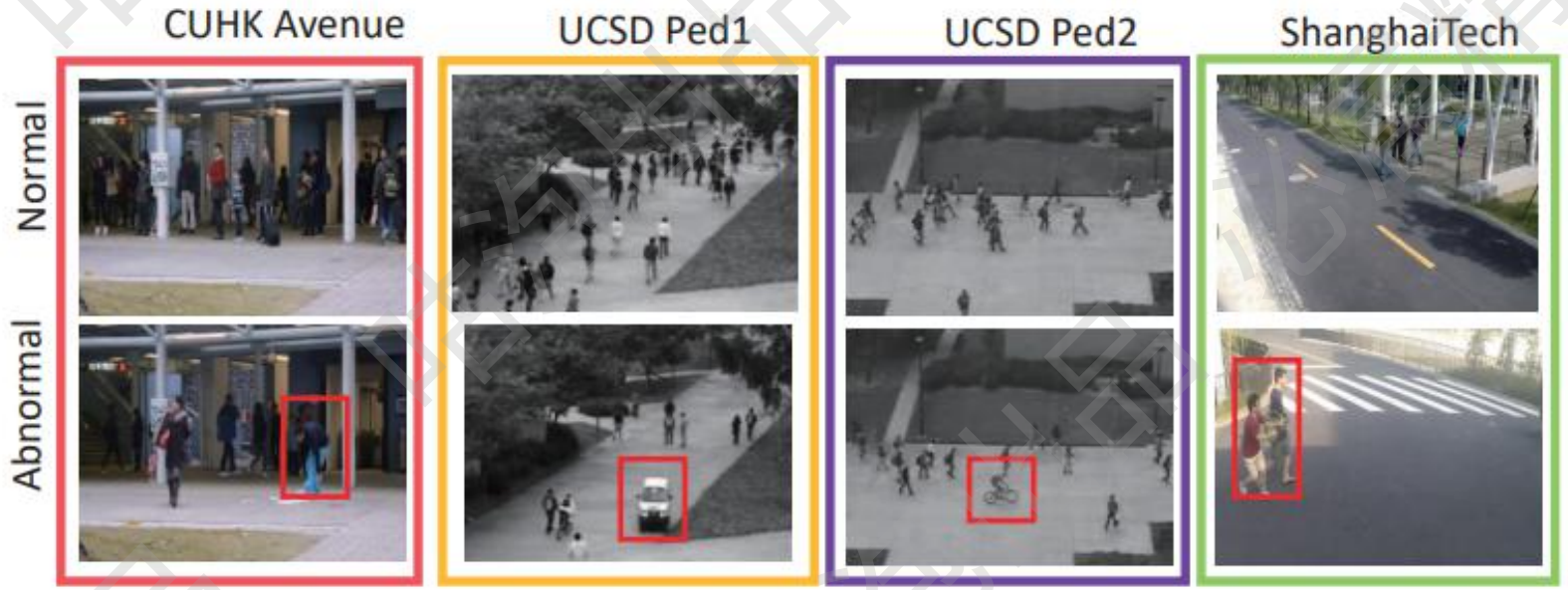
视频异常检测

- 什么是异常呢?

  - 除了正常的，其他都是异常（并不对异常做明确规定）

  - 校园里突然开出一辆车，马路上窜出个野猪（但凡不常规的都是异常）

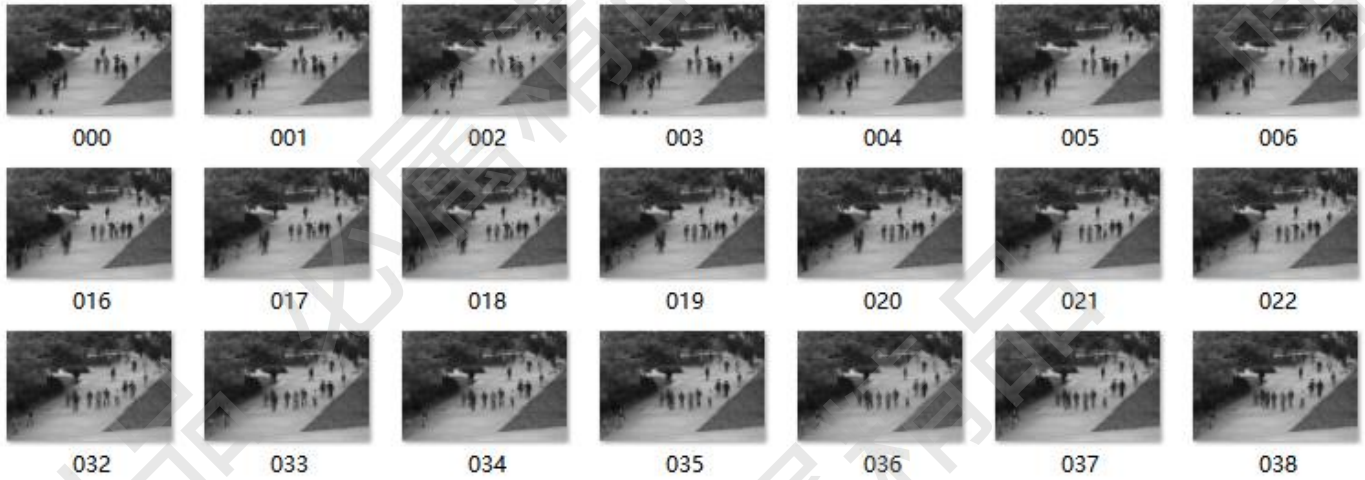


异常检测

- 数据集介绍

  - Avenue，UCSD Ped等

  - 异常主要是跑，车等行为



Video Description

The training videos capture normal situations. Testing videos include both normal and abnormal events. Three abnormal samples are shown as follows.

Strange action	Wrong direction	Abnormal object
		

异常检测

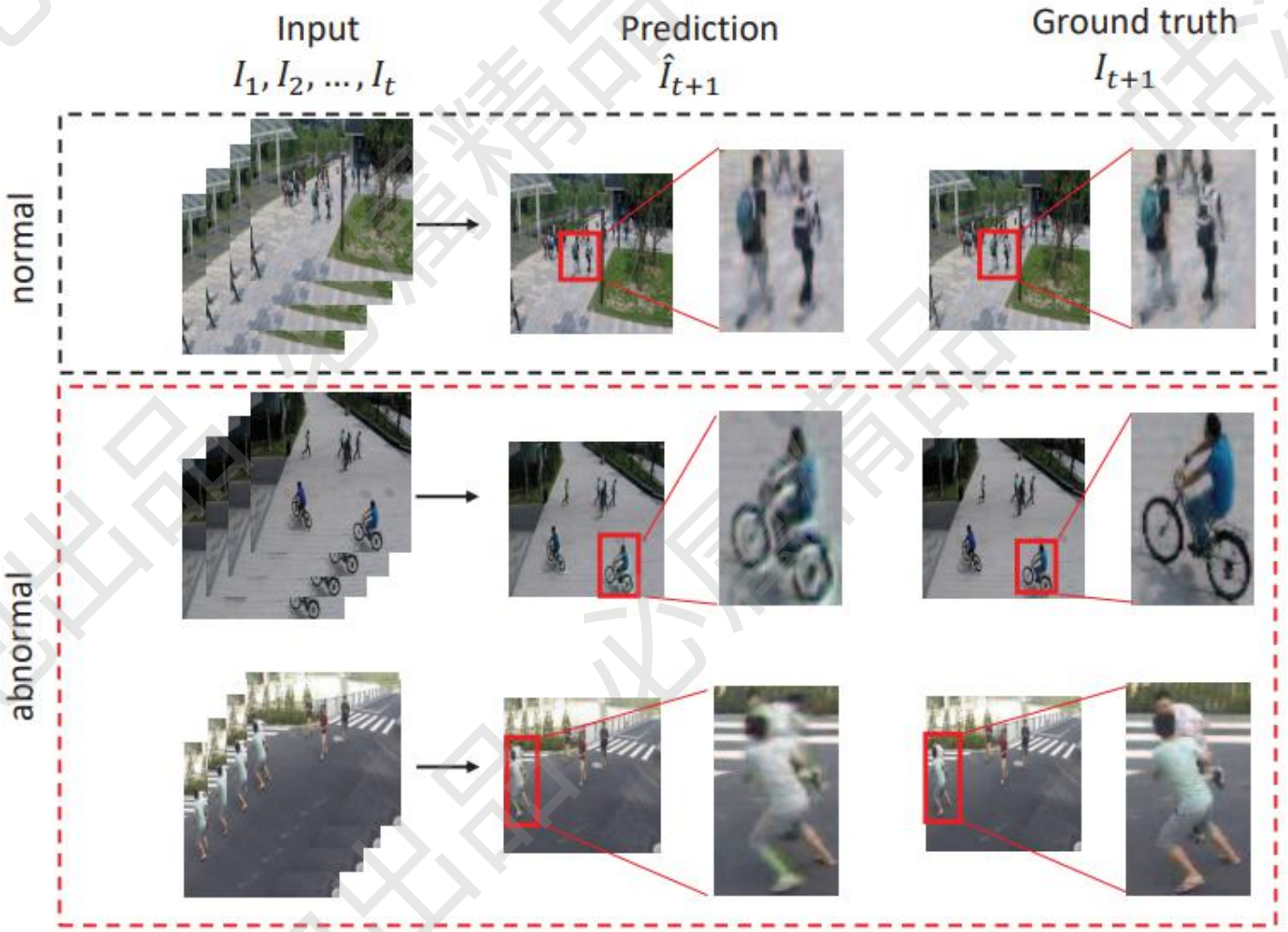
- 数据集的特点
  - 通常数据集中都是正样本，负样本（异常）很少
  - 基本都是固定位置的，背景相对不变，前景（人）变化差异较大
  - 标注信息很少，导致了这行很多任务都得靠无监督方法来做
  - 跨域任务很难，场景基本定死了，拓展比较费劲

异常检测

- 基本套路
  - 无监督方法多一些，训练样本都是正常情况
  - 将视频数据切帧，构建成序列（x1,x2,x3,x4），预测下一帧(x5)
  - 如果下一帧是正常情况，那应该预测的不错（训练数据都是正常的）
  - 如果下一帧是异常情况，那它应该预测的不咋地（没见到，整不出来）

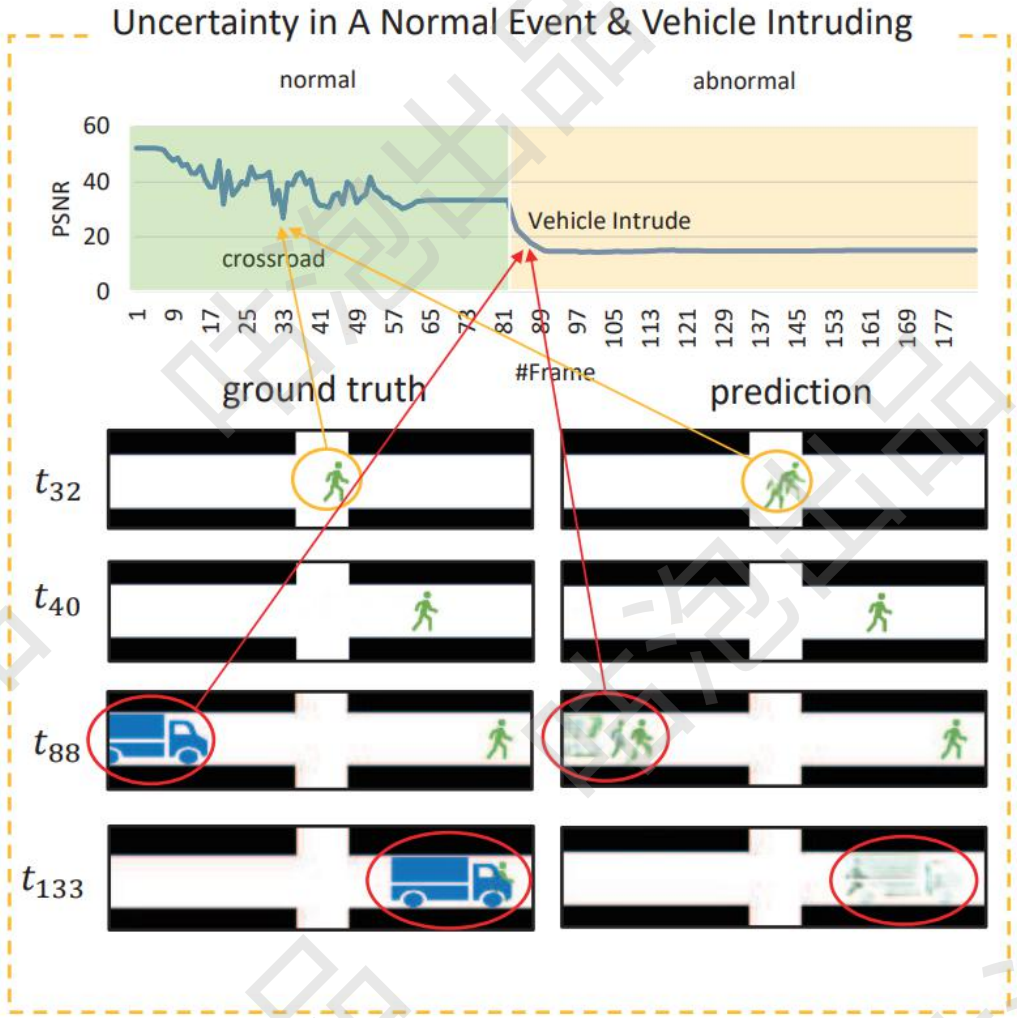
异常检测

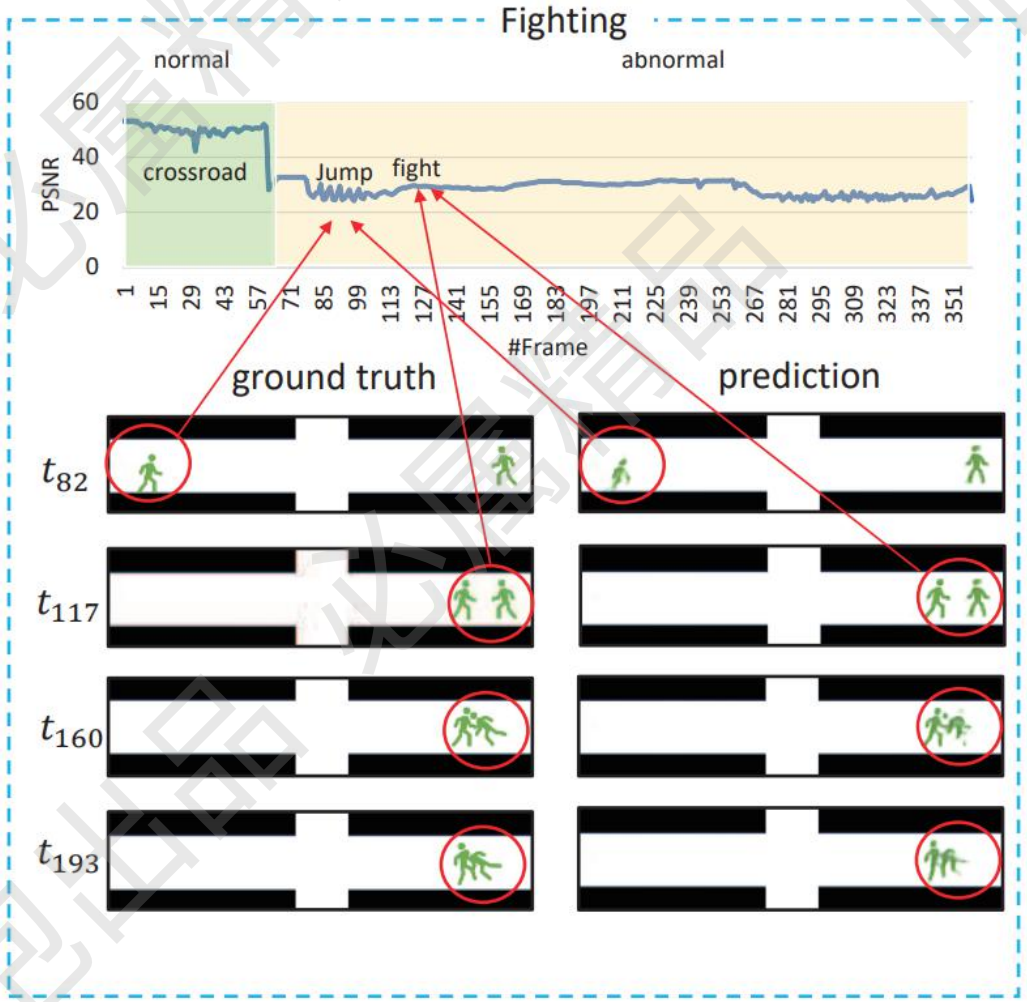
- 基本思想
  - 输入的是一个序列
  - 预测下一帧的图像
  - 正常的做的和GT很像
  - 异常的做的就开始模糊



异常检测

- 通过预测和GT的差异来描述异常（PSNR，信噪比越高表示越正常）

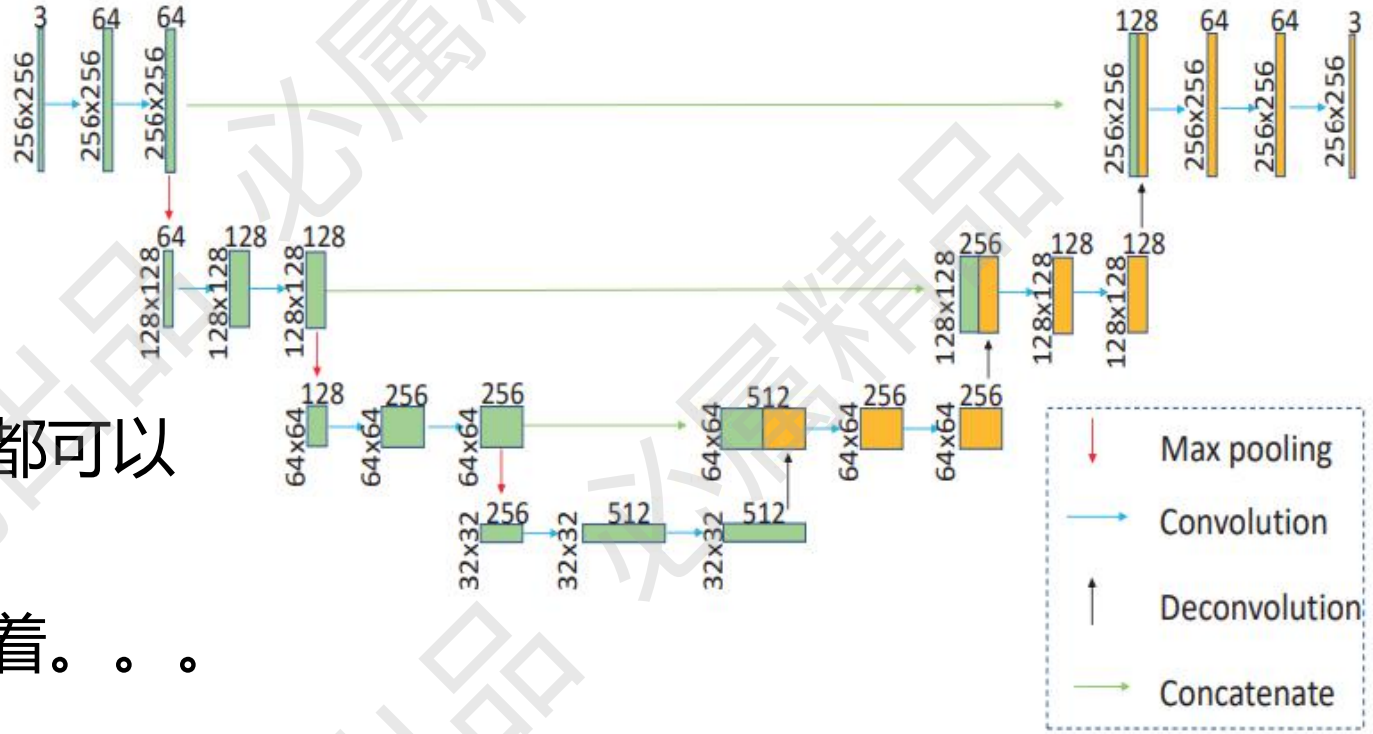




异常检测

- 如何预测呢？

  - 其实就是生成下一帧图像
  - 基于GAN的，AE的都可以
  - 融入注意机制的，各种模块都可以
  - 但是，有一个问题一直困扰着。。。



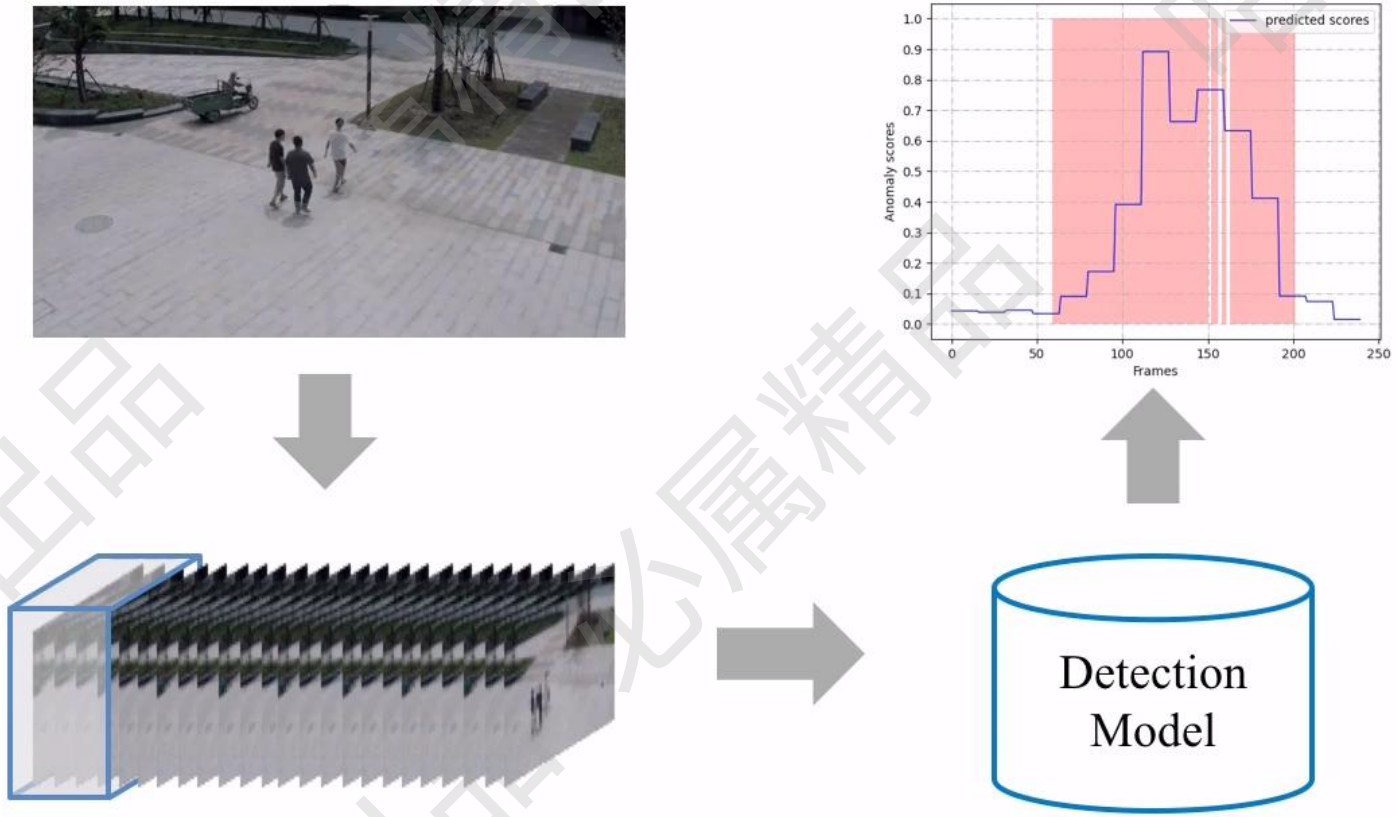
异常检测

- 最大的问题

  - 1.无论是CNN还是transformer等现在主流模块都能力太强了！
  - 太强就是，即便是异常的行为，也能预测的非常好（泛化能力太强）
  - 2.训练数据与场景太固定，如果换一个场景该怎么快速融入进去呢？
  - meta-learn可以来试一试，这人场景再好不过了，快速融入新圈子

异常检测

- 俗称VAD
- Video Anomaly Detection
- 针对视频进行异常检测
- 通过异常分数来判定
- 训练数据通常仅正样本



MetaLearning

- 主要解决啥问题呢？
  - 干过深度学习的同学肯定都知道，模型初始化的权重参数尤为重要
  - 各种超参数对都有结果影响，这肯定也得管管的，但是很难调
  - 数量少的时候经常训练不好，家里没那个条件啊，这时候咋办
  - 模型容易钻牛角尖，陷进一个任务就出不来了，如何提高通用性

MetaLearning

- 如果你是大老板，今天你要面试员工了
  - A：仅在一个领域特别厉害，但是太专精了，他很难适应其他领域
  - B：他在某一个领域一般般，但是学习能力特别强，能快速融入其他领域
  - A翻译过来就是有点过拟合了，因为训练数据都是那嘎达的
  - B就是我们希望得到的，可以快速转换到其他应用场景的一组权重参数

MetaLearning

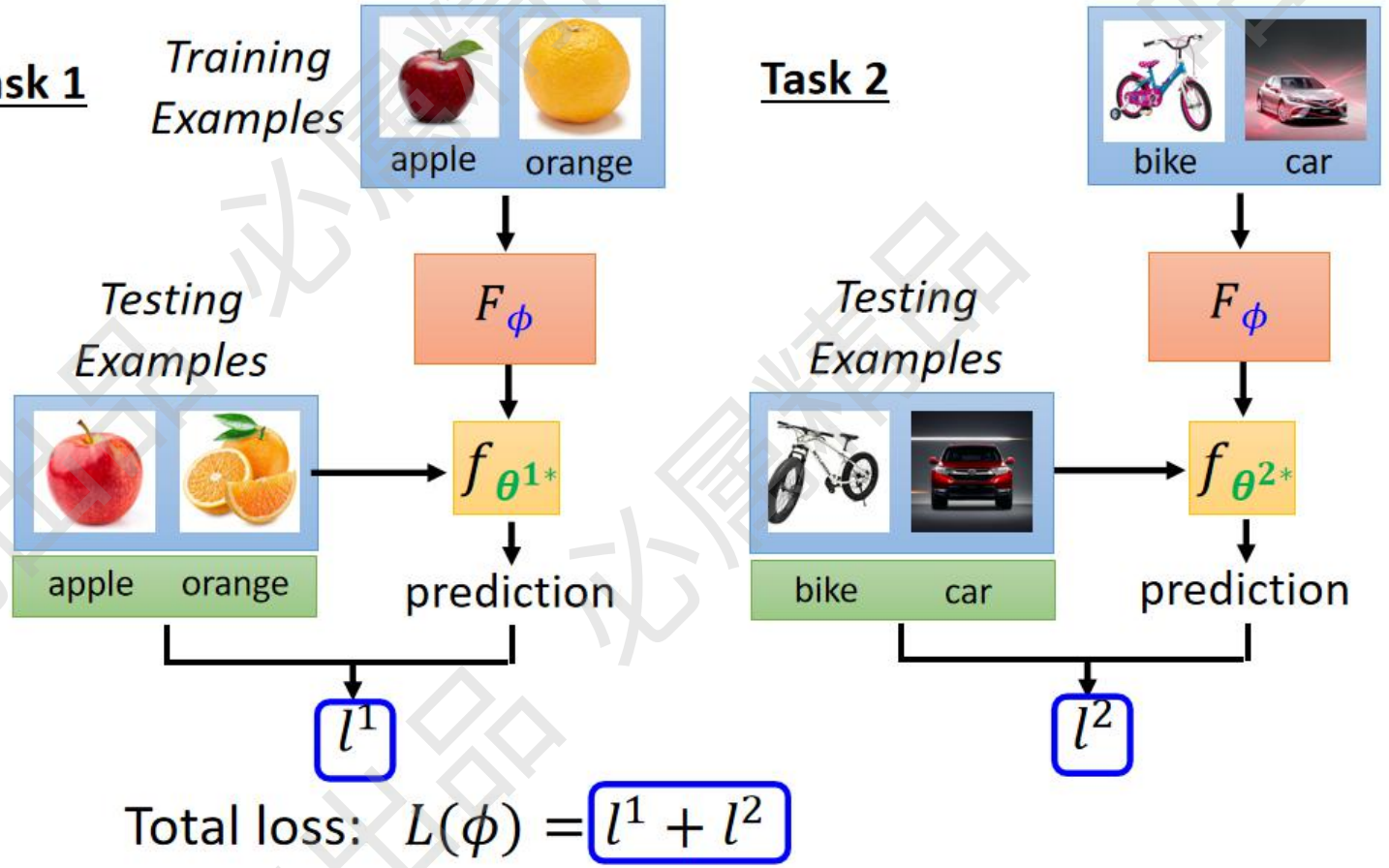
- 学习能力哪来的?
  - 都说深度学习是一个玄学，参数初始化的好，模型大概率也能学好
  - 如果初始化的这组权重参数学习能力就很强，是不是我们后期干事就容易多啦!
  - 但是如果得到合适的初始化参数呢，随机的够呛，主要看他前期学的咋样
  - 就好比我玩C，前期打的顺，后期基本无敌；前期让人家针对了，基本废了

MetaLearning

- 超参数如何定义
  - 基本所有干这行的都会告诉我，调参！大量实验尝试！
  - 但是，但是，咱们搞项目真的尝试了这么多吗？（点到为止）
  - 能不能让模型把这些烦人的参数（例如学习率）学出来呢？
  - MetaLearning也可以处理这个事的！

MetaLearning

- 如何得到合适的初始化参数呢?
  - 它得学习能力强!
  - 不同任务都能取得好成绩
  - 准备大量的任务让它试
  - 这回更新后的参数就全能了



MetaLearning

- 如何得到合适的初始化参数呢？
  - 我们选了多个任务来进行训练，但是计算损失时也用训练数据吗？
  - 每一个任务中都应该有训练数据和测试数据，实际损失计算用的测试数据
  - L1与L2的计算分别由其测试集来计算得到，在求和即可
  - 经过多个任务后计算总的损失并更新参数：$L(\phi) = \sum_{n=1}^{N} l^n$

MetaLearning

- MAML：Model-Agnostic Meta-Learning
  - 如何来学一个更好的初始化的参数，使其能更好的应用在不同任务中
  - 与预训练模型有啥不同呢？预训练模型描述的是在当前的数据上训练的很好
  - MAML希望的是当前好不好不太重要，重点是拓展到其他任务上能好
  - 预训练模型打前期，MAML打后期

MetaLearning

- MAML计算流程

  - 前提：数据要能采样，α，β是学习率
  - 1.随机初始化当前模型权重参数
  - 2，3.采样得到每一个任务
  - 5.计算在当前任务上的权重更新

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

MetaLearning

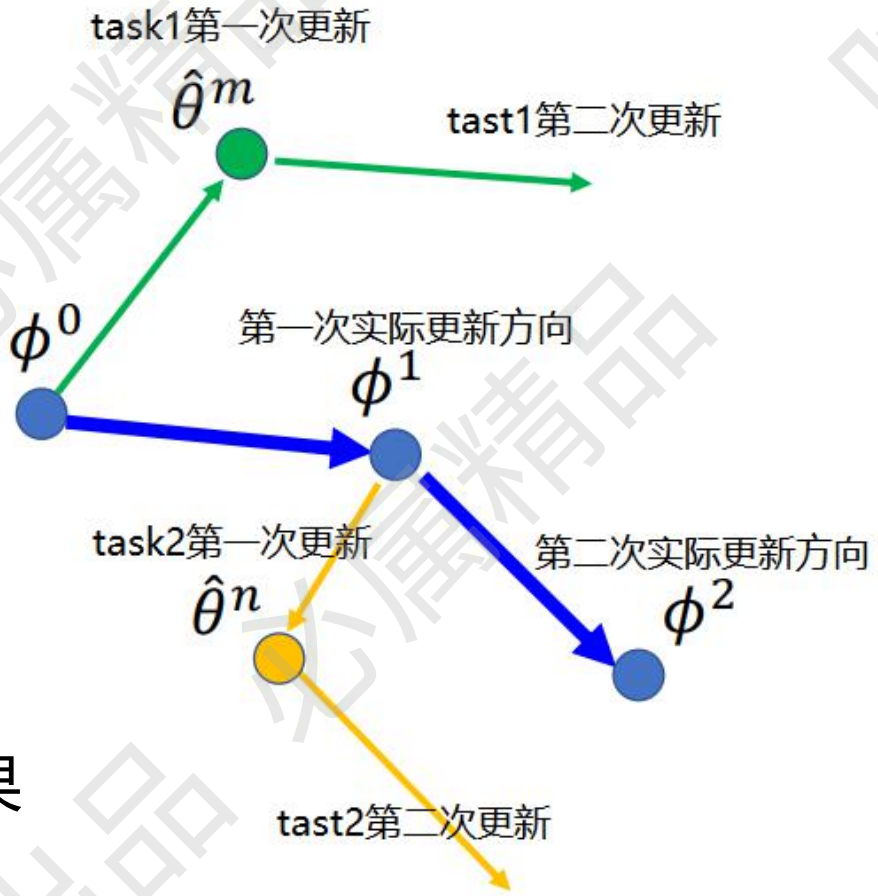
- MAML计算流程
  - 6.更新权重参数（梯度下降）
  - 4,5,6,7.可以迭代多个任务后再8
  - 根据这些任务的更新，再更新权重Θ
  - 相当于两次更新的过程

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

MetaLearning

- MAML计算流程
  - 假设每一次只有一个task
  - 注意实际参数更新分两步
  - 先得到中间结果，再对其进行更新
  - 根据上一步更新方向，得到实际更新结果



MetaLearning

- Few-Shot-Classification
  - N-way K-show表示每一个任务有N个类别，每个类别K个样本
  - 如果不同任务里面的类别和数据样本数不一样咋整?
  - 这就需要采样来完成了，通常固定N和K的个数
  - 主要针对数据少的情况，数据多的时候我觉得就算了