

# 数据挖掘与应用

## 深度异常检测

授课教师：周晟

浙江大学 软件学院

2021.10



# 教学内容

- ① 深度异常检测
- ② 深度特征提取
- ③ 基于特征提取的异常性判断
- ④ 面向异常检测的特征学习
- ⑤ 端到端异常检测模型



# 深度异常检测的主要挑战

- ① 低召回率
- ② 高维数据异常检测
- ③ 条件异常
- ④ 数据效率
- ⑤ 噪声与异常难以区分
- ⑥ 复杂类型的异常
- ⑦ 异常检测可解释



# 深度异常检测算法

深度异常检测算法按照模型结构大致可以分为四大类：

- ① 深度特征提取（特征提取与异常检测相互独立）
- ② 基于特征提取的异常性判断
- ③ 面向异常检测的特征学习
- ④ 端到端的异常检测模型



① 深度异常检测

② 深度特征提取

③ 基于特征提取的异常性判断

④ 面向异常检测的特征学习

⑤ 端到端异常检测模型



# 基于深度学习的特征提取

## 模型假设

相比于传统降维方法，深度学习可以学习样本间更有区分度的表征，从而提升异常检测的效果。

常见做法：

- ① 无监督特征提取模型
- ② 大规模预训练特征提取模型
- ③ 多个无监督预训练模型的集成

应用场景

- ① 图像特征提取
- ② 视频特征提取
- ③ 图特征提取



# 基于深度学习的特征提取

## 优点

- ① 有大量现成的深度学习算法可以用于有效提取特征
- ② 可以学习出有效的低维向量，提升异常检测的效率
- ③ 实现简单

## 缺点

- ① Two-stage, 特征提取与异常检测分离，不能保证能对异常检测有效
- ② 预训练模型只能应用于特定场景的数据



- ① 深度异常检测
- ② 深度特征提取
- ③ 基于特征提取的异常性判断
- ④ 面向异常检测的特征学习
- ⑤ 端到端异常检测模型



# 基于特征提取的异常性判断

通用框架：

$$\begin{aligned}\{\Theta^*, \mathbf{W}^*\} &= \arg \min_{\Theta, \mathbf{W}} \sum_{\mathbf{x} \in X} \ell(\psi(\phi(\mathbf{x}; \Theta); \mathbf{W})) \\ s_{\mathbf{x}} &= f(\mathbf{x}, \phi_{\Theta^*}, \psi_{\mathbf{W}^*})\end{aligned}$$

常见模型：

- ① 自编码器 Auto-encoder
- ② 生成模型 Generative Modeling
- ③ 预测模型 Predictability Modeling
- ④ 自监督分类 Self-supervised Classification



# 基于自编码器的异常检测

## 基本假设

自编码器的目标是最小化所有数据的重构损失，按照少数服从多数的原则，正常样本可以被更好地重构而异常样本则难以被完美重构。

## 模型结构

$$\begin{aligned} \mathbf{z} &= \phi_e(\mathbf{x}; \Theta_e), \hat{\mathbf{x}} = \phi_d(\mathbf{z}; \Theta_d) \\ \{\Theta_e^*, \Theta_d^*\} &= \arg \min_{\Theta_e, \Theta_d} \sum_{\mathbf{x} \in X} \|\mathbf{x} - \phi_d(\phi_e(\mathbf{x}; \Theta_e); \Theta_d)\|^2 \\ s_{\mathbf{x}} &= \|\mathbf{x} - \phi_d(\phi_e(\mathbf{x}; \Theta_e^*); \Theta_d^*)\|^2 \end{aligned}$$



# 基于自编码器的异常检测

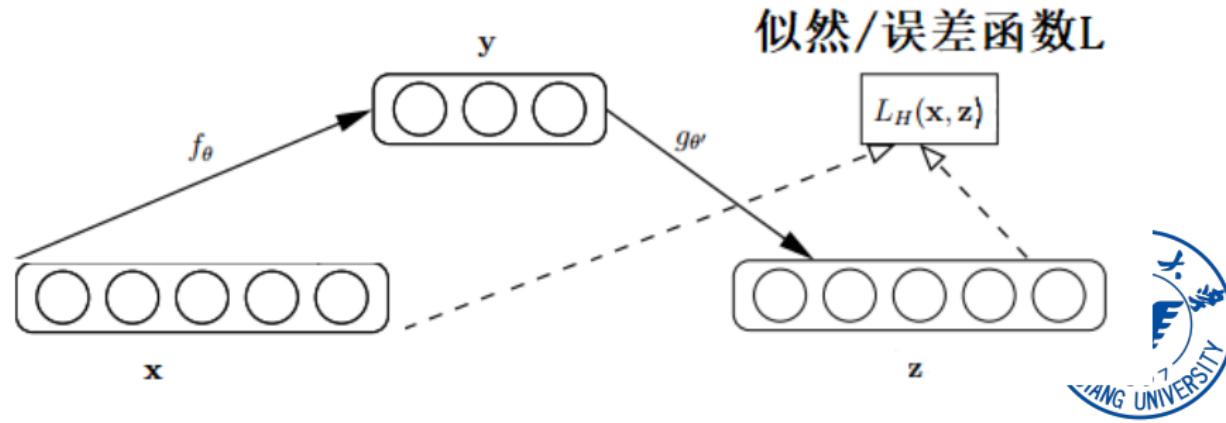
常用的 AutoEncoder 结构：

- ① Denoising AutoEncoder
- ② Sparse AutoEncoder
- ③ Contractive AutoEncoder
- ④ Variational AutoEncoder
- ⑤ Robust AutoEncoder



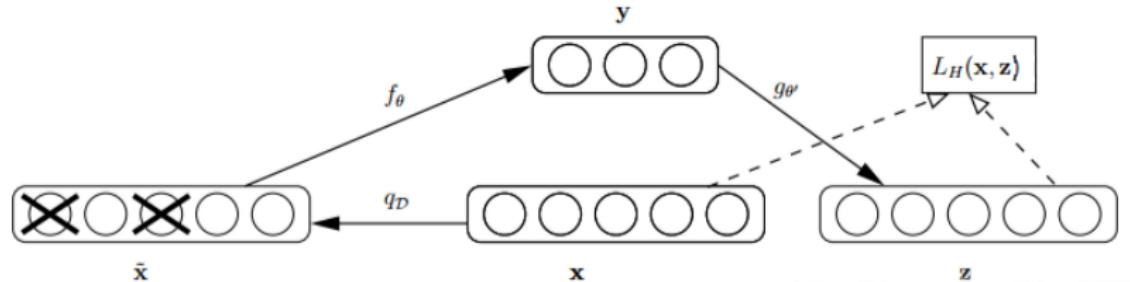
# Denoising AE

- 为了防止过拟合问题，降噪自动编码器（Denoising AE, DAE）在输入层的数据中人为地加入 noise，使学习到的模型具有更好的鲁棒性。
- 最基础的 AE 模型表示如下：



# Denoising AE

- DAE 的模型示意图如下



- $x$  为原始数据，DAE 以一定的概率（通常使用二项分布）把输入层的节点置为 0，以此来模拟 noise，从而获得含 noise 的输入数据  $\bar{x}$ 。
- 注意：最终计算的是重构  $z = g(f(x))$  和  $x$  之间的 Loss，而不是含有 noise 的  $\bar{x}$ 。



# Denoising AE

- 为什么用随机置 0 来模拟 noise ?
  - 随机置 0 时有可能会把数据中本来就自带的 noise 给擦除了。
  - 随机擦除后的数据在一定程度上减小了训练数据与测试数据之间的 gap，使训练得到的模型更适用于测试集。



## Denoising AE

- 传统 AE 的目标函数可以表示为

$$\mathcal{J}_{AE}(\theta) = \sum_{x \in D_n} L(x, g(f(x)))$$

其中  $f$  和  $g$  分别为 encoder 和 decoder,  $L$  为重构 Loss, 常用 MSE 或 cross-entropy。

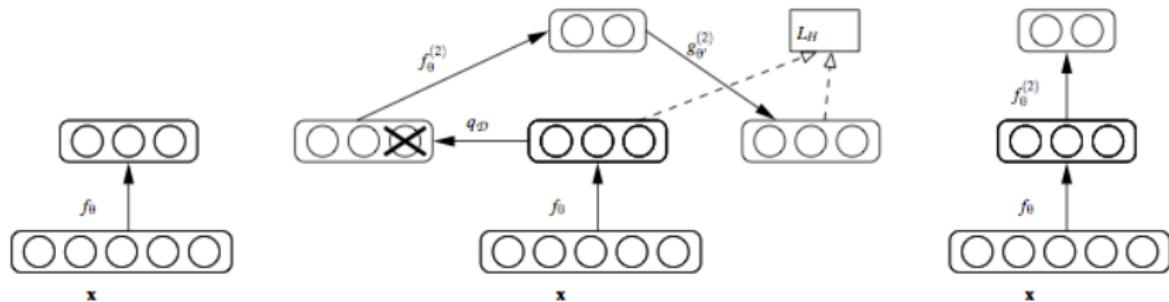
- DAE 的目标函数只是在此基础上对输入进行了随机擦除  $q$

$$\mathcal{J}_{DAE}(\theta) = \sum_{x \in D_n} L(x, g(f(q(x))))$$



# Denoising AE

- 多个 DAE 堆叠在一起，就构成了堆叠降噪自动编码器（Stacked Denoising AutoEncoder, SDAE）。
- 训练时逐层进行，且只有训练当前层时才需要对输入进行腐蚀（加噪），训练完成后不需要。下图是 SDAE 训练第二层时的示意。



# Sparse AE

- 除了显式地限制隐含层的维度，我们还可以对网络施加其他限制条件来学习有用的表征。
- 稀疏自动编码器（Sparse AutoEncoder, SAE）对隐含层施加了稀疏性约束，希望学习样本表征时可以学到稀疏的特征。
- 稀疏性约束：使神经网络中大部分的神经元的状态为抑制。
  - 激活：神经元的输出值接近 1 的状态
  - 抑制：神经元的输出值接近 0 的状态



# Sparse AE

- 那么，如何对 AE 进行修改，使其满足我们需要的约束呢？
- 一种常见的优化形式是在 AE 的目标函数中加入正则化项。最简单的形式是权重衰减 (weight-decay)，优化目标如下

$$\mathcal{J}_{AE+wd}(\theta) = \left( \sum_{x \in D_n} L(x, g(f(x))) \right) + \lambda \sum_{ij} W_{ij}^2$$

其中  $W$  表示 AE 中的网络权重， $\lambda$  为控制正则化强度的参数。



# Sparse AE

- 定义稀疏性参数  $p$  (一般取接近 0 的数, 比如 0.05), 计算隐含层各个单元的稀疏性  $\hat{p}_j$

$$\hat{p}_j = \frac{1}{n} \sum_{i=1}^n [a_j(x_i)]$$

其中  $a_j$  为第  $j$  的隐含单元的激活值。

- 我们希望  $\hat{p}_j = p$ , 使用 KL 散度来进行约束:

$$\min \sum_{j=1}^m KL(p||\hat{p}_j)$$

其中  $m$  为隐含层的单元个数。



# Sparse AE

- 将目标函数中的正则化项替换为稀疏性约束的 KL 散度，可以获得 SAE 的目标函数

$$\mathcal{J}_{SAE}(\theta) = \left( \sum_{x \in D_n} L(x, g(f(x))) \right) + \lambda \sum_{j=1}^m KL(p || \hat{p}_j)$$

其中  $\lambda$  为控制约束强度的参数。



# Contractive AE

- 收缩自动编码器 (Contractive AutoEncoder, CAE) 是 Bengio 等人在 2011 年提出的一种新的 AutoEncoder, 在传统的 AutoEncoder 的重构误差上加上了新的惩罚项。
- CAE 将正则项换成了 F 范数下的雅克比矩阵的形式, 具体公式如下:

$$\mathcal{J}_{CAE}(\theta) = \sum_{x \in D_n} (L(x, g(f(x))) + \lambda \|J_f(x)\|_F^2)$$

同样,  $\lambda$  控制正则化的强度。



# Contractive AE

- 雅克比矩阵表示一阶偏导。假设有  $(x_1, x_2, \dots, x_n)$  到  $(y_1, y_2, \dots, y_m)$  的映射，相当于  $m$  个  $n$  元函数，其 Jacobian Matrix 如下：

$$J = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix}_{m \times n}$$

该矩阵可以表示  $x$  的微小波动对  $y$  的影响。



# Contractive AE

- 假设 CAE 隐含层的表征为  $h$ , 则 F 范式下的雅克比矩阵表示为

$$\|J_f(x)\|_F^2 = \sum_{ij} \left( \frac{\partial h_j(x)}{\partial x_i} \right)^2$$

- 我们再来看 CAE 的目标函数

$$\mathcal{J}_{CAE}(\theta) = \sum_{x \in D_n} (L(x, g(f(x))) + \lambda \|J_f(x)\|_F^2)$$

第一部分即 AE 的最小化重构 Loss, 而第二部分是使偏导尽可能小, 从而提升模型对输入数据的鲁棒性:

此时如果在输入数据上加一些 noise, 隐含层的值基本不变



# Contractive AE

- Contractive AE 与 Sparse AE 的关系：

SAE 的希望每个样本的大部分隐含层特征为 0，对于激活函数 Sigmoid 图像的左半部分，相应的导数很小。在 Contractive AE 中，也希望雅克比矩阵的对应部分值很小，因此 CAE 和 SAE 的效果很相似。

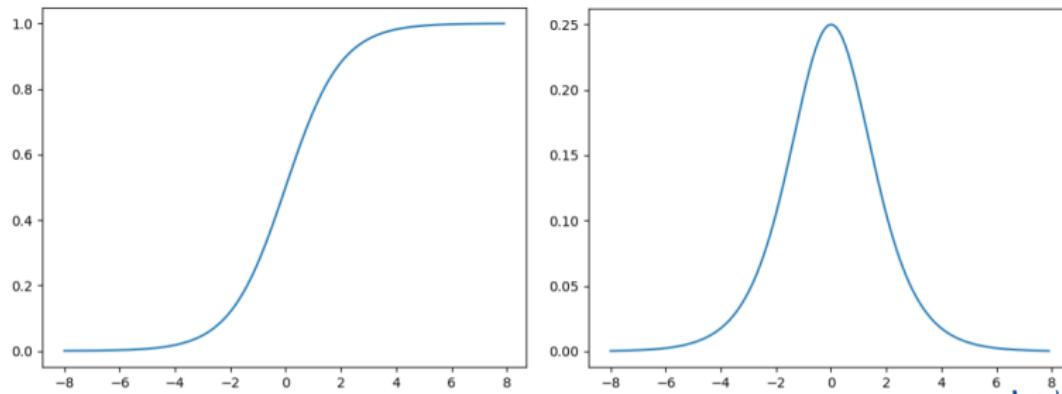
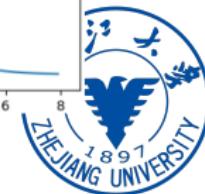


图: Sigmoid 及其导数图像



# Contractive AE

- Contractive AE 与 Denoising AE 的关系：

- 两者都能提升模型的鲁棒性，减少 noise 带来的影响。
- DAE 关注的是整个模型  $g(f(x))$  的鲁棒性，对于编码器  $f(x)$  只能间接影响；而 CAE 关注的则是编码器  $f(x)$  的鲁棒性。
- 事实上，我们一般都是只提取编码器  $f(x)$  的结果作为特征进行下游任务，因此特征的鲁棒性比数据重构的鲁棒性更重要。



# Variational AE

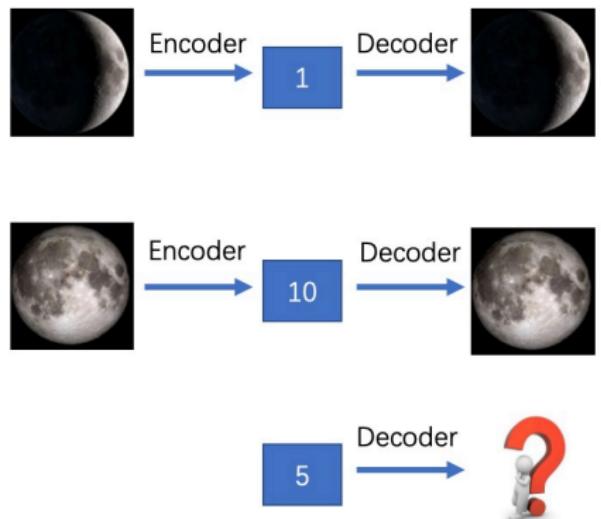
- 上述的几种 AE 的变种虽然都在 AE 的基础上进行了一些优化，但其本质缺点还是存在：从输入到输出的整个过程，都是基于已有的训练数据的映射。
- 换句话说，对于一个训练好的 AE，输入某个图片，就只会将其编码为某个确定的 code，输入某个确定的 code 就只会输出某个确定的图片。如果 code 是随机生成的，那么解码后的图片也有可能是乱七八糟的。



# Variational AE

- 假设我们现在有一个 AE，可以将“新月”图片编码为 code=1（假设 code 为 1 维），解码后能得到“新月”图片；将“满月”图片编码为 code=10，解码后能得到“满月”图片。

- 此时，我们手上有一个 code=5，希望将其解码后获得“半月”图片。
- 然而，之前训练集中并没有 code=5 的“半月”图片，因此我们不太可能得到“半月”的图片。



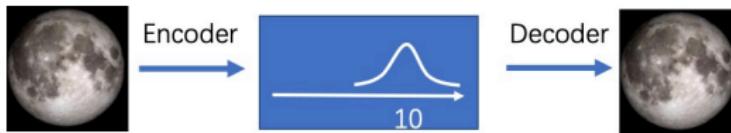
# Variational AE

- 如何解决这个问题呢？我们不妨转变思路，将图片映射由“数值编码”变为“分布”。
- 具体而言，将“新月”图片映射成  $\mu = 1$  的正态分布。这相当于在 1 附近加了噪声，此时 1 附近的数值也表示“新月”，只是 code=1 时最像“新月”。



## Variational AE

- 将“满月”映射为  $\mu = 10$  的正态分布，10 的附近也表示“满月”。
  - 由此，当 `code=5` 时，同时拥有了“新月”和“满月”的特征，解码出来的概率就是“半月”了。



# Variational AE

- 变分自动编码器 (Variational AutoEncoder, VAE) 结构如下所示。
- VAE 的 Encoder 输出若干个正态分布的均值 ( $\mu_1, \mu_2, \dots, \mu_m$ ) 和标准差 ( $\sigma_1, \sigma_2, \dots, \sigma_m$ )，然后从每个正态分布  $\mathcal{N}(\mu_1, \sigma_1^2), \mathcal{N}(\mu_2, \sigma_2^2), \dots, \mathcal{N}(\mu_m, \sigma_m^2)$  采样得到  $\text{code}(Z_1, Z_2, \dots, Z_m)$ ，再将 code 送入 Decoder 中解码。

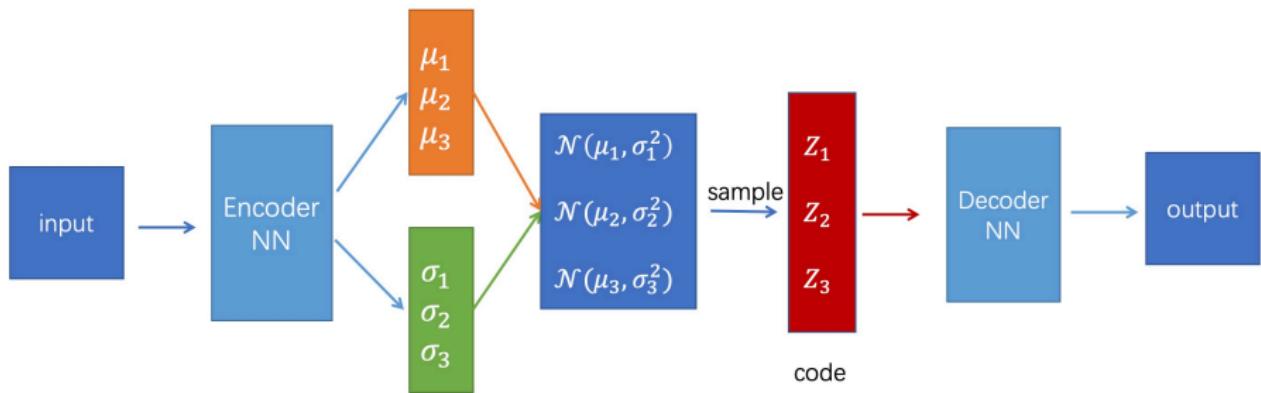


图: VAE 整体结构

# Variational AE

- VAE 的目标函数可以表示为：

$$\mathcal{J}_{VAE}(\theta) = \left( \sum_{x \in D_n} L(x, g(Z)) \right) + \lambda \text{KL}(\mathcal{N}(\mu, \sigma^2) \| \mathcal{N}(0, 1))$$

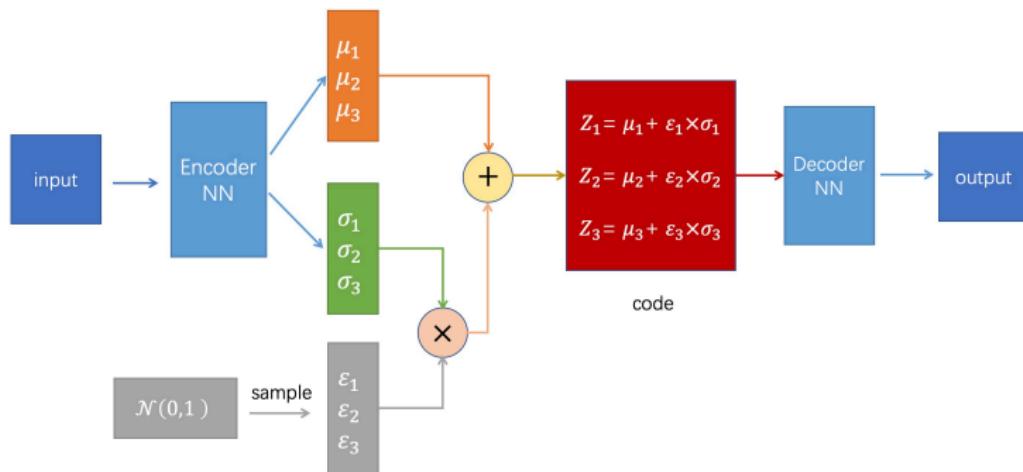
其中  $Z$  为从正态分布中采样得到的 code。

- 然而，这个采样操作是不可导的，会导致反向传播时  $Z$  对  $\mu$  和  $\sigma$  无法直接求导。



## Variational AE

- 解决方法：重参数化技巧（reparametrize）。具体思想为：从  $\mathcal{N}(0, 1)$  中采样一个  $\epsilon$ ，使  $Z = \mu + \epsilon \times \sigma$ ，这相当于直接从  $\mathcal{N}(\mu, \sigma^2)$  中采样  $Z$ 。



# Variational AE

- 由此，VAE 的目标函数可以重写为

$$\mathcal{J}_{VAE}(\theta) = \left( \sum_{x \in D_n} L(x, g(\mu + \epsilon \times \sigma)) \right) + \lambda \text{KL}(\mathcal{N}(\mu, \sigma^2) \parallel \mathcal{N}(0, 1))$$

- 第二项 KL 散度如何解释？



# Variational AE

$$\text{KL}(\mathcal{N}(\mu, \sigma^2) \| \mathcal{N}(0, 1))$$

- 训练过程中，如果仅仅使输入和输出的误差尽可能小，那么随着不断训练， $\mu$  会趋近于 0，这样就使得 VAE 越来越像 AE，对数据产生过拟合，编码的噪声也会消失，导致无法生成未见过的数据。
- 为了解决这个问题，我们需要对  $\mu$  和  $\sigma$  加以约束，使其构成的正态分布  $\mathcal{N}(\mu, \sigma^2)$  尽可能像标准正态分布  $\mathcal{N}(0, 1)$ ，具体通过 KL 散度来约束。



# Variational AE

- 以上 VAE 的介绍都是从便于理解的角度出发的，事实上，VAE 还有一套复杂的数学推导和证明，有兴趣的同学可以自学。
- VAE 也存在一定的缺点：它生成的数据不一定那么“真”。如果要使生成的数据比较“真”，则需要用到 GAN。



# Outlier Detection with Robust Deep AutoEncoders(KDD 2017)

## 研究动机

- ① 自编码器在训练过程中容易受到异常样本的影响
- ② Robust Principal Component Analysis (RPCA) 也是一种降维方法，但是为异常样本做了专门的优化

首次提出 Robust Deep AutoEncoder，用于克服异常数据对自编码器的影响



# RPCA

RPCA 将数据矩阵  $X$  拆分为低秩的矩阵  $L$  和一个稀疏矩阵  $S$ :

$$X = L + S$$

矩阵分解的过程可以理解为如下的优化目标:

$$\begin{aligned} & \min_{L,S} \|L\|_* + \lambda \|S\|_1 \\ \text{s.t. } & \|X - L - S\|_F^2 = 0 \end{aligned}$$

使用交替方向乘子法 Alternating Direction Method of Multipliers(ADMM) 方法进行优化.

Tips: ADMM 是一种将原问题的目标函数等价的分解成若干个可求解的子问题, 然后并行求解每一个子问题的方法, 可以应用于大规模分布式系统。



# Robust Deep AutoEncoders(RDA)

$$X = L_D + S$$

$L_D$  是指能被 AutoEncoder 重构的特征， $S$  包含了难以被 AutoEncoder 重构的噪声和异常。

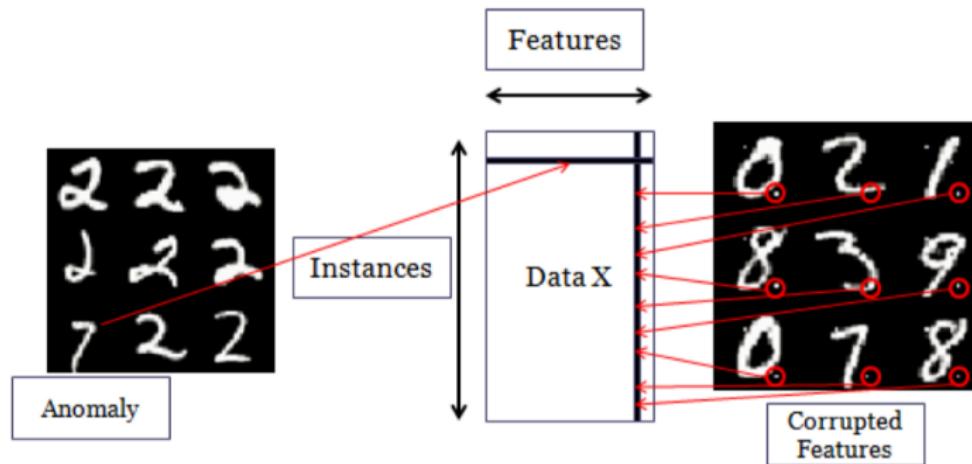
$$\begin{aligned} \min_{\theta} & \|L_D - D_{\theta}(E_{\theta}(L_D))\|_2 + \lambda \|S\|_1 \\ \text{s.t. } & X = L_D + S = 0 \end{aligned}$$



# Robust Deep AutoEncoders(RDA)

## Group Anomalies

- ① 许多样本共享一个相同的特征维度（系统噪声而不是异常）
- ② 一个样本中异常的特征应当相对确定



## Robust Deep AutoEncoders(RDA)

$\mathcal{L}_{2,1}$  norm

$\mathcal{L}_2$  norm regularizer 作用于每一种异常类型,  $\mathcal{L}_1$  作用于所有的异常类型

$$\|X\|_{2,1} = \sum_{j=1}^n \|x_j\|_2 = \sum_{j=1}^n \left( \sum_{i=1}^m |x_{ij}|^2 \right)^{1/2}$$

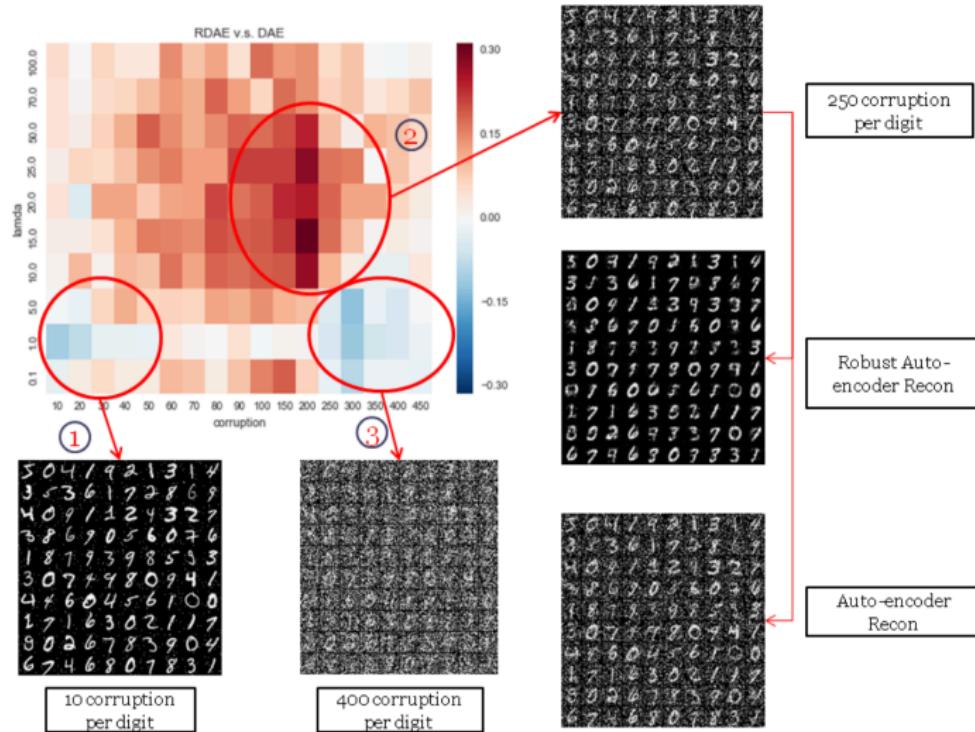
$$\min_{\theta, S} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda \|S\|_{2,1}$$

$$\min_{\theta, S} \|L_D - D_\theta(E_\theta(L_D))\|_2 + \lambda \|S^T\|_{2,1}$$

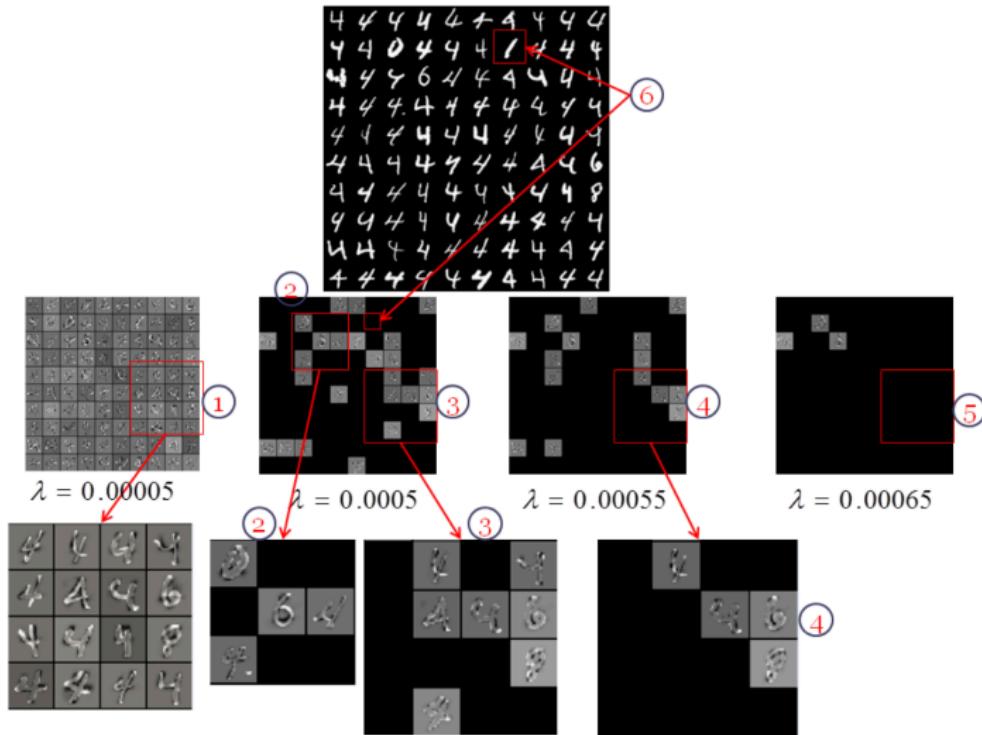
$$\text{s.t. } X - L_D - S = 0$$



# Robust Deep AutoEncoders(RDA)



# Robust Deep AutoEncoders(RDA)



# 基于自编码器的异常检测

## 优点

- ① 想法直接，可适用于不同类型的数据
- ② 不同的 AutoEncoder 变种可以针对性地解决异常检测中的问题。

## 缺点

- ① 自编码器由于没有受到任何信息指导，容易受到异常数据的影响。
- ② 自编码器的学习过程没有为异常检测优化，得到的异常检测结果可能不是最优的。



# 基于生成对抗网络的异常检测算法

## 模型假设

在生成对抗网络中，生成器更容易生成正常的样本而不是异常的样本。

## 经典方法

- ① AnoGAN
- ② f-AnoGAN
- ③ Bi-directional GAN
- ④ GANomaly



# 基于生成对抗网络的异常检测算法

## 生成对抗网络 Generative Adversarial Network

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_X} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_Z} [\log(1 - D(G(\mathbf{z})))]$$

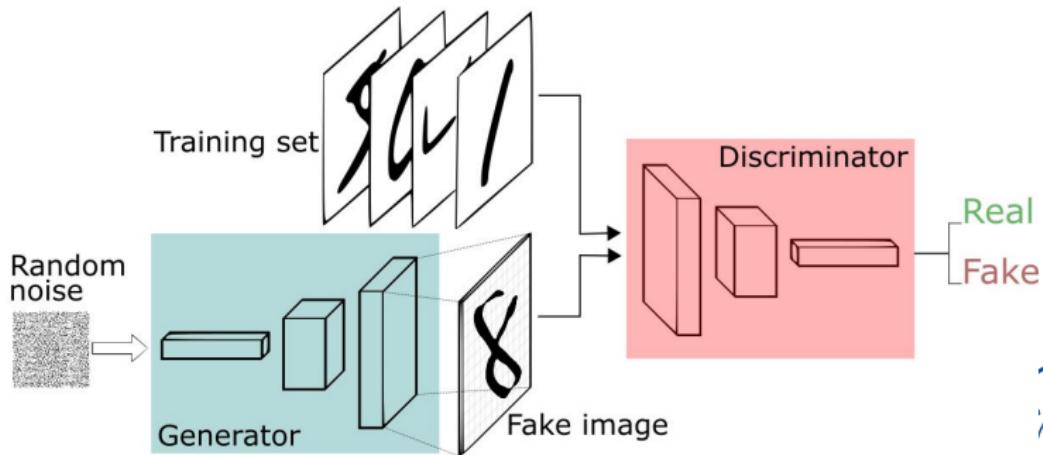


图: GAN 基本结构



# 生成对抗网络 GAN

- GAN 有 2 个对抗模型：生成网络和判别网络。
- 生成网络  $G$  从潜在空间中随机采样（随机产生噪声）作为输入，其输出结果需要尽量模仿训练集中的真实样本。
- 判别网络  $D$  的输入则为真实样本或生成网络的输出，输出为一个标量，代表其为真实样本而不是生成样本的可能性，其目的是将生成网络的输出从真实样本中尽可能分辨出来。
- 生成网络要尽可能地欺骗判别网络。两个网络相互对抗、不断调整参数，最终目的是使判别网络无法判断生成网络的输出结果是否真实。



# AnoGAN

- AnoGAN[SSW<sup>+</sup>19] 是基于 WGAN 的异常检测方法。
- 首先让我们来了解一下 Wasserstein GAN (简称 WGAN) 成功地做到了以下爆炸性的几点：
  - ① 彻底解决 GAN 训练不稳定的问题，不再需要小心平衡生成器和判别器的训练程度
  - ② 基本解决了模型塌缩的问题，确保了生成样本的多样性
  - ③ 训练过程中终于有一个像交叉熵、准确率这样的数值来指示训练的进程，这个数值越小代表 GAN 训练得越好，代表生成器产生的图像质量越高
  - ④ 以上一切好处不需要精心设计的网络架构，最简单的多层全连接网络就可以做到



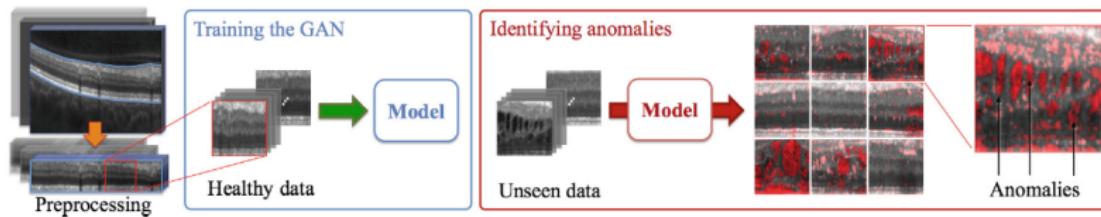
# Wasserstein GAN

- 作者用了两篇文章 [AB17][ACB17] 论证了 GAN 的问题和 WGAN 的具体实现方式
- 改进后相比原始 GAN 的算法实现流程只改了四点：
  - ① 判别器最后一层去掉 sigmoid
  - ② 生成器和判别器的 loss 不取 log
  - ③ 每次更新判别器的参数之后把它们的绝对值截断到不超过一个固定常数  $c$
  - ④ 不要用基于动量的优化算法（包括 momentum 和 Adam），推荐 RMSProp，SGD 也行



# AnoGAN

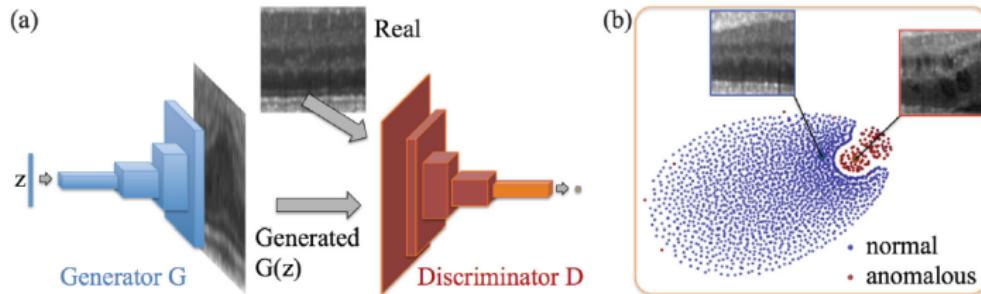
- 严谨而又细致的推导本节课不作讨论，只需知道 WGAN 是一种比较优秀的 GAN。
- 这是第一篇将 GAN 思想用于图像异常检测的论文，应用场景是医疗图像的病灶（异常）检测



**Fig. 1.** Anomaly detection framework. The preprocessing step includes extraction and flattening of the retinal area, patch extraction and intensity normalization. Generative adversarial training is performed on healthy data and testing is performed on both, unseen healthy cases and anomalous data. (Color figure online)



# AnoGAN



**Fig. 2.** (a) Deep convolutional generative adversarial network. (b) t-SNE embedding of normal (blue) and anomalous (red) images on the feature representation of the last convolution layer (orange in (a)) of the discriminator. (Color figure online)



# AnoGAN

- 训练阶段：仅利用正常样本在 GAN 上无监督地学习正常样本的一个在潜在空间中的流形分布
- 测试阶段：读入测试样本（可能是正常样本或异常样本），进行多次迭代找到一个流形空间内最接近的向量  $z$ ， $z$  对应的生成器输出与原图比对可以找到异常区域，辨别器输出又可以作为异常值，超过一定阈值则可认为是异常样本
- 两个问题
  - ① 如何设计 loss
  - ② 如何比较比较图像  $G(z'_y)$  与图像  $x$  的差距



# AnoGAN

Residual Loss

$$R(z_y) = \sum |x - G(z'_y)|$$

Discrimination Loss

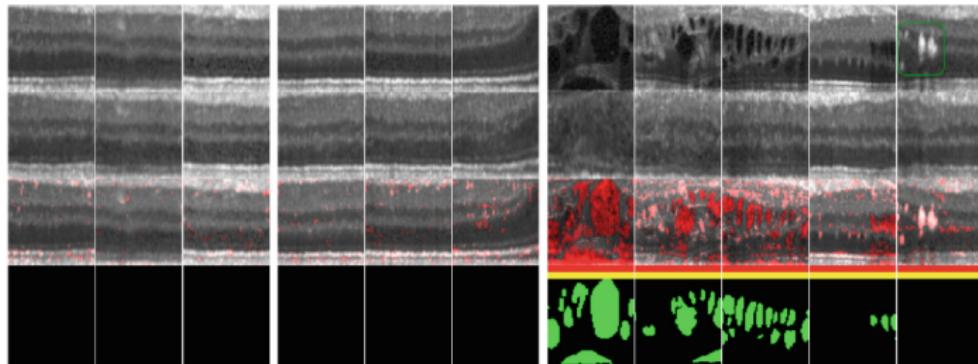
$$D(z_y) = \sum |f(x) - f(G(z'_y))|$$

最终定义的 loss 函数为

$$A(z_y) = (1 - \lambda)R(z_y) + \lambda D(z_y)$$



# AnoGAN



**Fig. 3.** Pixel-level identification of anomalies on exemplary images. First row: Real input images. Second row: Corresponding images generated by the model triggered by our proposed mapping approach. Third row: Residual overlay. Red bar: Anomaly identification by *residual score*. Yellow bar: Anomaly identification by *discrimination score*. Bottom row: Pixel-level annotations of retinal fluid. First block and second block: Normal images extracted from OCT volumes of healthy cases in the training set and test set, respectively. Third block: Images extracted from diseased cases in the test set. Last column: Hyperreflective foci (within green box). (Best viewed in color)



# f-AnoGAN

f-AnoGAN[SSW<sup>+</sup>19] 对 AnoGAN 提出了改进：AnoGAN 需要迭代优化，势必会耗费大量时间，而 f-AnoGAN 通过引入 Encoder，解决了这个问题。

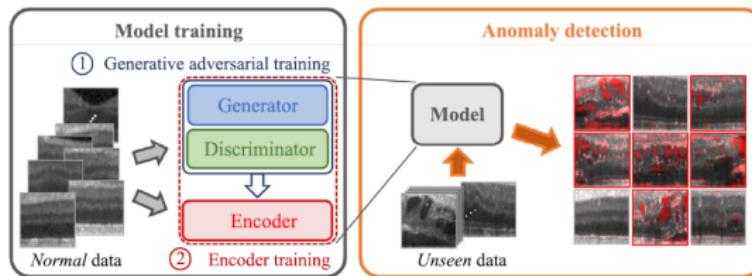


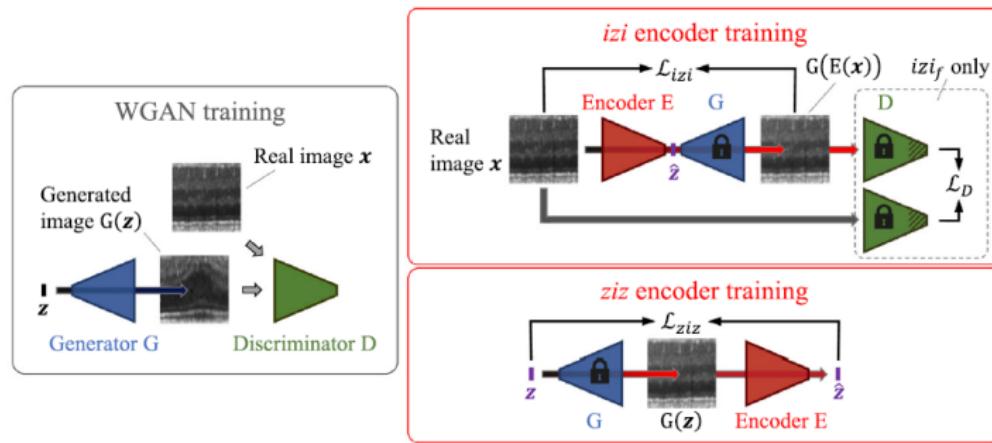
Fig. 1. Anomaly detection framework. Both steps of model training, generative adversarial training (yields a trained generator and discriminator) and encoder training (yields a trained encoder), are performed on normal ("healthy") data and anomaly detection is performed on both, unseen healthy cases and anomalous data. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



# f-AnoGAN

Encoder 的作用和训练方式：

WGAN 训练完毕后，不再改变，由生成器充当 decoder，与 Encoder 一起构成了 auto-encoder 结构，Encoder 负责将训练图片映射为隐空间中的点 Z，生成器将 Z 映射为图片。



# f-AnoGAN

训练方式一:  $izi$

- ① Encoder  $\mathbb{F}$  图片  $\times$  映射为隐空间中的点  $\hat{z}$
- ② 生成器将  $\hat{z}$  映射为图片  $G(\hat{z})$
- ③ 损失函数为 MSE:

$$L_{izi}(x) = \frac{1}{n} \|x - G(\hat{z})\|^2$$

$n$  为像素的个数

训练方法二:  $ziz$

- ① 在隐空间中随机选取一个点  $z$ , 生成器将  $z$  映射为图片  $G(z)$
- ② Encoder 将  $G(z)$  映射为隐空间中的点  $\hat{z}$
- ③ 损失函数为 MSE:

$$L_{ziz}(z) = \frac{1}{d} \|z - \hat{z}\|^2$$

$d$  为隐空间的维数



# f-AnoGAN

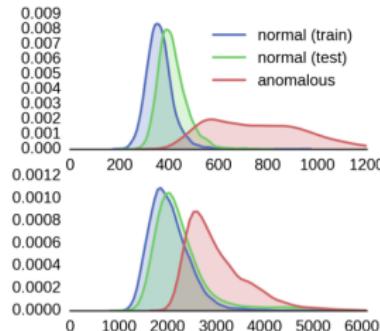
训练方式三:  $izi_f$  具体步骤:

- ① Encoder 将图片  $x$  映射为隐空间中的点  $z$
- ② 生成器将  $\hat{z}$  映射为图片  $G(\hat{z})$
- ③ 损失函数为

$$L_{izi_f}(x) = \frac{1}{n} \|x - G(\hat{z})\|^2 + \lambda \frac{1}{n_d} \|f(x) - f(G(\hat{z}))\|^2$$



# f-AnoGAN



$x$  轴表示  $L_{izi}$  与  $L_D$  的值,  $y$  轴表示频率, 可以看出

- ① 异常图片的  $L_{izi}$  与  $L_D$  普遍大于正常图片
- ② 正常图片与异常图片在  $L_{izi}$  与  $L_D$  上的取值分布重叠部分小, 说明  $L_{izi}$  与  $L_D$  对于正常图片与异常图片的区分度高

因此,  $L_{izi_f}(x)$  可用于计算异常得分



# f-AnoGAN

问题一：隐空间是否平滑连续？如果隐空间不够平滑连续，只有部分隐空间中的点能生成真实度较高的图片，为了验证隐空间是连续的，论文进行了两个实验

- ① 实验一：随机选择隐空间中的两个点，两点之间做一条位于高维度空间的直线，生成这条直线上的点对应的图片
- ② 实验二：依据真实图片在隐空间中选择两个点，两点之间做一条位于高维度空间的直线，生成这条直线上的点对应的图片

两个实验的结果如下



# f-AnoGAN

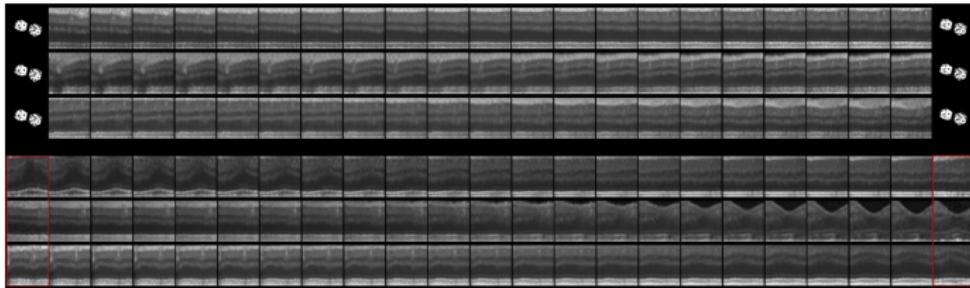
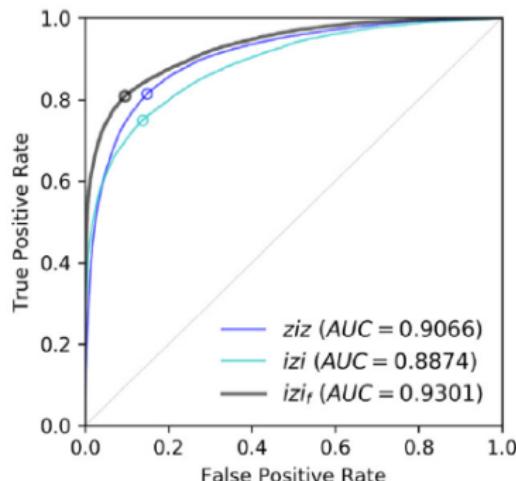
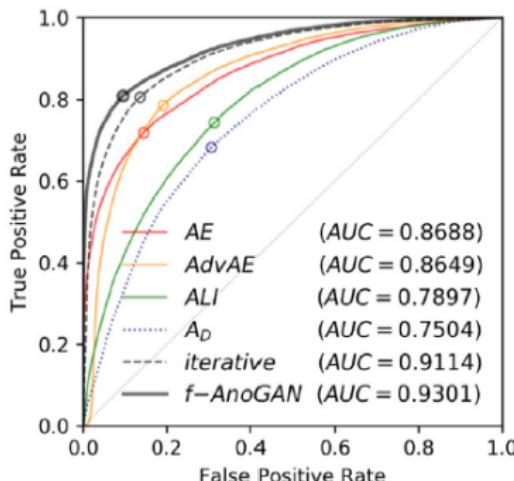


Fig. 5. Interpolations in the  $z$ -space of the trained WGAN. First three rows: Linear interpolation in  $z$ -space between randomly sampled endpoints. Last three rows: Linear interpolation in  $z$ -space between two  $z$  locations conditioned by real images (images with red edgings) taken from the training set. (A more detailed visualization can be found in B.1.) (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

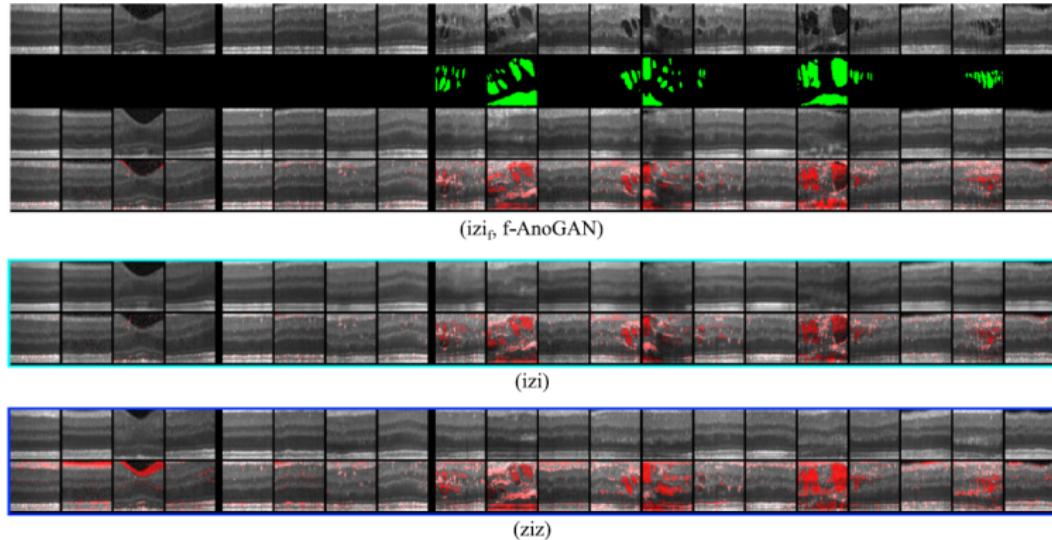
可以看到，图像之间的变化非常自然，由此可见隐空间还是比较平滑的，如果隐空间是剧烈抖动的，那么图像之间的跳变效果应该会非常明显。

# f-AnoGAN

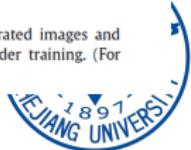
	Precision	Sensitivity	Specificity	f-score	AUC
AE	0.6824	0.7195	0.8550	0.7005	0.8688
AdvAE	0.6405	0.7856	0.8092	0.7057	0.8649
ALI	0.5063	0.7434	0.6863	0.6023	0.7897
$A_D$	0.4909	0.6831	0.6931	0.5713	0.7504
iterative	0.7202	0.8049	0.8645	0.7602	0.9114
<i>f-AnoGAN</i>	<b>0.7863</b>	<b>0.8091</b>	<b>0.9049</b>	<b>0.7975</b>	<b>0.9301</b>



## f-AnoGAN



**Fig. 7.** Encoder training comparison on pixel-level localization of anomalous image regions. Starting with the third row, we show in one row the generated images and in subsequent rows the real query images with overlaid residual: Proposed *f-AnoGAN* *izi* (row 3 and 4), *izi* (row 5 and 6), and *ziz* (row 7 and 8) encoder training. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



## Bi-directional GAN

最初的 GAN 只含有从 feature 空间到 data 空间的一个生成器 G, BiGAN 就又设计了一个从 data 空间到 feature 空间的生成器 E (Encoder), 这就形成了一个双向的结构, 目的是能够无监督地利用 E 来提取数据特征。

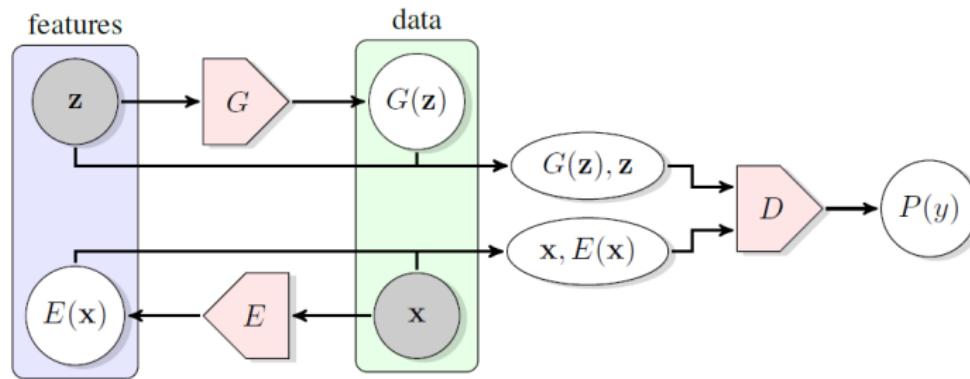


Figure 1: The structure of Bidirectional Generative Adversarial Networks (BiGAN).



# GANomaly

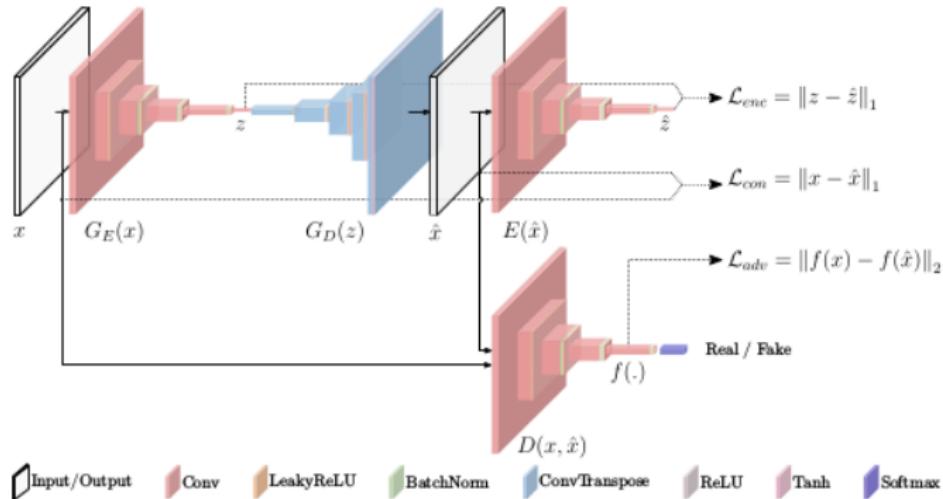


Figure 2: Pipeline of the proposed approach for anomaly detection.

图: GANomaly 网络结构



# GANomaly

- 生成网络部分由编码器  $GE(x)$  和解码器  $GD(z)$  构成，对于送入数据  $x$  经过编码器  $GE(x)$  得到 embedding  $z$ ,  $z$  经过解码器  $GD(z)$  得到  $x$  的重构数据  $\hat{x}$
- 模型的第二部分就是判别器  $D$ , 把原始图像  $x$  判为真, 重构图像  $\hat{x}$  判为假, 从而不断优化重构图像与原始图像的差距, 理想情况下重构图像与原始图像完全一致。
- 模型的第三部分是对重构图像  $\hat{x}$  再做编码的编码器  $E(\hat{x})$  得到重构图像编码的潜在变量  $\hat{z}$  。



# GANomaly

- 在训练阶段，整个模型均是通过正常样本做训练。也就是编码器  $G_E(x)$ ，解码器  $G_D(z)$  和重构编码器  $E(\hat{x})$ ，都是适用于正常样本的
- 当模型在测试阶段接受到一个异常样本，理论上此时模型的编码器，解码器将不适用于异常样本，此时得到的编码后潜在变量  $z$  和重构编码器得到的潜在变量  $\hat{z}$  的差距是大的。这个差距记为：

$$\mathcal{A}(\mathcal{X}) = \|G_E(x) - E(G(x))\|_1$$

通过设定阈值  $\phi$ ，一旦  $\mathcal{A}(x) > \phi$  模型就认定送入的样本  $x$  是异常数据。



# 基于生成对抗网络的异常检测算法

## 优点

- ① GAN 作为最经典的深度生成模型之一，可以广泛用于生成与真实数据相似的样本。而难以从潜在空间生成的样本可能是异常样本。
- ② GAN 经过多年发展，已有大量成熟的模型可用于异常检测。

## 缺点

- ① GAN 模型的训练相对困难，容易出现模型坍塌等问题
- ② 当待检测数据较为复杂时，GAN 很容易生成与大部分样本不同的样本。异常数据集容易进一步加剧 GAN 模型的训练。
- ③ 基于 GAN 的异常检测模型本质上还是训练 GAN，而不是异常检测。



# 基于自监督分类的异常检测算法

## 研究动机

由于表征学习过程自然捕获大部分数据的特性，正常样本的语义特征能够被表征学习捕获且不会受到数据扰动的影响。

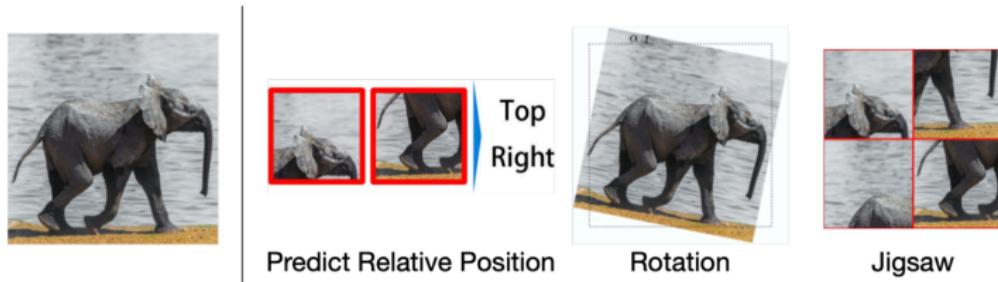
## 挑战

- ① 缺少数据标签
- ② 难以获得语义不变性
- ③ 模型训练困难

使用基于数据增强的自监督分类器进行表征学习并从中区分正常样本与异常样本。



# Deep Anomaly Detection Using Geometric Transformations (NIPS 2018)



图：常见的自监督分类

## 异常分值的定义

$$n_S(x) \triangleq \sum_{i=0}^{k-1} \log p(\mathbf{y}(T_i(x)) \mid T_i)$$



# Deep Anomaly Detection Using Geometric Transformations (NIPS 2018)

Dataset	$c_i$	RAW	OC-SVM		DAGMM	DSEBM	AD-GAN	OURS
			CAE	E2E				
CIFAR-10 (32x32x3)	0	70.6	<b>74.9</b>	61.7±1.3	41.4±2.3	56.0±6.9	64.9	74.7±0.4
	1	51.3	51.7	65.9±0.7	57.1±2.0	48.3±1.8	39.0	<b>95.7±0.0</b>
	2	69.1	68.9	50.8±0.3	53.8±4.0	61.9±0.1	65.2	<b>78.1±0.4</b>
	3	52.4	52.8	59.1±0.4	51.2±0.8	50.1±0.4	48.1	<b>72.4±0.5</b>
	4	77.3	76.7	60.9±0.3	52.2±7.3	73.3±0.2	73.5	<b>87.8±0.2</b>
	5	51.2	52.9	65.7±0.8	49.3±3.6	60.5±0.3	47.6	<b>87.8±0.1</b>
	6	74.1	70.9	67.7±0.8	64.9±1.7	68.4±0.3	62.3	<b>83.4±0.5</b>
	7	52.6	53.1	67.3±0.3	55.3±0.8	53.3±0.7	48.7	<b>95.5±0.1</b>
	8	70.9	71.0	75.9±0.4	51.9±2.4	73.9±0.3	66.0	<b>93.3±0.0</b>
	9	50.6	50.6	73.1±0.4	54.2±5.8	63.6±3.1	37.8	<b>91.3±0.1</b>
	avg	62.0	62.4	64.8	53.1	60.9	55.3	<b>86.0</b>
CIFAR-100 (32x32x3)	0	68.0	68.4	-	43.4±3.9	64.0±0.2	63.1	<b>74.7±0.4</b>
	1	63.1	63.6	-	49.5±2.7	47.9±0.1	54.9	<b>68.5±0.2</b>
	2	50.4	52.0	-	66.1±1.7	53.7±4.1	41.3	<b>74.0±0.5</b>
	3	62.7	64.7	-	52.6±1.0	48.4±0.5	50.0	<b>81.0±0.8</b>
	4	59.7	58.2	-	56.9±3.0	59.7±6.3	40.6	<b>78.4±0.5</b>
	5	53.5	54.9	-	52.4±2.2	46.6±1.6	42.8	<b>59.1±1.0</b>
	6	55.9	57.2	-	55.0±1.1	51.7±0.8	51.1	<b>81.8±0.2</b>
	7	64.4	62.9	-	52.8±3.7	54.8±1.6	55.4	<b>65.0±0.1</b>
	8	66.7	65.6	-	53.2±4.8	66.7±0.2	59.2	<b>85.5±0.4</b>
	9	70.1	74.1	-	42.5±2.5	71.2±1.2	62.7	<b>90.6±0.1</b>
	10	83.0	84.1	-	52.7±3.9	78.3±1.1	79.8	<b>87.6±0.2</b>
	11	59.7	58.0	-	46.4±2.4	62.7±0.7	53.7	<b>83.9±0.6</b>
	12	68.7	68.5	-	42.7±3.1	66.8±0.0	58.9	<b>83.2±0.3</b>
	13	<b>65.0</b>	64.6	-	45.4±0.7	52.6±0.1	57.4	58.0±0.4
	14	50.7	51.2	-	57.2±1.3	44.0±0.6	39.4	<b>92.1±0.2</b>
	15	63.5	62.8	-	48.8±1.5	56.8±0.1	55.6	<b>68.3±0.1</b>
	16	68.3	66.6	-	54.4±3.1	63.1±0.1	63.3	<b>73.5±0.2</b>
	17	71.7	73.7	-	36.4±2.3	73.0±1.0	66.7	<b>93.8±0.1</b>
	18	50.2	52.8	-	52.4±1.4	57.7±1.6	44.3	<b>90.7±0.1</b>
	19	57.5	58.4	-	50.3±1.0	55.5±0.7	53.0	<b>85.0±0.2</b>
	avg	62.6	63.1	-	50.5	58.8	54.7	<b>78.7</b>



# Effective End-to-end Unsupervised Outlier Detection via Inlier Priority of Discriminative Network(NIPS2019)

## 研究动机

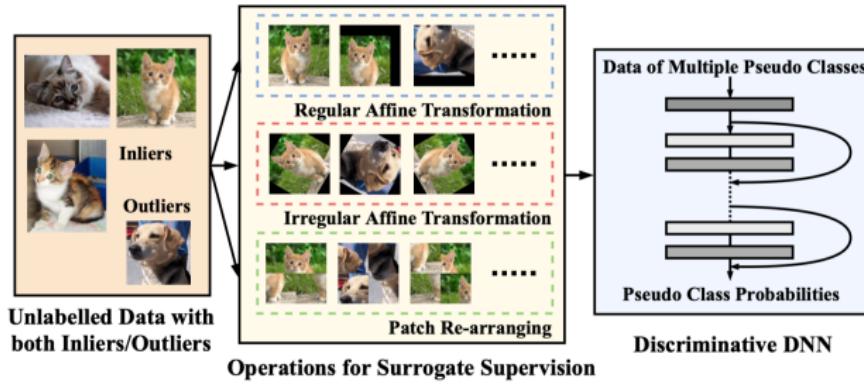
- ① 现有的仅使用 Auto-Encoder 或者卷积自编码器的方法主要捕获数据的低阶特征。
- ② 表征学习中有监督学习得到了广泛的研究，但是在无监督场景下难以应用。
- ③ 现有的大部分异常检测算法都是通过距离度量进行异常判断。



# $E^3$ Outlier

## 数据增强

- ① 旋转
- ② 翻折
- ③ 平移
- ④ 区块打乱



# 表征学习对正常样本的偏好

## Motivation

- ① 类别不平衡状态下，有监督训练会在训练过程中偏向于捕获规模较大的类的信息
- ② 正常样本在训练过程中会提供更强的梯度方向指引（模型优化方向），且与异常样本有显著差异

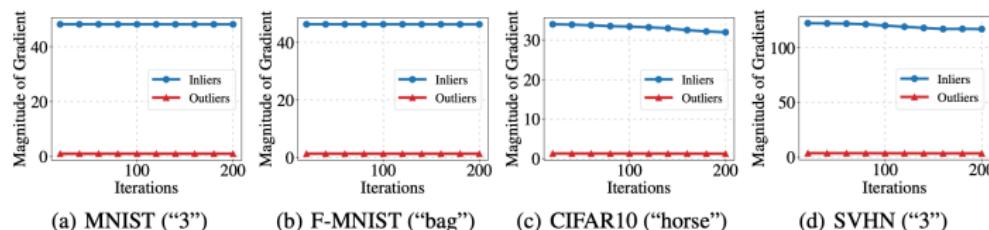


Figure 2: Inliers and outliers' gradient magnitude on example cases of benchmark datasets during SSD training. The class used as inliers is in brackets.

图：训练过程中的梯度对比



## 异常分值

### Pseudo Label based Score (PL):

$$S_{pl}(\mathbf{x}) = \frac{1}{K} \sum_{y=1}^K P^{(y)} \left( \mathbf{x}^{(y)} | \boldsymbol{\theta} \right)$$

Maximum Probability based Score (MP):

$$S_{mp}(\mathbf{x}) = \frac{1}{K} \sum_{y=1}^K \max_t P^{(t)} \left( \mathbf{x}^{(y)} | \boldsymbol{\theta} \right)$$

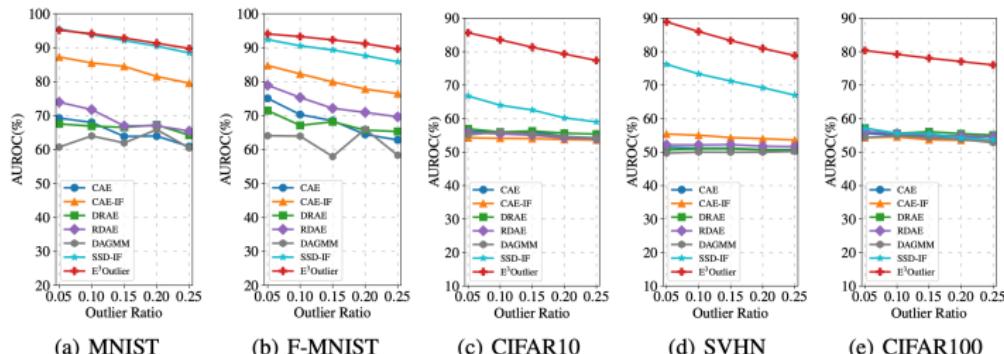
## Negative Entropy based Score (NE)

$$S_{ne}(\mathbf{x}) = \frac{1}{K} \sum_{y=1}^K \sum_{t=1}^K P^{(t)} \left( \mathbf{x}^{(y)} | \boldsymbol{\theta} \right) \log \left( P^{(t)} \left( \mathbf{x}^{(y)} | \boldsymbol{\theta} \right) \right)$$



# $E^3$ Outlier

Dataset	$\rho$	CAE	CAE-IF	DRAE	RDAE	DAGMM	SSD-IF	$E^3$ Outlier
MNIST	10%	68.0/92.0/32.9	85.5/97.8/49.0	66.9/93.0/30.5	71.8/93.1/35.8	64.0/92.9/26.6	93.8/99.2/ <b>68.7</b>	<b>94.1/99.3/67.5</b>
	20%	64.0/82.7/40.7	81.5/93.6/57.2	67.2/86.6/42.5	67.0/84.2/43.2	65.9/86.4/41.3	90.5/97.3/71.0	<b>91.3/97.6/72.3</b>
F-MNIST	10%	70.3/94.3/29.3	82.3/97.2/40.3	67.1/93.9/25.5	75.3/95.8/31.7	64.0/92.7/30.3	90.6/98.5/68.6	<b>93.3/99.0/75.9</b>
	20%	64.4/85.3/36.8	77.8/92.2/49.0	65.7/86.9/36.6	70.9/89.2/41.4	66.0/86.7/43.5	87.6/95.6/71.4	<b>91.2/97.1/78.9</b>
CIFAR10	10%	55.9/91.0/14.4	54.1/90.2/13.7	56.0/90.7/14.7	55.4/90.7/14.0	56.1/91.3/15.6	64.0/93.5/18.3	<b>83.5/97.5/43.4</b>
	20%	54.7/81.6/25.5	53.8/80.7/25.3	55.6/81.7/26.8	54.2/81.0/25.7	54.7/81.8/26.3	60.2/85.0/28.3	<b>79.3/93.1/52.7</b>
SVHN	10%	51.2/90.3/10.6	55.0/91.4/11.9	51.0/90.3/10.5	52.1/90.6/10.8	50.0/90.0/19.3	73.4/95.9/22.0	<b>86.0/98.0/36.7</b>
	20%	50.7/80.2/20.7	54.0/82.0/22.4	50.6/80.4/20.5	51.8/80.9/21.1	50.0/79.9/29.6	69.2/89.5/33.7	<b>81.0/93.4/47.0</b>
CIFAR100	10%	55.2/91.0/14.5	54.5/90.7/13.8	55.6/90.9/15.0	55.8/90.9/15.0	54.9/91.1/14.2	55.6/91.5/13.0	<b>79.2/96.8/33.3</b>
	20%	54.4/81.7/25.6	53.5/80.9/25.1	55.5/81.8/27.0	54.9/81.5/26.5	53.8/81.5/24.7	54.3/82.1/23.4	<b>77.0/92.4/46.5</b>



# 基于自监督分类的异常检测算法

## 优点

- ① 在半监督和无监督场景下均可以实现
- ② 异常的分值是由梯度下降的方向和模型更新的方向决定的。

## 缺陷

- ① 数据增强的形式由数据驱动，难以推广
- ② 使用 pretext task 进行的分类任务，本质上不是为异常检测做优化。



- ① 深度异常检测
- ② 深度特征提取
- ③ 基于特征提取的异常性判断
- ④ 面向异常检测的特征学习
- ⑤ 端到端异常检测模型



# 面向异常检测的特征学习

## 研究动机

前述的表征学习模型，本质上并不是为了异常检测优化。一个好的深度异常检测模型应当学习适用于异常检测场景的表征。

通用框架：

$$\begin{aligned} \{\Theta^*, \mathbf{W}^*\} &= \arg \min_{\Theta, \mathbf{W}} \sum_{\mathbf{x} \in \mathcal{X}} \ell(f(\phi(\mathbf{x}; \Theta); \mathbf{W})) \\ s_{\mathbf{x}} &= f(\phi(\mathbf{x}; \Theta^*); \mathbf{W}^*) \end{aligned}$$

其中  $f$  是一个现有的异常检测目标函数。



# 基于距离的深度异常检测方法

## 基于距离的异常检测方法

- ① DB-outlier
- ② KNN 及其变种

## 基于距离的深度异常检测方法

### 研究动机

经过深度表征学习后，正常样本在特征空间会处于相对稠密的区域，而异常样本则会处于相对稀疏的空间。



# Learning Representations of Ultrahigh-dimensional Data for Random Distance-based Outlier Detection(KDD 2018)

## 研究动机

在随机采样的样本中，pseudo-label 为异常样本与邻居的距离大于 pseudo-label 为正常样本与邻居的距离。

pseudo-label 是用现成的异常检测算法得到。

与深度聚类方法中使用 K-means 获得样本伪标签的思想一致



# Learning Representations of Ultrahigh-dimensional Data for Random Distance-based Outlier Detection

## 优化目标

$$L_{\text{query}} = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{A}, \mathbf{x}' \in \mathcal{N}} \max \left\{ 0, m + f(\mathbf{x}', \mathcal{S}; \Theta) - f(\mathbf{x}, \mathcal{S}; \Theta) \right\}$$

$$f(\mathbf{x}, \mathcal{S}; \Theta) = \min_{\mathbf{x}' \in \mathcal{S}} \left\| \phi(\mathbf{x}; \Theta), \phi(\mathbf{x}'; \Theta) \right\|_2$$

在学习到的表征空间中，异常样本与其他样本的最小距离，正常样本与其他样本的最小距离，差值需要大于  $m$ 。

距离的计算方式可以替换为任意已有的距离度量，例如 KNN distance



# Learning Representations of Ultrahigh-dimensional Data for Random Distance-based Outlier Detection

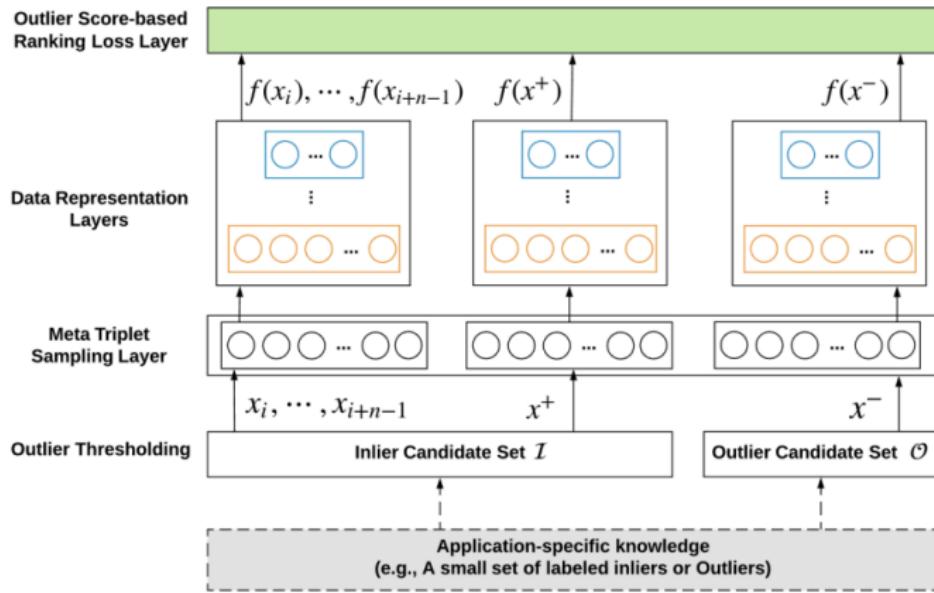


图: RAMODO 模型框架



# Unsupervised Representation Learning by Predicting Random Distances(IJCAI 2020)

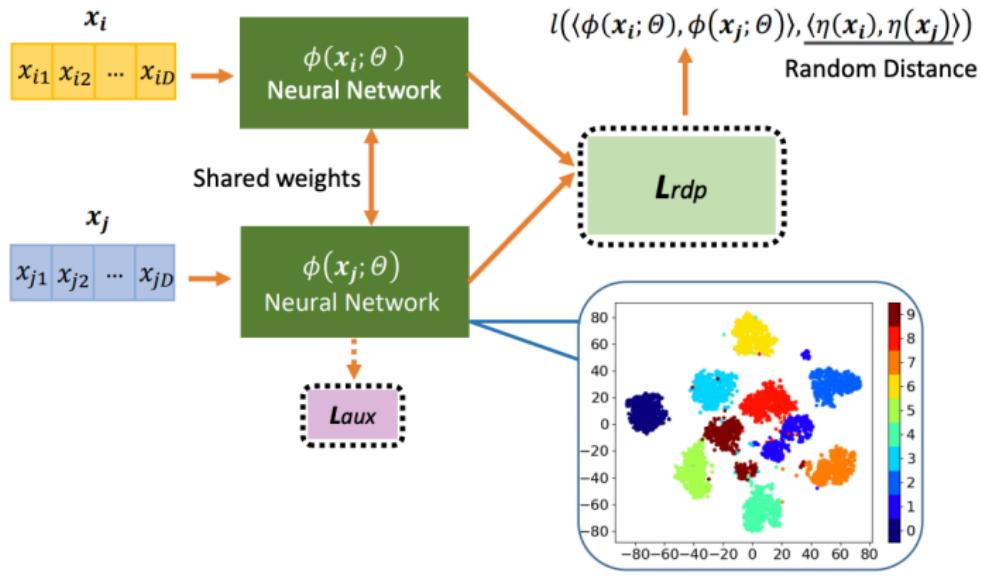


图: RDP 模型结构



# Unsupervised Representation Learning by Predicting Random Distances(IJCAI 2020)

## Random Projection Distance

$$L_{rdp}(\mathbf{x}_i, \mathbf{x}_j) = l(\langle \phi(\mathbf{x}_i; \Theta), \phi(\mathbf{x}_j; \Theta) \rangle, \langle \eta(\mathbf{x}_i), \eta(\mathbf{x}_j) \rangle)$$

## 简单的 RDP 实例

$$L_{rdp}(\mathbf{x}_i, \mathbf{x}_j) = (\phi(\mathbf{x}_i; \Theta) \cdot \phi(\mathbf{x}_j; \Theta) - \eta(\mathbf{x}_i) \cdot \eta(\mathbf{x}_j))^2$$

## 基于 RDP 的异常检测

$$L_{aux}^{ad}(\mathbf{x}) = (\phi(\mathbf{x}; \Theta) - \eta(\mathbf{x}))^2$$



# Unsupervised Representation Learning by Predicting Random Distances(IJCAI 2020)

## 关于 RDP 的一些思考

- ① 从点特征到关系特征
- ② 变换不变性与数据增强
- ③ RDP 与对比学习

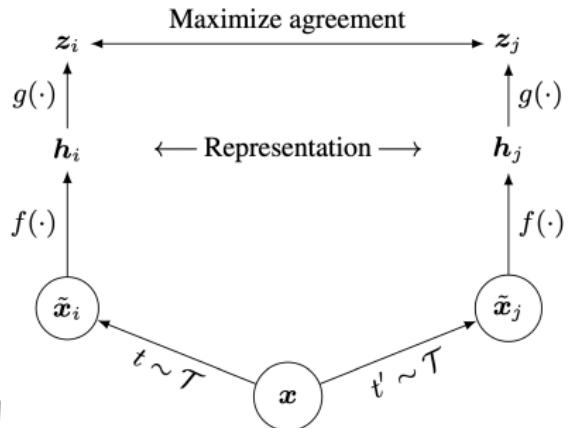


图: SimCLR



# 基于距离的深度异常检测方法

基于距离的深度异常检测方法总结：

## 优点

- ① 基于距离的异常检测方法简单直接，得到了广泛的研究
- ② 同时适用于传统的低维特征以及高维复杂特征数据的异常检测

## 缺点

- ① 基于距离的方法需要计算两两之间的相似度，难以应用于大规模数据集
- ② 基于距离的方法，总体效果一般



# 基于聚类的深度异常检测方法

## 基于聚类的异常检测方法

- ① 聚类大小
- ② 样本到聚类中心的距离
- ③ 属于某个类的概率

## 基于深度聚类的深度异常检测方法

- ① 在深度聚类的基础上进行异常检测
- ② 在深度聚类的过程中考虑异常样本的影响



# Deep Autoencoding Gaussian Mixture Model (DAGMM)

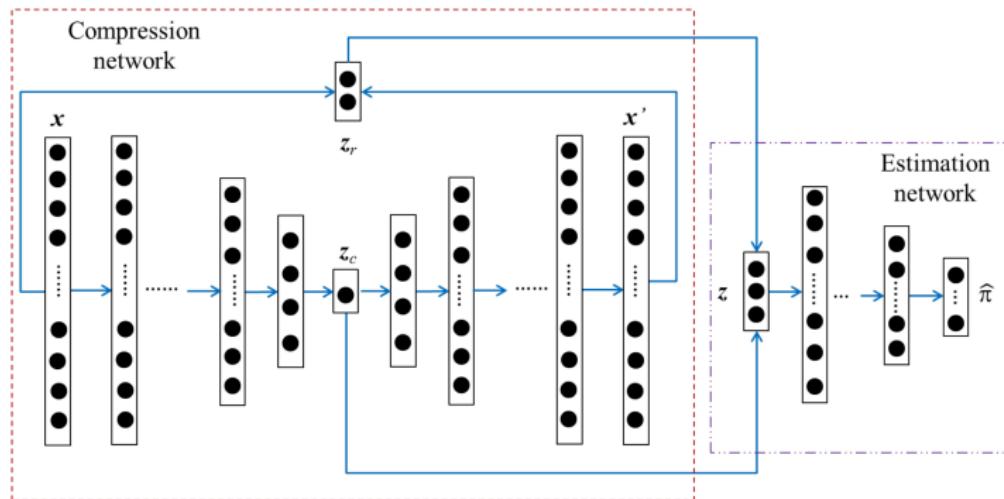


图: DAGMM 模型结构



# DAGMM

## Compression Network

$$\begin{aligned}\mathbf{z}_c &= h(\mathbf{x}; \theta_e), \mathbf{x}' = g(\mathbf{z}_c; \theta_d), \\ \mathbf{z}_r &= f(\mathbf{x}, \mathbf{x}') \\ \mathbf{z} &= [\mathbf{z}_c, \mathbf{z}_r]\end{aligned}$$

将样本重构的损失值与降维后的表征融合作为样本的低维表征。

## Estimation Network

$$\mathbf{p} = MLN(\mathbf{z}; \theta_m), \quad \hat{\gamma} = \text{softmax}(\mathbf{p})$$

$$\begin{aligned}\hat{\phi}_k &= \sum_{i=1}^N \frac{\hat{\gamma}_{ik}}{N}, \quad \hat{\mu}_k = \frac{\sum_{i=1}^N \hat{\gamma}_{ik} \mathbf{z}_i}{\sum_{i=1}^N \hat{\gamma}_{ik}} \\ \hat{\Sigma}_k &= \frac{\sum_{i=1}^N \hat{\gamma}_{ik} (\mathbf{z}_i - \hat{\mu}_k) (\mathbf{z}_i - \hat{\mu}_k)^T}{\sum_{i=1}^N \hat{\gamma}_{ik}}\end{aligned}$$



DAGMM

基于 GMM 的 Sample Energy

$$E(\mathbf{z}) = -\log \left( \sum_{k=1}^K \hat{\phi}_k \frac{\exp \left( -\frac{1}{2} (\mathbf{z} - \hat{\mu}_k)^T \hat{\Sigma}_k^{-1} (\mathbf{z} - \hat{\mu}_k) \right)}{\sqrt{|2\pi \hat{\Sigma}_k|}} \right)$$

## 损失函数

$$J(\theta_e, \theta_d, \theta_m) = \frac{1}{N} \sum_{i=1}^N L(\mathbf{x}_i, \mathbf{x}'_i) + \frac{\lambda_1}{N} \sum_{i=1}^N E(\mathbf{z}_i) + \lambda_2 P(\hat{\Sigma})$$



# DAGMM

Method	KDDCUP			Thyroid		
	Precision	Recall	$F_1$	Precision	Recall	$F_1$
OC-SVM	0.7457	0.8523	0.7954	0.3639	0.4239	0.3887
DSEBM-r	0.1972	0.2001	0.1987	0.0404	0.0403	0.0403
DSEBM-e	0.7369	0.7477	0.7423	0.1319	0.1319	0.1319
DCN	0.7696	0.7829	0.7762	0.3319	0.3196	0.3251
GMM-EN	0.1932	0.1967	0.1949	0.0213	0.0227	0.0220
PAE	0.7276	0.7397	0.7336	0.1894	0.2062	0.1971
E2E-AE	0.0024	0.0025	0.0024	0.1064	0.1316	0.1176
PAE-GMM-EM	0.7183	0.7311	0.7246	0.4745	0.4538	0.4635
PAE-GMM	0.7251	0.7384	0.7317	0.4532	<b>0.4881</b>	0.4688
DAGMM-p	0.7579	0.7710	0.7644	0.4723	0.4725	0.4713
DAGMM-NVI	0.9290	<b>0.9447</b>	0.9368	0.4383	0.4587	0.4470
DAGMM	<b>0.9297</b>	0.9442	<b>0.9369</b>	<b>0.4766</b>	0.4834	<b>0.4782</b>
Method	Arrhythmia			KDDCUP-Rev		
	Precision	Recall	$F_1$	Precision	Recall	$F_1$
OC-SVM	<b>0.5397</b>	0.4082	0.4581	0.7148	<b>0.9940</b>	0.8316
DSEBM-r	0.1515	0.1513	0.1510	0.2036	0.2036	0.2036
DSEBM-e	0.4667	0.4565	0.4601	0.2212	0.2213	0.2213
DCN	0.3758	0.3907	0.3815	0.2875	0.2895	0.2885
GMM-EN	0.3000	0.2792	0.2886	0.1846	0.1746	0.1795
PAE	0.4393	0.4437	0.4403	0.7835	0.7817	0.7826
E2E-AE	0.4667	0.4538	0.4591	0.7434	0.7463	0.7448
PAE-GMM-EM	0.3970	0.4168	0.4056	0.2822	0.2847	0.2835
PAE-GMM	0.4575	0.4823	0.4684	0.6307	0.6278	0.6292
DAGMM-p	0.4909	0.4679	0.4787	0.2750	0.2810	0.2780
DAGMM-NVI	0.5091	0.4892	0.4981	0.9211	0.9211	0.9211
DAGMM	0.4909	<b>0.5078</b>	<b>0.4983</b>	<b>0.9370</b>	0.9390	<b>0.9380</b>

图: DAGMM 实验结果



# DAGMM

## 优点

- ① 融合了 AutoEncoder 对正常样本的重构能力
- ② 显式使用重构分数作为特征用于 GMM
- ③ 使用深度网络与 GMM 结合

## 缺点

- ① 只能用于特征型数据
- ② 难以直接获得 K
- ③ 模型需要预训练



# 基于聚类的深度异常检测模型

## 优点

- ① 深度聚类技术快速发展，有很多现成的深度聚类技术
- ② 在表征学习的过程中融入异常检测场景特征，获得更好的聚类效果

## 缺点

- ① 深度聚类本身效果不够理想
- ② 现有深度聚类方法对异常数据的鲁棒性不强



- ① 深度异常检测
- ② 深度特征提取
- ③ 基于特征提取的异常性判断
- ④ 面向异常检测的特征学习
- ⑤ 端到端异常检测模型



# 端到端 One-Class 模型

## 模型假设

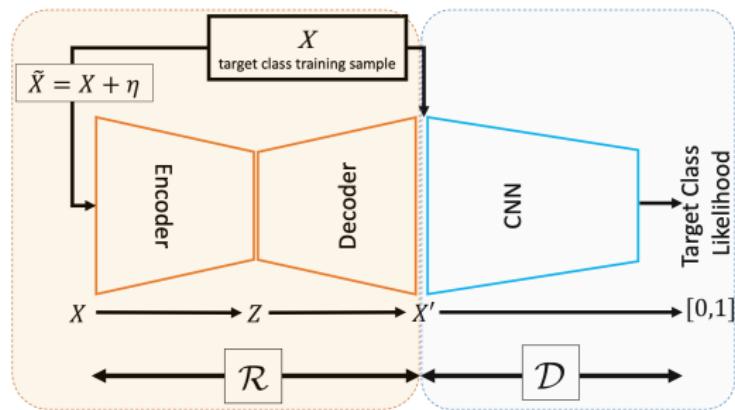
- ① 异常数据可以通过 GAN 等生成模型生成
- ② 所有的正常样本均可以通过一个 one-class 判别模型表达。

## 与基于 GAN 的异常检测模型的区别

- ① GAN 的目标是生成与真实分布相近的数据，而端到端模型的目标是学习能够判断正常异常的分类器。
- ② 端到端模型可以直接判断样本是否为异常，而基于 GAN 的方法只能从真实样本与生成样本之间的差异来判断。



## Adversarially Learned One-Class Classification(ALOCC)



$$\min_{AE} \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_X} [\log D(\mathbf{x})] + \mathbb{E}_{\hat{\mathbf{x}} \sim p_{\hat{X}}} [\log(1 - D(AE(\hat{\mathbf{x}})))]$$

$$\mathcal{L}_{\mathcal{R}} = \|X - X'\|^2$$



# ALOCC

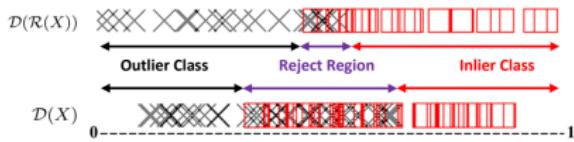
判别网络  $\mathcal{D}$  用于 one-class 分类，从所有样本中找到正常样本。生成网络  $\mathcal{R}$  用于增强正常样本并且生成异常样本。

ALOCC 模型的特点：

- ① 融合了基于 AE 的异常检测方法，即 AE 可以更好地重构正常样本
- ② 仅使用基于 AE 的方法，容易收到输入数据中的异常值以及生成器中生成的异常数据的影响
- ③ 判别器使用 AE 重构之后的样本进行学习，可以更好地判断数据是否是异常。



## ALOCC



图：判别器对不同输入的判别能力

图：正常样本为 1 时，ALOCC 对不同数据的重构能力。



# Fence GAN: Towards Better Anomaly Detection

## 研究动机

传统的 GAN 希望生成与输入数据空间中相同的样本，这使得生成的样本与原始样本重合，不利于异常检测（异常样本在输入数据中处于相对边缘的位置）。

## Encirclement Loss (惩罚非边缘的数据)

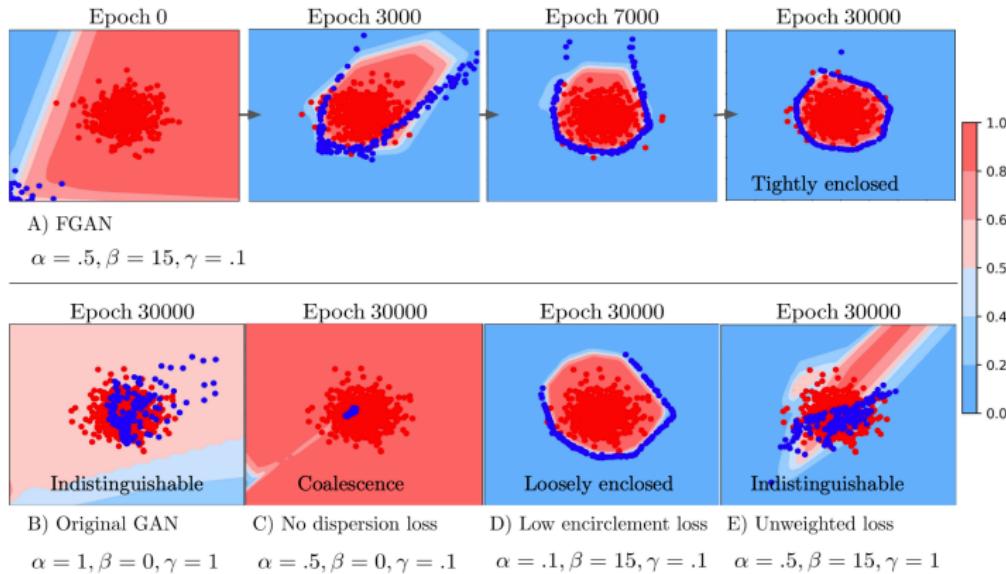
$$EL(G_\theta, D_\phi, \mathcal{Z}) = \frac{1}{N} \sum_{i=1}^N [\log(|\alpha - D_\phi(G_\theta(z_i))|)]$$

## Dispersion loss(防止模型坍塌)

$$DL(G_\theta, \mathcal{Z}) = \frac{1}{\frac{1}{N} \sum_{i=1}^N (\|G_\theta(z_i) - \mu\|_2)}$$



# FGAN



图：FGAN 学习的数据边界



# 端到端 One-Class 模型

## 优点

- ① 直接优化异常检测
- ② 融合了 GAN 对数据分布的学习能力以及 One-Class 分类的异常检测能力

## 缺点

- ① 生成的样本难以模拟所有的异常情况
- ② 训练过程不稳定



- ① 深度异常检测
- ② 深度特征提取
- ③ 基于特征提取的异常性判断
- ④ 面向异常检测的特征学习
- ⑤ 端到端异常检测模型



# 参考文献

## ① Deep Learning for Anomaly Detection: A Review



# References I

-  Martin Arjovsky and Léon Bottou, *Towards principled methods for training generative adversarial networks*, arXiv preprint arXiv:1701.04862 (2017).
-  Martin Arjovsky, Soumith Chintala, and Léon Bottou, *Wasserstein generative adversarial networks*, International conference on machine learning, PMLR, 2017, pp. 214–223.
-  Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Georg Langs, and Ursula Schmidt-Erfurth, *f-anogan: Fast unsupervised anomaly detection with generative adversarial networks*, Medical image analysis **54** (2019), 30–44.

