

L01 Overview

Visual Tasks 从 Low-Level 到 High-Level: Sensation → Processing → Perception → Cognition.

· **Data acquisition:** RGB 相机、深度相机、LiDAR 等. 得到照片、视频、点云、mesh 等.

· **Low-Level:** Processing 和特征提取. 图片降噪、去模糊、去水印、Edge/Corner 等.

· **Mid-Level:** 会对现实世界开始进行推断分析. 比如 3D 重建、全景照片合成.

· **High-Level:** 主要特征是会进行 semantic 级别的表征和解释. 比如场景内容理解、AR.

L02 Classic Methods

卷积定理: 卷积后的傅里叶变换 = 傅里叶变换的乘积.

导数定理: 可以预先计算 $\frac{d}{dx}g$, 则 $\frac{d}{dx}(f * g)$ 等于 $f * (\frac{d}{dx}g)$.

Key-point Detect 的要求: Repeatability、Saliency、Accurate localization、Sufficient number.

对 Flat region、Edge、Corner 的定义: 考察将窗口移动 (u, v) 后窗口内强度的改变量. 特别地对于 Corner 而言, 向任何方向移动都应该有显著的强度变化.

$$E(u, v) = \sum_{x,y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$

对上述公式中的中括号内使用一阶 Taylor 近似, 得到关于 u 和 v 的二次型. 记二次型矩阵为 M , 则

$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R \quad (\lambda_1 \geq 0, \lambda_2 \geq 0)$$

如果 λ_1 和 λ_2 都很大, 说明这是 Corner. 因为这样说明存在两个正交的方向, 窗口在这两个方向偏移后 E 都有显著变化!

不过考虑到正交对角化仍然具有一定的计算量, 使用快速近似:

$$\theta = \det(M) - \alpha \text{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

步骤总结:

- Compute second moment matrix (autocorrelation matrix)

$$M(\sigma_x, \sigma_y) = g(\sigma_x) \begin{bmatrix} I_x(\sigma_x) & I_x I_y(\sigma_x) \\ I_x I_y(\sigma_y) & I_y(\sigma_y) \end{bmatrix}$$

- 2. Square of derivatives



- 3. Gaussian filter $g(\sigma)$



- 4. Cornerness function - two strong eigenvalues

$$\theta = \det(M(\sigma_x, \sigma_y)) - \alpha(\text{trace}(M(\sigma_x, \sigma_y))^2 - g(I_x^2)g(I_y^2) - g(I_x I_y)^2)$$

- 5. Perform non-maximum suppression



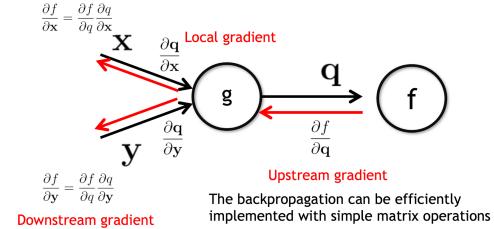
注意第 4 步中相当于为原图逐点计算了一个 θ . 如果用 numpy 实现, $I_x I_y$ 直接是矩阵即可. 当然如果窗口大小不是

1×1 , 那么 $I_x I_y$ 要改成 $\sum I_x \sum I_y$, 用全 $1/n$ 核卷积即可.

Harris Corner 的性质: 对平移和旋转 (Gauss or 1×1 才行吧...) 具有不变性, 但对缩放不具有不变性!

Line Fitting: LSE (法线方程 or SVD 技巧 (取 v_n)), RANSAC (Random Sample Consensus(共识)), RANSAC 抽 k 次全 fail 的概率 $(1-w)^k$. 选取 k 足够大即可. Hough 变换: 对于落入直线 $y=mx+n$ 的一堆 (x_i, y_i) , 为了求出 m 和 n , 以 m 和 n 为坐标轴绘制一堆直线 $y=mx+n$.

L03 Deep Learning I



Convolution layer: summary

Common settings:

Let's assume input is $W_1 \times H_1 \times C$. Conv layer needs 4 hyperparameters:

- Number of filters K
- $F = 3, S = 1, P = 1$
- $F = 5, S = 1, P = 2$
- $F = 5, S = 2, P = ?$ (whatever fits)
- The filter size F
- The stride S
- The zero padding P

This will produce an output of $W_2 \times H_2 \times K$ where:

- $W_2 = (W_1 - F + 2P)/S + 1$
- $H_2 = (H_1 - F + 2P)/S + 1$

Number of parameters: F^2CK and K biases

L04 Deep Learning II

General definition of equivariance:

$$S_A[\phi(X)] = \phi(T_A(X))$$

Here A is an operation, T_A and S_A are their transformation function in the space of X and $\phi(X)$.

CNN: Parameter sharing => Equivariance to Translation

Data Preprocessing

否则试想如果所有 data 都是正的, 则梯度符号相同, 考虑 2 个参数, 则只能在第一三象限前进, 如果真实最优在第四象限, 则会 zig-zag!

Weight Initialization

· Xavier Initialization: $W = \frac{1}{\sqrt{Din}} N(0, 1)$. 【推导: 希望 $y=Wx$ 满足 $\text{Var}(x)=\text{Var}(y)$, 假设 iid, 则 $\text{Var}(y)=\text{Din}\text{Var}(x)\text{Var}(w)$, 可知 w 的方差必须是 $1/\text{Din}$.】

· He Initialization: $W = \sqrt{\frac{2}{\text{Din}}} N(0, 1)$. 这是对 ReLU 有效的.

(对 ReLU 使用 Xavier 是不行的!)

对于卷积层而言, $Din = \text{filter_size}^2 \cdot n_input_channels$.

SGD: 条件数过高时抖动、局部极值、mini-batch 的 noise. 解决: Momentum (一阶), 或者 Adam (一阶和二阶).

SGD

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

while True:

$$dx = \text{compute_gradient}(x)$$

$$x \leftarrow \text{learning_rate} * dx$$

SGD+Momentum

$$v_{t+1} = \rho v_t + \nabla f(x_t)$$

$$x_{t+1} = x_t - \alpha v_{t+1}$$

vx = 0

while True:

$$dx = \text{compute_gradient}(x)$$

$$vx = rho * vx + dx$$

$$x \leftarrow \text{learning_rate} * vx$$

- Build up "velocity" as a running mean of gradients
- Rho gives "friction"; typically rho=0.9 or 0.99

Adam:

first_moment = 0

second_moment = 0

for t in range(1, num_iterations):

dx = compute_gradient(x)

$$\text{first_moment} = \beta_1 * \text{first_moment} + (1 - \beta_1) * dx$$

$$\text{second_moment} = \beta_2 * \text{second_moment} + (1 - \beta_2) * dx^2$$

$$\text{first_bias} = \text{first_moment} / (1 - \beta_1^{t+1})$$

$$\text{second_bias} = \text{second_moment} / (1 - \beta_2^{t+1})$$

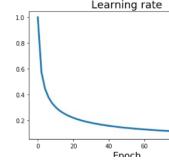
$$x = \text{learning_rate} * \text{first_bias} / (\text{np.sqrt}(\text{second_bias}) + \epsilon - 7)$$

Momentum

Bias correction

Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\text{learning_rate} = 1e-3$ or $5e-4$ is a great starting point for many models!

LR Schedule:



但初始 lr 太高可能会 loss 爆炸, 可以用 Linear Warmup.

训练 CNN 的两大难题 (train 时 underfit / test 时 overfit).

解决 underfit: BN (Batch Norm) 和 Skip link;

BN 的总结如下: 其中 N 是 Batch Size.

Input: $x : N \times D$

$\mu_j = \frac{1}{N} \sum_{i=1}^N x_{i,j}$ Per-channel mean, shape is D

Learnable scale and shift parameters:

$\gamma, \beta : D$

Learning $\gamma = \sigma$, $\beta = \mu$ will recover the identity function!

$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$ Normalized x, Shape is $N \times D$

$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$ Output, Shape is $N \times D$

γ 和 β 保证了这个正态分布

数在什么节点、以什么斜率被 ReLU 处理

在测试时, σ 和 μ 可以用恒定值. 这个恒定值可以在训练阶段通过 $\mu \leftarrow (1-p)\mu + p\mu$ 来获得, 仅用于预测.

对比 FC 和 Conv 的 BN:

Batch Normalization for fully-connected networks

Normalize

$\mu, \sigma : 1 \times D$

$\gamma, \beta : 1 \times C \times H \times W$

$y = \gamma(x - \mu) / \sigma + \beta$

Batch Normalization for convolutional networks (Spatial BatchNorm, BatchNorm2D)

Normalize

$\mu, \sigma : 1 \times C \times H \times W$

$\gamma, \beta : 1 \times C \times 1 \times 1$

$y = \gamma(x - \mu) / \sigma + \beta$

BN 的变种:

为什么不是 N 的一维数组 norm? 因为一维会破坏 translation invariance, 也就是平移一格变化很大

③ Instance Norm

④ Group Norm

⑤ Layer Norm

⑥ 对应维数平均

Widely used in NLP!

这里假设每个 channel 的分布相同...

很难成立!

而不用②③.

⑦ 如果实在太小, 用这个!

而不使用②③.

Group Norm 当 Batch size 非常小的时候, 或者每个 Batch

的相关度太高 (correlated) 时, 会 outperform Batch Norm.

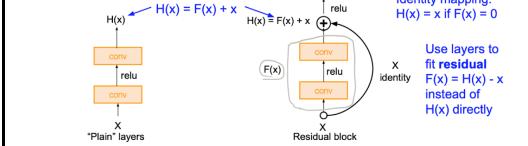
Batch Norm 一般插入在 FC/Conv 后, 但在 nonlinear 前.

比如介于 FC 和 tanh 之间.

- BN 的 Pros: 让深层网络非常容易训练, 改善了梯度流动, 允许更高的 lr, 更快收敛, 对初始化更 robust, 作为训练时的正则化, 测试时零开销 (可以和 Conv 融合).
- BN 的 Cons: 在 Train 和 Test 阶段具有不同的表现, 是 bug 的常见来源!

ResNet: 用到 Skip Link / Residual Link 的技巧

Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping



Skip Link 可以让 Loss Landscape 变得平滑从而容易优化.

L05 Deep Learning III

Generalization gap: the difference between a model's performance on training data and its performance on unseen data drawn from the same distribution.

如果分布不同, 叫 Domain Gap. (e.g. 猫狗 min 野猪 test)

缓解 Generalization Gap 的思想: 平衡 data variability 和 model capacity.

· 从 data 角度: Data Augmentation、Batch Norm...

· 从 model 角度: Regularization、Dropout... (Slides 说这两种一般仅用于大型 FC Layer, 而上面两种则总可以使用.)

Softmax: sigmoid 的高维推广. 一般来说用到 $\exp(x)$, 不过一般形式是 $\exp(\beta x)$, 其中 β 如果是 ∞ 则退化为 argmax.

$$D_{KL}(P || Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right).$$

KL-Divergence:

其中 P 被视为 reference / ground truth prob., 而 Q 则是预测的 prob. 这样上式可以推出 Cross-Entropy Loss ($H(P, Q)$).

$$D_{KL}(P || Q) = - \sum_{x \in \mathcal{X}} P(x) \log Q(x) - \left(- \sum_{x \in \mathcal{X}} P(x) \log P(x) \right)$$

$$H(P) = \sum_{x \in \mathcal{X}} P(x) \log P(x)$$

VGG-Net: Small Filter & Deeper Network

· Receptive Field 的概念: 3 个 3x3 conv filter 的 receptive field 是 7x7, 但相比一个 7x7 核不仅少了参数, 还增加了 non-Linear.

ResNet 的 Bottleneck 技巧

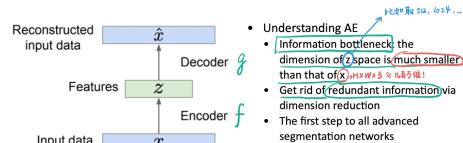


Dense-Net: Dense Block 内部的层之间全连接, 缓解梯度消失, 鼓励特征重用.

MobileNet 的所谓 Depth-Wise Conv: 假设有 C 个 channel, 那么现在每个 filter 不再是 nxnxC, 而是只是 nxn1, 即每个 channel 对应一个 filter! 这样减轻了 C 倍计算量.

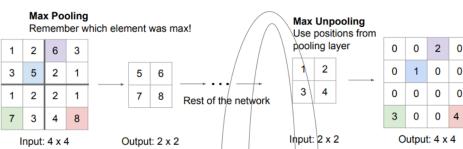
L06 Deep Learning IV

Auto Encoder

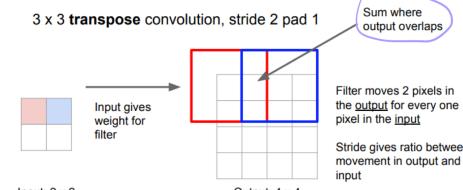


$$\text{Re: reconstruction loss} \triangleq \|\hat{x} - x\|^2$$

Max unpooling:



Learnable unsampling: Transposed Convolution:



纯 Conv 的 Semantic Segmentation 中 Bottle-Nect 的意

义: 很小的内存占用+极大的 receptive field (global context).

UNet: 轮廓信息难以 encode 进 bn, 使用 skip-link.

· 评估 segmentation 的 metric: IoU (Intersection over Union). 以及 Soft IoU:

$$IoU = \frac{\sum_{v \in V} X_v * Y_v}{\sum_{v \in V} X_v + \sum_{v \in V} Y_v}$$

$$Soft\ IoU = 1 - IoU = 1 - \frac{\sum_{v \in V} X_v * Y_v}{\sum_{v \in V} X_v + \sum_{v \in V} Y_v}$$

L07 3D Vision

$$P = \begin{bmatrix} \alpha & -\alpha \cot\theta & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \begin{array}{l} x \\ y \\ z \\ 1 \end{array} \quad \begin{array}{l} f = \text{focal length} \\ u_0, v_0 = \text{offset (Or } C_x, C_y) \\ \alpha, \beta \rightarrow \text{non-square pixels} \\ \theta = \text{skew angle} \end{array}$$

K has 5 degrees of freedom!

综合 Intrinsic 和 Extrinsic 的变换, 记整体变换为 M, 则有

$$M_{3 \times 4} = K_{3 \times 3} [R_{3 \times 3} T_{3 \times 1}]$$

[Eq. 9]	Internal parameters	External parameters
$P = K \begin{bmatrix} I & 0 \end{bmatrix}$	$P = K \begin{bmatrix} I_m & 0 \end{bmatrix}$	$P_w = K \begin{bmatrix} R & T \end{bmatrix} P_w$
3维	w=cam	world

[Eq. 11]

记 M 的行向量为 m_1, m_2, m_3 , 记世界坐标系下点坐标为 P_w , 则映射后得到的 2 维点为 $(\frac{m_1 P_w}{m_3 P_w}, \frac{m_2 P_w}{m_3 P_w})$.

Weak Projective Camera: 对 Projective 的简化. 变换矩阵的对比如下图所示:

Projective (perspective)	Weak perspective
$M = K[R \ T] = \begin{bmatrix} A_{2 \times 3} & b_{2 \times 1} \\ V_{2 \times 3} & 1 \end{bmatrix} \rightarrow M = \begin{bmatrix} A & b \\ 0 & 1 \end{bmatrix}$	

Calibration problem

一对已知的 p_i, p_i' 对可以给出 2 行方程. 由于 M 的自由度为 $5 + 3 + 3 = 11$, 所以需要 6 对.

$$[Eq. 1] \quad \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} m_1 P_i \\ m_2 P_i \\ m_3 P_i \end{bmatrix} \quad \begin{array}{l} \text{-对 correspondence} \\ \text{可得到2个方程.} \end{array}$$

$$u_i = \frac{m_1 P_i}{m_3 P_i} \rightarrow u_i(m_3 P_i) = m_1 P_i \rightarrow u_i(m_3 P_i) - m_1 P_i = 0$$

$$v_i = \frac{m_2 P_i}{m_3 P_i} \rightarrow v_i(m_3 P_i) = m_2 P_i \rightarrow v_i(m_3 P_i) - m_2 P_i = 0$$

$$\begin{array}{l} -u_i(m_3 P_i) + m_1 P_i = 0 \\ -v_i(m_3 P_i) + m_2 P_i = 0 \\ \vdots \\ -u_n(m_3 P_n) + m_1 P_n = 0 \\ -v_n(m_3 P_n) + m_2 P_n = 0 \end{array} \quad \begin{array}{l} \text{known} \\ \text{unknown} \end{array}$$

$\boxed{P \ m = 0}$ [Eq. 4]

Homogenous linear system

$$P \ def \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} \quad 12 \times 1$$

考虑到存在平凡解 $m = 0$, 将问题转化为约束 $|m| = 1$ 下最小化 $|Pm|$ 的问题. 这等价于 SVD 中求最后一个 v_n ! 即 $M = v_n$.

但首先这样的 M 是单位化的, 真实的 M 还会相差某个 ρ 倍.

另外问题是如何反解出每个参数 $\alpha, \beta, \theta, u_0, v_0, r_1, r_2, r_3$. T. 结论为:

$$M = \rho \hat{M} = \begin{bmatrix} \alpha r_1^T & \alpha \cot\theta r_2^T + u_0 r_3^T \\ \beta r_2^T & v_0 r_3^T \\ r_3^T \end{bmatrix}$$

A B $K[R \ T]$

Intrinsic

$$\hat{M} = \begin{bmatrix} \hat{A}_1^T & \hat{B}_1 \\ \hat{A}_2^T & \hat{B}_2 \\ \hat{A}_3^T & \hat{B}_3 \end{bmatrix}$$

$$\hat{A} = \begin{bmatrix} \hat{a}_1^T \\ \hat{a}_2^T \\ \hat{a}_3^T \end{bmatrix} \quad \hat{B} = \begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \end{bmatrix}$$

$$\hat{a}_i = \pm \frac{1}{\hat{\rho}} \hat{a}_i \quad \hat{b}_i = \pm \frac{1}{\hat{\rho}} \hat{b}_i$$

$$\cos\theta = (\hat{a}_1 \times \hat{a}_3) \cdot (\hat{a}_2 \times \hat{a}_3) / |\hat{a}_1 \times \hat{a}_3| |\hat{a}_2 \times \hat{a}_3|$$

Extrinsic

$$r_1 = \frac{\hat{a}_2 \times \hat{a}_3}{\hat{\rho}} \quad r_2 = \frac{\hat{a}_1 \times \hat{a}_3}{\hat{\rho}} \quad T = \rho K^{-1} \hat{b}$$

True edge
Low precision
Poor localization
Too many responses

Precision = $\frac{TP}{TP + FP}$
Recall = $\frac{TP}{TP + FN}$

TP: true positives, FP: false positives
TN: true negatives, FN: false negatives

· High precision: make sure all detected edges are true edges (via minimizing FP).

· High recall: make sure all edges can be detected (via minimizing FN).

· Good localization: minimize the distance between the detected edge and the ground truth edge

· Single response constraint: minimize redundant responses

现在使用方向信息和较小的阈值, 我们将“生长”这些边缘。

· 如果当前像素不是边缘, 请检查下一个。

· 如果它是边缘, 请检查沿着边缘方向 (即垂直于梯度方向) 的两个像素。如果其中任何一个 (或者两者都) :

· 方向与中心像素处于同一区间

· 梯度幅度小于 minVal

· 它们相邻边缘是最大的 (对这些像素进行非极大值抑制NMS), 那么你可以标记这些像素为边缘像素

· 循环直到图像没有变化

· 一旦图像停止变化, 你就得到了 canny 边缘!

跟BN不同, 后面三个在train和test上都是一样的, 没有moving avg mean, 因为没有对batch size做normalization. BN需要在test的时候使用train的moving avg mean

为了防止test的时候batch size过小, BN需要在test的时候使用train的moving avg mean

三层3*3比7*7多了两个relu, 相信性能更好并且参数变少

每pool一层, 长宽都是channel数量变成两倍

参数-k*2*2*2, 参数变成四倍

显存-smC, 显存变成二分之一

Summary of Semantic Segmentation: A top-down approach

Bottleneck structure:

· Large receptive field and provides global context

· Get rid of redundant information

· Lower the computation cost

· Skip link: Assist final segmentation, Avoid memorization

假设我们在知道一个照片以外, 还知道每个像素的深度, 那么可以找出真实世界中两点的距离吗? 不可以。因为不知道相机内参, Δu 和 Δv 代表着像素差, 无法确定相机参考系下的 $\Delta x/\Delta y$ 真实距离差。

1. Depth back projection: intrinsic $K, u, v, z, \rightarrow \Delta x \Delta y$ 使用K的定义

2. Camera calibration: $x, y, z, \rightarrow K$ 如果不知道K, 那么就是相机标定。

一些问题:

1. 为什么相机标定的时候所有参考点不能在同一个平面上?

如果所有参考点都在同一个平面上, 那么相机的观测结果缺乏深度信息, 因为所有的标定点都在同一个平面上, 这会导致所谓的退化配置 (degenerate configuration)。在这种配置下, 我们不能唯一地确定相机的内外参数, 尤其是关于深度和空间位置的信息, 例如, 我们无法区分相机距离标定平面远近焦距, 与相机距离标定平面近低焦距的情况。这两种情况在所有参考点都在同一平面上时可能会产生相似的投影图像。

2. 根据 depth back projection 计算出来的相机坐标系下的 $\Delta x \Delta y$ 是不是 world coordinate 之间的距离? 是的, 因为角保距离不变。

Chamfer distance: We define the Chamfer distance between $S_1, S_2 \subseteq \mathbb{R}^3$ as:

$$d_{Cham}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2$$

Earth Mover's distance: Consider $S_1, S_2 \subseteq \mathbb{R}^3$ of equal size $|S_1| = |S_2|$. The EMD between A and B is defined as:

$$d_{EMD}(S_1, S_2) = \phi(S_1) \rightarrow S_2 \min_{\pi} \sum_{(x, y) \in S_1 \times S_2} \|x - \phi(x)\|_2$$

where $\phi: S_1 \rightarrow S_2$ is a bijection.

Point Set Generation Network for 3D Object

What Points are Keypoints?

· 重复性: 在两幅图像中独立检测相同的点

· 显著性: 有趣的点

· 精确定位

· 数量: 足够多

为了使关键点检测器具有可重复性, 它必须对以下因素保持不变: · 照明 · 图像尺度 · 观察点

First-order Taylor expansion: $I[x + u, y + v] - I[x, y] \approx I_u u + I_v v$

$$\therefore D(x, y) = (I[x + u, y + v] - I[x, y])^2 \approx (I_u u + I_v v)^2 = [u, v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} [u, v]$$

对平移和图像旋转等效, 对规模不是等效的。
等变: $F(Tx) = F(x)$, 对于 translation 和 rotation 是等变的
不变: $F(Tx) = F(x)$, 也就是对于不同位置导出的角点还是那样, 所以其实我们想要的是等变, 也就是对于不同位置导出的角点做了同样的变化

如何证明 Harris detector is equivariant?
只要说明角点检测函数也是equivariant即可。角点检测函数包括了求导和卷积两个操作, 显然求导是equivariant的, 因为导数会随着trans和rot做相同的变化。很有趣的是卷积也是equivariant的: 当你的filter function是各向同性的, 那么这个卷积就是equivariant的; 但是如果是一个椭圆形的window, 那这个卷积就不是equivariant的了

如何对corner-response-function做NMS? 先给出一个阈值, 把所有response排序, 成为一个list, 从上到下按照把这个pixel周围的大于阈值的踢出list。这个跟之前的NMS区别在于之前需要一条边, 现在只需要一个点, 那么现在比之前踢出的更多。

这个高光说明不是env Illumination variant的

Scale Invariant Detectors: Harris-Laplacian, SIFT (Lowe)

$A = h = 0$ A is rank deficient

Minimize $\|A\|$ subject to $a^2 b^2 c^2 = 1, h=(a,b,d)$ 如果A是矩阵, 那么h取最小特征值的特征向量

$A = UDV^T$

h = last column of V 特征向量

$$\mathcal{L}_{CE} = H(P, Q) = - \sum_{x \in \mathcal{X}} P(x) \log Q(x)$$

· With random initialization, $\mathcal{L}_{CE} \approx 1/(\# \text{ of classes})$ 所以loss曲线上升在log(N)的地方开始下降

· No upper bound.

· Minimum = 0.

CEL and acc 的关系:

1. CEL 最低在log2的时候, acc 仍然可能是0, 因为 probability=[0.499, 0.501], 仍然输出错误答案, 但是 loss=log2比较小

2. acc是100%的时候, CEL仍然可能是初始化的log(N), 同理probability=[0.499, 0.001, 0.001, ..., 1.000], 总结所述, 两者没有确定关系, 训练一定要同时画两个曲线

INPUT: 224x224x3 memory: 224*224*3*16MB, params: 0 (not counting biases)

CONV-1: 224x224x32 memory: 224*224*3*2*32=128MB, params: 0

POOL-1: 112x112x32 memory: 112*112*32*32=800K, params: 0

CONV-2: 112x112x64 memory: 112*112*32*64=2.256MB, params: 0

POOL-2: 56x56x64 memory: 56*56*64*64=204.912

CONV-3: 56x56x128 memory: 56*56*64*128=2.56K

CONV-3: 28x28x128 memory: 28*28*128*128=2.56K

CONV-4: 28x28x256 memory: 28*28*128*256=2.56K

CONV-4: 14x14x256 memory: 14*14*128*256=2.56K

CONV-5: 14x14x512 memory: 14*14*256*512=2.56K

CONV-5: 7x7x512 memory: 7*7*512*512=2.56K

FC: 7x512 memory: 7*7*512*4096=2.56K

FC: 4096*4096 memory: 4096*4096=16.772.000

TOTAL memory: 24 bytes * 4 bytes / image (only forward) -*2 for bwd

TOTAL parameters: 138M parameters

Stereo System

$$\frac{u-w}{f} = \frac{B \cdot f}{z} = \text{disparity} \quad [\text{Eq. 1}]$$

Note: Disparity is inversely proportional to depth

只有内参或者深度信息, 都不能确定一个物体

所以必须两个都知道才能得到物体真实数据

只有内参: 外参和物体大小同时变化

只有深度信息: 物体在平行于相机的那个平面上的大小可以变化, 因为这个平面上的物体是同一个深度