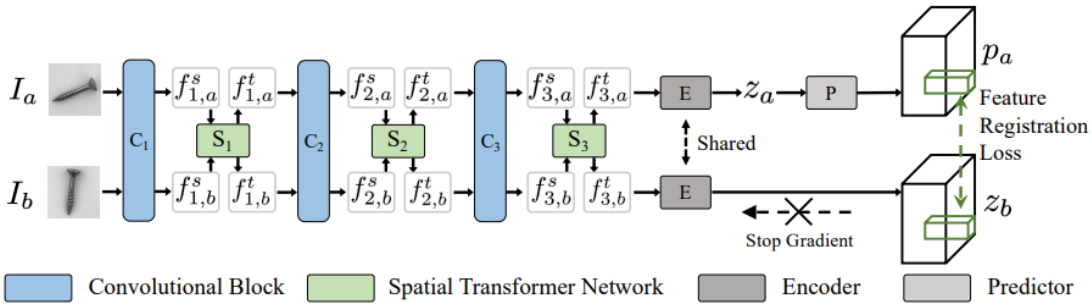
RegAD

--Few-Shot Anomaly Detection

Registration based Few-Shot Anomaly Detection: 无需微调即可推广，上交大、上海人工智能实验室等提出基于配准的少样本异常检测框架

RegAD的模型架构：



The model architecture of the proposed RegAD. Given paired images from the same category, features are extracted by three convolutional residual blocks each followed by a spatial transformer network. A Siamese network acts as the feature encoder, supervised by a registration loss for feature similarity maximization.（给定来自同一类别的图像对，通过三个卷积 残差块提取特征，每个残差块后跟一个空间变换网络。孪生网络充当特征编码器， 由配准损失来监督 特征相似度最大化）.

一、Paper Reading

First:

1、Abstract:

本文考虑了少样本异常检测（FSAD），这是一种实用但尚未充分研究的异常检测（AD）设置，其中在训练时仅为每个类别提供有限数量的正常图像。到目前为止，现有的 FSAD 研究遵循标准 AD 所使用的每类别一个模型的学习范式，并且尚未探索类别间的共性。受到人类如何检测异常的启发，即将有问题的图像与正常图像进行比较，我们在这里利用配准（一种本质上可跨类别推广的图像对齐任务）作为代理任务来训练与类别无关的异常检测模型。在测试过程中，通过比较测试图像及其相应的支持（正常）图像的注册特征来识别异常。据我们所知，这是第一个训练单个可泛化模型并且不需要针对新类别进行重新训练或参数微调的 FSAD 方法。实验结果表明，该方法在 MVTec 和 MPDD 基准上的 AUC 优于最先进的 FSAD 方法 3%-8%。源代码位于：https://github.com/MediaBrain-SJTU/RegAD

2、论文中的图片、表格

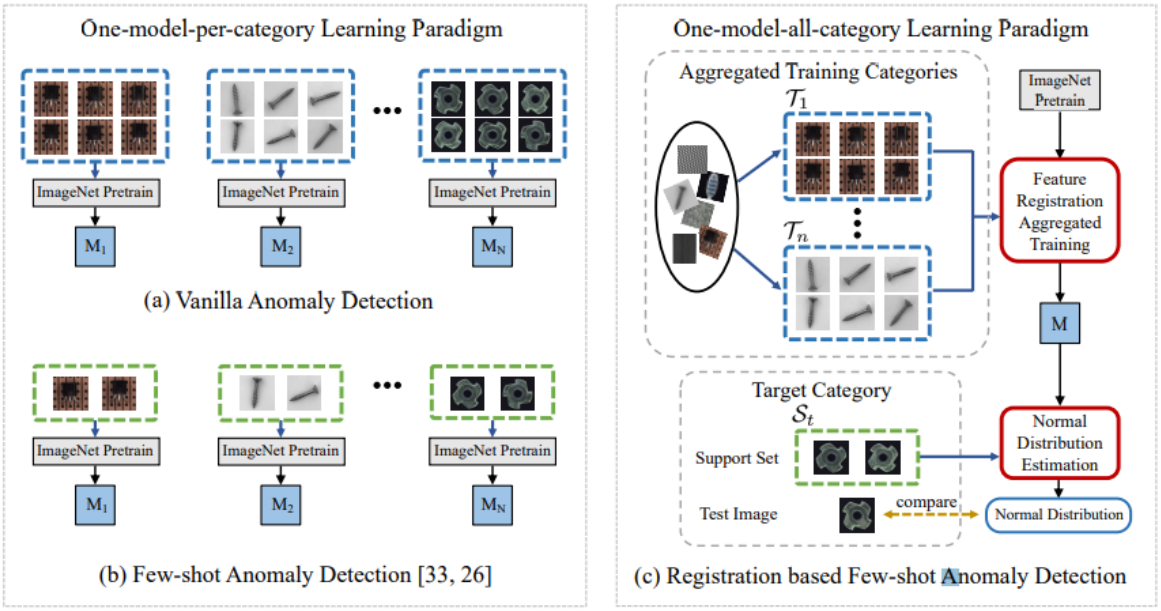


图 1. 与 (a) vanilla AD 和 (b) 每个类别一个模型学习范式下的现有 FSAD 方法不同，所提出的方法 (c) 利用特征注册作为 FSAD 的类别不可知方法，在-模型全类别学习范式。该模型使用多个类别的聚合数据进行训练，无需任何参数微调即可直接适用于新类别，只需要在给定相应支持集的情况下估计正态特征分布。

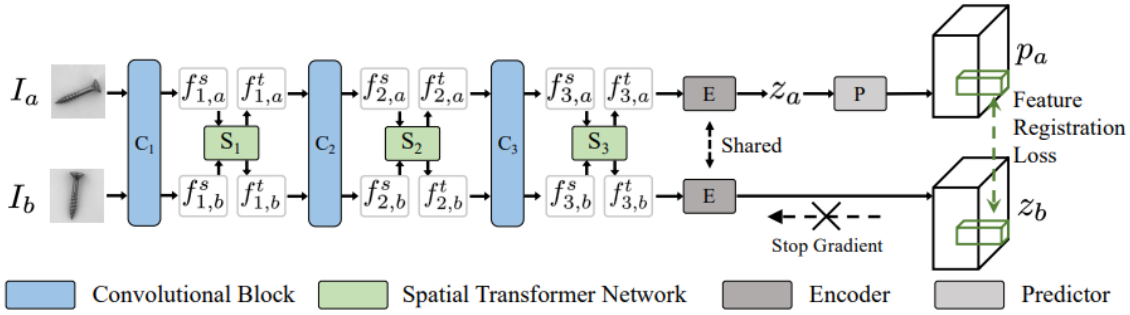


图 2. 所提出的 RegAD 的模型架构。给定来自同一类别的配对图像，通过三个卷积残差块提取特征，每个残差块后面跟着一个空间变换器网络。孪生网络充当特征编码器，通过配准损失进行监督，以实现特征相似性最大化。

Category	k=2			k=4			k=8		
	TDG+ [36]	DiffNet+ [29]	RegAD (ours)	TDG+ [36]	DiffNet+ [29]	RegAD (ours)	TDG+ [36]	DiffNet+ [29]	RegAD (ours)
Bottle	69.3	99.3	**99.4**	69.6	99.3	**99.4**	70.3	99.4	**99.8**
Cable	68.3	**85.3**	65.1	70.3	**85.2**	76.1	74.7	**87.9**	80.6
Capsule	55.1	**73.0**	67.5	47.6	**80.3**	72.4	44.7	**78.6**	76.3
Carpet	66.2	78.4	**96.5**	68.7	78.6	**97.9**	78.2	78.5	**98.5**
Grid	83.8	62.1	**84.0**	86.2	60.5	**91.2**	87.6	78.5	**91.5**
Hazelnut	67.2	94.9	**96.0**	71.2	**95.8**	95.8	82.8	**97.9**	96.5
Leather	93.6	90.7	**99.4**	93.2	91.2	**100**	93.5	92.2	**100**
Metal Nut	67.1	61.9	**91.4**	69.2	67.3	**94.6**	68.7	67.6	**98.3**
Pill	69.2	**83.2**	81.3	64.7	**84.0**	80.8	67.9	**82.1**	80.6
Screw	**98.8**	73.4	52.5	**98.8**	72.5	56.6	**99.0**	75.0	63.4
Tile	86.3	**97.0**	94.3	87.2	**98.0**	95.5	87.4	**99.6**	97.4
Toothbrush	54.4	60.8	**86.6**	57.8	62.5	**90.9**	57.6	60.8	**98.5**
Transistor	55.9	61.8	**86.0**	67.7	62.2	**85.2**	71.5	63.3	**93.4**
Wood	98.4	98.1	**99.2**	98.3	96.4	**98.6**	98.4	**99.4**	**99.4**
Zipper	64.4	**89.2**	86.3	65.3	84.8	**88.5**	66.3	87.3	**94.0**
Average	73.2	80.6	**85.7**	74.4	81.3	**88.2**	76.6	83.2	**91.2**

表 1. MVTec 数据集上的 k-shot 异常检测结果，与最先进的方法进行比较。结果以 10 次运行的平均 AUC 百分比形式列出，并针对每个类别单独标记。最后一行还报告了所有类别的宏观平均分数。效果最好的方法以粗体显示。

Methods	Data	ImageNet Pretrain	Backbone	MVTec		MPDD	
				image	pixel	image	pixel
RegAD (k=4)	4 images	✓	Res18	88.2	95.8	68.8	93.9
RegAD (k=8)	8 images	✓	Res18	91.2	96.7	71.9	95.1
RegAD (k=16)	16 images	✓	Res18	92.7	96.6	75.3	96.3
RegAD (k=32)	32 images	✓	Res18	94.6	96.9	76.8	96.3
GANomaly [1]	full data	✗	UNet	80.5	-	64.8	-
ARNet [42]	full data	✗	UNet	83.9	-	69.7	-
MKD [33]	full data	✓	Res18	87.7	90.7	-	-
CutPaste [21]	full data	✓	Res18	95.2	96.0	-	-
FYD [45]	full data	✓	Res18	97.3	97.4	-	-
PaDiM [8]	full data	✓	WRN50	97.9	97.5	74.8	96.7
PatchCore [28]	full data	✓	WRN50	99.1	98.1	82.1	95.7
CflowAD [14]	full data	✓	WRN50	98.3	98.6	86.1	97.7

表 4. MVTec 和 MPDD 数据集上的异常检测和异常定位结果，与最先进的普通 AD 方法进行比较。结果以 AUC（以百分比表示）列出，作为每个数据集中所有类别的宏观平均得分。

- 公式

$$\begin{pmatrix} x_i^t \\ y_i^t \end{pmatrix} = S_i(f_i^s) = A_i \begin{pmatrix} x_i^s \\ y_i^s \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^s \\ y_i^s \\ 1 \end{pmatrix}, \tag{1}$$

$$\mathcal{D}(p_a, z_b) = -\frac{p_a}{||p_a||_2} \cdot \frac{z_b}{||z_b||_2}, \tag{2}$$

$$\mathcal{L} = \frac{1}{2}(\mathcal{D}(p_a, z_b) + \mathcal{D}(p_b, z_a)). \tag{3}$$

$$\Sigma_{ij} = \frac{1}{N-1}\sum_{k=1}^{N}\left(f_{ij}^{k} - \mu_{ij}\right)\left(f_{ij}^{k} - \mu_{ij}\right)^{\mathrm{T}} + \epsilon I, \tag{4}$$

$$\mathcal{M}\left(f_{ij}\right) = \sqrt{\left(f_{ij} - \mu_{ij}\right)^{T} \Sigma_{ij}^{-1}\left(f_{ij} - \mu_{ij}\right)}. \tag{5}$$

3、结论

本文提出了一种利用注册（一种本质上可跨类别推广的任务）作为代理任务的 FSAD 方法。只给每个类别一些正常样本，我们用聚合数据训练了一个与类别无关的特征注册网络。该模型被证明可以直接推广到新类别，不需要重新训练或参数微调。通过比较测试图像及其相应的支持（正常）图像的注册特征来识别异常。对于异常检测和异常定位，即使与使用大量数据进行训练的普通 AD 方法相比，该方法也具有竞争力。令人印象深刻的结果表明所提出的方法在现实世界的异常检测环境中具有很高的应用潜力。

- 本文的贡献主要如下：

1、引入特征配准作为少样本异常检测（FSAD）的一种类别无关方法。据我们所知，这是第一种FSAD方法，可以训练单一的可推广模型，并且不需要对新类别进行再训练或参数微调。
2、在最近的基准数据集上进行的大量实验表明，所提出的RegAD在异常检测和异常定位任务上都优于最先进的FSAD方法。

Second:

参考： https://blog.csdn.net/qq_43428929/article/details/126955989

1、大概流程:

基于配准的少样本异常检测的框架。与常规的异常检测方法（one-model-per-category）不同，这项工作（one-model-all-category）首先使用多类别数据联合训练一个基于配准的异常检测通用模型。来自不同类别的正常图像一起用于联合训练模型，随机选择来自同一类别的两个图像作为训练对。在测试时，为目标类别以及每个测试样本提供了由几个正常样本组成的支撑集。给定支撑集，使用基于统计的分布估计器估计目标类别注册特征的正态分布。超出统计正态分布的测试样本被视为异常。

这项工作采用了一个简单的配准网络，同时参考了 Siamese [1], STN [2] 和 FYD [3]。具体地说，以孪生神经网络（Siamese Network）为框架，插入空间变换网络（STN）实现特征配准。为了更好的鲁棒性，本文作者利用**特征级的配准损失**，而不是像典型的配准方法那样逐像素配准，这可以被视为像素级配准的松弛版本。（https://www.linkresearcher.com/theses/3d5dafdb-cee1-4a79-ac76-f61dc2608b10）

- 训练过程：

训练过程中，作者提出了一个创新点即将多种类别的图片输入进去进行混合训练，每次训练随机选取某个类别中的图像对。
训练数据集均为normal样本。

- 测试过程：

测试过程分为两步，因为作者要希望模型能够预测其它类别的异常，即训练得到的模型不需要微调等操作直接可以拿来使用预测出新的类别的高斯分布：

a、把新的类别的normal数据集输入训练好的模型得到特征值，计算均值和方差得到新类别的高斯分布；

b、需要输入新类别的测试数据图片，然后利用一定的度量手段来度量（马氏距离）该数据输入进网络得到特征图是否满足新类别的高斯分布；

在测试时，作者对新类别的数据进行了数据增强。

2、STN(spatial transformer network, 空间变换网络)

参考：https://blog.csdn.net/qq_43428929/article/details/126984825

STN可以通过网络训练出单应性矩阵并利用单应性矩阵对输入图像进行变换。

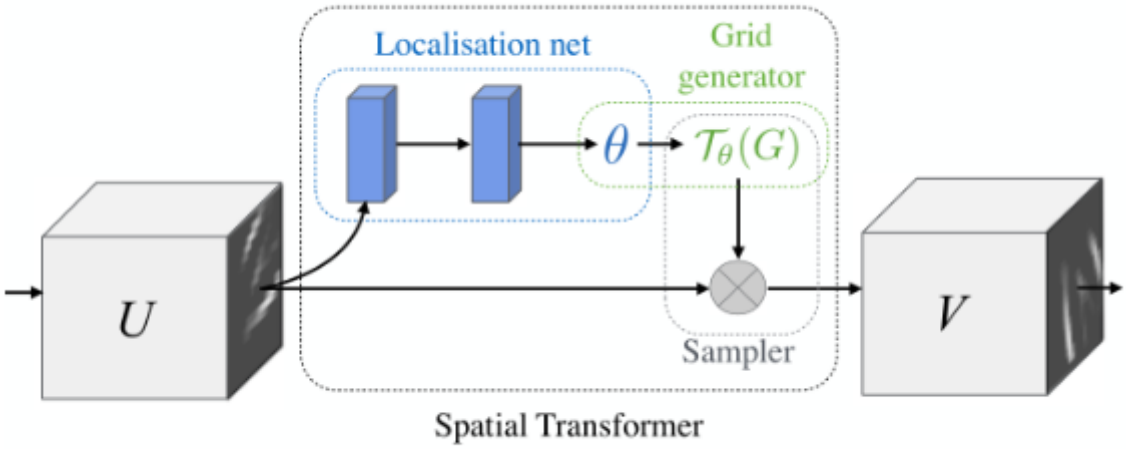
- 单应性矩阵

单应性矩阵A是用来进行图像变换：X'=HX的变换矩阵（3x3维）。

何是3x3维，不是2x3维？ ： 仿射变换可以用一个2x3的仿射矩阵表示，其中矩阵的元素由变换参数决定。透视变换通常由一个3x3的透视矩阵来表示，其中矩阵的元素由变换参数决定。如果 是一个2x3的矩阵，那么上述关系将无法完全描述透视变换，因为这个矩阵不足以包含所有必要的信息。

- STN

STN通过自身网络设置，能够学习得到模型变换的单应性矩阵（这里是2x3维矩阵不是3x3维矩阵）。通过学习得到的单应性矩阵变换图像。



STN网络包含三部分：

a、Localisation net——神经网络+FC—> theta（单应性矩阵）

通过神经网络提取特征（常用的有CNN）+FC得到最终的theta值，theta值为六维。（2x3维---仿射变换）

b、Grid generator——图像空间变换

通过映射得到对应的变换图片，注意公式里输入图像和目标图像的位置。

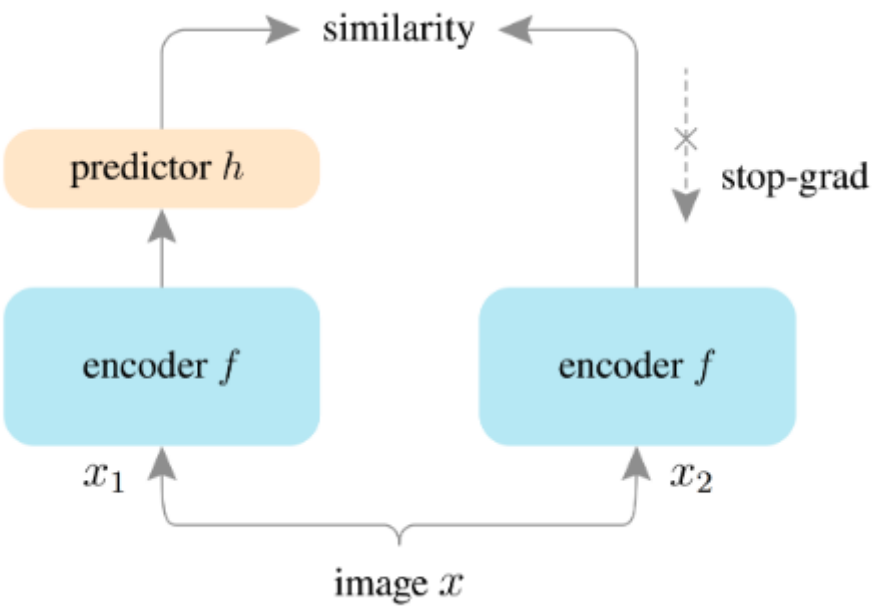
c、Sampling——利用插值方法，使得STN可微。

3、Siamese Network(孪生神经网络)

特征配准网络。

- **SimSiam--Exploring Simple Siamese Representation Learning**

```
https://github.com/facebookresearch/simsiam
```



该结构是何大神提出的用来解决“**崩溃解**”问题，同时他使用了**对称损失函数**和**stop-grad**来提高模型性能。

> 所谓崩塌，也就是模型为了偷懒，无论什么图片都会输出同样表示，这样结果 loss 很小，然后却没学到任何东西。

伪代码：

Algorithm 1 SimSiam Pseudocode, PyTorch-like

```
# f: backbone + projection mlp
# h: prediction mlp

for x in loader: # load a minibatch x with n samples
   x1, x2 = aug(x), aug(x) # random augmentation
   z1, z2 = f(x1), f(x2) # projections, n-by-d
   p1, p2 = h(z1), h(z2) # predictions, n-by-d

   L = D(p1, z2)/2 + D(p2, z1)/2 # loss

   L.backward() # back-propagate
   update(f, h) # SGD update

def D(p, z): # negative cosine similarity
   z = z.detach() # stop gradient

   p = normalize(p, dim=1) # l2-normalize
   z = normalize(z, dim=1) # l2-normalize
   return -(p*z).sum(dim=1).mean()
```

> 在y.backward()时，如果y是标量，则不需要为backward()传入任何参数；否则，需要传入一个与y同形的Tensor。
>
> 如果不想要被继续追踪，可以调用.detach()将其从追踪记录中分离出来，这样就可以防止将来的计算被追踪，这样梯度就传不过去了。还可以用with torch.no_grad()将不想被追踪的操作代码块包裹起来，这种方法在评估模型的时候很常用，因为在评估模型时，我们并不需要计算可训练参数（requires_grad=True）的梯度。

- 图像配准

4、正态分布估计

孪生网络的两个分支完全相同，因此仅使用一个分支特征进行正态分布估计。在获得配准特征后，使用基于统计的估计器来估计目标类别特征的正态分布，该估计器使用多元高斯分布来获得正态类的概率表示。

- **模型预测其他类别的异常**（即训练得到的模型不需要微调等操作直接可以拿来使用预测出新的类别的高斯分布）：

a、需要把新的类别的normal数据集输入训练好的模型得到特征值，计算均值和方差得到新类别的高斯分布；

b、需要输入新类别的测试数据图片，然后利用一定的度量手段来度量该数据输入进网络得到特征图是否满足新类别的高斯分布。

- **模型特征值**：

论文中写的是合并三个STN模块输出的特征得到的特征值。因为该特征需要保留原图像的空间信息，所以STN模块输出的特征图需要进行一个**逆仿射变换**，这样便可以检测出异常像素所在位置。

- **如何度量**：

*fij*是测试图片通过训练好模型得到的特征，均值和方差是测试过程计算得到的均值和方差。经过计算便可以得到对应测试图片每一个像素点的异常分数值，这张地图上的高分表示异常区域。**整个图像的最终异常得分是异常图最大值。**

$$\mathcal{M}\left(f_{ij}\right)=\sqrt{\left(f_{ij}-\mu_{ij}\right)^{T}\Sigma_{ij}^{-1}\left(f_{ij}-\mu_{ij}\right)}. \tag{5}$$

在推理过程中，超出正态分布的测试样本被认为是异常的。对于T中的每个测试图像试验，我们使用马氏距离M(fij)，给位置(i、j)一个异常分数。

- 马氏距离：

马氏距离（Mahalanobis Distance）是一种用于测量多维空间中点与点之间距离的方法。与欧氏距离不同，马氏距离考虑了各个维度之间的相关性。这个距离常用于多变量统计分析和模式识别领域。

给定一个p维向量 $\mathbf{x}$ 和一个p维向量 $\mathbf{y}$，它们之间的马氏距离 $D_M(\mathbf{x},\mathbf{y})$ 的公式为：

$$D_M(\mathbf{x},\mathbf{y})=\sqrt{(\mathbf{x}-\mathbf{y})^T\mathbf{S}^{-1}(\mathbf{x}-\mathbf{y})}$$

其中，$\mathbf{S}$ 是协方差矩阵的逆矩阵。协方差矩阵用于描述数据的变化和关系，而矩阵的逆则用于考虑数据的分布形状。

5、损失函数：

使用L2范数的特征级配准损失，而不是逐像素配准图像，这可以被认为是像素级配准约束的松弛版本，以获得更好的鲁棒性。采用负余弦相似度损失。

负余弦相似度：*cos*0=1，两个向量的余弦值越接近1，说明两个向量的夹角越小，两个向量离得越近越相似。

$$\mathcal{D}(p_a, z_b) = -\frac{p_a}{||p_a||_2} \cdot \frac{z_b}{||z_b||_2}, \tag{2}$$

$$\mathcal{L} = \frac{1}{2}(\mathcal{D}(p_a, z_b) + \mathcal{D}(p_b, z_a)). \tag{3}$$

损失函数如下，该损失函数使用的和SiaSim的损失函数一样，作者解释的是从特征角度出发而不是从像素角度出发。p和z是提取出来的特征。

包含空间信息的特征有利于AD任务，它需要提供异常得分图作为预测结果。与SimSiam [5]将输入定义为一幅图像的两个增强，并最大化它们的相似性以增强模型表示不同，所提出的特征配准利用两个不同的图像作为输入，并最大限度地提高特征之间的相似性来学习配准。

6、Pretext tasks

"Pretext tasks" 指在深度学习领域中，为了帮助模型学习有用的表示或特征，而在模型最终的目标任务之前，设计的一系列辅助任务。这些预训练任务的目标是通过大规模、多样的数据学习通用的特征表示，使得模型在后续的具体任务上更容易、更有效地进行迁移学习。

> 1.这种训练不是我们本身的训练任务，并不是本身这次训练需要做的事情。
> 2.虽然不是这次训练需要做的事情，但是他可以促进我们的训练，达到更好的效果。

7、高斯滤波器：

> 高斯滤波器是一种线性滤波器，能够有效的抑制噪声，平滑图像。其作用原理和均值滤波器类似，都是取滤波器窗口内的像素的均值作为输出。其窗口模板的系数和均值滤波器不同，均值滤波器的模板系数都是相同的为1；而高斯滤波器的模板系数，则随着距离模板中心的增大而系数减小。所以，高斯滤波器相比于均值滤波器对图像个模糊程度较小。
> 高斯滤波器可以看为一个固定卷积核的卷积操作。

二、Code

```
https://github.com/mediabrain-sjtu/regad
```

EX1:

```
parser = argparse.ArgumentParser(description='Registration based Few-Shot Anomaly Detection')
parser.add_argument('--obj', type=str, default='bottle')
parser.add_argument('--data_type', type=str, default='mvtec')
parser.add_argument('--data_path', type=str, default='/root/dataset/public/mvtec_anomaly_detection/')
parser.add_argument('--epochs', type=int, default=50, help='maximum training epochs')
parser.add_argument('--batch_size', type=int, default=32)
parser.add_argument('--img_size', type=int, default=224)
parser.add_argument('--lr', type=float, default=0.0001, help='learning rate of others in SGD')
parser.add_argument('--momentum', type=float, default=0.9, help='momentum of SGD')
parser.add_argument('--seed', type=int, default=668, help='manual seed')
parser.add_argument('--shot', type=int, default=2, help='shot count')
parser.add_argument('--inferences', type=int, default=10, help='number of rounds per inference')
parser.add_argument('--stn_mode', type=str, default='rotation_scale',
                    help='[affine, translation, rotation, scale, shear, rotation_scale, translation_scale, rotation_translation, rotation_translation_scale]')
args = parser.parse_args()
args.input_channel = 3
```

```
pip install kornia
python train_wood.py
```

- Result:

```
Img-level AUC: 0.9967460317460318
Pixel-level AUC: 0.9849798816631598
Test Epoch(img, pixel): 49 (0.991429, 0.985066) best: (0.997, 0.985)
100%|
|
219/219 [01:33<00:00,  2.35it/s]
Train Epoch: 50 Total_Loss: -0.616937
```