基于对抗生成网络(GAN)的人脸图像修复

汇报人：陈阳

1. GAN 的介绍

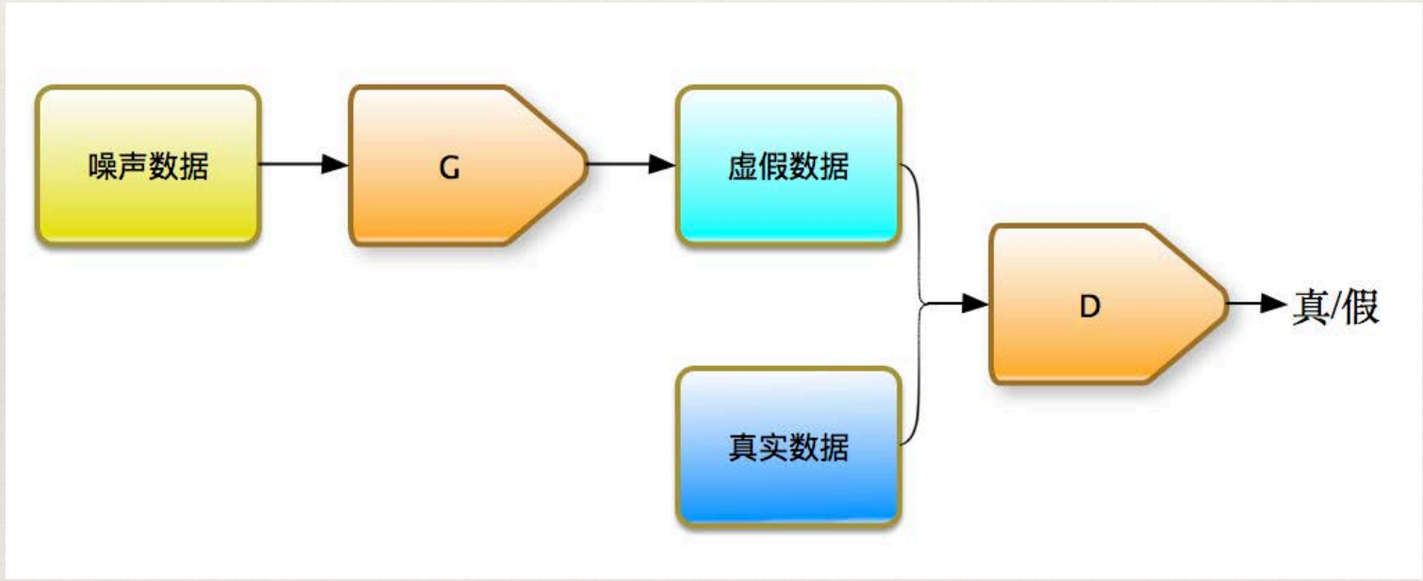
对抗生成网络 (generative adversarial network, GAN):
最早是由 Google Brain小组的人工智能专家 Ian Goodfellow 和 Yoshua Bengio 于 2014 年提出一个网络模型，灵感来源于二人博游中的零和博弈，也是当今非常火热的非监督深度学习模型之一。

Facebook 的人工智能专家 Yann Lecun 说 (2016.7.29 于 问答网站 Quora):
“There are many interesting recent development in deep learning, probably too many for me to describe them all here. The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks).”

GAN 的基本思想

GAN 包括两部分:
生成模型 G (Generative Model)，判别模型 D (Discriminative Model)

G: 捕获数据的分布
D: 估计数据来源于真实样本的概率



类比：警察 (D) 和假币制造者 (G) 之间的博弈

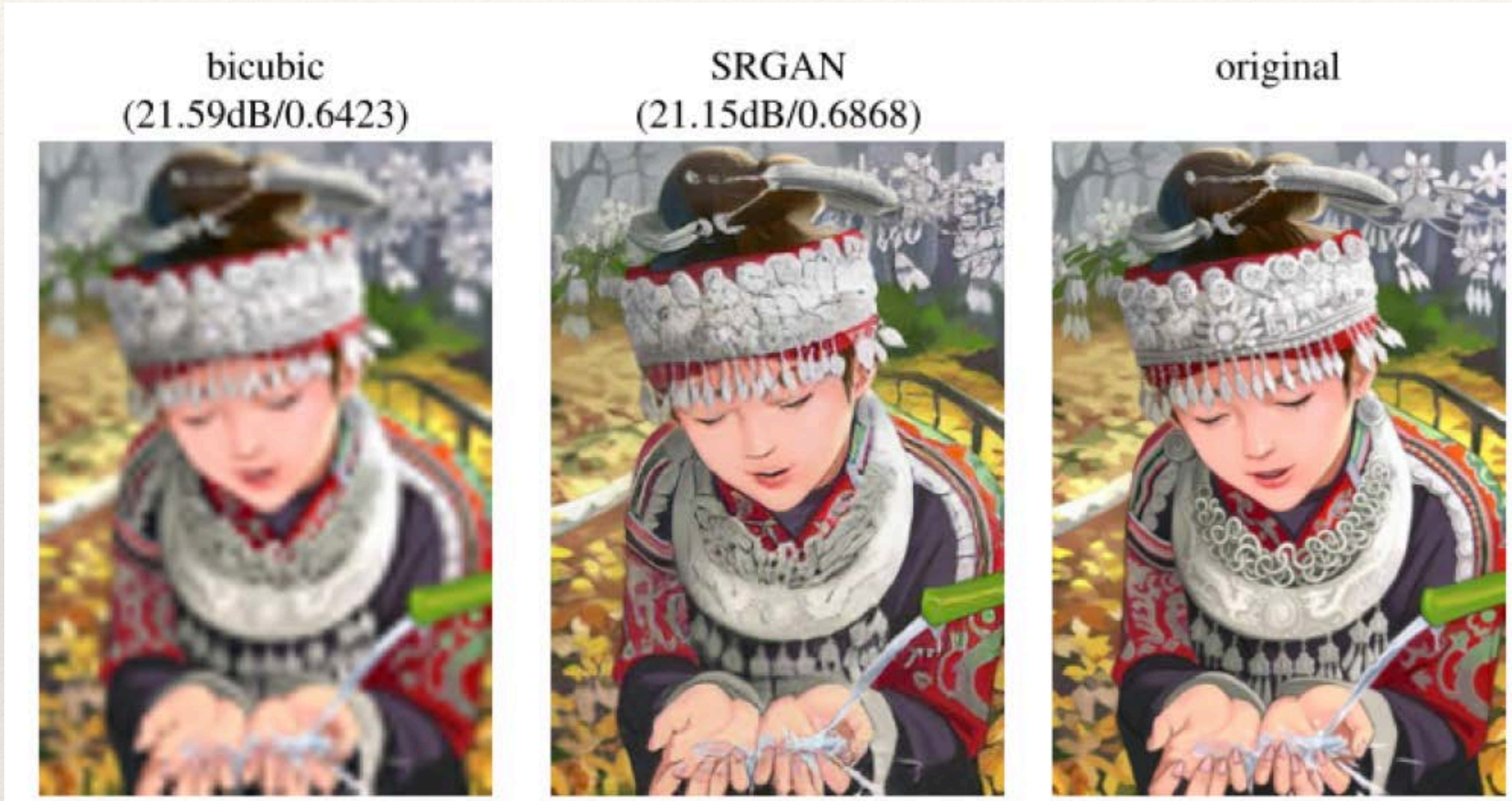
GAN 的应用

人脸生成

案例：2016.9 Uber 中国幽灵车事件

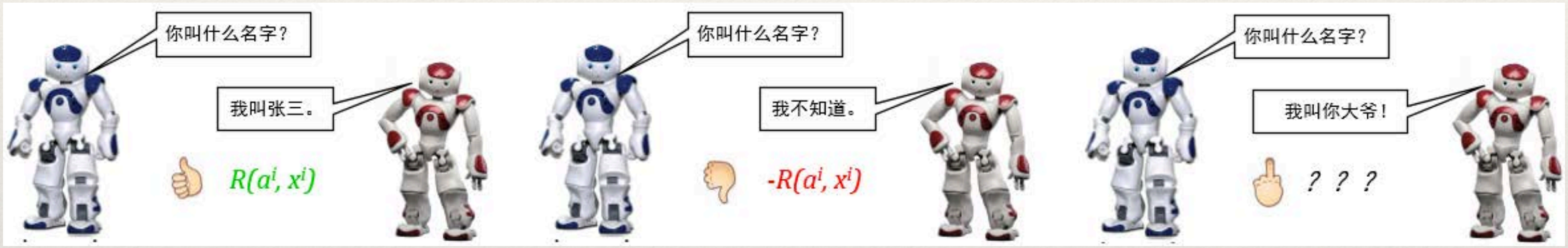
GAN 的应用

图像超分辨率重建



GAN 的应用

NLP 对话机器人



2. GAN 的理论推导

搭建模型：

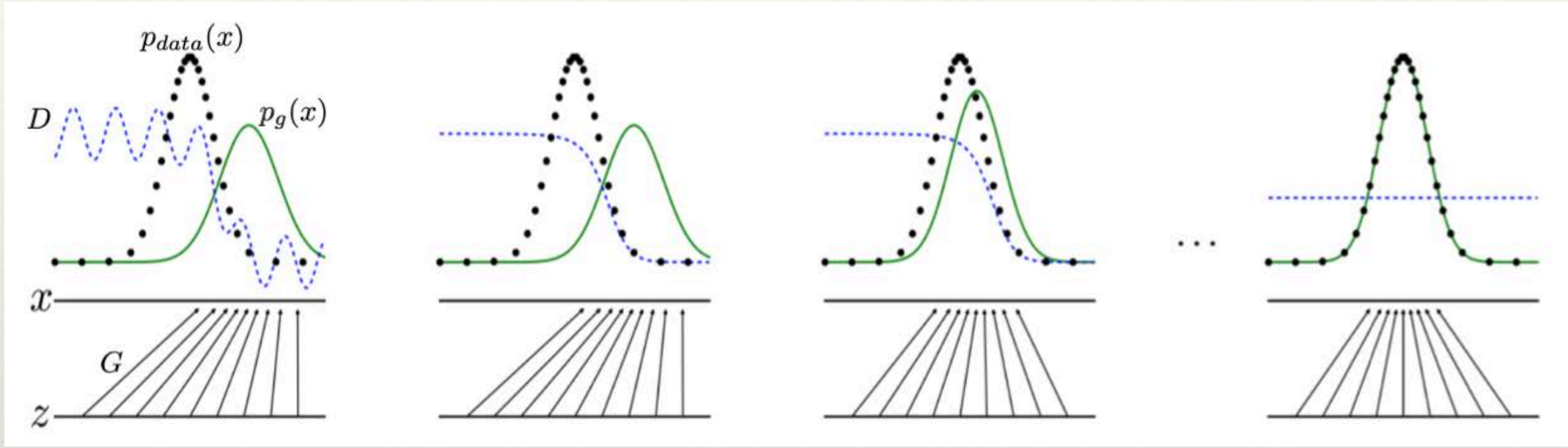
GAN 的损失函数构建

模型检查：

模型存在全局最优解

GAN 的损失函数构建

1. 符号的定义和直观理解



x: 真实数据，其概率分布为 $p_{data}(x)$

z: 噪声，其概率分布为 $p_z(z)$

$G(z;\theta_g)$: 根据 z 的分布将 z 映射到数据空间，得到的数据的概率分布为 $p_g(x)$

$D(s;\theta_d)$: 接收输入 s, 输出单个标量，表示样本 s 来源于真实数据而不是 G 生成的数据

最终目的: $p_{data}(x) = p_g(x) \quad \& \quad D(x) = 1/2$

GAN 的损失函数构建

2. 构建损失函数

(1) 考虑判别器 D

来自真实分布：$D(x) = 1 \quad when \quad x \sim p_{data}(x)$

$$E_{x \sim p_{data}(x)}\ log[D(x)] \Rightarrow max$$

来自 G 的分布: $D(x) = 0 \quad when \quad x \sim p_g(x)$

$$E_{z \sim p_z(z)}\ log[D(G(z))] \Rightarrow min \quad 等价于 \quad E_{z \sim p_z(z)}\ log[1 - D(G(z))] \Rightarrow max$$

结合前两者，当生成器 G **给定**时：

不妨构造损失函数：

$$V(G, D) = E_{x \sim p_{data}(x)}\ log[D(x)] + E_{z \sim p_z(z)}\ log[1 - D(G(z))]$$

D 的最优解：$D_G^* = \arg\max_D\ V(G, D)$

GAN 的损失函数构建

(2) 考虑生成器 G

生成器 G 的目的:

能以假乱真　　$p_g(x) = p_{data}(x)$

实现方式:

最大化 $D(G(z))$，即最小化 $E_{z\sim p_z(z)}\ log[1 - D(G(z))]$

构建损失函数:

$$V(G, D) = E_{x\sim p_{data}(x)}\ log[D(x)] + E_{z\sim p_z(z)}\ log[1 - D(G(z))]$$

当判别器 D **给定**时:

G 的最优解为: $G_D^* = \arg\min_G\ V(G, D)$

GAN 的损失函数构建

(3) 综合考虑 G 和 D:

对抗过程是个极大极小博弈问题(Minimax Game):

$$\min_G \max_D V(G,D) = \min_G \max_D E_{x\sim p_{data}(x)}\, log[D(x)] + E_{z\sim p_z(z)}\, log[1-D(G(z))]$$

对抗的优化过程:

固定 G，最大化 V(G, D), 更新 D 的参数 θ_d

固定 D，最小化 V(G, D), 更新 G 的参数 θ_g

通过交替迭代，期望最终达到“平衡”。也就是通过优化，使得:

对于生成器 G: $p_{data}(x) = p_g(x)$

对于判别器 D: $D(x) = 1/2$（真实数据和伪造数据完全分不出来）

存在的疑问

正向考虑

分别考虑 G 和 D 的优化目标，提出了损失函数：

$$V(G,D) = E_{x\sim p_{data}(x)}\ log[D(x)] + E_{z\sim p_z(z)}\ log[1 - D(G(z))]$$

方向考虑

训练过程是一个迭代过程，G 和 D 互相高度依赖

以该损失函数构建模型，对抗的博弈问题最终是否对 G 和 D 存在一个全局的最优解

否则，可能出现G 和 D 某一方在训练过程中陷入局部最优，导致另一方训练失败，从而整个模型崩塌?

GAN 的博弈过程存在全局最优解

1. 最优判别器 D 存在性的证明：

$$V(G,D) = E_{x\sim p_{data}(x)}\ log[D(x)] + E_{z\sim p_z(z)}\ log[1-D(G(z))]$$

将 V(G, D) 展开为积分形式：

$$V(G,D) = \int_x p_{data}(x)[logD(x)]\ dx + \int_z p_z(z)log[1-D(G(z))]\ dz$$

$$= \int_x (\ \underbrace{p_{data}(x)}_{\mathbf{a}}[log\underbrace{D(x)}_{\mathbf{y}}] + \underbrace{p_g(x)}_{\mathbf{b}}log[1-\underbrace{D(x)}_{\mathbf{y}}]\ )\ dx \qquad \text{（积分换元）}$$

不涉及 D(x) 的都看成常数项，令 y = D(x)

问题转化为：只考虑被积函数 $f(y) = a\log y + b\log(1-y)$

GAN 的博弈过程存在全局最优解

$$V(G,D)=\int_x p_{data}(x)[logD(x)]\ dx+\int_z p_z(z)log[1-D(G(z))]\ dz$$

$$=\int_x (\underbrace{p_{data}(x)}_{\mathbf{a}}[log\underbrace{D(x)}_{\mathbf{y}}]+\underbrace{p_g(x)}_{\mathbf{b}}log[1-\underbrace{D(x)}_{\mathbf{y}}])\ dx$$

考虑被积函数 $f(y)=a\log y+b\log(1-y)$

令 $f'(y)=0\Rightarrow y=\frac{a}{a+b}$

令 $f''(\frac{a}{a+b})=-\frac{a}{(\frac{a}{a+b})^2}-\frac{b}{1-\frac{a}{a+b}}<0$

$\Longrightarrow$ $y=\frac{a}{a+b}$ 时被积函数f(x)取最大值

最优判别器 D 是存在的，且唯一的。即: D 存在全局最优解

最优判别器为: $D(x)=\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$

GAN 的博弈过程存在全局最优解

2. 最优生成器 G 存在性的证明:

2.1 KL 散度和 JS 散度

香农信息量: $\log \frac{1}{p} = -\log p$

信息熵: 香农信息量的期望: $H(p) = -\sum p(x) \log p(x)$

H(x) 是分布为 p 的样本集的信息量，为最小信息量

交叉熵: 用 q 来估计真实分布为 p 的样本集的信息量: $H(p,q) = -\sum p(x) \log q(x)$

相对熵 (KL 散度): 上述估算多出来的冗余信息量

$$D(p||q) = H(p,q) - H(p) = \sum p(x) \log \frac{p(x)}{q(x)}$$

显然有: $D(p||q) \geq 0$ (当且仅当 p=q 时取等号)

GAN 的博弈过程存在全局最优解

KL 散度 (Kullback-Leibler Divergence):

$$KL(p(x)||q(x)) = \sum p(x) \log \frac{p(x)}{q(x)}$$

KL 散度有三个结论:
(1) 两个函数完全相同时，KL = 0
(2) KL 越大，两个函数的差异越大
(3) 对概率分布或者概率密度函数(>0), KL可用来衡量两个随机变量分布的差异性

KL散度是不对称的，因此琴生和香农提出 JS 散度 (Jenson-Shannon Divergence):

$$JS(p(x)||q(x)) = \frac{1}{2}KL(p(x)||\frac{p(x)+q(x)}{2}) + \frac{1}{2}KL(q(x)||\frac{p(x)+q(x)}{2})$$

JS 散度的取值范围为 0 ~ log 2

若两个分布完全没有交集，那么 JS 散度取最大值 log2；若两个分布完全一样，那么 JS 散度取最小值 0。

GAN 的博弈过程存在全局最优解

2.2 生成器 G 存在全局最优解：

G 的最终目标是: $p_g(x) = p_{data}(x)$

(最优判别器 $D(x) = \dfrac{p_{data}(x)}{p_{data}(x) + p_g(x)} = \dfrac{1}{2}$ ，意味着 D 已经完全分不出来了)

下面证明：当且仅当 $p_g(x) = p_{data}(x)$ 时，$V(G, D_G^*)$ 达到全局最小值

(1) $p_g(x) = p_{data}(x)$ 时:

$$V(G, D_G^*) = \int_x \left(p_{data}(x)[logD(x)] + p_g(x)log[1 - D(x)] \right) dx$$

$$= \int_x \left(p_{data}(x)log(\frac{1}{2}) + p_g(x)log(\frac{1}{2}) \right) dx$$

$$= -log2 \int_x p_{data}(x)\ dx - log2 \int_x p_g(x)\ dx$$

$$= -log4$$

下面只需证明: -log4就是最小值，且只在 $p_g(x) = p_{data}(x)$ 时取到

GAN 的博弈过程存在全局最优解

(2) 证明-log4 是全局最小值

当 D 最优时:

$$V(G, D_G^*) = \int_x \left(p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} + p_g(x) \log \frac{p_g(x)}{p_{data}(x) + p_g(x)} \right) dx$$

$$+ \int_x \left((\log 2 - \log 2) p_{data}(x) + \int_x (\log 2 - \log 2) p_g(x) \right) dx$$

$$= -\log 2 \underbrace{\int_x p_{data}(x) dx}_{\mathbf{1}} - \log 2 \underbrace{\int_x p_g(x) dx}_{\mathbf{1}} +$$

$$= \int_x \left(p_{data}(x) (\log 2 + \log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}) + p_g(x) (\log 2 + \log \frac{p_g(x)}{p_{data}(x) + p_g(x)}) \right) dx$$

$$\underbrace{\qquad\qquad}_{KL(p_{data}(x) || \frac{p_{data}(x) + p_g(x)}{2})} \quad \underbrace{\qquad\qquad}_{KL(p_g(x) || \frac{p_{data}(x) + p_g(x)}{2})}$$

$$= -\log 4 + 2JS(p_{data}(x) || p_g(x))$$

所以，可看出，当且仅当 $p_g(x) = p_{data}(x)$ 时 $V(G, D_G^*)$ 取得全局最优值 -log4

重述GAN 的训练过程

1. 理想情况下：

根据上述推导证明，GAN 的训练是一个 G 和 D 互相迭代优化的过程：

固定 G_0，最大化 V(G_0, D)，求得最佳 D_0*

固定 D_0*，最小化 V(G, D_0*)，求得最佳 G_1*

固定 G_1*，最大化 V(G_1*, D)，求得最佳 D_1*

固定 D_1*，最小化 V(G, D_1*)，求得最佳 G_2*

……

重述GAN的训练过程

2. 实际情况下：

数据分布是离散的

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

相较于理想情况下，每次循环时，依次交替完整地优化 D 和 G 在计算上是不被允许的

而且采用 minibatch 随机梯度下降时，每个 minibatch 充分训练容易导致过拟合

因此我们可以在 k 个优化 D 的步骤和一个优化 G 的步骤间交替进行。那么我们只需慢慢地更新 G，D 就会一直处于最优解的附近。

3. 基于 GAN 的人脸图像修复

文章来源:

《Semantic Image Inpainting with Deep Generative Models》（2017 CVPR)

程序实现:

基于 DCGAN (2016 ICLR) 改进实现

人脸图像修复

1. 准备数据集：

   实验使用的数据集是 LFW (Labeled Faces in the Wild Home) 数据集，该数据集由美国马萨诸塞大学阿姆斯特朗分校计算机视觉实验室整理而成。

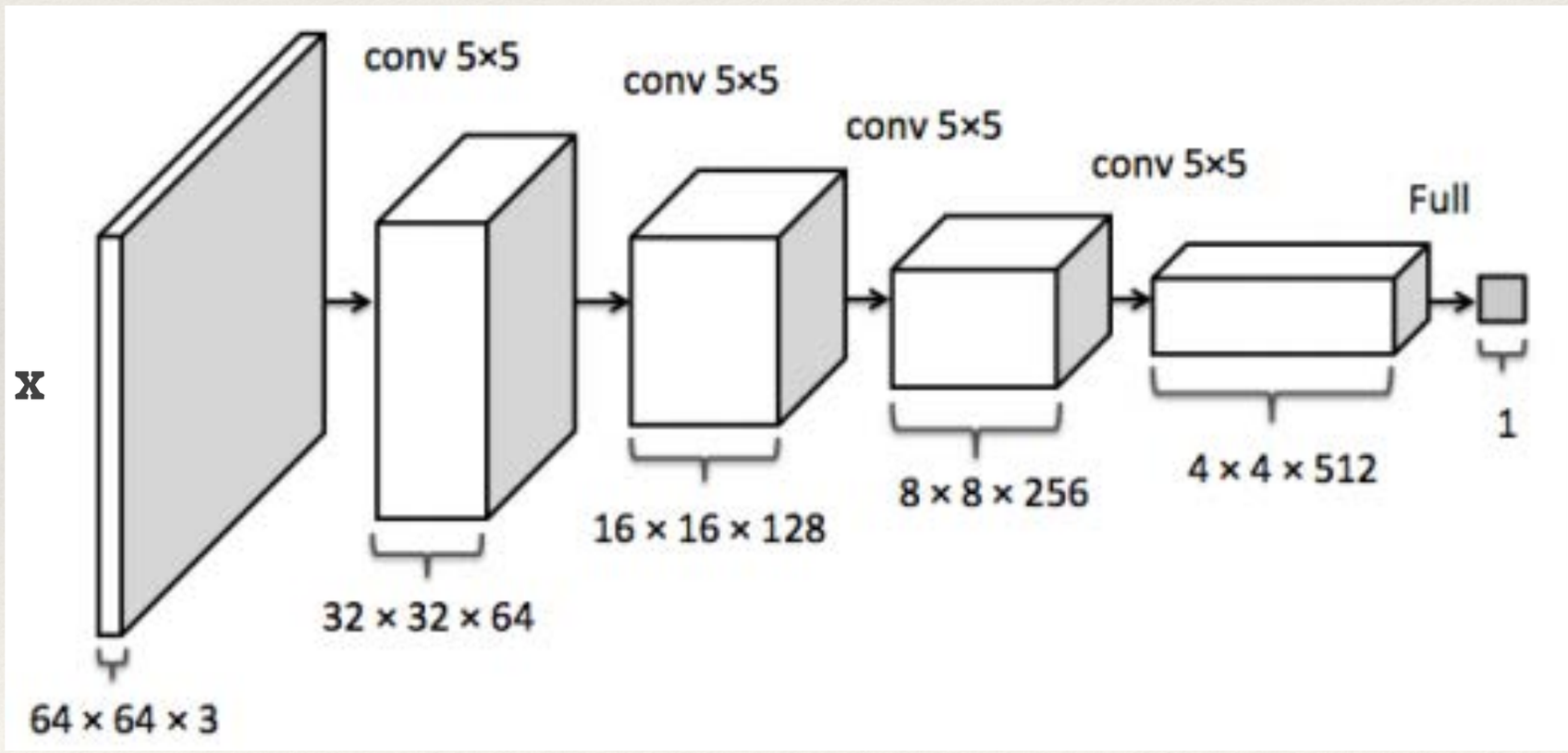
   LFW 一共包含13233张图片，共5749人，其中4096人只有一张图片，1680人的图片多于1张。每张照片的尺寸是250 x 250。

人脸图像修复

2. GAN 结构:

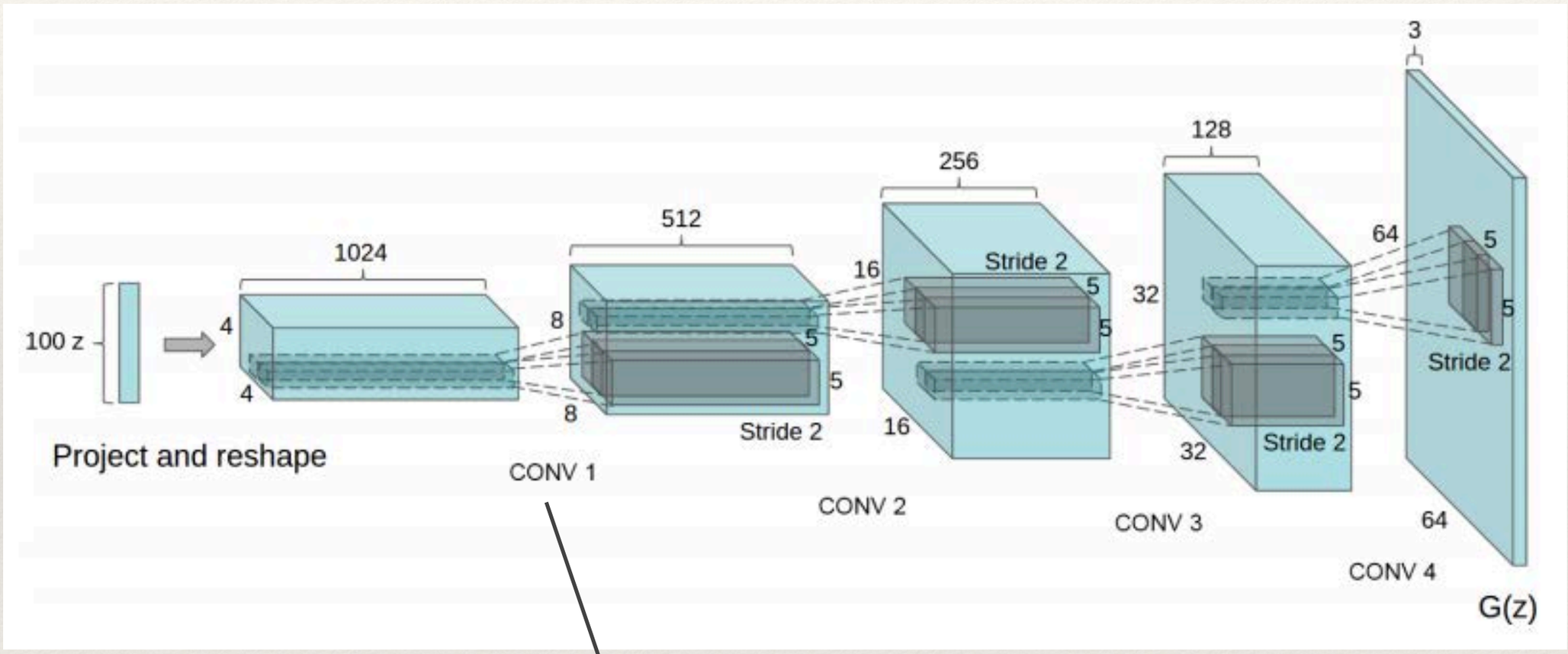
模型来自于《Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks》(ICLR 2016)

判别器 D:



人脸图像修复

生成器 G:

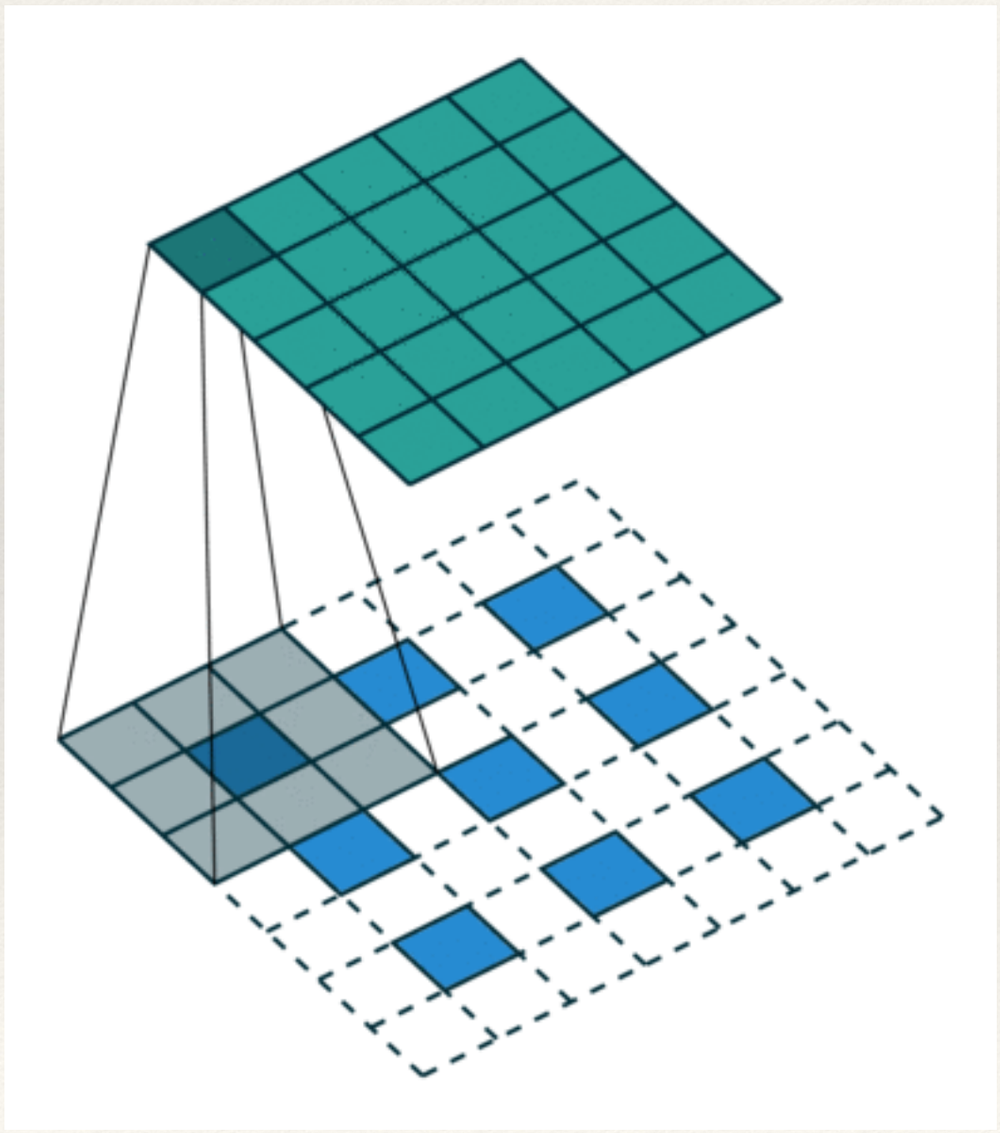


反卷积 (deconvolution)

人脸图像修复

反卷积(deconvolution):

DCGAN 原文使用微步幅卷积(fractionally-strided convolution):



微步幅卷积 = 上采样 + 卷积

改进:

更改为: transposed convolution

tensorflow:
tf.contrib.layers.conv2d_transpose

人脸图像修复

3. 生成 Fake face images:

$$\min_G \max_D V(G,D) = \min_G \max_D E_{x\sim p_{data}(x)}\, log[D(x)] + E_{z\sim p_z(z)}\, log[1-D(G(z))]$$

for number of training iterations **do**
  for k steps **do**
    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \Big[\underbrace{\log D\left(\boldsymbol{x}^{(i)}\right)}_{d_{loss}(real)} + \underbrace{\log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)}_{d_{loss}(fake)} \Big].$$

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

X: 真实人脸照片

Z: [-1, 1] 之间的随机噪声

m: batch size (实验取64)

k 取 2

损失函数: 交叉熵

D(s)部分：$d_{loss}(real)$: 来自 X 的样本标签打1　$d_{loss}(fake)$: 来自 G 的样本标签打0

g_{loss}: 来自 G 的样本标签打 1

激活函数: ReLU，优化器：Adam

人脸图像修复

Fake face images 生成效果：

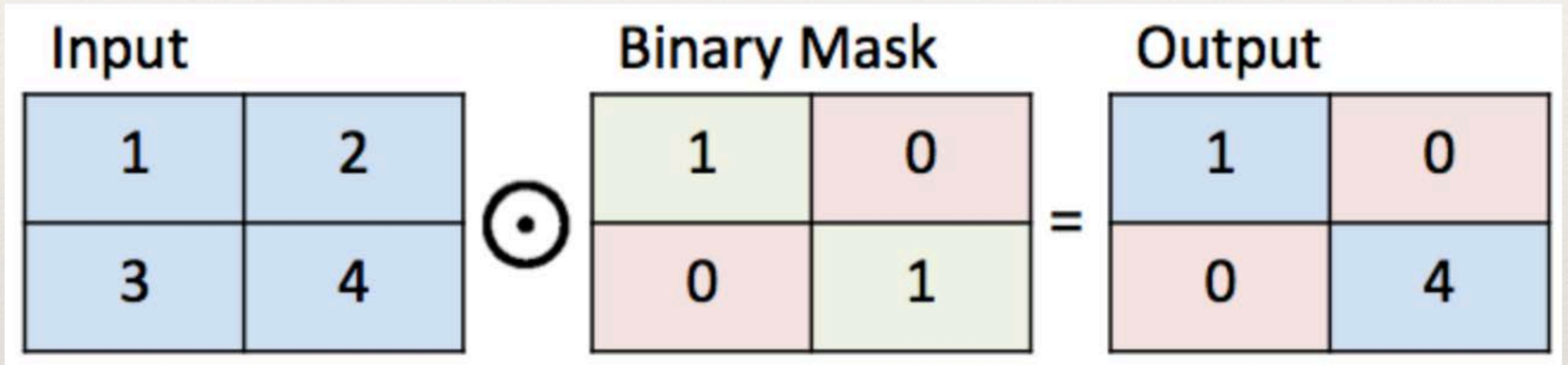
30 epochs

人脸图像修复

4. 人脸修复

思路来源:

《Semantic Image Inpainting with Deep Generative Models》 (2017 CVPR)



M: Binary Mask，元素值取0或者1。

1表示这部分原图像需要保留，0表示这部分缺失的图像需要补全

$$x_{resconstructed} = M \odot y + (1 - M) \odot G(z)$$

$M \odot y$: 原始图像中完整的部分，予以保留

$(1 - M) \odot G(z)$: 原始图像中缺失的部分，用 GAN 的 G 来生成

人脸图像修复

构建损失函数：

目的：利用 GAN 的生成器 G 部分来补全缺失的图像

要求：G 生成的图像符合原图语义关联 → 需要新的损失函数来约束 G

损失函数可分两部分：$loss = l_{contextual} + l_{perceptual}$

(1) contextual loss: $l_{contextual}(z) = ||M \odot G(z) - M \odot y||_1$

G生成的未缺失图像对应区域与原图要相似

(2) perceptual loss: $l_{perceptual}(z) = log(1 - D(G(z)))$

G生成的全图整体具有语义性

(3) total loss: $loss = l_{contextual}(z) + \lambda l_{perceptual}(z)$

λ: 两种 loss 的比重(实验中取了0.2)

人脸图像修复

人脸修复效果:

30 epochs

30

4. 深度学习中的权重初始化

为什么权重初始化很重要

深度学习框架是如何实现卷积操作的

如何进行有效的权重初始化

为什么权重初始化很重要

考虑一种最简单的情况：

所有层均为全连接层，激活函数均为 tanh（在0附近近似线性函数）

输入 X I.I.D. 且 $x \sim N(0, \sigma_x^2)$　　参数 W I.I.D. 且 $w \sim N(0, \sigma_w^2)$

考虑前向传播过程：

$$z_j = f(\sum_{j=1}^{n} w_i x_i) = \sum_{j=1}^{n} w_i x_i$$

显然：$z_j \sim N(0, n\sigma_x\sigma_w)$

前向传播，依次计算到最后一层：

$$z^1 \sim N(0, n^1\sigma_x^1\sigma_w^1)$$

$$z^2 \sim N(0, n^2\sigma_x^2\sigma_w^2)$$

$$\vdots$$

$$z^k \sim N(0, n^k\sigma_x^k\sigma_w^k)$$

$$\Rightarrow \quad z^k \sim N(0, \sigma_x^1 * \boxed{\prod_{i=1}^{k-1} n^i} * \sigma_w^i)$$　　方差累积

为什么权重初始化很重要

考虑反向传播过程：

$$\frac{\partial loss}{\partial x_j^{k-1}} = \sum_{i=1}^{\hat{n}} \frac{\partial loss}{\partial x_i^k} \frac{\partial x_i^k}{\partial x_j^{k-1}} = \sum_{i=1}^{\hat{n}} \frac{\partial loss}{\partial x_i^k} w_j^k$$

假设每一层的参数也服从某种均值为0，方差为某值的分布

容易得到：

$$Var(\frac{\partial loss}{\partial x_j^1}) = Var(\frac{\partial loss}{\partial x_j^k}) * \boxed{\prod_{i=1}^{k-1} \hat{n}^i} * \sigma_w^i$$

方差累积

综合考虑前向和反向：

$$\prod_{i=1}^{k-1} n^i = 1 \qquad \sigma_w^k = \frac{1}{n^k}$$

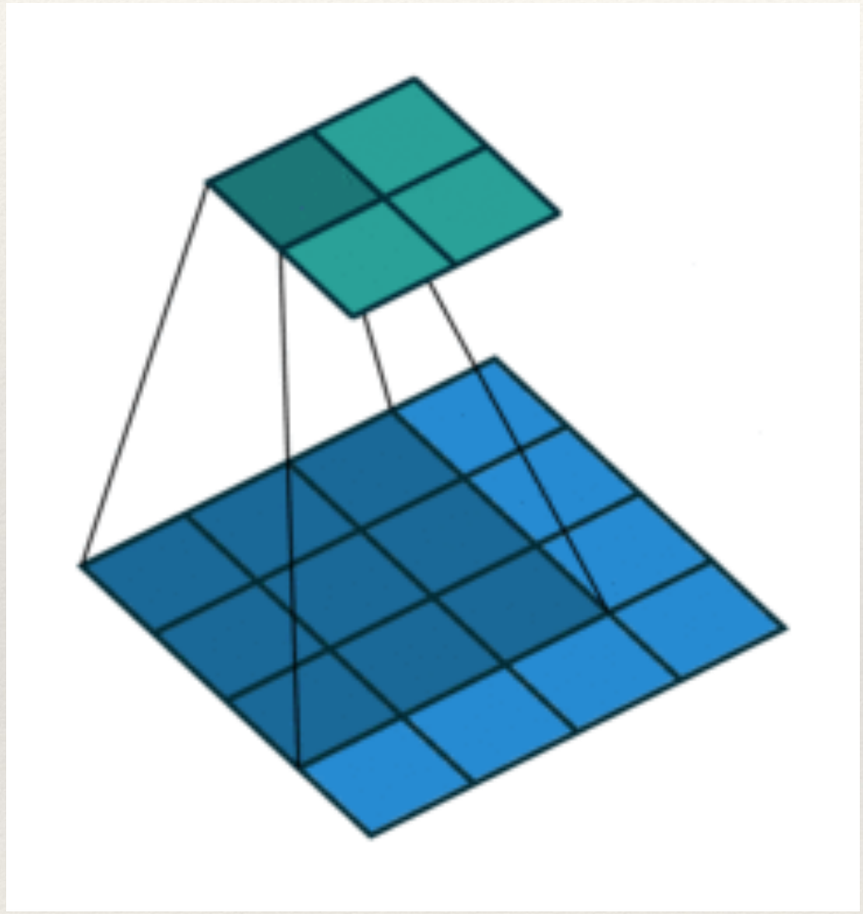
$$\prod_{i=1}^{k-1} \hat{n}^i = 1 \Rightarrow \sigma_w^k = \frac{1}{\hat{n}^k} = \frac{1}{n^{k+1}} \Rightarrow \sigma_w^k = \frac{2}{n^k + n^{k+1}} \Rightarrow [-\sqrt{\frac{6}{n^{k+1} + n^k}}, \sqrt{\frac{6}{n^{k+1} + n^k}}]$$

Xavier 初始化 (2010)

均匀分布

深度学习框架是如何实现卷积操作的

卷积的直观理解：

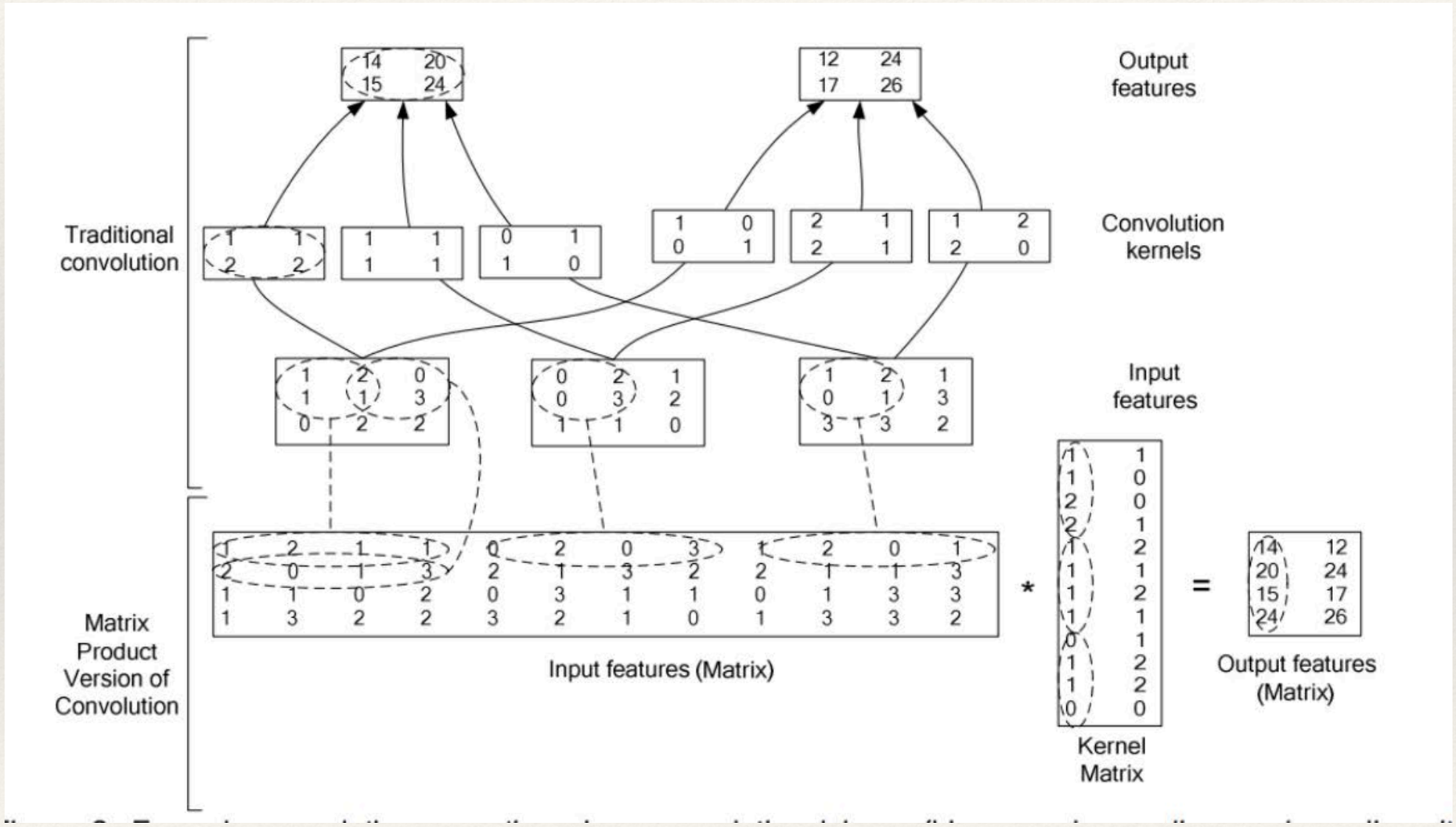


深度学习中的卷积核没有翻转

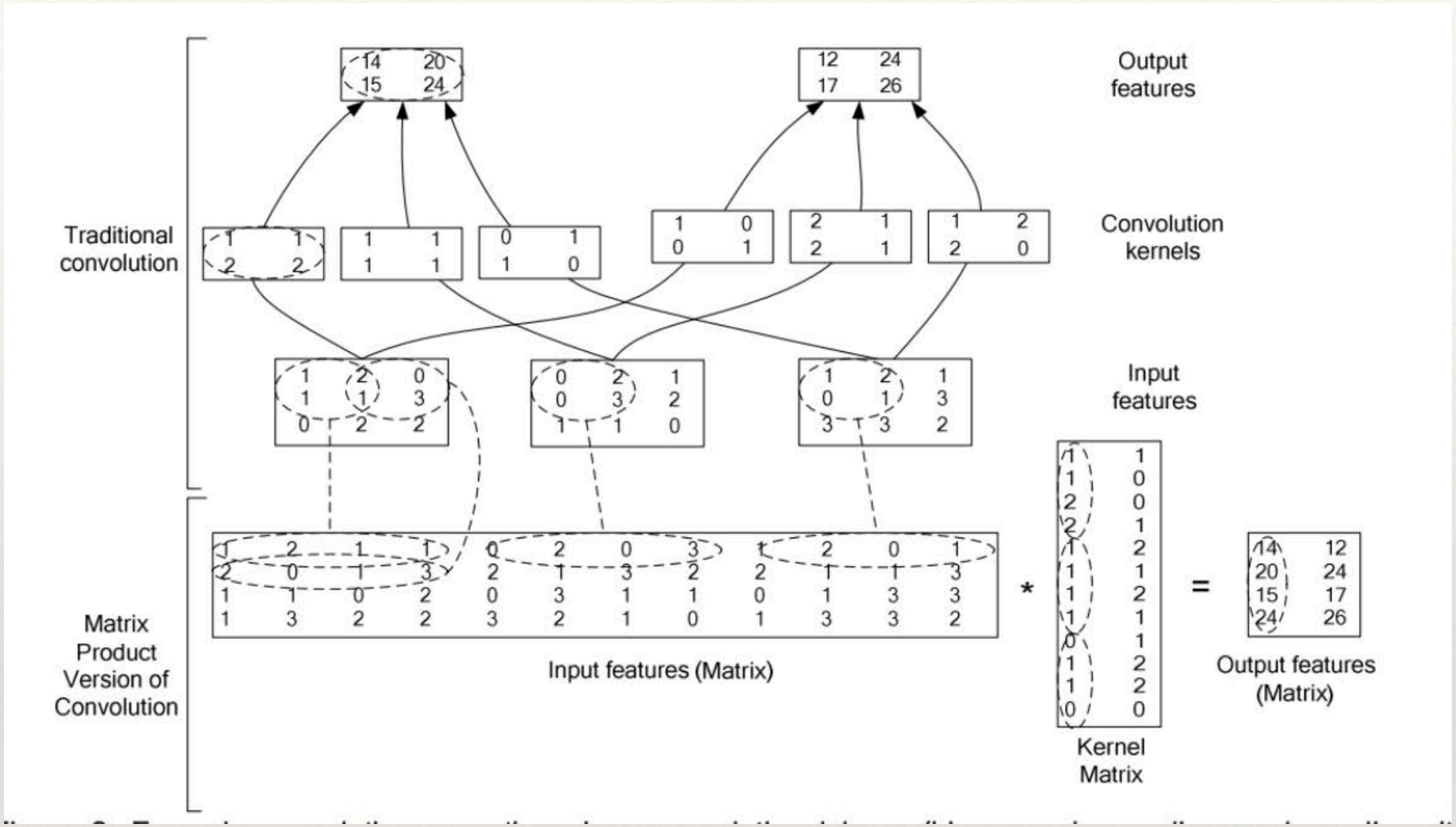
深度学习框架是如何实现卷积操作的

各大框架的实现：im2col 算法(2006)

卷积转换为矩阵相乘，方便 GPU 进行加速运算



深度学习框架是如何实现卷积操作的



对某一层:

设 Input Feature Map 有 c 个

卷积核的个数为 d, 卷积核大小为 k * k

$$Y = XW+b$$

X: $[N, k^2c]$

W: $[k^2c, d]$

Y: $[N, d] \rightarrow$ reshape…

如何进行有效的权重初始化

思路来源：

《Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification》(Kaiming He et.al 2015 ICCV)

针对卷积神经网络，使用 ReLU 函数作为激活函数的情况下，
提出了一套网络参数初始化的理论

使用 VGG16，VGG19，GoogLeNet 等主流网络在 Imagenet 上进行了完整测试，无论是模型准确率还是收敛速度均获得显著提高，并首次实现了 ImageNet 识别准确率超越人类，是深度学习在图像识别领域的一个里程碑。

如何进行有效的权重初始化

理论推导：

(1) 考虑前向传播：

$$Y = XW+b$$

X: $[N, k^2c]$

W: $[k^2c, d]$

Y: $[N, d]$

$\Rightarrow$

对(卷积)层 l

$$Y_l = W_l X_l + b_l$$

X: $[k^2c, 1]$

W: $[d, n]$ $(n = k^2c)$

Y: response，卷积后的输入，不经过激活函数

c: channel_input

d: channel_output

$c_l = d_{l-1}$

显然，有：$X_l = f(Y_{l-1})$ (f 为激活函数)

假设：W_l 满足 I.I.D，X_l 满足 I.I.D，且 W 和 X 互相独立

因此，$Var[y_l] = n_l Var[w_l x_l]$

$= n_l[Ew_l^2 Ex_l^2 - E^2 w_l E^2 x_l]$(这里再假设 w_l 均值为0)

$= n_l Var[w_l] Ex_l^2$

在 $E[w_{l-1}]$=0的情况下，我们再假设让 w_{l-1} 在0附近成对称分布，那么：$E[y_{l-1}]$=0 且 y_{l-1} 也在0附近对称分布。考虑到在 ReLU 激活函数下：$x_l = max(0, y_{l-1})$

则有：$E[x_l^2] = \frac{1}{2} Var[y_{l-1}]$

从而有：$Var[y_l] = \frac{1}{2} n_l Var[w_l] Var[y_{l-1}]$

如何进行有效的权重初始化

$Var[y_l] = \frac{1}{2} n_l Var[w_l] Var[y_{l-1}]$　　→多层累积：　$Var[y_L] = Var[y_1](\prod_{l=2}^{L} \frac{1}{2} n_l Var[w_l])$

好的初始化是不会让方差呈指数叠加的：

因此，可令：　$Var[w_l] = \frac{2}{n_l}\ (l \geq 2)$，同时 b 初始化为0

对第1层，没有 ReLU 的作用：

$$Var[y_1] = n_1 Var[w_1] E[x_1^2] = n_1 Var[w_1] Var[x_1] \ \Rightarrow Var[w_1] = \frac{1}{n_1} \quad (\text{假设 } E[x_1] = 0\ )$$

第一层参数差了1/2，影响不大，因此统一为：

正向传播时：

$$Var[w_l] = \frac{2}{n_l}\ (l \geq 1)$$

如何进行有效的权重初始化

(2) 考虑反向传播：

因为 $Y_l = W_l X_l + b_l$

由矩阵求导: $\frac{\partial l}{\partial X_l} = \hat{W}_l \frac{\partial l}{\partial Y_l}$

表示为： $\Delta X_l = \hat{W}_l \Delta Y_l$

$\hat{W}_l$：W_l 转置后重新 reshape

$[c, \hat{n}]$ $(\hat{n} = k^2 d)$

ΔY_l: $[k^2 d, 1]$

ΔX_l: $[c, 1]$

同样假设：$\hat{W}_l$，ΔY_l 相互独立，w_l 均值为0且对称分布

前向推导时，有 y_l 在0值对称分布

所以，在 ReLU 激活函数下，$f'(y_l)$ 等概率为0或者1

在正向时，$x_{l+1} = f(y_l)$

$$\Delta y_l = \frac{\partial loss}{\partial x_{l+1}} \frac{\partial x_{l+1}}{\partial y_l} = \Delta x_{l+1} f'(y_l)$$

则有： $E(\Delta y_l) = \frac{1}{2} E[\Delta x_{l+1}] = 0$

$$Var[\Delta y_l] = E[(\Delta y_l)^2] = E[(\Delta x_{l+1})^2 (f'(y_l))^2] = \frac{1}{2} E[(\Delta x_{l+1})^2] = \frac{1}{2} Var[\Delta x_{l+1}]$$

如何进行有效的权重初始化

所以：

$$\begin{aligned} Var[\Delta x_l] &= \hat{n}_l Var[w_l] Var[\Delta y_l] \\ &= \frac{1}{2} \hat{n}_l Var[w_l] Var[\Delta x_{l+1}] \end{aligned}$$

多层累计起来：

$$Var[\Delta x_2] = Var[\Delta x_{L+1}](\prod_{l=2}^{L} \frac{1}{2} \hat{n}_l Var[w_l])$$

$$\Rightarrow \frac{1}{2} \hat{n}_l Var[w_l] = 1 \Rightarrow Var[w_l] = \frac{2}{\hat{n}_l} \ (l \geq 1)$$

如何进行有效的权重初始化

综合考虑前向和反向传播：

前向传播要求：
$$Var[w_l] = \frac{2}{n_l} \quad (1)$$

反向传播要求：
$$Var[w_l] = \frac{2}{\hat{n}_l} \quad (2)$$

无论是满足条件(1)还是条件(2)均可以。Caffe 作者认为前向传播更重要，因此 Caffe 中默认为满足(1)式的正态分布。

例如，当满足反向条件(2)时：

那么前向时，
$$\prod_{l=2}^{L} n_l Var[w_l] = \prod_{l=2}^{L} \frac{n_l}{\hat{n}_l} = \prod_{l=2}^{L} \frac{d_{l-1}}{d_l} = \frac{d_1}{d_L}$$

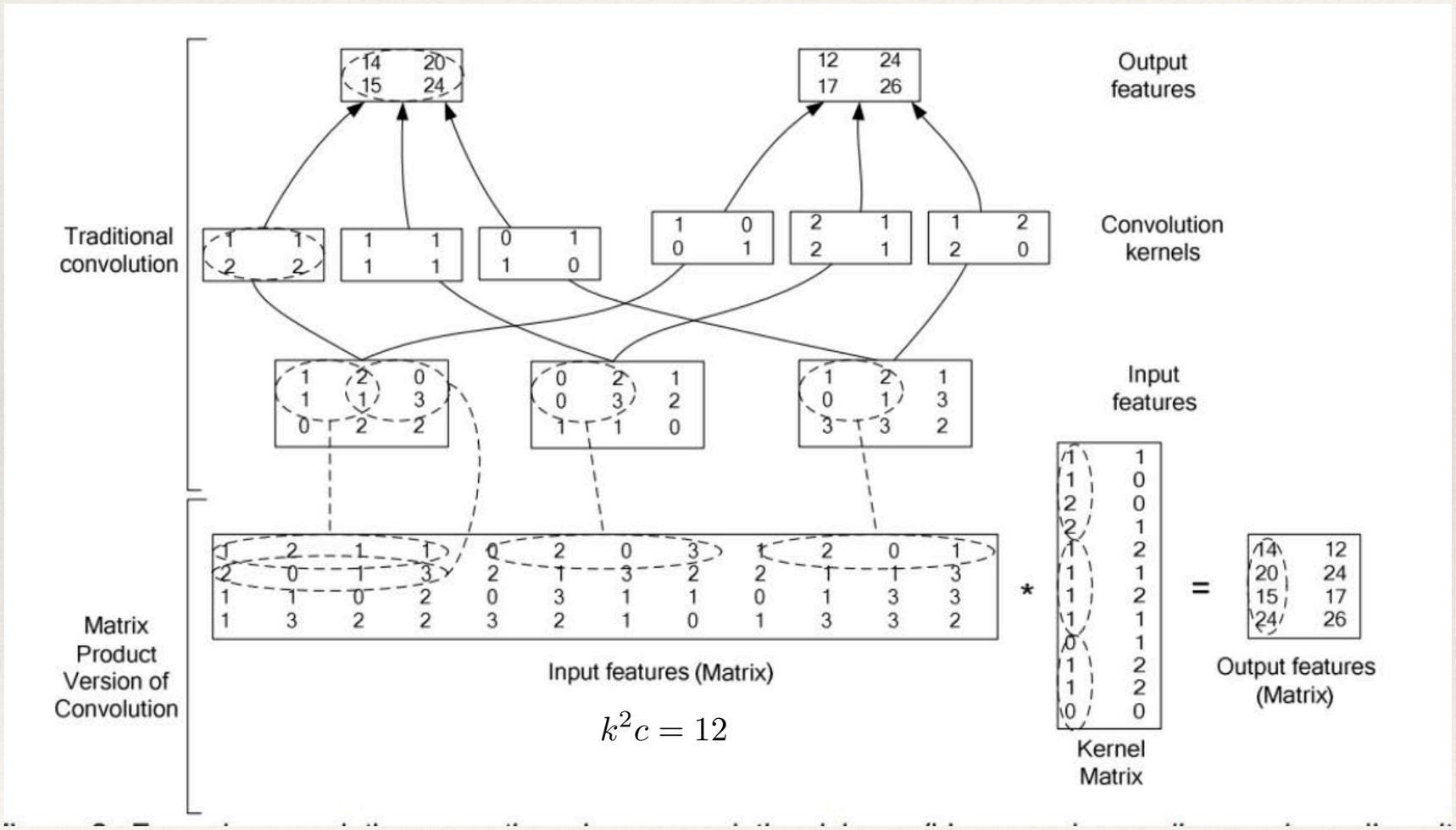
只相差一个常数项，满足反向条件的时候对前向影响不大

如何进行有效的权重初始化

前面推导时，有一项很重要的假设：

输入 X 均值为0

X: $[k^2c, 1]$



当各通道对应位置处均值为0时，上述E(x)=0 被满足

CS231n推荐：
numpy 取均值时，对数据第0轴取均值：

mean=np.mean(data, axis=0)

请大家批评指教