随着信息时代技术的不断发展，视频摄像头被广泛部署，发挥着越来越重要的作用。而另一方面，计算机视觉和人工智能领域技术的快速发展，也为视频分析任务提供了以较高精度自动化处理视频内容的技术方案。然而常见的深度学习模型通常也具有较高的资源开销，因而通常的做法是将摄像头采集的视频流发送给资源充沛的云，由云来运行神经网络模型并处理视频帧。但是由于视频帧之间存在着内在的时空相关性，连续的帧经由模型推理后，很有可能得到相同的结果，传输和处理这些冗余的帧显然只会对网络带宽资源和云的计算资源产生无谓的浪费。由此便产生了一种视频分析中的研究方向，即**帧过滤**。人们希望能通过某种算法，识别并过滤冗余帧，以此达到在不影响精度的前提下，降低视频分析带宽和计算开销的目的。

到目前为止，已经有许多有效的帧过滤方案被提出。比如Reducto,对帧的低级特征作帧差，并采取离线聚类产生的哈希表中的阈值过滤相似帧；FilterForward用特征提取器提取输入帧的特征，再作为微分类器的输入，输出一个帧与特定应用有关的概率，并只发送有关的帧；Glimpse只向服务器发送Trigger Frames，而缓存上一个返回的关键帧到目前帧间所有帧。这些技术或者使用轻量的神经网络，或者利用诸如像素的特征进行帧过滤，但它们都有一个共同点：即它们都需要在边缘对每一个帧都作相应处理来判断是否过滤。比如Reducto，当帧率为30fps时，需要对帧的低级特征两两作差得到一个29维的向量，并用这个向量作为键查询哈希表中对应的阈值。那么有没有我们有没有可能不需要逐帧处理来判断是否过滤该帧呢？作者认为这是一个具有前景的改进方向，并提出了FrameHopper，一种新的帧过滤方案。正如其名，它基于强化学习训练了一个模型，对当前帧，可以直接得出适宜的跳过帧数，而不需要对两帧间被过滤的帧作处理。

接下来是这个方案的设计部分。作者首先提出了主要目标，即最小化帧带过滤带来的误差的同时，最大化过滤帧的数量，它可以用如下的数学形式来表示，其中fi指一个帧序列中的第i张帧，k(i)指的是当fi帧被选为需要发送的帧时，应该跳过的后续帧数，而E就是由于过滤了fi之后k个帧而产生的误差。这个误差可以通过对被过滤帧和fi间的差异D求和得到。作者用F1分数来表示两帧间的相似性，那么差异D就可以用1-F1分数来表示。这里作者首先用一个可靠的目标检测模型对fi帧和fj帧生成边界框，再考虑如果两帧中相同label的bounding box有超过一定阈值（文中设定为0.5）的IoU，则视为一次命中。那么F1分数就可以这样计算。fj中的边界框数就作为检测到的物体数，而fi中的边界框数就作为实际的物体数。P表示保留帧的集合，P绝对值就表示保留帧的数目，而λ是一个可调节的系数，用来tradeoff帧过滤效率和精度。

作者将根据当前帧得到跳帧步长的问题归结为一个序列决策问题，那么就可以用强化学习来解决这个难以求解的问题了。这也是作者的inspiration。

下面我们来看FrameHopper的整体设计。我们可以将他的pipeline分为offline的模型训练和online的推理两个阶段。首先，将视频流传给靠近边缘的一个用于训练强化学习模型的服务器上，训练得到一个Agent。这个Agent再被部署到边缘节点上，执行帧过滤的工作，即Agent的推理过程，再将过滤后的帧发送给远程服务器。

我们先看具体的训练流程。作者把一个帧分为3\*3的chunk，计算前后帧每个chunk中发生改变的像素的比例，得到了一个9个元素的向量，作为state的定义。但是因为视频内容的动态性，这样一个9个元素的state的可能取值非常多，所以为了便于训练，减少状态数，还对state作了一次K-means聚类，得到了一个聚类模型。而action就是跳过帧的个数，即先前提到的k。但这里作者并未详细介绍action的集合是怎么选取的。对于reward的定义，我们首先认定在达到精度要求即一个阈值的情况下，跳过帧越多越好，而一旦没有达到精度要求，则认为是不好的。所以作者设计了这样一个函数作为reward，当误差低于阈值时，reward会随k单调递增；当误差高于阈值时，就为一个负值，负的reward就意味着惩罚。明确了这三大定义后，我们再来看他的训练过程。可以看到采取的是SARSA算法，每次迭代根据当前状态，选择Q值最大的action，即跳帧数k，再利用上面reward的定义计算Reward，到达下一个状态后再次选择action，并更新Q表和状态。不过这里我有个疑问，SARSA按理说是在线学习，但是作者又在文中强调他们是离线学习，就比较迷惑。还有一个疑问就是，这里的状态都与视频内容本身有关，是否意味着这样训练出来的一个Agent只适用于与训练数据集相同类型的视频？以及Reward与目标检测给出的F1分数有关，如何能保证目标检测的结果一直都是准确的？

还有一个疑问，大多数情况下，用户的期望准确率是会变化的，这样一来，每次对于不同的目标准确率，我们都需要重新训练一次Agent，感觉不太合理。

Agent训练完成后，就可以部署到边缘节点上了。Agent推理的算法如图。可以看到对于每个目前选中需要传送的帧，需要决定接下来需要跳过的帧的个数。在利用帧差加聚类得到当前状态后，直接使用训练得到的Q表得到a，即跳帧数，并且执行这个action。这里我也有个疑问，在起始时刻，这个算法是如何进行的？如果是把第一帧作为一个推理帧，在他之前没有其它推理帧，这个初始状态如何确定？作者也没有细说。

以上就是FrameHopper的全部设计部分。简而言之，就是使用强化学习来预测合适的跳帧步长，从而只需计算每个保留帧的帧差得到状态，而无需对被过滤的帧作任何处理，相较于传统方案，减少了计算量。

接下来是implementaion和evaluation。作者采用Jetson nano作为边缘节点，这是一个英伟达家的AI开发板，算力和GTX TITAN X差不多，不算高。云的配置是Quadro RTX 4000，和12gb显存的RTX 2060算力差不多。Dataset选取的是从youtube中的一些直播流截取下来的视频，我看了一下，内容都是美国一些地方的路况。实验选择了这四种方法互相比较，其中Fhop-slice就是FrameHopper，Fhop-diff是FrameHopper的一个变种，它不再将帧划分为九宫格，而是将目标检测结果，即各boundingbox的准确率作为vector。但是作者并没说这个vector的维数，所以具体的state定义不得而知。除此之外，后续如聚类之类的步骤都一致。以及这里的oracle solution，是作者自己设定的一种人工方法，通过大量实验，直接指定了哪些帧被过滤，哪些帧被保留，并且使之前提到的误差与保留帧数是所有可行方案中最小的，即可以理解为一种对真实情况下帧过滤所能达到的最好效果的模拟，我们可以暂且认为作者提出的这个oracle方案是合理且正确的。最后是实验的baseline，这里作者觉得手工给数据集上标签费时费力，就用EfficientDet作为目标检测模型，直接用它对数据集的检测结果作为label。这肯定会导致误差，但是作者特意解释了没有发生显著的错误。

这是在其中一个数据集上训练得到的结果。首先纵观abcd，可以看到在设定的目标准确度下，Reducto倾向于过滤更多帧，但准确率较低；而FrameHopper更偏向于维持准确度，过滤的帧比Reducto较少。但FrameHopper的表现比Reducto更贴近于理想状态下的oracle方案。作者据此认定FrameHopper优于Reducto。此外，还可以发现FrameHopper的变种效果并没有原版好。其实这里我也不太清楚作者搞这么个变种出来的意义。

接着作者还作了一个定性分析比较。如图，对于较高目标准确度，Reducto常常比理想方案过滤更多的帧，而FrameHopper与oracle保持一致；对于较低目标准确度，Reducto只在其中一小段时间内优于FrameHopper。

但是我们不难发现，在以上的实验中，围绕的评估标准并没有资源消耗、处理速度等因素。FrameHopper虽然需要运行强化学习模型，但是处理的帧数大大减少；Reducto只需要计算低级特征的帧差和查询哈希表，但每一个帧都需要处理，因此我个人觉得至少还应该对这两种方案的资源开销作实验，才能更进一步说明FrameHopper优于Reducto。

以及，在最后，作者还做了这样一个实验：将Reducto和FrameHopper都部署在边缘节点上，并发现FrameHopper在边缘的处理速度更快。但是Reducto设计的初衷是在资源更受限制的摄像头上过滤，而不是在边缘节点上过滤，我认为这样的比较实际上意义不大。

问题：

1. 强化学习训练阶段，action的集合是怎么选取的？

2. SARSA按理说是在线学习，但是作者又在文中强调他们是离线学习

3. 状态都与视频内容本身有关，是否意味着这样训练出来的一个Agent只适用于与训练数据集相同类型的视频？

4. Reward与目标检测给出的F1分数有关，如何能保证目标检测的结果一直都是准确的？

5. 大多数情况下，用户的期望准确率是会变化的，这样一来，每次对于不同的目标准确率，我们都需要重新训练一次Agent，感觉不太合理。

6.在起始时刻，Agent Inference是如何进行的？如果是把第一帧作为一个推理帧，在他之前没有其它推理帧，这个初始状态如何确定？

7. 变种方案的state定义不明确，以及这个变种存在的意义？

8. Reducto设计的初衷是在资源更受限制的摄像头上过滤，而不是在边缘节点上过滤，最后处理速度的比较实际上意义不大。

9. 将帧化为九宫格，把状态定义为9元素的向量，背后有什么道理？为什么这么做？