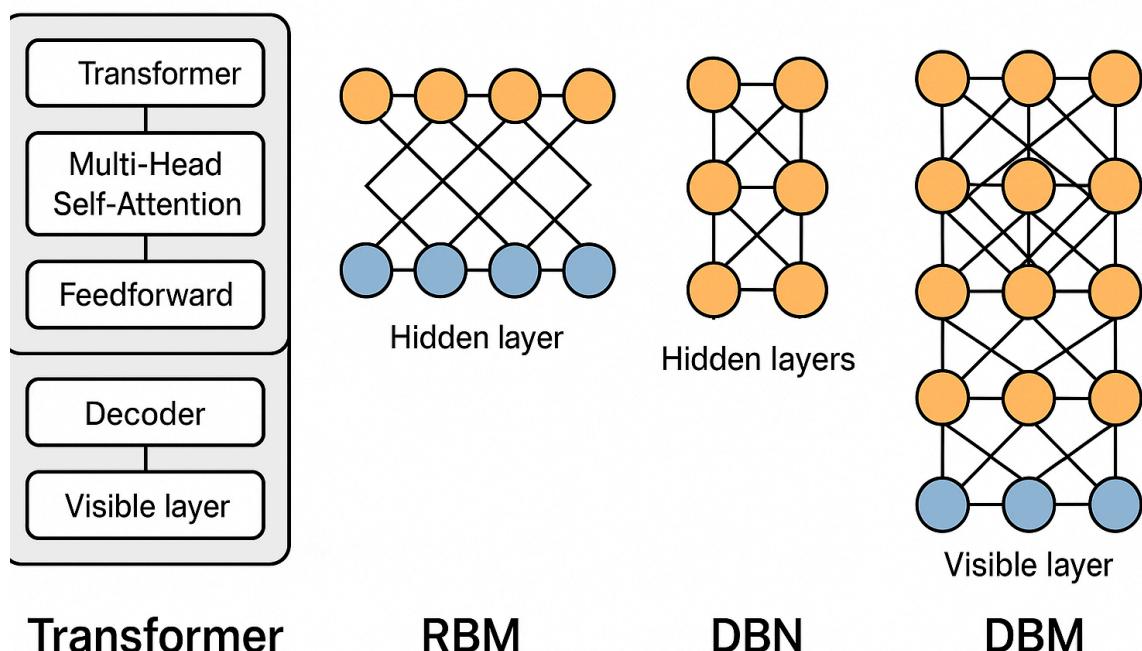
7. 深度自编码器（Deep Autoencoder）

计算流程：

- 编码器：高维输入 → 多层压缩 → 低维表示（瓶颈层）。
- 解码器：低维表示 → 重构原始输入。
- 损失函数：最小化输入与重构之间的误差。

理解：

一种无监督学习结构，适用于降维、特征提取、去噪、预训练等任务。可以看作是压缩器+解压器。



Chapter2 CNN

卷积神经网络

卷积神经网络结构

卷积神经网络一般由卷积层（含激活函数）、池化层、全连接层和输出层构成，其中卷积层与池化层一般交替排列，之后接一层或者多层全连接层，最后是输出层。

卷积层

卷积运算

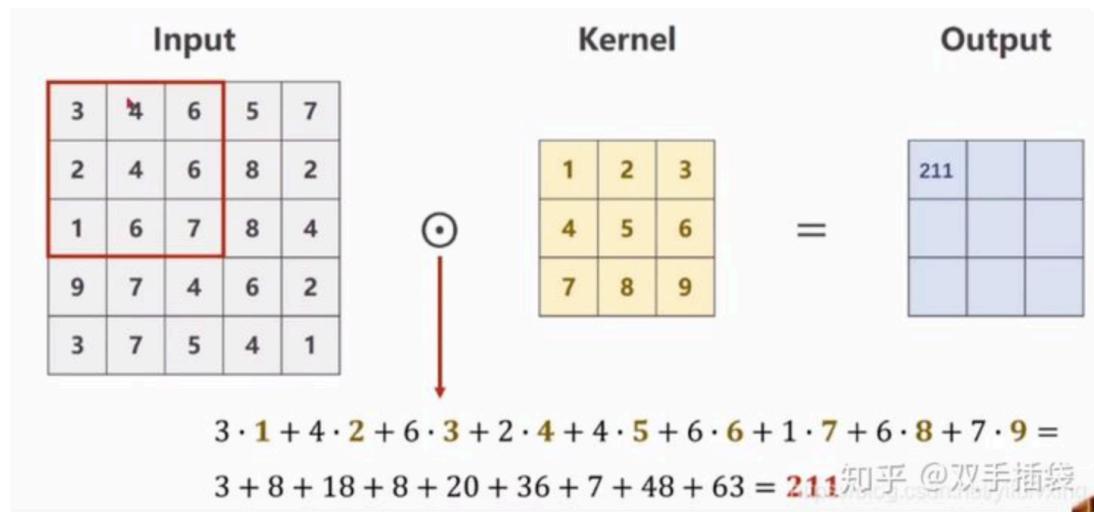
- 卷积运算（Convolution）是数学中的常见运算，分为离散卷积与连续卷积

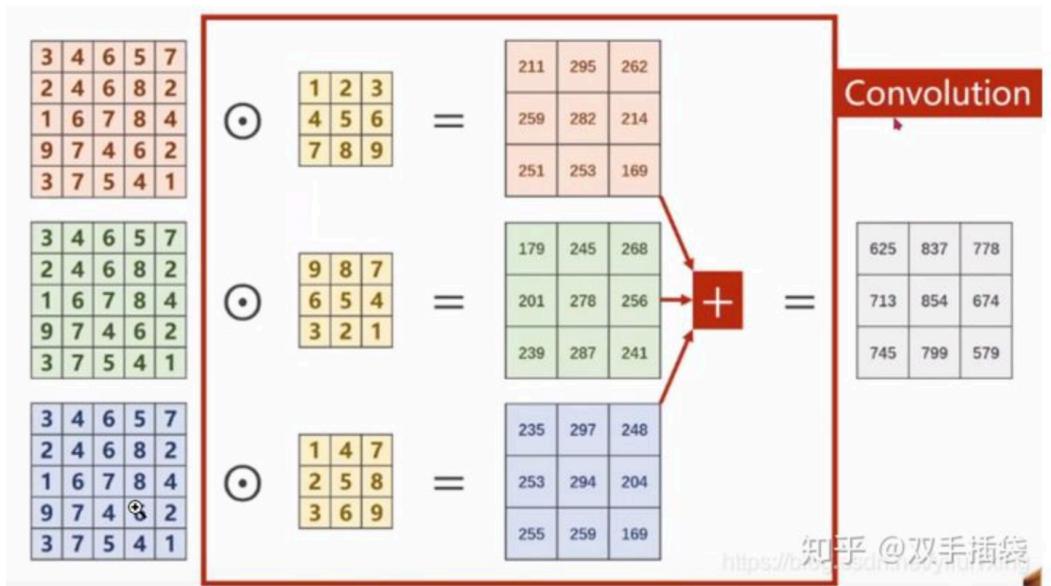
卷积公式

- 一维连续卷积： $(f * g)(n) = \int_{-\infty}^{\infty} f(\tau)g(n - \tau)d\tau$
- 一维离散卷积： $(f * g)(n) = \sum_{\tau=-\infty}^{\infty} f(\tau)g(n - \tau)$
- 二维离散卷积： $(f * g)(x, y) = \sum_{\tau_1=-\infty}^{\infty} \sum_{\tau_2=-\infty}^{\infty} f(\tau_1, \tau_2)g(x - \tau_1, y - \tau_2)$
- 二维连续卷积： $(f * g)(x, y) = \int_{\tau_1=-\infty}^{\infty} \int_{\tau_2=-\infty}^{\infty} f(\tau_1, \tau_2)g(x - \tau_1, y - \tau_2)d\tau_1 d\tau_2$

- 卷积运算：给定二维图像 I 与二维卷积核 K ，我们将其看作矩阵，根据上述二维离散卷积运算公式，图像 I 与卷积核 K 的卷积运算可表示为：

$$(I * K)(u, v) = \sum_i \sum_j I_{i,j} K_{(u-i),(v-j)}$$





卷积是点乘喔，多通道是最后结果相加



卷积翻转问题

出现这种差异的本质原因是：数学中的卷积和卷积神经网络中的卷积严格意义上是两种不同的运算

1. 上文可以清晰地看到数学中卷积运算的特点：

卷积核与原始的矩阵乘积，是围绕着中心元素进行180度旋转后，才是对应的元素。

2. 卷积神经网络的卷积本质上是一种spatial filter 为什么这么说呢？

我们来看两种图像空间滤波的常见操作

a) 平滑滤波

b) 边缘提取

这两种事情，很容易通过设计特定的“卷积核”，然后将其与像素矩阵的对应元素（不进行上述的旋转）相乘得到。

3. 此“卷积”与彼“卷积”的联系与区别：

最直观的就是：是否进行翻转，然后再进行对应元素的加权求和。

Sobel梯度算子 45° 方向模板被设计为：

$$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 0^* & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

数据填充

如果我们有一个 $n \times n$ 的图像，使用 $f \times f$ 的卷积核进行卷积操作，在进行卷积操作之前我们在图像周围填充 p 层数据，输出的维度：

$$(n - f + 1) \times (n - f + 1) \rightarrow (n + 2p - f + 1) \times (n + 2p - f + 1)$$

- 在使用PyTorch等深度学习框架时，卷积层有Padding参数，有三种选择：'Full'、'Valid'和'Same'。'Full'表示需要填充，当卷积核与输入开始接触时进行卷积操作，'Valid'表示不需要填充，'Same'表示需要填充并保证输出与输入具有相同的尺寸。

步幅

如果我们有一个 $n \times n$ 的图像，使用 $f \times f$ 的卷积核进行卷积操作，在进行卷积操作之前我们在图像周围填充 p 层数据，步幅为 s 。输出的维度为：

$$\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor$$

- 通道 (Channel)：一般指的图像的颜色通道。
- 单通道图像：一般指的灰度图像。
- 多通道图像：一般指的基于RGB的图像，有R、G、B三个通道。
- 特征图 (Feature map)：经卷积和激活函数处理后的图像。

单通道卷积/多通道卷积

- 单通道卷积：单通道图像的卷积。
- 单卷积核单通道卷积
- 多卷积核单通道卷积
- 多通道卷积：多通道图像的卷积。
- 单卷积核多通道卷积（一般不这样做）
- 多卷积核多通道卷积（求和加偏置bias）

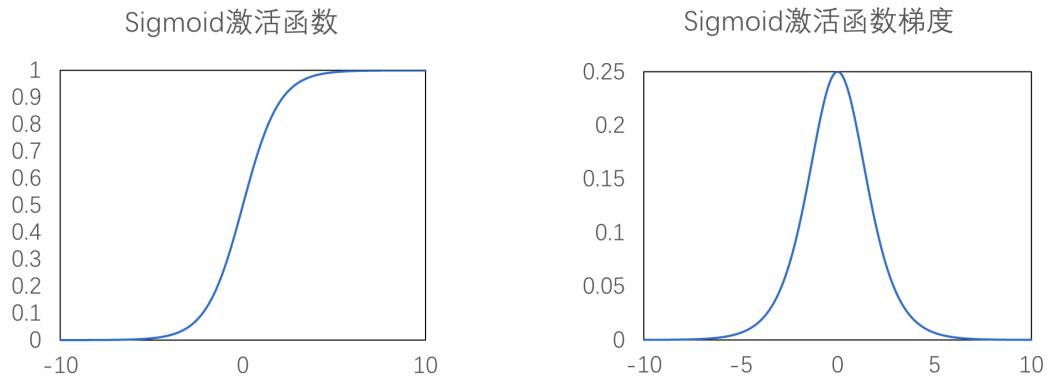
卷积作用：提取特征

激活函数

- 激活函数的引入，增强了人工神经网络的非线性表达能力，从而提高了模型的学习能力。
- 在人工神经网络发展的初期，Sigmoid激活函数起到了十分重要的作用，但随着人工神经网络层数的增加以及反向传播算法的使用，会产生梯度消失问题。
- 在卷积神经网络中，为了缓解梯度消失问题，常用的激活函数有ReLU、PReLU、ERU和Maxout等

Sigmoid

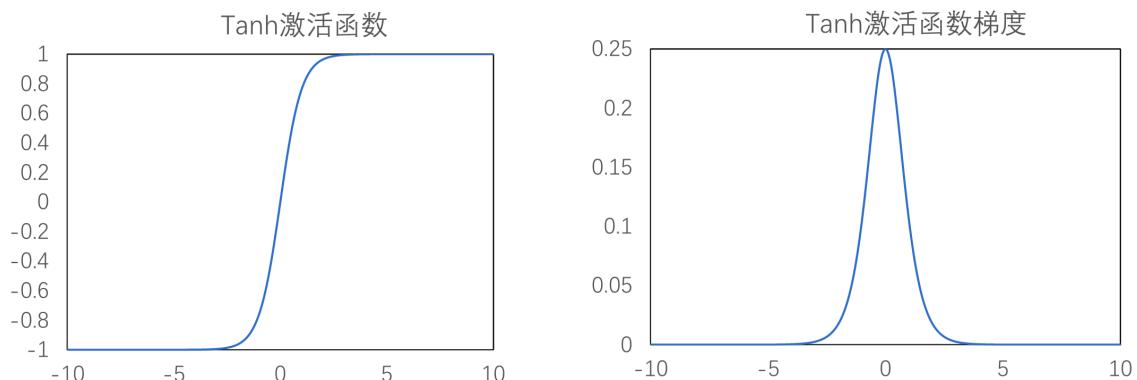
- Sigmoid激活函数的定义: $\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}$



问题：Sigmoid激活函数存在“梯度饱和效应”问题，即Sigmoid激活函数两端梯度都趋于0，因此在使用误差反向传播算法进行网络训练时，该区域的误差无法传递到前一层，从而导致网络训练失败。

Tanh

- Tanh激活函数的定义: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

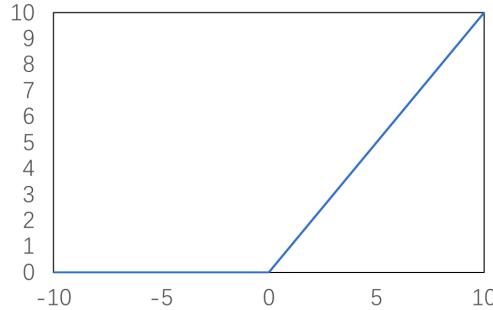


问题：和sigmoid一样问题

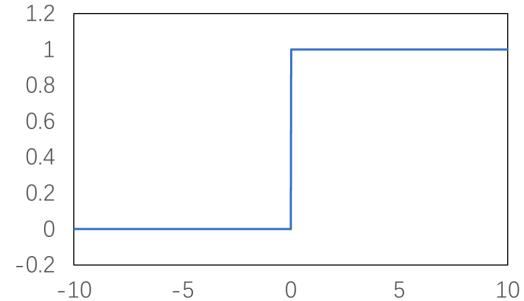
ReLU

- ReLU激活函数的定义: $\text{ReLU}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$

ReLU激活函数



ReLU激活函数梯度

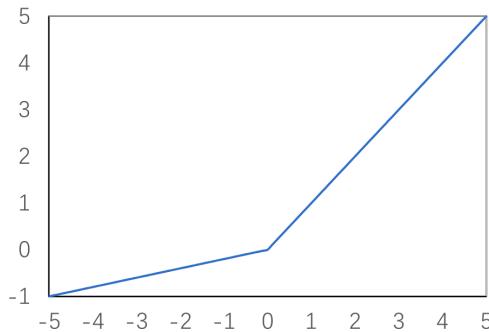


- 与Sigmoid激活函数相比, ReLU在 $x \geq 0$ 部分消除了“梯度饱和 效应”, 且ReLU的计算更简单, 计算速度更快。
- 但ReLU本身也存在缺陷, 如果输入为负值, 其梯度等于0, 导致“神经元死亡”, 将无法进行权重更新, 进而无法完成网络训练。

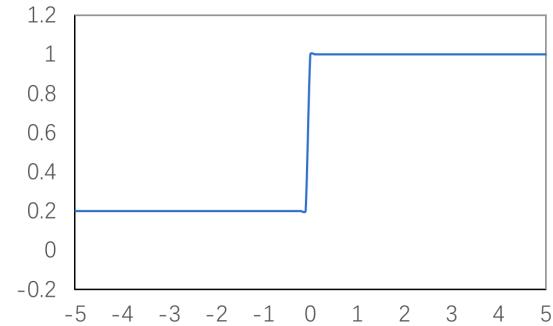
PReLU/Leaky ReLU ($\alpha=0.01$)

- PReLU激活函数的定义: $f(x) = \begin{cases} \alpha x, & x < 0 \\ x, & x \geq 0, x \in (-\infty, +\infty) \end{cases}$
- 其中 α 是一个需要学习的参数, 当指定 $\alpha = 0.01$ 时, 又称为Leaky ReLU, 当 α 通过高斯分布随机产生时, 又称为随机化的ReLU (Randomized ReLU, RReLU)

PReLU激活函数



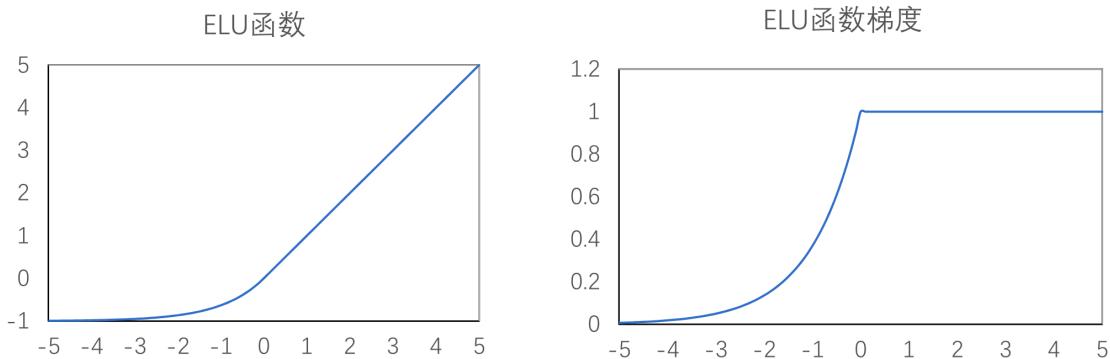
PReLU激活函数梯度



- PReLU激活函数的优点是比Sigmoid激活函数收敛快, 解决了 ReLU激活函数的“神经元死亡”问题。
- PReLU激活函数的缺点是需要再学习一个参数 α , 工作量变大。

ELU

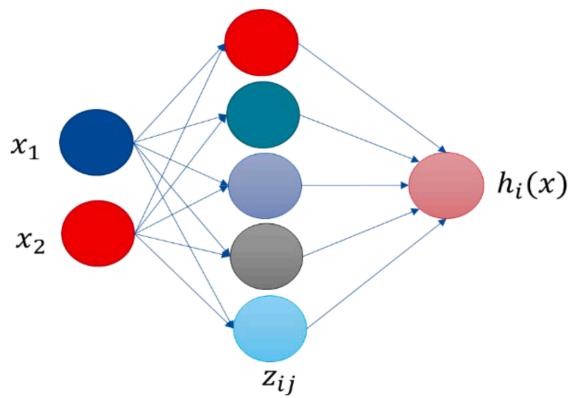
- ELU激活函数的定义: $f(x) = \begin{cases} \alpha(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases}, x \in (-\infty, +\infty)$



- ELU 激活函数的优点是处理含有噪声的数据有优势，与 Sigmoid激活函数相比更容易收敛。
- ELU激活函数的缺点是计算量较大，与ReLU激活函数相比，收敛速度较慢。

Maxout

- Maxout激活函数实际上是增加了一层全连接神经网络，神经元个数 k 自定，其函数定义为: $h_i(x) = \max_{j \in [1, k]} z_{ij}$, 其中 $z_{ij} = x^\top w_{ij} + b_{ij}$



- Maxout激活函数的优点是能够缓解梯度消失问题，规避了 ReLU激活函数“神经元死亡”的缺点。
- Maxout激活函数的缺点是增加了一层神经网络，无形中增加了参数和计算量。
- 卷积神经网络中的激活函数选择
 - CNN 在卷积层尽量不要使用Sigmoid和Tanh，将导致梯度消失。
 - 首先选用ReLU，使用较小的学习率，以免造成神经元死亡的情况。
 - 如果ReLU失效，考虑使用Leaky ReLU、PReLU、ELU或者 Maxout，此时一般情况都可以解决。

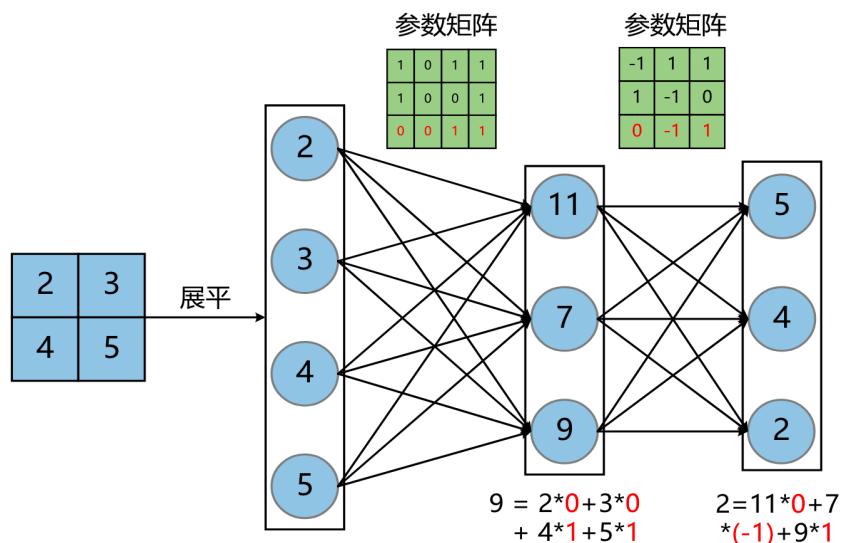
池化层

- 池化操作使用某位置相邻输出的总体统计特征作为该位置的输出，常用最大池化（max-pooling）和均值池化（averagepooling）。
- 池化层不包含需要训练学习的参数，仅需指定池化操作的核大小、步幅以及池化类型。
- **池化层的作用**
 - 对输入对象进行“降采样（Downsampling）”操作，一定程度上提高了模型的容错能力
 - 保证了当输入出现少量平移时，输出近似不变，增强了网络对输入图像中的小变形、扭曲、平移的鲁棒性（输入里的微小扭曲不会改变池化输出）
 - 池化核的指定相当于在空间范围内对特征图的特征进行了维度约减，同时缩小了下一层输入的特征图尺寸，进而一定程度上减少了网络的参数个数和计算量。

全连接层

- 全连接层一般由一到多层的全连接神经网络组成，功能是对卷积层或池化层输出的特征图（二维）进行降维。

会在训练中进行更新



全连接层作用：

- 可以将不同的区域特征合并为一个完整的特征。

输出层

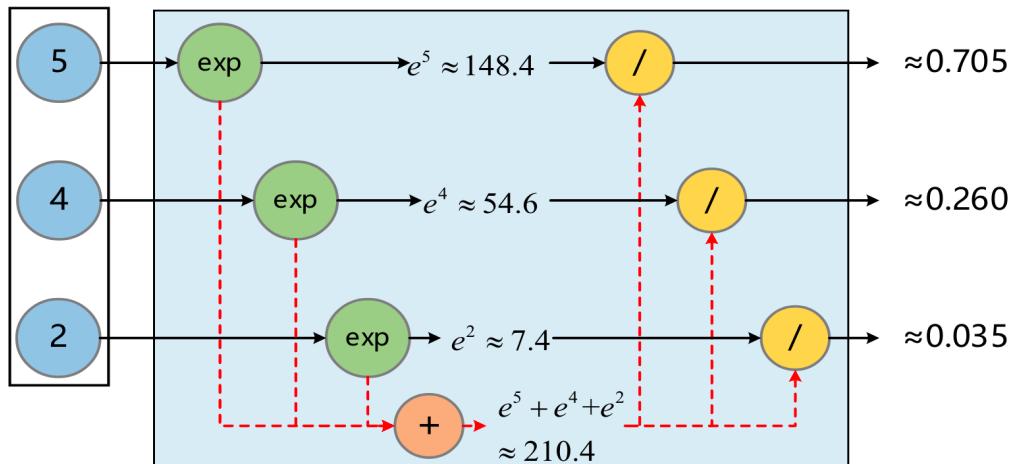
- 对于分类问题：使用Softmax函数

$$y_i = \frac{e^{z_i}}{\sum_{i=1}^n e^{z_i}}$$

..

- 对于回归问题：使用线性函数

$$y_i = \sum_{m=1}^M w_{im} x_m$$



卷积神经网络的训练

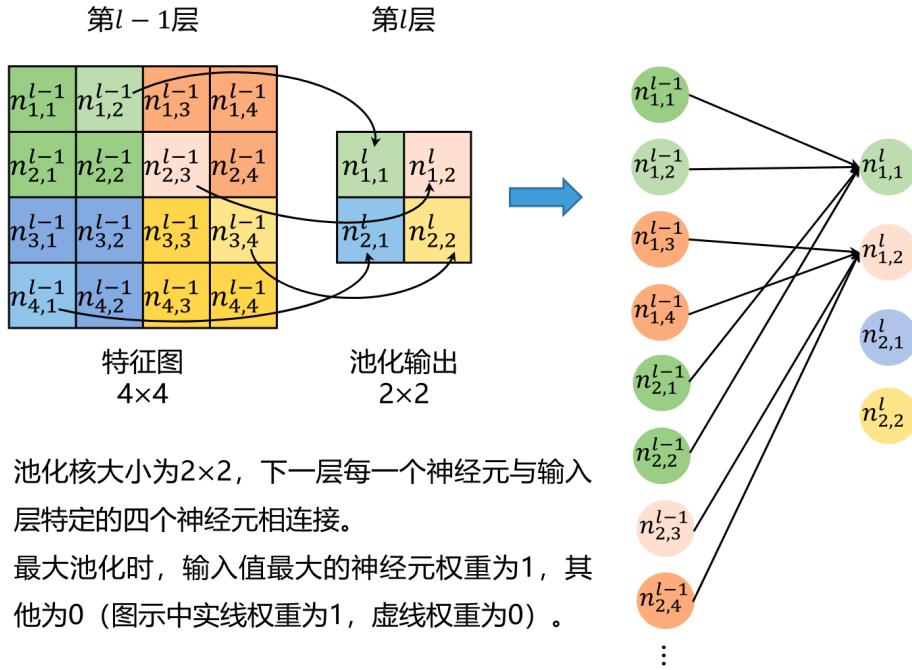
- 以图像分类任务为例

- 用随机数初始化网络需训练的参数（如权重、偏置）。
- 将训练图像作为输入，进行卷积层、ReLU、池化层以及全连接层的前向传播，并计算每个类别的对应输出概率。
- 计算输出层的总误差：总误差=- Σ (目标概率 $\times \log(\text{输出概率})$)。
- 使用BP算法计算总误差相对于所有参数的梯度，并用梯度下降法或其他优化算法更新所有参数的值，以使输出误差最小。

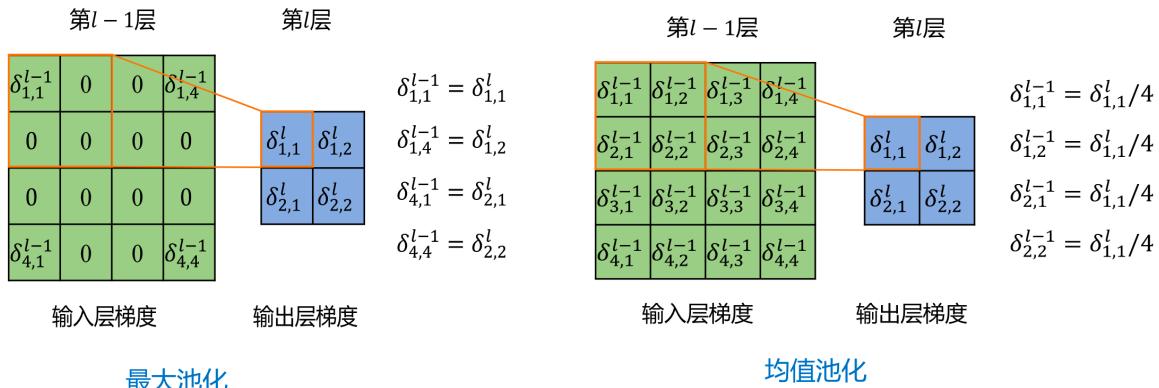
卷积核个数、卷积核大小以及网络架构，是在步骤1之前就已经确定的，且不会在训练过程中改变，只有网络的其他参数，如神经元的权重、偏置会更新。

- 池化层的训练：

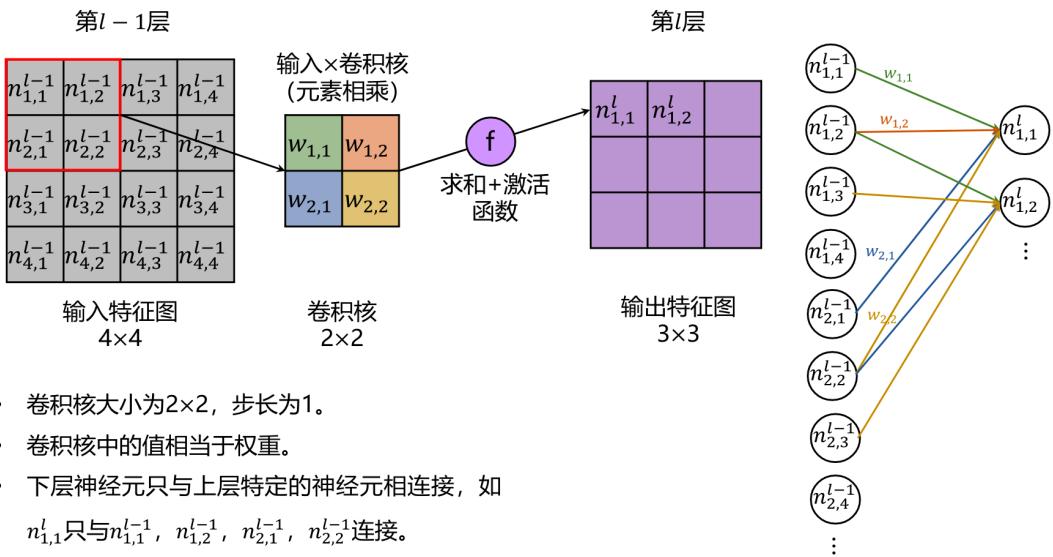
把池化层改为多层神经网络的形式。



最大池化和均值池化的误差反向传播。



- 卷积层的训练：
- 首先把卷积层也改为多层神经网络的形式，之后使用BP算法进行训练。



- 卷积核大小为 2×2 , 步长为1。
- 卷积核中的值相当于权重。
- 下层神经元只与上层特定的神经元相连接, 如
 $n_{1,1}^l$ 只与 $n_{1,1}^{l-1}, n_{1,2}^{l-1}, n_{2,1}^{l-1}, n_{2,2}^{l-1}$ 连接。

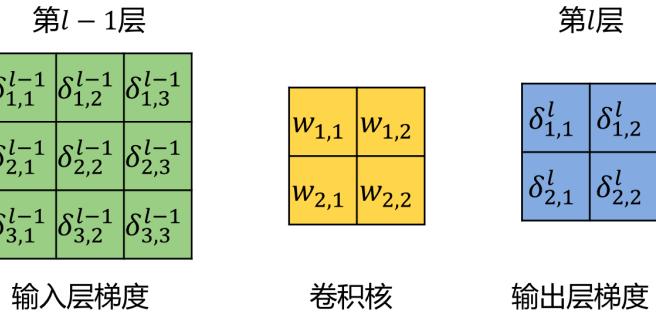
- 卷积层的输出：

- 卷积层的输出： $\text{net}^l = \text{conv}(\mathbf{W}^l, \mathbf{a}^{l-1}) + W_b^l, a_{i,j}^l = f^l(\text{net}_{i,j}^l)$
 - 下标 i, j 分别表示输入层输入的行索引与列索引
 - \mathbf{W}^l 为卷积核参数
 - W_b^l 为偏置参数
 - \mathbf{a}^{l-1} 为 $(l-1)$ 层的输出
 - conv 为卷积运算
 - f^{l-1} 为 $(l-1)$ 层的激活函数

- 误差项计算：

- **误差项计算：** $\delta_{i,j}^{l-1} = \frac{\partial E}{\partial net_{i,j}^{l-1}} = \frac{\partial E}{\partial a_{i,j}^{l-1}} \frac{\partial a_{i,j}^{l-1}}{\partial net_{i,j}^{l-1}} = \frac{\partial E}{\partial a_{i,j}^{l-1}} f'(net_{i,j}^{l-1})$

- 其中 m, n 分别表示卷积核中值的行索引与列索引
- M, N 分别表示卷积核的高度与宽度，在本节示例， $M=N=2$



经典卷积神经网络

LeNet -5

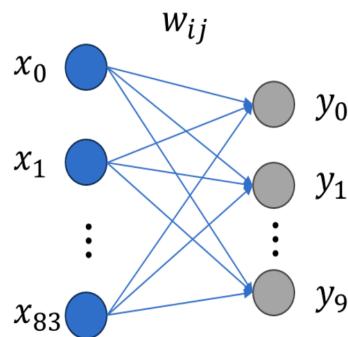
手写数字识别和英文字母识别

输入层 $\rightarrow C1 \rightarrow S2 \rightarrow C3 \rightarrow S4 \rightarrow C5 \rightarrow F6 \rightarrow Output$

输出层采用径向基函数（Radial Basis Function, RBF）的网络连接方式。

径向基函数的公式

径向基函数的公式： $y_i = \sum_j (x_j - w_{ij})^2$ ，其中 w_{ij} 的值由 i 的比特图编码确定， i 从 0 到 9， j 取值从 0 到 $7 \times 12 - 1$ 。RBF 输出的值越接近于 0，表示当前网络输入的识别结果与字符 i 越接近。



73

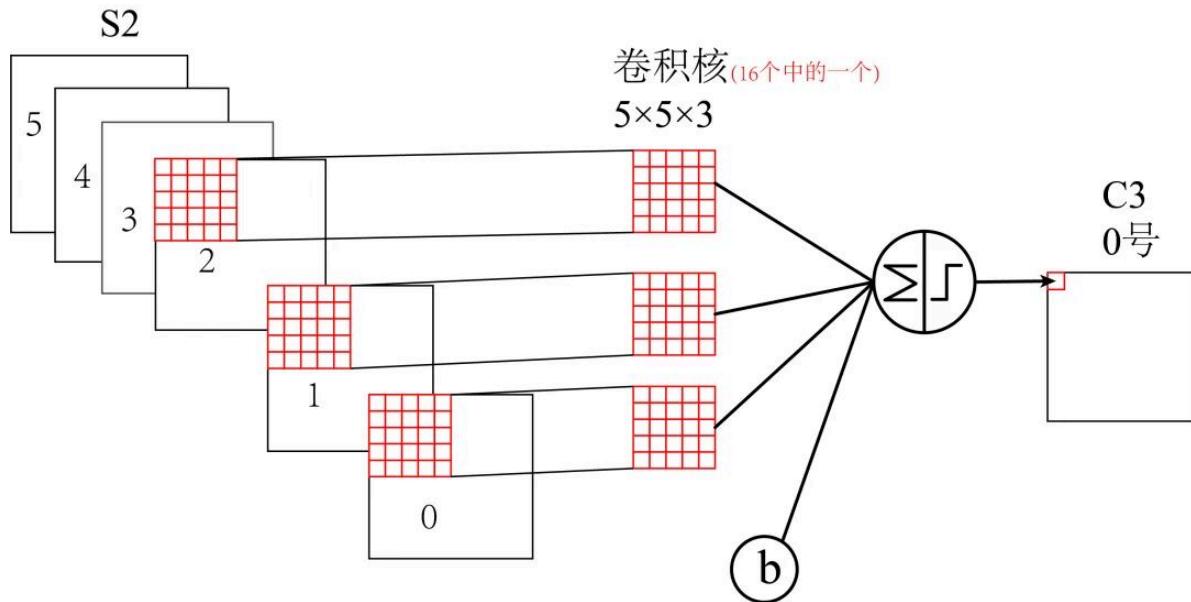
特别的：C3 卷积层

16个 5×5 卷积核组（前6个卷积核组中的卷积核个数为3，中间6个为4，之后3个为4，最后1个为6）

16: 输出通道数

5×5: 卷积核大小

前6个：输入通道数



非对称连接结构 (partial connectivity)，通过“有选择地连接”，来：

- 减少参数数量
- 增加特征的多样性
- 避免过拟合
- 增强对局部结构组合的学习能力

AlexNet

- 首次成功应用ReLU作为CNN的激活函数。
- 使用Dropout丢弃部分神经元，避免了过拟合。

Dropout：随机的断开全连接层某些神经元的连接，通过不激活某些神经元的方式防止过拟合。



Dropout应该算是AlexNet中一个很大的创新，现在神经网络中的必备结构之一。Dropout也可以看成是一种模型组合，每次生成的网络结构都不一样，通过组合多个模型的方式能够有效地减少过拟合，Dropout只需要两倍的训练时间即可实现模型组合（类似取平均）的效果，非常高效。

- 使用重叠MaxPooling（让池化层的步长小于池化核的大小），一定程度上提升了特征的丰富性。
- 首次使用CUDA加速训练过程。
- 使用了数据增强，原始图像大小为 256×256 的原始图像中重复截取 224×224 大小的区域，大幅增加了数据量，大大减轻了过拟合，提升了模型的泛化能力。

- 对图像数据通过**主成分分析方法**进行降维处理。

输入层 → 第一个卷积层（含池化）→ 第二个卷积层（含池化）→ 第三层（卷积层）→ 第四层（卷积层）→ 第五层（卷积层）→ 第一个全连接层 → 第二个全连接层 → 第三个全连接层 → 输出
(Softmax)

VGGNet/VGG-16

两个卷积核大小为3的卷积层串联后的感受野尺寸为5，相当于单个卷积核大小为 5×5 的卷积层。

3个 $3 \times 3 \rightarrow 7 \times 7$

两者参数数量比值为 $(233)/(5 \times 5) = 72\%$ ，前者参数量更少。

为什么需要激活函数？

卷积层本质上是一个**线性变换**（加权求和 + 偏置），如果你连续堆叠多个没有激活函数的卷积层，就相当于**多个线性变换的组合，最终还是一个线性变换**。

这会导致神经网络的表达能力极弱，无法拟合复杂的非线性映射。

所以我们通常在**每个卷积层后面加上激活函数**，使网络具备**非线性建模能力**。

GoogleNet

- 它的主要思想是除了在网络深度上加深（22层）之外，在宽度上也加宽。
- GoogleNet的核心是Inception模块，Inception模块包含4个分支，每个分支均使用了 1×1 卷积，它可以跨通道组织信息，提高网络的表达能力，同时还可以对输出通道进行升维和降维。
- Inception模块中包含了 1×1 、 3×3 、 5×5 三种不同尺寸的卷积和1个 3×3 最大池化，增强了网络对不同尺度特征图的适应性。

GoogLeNet 提出了一种并联结构，下图是论文中提出的inception原始结构，将特征矩阵**同时输入到多个分支**进行处理，并将输出的特征矩阵**按深度进行拼接**，得到最终输出。

inception的作用：增加网络深度和宽度的同时减少参数。

在 inception 的基础上，还可以加上降维功能的结构。在原始 inception 结构的基础上，在分支2, 3, 4上加入了**卷积核大小为 1×1 的卷积层**，目的是为了降维（减小深度），减少模型训练参数，减少计算量。

ResNet

- 随着卷积网络层数的增加，误差的逆传播过程中存在的**梯度消失 和梯度爆炸**问题同样也会导致模型的训练难以进行。
- 甚至会出现随着网络深度的加深，模型在训练集上的训练误差会 出现先降低再升高的现象。
- 残差网络的引入则有助于解决梯度消失和梯度爆炸问题。

• 梯度消失和梯度爆炸问题原因

1. 深度神经网络训练问题

◆ 反向传播中梯度链式相乘：

| 神经网络中，误差通过链式法则从输出层向输入层逐层传递。

如果每一层的梯度 < 1 , 很多层相乘会 $\rightarrow 0$ (梯度消失) ; 如果 > 1 , 很多层相乘会 $\rightarrow \infty$ (梯度爆炸)

2. 激活函数问题



梯度消失和爆炸是深度网络训练中的核心挑战, 主要来源于**深层网络的链式乘法、激活函数的饱和性、权重初始化问题等**。使用**非饱和激活函数 (如ReLU)、残差结构、BatchNorm 和良好初始化策略**是解决这些问题的有效手段。

ResNet的核心是叫做残差块 (Residual block) 的小单元, 残差 块可以视作在标准神经网络基础上加入了跳跃连接 (Skip connection)。

- 提出residual结构 (残差结构), 并搭建超深的网络结构(突破1000层)
- 使用Batch Normalization加速训练(丢弃dropout)

卷积神经网络的主要应用

目标检测

定义：目标检测是指将图像或者视频中的目标物体用边框 (Bounding Box) 标记并识别出该目标物体的类别。

目前目标检测任务有两类模型：

- 一类是以区域卷积神经网络 (Region-CNN, R-CNN) 系列为代表的**两阶段模型**
- 一类是以YOLO系列为代表的一阶段模型

类型	流程阶段数	是否先提候选框	检测速度	精度	代表模型
两阶段模型	两步：候选区域 + 分类/回归	✓ 是的, 先提取候选框 (Region Proposals)	慢	高	R-CNN、Fast R-CNN、Faster R-CNN
一阶段模型	一步：直接从图像中预测结果	✗ 没有候选框步骤	快	略低 (但 YOLOv8 已接近)	YOLO (v1~v8)、SSD、RetinaNet

R-CNN系列

- 区域划分：给定一张输入图片，采用Selective Search 算法从图片 中提取 2000 左右类别独立的候选区域。

Selective Search：核心思想：图像中物体可能存在的区域应该有某些相似性或者连续性的，算法采用子区域合并的方法提取候选边界框。

- 特征提取：对于每个区域利用Alexnet抽取一个固定长度的特征向量。

- 目标分类：对每个区域利用 SVM 进行分类。

SVM：在样本空间中找到一个“最优超平面”，将不同类别的样本分开，且使分类间隔最大。

- 边框回归：使用Bounding box Regression (Bbox回归)进行边框坐标偏移优化和调整。

边框回归：是使得预测的边框尽可能与人工标定的边框越接近越好



为什么使用相对坐标差？

一句话概括：进行尺度归一化。

为什么宽高比要取对数(为什么不直接使用宽高比值来作为目标进行学习)？

一句话概括：保证缩放尺度的非负性。

要得到一个放缩的尺度，这里必须限制尺度大于0。那么，我们学习的 t_w, t_h 怎么保证满足大0呢？最直观的想法就是引入EXP函数，如公式(4)所示，那么反过来推导就是Log函数的来源了。

为什么IoU较大时边界框回归可视为线性变换？

在这里我们需要回顾下在高等数学中有关等价无穷小的结论：

$$\lim_{x \rightarrow 0} \frac{\log(1+x)}{x} = 1 \quad (6)$$

也就是说当 x 趋向于0时，我们可有 $\log(1+x) \approx x$ ，即可将 $\log(1+x)$ 近似看成线型变换。接下来我们将式(4)的后两个公式进行重写：

$$\begin{cases} t_w = \log \frac{G_w}{P_w} = \log \frac{G_w - P_w + P_w}{P_w} = \log \left(1 + \frac{G_w - P_w}{P_w}\right) \\ t_h = \log \frac{G_h}{P_h} = \log \frac{G_h - P_h + P_h}{P_h} = \log \left(1 + \frac{G_h - P_h}{P_h}\right) \end{cases} \quad (7)$$

也就是说，当 $\frac{G_w - P_w}{P_w}$ 和 $\frac{G_h - P_h}{P_h}$ 趋向于0时，即 $G_w \approx P_w$ 和 $G_h \approx P_h$ 时，

上式(6)可以近似将对数变换看成线性变换。 $G_w \approx P_w$ 和 $G_h \approx P_h$ 时候选目标框和真实目标框非常接近，即IoU值较大。按照RCNN论文的说法，IoU大于0.6时，边界框回归可视为线型变换。

CSDN @迪菲赫尔曼

AP (Average Precision)

AP衡量的是模型在一个类别上的整体表现。

步骤：

1. 按预测置信度从高到低排序。
2. 逐个计算 Precision 和 Recall。
3. 绘制 PR 曲线 (Precision vs Recall)。
4. 计算 PR 曲线下面积 → 得到 AP 值。

所以 AP 是对 PR 曲线积分的结果，代表模型对“某个类别”的检测能力。

mAP (mean Average Precision)

有多个类别时（如猫、狗、车、人等），我们对每个类别都计算一个 AP 值，然后对所有类别的 AP 做平均，得到：

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^N AP_i$$

其中 N 是类别数， AP_i 是第 i 个类别的 AP。

AP 衡量模型在一个类别上的检测能力；mAP 是多个类别上的平均检测性能，是目标检测模型综合能力的重要指标。



目标检测中，不只是类别对了，还要 **预测框的位置准确！**

因此：

- 对预测框是否算作“正确”，通常用 **IoU（交并比）** 来衡量。
- 例如：
 - 若预测框和真实框的 $\text{IoU} \geq 0.5$ ，就认为这是一个正确的 TP。
 - mAP@0.5 代表以 0.5 为阈值计算。
 - mAP@[0.5:0.95] 是 COCO 常用的更严格评估方式：平均多个 IoU 阈值下的 mAP。

SPPNet

解决 **固定输入尺寸限制** 和 **加快训练速度** 的问题。

它的关键创新是：**引入 SPP 层**，可以接收任意尺寸的输入图像，不再要求输入图像必须 resize 成固定大小（如 224×224 ）。

SPPNet 的解决方法是：

在最后一个卷积层之后，使用 **Spatial Pyramid Pooling 层**，将不同尺寸的 feature map 映射为固定长度的向量，之后再连接到全连接层。



但是为什么 CNN 需要固定的输入呢？

CNN 网络可以分解为卷积网络部分以及全连接网络部分。我们知道卷积网络的参数主要是卷积核，完全能够适用任意大小的输入，并且能够产生任意大小的输出。但是全连接层部分不同，全连接层部分的参数是神经元对于所有输入的连接权重，也就是说输入尺寸不固定的话，全连接层参数的个数都不能固定。

空间金字塔池化层

其主要目的是对于任意尺寸的输入产生固定大小的输出。思路是对于任意大小的feature map首先分成16、4、1个块，然后在每个块上最大池化，池化后的特征拼接得到一个固定维度的输出。以满足全连接层的需要。不过因为不是针对于目标检测的，所以输入的图像为一幅图像。

Fast R-CNN

使用Softmax分类替换R-CNN中的SVM分类

- 将候选框目标分类与边框回归同时放入全连接层，形成一个多任务学习（Multi-task Learning）模型，设计了联合损失函数，将 Softmax分类、边框回归一起训练。
 - 添加感兴趣区域池化（Region of Interest Pooling, ROI Pooling）层，实现了不同大小区域特征图的池化。
 - 训练时所有的特征存在缓存中，不再存到硬盘上，提升了速度。
 - (1) 输入图像；
 - (2) 通过深度网络中的卷积层（VGG、Alexnet、Resnet等中的卷积层）对图像进行特征提取，得到图片的特征图；
 - (3) 通过选择性搜索算法得到图像的兴趣区域（通常取2000个）；
 - (4) 对得到的
- 感兴趣区域进行ROI pooling（感兴趣区域池化）**：即通过坐标投影的方法，在特征图上得到输入图像中的感兴趣区域对应的特征区域，并对该区域进行最大值池化，这样就得到了感兴趣区域的特征，并且统一了特征大小，如图2所示；
- (5) 对ROI pooling层的输出（及感兴趣区域对应的特征图最大值池化后的特征）作为每个感兴趣区域的特征向量；
 - (6) 将感兴趣区域的特征向量与全连接层相连，并定义了多任务损失函数，分别与softmax分类器和boxbounding回归器相连，分别得到当前感兴趣区域的类别及坐标包围框；
 - (7) 对所有得到的包围框进行非极大值抑制（NMS），得到最终的检测结果。



ROI pooling

它的输入是特征图，输出则是大小固定的channel \times H \times W的vector。ROI Pooling是将一个个大小不同的region proposals，映射成大小固定的(W \times H)的矩形框。它的作用是根据region proposals的位置坐标在特征图中将相应区域池化为固定尺寸的特征图，以便进行后续的分类和输出回归框操作。它可以加速处理速度。

ROI Pooling有两个输入，一个是图片进入CNN后的特征图，另一个是区域的边框。ROI的输出是一个region_nums \times channels \times W \times H的向量。

ROI可以看成是SPP的简化版本，原版SPP是多尺度池化后进行concat组成新特征，而ROI只使用一个尺度，可以将任意维度的特征矩阵缩放成固定维度。论文中的具体做法是，把高和宽都平均分为7*7的小块，然后在每一个小块做max pooling操作，channel维度不变，这样做能使输出维度固定，同时ROI Pooling不是多尺度的池化，梯度回传非常方便，为fine-tune卷积层提供了条件。（SPP Net不能fine-tune卷积层）

使用了多任务的损失函数来简化R-CNN中的多阶段训练：

联合损失函数

Multi-task loss

分类损失 边界框回归损失

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v)$$

p 是分类器预测的softmax概率分布 $p = (p_0, \dots, p_k)$

u 对应目标真实类别标签

t^u 对应边界框回归器预测的对应类别 u 的回归参数 $(t_x^u, t_y^u, t_w^u, t_h^u)$

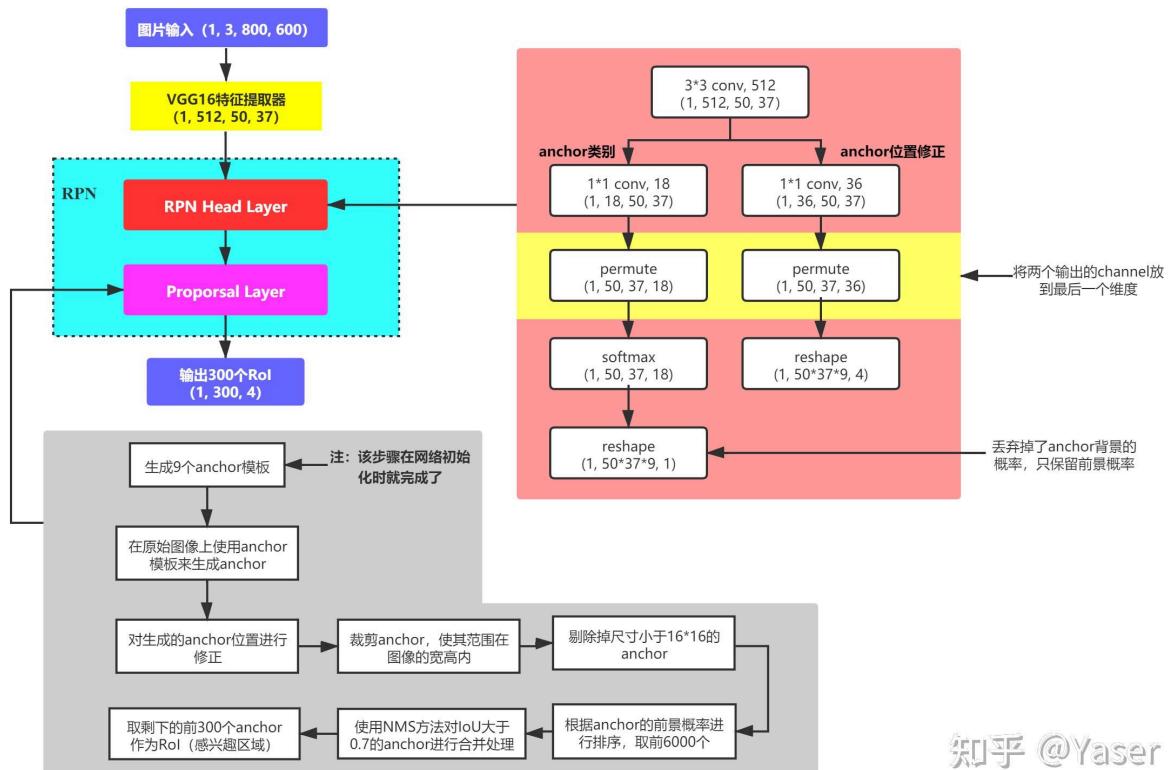
v 对应真实目标的边界框回归参数 (v_x, v_y, v_w, v_h)



Faster R-CNN

RPN (Region Proposal Network)：使用全卷积 神经网络来生成区域建议（Region proposal），替代之前的Selective Search。

RPN用 于生成region proposals。它通过Softmax判断区域是有目标的 正例（Positive）还是没有目标的负例（Negative），再利用边框回归获得候选区域。





RPN Head主要负责anchor的生成、anchor位置偏移量预测以及anchor的类别判断；
Proposal负责对生成的anchor进行进一步的筛选，将筛选后的anchor作为RoI输入到后续的
网络中。

需要注意的是，因为不管是anchor的类别预测还是位置偏移量预测，这些都是在channel里
面的，所以对这两个输出都需要使用pytorch的permute函数来将维度进行换位。

Faster R-CNN算是RCNN系列算法的最杰出产物，也是two-stage中最为经典的物体检测算法。

- 第一阶段生成图片中待检测物体的anchor矩形框（对背景和待检测物体进行二分类）
- 第二阶段对anchor框内待检测物体进行分类。

Faster RCNN可以看作 RPN+Fast RCNN，其中RPN使用CNN来生成候选区域，并且RPN网络可以认
为是一个使用了注意力机制的候选区域选择器。

具体算法步骤如下

1. 输入图像到特征提取器中，得到整张图片的feature map。
2. 使用RPN生成候选框，并投影到feature map上，得到每一个候选区域的特征矩阵。
3. 将每一个特征矩阵经过ROI Pooling缩放到7*7大小，然后经过展平处理后通过全连接层获得预测的
分类以及候选区域位置偏移信息。

Faster R-CNN训练方式

Faster R-CNN 在训练时由于引入了 **Region Proposal Network (RPN)**，需要同时训练两个子模块：

- **RPN：生成候选区域 (Region Proposal)**
- **Fast R-CNN：对候选区域进行分类和边界框回归**
- Alternating training



交替地训练 RPN 和 Fast R-CNN，而不是一起训练。

1. 先训练 RPN：

- 初始化一个 CNN（如 VGG16）；
- 用它训练一个 RPN 网络，学习如何生成候选框。

2. 再训练 Fast R-CNN：

- 用 RPN 生成的 proposals；
- 训练 Fast R-CNN 模块进行分类和回归。

3. Fine-tune RPN（共享特征）：

- 固定共享的卷积层，只训练 RPN 相关部分；

4. Fine-tune Fast R-CNN（共享特征）：

- 同样固定共享层，微调 Fast R-CNN。
- 优点：结构清晰，易于实现。
- 缺点：训练过程繁琐，需要多次切换；训练时间长。

- Approximate joint training



在一次前向传播中，同时训练 RPN 和 Fast R-CNN，两者共享前面的卷积层，称为“近似”联合训练。

- RPN 在前向传播时生成 proposals；
- 然后这些 proposals 输入到 Fast R-CNN 中；
- 但 反向传播时，RPN 生成 proposals 的过程不参与梯度回传（即候选框位置不变，固定使用）。

所以称之为 **Approximate**：RPN 的 proposal 生成过程是非可导的，跳过。

- 优点：训练更快、更简洁；
- 缺点：不是真正的联合训练，略影响精度；
- 通常用于 实际部署与实现。

- Non-approximate joint training



真正地将 RPN 和 Fast R-CNN 完整地端到端训练，包括 **proposal 生成过程** 也参与反向传播和梯度更新。

- Proposal 是根据 RPN 的回归结果生成的（如 anchor 的偏移），涉及非连续的操作（如 NMS、ROI Align）；
- 所以要实现反向传播，需要对 **proposal 生成过程** 做“可导近似”，比较复杂。

特点

- 优点：真正端到端训练，理论精度最高；
- 缺点：工程实现难度大，开销大；
- 论文中未实现，只是提出了方向，后来部分模型（如 Cascade R-CNN）朝这个方向改进。

YOLO系列

YOLO作为一种单阶段目标检测器，能够在一次前向传播中完成图像中所有目标的检测与分类任务，相比于其他两阶段的目标检测方法，如Faster R-CNN，YOLO的实时性和检测效率显得尤为突出。正因如此，YOLO迅速成为了实时目标检测领域的主流方法，并在多个应用场景中得到了成功实践。

YOLO，即You Only Look Once，是一种基于深度学习的实时目标检测算法。

核心理念：将目标检测任务转化为一个单一的回归问题，从输入图像直接预测目标的类别和边界框。

YOLO的关键在于其**单阶段的设计**，这与传统的两阶段检测器形成鲜明对比，两阶段检测器，如Faster R-CNN，首先生成候选区域，然后对这些区域进行分类和边界框回归，虽然这种方法在精度上有优势，但速度相对较慢。

YOLO的设计简化了这一流程，使得目标检测的速度得到了极大的提升。

YOLO模型将输入图像划分为 $S \times S$ 的网格，每个网格单元负责预测多个边界框及其对应的类别置信度，模型的输出是一系列边界框、类别标签以及对应的置信度分数。YOLO通过一次性处理整个图像，生成所有目标的检测结果，这种方法显著提高了检测速度。



模型结构（基于 GoogLeNet 修改版）

- 输入图像： 448×448
- 特征提取网络：24 层卷积 + 2 层全连接（借鉴 GoogLeNet）
- 输出层： $S \times S \times (B \times 5 + C)$
 - $S \times S$ ：将图像划分为 $S \times S$ 个网格（YOLOv1 中 $S = 7$ ）
 - 每个网格负责检测目标中心点落在其内部的目标
 - B ：每个网格预测 B 个 bounding box（YOLOv1 中 $B=2$ ）
 - 每个 box 预测 5 个值： $x, y, w, h, confidence$
 - C ：类别数量（例如 VOC 数据集中 $C=20$ ）

输出维度举例（VOC数据集）：

最终输出是一个张量：

复制编辑

$$7 \times 7 \times (2 \times 5 + 20) = 7 \times 7 \times 30$$

流程：

- 图像输入网络，提取高维特征。
- 网络输出一个固定大小的张量。
- 对每个网格单元：
 - 预测多个边界框和置信度。
 - 同时预测该网格内目标的类别概率。
- 综合置信度和类别概率，经过 NMS 筛选出最终目标框。

YOLOv2

- **Batch normalization:** 在每一个卷积层后添加batch normalization，极大的改善了收敛速度同时减少了对其它regularization方法的依赖（舍弃了 dropout优化后依然没有过拟合），使得mAP获得了2%的提升。



Batch Normalization 是一种对 **每一层神经网络的输入做标准化处理** 的方法，它的基本操作是：

对每一层的 mini-batch 输出做归一化（均值为 0，方差为 1），再引入两个可学习参数（缩放因子 γ 和平移因子 β ）恢复表达能力。

数学表达如下：

$$\hat{x}^{(i)} = \frac{x^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad y^{(i)} = \gamma \hat{x}^{(i)} + \beta$$

YOLOv2 的网络结构（Darknet-19）中：

- 在 **每一层卷积层之后都加入了 BN 层**（即 conv → BN → LeakyReLU）
- 并将以往的 dropout 层完全去除
- 输出层除外（避免干扰置信度预测）

1. 加速训练收敛速度

- 通过标准化激活值，BN 减少了不同层之间的“协变量偏移”（Covariate Shift）。
- 这样可以使用**更大学习率**，收敛更快。

YOLOv2 使用 BN 后训练速度提升了 2 倍以上。

2. 提高模型泛化能力

- 在训练时使用 mini-batch 的均值和方差，实际上起到了一定的正则化作用。
- 类似于 dropout 的效果，可以减少 overfitting。

3. 降低对初始化的依赖

- 不再像以前那样对权重初始化敏感，大大简化了调参过程。

4. 去除 Dropout 的必要性

- YOLOv2 中完全移除了 dropout 层，仅依靠 BN 就实现了良好的正则效果。

5. 提高检测精度（mAP 提升）

- 在 YOLOv2 中，**单独加入 BatchNorm** 就可以 mAP 提升 2% 以上，这在目标检测任务中是非常可观的。

📌 1. 标准化 (Normalization)

对 每个 mini-batch 中的数据计算均值和方差：

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i \quad \sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$$

- “ m : mini-batch 中的样本数量 (比如 64) ”
- “这一步是在 每个特征维度 (或卷积的通道) 上做的”

📌 2. 归一化 (Zero Mean, Unit Variance)

将每个样本标准化 (减均值除标准差) :

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

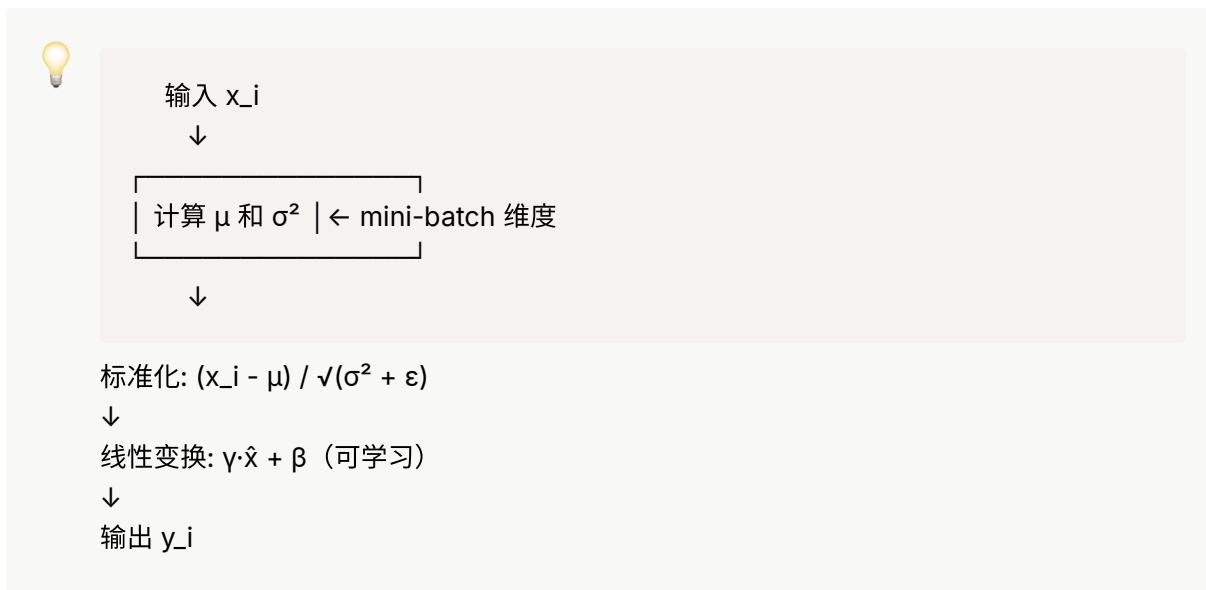
- “ ϵ : 是一个很小的常数，用于避免除以 0”
- “此时数据已经标准化为均值为 0、方差为 1”

📌 3. 缩放和平移 (可学习参数)

BN 并不直接使用标准化后的值，而是再加入两个可学习参数：

$$y_i = \gamma \cdot \hat{x}_i + \beta$$

- γ 是缩放系数
- β 是偏移 (平移) 系数
- 这两个参数是可以 通过反向传播学习的，保证网络保留足够的表示能力



- 分辨率修改：YOLOv2首先修改预训练分类网络的分辨率为448，在ImageNet数据集上训练10轮（10 epochs），mAP获得了4%的提升。

原始YOLO使用的是 224×224 的输入图像，后来在YOLOv2中将输入图像的分辨率改为 **448×448**，这样做带来的好处是：

🔍 更高分辨率带来更丰富的空间细节

- 高分辨率的图像中，小目标不会被压缩得太小，有更多像素用于表达。
- 这样网络提取到的特征就包含更准确的位置和边缘信息，有利于检测精度（mAP）提升。

🔧 在ImageNet分类任务上预训练10轮

- 在ImageNet上使用 448×448 训练分类网络，可以让网络提前适应高分辨率输入。
- 然后再迁移到目标检测任务上，效果比直接在检测任务中加大分辨率更稳定。
- YOLOv2中这样做，mAP提升了4%，说明这个策略非常有效。

- 多尺度训练：YOLO使用全连接层特征进行Bbox预测（要把14701的全连接层reshape为7730的最终特征），这会丢失较多的空间信息导致定位不准，丢弃全连接层使用RPN中的anchor box去做Bbox回归。

◆ 1. 多尺度训练 (Multi-scale Training)

YOLOv2每隔几轮训练就会随机改变输入图像的尺寸（从320到608之间的任意32倍数）。这有什么好处？

- 训练出来的模型就能适应不同分辨率的输入图像。
 - 增强网络的鲁棒性 (robustness)，测试时也能动态选择分辨率以兼顾速度和精度。
-

◆ 2. 丢弃全连接层，改用全卷积结构

原始YOLO中，最后的 bbox 是从全连接层生成的（比如 `reshape 1470 × 1 → 7 × 7 × 30`）。这其实会

- 破坏空间结构信息（全连接层打平后没有空间含义）
- 导致定位不准

YOLOv2 改进做法：

- 丢掉全连接层，改为全卷积网络 (Fully Convolutional Network, FCN)
- 最后几层用 1×1 卷积直接输出预测结果（包括 bbox、类别、置信度）

优点：空间结构保持得更好，定位更准确，速度也更快。

◆ 3. 引入 Anchor Boxes (借鉴Faster R-CNN的 RPN)

YOLOv2 不再让网络直接预测 bbox 的坐标（如中心点和长宽），而是像 Faster R-CNN 的 RPN 一样：

- 使用多个 预定义形状和大小的 anchor boxes
- 网络只需预测相对于这些 anchor 的偏移量 (dx, dy, dw, dh)

好处是：

- 提高了定位的灵活性
- 检测多个尺寸/比例的目标更容易

YOLO9000



WordTree

YOLO9000 的关键思想之一是使用 **WordTree**，这是一种建立在 **WordNet** 上的层次化语义结构。

解释：

- WordNet 是一个大型英语词汇数据库，它为单词建立了层级关系，比如：
 - 动物 → 哺乳动物 → 犬类 → 狗 → 边牧
 - 交通工具 → 陆地车辆 → 汽车 → SUV
- WordTree 利用这种父子层级关系，构建成树状的标签结构。不像传统分类任务中直接用一个 softmax 输出 1000 个类别，YOLO9000 会：
 - 分层输出概率（每层一个 softmax）
 - 预测一条路径，比如 animal → mammal → dog → German shepherd

好处：

- 分类类别可以扩展得很多（不局限于 COCO 的 80 个类）
- 支持 **细粒度识别**（例如不仅识别“狗”，还能识别“萨摩耶犬”）
- 实现**层次兼容**：如果某一层不确定，就返回上层，例如识别不出“德国牧羊犬”可以退而求其次返回“狗”。

联合训练

YOLO9000 同时用两类数据集训练模型：

数据类型	数据集	用于任务	特点
图像分类数据	ImageNet	分类任务	只有图像级标签，没有 bbox
检测数据	COCO/VOC	目标检测任务	有 bbox 和类别标签

联合训练流程：

- 两个输入源：
 - 从 ImageNet 输入图像：模型学习如何将图像分类为 WordTree 中的某个类别。
 - 从 COCO 输入图像：模型学习如何检测 bbox + 分类（也用 WordTree 标签）。
- 共享模型参数：
 - 两种数据共享 YOLO 的主干网络（特征提取部分）。
 - 在分类图像上优化分类损失，在检测图像上优化检测损失。
- 统一的分类头：
 - 不论分类还是检测，都统一到 WordTree 的 softmax 路径上。

YOLOv3

为什么要使用残差（Residual）神经网络？

答：网络越深，梯度消失的现象就越来越明显，网络的训练效果也不会很好。残差神经网络就是为了在加深网络的情况下又解决梯度消失的问题。残差结构可以不通过卷积，直接从前面一个特征层映射到后面的特征层（跳跃连接），有助于训练，也有助于特征的提取，容易优化。

多尺度预测

在每个grid cell预先设定一组不同大小和宽高比的边框，来覆盖整个图像的不同位置和多种尺度。每种尺度预测3个box, anchor的设计方式仍然使用聚类，得到9个聚类中心，将其按照大小均分给3个尺度，利用这三种尺度的特征层进行边框的预测。



特征图金字塔网络FPN（Feature Pyramid Networks）是2017年提出的一种网络，FPN主要解决的是物体检测中的多尺度问题，通过简单的网络连接改变，在基本不增加原有模型计算量的情况下，大幅度提升了小物体检测的性能。

目的是只使用一个尺度的输入创建一个所有层次都具有很强语义特征的特征金字塔。主要分了自下向上的一条路径和自上到下的一条路径。**自下向上就是深度卷积网络的前向提取特征的过程，自上而下则是对最后卷积层的特征图进行上采样的过程，横向的连接则是融合深层的卷积层特征和浅层卷积特征的过程。** 这也就是为什么对小物体也有很好的检测效果，它融合了深层卷积层的高级别特征和浅层卷积层的低级别特征。



流程：

1. 输入预处理

- **输入图像尺寸**：YOLOv3 默认使用 416×416 或 608×608 的正方形输入（也支持多尺度训练）
- **预处理操作**：
 - Resize 到固定大小 (416×416)
 - Normalize 到 $0,1,10,1$ 范围（除以 255）
 - 填充 padding 以保持图像长宽比（如果需要）

2. 特征提取 Backbone : Darknet-53

- **Darknet-53 网络结构**：
 - 包含 53 个卷积层（只用 3×3 和 1×1 卷积核）
 - 使用 **残差连接 (ResNet-like)** 来避免梯度消失
 - 每层后跟着 BN + LeakyReLU 激活函数
- **输出 3 个尺度的特征图**：
 - **13×13**：检测大目标（来自深层）
 - **26×26**：中目标
 - **52×52**：小目标

3. 检测头 (Detection Head) 多尺度预测

YOLOv3 不再只用最后一层做检测，而是在 3 个尺度上进行目标检测：

◆ 每个尺度做如下预测：

对于每个 cell (网格)：

- 预测 3 个 **anchor boxes**
- 每个 anchor 输出：
 - **4** 个边框参数 (tx, ty, tw, th) → 用于还原 x, y, w, h
 - **1** 个 objectness score（是否有物体）
 - **C** 个类别概率 (C=80 for COCO, 使用 **sigmoid** 而非 softmax)

所以每个 cell 输出： $(4 + 1 + C) \times 3 = (85) \times 3$ 数值（以 COCO 为例）

◆ 多尺度融合：

YOLOv3 采用类似 FPN 的特征金字塔结构：

- 将高层特征上采样（例如 $13 \times 13 \rightarrow 26 \times 26$ ）

- 与浅层特征图拼接 (concat) 融合
- 再进行卷积预测

这样可以：

- 保留高层语义信息（有助于分类）
- 融合低层细节信息（有助于定位）

4. 输出后处理

对所有预测框进行以下操作：

1. 边框解码：

- 将网络输出的 `tx, ty, tw, th` 转换为实际坐标（结合 anchor box 和 grid cell 的位置）

2. 置信度计算：

- `confidence = objectness × class_probabilities`

3. 筛选低置信度框：

- 通常设一个阈值（如 0.5），去除低分框

4. 非极大值抑制（NMS）：

- 对每个类别的预测框，保留得分最高的，并移除与其 $\text{IoU} >$ 阈值（如 0.5）的其它框
- 用于去除重复框、冗余检测

YOLOv4

实用性和性能的大提升

YOLOv4 是在 YOLOv3 基础上的一次 **大幅改进版本**，它的核心理念是用工业界可用的、无需特别硬件支持的方法，实现更高精度和更强的实用性。

核心改进点：

模块	改进	说明
Backbone	CSPDarknet-53	CSPNet (Cross-Stage Partial) 结构引入到 Darknet-53 中，减少计算量同时保持精度。
激活函数	Mish	替代 LeakyReLU，提高非线性表达能力，训练更稳定。
特征融合	SPP + PANet	SPP：多尺度池化 (Spatial Pyramid Pooling) 提取多尺度上下文信息；PANet：增强特征传递和融合。
数据增强	Mosaic	将4张图片拼接成1张，模拟复杂场景，提升鲁棒性与小目标检测能力。
正负样本匹配	CIOU Loss	更稳定地优化目标框的位置和尺寸（相比 YOLOv3 的 BCE 和 GIoU）。
正样本分配策略	Self-adversarial Training (SAT)	模型故意扰乱目标区域进行训练，使模型更鲁棒。

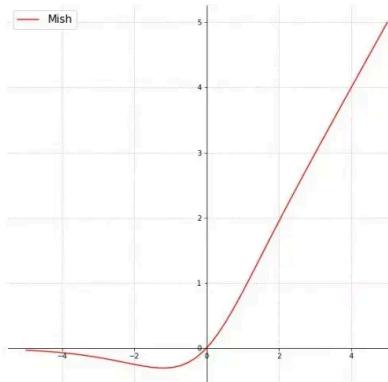


Figure 1. Mish Activation Function

YOLOv4 的精度和速度在 COCO 上超越 EfficientDet 等模型，但仍保持实用性强、部署简单的风格。

YOLOv5

模块	改进	说明
Pooling 模块	最大池化改为串行	YOLOv4 中部分池化是并行结构（如 SPP），YOLOv5 改为串行，减少模型复杂度。

YOLOv6

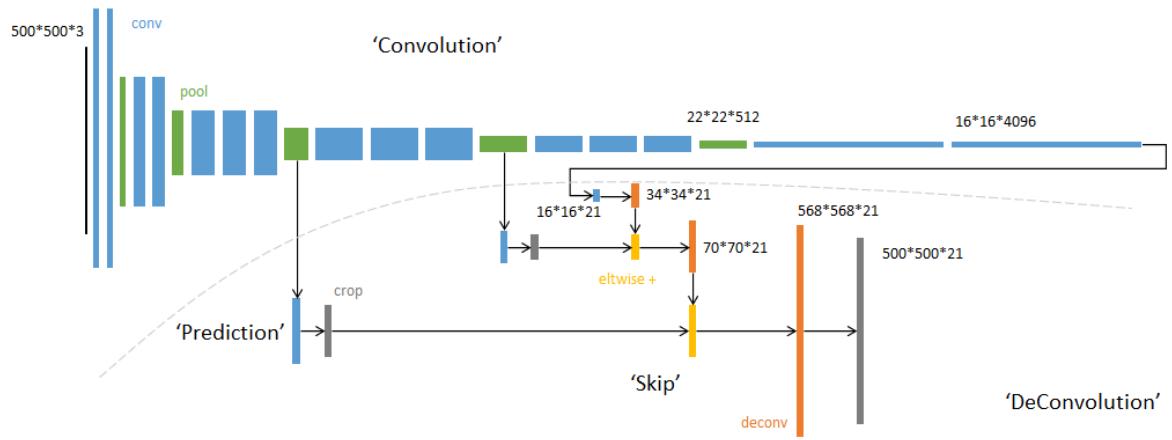
模块	改进	说明
Backbone	EfficientRep	类似 RepVGG 的思想，将训练时的多分支结构转化为推理时的单分支卷积结构，提升速度。

图像分割

FCN

- 全卷积网络 (Fully Convolutional Networks, FCN) 是使用深度神经网络进行图像分割的开山之作。
- FCN与传统的卷积神经网络不同，仅包含卷积层和池化层，不再包含全连接层和输出层。
- 因此，它也不再对整幅图像进行分类，而是实现了像素级的分类，进而输出图像分割的结果。

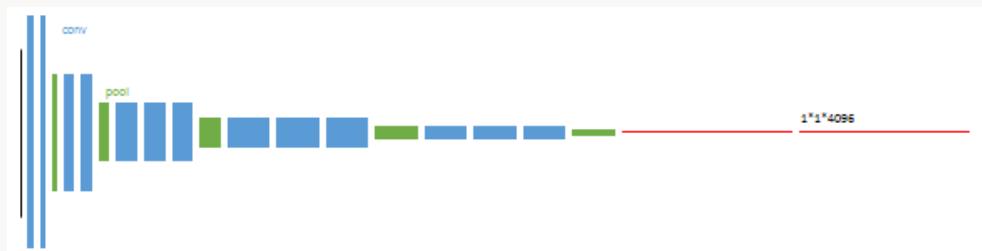
FCN网络结构主要分为两个部分：全卷积部分和反卷积部分。



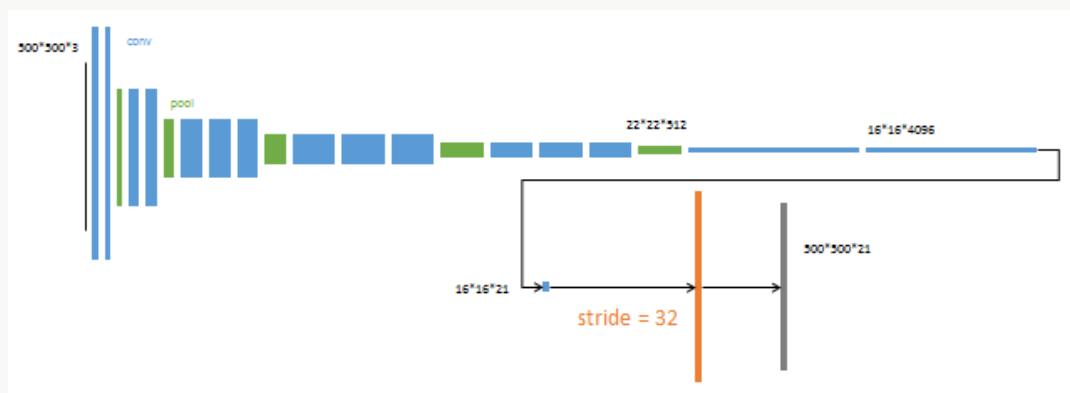


FCN训练

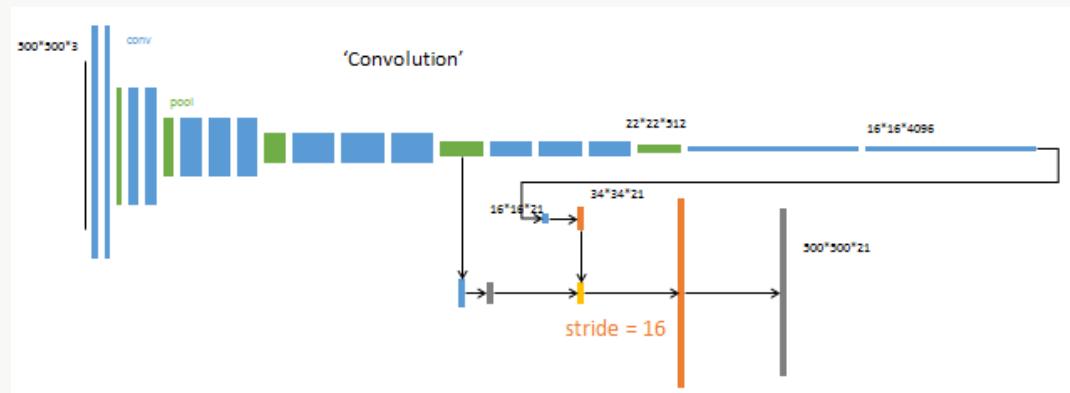
- 阶段1：以经典的分类网络为初始化，最后两级为全连接（红色），参数弃去不用。



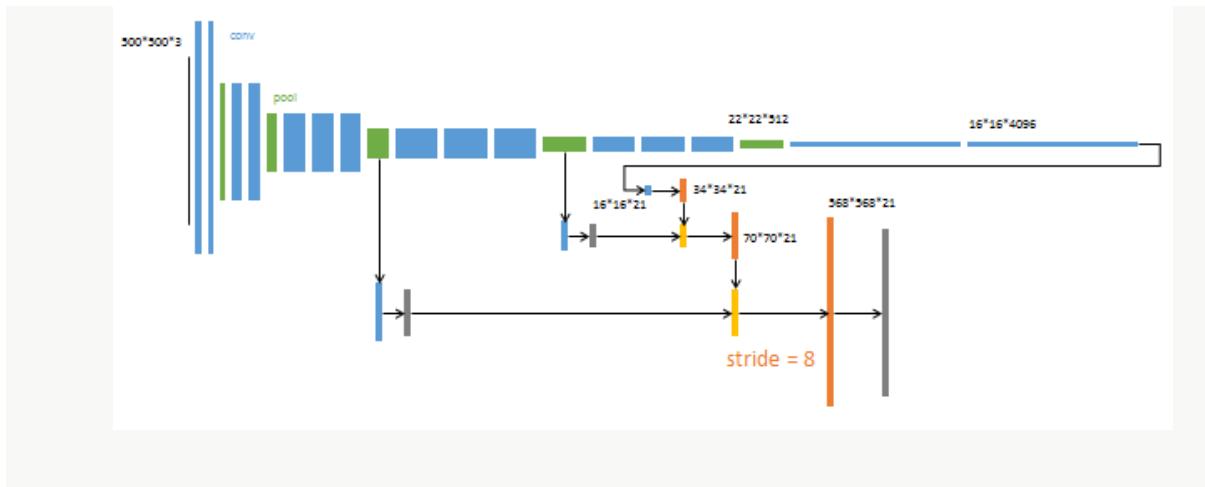
- 阶段2：FCN-32s 网络—从特征小图预测分割小图，之后直接升采样为大图。



- 阶段3：FCN-16s 网络—上采样分为两次完成。在第二次升采样前，把第4个pooling层的预测结果融合进来，使用跳级结构提升精确性。



- 阶段4：FCN-8s 网络—升采样分为三次完成。进一步融合了第3个pooling层的预测结果。



SegNet, UNet • 姿态估计：DeepPose，基于沙漏网络的姿态估计 • 人脸识别：DeepFace, DeepID/DeepID2, FaceNet

SegNet

SegNet是一款端到端 (end-to-end) 深度学习语义分割模型，其模型主要流程为：

图像输入 → 编码器网络 → 解码器网络 → softmax分类器 → 像素分割结果输出

编码器网络

编码器网络由13个卷积层组成，对应于VGG16网络中的前13个卷积层，并**丢弃了VGG16的全连接层**，这大大减少了SegNet编码器网络中的参数数量。将13个卷积层和5个池化层划分为5个编码器，单个编码器的组成部分如下：

- 滤波器组：由负责提取图像特征的多个卷积层组成
- BN层：为执行批量归一化操作的Batch Normalization
- 非线性层：为执行非线性操作的ReLU激活函数
- 最大池化层：使用 2×2 大小、步长为2的滑动窗口进行下采样操作

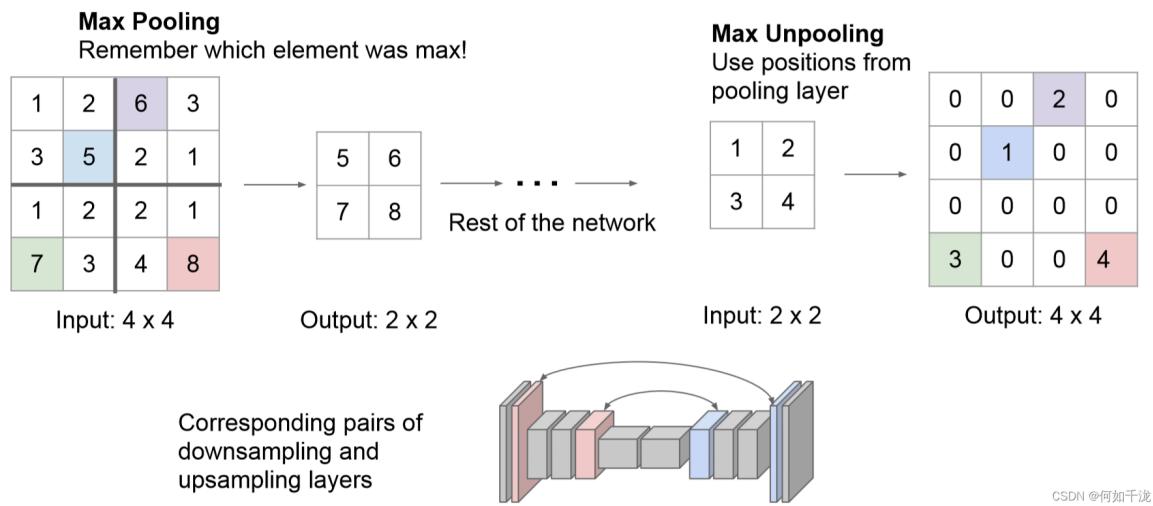
解码器网络

在SegNet网络结构中，编码器网络部分负责提取图像特征，并生成特征图，而解码器部分主要负责将特征图恢复到原始输入图像相同的空间分辨率，并生成分割结果。SegNet的编码器网络-解码器网络结构呈现对称结构，因此解码器网络也包含5个解码器组成，单个解码器组成部分如下：

- 上采样层：用于将图像恢复到更高空间分辨率；
- 滤波器组：由对经过上采样的特征图进行特征提取和整合的多个卷积层组成；
- BN层：为执行批量归一化操作的Batch Normalization；
- 非线性层：采用执行非线性操作的ReLU激活函数；
- softmax层：为k类的分类器，用于预测每个像素的类别，其在解码器的最后一层。

存储最大池化索引

即便编码器网络的多层最大池化可以通过**平移不变性**增强模型的**鲁棒性**，但随着特征图的空间分辨率不断降低，图像的边界细节也会逐渐损失，造成分割中不利于划定物体边界的问题，为此作者提出了独到的解决思路：**存储最大池化索引**。



SegNet在损失函数选取上较为简单，采用交叉熵的方式进行损失函数的计算，交叉熵损失函数**适用于多分类问题**，对概率分布较为敏感，有利于梯度下降。

$$L = \frac{1}{N} \sum_i L_i = -\frac{1}{N} \sum_i \sum_{c=1}^M y_{ic} \log(p_{ic})$$

- M : 类别的数量；
- y_{ic} : 符号函数（0 或 1），如果样本 i 的真实类别等于 c 取 1，否则取 0；
- p_{ic} : 样本 i 属于类别 c 的预测概率；

UNet

UNet网络结构，最主要的两个特点是：U型网络结构和Skip Connection跳层连接。



UNet是一个对称的网络结构，左侧为下采样，右侧为上采样。

按照功能可以将左侧的一系列下采样操作称为encoder，将右侧的一系列上采样操作称为decoder。

Skip Connection中间四条灰色的平行线，Skip Connection就是在上采样的过程中，融合下采样过过程中的feature map。

Skip Connection用到的融合的操作也很简单，就是将feature map的通道进行叠加，俗称Concat。

Skip Connection优点

优点	描述
空间细节保留	低层特征弥补上采样过程中的空间信息丢失
梯度传递顺畅	缓解梯度消失，提升训练效率
特征融合	高层语义 + 低层纹理，提升模型表达能力
分割准确	边缘更清晰、定位更精准，特别适用于医学图像等任务

姿态估计

姿态估计分类

- 单人姿态估计 (Single Person Pose Estimation, SPPE)
- 多人姿态估计 (Multiple Person Pose Estimation, MPPE)

Top-Down方法：首先利用目标检测器检测图像中出现的多个人，然后 使用SPPE模型估计每一个人的姿态。

Bottom-Up方法：首先检测出图像中所有的关键点位置，然后问题转换 为关键点的分配问题，将属于不同人的关键点进行关联和组合。

- 姿态估计方法最初集中于SPPE，但MPPE更符合实际情况但也更难，随着更多的MPPE数据集出现，针对MPPE的工作越来越多。

DeepPose

单人姿态估计

它将2D人体姿态估计问题由原本的**图像处理和模板匹配问题**转化为**CNN图像特征提取和关键点坐标回归问题**，并使用了一些回归准则来估计被遮挡/未出现的人体关节节点。

模型包括7层Alexnet和额外的回归全连接层，输出为 $2 \times$ 关节点坐标数目，表示在二维图像中的坐标。

作者在这个CNN网络的基础上使用了一个Trick:**级联回归器** (Cascaded Regressors)。其思路就是针对当时浅层CNN学习到的特征尺度固定、回归性能差的问题，将网络得到的粗分回归(x, y)坐标保存，增加一个阶段：在原图中以(x, y)为中心，剪切一个区域图像，将区域图像传入CNN网络学习更高分辨率的特征，进行较高精度的坐标值回归。

其主要局限性在于：1. Alexnet网络的学习能力有限；2. 直接回归2D坐标点太困难；3. 泛化能力差。

DeepPose 的基本结构如下：

objectivec

复制编辑

输入图像 → CNN (AlexNet) → 全连接层 → 输出关键点坐标 (2k维向量)

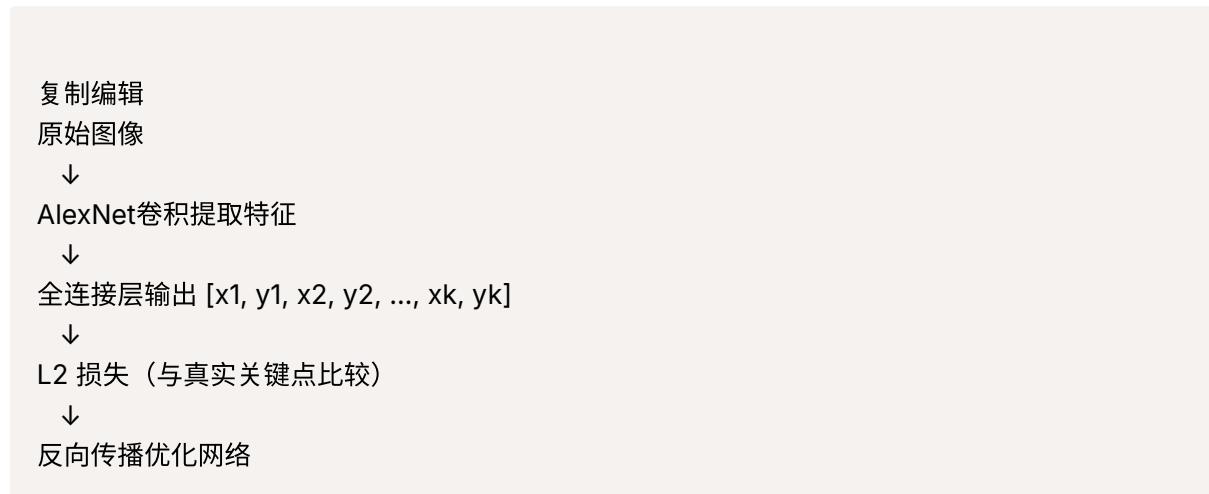
- 主干网络**：采用经典的 **AlexNet**，用于提取图像中的高级语义特征。
- 额外的全连接层**：在AlexNet原有结构的基础上添加，最终输出一个长度为 **2k** 的向量，其中 **k** 是 **关键点个数**，每个关键点对应两个数值 (x, y 坐标)。

DeepPose 并不是直接用一层网络完成预测，而是采用**粗到细 (coarse-to-fine)** 的方式逐步细化关键点预测：

1. 初步预测阶段：利用整个图像进行关键点粗定位；
2. 局部精细化阶段：以粗预测结果为中心，裁剪出图像块，进行更精细的预测；
3. 多阶段（cascade）结构，逐步逼近真实关键点位置。

这样可以缓解直接全局回归带来的误差较大问题。

总结流程图



Stacked Hourglass

其本质上是一种加入了残差的编码器-解码器-网络，其中编码器-解码器用于提取特征、恢复尺度，而残差则用来将不同尺度的特征通融合起来，并经由上采样过程逐步进入恢复到原始特征尺寸。残差在保留原始特征的同时也解决了深层网络的梯度消失问题，使得更深的网络结构中更加抽象的特征得以被提取和计算。

整个算法使用多个沙漏网络串行堆叠在一起，每个沙漏网络的输入输出尺寸可以被设置成相同，从而可以在训练中使用相同的标签对每个沙漏网络进行监督学习。多个沙漏网络之间也有残差结构，使得下层沙漏能够使用上层沙漏的特征进行训练，并且进行串行深度的拓展。

上层沙漏的结果（蓝色）通过 1×1 的卷积加入原有的特征，这个过程类似于DeepPose的级联回归器，从而进行由粗糙到细致的估计。

在网络输出结果时，为了避免直接回归坐标点带来的计算困难，堆叠沙漏网络输出关节节点位置的热力图，用于表示关节点出现在各个位置的概率，对每个输入为HWN沙漏网络，输出热图尺寸为 $H/2W/2K$ ，在其中选取最大的xy点，作为预测结果。

人脸识别

“非受限人脸识别”（**Unconstrained Face Recognition**）指的是在**非理想、非受控的环境下**对人脸进行识别的一种技术场景。它对比于“受限人脸识别”（如公安拍照、证件照对比）来说，挑战更大，实用性更强。