Contents

I CNN Architectures

1. Case Studies

1.1 AlexNet

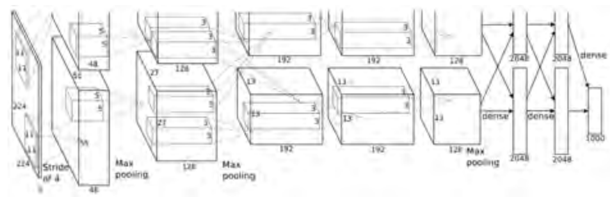

Figure I.1: AlexNet

通信网络

Architecture: Input $[227 \times 227 \times 3]$

1) CONV1: 96 filters 11×11 applied at stride 4, pad 0
   Output volume: $[55 \times 55 \times 96]$
2) MAX POOL1: 3×3 filters applied at stride 2
   Output volume: $[27 \times 27 \times 96]$
3) NORM1
4) CONV2: 256 filters 5×5 applied at stride 1, pad 2
   Output volume: $[27 \times 27 \times 256]$
5) MAX POOL2: filters 3×3 applied at stride 2
   Output volume: $[13 \times 13 \times 256]$
6) NORM2
7) CONV3: 384 filters 3×3 applied at stride 1, pad 1
   Output volume: $[13 \times 13 \times 384]$
8) CONV4: 384 filters 3×3 applied at stride 1, pad 1
   Output volume: $[13 \times 13 \times 384]$
9) CONV5: 256 filters 3×3 applied at stride 1, pad 1
   Output volume: $[13 \times 13 \times 256]$
10) Max POOL3: filters 3×3 applied at stride 2
    Output volume: $[6 \times 6 \times 256]$
11) FC6: 4096 neurons
    Output volume: [4096]
12) FC7: 4096 neurons
    Output volume: [4096]
13) FC8: 1000 neurons
    Output volume: [1000]

1.2 VGG

Small filters, Deeper networks.
小卷积核的堆叠感受野可以等效于大卷积核.

1.3 GoogLeNet

参数较少. 平行使用多个 filter 操作并将他们整合. 但复杂度高. 使用 1×1 conv 来改变特征维度以减少计算.

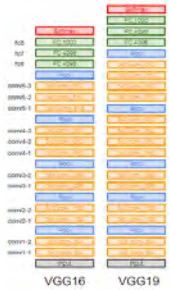

Figure I.2: VGG

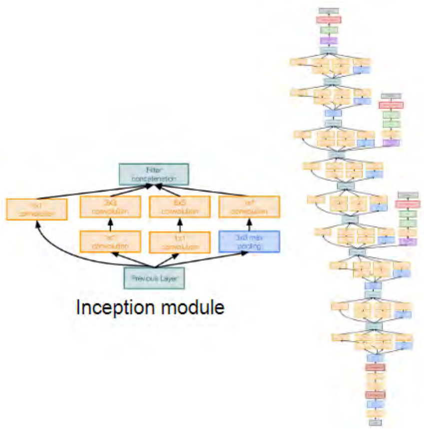

Figure I.3: GoogLeNet

因为太深了, 所以在中间增加了输出以更早的给出梯度.

1.4 ResNet

使用残差链接构建的极深的网络

单纯加深网络层数, 会让模型表现变差, 且不是由于过拟合造成的.

假设: 这是个优化问题, 更深的模型更难以优化.

解决方案: 所以只优化增量以降低优化难度 (直觉, 没有证明). $H(x) = F(x) + x$. Use layers to fit residual $F(x) = H(x) - x$ instead of $H(x)$ directly.

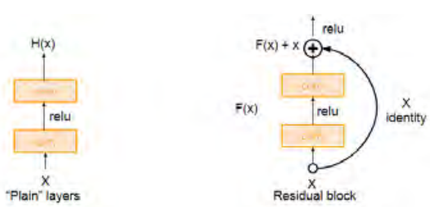

Figure I.4: Residual

这样有两个好处:

1) 若初始化所有权重为 0, 则表示此模块不进行计算, 这让网络更容易不使用不必要的层. 在使用 L2 正则时, 会让所有权重趋向于 0, 在传统网络中权重为 0 是无意义的, 但在残差网络中, 趋向于 0 鼓励网络不使用不必要的层.

2) 在梯度的传递中, 可以使用直接的链接更快速的传递上层的梯度, 降低了训练的难度.

2. Other architectures

2.1 NiN (Network in Network)

卷积层配合 MLP.

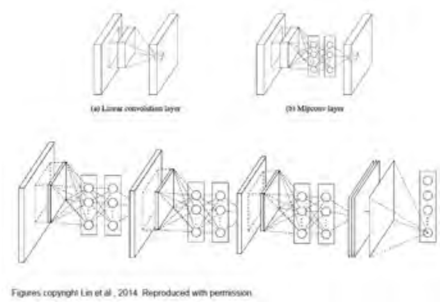

Figure I.5: NiN

2.2 Wide ResNet

更多的卷积层.

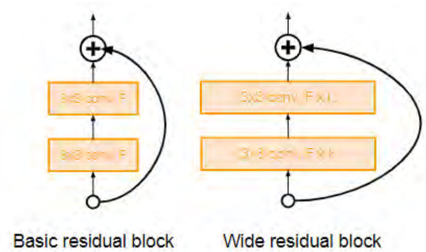

Figure I.6: Wide ResNet

2.3 ResNeXT

使用平行卷积提升宽度.

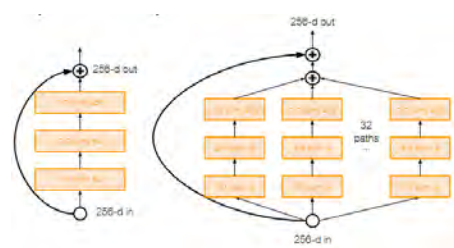

Figure I.7: ResNeXT

II Recurrent Neural Networks

Process Sequences

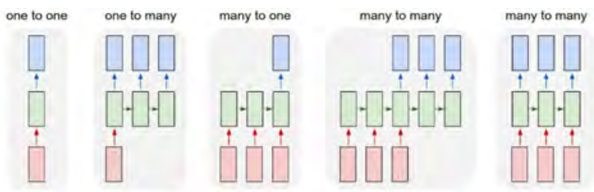

Figure II.1: Process Sequences

Sequential Processing of Non-Sequence Data, e.g. Classify images by taking a series of “glimpses”.

RNN process a sequence of vectors x by applying a recurrence formula at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$

- h_t: new state
- h_{t-1}: old state
- f_W: some functions with parameters W
- x_t: input vector at some time step

将在每个时间重复使用 f_W 及 W.

1. (Vanilla) Recurrent Neural Network

The state consists of a single “hidden” vector **h**:

$$\begin{aligned} h_t &= f_W(h_{t-1}, x_t) \\ &\downarrow \\ h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\ y_t &= W_{hy}h_t \end{aligned}$$

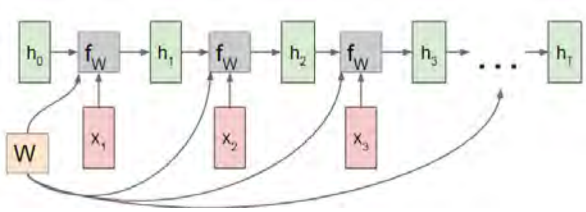

Figure II.2: RNN Computational Graph

e.g. Character-level Language Model Sampling

1.1 Backpropagation through time

Backpropagation through time: 在整个序列中前向传播计算 loss, 然后反向传播计算损失. (每次要过全部的训练数据, 难以接受)

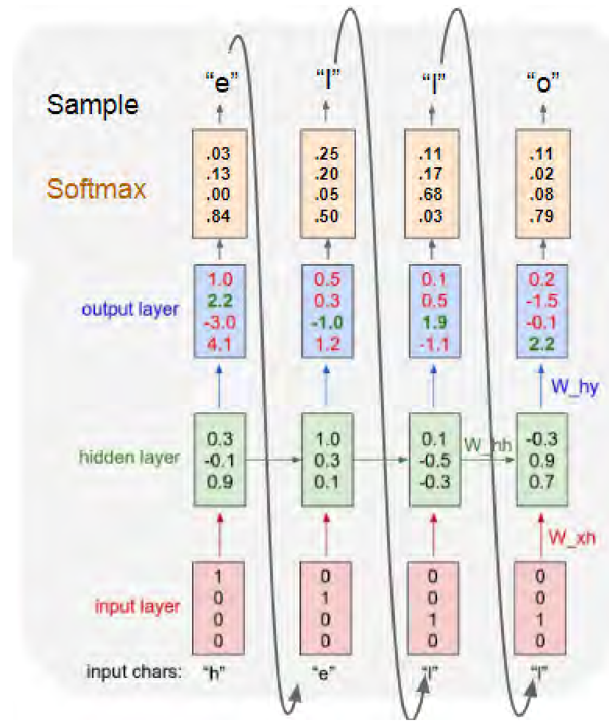

Figure II.3: Character-level Language Model Sampling

Truncated Backpropagation through time: 一块块序列进行 forward 与 backward. 具体的, 维护 forward 的隐藏层, 然后仅 backward 几层.

e.g. From 教科书的 latex or linux source generated 新的东西. 就是预测下一个字符.

探索向量的作用: e.g. 管理引号的, 管理换行的, 在代码中管理 if 语句的, 缩进的.

2. Image Captioning

输入: 图片的特征向量 (经过了例如 VGG 的前几层)
输出: 总结图片内容的单词.

2.1 Image Captioning with Attention

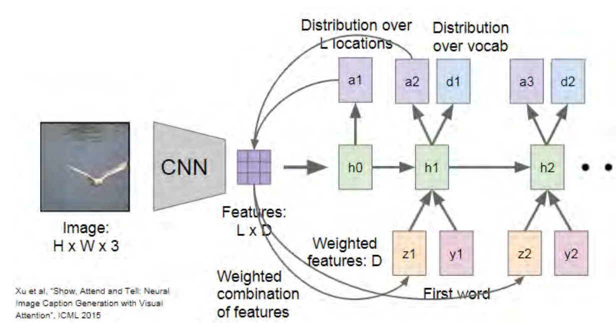

Figure II.4: attention

将图片的位置同样作为输入. 有 soft 和 hard 的区别, soft

是全局位置加权, hard 是强制选择一个位置.

RNN 可以多层.

3. Vanilla RNN Gradient Flow

Backpropagation from h_t to h_{t-1} multiplies by W (actually by W_{hh}^T). 但这样 h_0 的梯度会被所有 W 影响, 基本要么爆炸, 要么消失.

$$\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\ &= \tanh\left(\begin{pmatrix} W_{hh} & W_{xh} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \\ &= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \end{aligned}$$

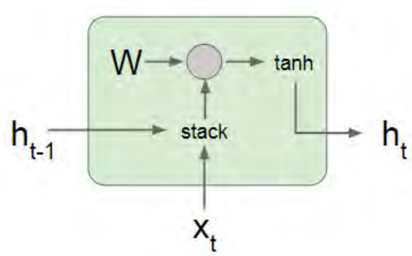

Figure II.5: Vanilla RNN

爆炸梯度可使用 gradient clipping 解决, 但消失需要使用更复杂的 RNN 结构.

4. Long Short Term Memory (LSTM)

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh c_t$$

h_t 是传递的, c_t 是本地的.

- **f**: **Forget gate**, Whether to erase cell
- **i**: **Input gate**, whether to write to cell
- **g**: **Gate gate** (?), How much to write to cell
- **o**: **Output gate**, How much to reveal cell

Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W. 可以保证 c 一条线的梯度被快速回传.

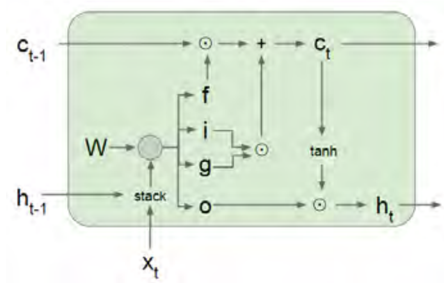

Figure II.6: LSTM

5. Other RNN Variants

e.g. GRU

III Detection and Segmentation

1. Semantic Segmentation

Label each pixel in the image with a category label. 对多个重叠物体不作区分.

Semantic Segmentation Idea:

1.1 Sliding Window

非常低效. 对重叠部分没利用共享的特征.

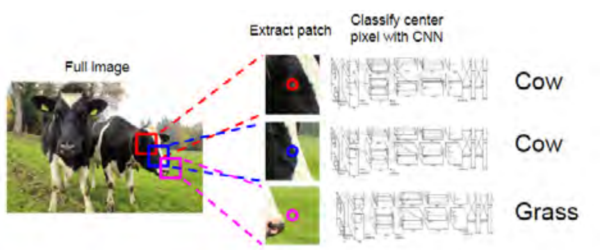

Figure III.1: Sliding Window

1.2 Fully Convolutional

源分辨率的卷积非常昂贵. 训练数据非常贵.

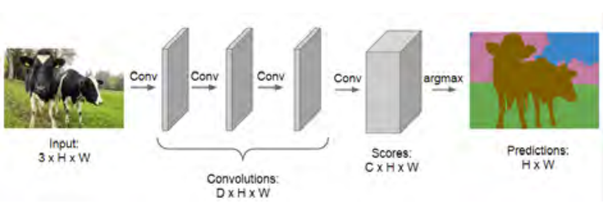

Figure III.2: Fully Convolutional

改进: with downsampling and upsampling inside

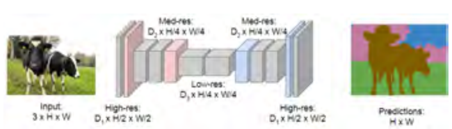

Figure III.3: downsampling and upsampling

- Downsampling: Pooling, strided, convolution
- Upsampling: Unpooling or strided transpose convolution

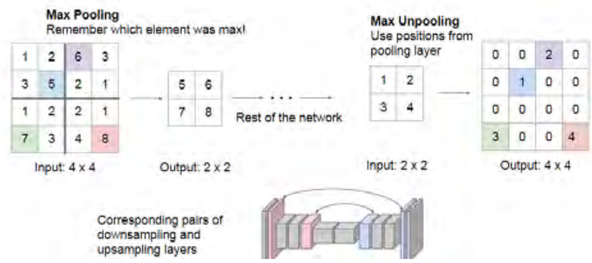

Figure III.4: Max Unpooling: 空值可以更好的传递 pooling 丢失的信息.

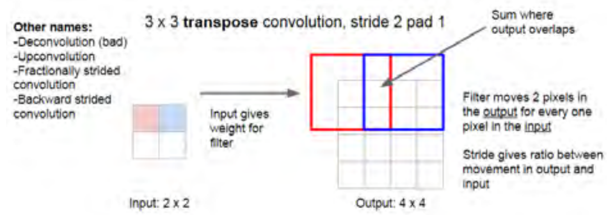

Figure III.5: Learnable Upsampling: Transpose Convolution(如果用矩乘表示卷积, 那么此方法是卷积的转置)

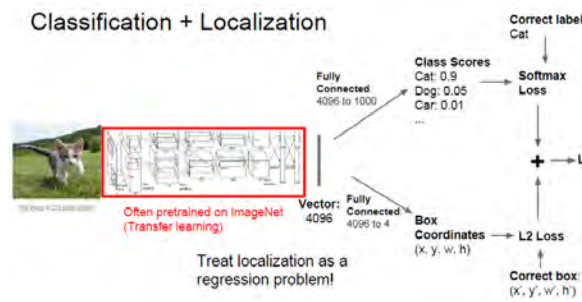

Figure III.6: Classification + Localization

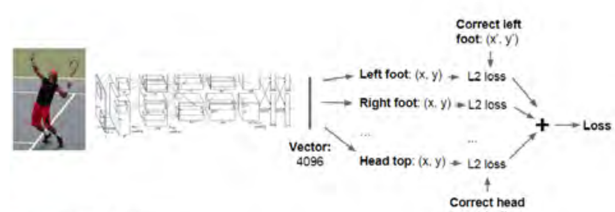

Figure III.7: Aside: Human Pose Estimation

2. Classification + Localization

两个 loss 通过超参数加权得到最终 loss, 用此反向传播.

3. Object Detection

As Regression: Each image needs a different number of outputs.

As Classification: Sliding Window. 获取许多 crop, 对其每个跑 CNN. Problem: Need to apply CNN to huge number of locations and scales, very computationally expensive!

3.1 Region Proposals

- Find "blobby" image regions that are likely to contain objects

- Relatively fast to run.

准确率不高, 但召回率高.

3.2 R-CNN

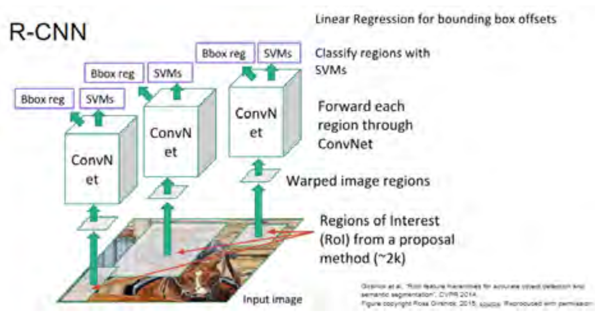

Figure III.8: R-CNN

Problems:

- 还是很贵
- 训练慢
- 测试也慢

3.3 Fast R-CNN

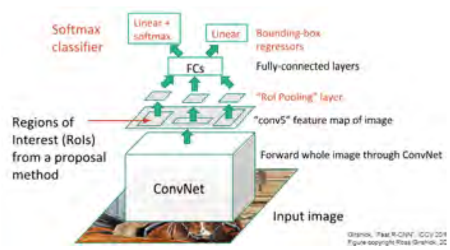

Figure III.9: Fast R-CNN

3.4 Faster R-CNN

Make CNN do proposals!

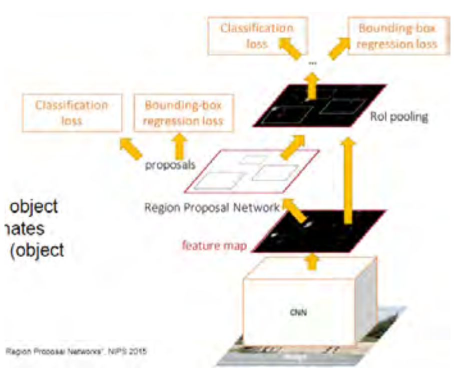

Figure III.10: Faster R-CNN

Jointly train with 4 losses:

1) RPN classify object / not object
2) RPN regress box coordinates
3) Final classification score (objec classes)
4) Final box coordinates

做了实验后表明只需要学一个.

3.5 Detection without Proposals: YOLO / SSD

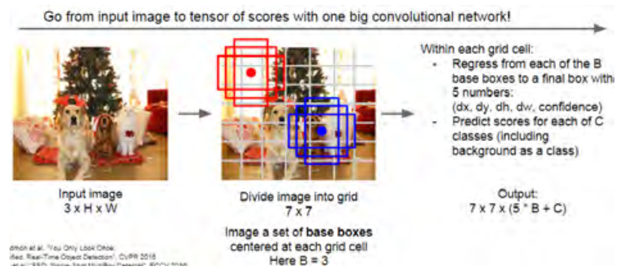

Figure III.11: Detection without Proposals

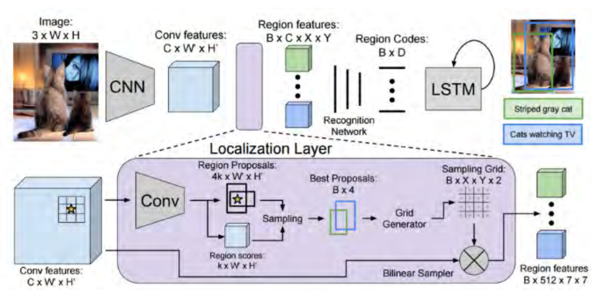

Figure III.12: Aside: Object Detection + Captioning = Dense Captioning

4. Instance Segmentation

4.1 Mask R-CNN

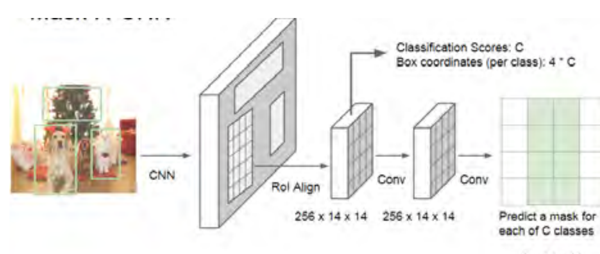

Figure III.13: Mask R-CNN

Also does pose