TranAD (2022)

一、解决的问题

1. **长时序依赖捕捉**：相比LSTM等递归模型，Transformer的并行化注意力机制能高效捕捉长期时序趋势，避免局部窗口限制（如USAD、MTAD-GAT），适合处理高维、长序列数据。
2. **训练效率与稳定性**：Transformer的并行化训练结合进化损失函数，使训练时间较基线（如LSTM-NDT、GDN）减少75%-99%，同时避免对抗训练中的不稳定性。
3. **异常检测与诊断一体化**：通过维度级异常分数计算（s_i）和POT动态阈值，既能判断是否异常（$y=\vee y_i$），又能定位异常来源维度，解决传统模型仅检测不诊断的局限。

二、创新点

1. **两阶段对抗训练**：

   第一阶段初步重构输入窗口，生成聚焦分数（重构误差）；第二阶段基于聚焦分数强化对异常区域的关注，通过对抗性学习放大误差，解决传统模型对微小异常不敏感的问题。
2. **进化损失函数**：

   随训练轮次（n）动态平衡重构损失与对抗损失（$L_1=\epsilon^{-n}\|O_1-W\|_2+(1-\epsilon^{-n})\|\hat{O}_2-W\|_2$），初期侧重重构以保证稳定，后期增强对抗性以提升泛化能力。
3. **元学习（MAML）**：

   通过快速调整模型权重（$\theta'=\theta-\alpha\nabla\theta L(f(\theta))$），使模型在有限数据下仍能高效学习时序趋势，解决联邦学习等场景中数据不足的问题。

三、模型架构

TranAD基于Transformer架构构建，核心是编码器-解码器结构，结合两阶段处理流程实现多元时序异常检测与诊断，模型结构图如下：

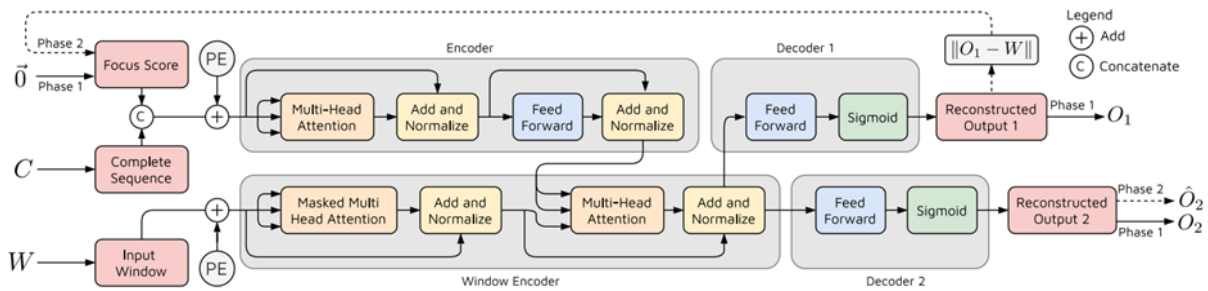

Figure 1: The TranAD Model.

具体架构介绍：

1. **编码器模块**：包含两个编码器——完整序列编码器和窗口编码器。
   - 完整序列编码器接收截至当前时间戳的完整序列（C）和初始聚焦分数（F），通过多头自注意力机制（含位置编码）捕捉全局时序趋势，输出编码后的全局特征（I_1^2）。
   - 窗口编码器接收局部滑动窗口（W）代码设置窗口大小为10，即10个时间点的测量，通过掩码多头自注意力（屏蔽未来时间步信息）结合全局特征（I_1^2），生成局部窗口的编码表示（I_2^3），兼顾局部上

下文与全局趋势。

$$W_t = \{x_{t-K+1}, \ldots, x_t\}$$

2. **解码器模块**：包含两个解码器，基于窗口编码器的输出（`I₂³`）生成重构结果。
   - 第一阶段解码器输出初步重构（`O₁`、`O₂`），计算`O₁`和`W`重构误差作为"注意力分数"。
   - 第二阶段解码器利用注意力分数调整注意力权重，生成强化重构（`Ô₂`），放大异常区域的误差。

   核心代码逻辑如下

```
    def forward(self, src, tgt): # tgt:(1,128,38)
        # Phase 1 - Without anomaly scores
        c = torch.zeros_like(src)  # c: (10,128,38) 初始条件c为零（无先验）
        # 解码器1输出O₁  # x1:(1,128,38)
        x1 = self.fcn(self.transformer_decoder1(*self.encode(src, c, tgt)))
        # Phase 2 - With anomaly scores
        c = (x1 - src) ** 2 # c: (10,128,38)c更新为第一阶段重构误差
        # 解码器2输出Ô₂ # x2:(1,128,38)
        x2 = self.fcn(self.transformer_decoder2(*self.encode(src, c, tgt)))
        return x1, x2
```

   对抗性体现：两个解码器分别输出`O₁`和`Ô₂`（对应代码中的`x1`和`x2`），后续通过损失函数（如文档中的`L1`和`L2`）形成对抗 —— 解码器 1 最小化误差，解码器 2 放大误差。

$$\min_{\text{Decoder1}} \max_{\text{Decoder2}} \|\hat{O}_2 - W\|_2$$

$$L_1 = \epsilon^{-n}\|O_1 - W\|_2 + (1-\epsilon^{-n})\|\hat{O}_2 - W\|_2$$

$$L_2 = \epsilon^{-n}\|O_2 - W\|_2 - (1-\epsilon^{-n})\|\hat{O}_2 - W\|_2$$

   代码通过一个`l1`融合了文档中提及的`L1`和`L2`。

```
#  损失计算：动态权重组合两阶段损失（L₁=ϵ⁻ⁿ‖O₁-W‖₂+(1-ϵ⁻ⁿ)‖Ô₂-W‖₂）  z[0]-> O₁  z[1]-> Ô₂
    l1 = l(z, elem) if not isinstance(z, tuple) else (1 / n) * l(z[0], elem) + (1 - 1 /
n) * l(z[1], elem) # l1:{1,128,38}
```

3. **核心机制**：通过多头自注意力（`MultiHeadAtt`）捕捉特征间依赖，位置编码保留时序信息，掩码机制确保训练时不泄露未来数据。

我的思考：

DTAAD(2024)