# EfficientAD: Accurate Visual Anomaly Detection at Millisecond-Level Latencies （毫秒级延迟的准确视觉异常检测）

# 一、Paper Reading

## First:

### 1、Abstract:

检测图像中的异常是一项重要任务，尤其是在实时计算机视觉应用中。在这项工作中，我们专注于计算效率，并提出了一种轻量级特征提取器，可以在现代 GPU 上在不到一毫秒的时间内处理图像。然后，我们使用学生-教师的方法来检测异常特征。我们训练学生网络来预测正常（即无异常训练图像）的提取特征。由于学生无法预测自己的特征，因此可以在测试时检测到异常情况。我们提出了一种训练损失，阻止学生模仿教师特征提取器超出正常图像的范围。它使我们能够大幅降低学生-教师模型的计算成本，同时改进异常特征的检测。我们还解决了具有挑战性的逻辑异常的检测，这些异常涉及正常局部特征的无效组合，例如对象的错误排序。我们通过有效地结合全局分析图像的自动编码器来检测这些异常。我们在来自三个工业异常检测数据集集合的 32 个数据集上评估我们的方法（称为 EfficientAD）。 EfficientAD 为异常检测和定位设定了新标准。它的延迟时间为 2 毫秒，吞吐量为每秒 600 张图像，可以快速处理异常情况。再加上其低错误率，这使其成为现实应用的经济解决方案，并为未来研究奠定了富有成果的基础.

### 2、论文中的图片、表格

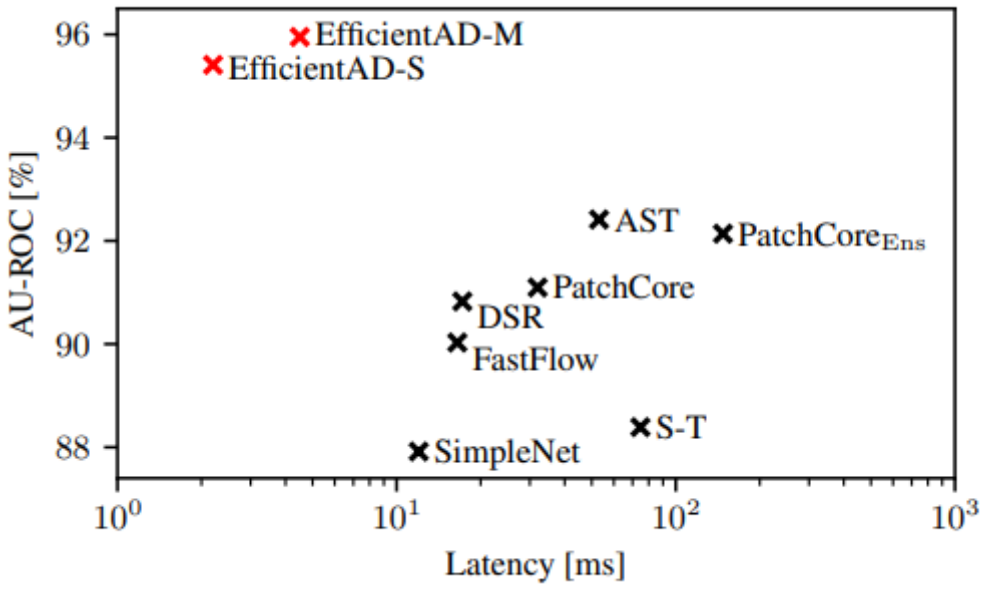


**图 1. NVIDIA RTX A6000 GPU 上的异常检测性能与每张图像的延迟。每个 AU-ROC 值是 MVTec AD [7, 9]、VisA [74] 和 MVTec LOCO [8] 数据集集合上图像级检测 AU-ROC 值的平均值**

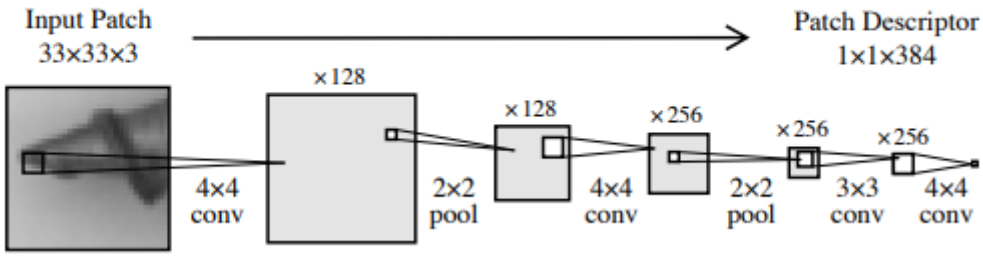


**图 2. EfficientAD-S 的补丁描述网络 (PDN) 架构。以完全卷积的方式将其应用于图像可以在一次前向传递中产生所有特征。**

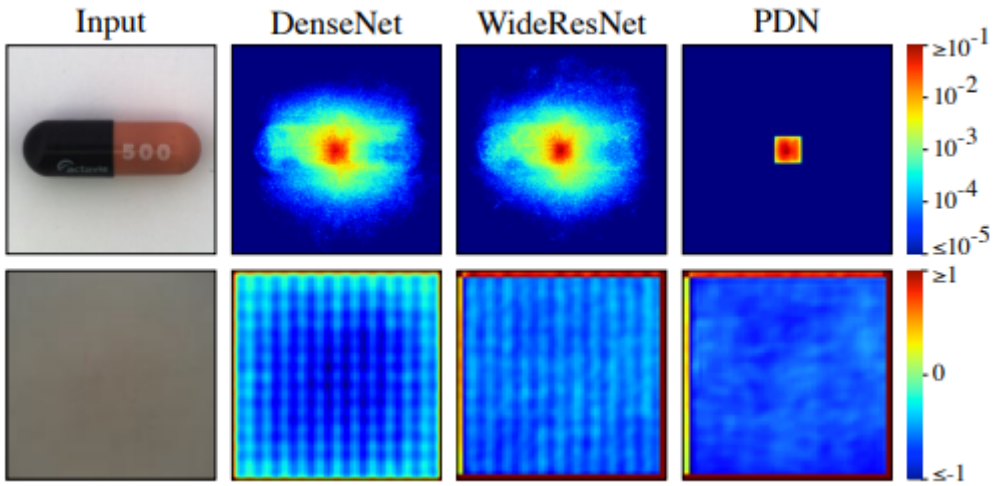


**图 3. 上行：位于输出中心的单个特征向量相对于每个输入像素的绝对梯度，在输入和输出通道上取平均值。下排：从 ImageNet [55] 中随机选择的 1000 张图像中第一个输出通道的平均特征图。这些图像的平均值显示在左侧。 DenseNet [25] 和 WideResNet 的特征图表现出很强的伪影。**

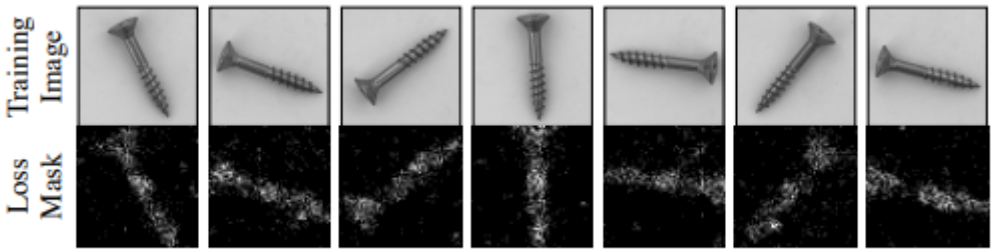


**图 4. 训练期间由硬特征损失生成的随机选取的损失掩码。掩模像素的亮度指示相应特征向量的多少维被选择用于反向传播。学生网络已经在背景上很好地模仿了老师，因此专注于学习不同旋转螺钉的特征**

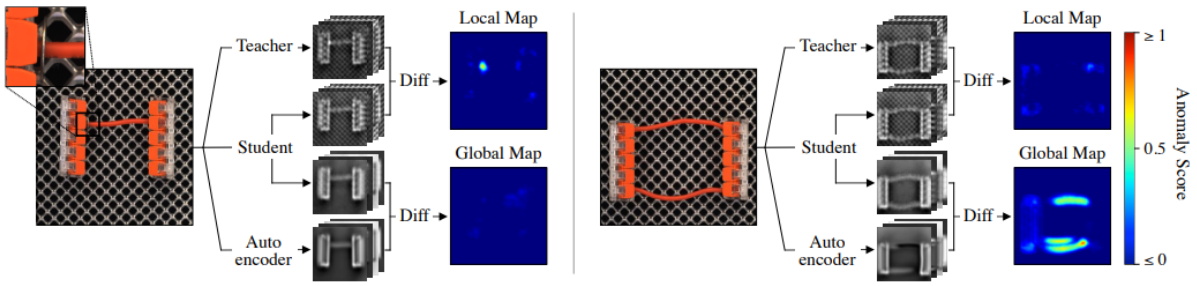

**图 5. EfficientAD 应用于 MVTec LOCO 的两个测试图像。正常输入图像包含连接任意高度的两个拼接连接器的水平电缆。左侧的异常现象是电缆末端有一个小金属垫圈形式的异物。它在局部异常图中可见，因为学生和教师的输出不同。右侧的逻辑异常是存在第二根电缆。自动编码器无法在教师的特征空间中重建右侧的两条电缆。除了老师的输出之外，学生还预测自动编码器的输出。由于其感受野仅限于图像的小块，因此它不受附加红色电缆存在的影响。这会导致自动编码器和学生的输出不同。 "Diff"是指计算两个输出特征图集合之间的逐元素平方差，并计算其跨特征图的平均值。为了获得像素异常分数，使用双线性插值调整异常图的大小以匹配输入图像。**

| Method | Detect. AU-ROC | Segment. AU-PRO | Latency [ms] | Throughput [img / s] |
|---|---|---|---|---|
| GCAD | 85.4 | 88.0 | 11 | 121 |
| SimpleNet | 87.9 | 74.4 | 12 | 194 |
| S–T | 88.4 | 89.7 | 75 | 16 |
| FastFlow | 90.0 | 86.5 | 17 | 120 |
| DSR | 90.8 | 78.6 | 17 | 104 |
| PatchCore | 91.1 | 80.9 | 32 | 76 |
| $\text{PatchCore}_{\text{Ens}}$ | 92.1 | 80.7 | 148 | 13 |
| AST | 92.4 | 77.2 | 53 | 41 |
| EfficientAD-S | 95.4 (± 0.06) | 92.5 (± 0.05) | **2.2** (± 0.01) | **614** (± 2) |
| EfficientAD-M | **96.0** (± 0.09) | **93.3** (± 0.04) | 4.5 (± 0.01) | 269 (± 1) |

**表 1. 异常检测和异常定位性能与延迟和吞吐量的比较。每个 AU-ROC 和 AU-PRO 百分比分别是 MVTec AD、VisA 和 MVTec LOCO 上平均 AU-ROC 和平均 AU-PRO 的平均值。对于 EfficientAD，我们报告五次运行的平均值和标准差。**

| Method | MAD | LOCO | VisA | Mean | LOCO Logic. | LOCO Struct. |
|---|---|---|---|---|---|---|
| GCAD | 89.1 | 83.3 | 83.7 | 85.4 | 83.9 | 82.7 |
| SimpleNet | 98.2 | 77.6 | 87.9 | 87.9 | 71.5 | 83.7 |
| S–T | 93.2 | 77.4 | 94.6 | 88.4 | 66.5 | 88.3 |
| FastFlow | 96.9 | 79.2 | 93.9 | 90.0 | 75.5 | 82.9 |
| DSR | 98.1 | 82.6 | 91.8 | 90.8 | 75.0 | 90.2 |
| PatchCore | 98.7 | 80.3 | 94.3 | 91.1 | 75.8 | 84.8 |
| $\text{PatchCore}_{\text{Ens}}$ | **99.3** | 79.4 | 97.7 | 92.1 | 71.0 | 87.7 |
| AST | 98.9 | 83.4 | 94.9 | 92.4 | 79.7 | 87.1 |
| EfficientAD-S | 98.8 | 90.0 | 97.5 | 95.4 | 85.8 | 94.1 |
| EfficientAD-M | 99.1 | **90.7** | **98.1** | **96.0** | **86.8** | **94.7** |

**表 2. 每个数据集集合的平均异常检测 AU-ROC 百分比（左）以及 MVTec LOCO 的逻辑和结构异常（右）。对于 EfficientAD，我们报告五次运行的平均值。仅在 MVTec AD (MAD) 上执行方法开发很容易导致设计选择过度拟合剩余的少数错误分类测试图像。**

| $a$ (for $q_a$) | 0.5 | 0.8 | **0.9** | 0.95 | 0.98 | 0.99 |
|---|---|---|---|---|---|---|
| AU-ROC | 95.9 | 95.9 | 96.0 | 95.9 | 95.9 | 95.8 |
| $b$ (for $q_b$) | 0.95 | 0.98 | 0.99 | **0.995** | 0.998 | 0.999 |
| AU-ROC | 95.8 | 95.9 | 96.0 | 96.0 | 95.9 | 95.9 |
| $p_{\text{hard}}$ | 0 | 0.9 | 0.99 | **0.999** | 0.9999 | 0.99999 |
| AU-ROC | 94.9 | 94.9 | 95.7 | 96.0 | 95.8 | 95.7 |

**表 3. 改变分位数位置时，MVTec AD、VisA 和 MVTec LOCO 上 EfficientAD-M 的平均异常检测 AU-ROC。这是基于分位数的图归一化和挖掘因子 phard 的两个采样点 a 和 b。将 phard 设置为零会禁用建议的硬特征丢失。我们实验中使用的默认值以粗体突出显示。**

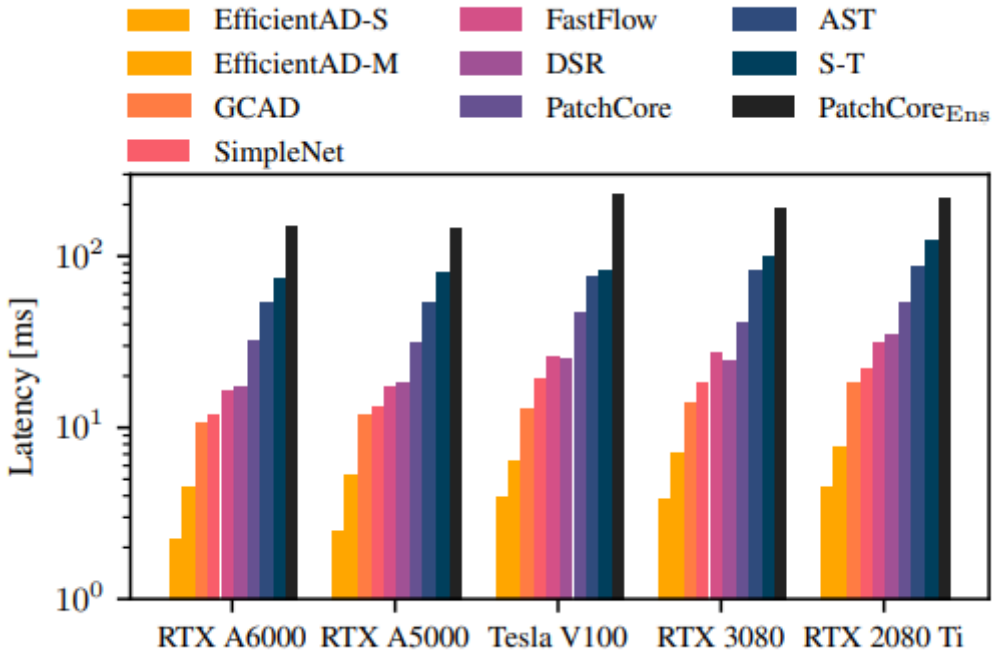


**图6.每GPU等待时间。在每个GPU上，方法的排名相同，除了DSR比FastFlow略快的情况下。**

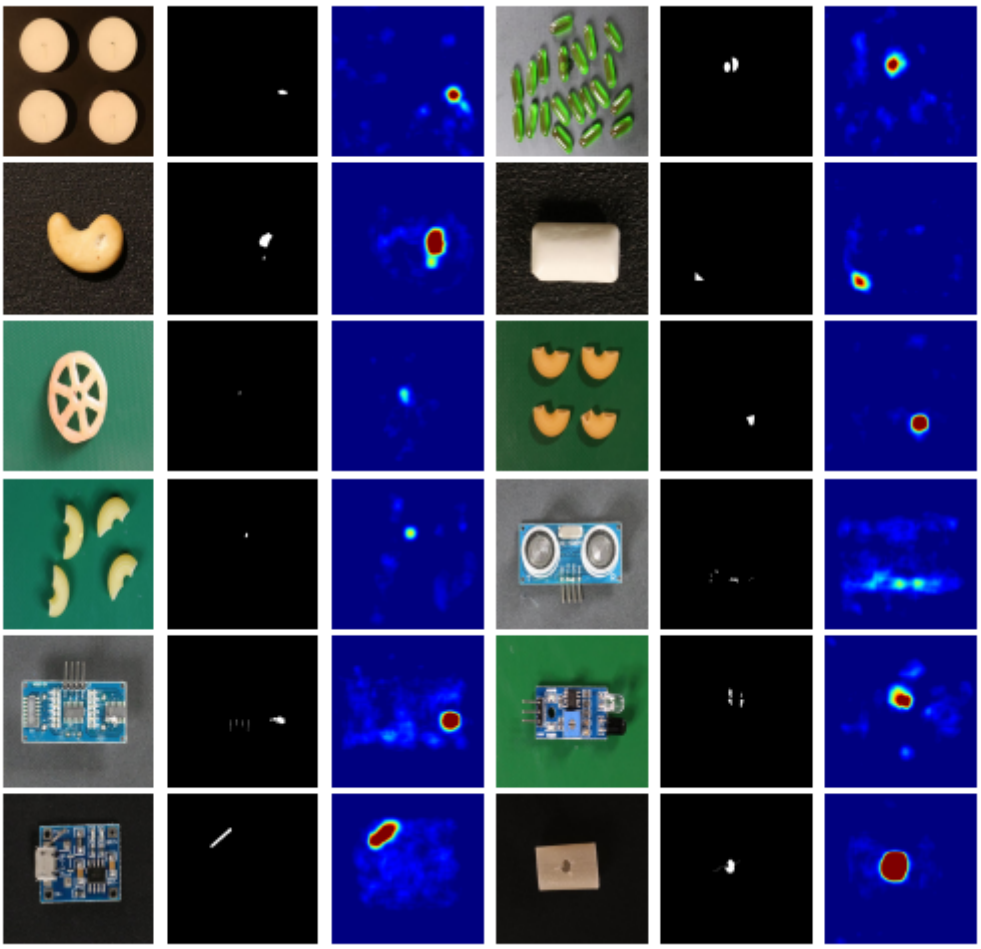

**图 7. EfficientAD 在 VisA 上的非精挑细选的定性结果。对于 12 个场景中的每一个，我们都展示了随机采样的缺陷图像、地面实况分割掩模以及 EfficientAD-M 生成的异常图。**

| | Detection AU-ROC | Diff. | Latency [ms] |
|---|---|---|---|
| PDN | 93.2 | | 2.2 |
| ↪ with map normalization | 94.0 | + 0.8 | 2.2 |
| ↪ with hard feature loss | 95.0 | + 1.0 | 2.2 |
| ↪ with pretraining penalty | 95.4 | + 0.4 | 2.2 |
| EfficientAD-S | 95.4 | | 2.2 |
| EfficientAD-M | 96.0 | + 0.6 | 4.5 |

**表 4. 累积消融研究，其中技术逐渐组合以形成 EfficientAD。每个 AU-ROC 百分比是 MVTec AD、VisA 和 MVTec LOCO 上平均 AU-ROC 的平均值。**

| | Detection AU-ROC | Diff. | Latency [ms] |
|---|---|---|---|
| EfficientAD-S | 95.4 | | 2.2 |
| Without map normalization | 94.7 | - 0.7 | 2.2 |
| Without hard feature loss | 94.7 | - 0.7 | 2.2 |
| Without pretraining penalty | 95.0 | - 0.4 | 2.2 |

**表 5. 孤立的消融研究，其中的技术是从 EfficientAD-S 中单独删除的.**

## 3、结论

在本文中，我们介绍了EfficientAD，一种具有强大的异常检测性能和高计算效率的方法。它为结构异常和逻辑异常的检测设定了新标准。 EfficientAD-S 和 EfficientAD-M 在异常检测和定位方面都远远优于其他方法。与第二好的方法 AST 相比，EfficientAD-S 延迟降低了 24 倍，吞吐量提高了 15 倍。其低延迟、高吞吐量和高检测率使其适合实际应用。对于未来的异常检测研究，EfficientAD 是重要的基线和富有成果的基础。例如，其高效的补丁描述网络也可以用作其他异常检测方法中的特征提取器，以减少其延迟。

局限性。学生-教师模型和自动编码器旨在检测不同类型的异常。自动编码器检测逻辑异常，而学生-教师模型检测粗粒度和细粒度的结构异常。然而，细粒度的逻辑异常仍然是一个挑战——例如，螺丝太长了两毫米。为了检测这些，从业者必须使用传统的计量方法[62]。至于与其他最近的异常检测方法相比的局限性：与基于 kNN 的方法相比，我们的方法需要训练，特别是让自动编码器学习正常图像的逻辑约束。在我们的实验设置中，这需要二十分钟.

# Sencond:

教师网络: 可以提取所有特征。（imagenet数据训练）

学生网络：只能提取正常特征，提取不了异常特征。（正常图片训练）。提出了一个困难特征loss（loss_hard），防止学生网络提取到图像上目标范围以外的的特征，降低计算成本提高异常检测能力。

## 本文贡献

参考https://zhuanlan.zhihu.com/p/630829222

- 该方法在各种工业常见下进行了测试，效果均有提升，而且时延只有2ms。

- 该方法提出了一个简称PDN的高效的网络结构。

1、只包含四个卷积层高效的特征提取器，简称PDN；在网络输出中，每个输出含有33X33像素大小的感受野。

2、当前流行的分类架构通常在早期执行下采样，以减小特征图的大小，从而降低运行时间和内存需求。PDN通过在第一和第二个卷积层后使用分步平均池化层来实现这一点。

3、在训练过程中，PDN（Patch Description Network）是通过在ImageNet数据集上对WideResNet-101进行蒸馏（distillation）而来的，损失函数采用均方误差（MSE，Mean Squared Error）。蒸馏是一种迁移学习的技术，通常用于将一个复杂的模型的知识传递给一个较简单的模型。在这个背景下，WideResNet-101充当教师模型，而PDN则是学生模型。通过使用均方误差作为损失函数，PDN试图在其输出和教师模型的输出之间最小化平方差。

- 该方法提出了能高效训练教师-学生网络的困难特征loss。

1、在标准的S-T框架中，增加训练图像的数量可以提高学生在异常情况下模仿教师的能力，这会恶化异常检测性能。同时，有意减少训练图像的数量可能会抑制有关正常图像的重要信息。网络设计的的目标是向学生展示足够的数据，以便它可以在正常图像上充分模仿教师，同时避免对异常图像进行泛化。

2、该网络的设计将学生的损失限制在图像最相关的部分。这里提出了一个困难特征loss，只计算输出的特征中loss最大的部分进行反向传播。抑制了了背景区域的假正例。

```python
        d_hard = torch.quantile(distance_st, q=0.999) #使用索引操作distance_st[distance_st >= d_hard]，选取了距离大于或等于阈值的部分，计算了这些距离值的平均值，得到了困难特征损失 loss_hard。
        loss_hard = torch.mean(distance_st[distance_st >= d_hard])
```

- 该方法实现了一个基于自编码器的有效的逻辑缺陷检测

1、逻辑异常检测：提出了一个检测组合逻辑的结构。
将学生网络的输出通道数量加倍，并训练学生网络的另一半来预测自编码器的输出以及教师模型的输出；
与基于patch的学生相比，自编码器必须通过64个隐层的bottleneck对完整图像进行编码和解码。在具有逻辑异常的图像上，自编码器通常无法生成正确通过隐层在教师的特征空间中重建图像。
然而，在正常图像上，它的重建也存在缺陷，因为自编码器通常难以重建细粒度特征。
使用教师输出和自编码器重建之间的差异作为异常特征，将导致这些情况下的假正例（被错误地标记为正类别的负样本，即实际上是负类别的样本被模型预测为正类别。在异常检测的背景下，这可能表示正常情况下的样本被错误地识别为异常。）。

2、学生学习自编码器在正常图像上的系统重建误差，例如模糊的重建。同时，它不会学习异常情况下的重建误差，因为这些特征并不在训练集中。这使得自编码器输出和学生输出之间的差分非常适合计算异常特征图。类似于学生-教师对，异常特征图是两个输出之间的MSE，跨通道进行平均。

3、异常特征图归一化
由于这个方法包含了两个模块，局部和全局的异常检测模块。因此，需要将局部和全局异常图归一化到类似的比例尺上，才能按通道平均计算组合异常图。否则，一个模块中的噪声可能会对另一个模块中的检测准确性产生干扰。
为了估计正常图像中噪声的比例尺，该方法使用验证集图像，即训练集的未出现的图像，对于每种异常模块计算一个可能出现的最大最小值，用这个比例尺来对两个模块的输出进行对齐。

# 二、Code

```
https://github.com/nelson1425/EfficientAD

# Training and inference:
python efficientad.py --dataset mvtec_ad --subdataset bottle
```

## EX:

```
def get_argparse():
    parser = argparse.ArgumentParser()
    parser.add_argument('-d', '--dataset', default='mvtec_ad',
                        choices=['mvtec_ad', 'mvtec_loco'])
    parser.add_argument('-s', '--subdataset', default='bottle',
                        help='One of 15 sub-datasets of Mvtec AD or 5' +
                             'sub-datasets of Mvtec LOCO')
    parser.add_argument('-o', '--output_dir', default='output/1')
    parser.add_argument('-m', '--model_size', default='small',
                        choices=['small', 'medium'])
    parser.add_argument('-w', '--weights', default='models/teacher_small.pth')
    parser.add_argument('-i', '--imagenet_train_path',
                        default='none',
                        help='Set to "none" to disable ImageNet' +
                             'pretraining penalty. Or see README.md to' +
                             'download ImageNet and set to ImageNet path')
    parser.add_argument('-a', '--mvtec_ad_path',
                        default='/root/dataset/public/mvtec_anomaly_detection',
                        help='Downloaded Mvtec AD dataset')
    parser.add_argument('-b', '--mvtec_loco_path',
                        default='./mvtec_loco_anomaly_detection',
                        help='Downloaded Mvtec LOCO dataset')
    parser.add_argument('-t', '--train_steps', type=int, default=70000)
    return parser.parse_args()
```

- Result:

```
Current loss: 0.4247  : 100%|
████████████████████████████████████████████████████████████| 70000/70000
[47:33<00:00, 24.53it/s]
Final map normalization: 100%|
████████████████████████████████████████████████████| 21/21 [00:00<00:00,
26.12it/s]
Final inference: 100%|████████████████████████████████████████████████████|
83/83 [00:03<00:00, 21.55it/s]
Final image auc: 100.0000
```