Foundation of Deep Learning(2)

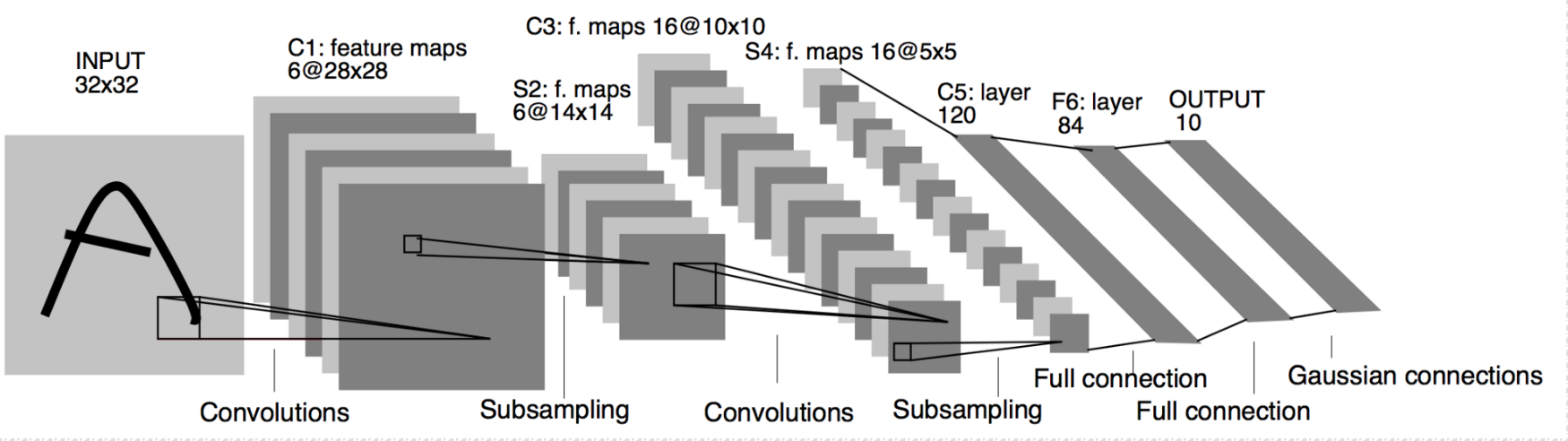
Bing Gong
（巩冰）
gongbing@shnu.edu.cn

Objectives:

- Convolutional Neural Network (卷积神经网络)
- Recurrent Neural Network (循环神经网络)
- Dataset for deep learning(深度学习数据)

Convolutional Neural Network

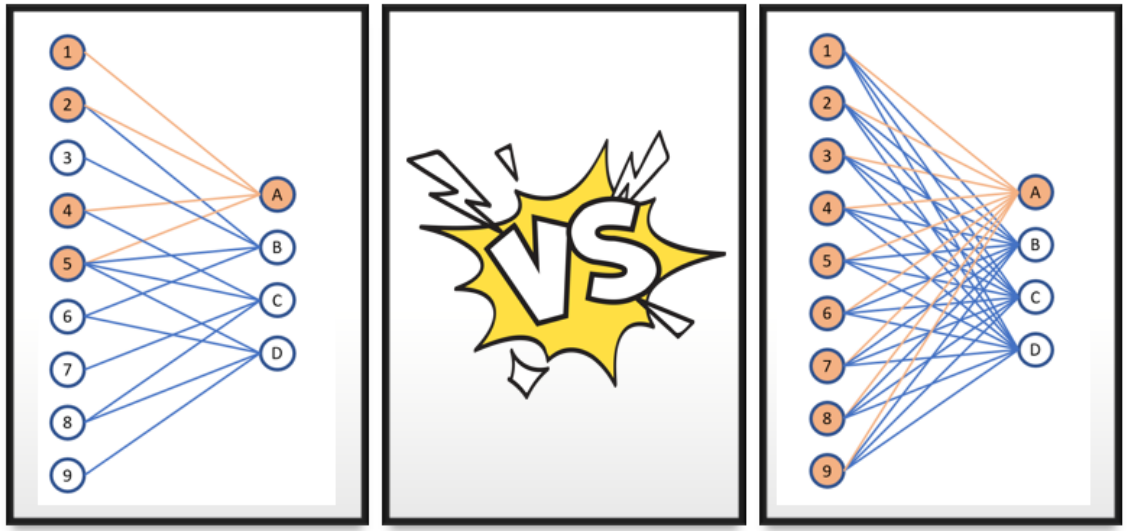
- **卷积神经网络（Convolutional Neural Network，CNN）**



卷积神经网络是一种常用于处理图像与视频数据的神经网络，其主要使用卷积层从图像中提取特征。

Convolutional Neural Network

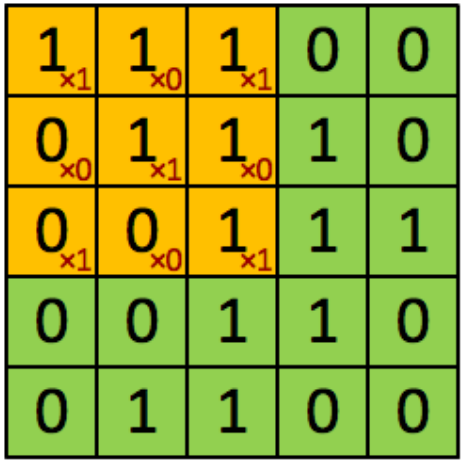
- **为什么不使用全连接层处理图像？**



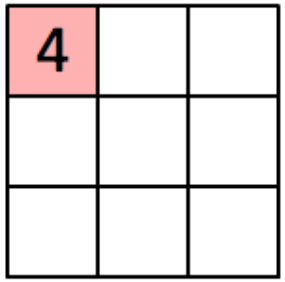
- 图像数据维度较高，使用全连接层处理会导致网络模型参数过多，难以训练和部署。
- 自然图像具备局部不变特性，即对图像进行平移、旋转等操作并不影响图像的语义信息，而全连接层难以捕获图像的这种特性。

Convolutional Neural Network

- **Convolution operation(卷积操作)**



Image



Convolved Feature

Convolutional Neural Network

- **Convolution operation(卷积操作)**

输入

0	0	0	0	0	0	0	0
0	3	3	4	4	7	0	0
0	9	7	6	5	8	2	0
0	6	5	5	6	9	2	0
0	7	1	3	2	7	8	0
0	0	3	7	1	8	3	0
0	4	0	4	3	2	2	0
0	0	0	0	0	0	0	0

$6 \times 6 \rightarrow 8 \times 8$

*

卷积核

1	0	-1
1	0	-1
1	0	-1

3×3

=

输出

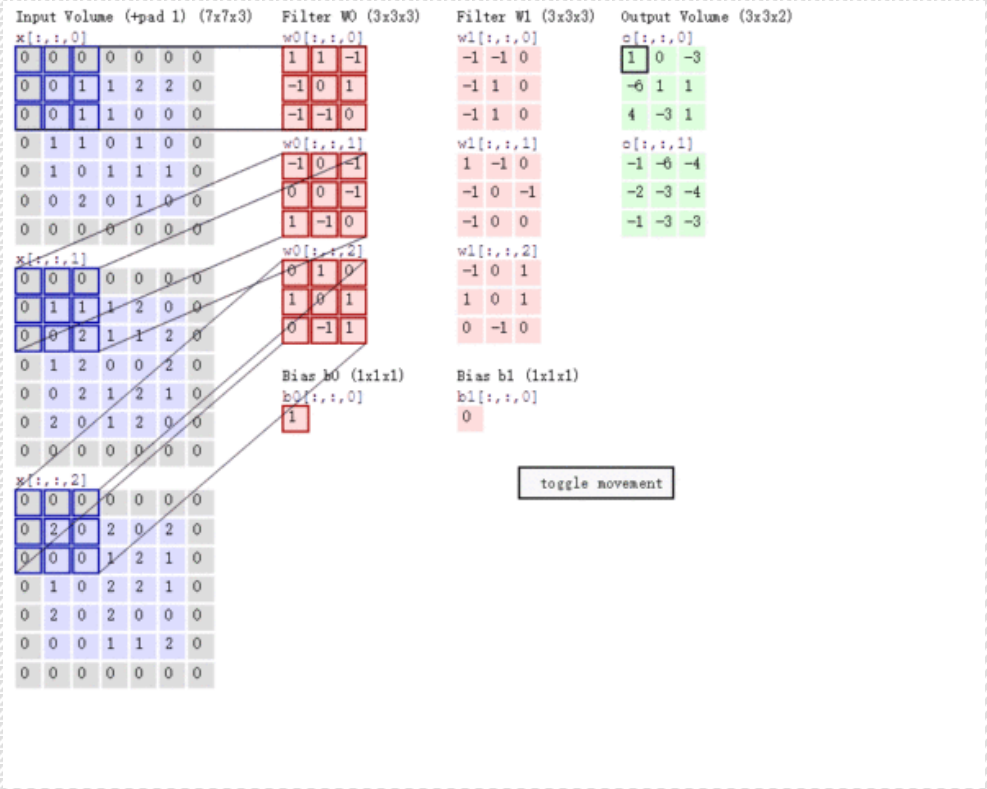
-10	-13	1			
-9	3	0			

6×6

特征图（Feature Map）:如上图右侧经过卷积核处理的图像即为特征图。
卷积核（Convolutional Kernel）：如上图，为一大小为3*3矩阵，与输入图像作滑动点积。
填充（Padding）：卷积会造成图像边缘像素损失，但有时我们希望输入与输出大小相同。为解决这一问题，我们在输入图像边缘填充一些像素。如上图我们在每一边额外填充了一行“0”。

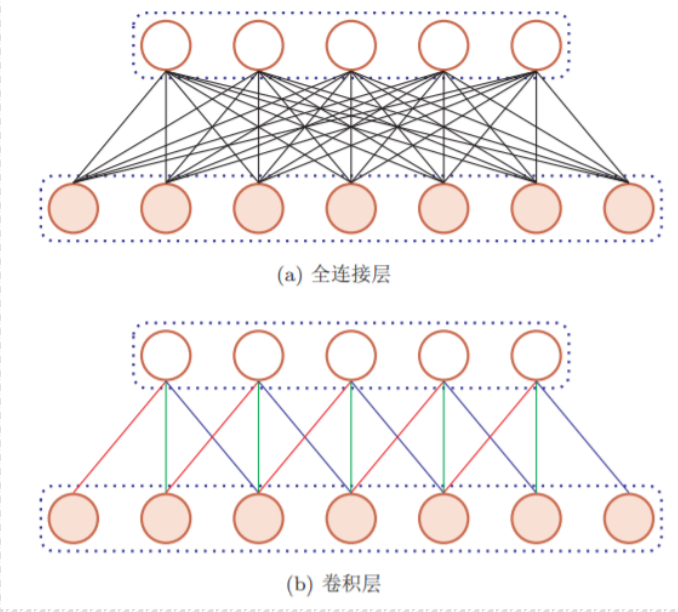
Convolutional Neural Network

- 卷积层



Convolutional Neural Network

- 卷积层

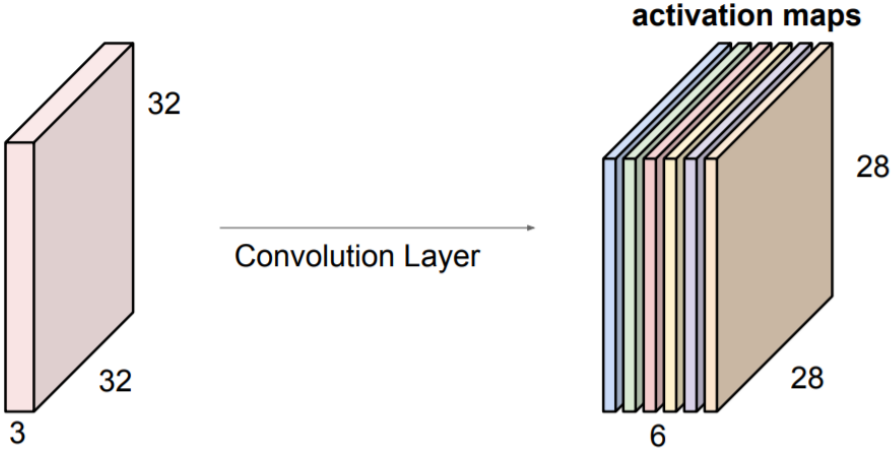


局部连接：与全连接层不同，卷积层的每一个神经元只和上一层局部窗口中的神经元连接，形成一种局部连接。
权重共享：卷积核的权重对于所有的神经元都是相同的。

Convolutional Neural Network

- 卷积层

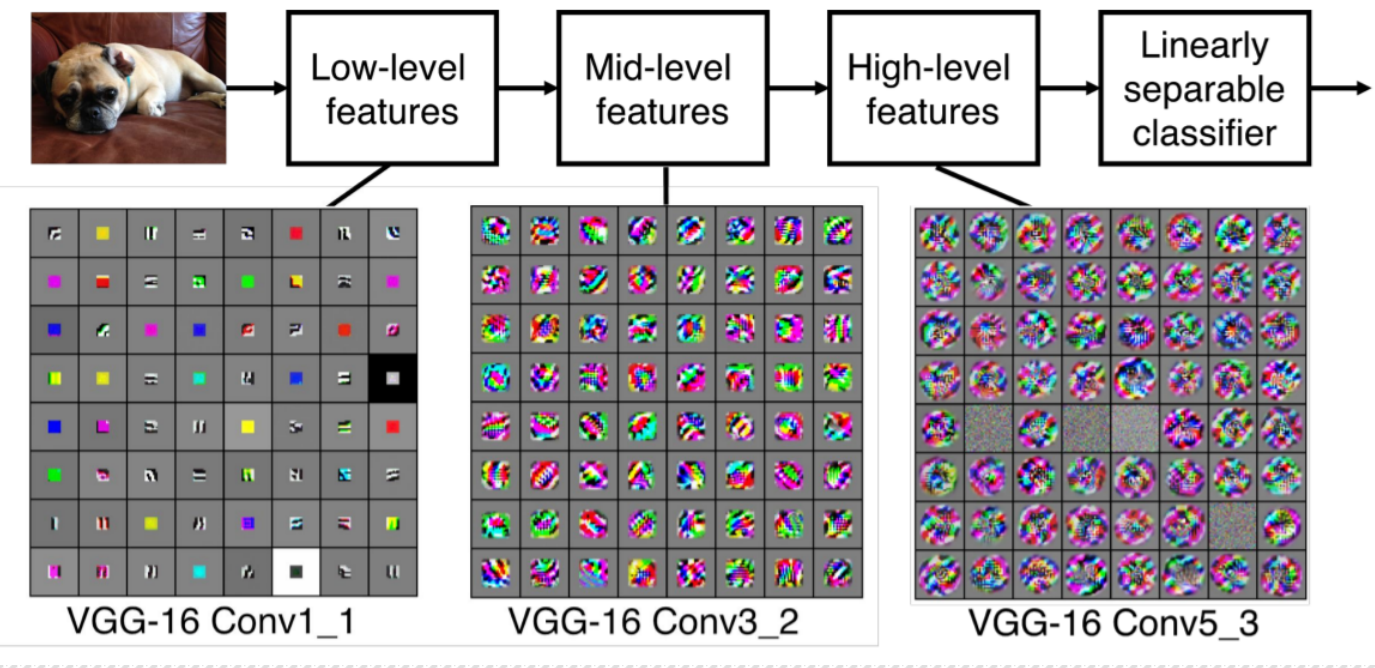
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

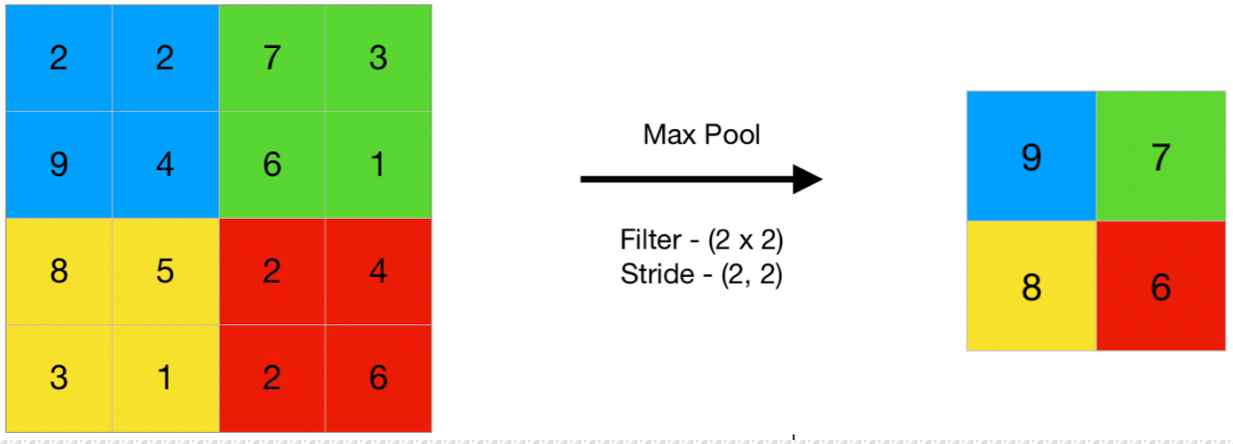
Convolutional Neural Network

- 卷积层



Convolutional Neural Network

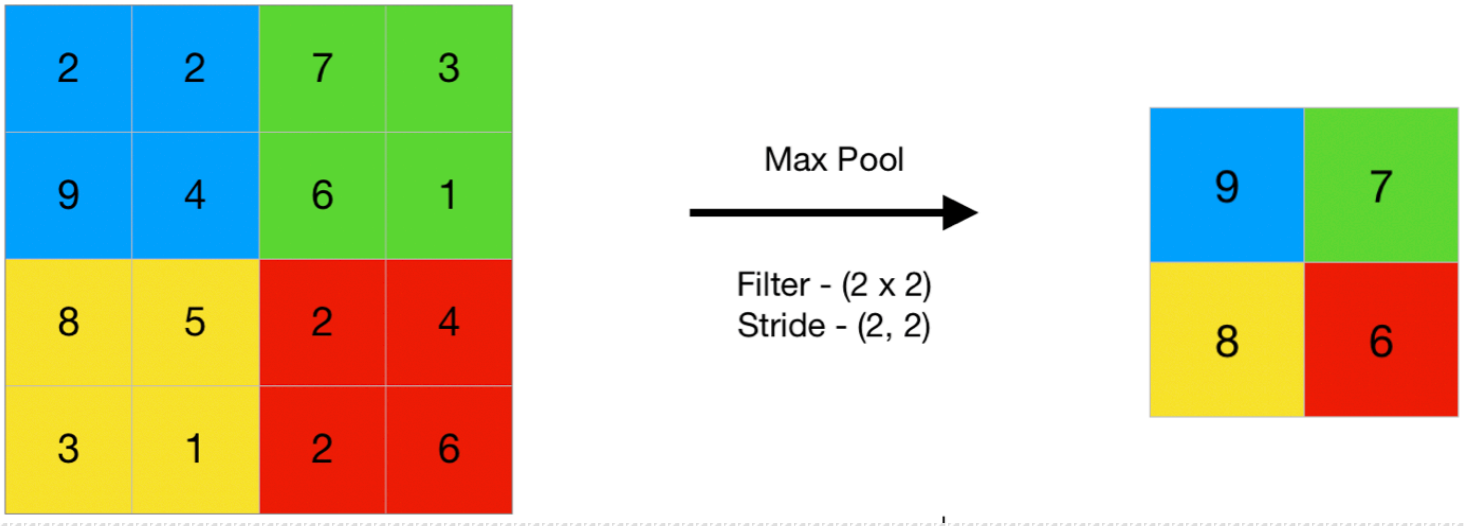
- 池化层（Pooling Layer）



池化层：池化层（or Subsampling Layer，子采样层）可以降低特征图的维数，从而减轻网络的计算负担。例如，在图像分类任务中，将池化层置于全连接层之前可以减少模型参数数量。

Convolutional Neural Network

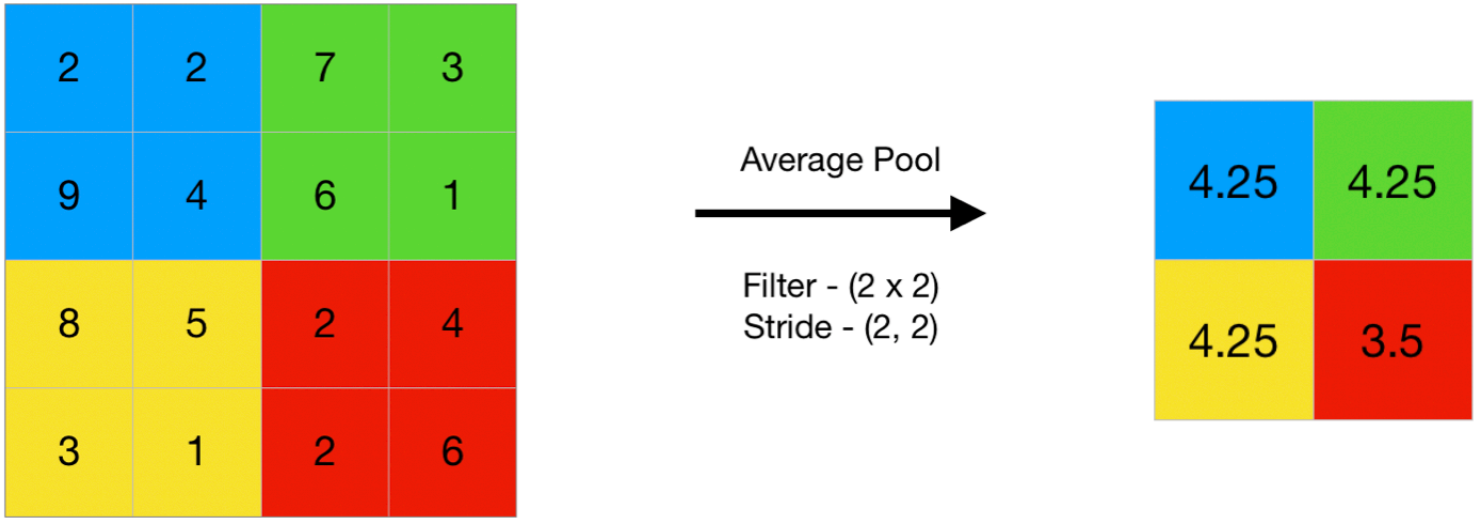
- 池化层（Pooling Layer）



最大池化（Max Pooling）：最大池化选择一个区域内的最大值作为这个区域的表示。

Convolutional Neural Network

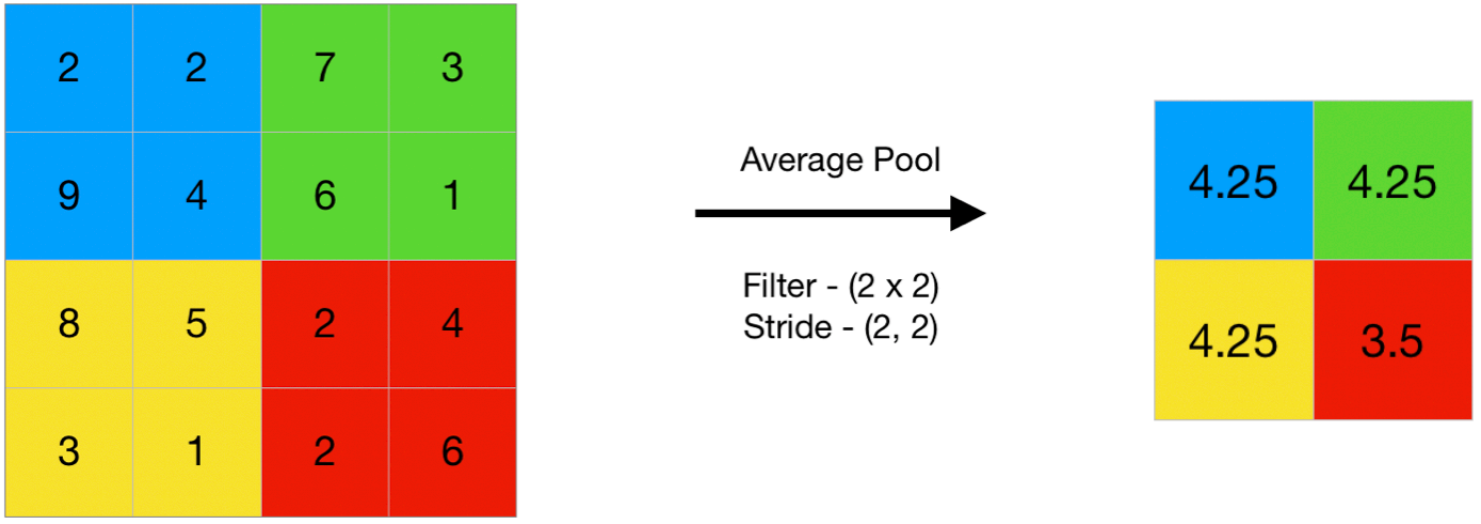
- 池化层（Pooling Layer）



平均池化（Max Pooling）：平均池化选择一个区域内的平均值作为这个区域的表示。平均池化和最大池化可以看作时一种特殊的卷积，它们可以概括本区域内的信息、扩大感受野，但同时也会造成一定信息损失。

Convolutional Neural Network

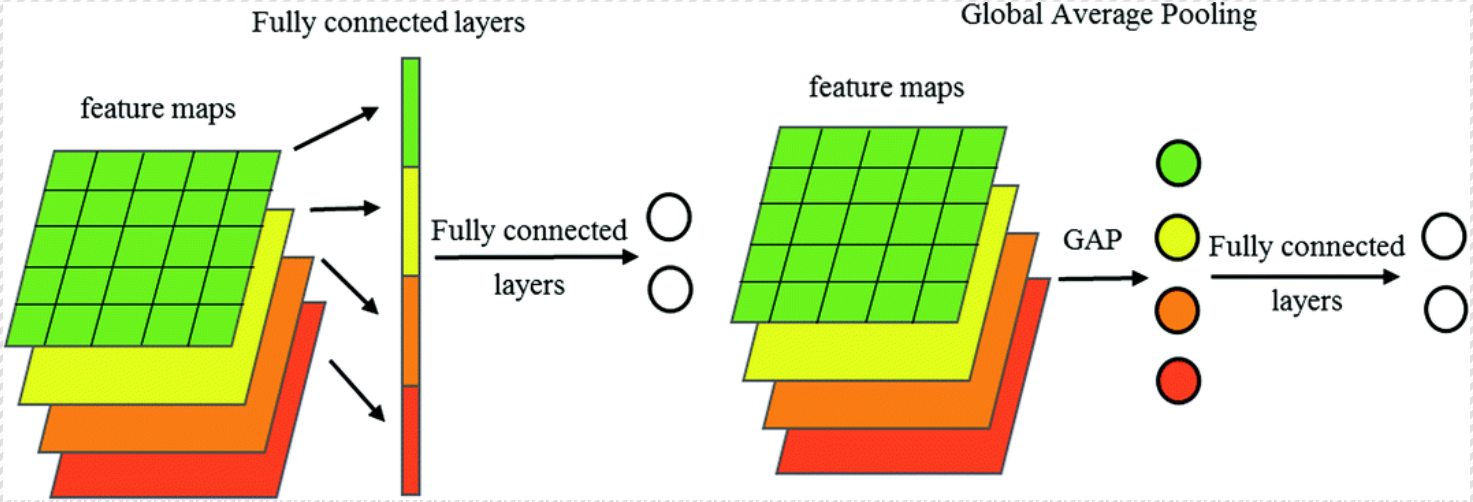
- 池化层（Pooling Layer）



平均池化（Max Pooling）：平均池化选择一个区域内的平均值作为这个区域的表示。平均池化和最大池化可以看作时一种特殊的卷积，它们可以概括本区域内的信息、扩大感受野，但同时也会造成一定信息损失。

Convolutional Neural Network

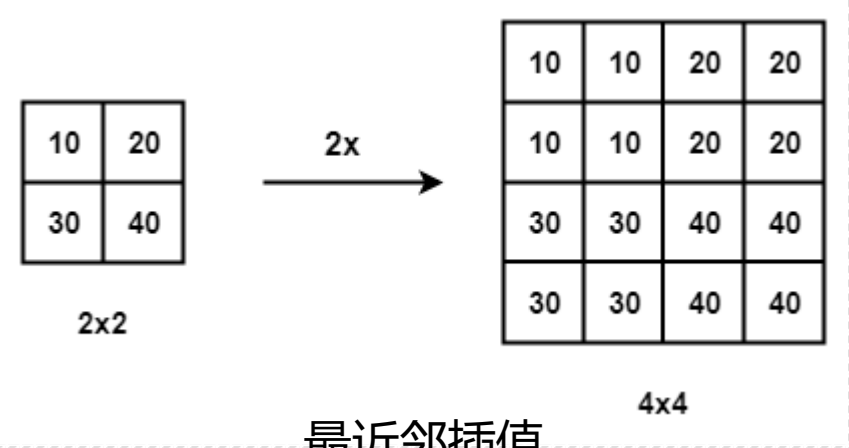
- 池化层（Pooling Layer）

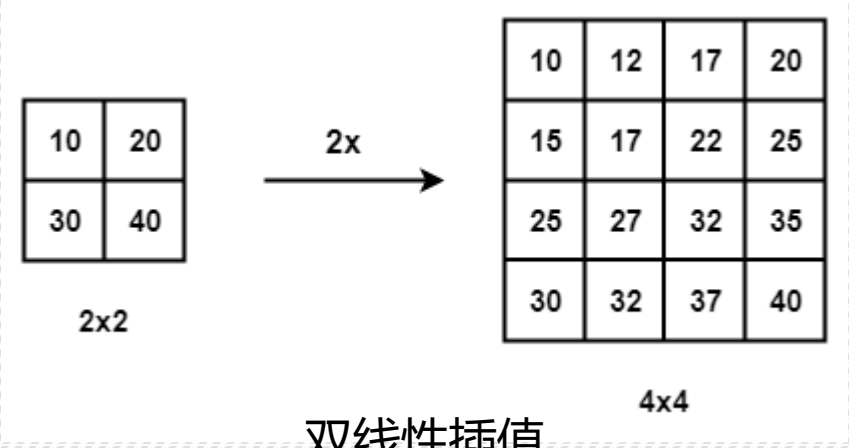


全局平均池化（Global Average Pooling）：全局平均池化取每个特征图的平均值作为输出。若输入特征图大小为$w * h * c$，则输出特征图大小为$1 * 1 * c$。

Convolutional Neural Network

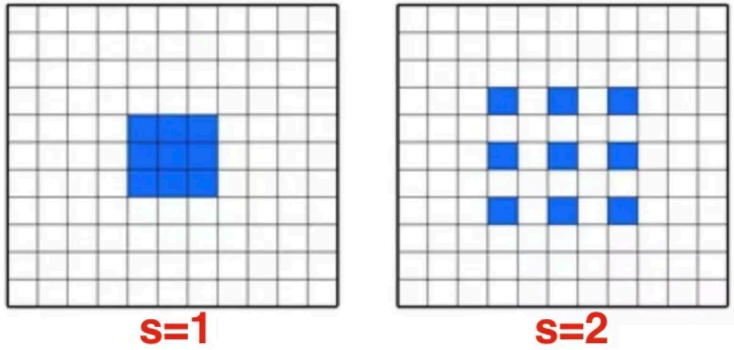
- **采样层（Upsampling Layer）**

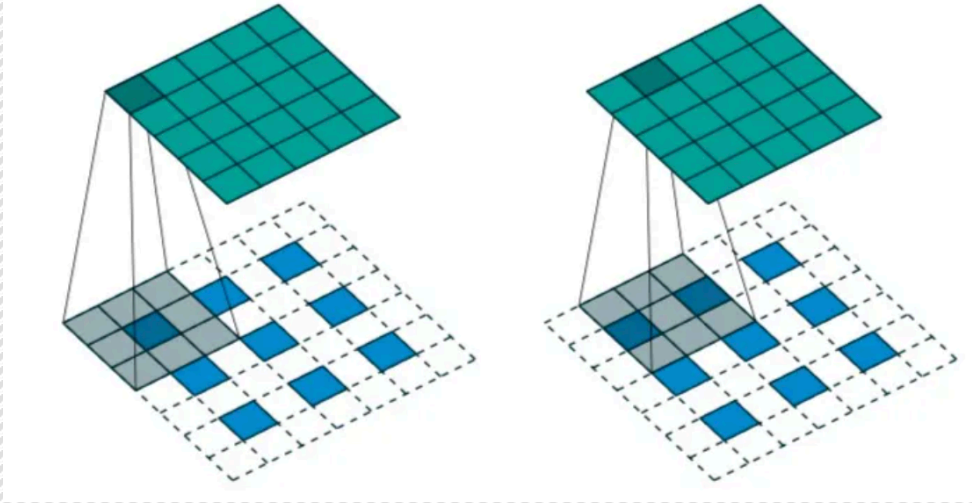




Convolutional Neural Network

- 转置卷积（Transposed Convolution）





Convolutional Neural Network

- **转置卷积（Transposed Convolution）**

转置卷积易造成棋盘状伪影

- **转置卷积（Transposed Convolution）**

为什么转置卷积会造成棋盘状伪影，什么情况下容易出现棋盘状伪影？

Convolutional Neural Network

- **转置卷积（Transposed Convolution）**

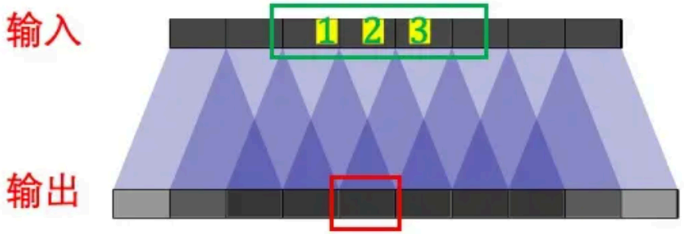


Fig. 2 stride=1, kernel size=3

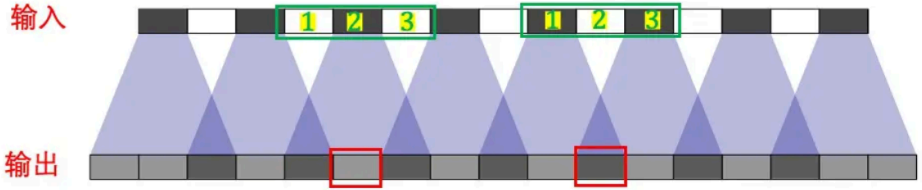
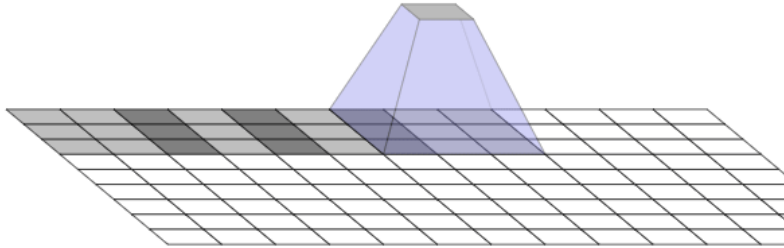


Fig. 3 stride=2, kernel size=3



不均匀重叠（uneven overlap）

Convolutional Neural Network

- 子像素卷积（Subpixel Convolution）

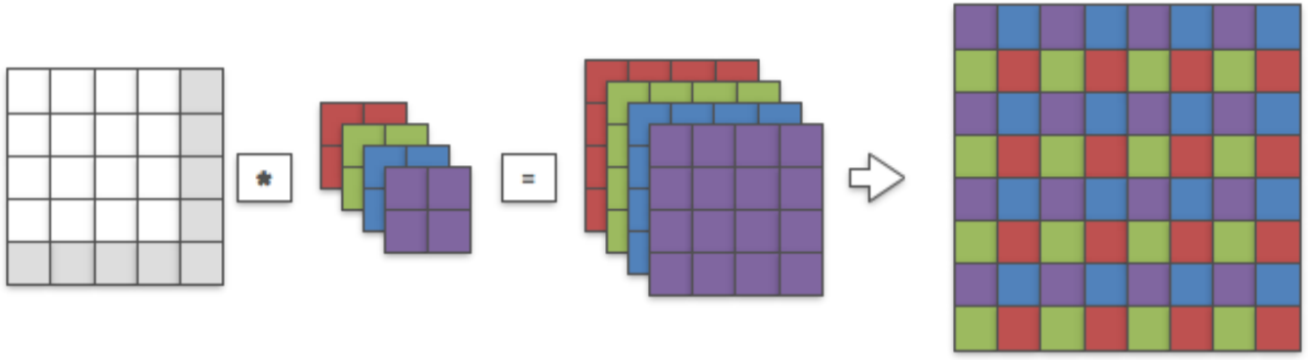
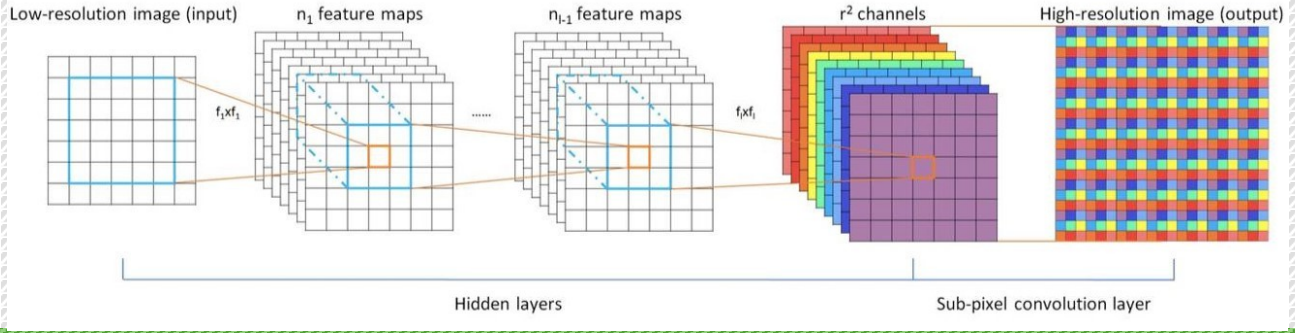
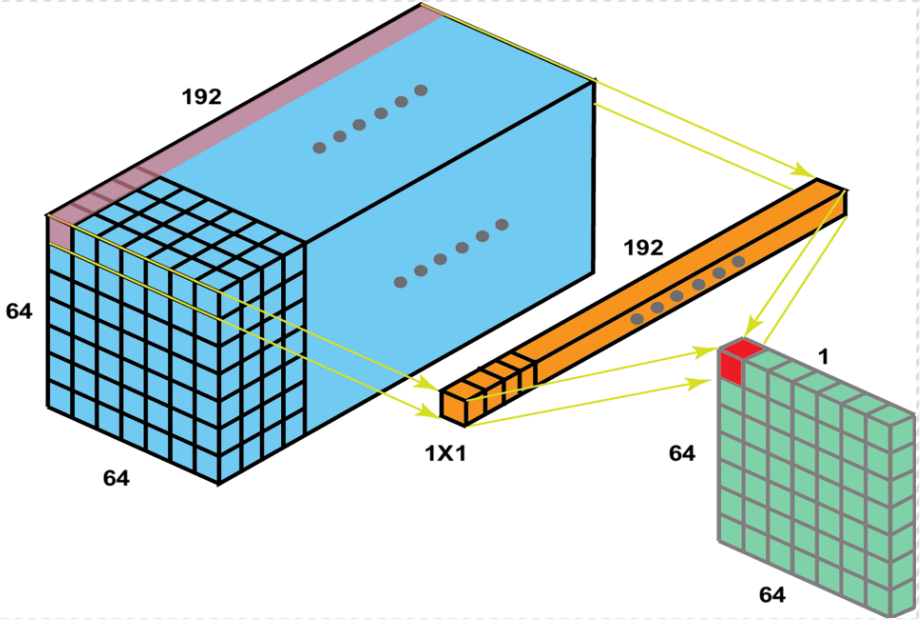


Figure 2: Sub-pixel convolution can be interpreted as convolution + shuffling.



Convolutional Neural Network

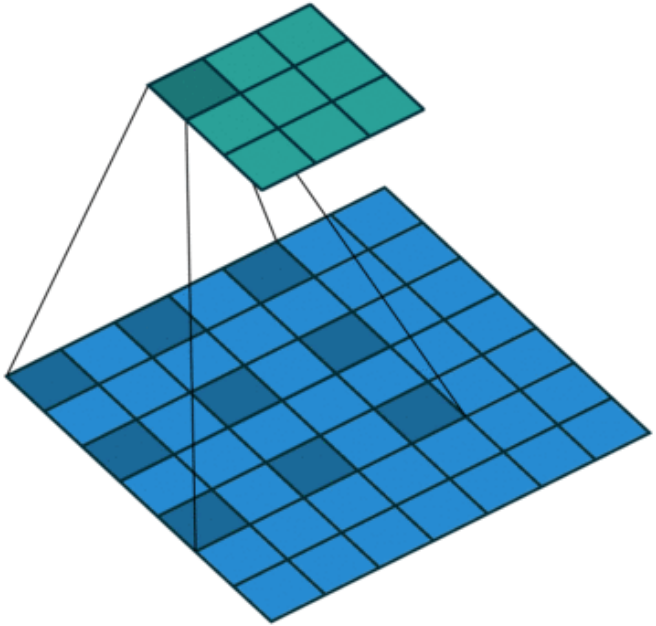
- 1×1卷积



1×1卷积：降低/提高特征图通道数

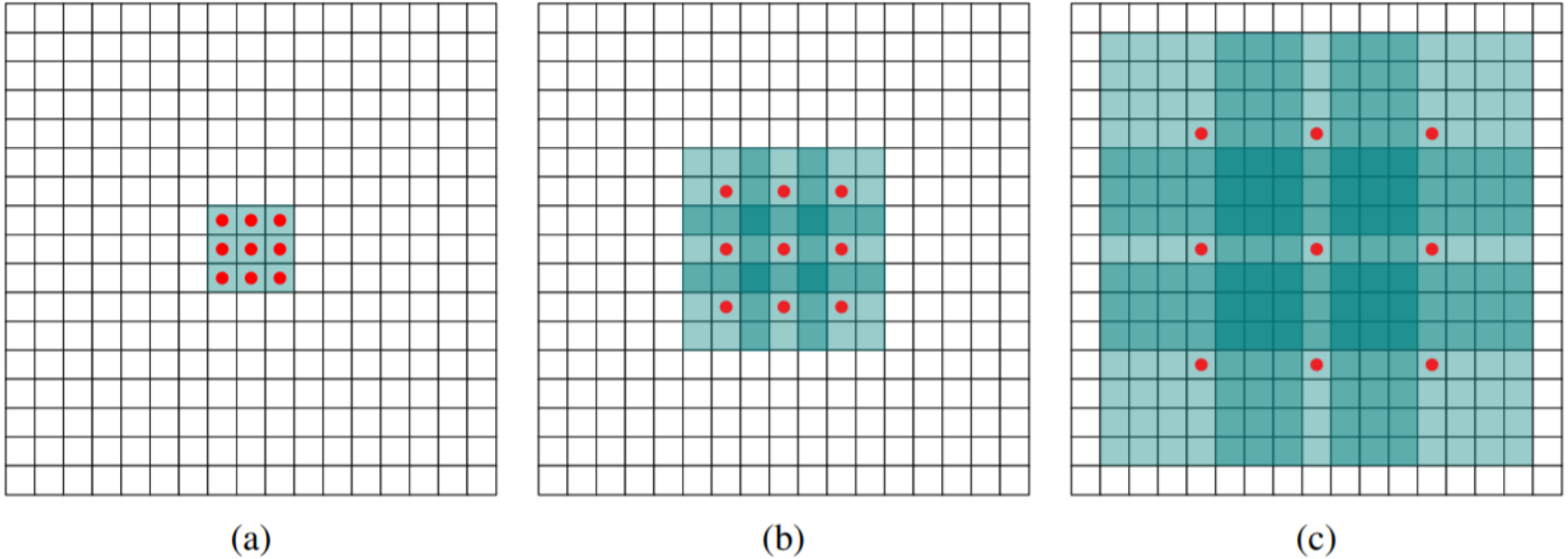
Convolutional Neural Network

- 空洞卷积（Dilated Convolution）



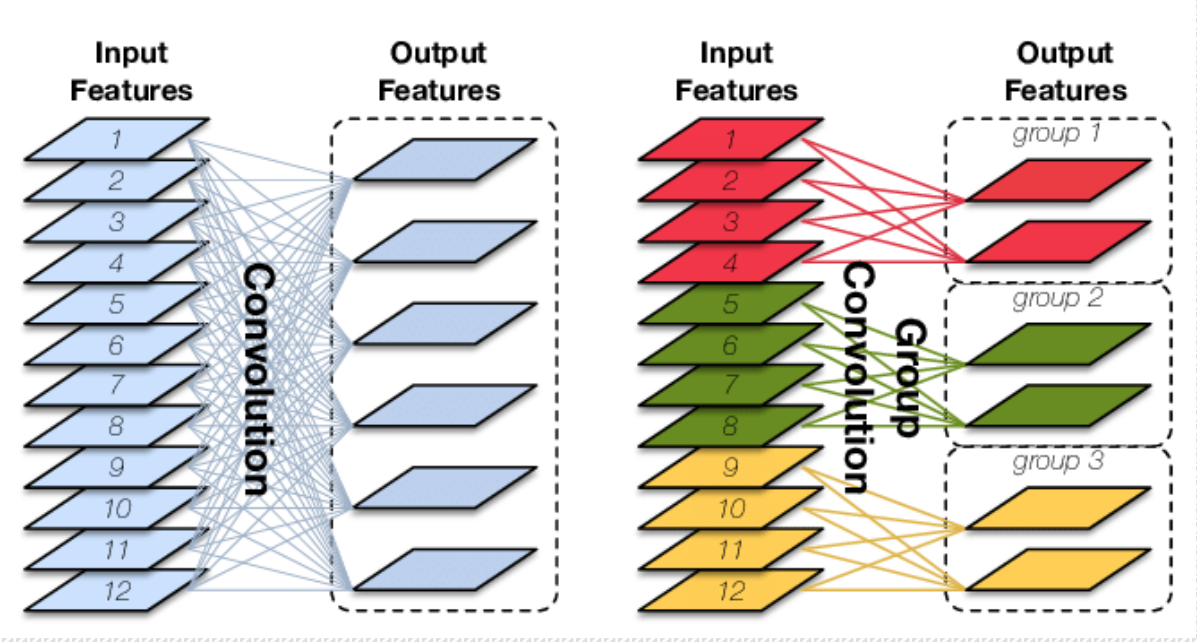
Convolutional Neural Network

- 空洞卷积（Dilated Convolution）



Convolutional Neural Network

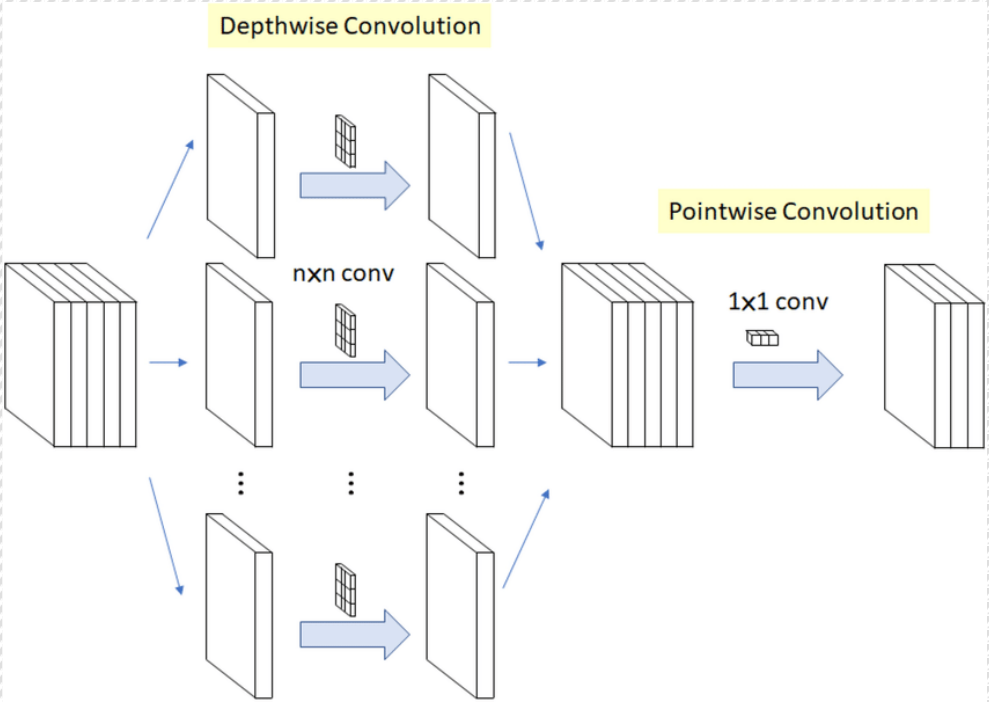
- **分组卷积（Grouped Convolution）**



分组卷积：减少网络参数、减轻计算负荷

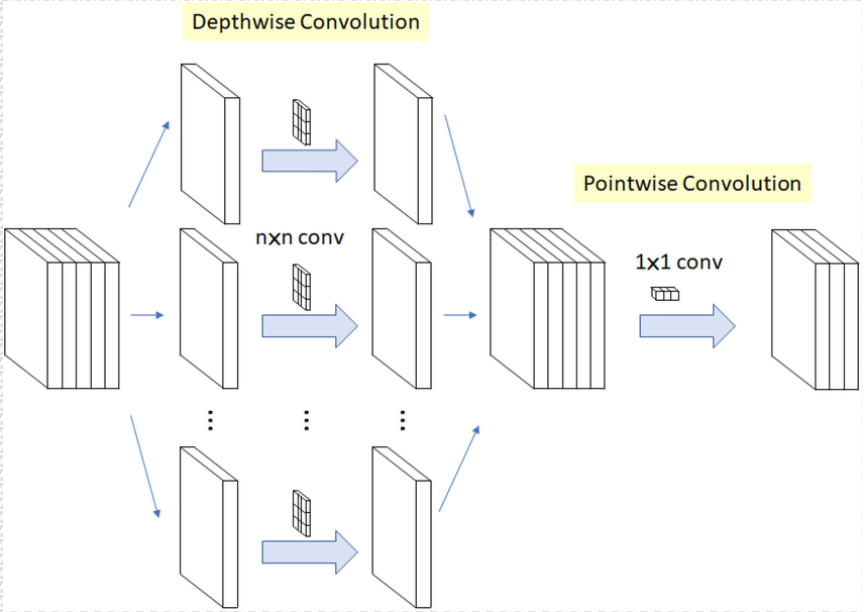
Convolutional Neural Network

- 深度可分离卷积（Depthwise Separate Convolution）



Convolutional Neural Network

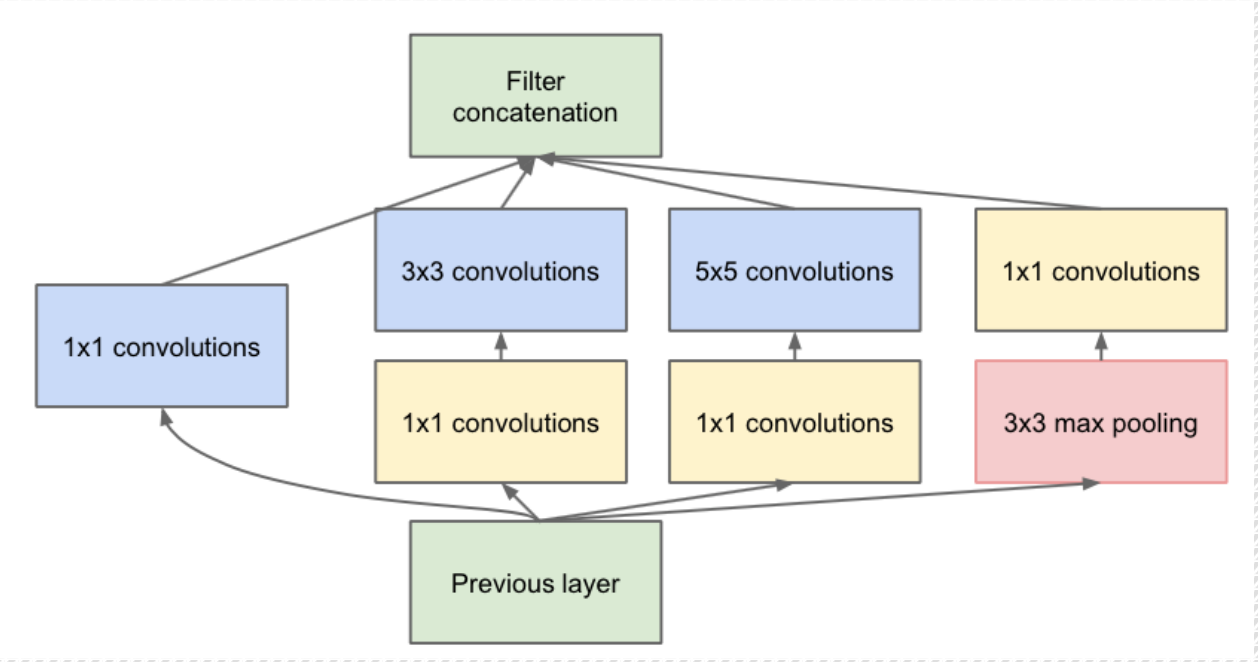
- **深度可分离卷积（Depthwise Separate Convolution）**



对于深度可分离卷积，若输入特征图大小为$W \times H \times C$，使用N个大小为$K \times K \times 1$的卷积核作单通道卷积，这样输出通道数为N，然后使用$1 \times 1 \times N$卷积核作1×1卷积，则参数数量为$K \times K \times C + C \times N$。需要注意的是，无论是分组卷积还是深度可分离卷积都会一定程度上降低性能。

Convolutional Neural Network

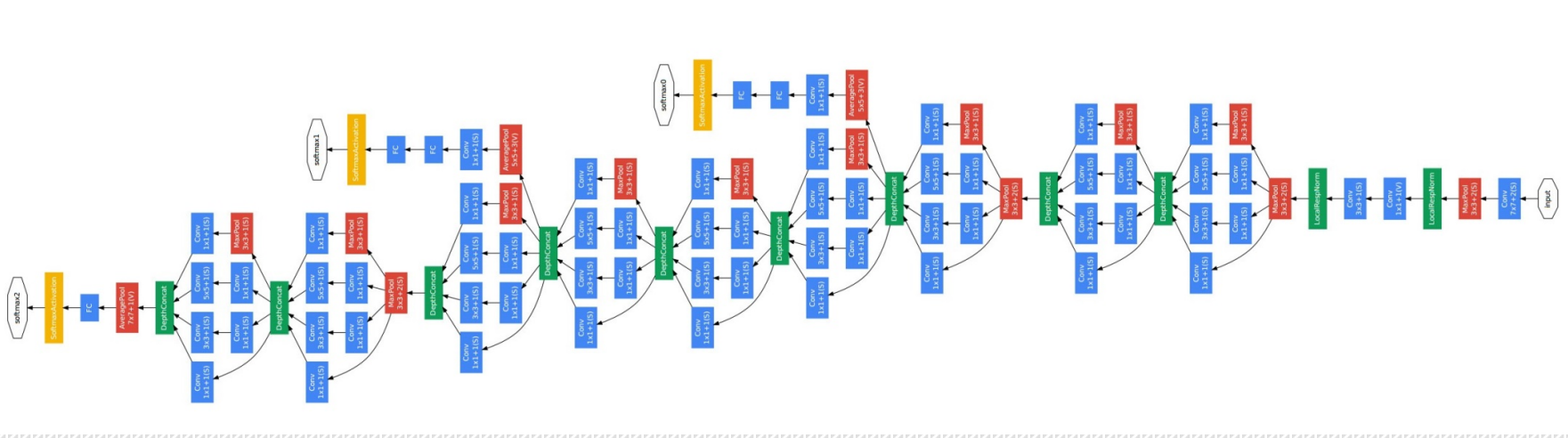
- **Inception系列**



Inception模块

Convolutional Neural Network

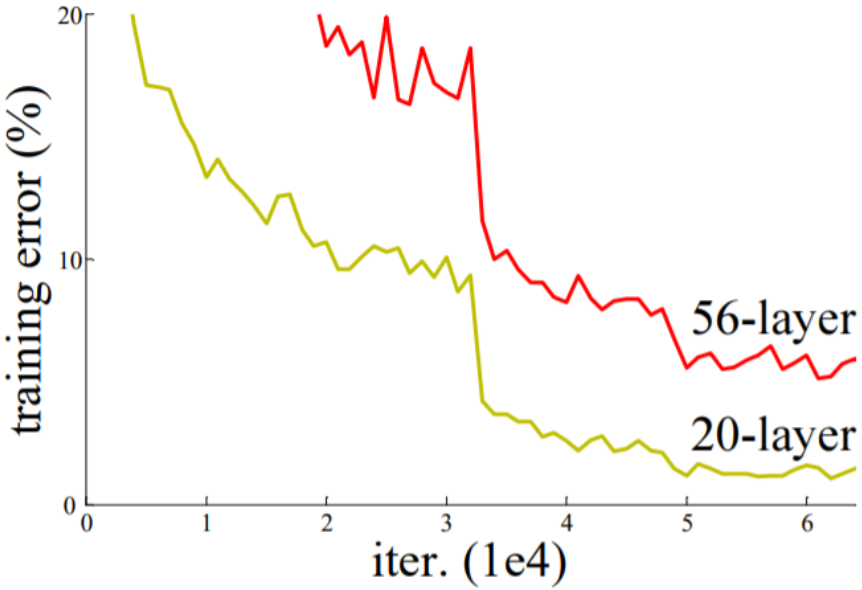
- **Inception系列**

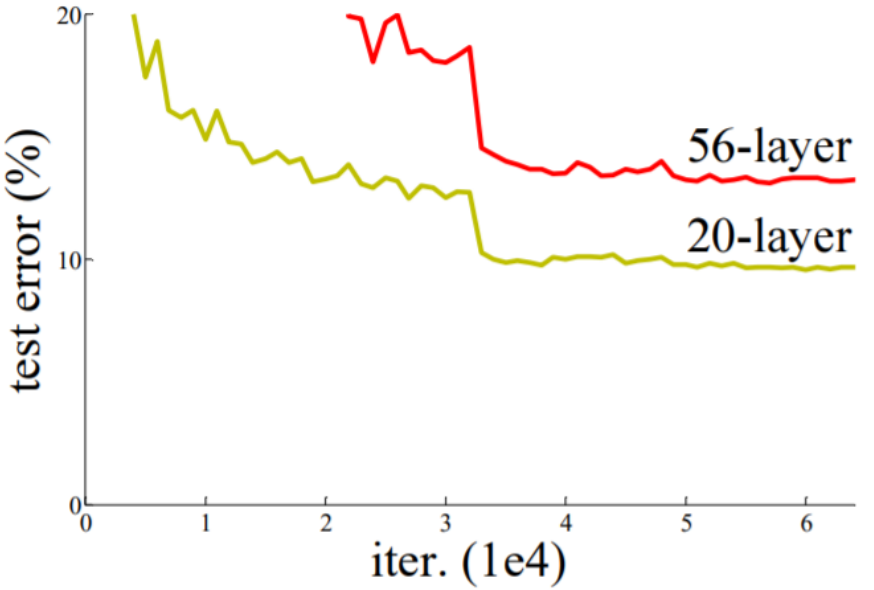


Inception V1

Convolutional Neural Network

- **ResNet系列**

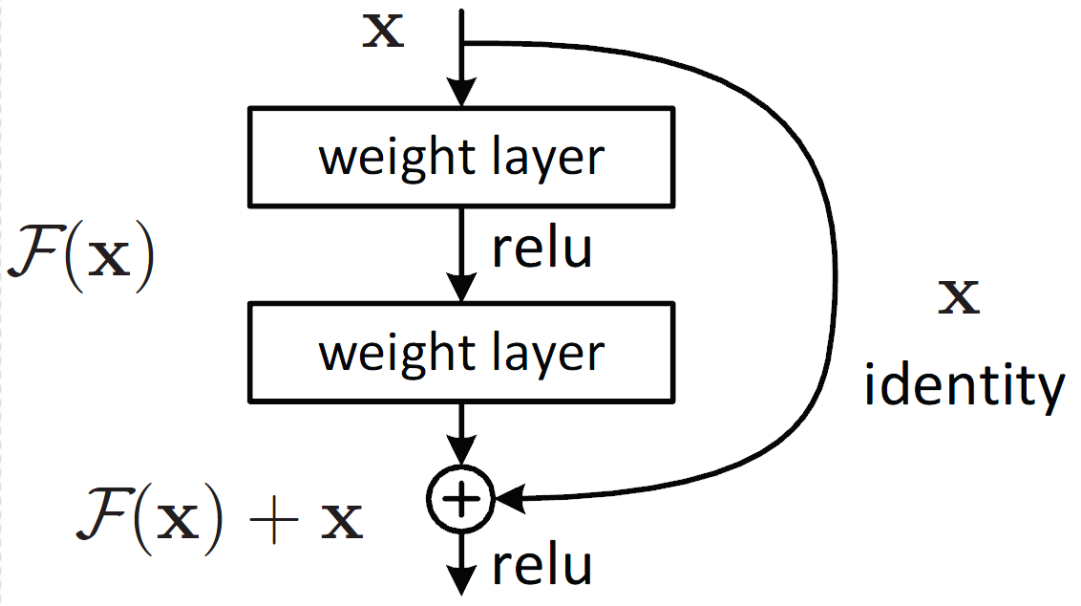




网络退化现象

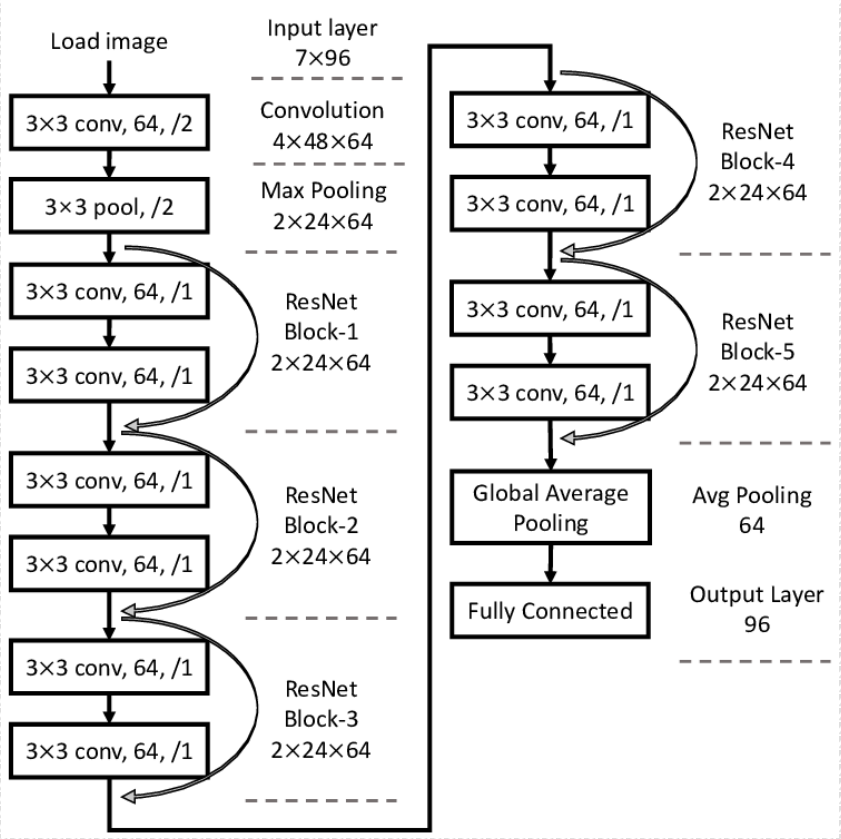
Convolutional Neural Network

- ResNet系列



Convolutional Neural Network

- **ResNet系列**

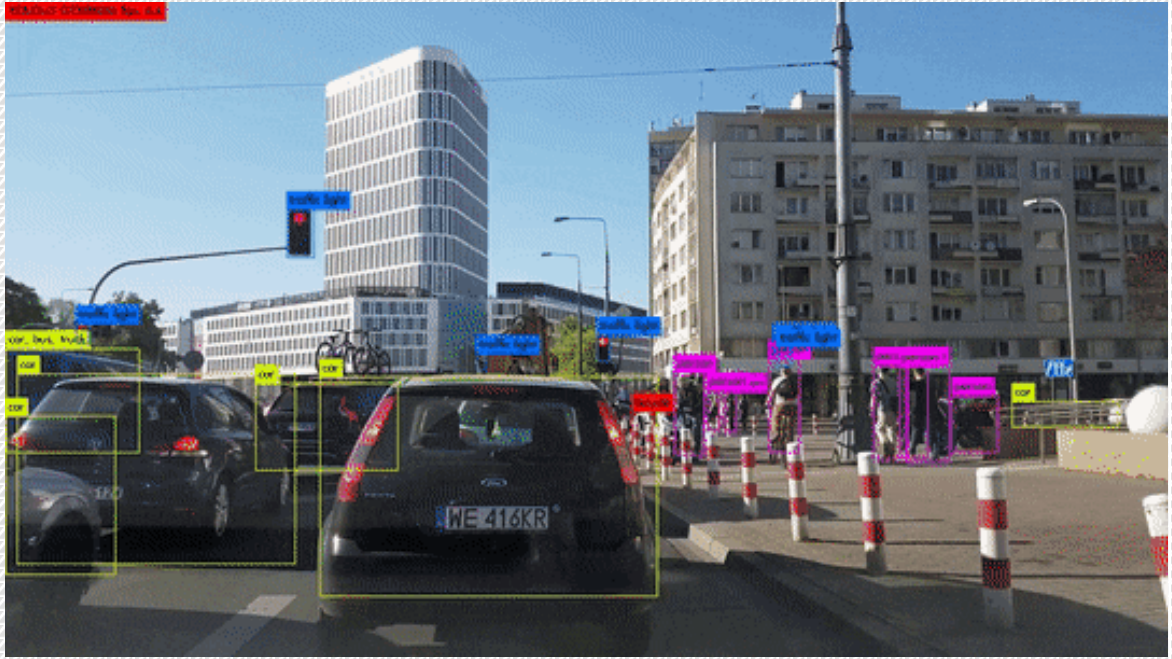


Convolutional Neural Network

- 应用：语义分割（Semantic Segment）

Convolutional Neural Network

- 应用：目标检测（Object Detection）



Convolutional Neural Network

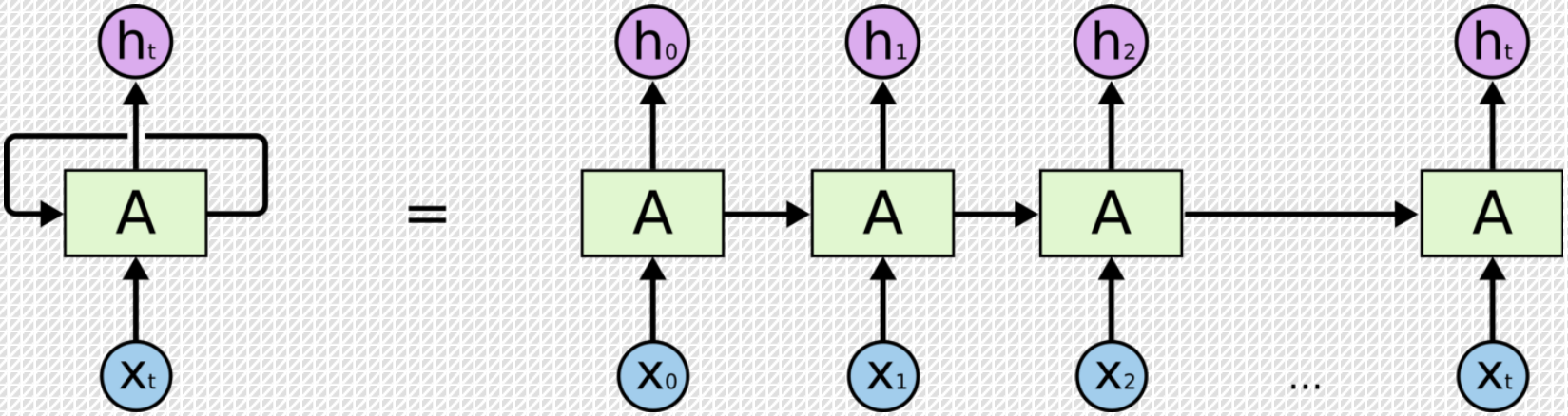
- 应用：图像超分辨率（Image Super Resolution）

original

?upscale=true

Recurrent Neural Network

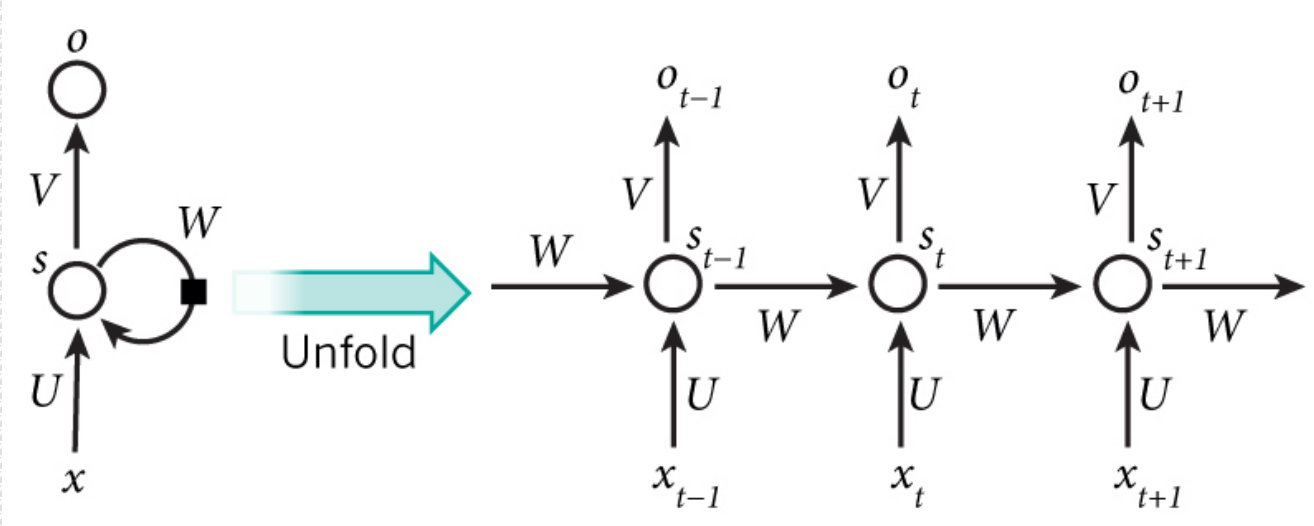
- **循环神经网络（Recurrent Neural Network, RNN）**



在一些情况下，我们需要处理视频、文本等长度不固定、有前后关系的序列数据，但一般的前馈网络要求固定的输入与输出，并不适合除了这一类数据,因此需要一种更强的模型——循环神经网络。

Recurrent Neural Network

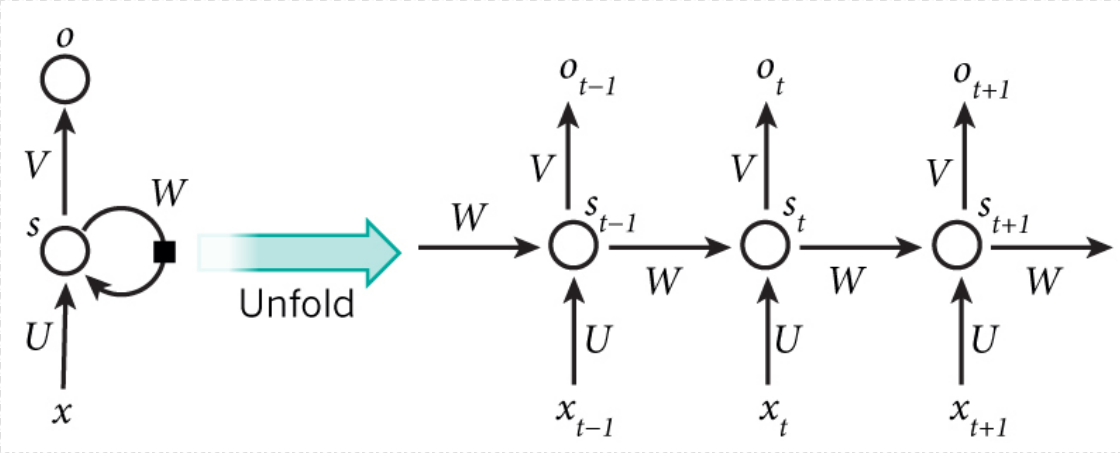
- **循环神经网络（Recurrent Neural Network, RNN）**



RNN的核心思想是利用序列数据的关联信息，一般的前馈网络会将输入的序列数据看作是不相关的独立个体，但RNN会“记忆”过往的信息，从而利用关联信息处理序列数据。

Recurrent Neural Network

- **循环神经网络（Recurrent Neural Network, RNN）**

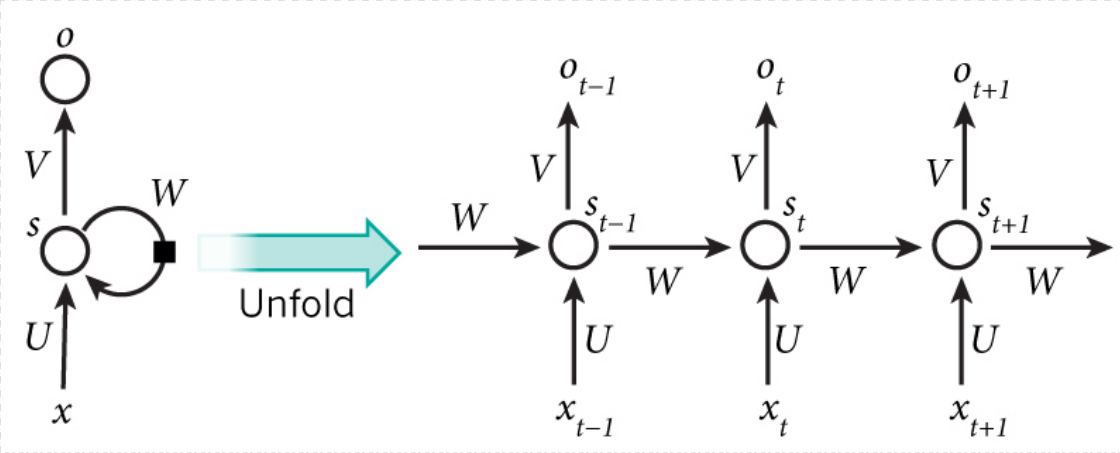


RNN展开

RNN基本包括输入层、隐藏层和输出层，其中U代表输入层到隐藏层的权重矩阵，V为隐藏层到输出层的权重矩阵，W为隐藏层到隐藏层的输入矩阵。

Recurrent Neural Network

- **循环神经网络（Recurrent Neural Network, RNN）**

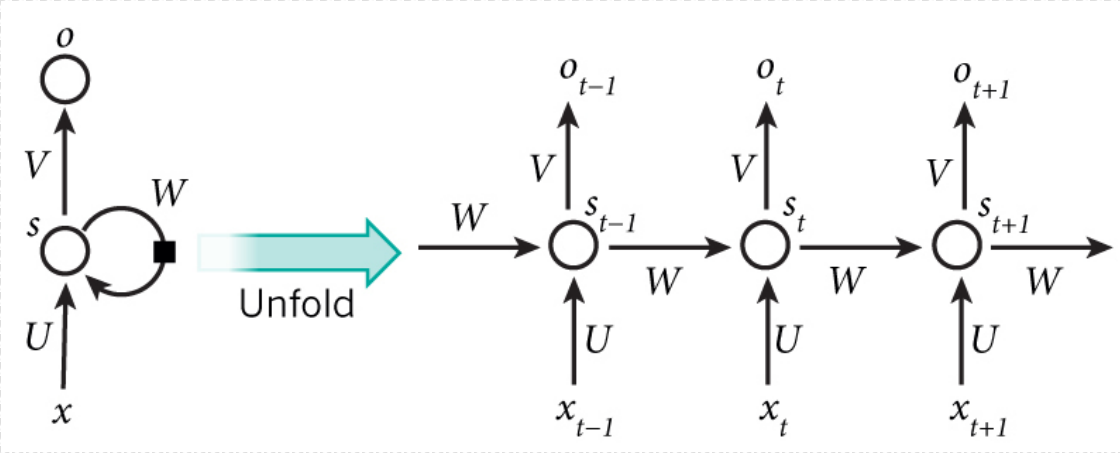


RNN展开

x_t为第t步的数据，o_t为第t步的输出，s_t为RNN的隐状态，可以理解为RNN的”记忆”。

Recurrent Neural Network

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



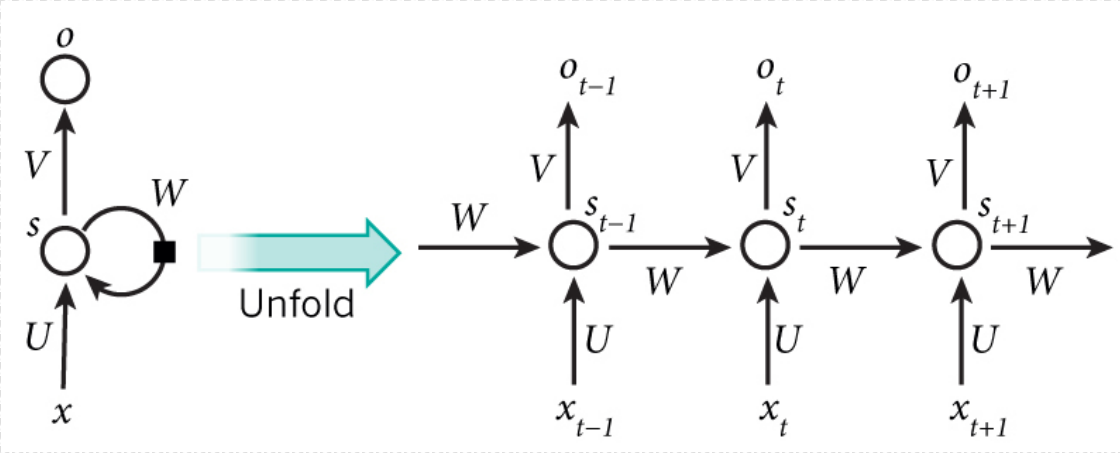
RNN展开

$$s_t = h(Ux_{t-1} + Ws_{t-1} + b)$$

$$o_t = Vs_t$$

Recurrent Neural Network

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



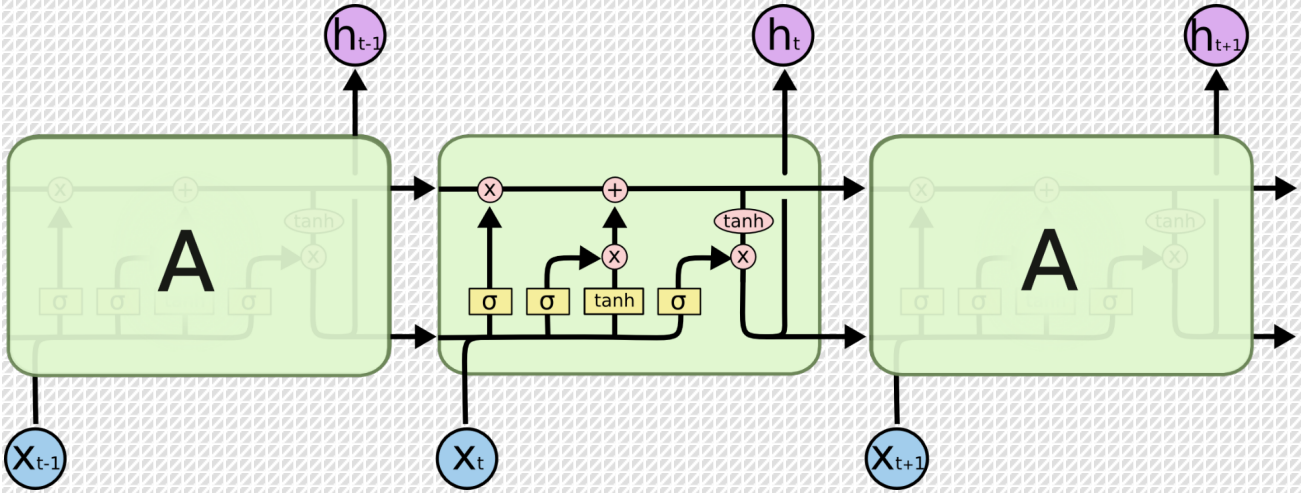
RNN展开

$$s_t = h(Ux_{t-1} + Ws_{t-1} + b)$$

$$o_t = Vs_t$$

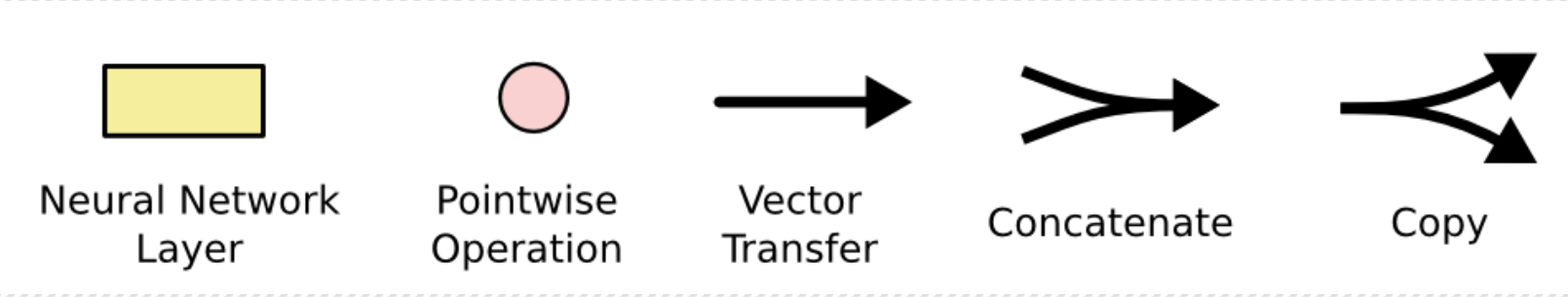
Recurrent Neural Network

- 长短记忆网络（Long Short Term Memory networks, LSTM）



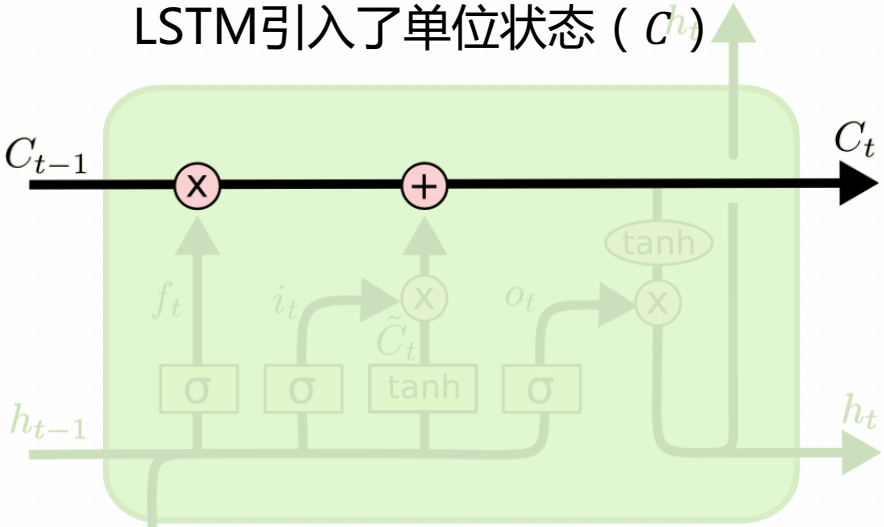
Recurrent Neural Network

- 长短记忆网络（Long Short Term Memory networks, LSTM）



- **长短记忆网络（Long Short Term Memory networks, LSTM）**

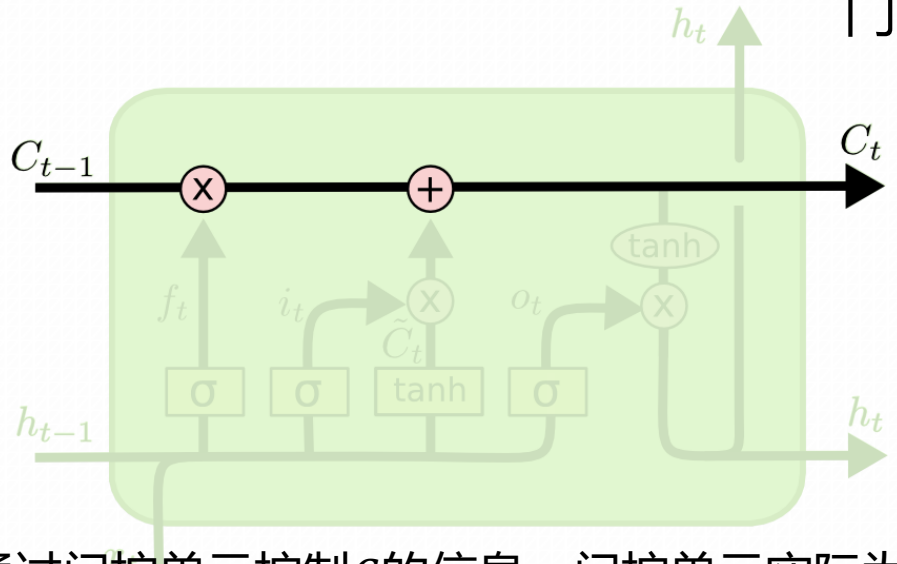
LSTM引入了单位状态（C）



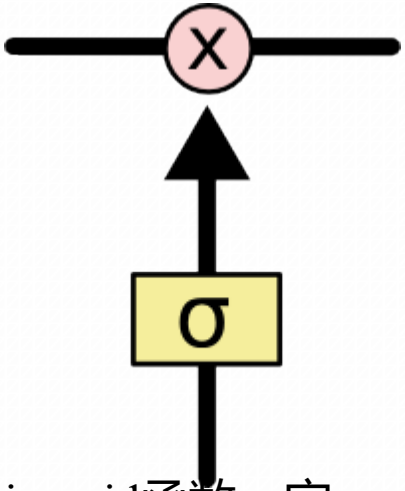
LSTM通过门控单元控制C的信息，门控单元实际为一Sigmoid函数，它可以将数值压缩至0，1之间，值越接近0，则表示不允许数据“流过”，越接近于1代表允许数据“流过”。可以看到LSTM中有三个这样的门控单元。

Recurrent Neural Network

- **长短记忆网络（Long Short Term Memory networks, LSTM）**



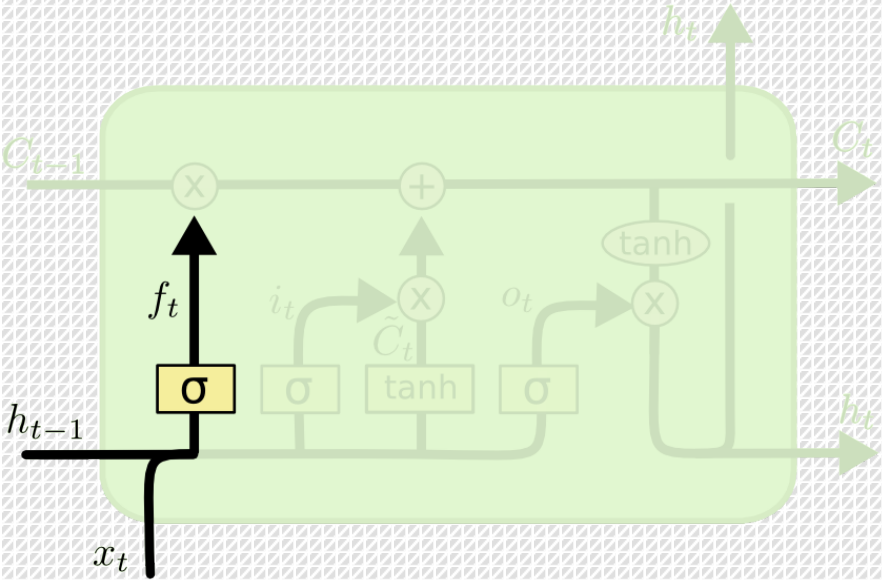
门控单元



LSTM通过门控单元控制C的信息，门控单元实际为一Sigmoid函数，它可以将数值压缩至0，1之间，值越接近0，则表示不允许数据“流过”，越接近于1代表允许数据“流过”。可以看到LSTM中有三个这样的门控单元。

Recurrent Neural Network

- **长短记忆网络（Long Short Term Memory networks, LSTM）**



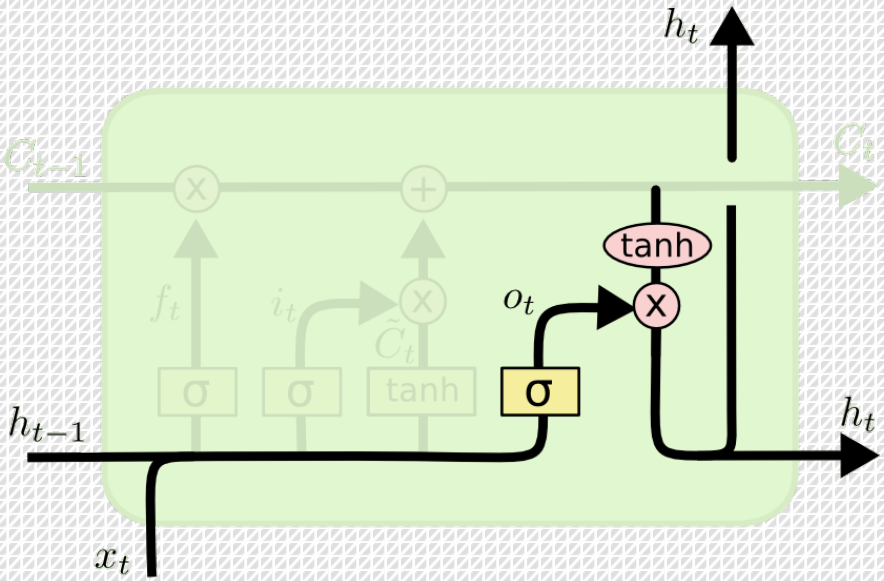
遗忘门控单元

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

LSTM运行的第一步便是决定要从C中“遗忘”掉多少信息。遗忘门对h_{t-1}和x_t进行拼接，然后通过线性变换和Sigmoid激活函数生成遗忘矩阵，最后对单位状态C作矩阵点乘。

Recurrent Neural Network

- 长短记忆网络（Long Short Term Memory networks, LSTM）



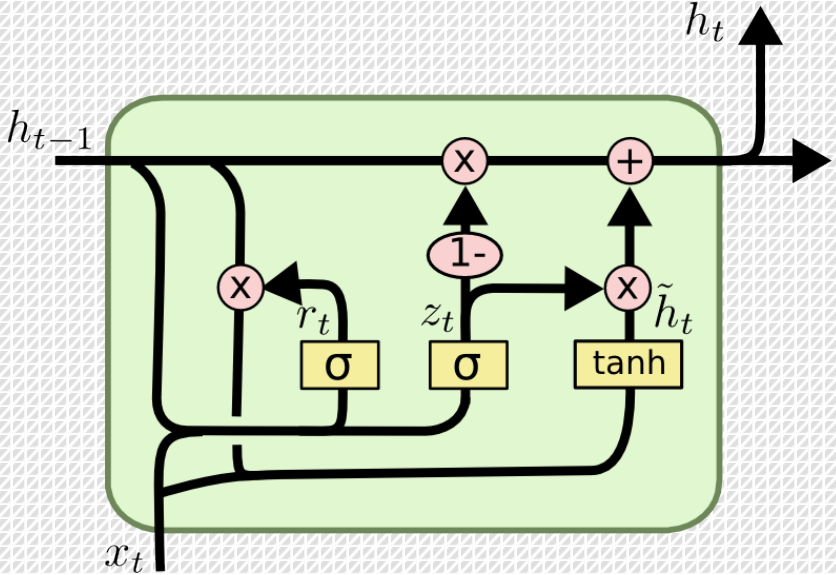
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM需要确定当前的输出，LSTM同样需要对h_{t-1}和x_t进行拼接并通过线性变换和Sigmoid激活函数生成输出。最后LSTM还需要通过o_t和C_t更新隐状态h_t 并将其输出至下一次迭代。

Recurrent Neural Network

- **长短记忆网络（Long Short Term Memory networks, LSTM）**



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

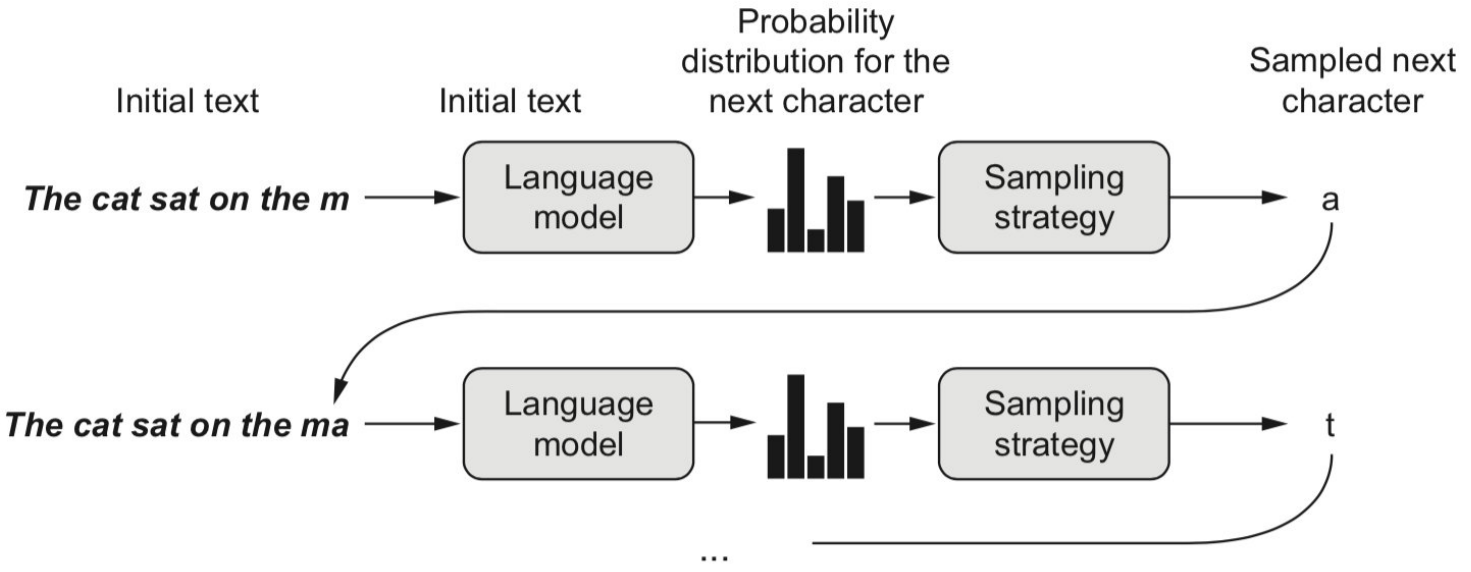
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

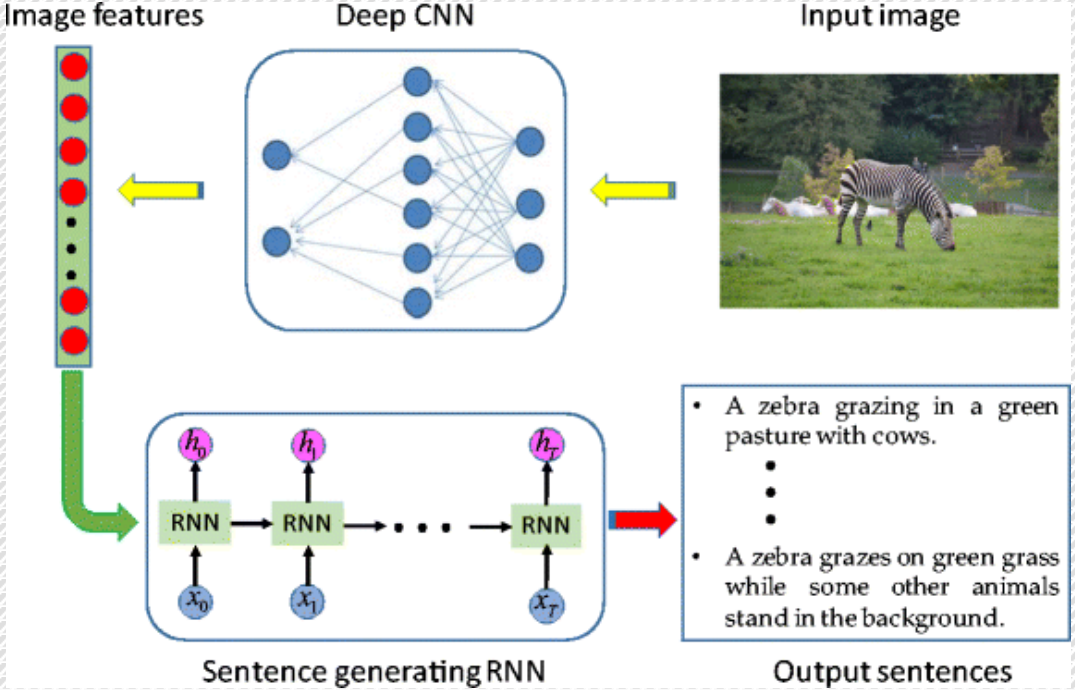
GRU是LSTM的一个改进版本，相比于LSTM，GRU将遗忘门与输出门结合为更新门，使其结构更加简单，其更新公式如图所示。

- 应用：文本生成（Text Generation）



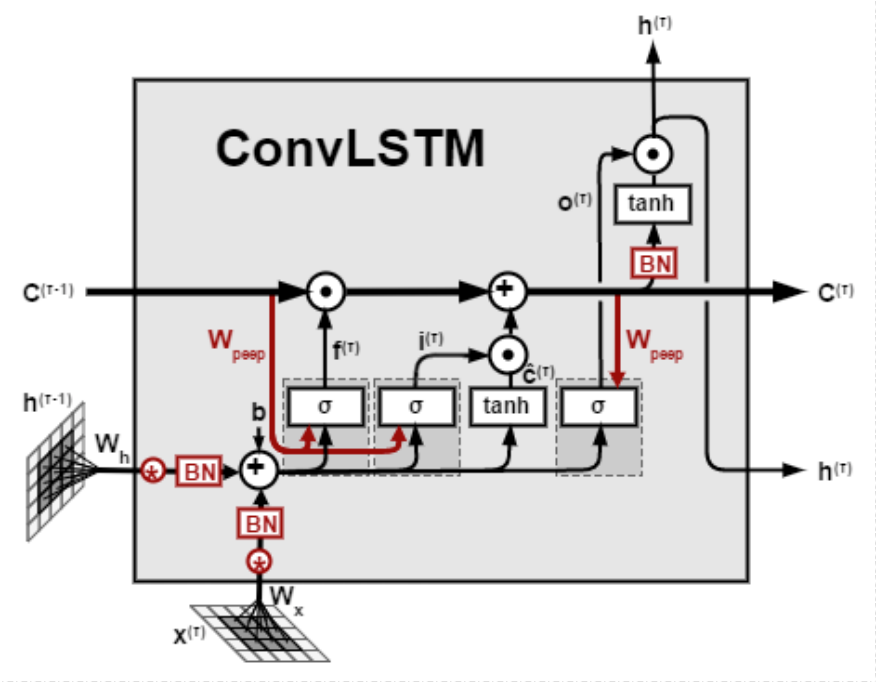
Recurrent Neural Network

- 应用：描述图像（Image Caption）

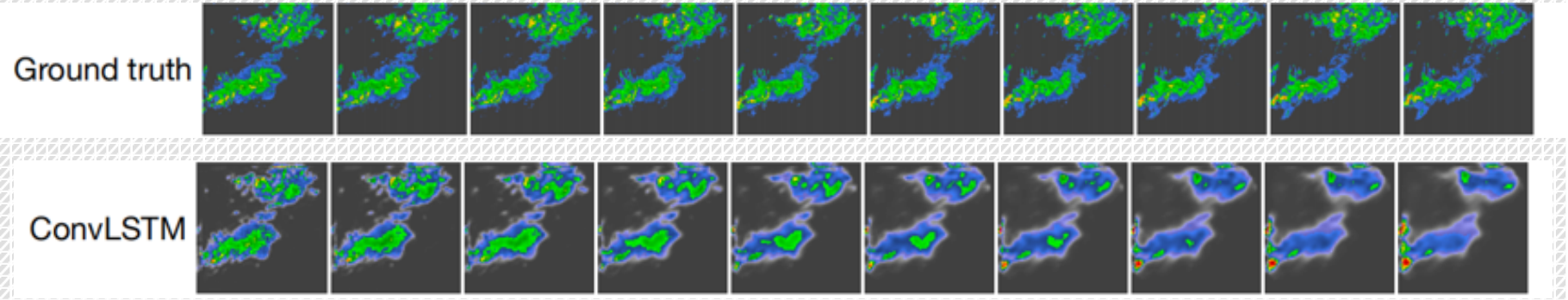


Recurrent Neural Network

- 卷积LSTM（ConvLSTM）



- 卷积LSTM（ConvLSTM）



Recurrent Neural Network

- RNNs are particularly effective in capturing short-term dependencies in sequences. However, they suffer from the vanishing gradient problem, where the influence of earlier inputs diminishes exponentially as the sequence progresses, making it difficult to capture long-term dependencies.
- LSTMs leverage memory cells and gates to selectively store and retrieve information over long sequences, making them effective for capturing long-term dependencies.

Transformer

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Transformer

- Difference between RNNs and Transformers
  - 3) Training and Parallelization:
    - For RNN, we mostly train it in a sequential approach, as the hidden state relies on previous steps. This makes parallelization more challenging, resulting in slower training times.
    - On the other hand, we train Transformers in parallel since they process data simultaneously. This parallelization capability speeds up training and enables the use of larger batch sizes, which makes training more efficient.
  - 4) Interpretability:
    - RNNs are well-suited for modeling sequential dependencies. They can capture contextual information from the past, making them effective for tasks like language modeling, speech recognition, and sentiment analysis.
    - Transformers excel at modeling dependencies between elements, irrespective of their positions in the sequence. They are particularly powerful for tasks involving long-range dependencies, such as machine translation, document classification, and image captioning.

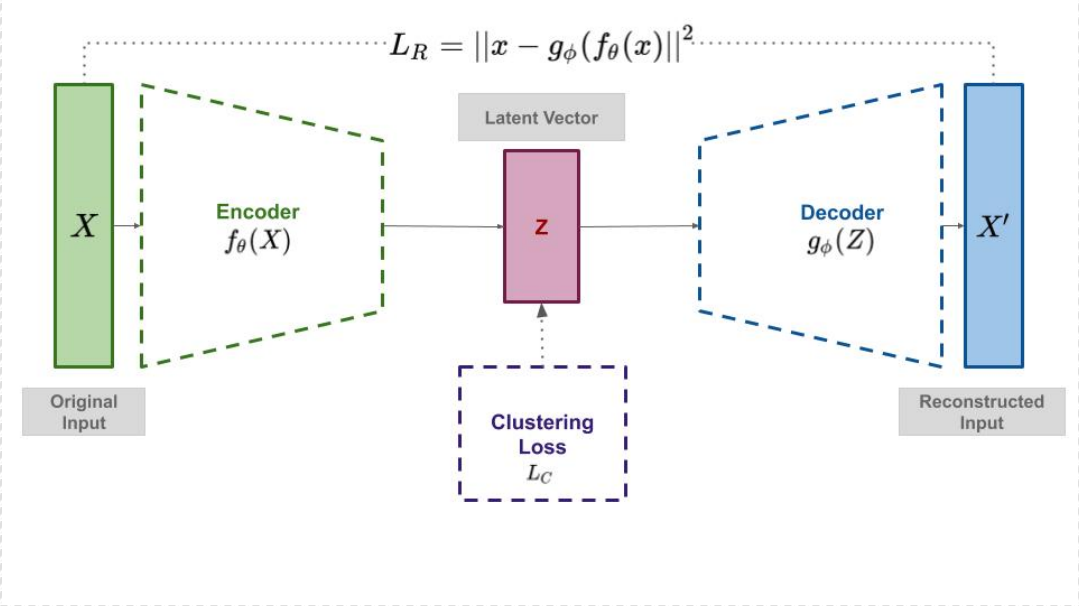
Transformer

- Difference between RNNs and Transformers
  - 3) Training and Parallelization:
    - For RNN, we mostly train it in a sequential approach, as the hidden state relies on previous steps. This makes parallelization more challenging, resulting in slower training times.
    - On the other hand, we train Transformers in parallel since they process data simultaneously. This parallelization capability speeds up training and enables the use of larger batch sizes, which makes training more efficient.
  - 4) Interpretability:
    - RNNs are well-suited for modeling sequential dependencies. They can capture contextual information from the past, making them effective for tasks like language modeling, speech recognition, and sentiment analysis.
    - Transformers excel at modeling dependencies between elements, irrespective of their positions in the sequence. They are particularly powerful for tasks involving long-range dependencies, such as machine translation, document classification, and image captioning.

- **自编码器（autoencoder，AE）**

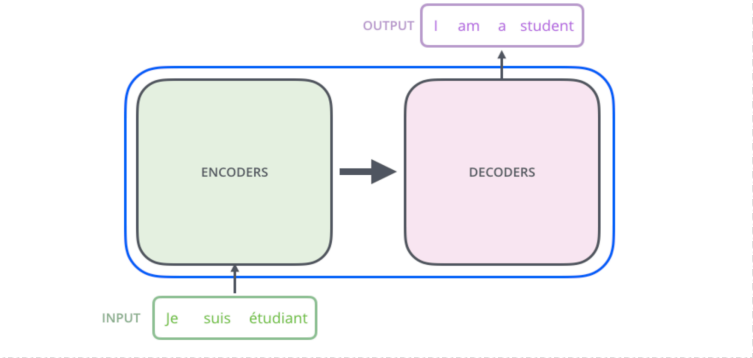
- Encoder-Decoder 模型是 NLP 领域里的概念。它并不特值某种具体的算法，而是一类算法的统称。Encoder-Decoder 算是一个通用的框架，在这个框架下可以使用不同的算法来解决不同的任务。
- Encoder-Decoder 这个框架很好的诠释了机器学习的核心思路：将现实问题转化为数学问题，通过求解数学问题，从而解决现实问题。

- 自编码器（autoencoder，AE）



Transformer

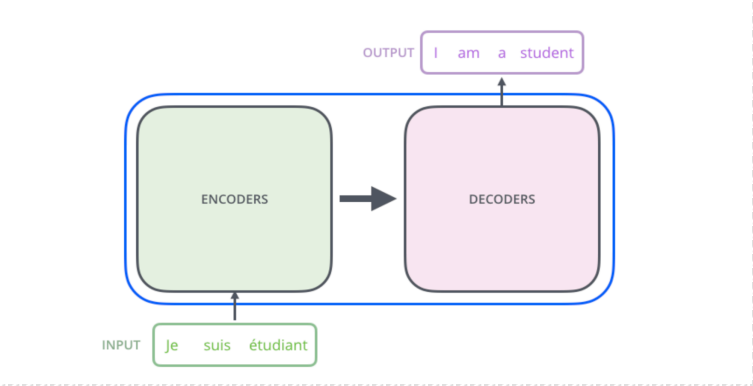
- A High-Level Look



https://jalammar.github.io/illustrated-transformer/

Transformer

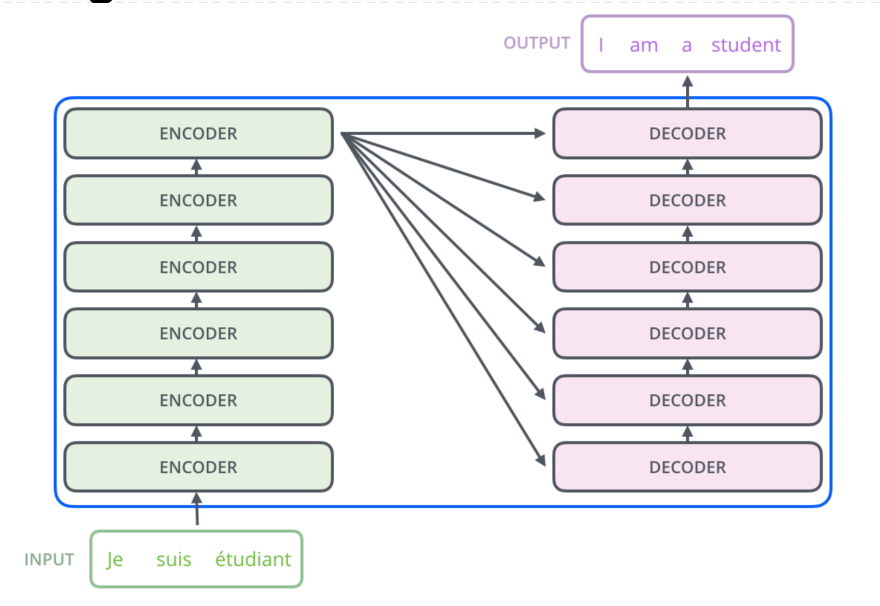
- **A High-Level Look**

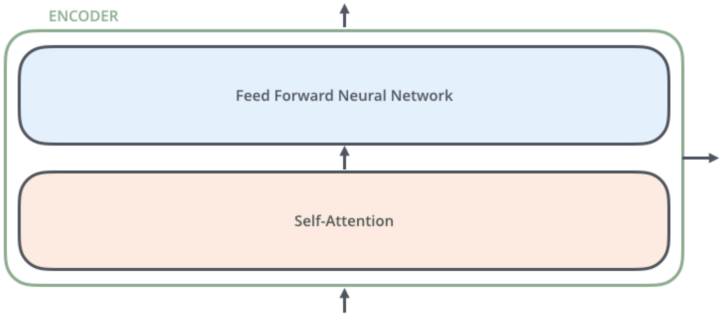


https://jalammar.github.io/illustrated-transformer/

Transformer

- **A High-Level Look**

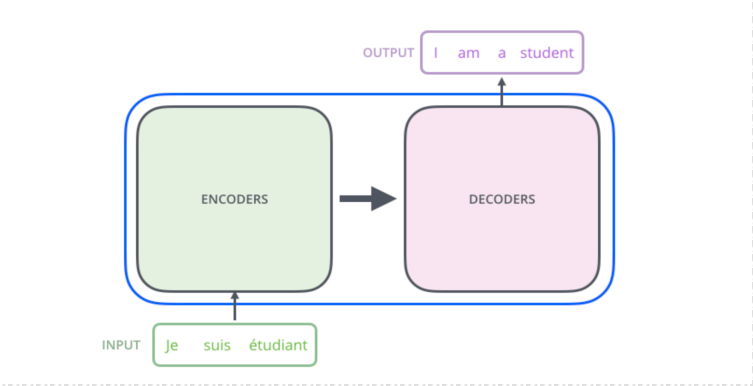




https://jalammar.github.io/illustrated-transformer/

Transformer

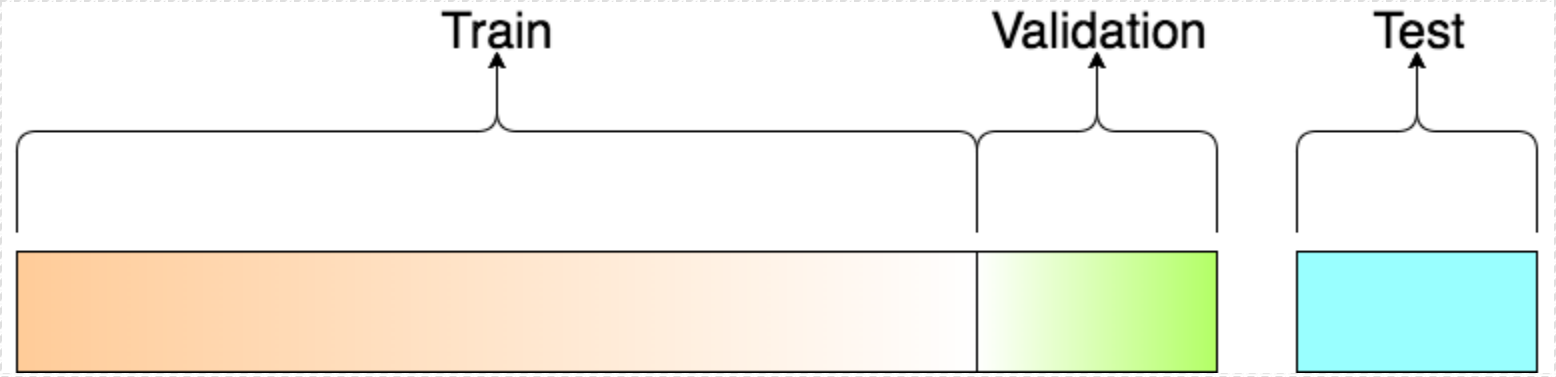
- **A High-Level Look**



https://jalammar.github.io/illustrated-transformer/

Dataset

- **数据的使用：数据集划分**



在我们使用数据集对网络模型进行训练时，我们通常不会使用全部数据对网络进行训练，而时先将其划分为训练集、验证集和测试集，其中训练集用来真正训练网络模型；验证集用来在训练过程中对网络模型作无偏估计，即验证网络性能；通常使用在验证集上表现较好的模型在测试集上作最终的测试。

Dataset

- **数据的使用：图像数据增强**

深度学习需要大量数据训练才能达到好的效果，但在一些情况下我们可能并没有足够多的数据，我们可以通过数据增强缓解这一问题。

Dataset

- **数据的使用：图像数据增强**
- 单样本数据增强：几何变换

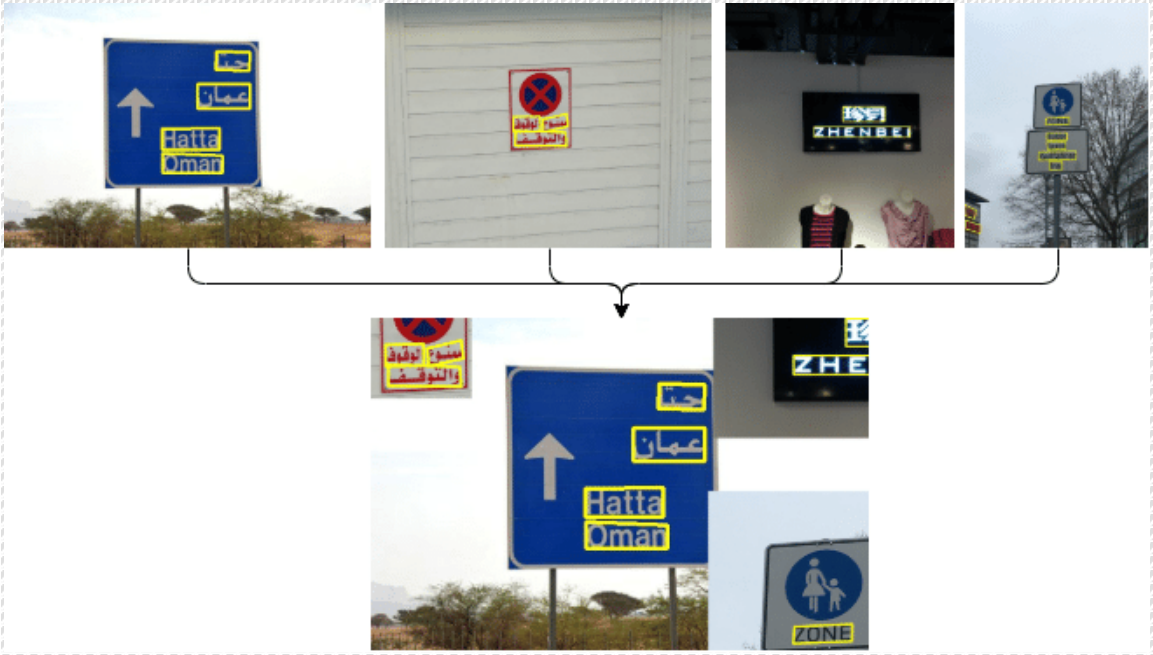
原图

裁剪

旋转

Dataset

- **数据的使用：图像数据增强**
- 多样本数据增强：Mosaic



Dataset

- **数据的使用：图像数据增强**
- 无监督数据增强：利用生成模型