STREAMING MULTISCALE DEEP EQUILIBRIUM MODELS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

CAN UFUK ERTENLİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2022

Approval of the thesis:

STREAMING MULTISCALE DEEP EQUILIBRIUM MODELS

submitted by **CAN UFUK ERTENLİ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar Dean, Graduate School of **Natural and Applied Sciences**	__________
Prof. Dr. Halit Oğuztüzün Head of Department, **Computer Engineering**	__________
Assist. Prof. Dr. Ramazan Gökberk Cinbiş Supervisor, **Computer Engineering, METU**	__________
Assist. Prof. Dr. Emre Akbaş Co-supervisor, **Computer Engineering, METU**	__________

Examining Committee Members:

Assoc. Prof. Dr. Sinan Kalkan Computer Engineering, METU	__________
Assist. Prof. Dr. Ramazan Gökberk Cinbiş Computer Engineering, METU	__________
Assist. Prof. Dr. Cemil Zalluhoğlu Computer Engineering, Hacettepe University	__________

Date: 02.09.2022

ABSTRACT

STREAMING MULTISCALE DEEP EQUILIBRIUM MODELS

Ertenli, Can Ufuk
M.S., Department of Computer Engineering
Supervisor: Assist. Prof. Dr. Ramazan Gökberk Cinbiş
Co-Supervisor: Assist. Prof. Dr. Emre Akbaş

September 2022, 46 pages

There have been numerous significant developments for addressing recognition problems in recent years. One important application area of such developments is recognition on streaming data. Efficient inference is typically critical for streaming data sources, especially for real-time applications such as autonomous driving and robot control. For this purpose, this thesis presents StreamDEQ, a method that infers frame-wise representations on videos with minimal per-frame computation. In contrast to conventional methods where compute time grows at least linearly with the network depth, we aim to update the representations in a continuous manner. For this purpose, we leverage the recently emerging implicit layer models, which infer the representation of an image by solving a fixed-point problem. Our main insight is to leverage the slowly changing nature of videos and use the previous frame representation as an initial condition on each frame. This scheme effectively recycles the recent inference computations and greatly reduces the needed processing time. Through extensive experimental analysis, we show that StreamDEQ is able to recover near-optimal representations in a few frames' time and maintain an up-to-date representation throughout the video duration. Our experiments on video semantic segmentation and video object detection show that StreamDEQ achieves on-par accuracy with the baseline (standard MDEQ) while being more than 3x faster.

Keywords: Implicit Layer Models, Video Analysis and Understanding, Video Object Detection, Video Semantic Segmentation

ÖZ

AKAN VİDEO İÇİN ÇOK ÖLÇEKLİ DERİN EKİLİBRİYUM MODELLERİ

Ertenli, Can Ufuk
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Dr. Öğr. Üyesi. Ramazan Gökberk Cinbiş
Ortak Tez Yöneticisi: Dr. Öğr. Üyesi. Emre Akbaş

Eylül 2022 , 46 sayfa

Son yıllarda tanıma sorunlarının çözümü için çok sayıda dikkate değer gelişme olmuştur. Akan veriler üzerindeki tanıma görevleri bunlardan önemli bir tanesidir. Özellikle otonom sürüş ve robot kontrolü gibi gerçek zamanlı uygulamalar için akan veriler üzerinden verimli çıkarsama yapmak çok kritik olmaktadır. Bu amaçla, bu tezde, videolarda kare başına öznitelikleri, yapılan hesaplama miktarını en aza indirerek çıkartabilen bir yöntem olan StreamDEQi sunuyoruz. Hesaplama süresinin ağ derinliği ile en azından doğrusal olarak arttığı geleneksel yöntemlerin aksine, öznitelikleri sürekli bir şekilde güncellemeyi amaçlıyoruz. Bu amaçla, bir sabit nokta problemini çözerek bir görüntünün özniteliğini çıkaran son zamanlarda ortaya çıkan örtük katman modellerinden yararlanıyoruz. Ana anlayışımız, videoların yavaş değişen doğasından yararlanmak ve önceki karenin özniteliğini her yeni karede başlangıç noktası olarak kullanmaktır. Bu yaklaşım, son çıkarsama hesaplamalarını etkin bir şekilde geri dönüştürerek gerekli işlem süresini büyük ölçüde azaltır. Kapsamlı deneysel analizlerimiz sayesinde, StreamDEQin birkaç karelik süre içinde en iyiye yakın öznitelikler elde edebildiğini ve video süresi boyunca kare bazında güncel bir öznitelik sürdürebildiğini gösteriyoruz. Video anlamsal bölümleme ve video nesne tanıma üzerindeki deneylerimiz, StreamDEQin referans modelden (standart MDEQ) 3 kattan daha hızlı olurken, onunla benzer doğrulukta sonuç üretebildiğini göstermektedir.

Anahtar Kelimeler: Örtük Katman Modelleri, Video Analizi ve Anlama, Video Nesne Tanıma, Video Anlamsal Bölümleme

To my beloved family

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincerest gratitude to my supervisors Assist. Prof. Dr. Ramazan Gökberk Cinbiş and Assist. Prof. Dr. Emre Akbaş for their invaluable support and contributions. They have been with me through this journey at every step. I cannot thank enough.

I would like to thank Assoc. Prof. Dr. Sinan Kalkan and Assist. Prof. Dr. Cemil Zalluhoğlu for being members of my thesis committee and for their valuable feedback.

The numerical calculations were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA), and METU Robotics and AI Technologies Research Center (ROMER) resources. Additionally, I would like to thank TUBITAK for providing me with financial support through the 2210-A program.

Getting through my dissertation required more than academic support, and I have many people to thank for listening to me and providing their support for me. I would like to express my special thanks to my dearest friends, Dersu Giritlioğlu and Berkay Karacaer. They stood with me, especially when things were tough. I am also grateful to all my other friends whom I cannot name here.

Last but not least, my parents and my sister deserve endless gratitude. None of this could have happened without their immense support and continuous encouragement.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

DEQ	Deep Equilibrium Models
Eq.	Equation
mAP	Mean Average Precision
MDEQ	Multiscale Deep Equilibrium Models
mIoU	Mean Intersection over Union

CHAPTER 1

INTRODUCTION

Modern convolutional deep networks excel at numerous recognition tasks. It is commonly observed that deeper models tend to outperform their shallower counterparts [1, 2, 3], *e.g.* the prediction quality tends to increase with network depth using the architectures with residual connections [1]. Due to the sequential nature of the layer-wise calculations, however, increasing the network depth results in longer inference times. While the increase in inference duration can be acceptable for various offline recognition problems, it is typically of concern for many streaming video analysis tasks. For example, in perception modules of autonomous systems, it is not only necessary to keep up with the frame rate but also desirable to minimize the computational burden of each recognition component to reduce the hardware requirements and save resources for additional tasks. Similar concerns arise in large-scale video analysis tasks, *e.g.* on video sharing platforms, a slight increase in per-frame calculations can add up to significant increments in total consumption.

Various techniques have been proposed to speed up the inference in deep networks. A widely studied idea is to apply a large model to selected *key frames* and then either interpolate its features to the intermediate frames [4, 5] or apply a smaller model to them [6, 7]. However, such approaches come with several potential complications: (i) each time the larger model is applied, the model lags behind, the handling of which demands a complicated system design. (ii) Most methods require optical flow or motion estimates [4, 5], which brings in an additional estimation problem and an additional point of failure. In addition, the time cost of the flow estimation naturally reduces the time budget for all dependent steps. (iii) Special techniques need to be developed to maintain the compatibility of the representations and confidence scores obtained across the key and intermediate frames. (iv) The training schemes tend to be complicated due to the need for training over video mini-batches. It is also noteworthy that several models, *e.g.* [8, 9], rely on forward **and** backward flow estimates, making them less suitable for streaming recognition problems due to non-causal processing.

A related approach is to select a subset of each frame to process. These methods typically aim to identify the most informative regions in the input [10, 11, 12]. The region selection process can continue for static images until the model becomes confident about its predictions. However, when applied to videos, such subset selection strategies share shortcomings similar to approaches relying on flow-based intra-frame prediction approximations. The inputs change over time; therefore, the models have to choose between relying on optical flow to warp the rest of the features or omitting them entirely, which may result in obsolete representations over time [5].

In this context, the recently introduced *implicit layer models*, pioneered by the work on Deep Equilibrium Model (DEQ) [13] and Multiscale Deep Equilibrium Model (MDEQ) [14], offer a fundamentally different alternative to deep neural networks. DEQ (and MDEQ) show that by using the fixed-points of a network as the representation, one can gain the representation power of deep models using a network with only a few layers. The potential of DEQ to eliminate long chains of computations over network layers, therefore, renders it an attractive candidate towards building efficient streaming recognition models.

However, while DEQ provides a way to learn deep representations using shallow networks, the test-time inference process involves iterative root-finding algorithms such as Newton's method or quasi-Newton methods (*e.g.* Broyden's method [15]) to estimate the fixed-point of the network. On the other hand, one can leverage fixed-point iterations by repeatedly applying the same layer to its output to again find the fixed-point, which can be thought of as the explicit version of DEQ. Since each iteration can be interpreted as an increment in the network depth in both cases, DEQ effectively constructs deep networks for inference and, therefore, can still suffer from run-time costs as in explicit deep networks.

1.1 Proposed Methods and Models

Our main insight is the potential to speed up the inference process, *i.e.* fixed-point estimation, by exploiting the temporal smoothness across neighboring frames in videos.[1] More specifically, we observe that the fully estimated MDEQ representation of a frame can be used for obtaining the approximate representations of the following frames, using only a few inference iterations. We further develop the idea and show that even without fully estimating the representation at any time step, the implicit layer representation can be kept up-to-date by running the inference iterations over the iteration steps *and* video time steps in a continuous manner. The final scheme, starting from scratch, accumulates and transfers the extracted information throughout the video duration. We, therefore, refer to the proposed method as Streaming DEQ, or StreamDEQ for short.

The main difference between standard DEQ and StreamDEQ is illustrated in Figure 1.1. While DEQ typically requires a large number of inference iterations, StreamDEQ enables inference with only a few iterations per frame by leveraging the relevance of the most recent frame's representation. At the start of a new stream, or after a major content change (*e.g.* a shot change), StreamDEQ quickly adapts to the video in a few frames, much like a person adapting her/his focus and attention when watching a new video. In the following frames, it continuously updates the representation to adapt to minor changes (*e.g.* objects moving, entering, or exiting the scene).

Overall, StreamDEQ provides a simple and *lean* solution to the streaming recognition both with implicit layer models and the explicit versions of said implicit models, where a single model naturally performs cost-effective recognition without relying on external inputs and heuristics, such as optical flow [16], post-processing methods (Seq-NMS [17] or tubelet re-scoring [18, 8]). Our method also maintains the causality of the system and executes in a continuous manner. We also note that it allows dynamic time budgeting; the duration of the inference process can be tuned on-the-fly by a controller, depending on the instantaneous compute system load, which can be a desirable feature in real-world scenarios.

[1] We do not refer to a mathematical definition of smoothness but rather emphasize that the changes between neighboring frames are small.

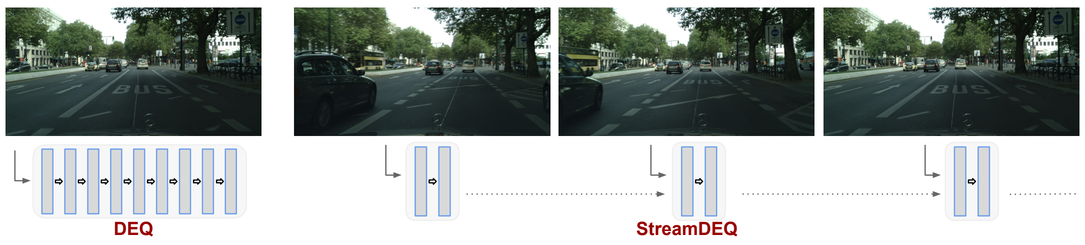

Figure 1.1: Our method, StreamDEQ, exploits the temporal smoothness between successive frames and extracts features via a small number of solver iterations, starting from the previous frame's representation as initial solution. StreamDEQ accumulates and transfers the extracted information continuously over successive frames, effectively sharing computations across video frames in a causal manner.

We verify the effectiveness of the proposed method through extensive experiments on video semantic segmentation and video object detection. Our experimental results show that StreamDEQ recovers near-optimal representations at much lower inference costs. More specifically, on the ImageNet-VID video object detection task, StreamDEQ converges to the mAP scores of 51.6 and 58.9 using only 4 and 8 inference iterations per frame, respectively. In comparison, the standard DEQ inference scheme yields only 8.2 and 32.6 mAP scores using 4 and 8 iterations, respectively. Similarly, on the Cityscapes semantic segmentation task, using StreamDEQ instead of the standard DEQ inference scheme improves the converged streaming mIoU score from 42.3 to 71.5 when 4 inference iterations are used per frame and from 73.2 to 78.2 when 8 iterations are used per frame.

1.2 Contributions and Novelties

Our contributions are as follows:

- We introduce a new inference scheme based on DEQ that can be applied to streaming videos to improve processing speeds drastically while maintaining a high level of performance,
- To the best of our knowledge, we provide the first attempt at applying implicit layer models to videos and specifically streaming videos,

- We empirically show that the fixed-points for neighboring video frames are near each other; therefore, previous frames' fixed-points provide a good starting point for the upcoming frames,
- In addition to applying the implicit layers to streaming videos, we also show the effectiveness of our method when applied in an explicit manner.

We note that part of this work has been accepted to and will appear at the European Conference on Computer Vision (ECCV 2022).

1.3 Outline of the Thesis

In the rest of the thesis, we first provide an overview of related work in Chapter 2. In Chapter 3, we describe our inference scheme and how we apply StreamDEQ to streaming videos. In Chapter 4, we first present the details of our experimental setup and demonstrate the results of our extensive experiments to verify the effectiveness of our method on challenging datasets. Finally, we conclude with a summary of our work in Chapter 5.

CHAPTER 2

RELATED WORK

Here, we summarize efficient video processing methods, video object detection and segmentation models. Furthermore, we discuss saliency-based techniques for image and video processing. Finally, we overview implicit models and introduce some important application areas.

2.1 Efficient Video Processing and Inference

There have been many efforts to improve video processing efficiency to reach real-time processing speeds. Most of these works take a system-oriented approach [19, 20, 21]. For example, Carreira et al. [19] develop an efficient parallelization scheme over multiple GPUs and process different parts of a model in separate GPUs to improve efficiency while sacrificing accuracy due to frame delays. Narayanan et al. [20] propose a novel scheduling mechanism that efficiently schedules and divides forward and backward passes over multiple GPUs. In another work, Li et al. [21] use a dynamic scheduler in which the model chooses to skip a frame when the delays build up to the point where it would be impossible to calculate the results of the next frame in the allotted time.

We also note that works on low-cost network designs, such as MobileNets [22, 23] and low-resolution networks [24, 25], are also relevant. Such efforts are valuable primarily for replacing network components with more compute-friendly counterparts. However, the advantages of such techniques can also be limited due to natural trade-offs between speed and performance [26], as the lower-cost network components tend to have lower expressive power. Nevertheless, one can easily incorporate low-cost model design principles into DEQ or StreamDEQ models, thanks to the architecture-agnostic definition of implicit layer models. While such efforts may bring reductions in inference wall-clock time, they are outside the scope of our work.

2.2 Video Semantic Segmentation

Semantic segmentation is a costly, spatially dense prediction task. Due to this specific nature of the task, its application to videos remains relatively limited. Most works rely on exploiting temporal relations between frames using methods such as feature warping [27, 6, 28, 29], feature propagation [30, 31, 32], feature aggregation [33], and knowledge distillation [34, 35, 36] to reduce the computational cost.

Gadde et al. [27] propose warping features of the previous frame at different depths based on optical flow. Xu et al. [6] evaluate regions of the input frame and decide whether to warp the features with a cheap flow network or use the large segmentation model based on a confidence score. Huang et al. [28] keep a moving average over time by combining the segmentation maps from the current frame with the warped map from the previous frame. Jain et al. [29] warp high-quality features from the last key frame and fuse them with lower-quality features calculated on the current frame to make predictions.

Shelhamer et al. [30] propose an adaptive method that schedules updates to the multi-level feature map so that features of layers with smaller changes are carried forward (without any transformation). Li et al. [31] introduce an adaptive key frame scheduling method based on the deviation of low-level features compared to the previous key frame. If the deviation is small, the features are propagated with spatially variant convolution. Liang et al. [32] apply the full model on key frames and propose to adaptively choose the depth of the network at each non-key frame with a dynamic gating module while they re-use the rest of the features.

Hu et al. [33] use a set of shallow networks, each calculating features of consecutive frames starting from scratch. Then, these features are aggregated at the current frame with an attention-based module. Liu et al. [34, 35] propose to use an expensive network during training including optical flow and applies knowledge distillation on a student network to benefit from the high capacity of a teacher network while cutting computational costs thanks to a smaller and more efficient student network which the paper uses during inference. On the other hand, Habibian et al. [36] use a teacher network on key frames, and through knowledge distillation, guide the student network to learn feature differences, *i.e.* delta differences, that are combined with the key frame features to simulate the teacher on non-key frames.

In contrast to all these approaches, the proposed StreamDEQ method directly leverages the similarities across video frames, without requiring any ad-hoc video handling strategies, as a way to adapt the implicit layer inference mechanism to efficient streaming video analysis.

2.3 Video Object Detection

Most modern video object detection methods exploit temporal information to improve the accuracy and efficiency. To this end, optical flow [4, 37, 8, 9], feature aggregation [37, 38, 39, 40] and post-processing techniques [17, 18] are prominently used.

Zhu et al. [4] introduce Deep Feature Flow (DFF) and use optical flow estimates to warp features on selected key frames to intermediate frames for increased efficiency. Zhu et al. [37] also propose Flow-Guided Feature Aggregation (FGFA) which uses optical flow to warp features of nearby frames to the current frame and aggregates these features adaptively based on weights calculated from the similarity between features. Kang et al. [8] create links between objects through time (tubelets) from the predictions calculated with optical flow across a video linking objects through time and apply tubelet re-scoring to keep detections of high confidence. Wang et al. [9] add an instance level calibration module to FGFA [37] and combine them to generate better predictions.

Bertasius et al. [38] sample features from neighboring support frames via deformable convolution that learns object offsets between frames and aggregates these features over these frames. Wu et al. [39] focus on linking object proposals in a video according to their semantic similarities. Chen et al. [40] propose a model aggregating local and global information with a long-range memory.

Another common way to improve performance is to apply a post-processing method. For example, Han et al. [17] introduce Seq-NMS to exploit temporal consistency by constructing a temporal graph to link objects in adjacent frames. With a similar idea, Kang et al. [18] generate tubelets by combining single image detections through the video and use a tracker to re-score the tubelets during post-processing to improve temporal consistency.

2.4 Saliency-Based Techniques

To reduce computational cost, another viable approach is to select important regions in an image and process only those small patches [10, 11, 12, 41]. Video extensions of these models also exist [42, 43, 5, 44, 45, 46, 47].

Mnih et al. [10] and Ba et al. [11] model human eye movements by capturing *glimpses* from images with a recurrent structure and process those glimpses at each step. Cordonnier et al. [12] propose selecting the most important regions to process by first processing a downsampled version of the image. Liu et al. [41] stops processing for regions with high-confidence predictions at an earlier stage.

Bazzani et al. [42] and Denil et al. [43] approach video processing in a human-like manner where the model *looks at* a different patch around the objects of interest at each frame and tracks them. Zhu et al. [5] take a key frame based approach. At each key frame the method processes the entire input, and at intermediate frames, it updates the feature maps partially based on temporal consistency. Rhee et al. [44] identify changing regions between frames and re-use the features of the static parts on non-key frames. Habibian et al. [45] introduce skip-convolutions where the model determines changing locations via frame difference and computes convolutions only on some part of each frame. Patchwork [46] uses a Q-learning based policy to select a sub-window in each frame and combines the sub-window features with the rest of the features via an attention mechanism. SALISA [47] focuses on an intelligent downsampling method that *magnifies* important regions in each frame and reduces the frame's resolution.

2.5 Implicit Layer Models

Implicit layer models have seen a recent surge of interest and have been outstanding at tackling numerous tasks. DEQ [13] is a new addition to the implicit model family aimed at solving sequence modeling tasks. DEQ reformulates the fixed-point solving problem as root finding and utilizes an iterative root-finding algorithm to find a solution. Multiscale Deep Equilibrium Models (MDEQ) [14] are the extension of the base DEQ to image-based models where there are multiple fixed-points for different feature scales. Since the introduction of these models, there have been many efforts to improve DEQs in terms of speed and accuracy and to find new application areas where DEQs can be successful.

Huang et al. [48] propose re-using the fixed-point across training iterations with the drawback of having to stay in full-batch mode for the training. Bai et al. [49] suggest a new initialization scheme that is realized through a small network. Furthermore, inferring information from the last few iterations reduces the number of solver iterations required for convergence. Pal et al. [50] propose mixing implicit models with explicit models. Usually, implicit models are initialized from scratch, *i.e.* from a random point or zero, to iteratively reach the equilibrium point. Instead, this method first utilizes an explicit model to calculate a "better" starting point for the implicit model. With this technique, the method obtains better performance and enjoys faster inference.

Even though DEQs offer a new avenue for research that can potentially surpass explicit models, they come with several problems. First of all, DEQ models grow increasingly unstable as the training progresses, and they are brittle to architectural choices meaning even minor modifications may harm the model's convergence [51]. Furthermore, their backward passes are especially costly [52, 53]. Bai et al. [51] propose adding a Jacobian regularization term to mitigate these issues to improve model training. Geng et al. [52] and Fung et al. [53] focus on the backward pass to make it less costly. Both works [52, 53] utilize the Neumann series expansion and show that calculating inexact gradients using low-level expansions are sufficient during training.

After the massive success of implicit layer models, they have been applied to several tasks in place of their explicit counterparts, such as optical flow estimation [54], normalizing flows [55], feature refinement [56], and Feature Pyramid Networks [57].

CHAPTER 3

PROPOSED METHOD

We build our method on the Deep Equilibrium Model (DEQ). In this chapter, we first give an overview of DEQ and then present the details of our method.

3.1 DEQ Overview

Weight-tied networks are models where some or all layers share the same weights [58, 59]. A DEQ is essentially a weight-tied network with only one shallow block. DEQ leverages the fact that continuously applying the same layer to its output tends to guide the output to an equilibrium point, *i.e.* a fixed-point. Let $\mathbf{x}$ represent the model's input, $\mathbf{z}^*$ the equilibrium point, and f_θ the applied shallow block, then an explicit weight-tied network can be described as

$$\lim_{i\to\infty} \mathbf{z}^{[i+1]} = \lim_{i\to\infty} f_\theta(\mathbf{z}^{[i]}; \mathbf{x}) \equiv f_\theta(\mathbf{z}^*; \mathbf{x}) = \mathbf{z}^*. \tag{3.1}$$

Each iteration in Eq. (3.1) is called an unrolling of the model, akin to recurrent neural networks (RNNs), and this scheme is called the fixed-point iteration. DEQ's fundamental difference from a standard weight-tied model is that the model is represented by an implicit equation, and the fixed-point is found by employing root-finding algorithms in both forward and backward passes, by rewriting Eq. (3.1) as follows:

$$g_\theta(\mathbf{z}; \mathbf{x}) = f_\theta(\mathbf{z}; \mathbf{x}) - \mathbf{z} = 0 \implies \mathbf{z}^* = \text{RootFind}(g_\theta; \mathbf{x}). \tag{3.2}$$

DEQ uses Broyden's method [15] to calculate the root of Eq. (3.2). In these settings (Eq. (3.1) and Eq. (3.2)), the accuracy of the solution depends on the number of unrollings or the number of Broyden iterations [49, 51, 56]. While more iterations yield better accuracy, they increase computation costs.

DEQs have been successfully adapted to computer vision tasks, too, with the introduction of Multiscale Deep Equilibrium Models (MDEQ) [14]. MDEQ is a multiscale model where each scale is driven to equilibrium together with other scales in the same manner as DEQs. Iterations start with $\mathbf{z}^{[0]} = 0$ and continue N times to obtain the final solution, $\mathbf{z}^{[N]}$. N is set to 26 for ImageNet classification and 27 for Cityscapes semantic segmentation in MDEQ [14].

3.2 Streaming DEQ

Let $\mathbf{X}$ be a $H \times W \times 3 \times T$ dimensional tensor representing a video where T is the temporal dimension. We represent the frame at time t with $\mathbf{x}_t$ which is a $H \times W \times 3$ tensor. It should be noted that we primarily target videos with temporal continuity, without too frequent shot changes. But we also study the effects of shot changes in Chapter 4.

To process a video, DEQ can be applied to each video frame $\mathbf{x}_t$ to obtain $\mathbf{z}_t$, the representation of that frame. This amounts to running the Broyden solver for N iterations starting from $\mathbf{z}_t^{[0]} = \mathbf{0}$ for each frame.

However, we know a priori that transitions between subsequent video frames are typically smooth, *i.e.* $\mathbf{x}_{t-1} \sim \mathbf{x}_t$. From this observation, we hypothesize that the corresponding fixed-points, *i.e.* representations $\mathbf{z}^*_{t-1}$ and $\mathbf{z}^*_t$, are likely to be similar. Therefore, the representation of the previous frame can be used effectively as a starting point for inferring the representation of the current frame. To validate this hypothesis, we run an analysis on the ImageNet-VID [60] dataset using the ImageNet pretrained MDEQ model. We assume that at each frame $\mathbf{x}_t$, we have access to the reference representation, $\mathbf{z}^*_{t-1}$, of the previous frame. Reference representations are obtained by running the MDEQ model until convergence ($N = 26$ iterations). At each frame, we use the reference representation of the previous frame as the starting point of the solver,

$$\mathbf{z}_t^{[0]} = \mathbf{z}^*_{t-1}, \tag{3.3}$$

and run the solver for various but small numbers of iterations, M. To analyze the amount of change in representations over time, we use an ImageNet-pretrained model

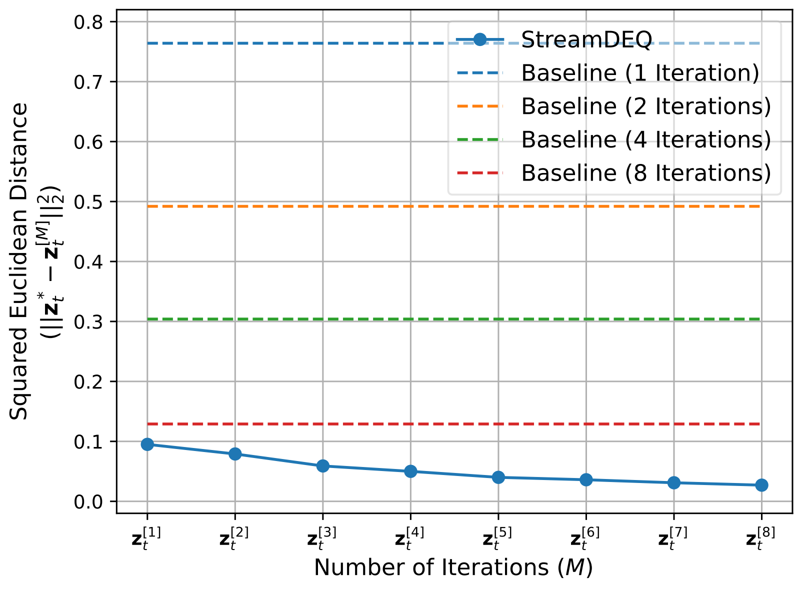

Figure 3.1: Squared Euclidean approximation error as a function of inference steps, when the solver is initialized with the reference representation of the preceding frame.

(MDEQ-XL) since ImageNet representations are known to be useful in many transfer learning tasks. In Figure 3.1, we show the squared Euclidean distance between $\mathbf{z}_t^{[M]}$ and $\mathbf{z}_t^*$ for various t values when the solver starts as in Eq. (3.3). Dashed lines correspond to the squared Euclidean distance between MDEQ-XL's reference and M-iteration based representations.

From the results presented in Figure 3.1, we observe that when we initialize the solver with the preceding frame's fixed-point, the inference process quickly converges towards the reference representation. We also observe that after starting from the reference representation of the previous frame and performing only 1 iteration on the current frame, the approximate representation is already more similar to the reference representation than starting from scratch and performing 8 iterations.

Next, we examine the case where the inference method is given access to the reference representations only at certain frames. To simulate this case, for each video clip, we compute the reference representation only at the first frame $\mathbf{x}_0$, *i.e.* $\mathbf{z}_0^*$. In all following ones, we initialize the solver with the estimated representation of the preceding frame and run the solver for M iterations. That is,

$$\mathbf{z}_1^{[0]} = \mathbf{z}_0^* \text{ and } \mathbf{z}_t^{[0]} = \mathbf{z}_{t-1}^{[M]}. \tag{3.4}$$

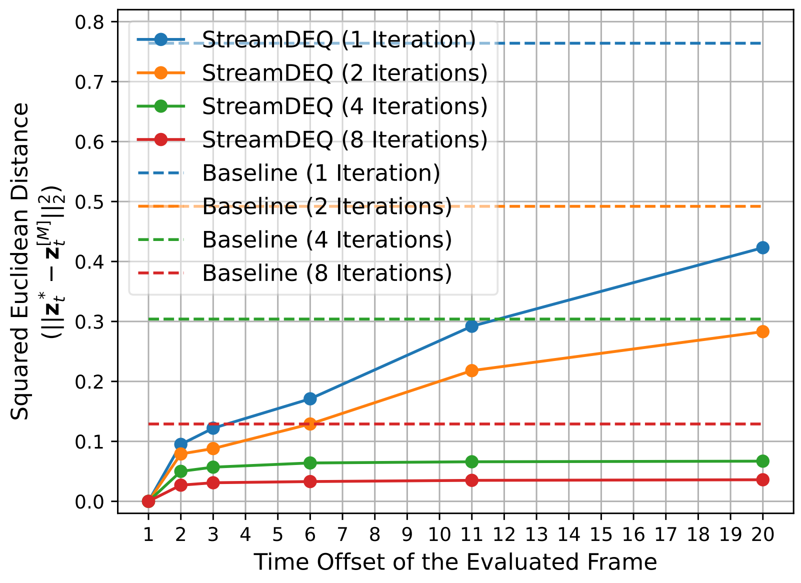

Figure 3.2: Distance between the reference representations and StreamDEQ estimations for varying number of iterations, when StreamDEQ is initialized with reference representations on the first frame.

We present the results of this scheme for $M \in \{1, 2, 4, 8\}$ in Figure 3.2. We observe that starting with the reference representation on the initial frame is still useful, but for longer clips, its effect diminishes. Still, this scheme helps us maintain a stable performance even after several frames. For example, starting with the reference representation and then applying $M = 2$ iterations per frame throughout the following 20 frames yields a representation closer to the reference representation of the final frame than the one given by baseline DEQ inference with 4 solver iterations. This result shows that the M-step inference scheme is able to keep up with the changes in the scene by starting from a good initial point.

While this scheme can provide efficient inference on novel frames, we would still need the reference representations of the initial frames, or key frame(s), which would share the same problems with key frame based video recognition approaches, *e.g.* [4, 6, 7]. To address this problem, we further develop the idea, and hypothesize that we can start from scratch (*i.e.* all zeros), do a limited number of iterations per frame, and pass the representation to the next frame as the starting point. That is,

$$\mathbf{z}_0^{[0]} = 0 \text{ and } \mathbf{z}_t^{[0]} = \mathbf{z}_{t-1}^{[M]}. \tag{3.5}$$

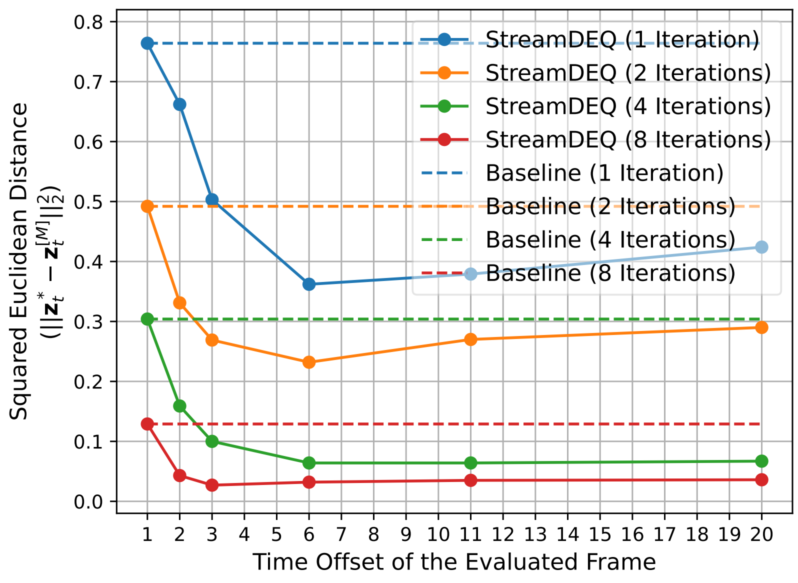

Figure 3.3: Distance between the reference representations and StreamDEQ estimations for varying number of iterations, when StreamDEQ is initialized with just zeros on the first frame.

We present the representation distance results for this final scheme in Figure 3.3. The representation distances to the reference representations stabilize in 20 frames. Converged distance values (in 20 frames) are almost the same as those of the previous scheme (Eq.(3.4)). Additionally, the initial representations have relatively large distances, but these differences get smaller as new frames arrive. We call this final scheme StreamDEQ. This scheme avoids heavy processing in any of the frames and completely avoids the concept of key frames. The number of Broyden iterations can be tuned, which allows easy control over the time-vs-accuracy trade-off. Therefore, the inference iterations can be run as much as the time budget allows. An illustration of the StreamDEQ inference process is given in Figure 3.4.

3.3 Unrolled Streaming DEQ

In recent works [13, 14, 51], one of the main advantages of using DEQ is indicated as asymptotically constant memory training in terms of the number of layers. Yet, training and inference with DEQs are computationally more costly than explicit models

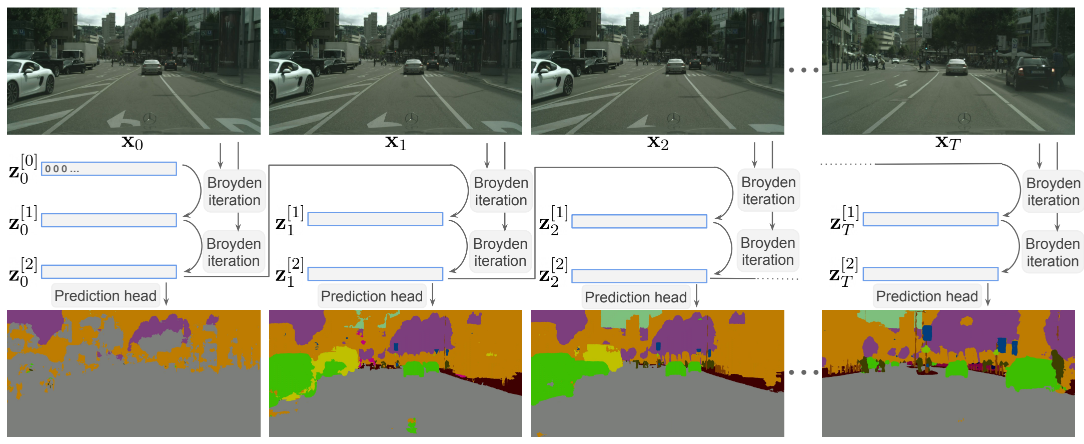

Figure 3.4: StreamDEQ applied to a streaming video, performing two iterations per frame. The representation inference process is initialized with zeros in the very first frame ($\mathbf{z}_0^{[0]} = \mathbf{0}$) and with the most recent representation ($\mathbf{z}_t^{[0]} = \mathbf{z}_{t-1}^{[2]}$) in the rest of the stream. This scheme effectively recycles all recent computations for time-efficient inference on a new frame and, therefore, allows approximating a long inference chain (*i.e.* a deep network) by a few inference steps (*i.e.* a few layers) throughout the video stream.

with comparable sizes (see Table 4.2 & 4.3). Furthermore, constant memory training is irrelevant during inference which is the primary aim of this thesis. Therefore, instead of using the implicit formulation of DEQ, we can also formulate our scheme as the fixed-point iteration problem from Eq. (3.1), *i.e.* the weight-tied setting. This allows us to approach the fixed-point solving problem through the more standard explicit point of view. This only changes how the fixed-point is found while keeping everything else the same, *e.g.* f_θ. We call this explicit version of StreamDEQ, Unrolled StreamDEQ (UR-StreamDEQ). We emphasize that UR-StreamDEQ may not be considered a DEQ model, but in order to keep our notation consistent throughout the thesis, we adhere to the same name.

As the only difference between these two versions is the solution procedure, our previous arguments regarding the *smoothness* of videos and the nature of fixed-points of neighboring frames are still valid for UR-StreamDEQ. Thus, we can still argue that starting with either the reference representations (Eq. (3.4)) or from scratch (Eq. (3.5)), UR-StreamDEQ eventually reaches a stable condition where its representations do not vary too much from the reference representations. We confirm this claim with our experimental findings in Chapter 4. Note that when considering Eq. (3.4) and Eq. (3.5) in the context of UR-StreamDEQ, the superscripts represent the number of unrollings instead of the Broyden iterations.

Most previous works [13, 14, 57, 56, 54], however, prefer to use the implicit variant and Broyden iterations over the explicit weight-tied version. They suggest that Broyden iterations are more effective in solving for the fixed-point. These works also claim that it is necessary to perform an infinite number of fixed-point iterations in order to reach equilibrium, whereas with Broyden iterations, an equilibrium point can be reached more quickly. In Chapter 4, we empirically demonstrate that this is not always the case and less than 20 fixed-point iterations are sufficient to reach an equilibrium in most scenarios.

Even when the model is unrolled very few times per frame in streaming evaluation, we achieve performances on the same level as the implicit variants with a significantly less computational burden. Moreover, even though it is not the focus of this thesis, with UR-StreamDEQ, we can also achieve much faster training (see Table 4.2) by avoiding the costly calculation of the Jacobian inverse and relying on the highly optimized explicit backpropagation with the chain rule.

CHAPTER 4

EXPERIMENTAL RESULTS

We evaluate our method on video semantic segmentation and video object detection. In the following, we provide technical details regarding training and inference setups, the datasets used, and present our experimental findings. We use the PyTorch [61] framework for all experiments.

4.1 Video Semantic Segmentation

4.1.1 Experimental Setup

We use the Cityscapes semantic segmentation dataset [62], which consists of 5K finely annotated and 20K coarsely annotated images. These finely annotated images are divided into train, validation, and test sets, each containing 2975, 500, and 1525 images, respectively. They correspond to frames extracted from video clips where each annotated image is the 20^{th} frame of its respective clip. To evaluate over videos, we use these clips up to the 20^{th} frame, which has fine annotations, and evaluate on that frame.

We use the pretrained MDEQ-XL segmentation model from the MDEQ paper [14] and do not perform any additional training. We also do not make any changes to its evaluation setup or hyperparameters, perform the evaluation on Cityscapes `val` and report *mean intersection over union* (mIoU) results. The architecture of the MDEQ-XL consists of 4 residual blocks of different scales where each block contains convolution layers, group normalization, and ReLU activation. For further details, we refer the reader to MDEQ [14].

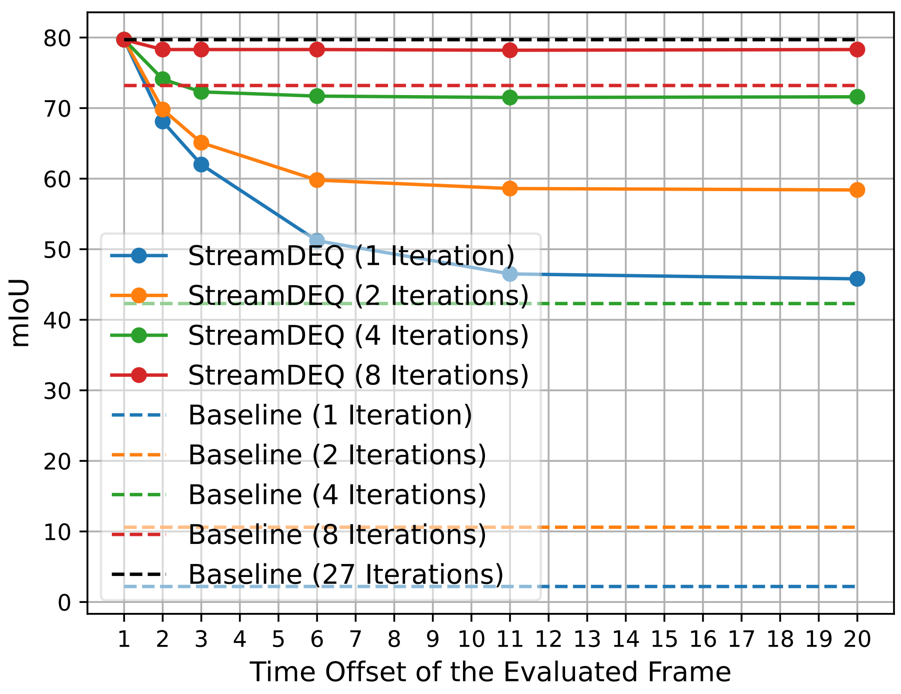

Figure 4.1: StreamDEQ semantic segmentation results (in mIoU) on the Cityscapes dataset as a function of solver iterations when the first frame representation is initialized with the reference representation.

4.1.2 Results

We present the results of StreamDEQ for two scenarios. The first scenario corresponds to Eq. (3.4), where we use the reference representations of the first frame to initialize the solver and apply StreamDEQ then on. Results of this experiment in Figure 4.1 show that as the offset of the evaluated frame increases, mIoU starts decreasing, which is expected because the further we move away from the first frame, the more irrelevant its representation will become. However, mIoU then stabilizes at a value proportional to the number of Broyden iterations (the more iterations, the better the mIoU). This shows that StreamDEQ is able to extract better features over time. StreamDEQ's performance with 8 iterations is still comparable with the baseline (MDEQ) with 27 iterations.

The second scenario corresponds to our final StreamDEQ proposal (*i.e.* Eq. (3.5)), where we initialize the solver from scratch, *i.e.* with all zeros, and apply StreamDEQ.

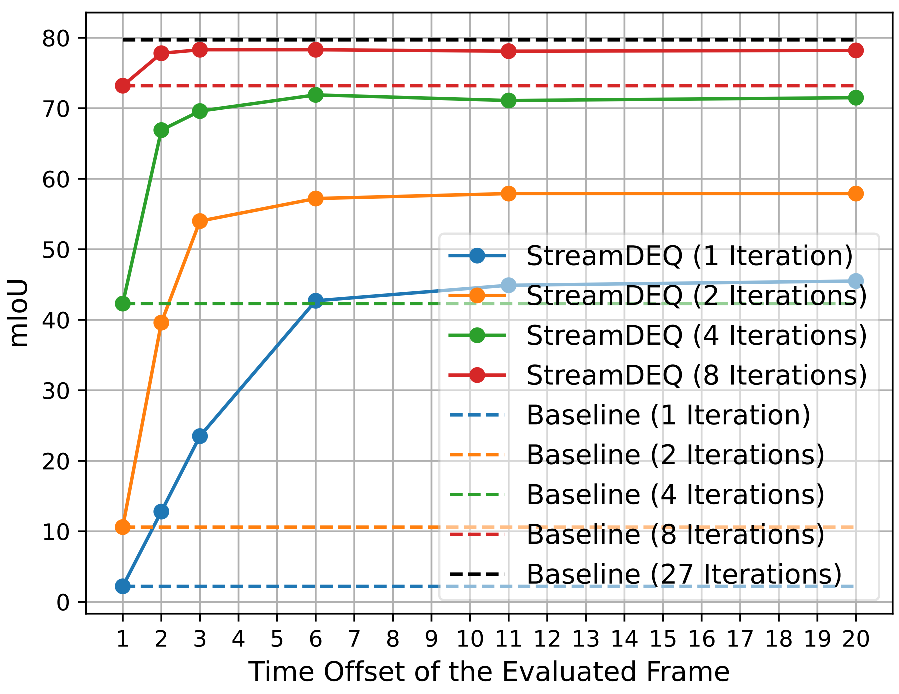

Figure 4.2: StreamDEQ semantic segmentation results (in mIoU) on the Cityscapes dataset as a function of solver iterations when the first frame representation is initialized with zeros.

The results of this case are shown in Figure 4.2. As the videos progress, one might expect that the Broyden solver cannot keep up with the changing scenes. However, we observe that even after 20 frames, the accuracy does not drop. Additionally, the impact of this method is more evident for the lower numbers of iterations. For example, performing 1 iteration on every frame without our method would only yield an mIoU score of 2.2. However, StreamDEQ obtains an mIoU score of 44.9 in 10 frames. This is an improvement of over 20×. For 8 iterations, StreamDEQ is able to obtain 78.1 mIoU in 10 frames, whereas the non-streaming baseline achieves only 73.2 mIoU. Moreover, the converged mIoU values (at larger frame offsets) are similar in Figure 4.1 and Figure 4.2. Therefore, we conclude that the initial point where we start the solver becomes less crucial as the video streams and the performance stabilizes at some value higher than in the non-streaming case.

We also illustrate these results qualitatively in Figure 4.3. For 1 iteration, while the baseline cannot produce any meaningful segmentation, StreamDEQ starts capturing

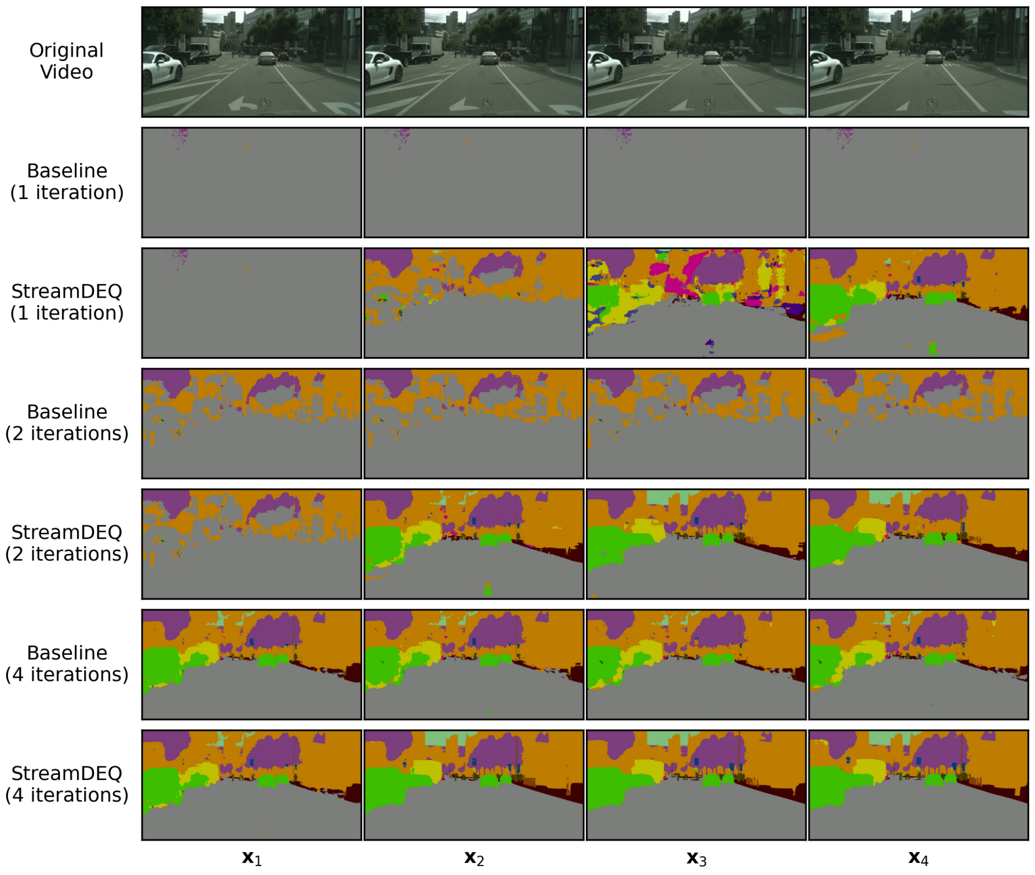

Figure 4.3: Qualitative comparison of the baseline with StreamDEQ with different numbers of iterations on the Cityscapes dataset.

many segments correctly at the 4^{th} frame. With 2 iterations, while the DEQ baseline still produces poor results, StreamDEQ starts to yield accurate predictions in early frames compared to the single iteration case. With 4 iterations, while both models provide rough but relevant predictions in the first frame, StreamDEQ predictions start to become clearly more accurate in the following frames; for example, tree trunks and the sky become visible only when StreamDEQ is applied.

We examine the effects of increasing the number of iterations on inference speed in Table 4.1. We note that our method does not introduce any computation overhead other than the time it takes to store the fixed-point representation of the previous frame. Therefore, we observe a linear increase in compute times as the number of iterations increases. StreamDEQ with 4 iterations achieves an mIoU score of 71.5 at 530 ms per image. MDEQ with 4 iterations can only achieve 42.3 mIoU.

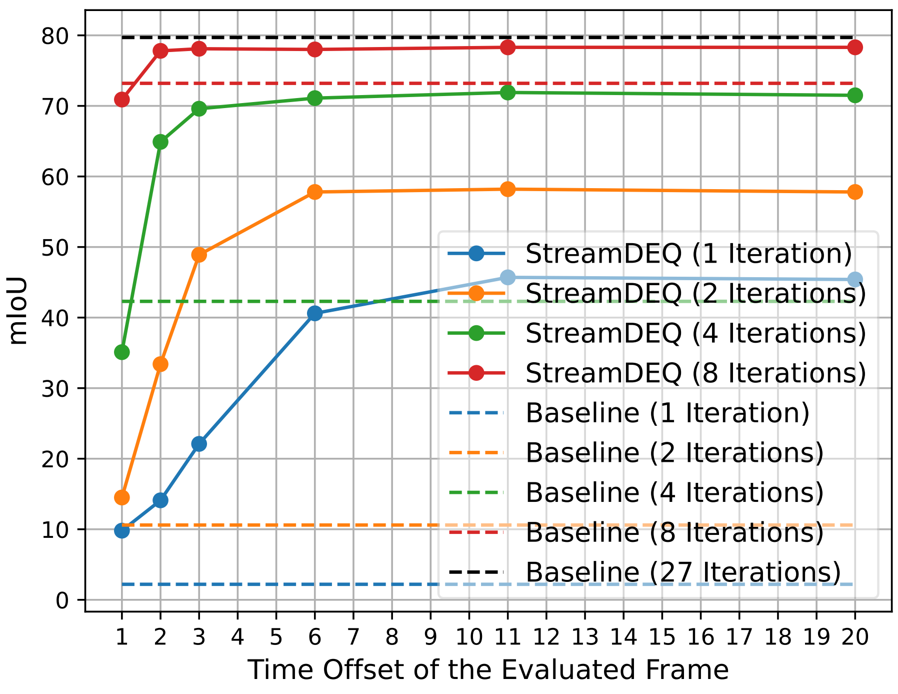

Figure 4.4: mIoU results of StreamDEQ with shot changes from the Cityscapes dataset.

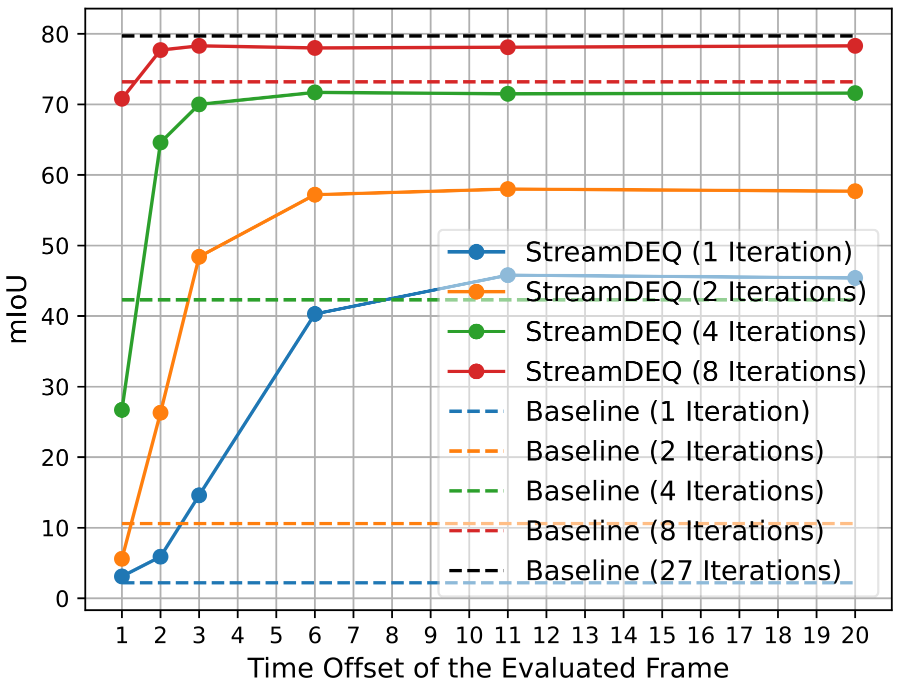

Figure 4.5: mIoU results of StreamDEQ with shot changes from the ImageNet-VID dataset.

4.1.3 Effect of Shot Changes

We also study the effects of shot changes for the video semantic segmentation task where we effectively connect two different clips together. To simulate this behavior, we initialize the solver with the reference representations from a random frame from either the Cityscapes dataset or the ImageNet-VID dataset and run StreamDEQ starting from the representations of that frame. We present the results of Cityscapes to Cityscapes shot change experiments in Figure 4.4 and ImageNet-VID to Cityscapes shot change experiments in Figure 4.5. The former of these experiments is simpler as the representations in two different videos from the same dataset are likely to be more similar. We notice that, for shot changes in similar contexts, *i.e.* Cityscapes to Cityscapes, the mIoU scores on initial frames are higher than our previous experiment in Figure 4.2 and also higher than the ImageNet-VID to Cityscapes shot change scenario in Figure 4.5. However, after the first few frames are processed, following a similar trajectory to the ones in Figure 4.2, mIoU scores stabilize at a value close to our original experiment. We conclude that, even with occasional shot changes, our method is able to adapt to the new scene in a few frames.

4.2 Video Object Detection

4.2.1 Experimental Setup

For the video object detection task, we evaluate our method on the ImageNet-VID dataset [60], a challenging video dataset with fast-moving objects, camera movement, and motion blur. We utilize the MMDetection [63] and MMTracking [64] frameworks for the implementation.

The ImageNet-VID dataset consists of 3862 training and 555 validation videos from 30 classes that are a subset of the 200 classes of the ImageNet-DET dataset. The frames and annotations for each video are available at a rate of 25-30 FPS per video. Note that the ImageNet-DET dataset consists only of images rather than videos. We follow the widely used protocol [37, 9, 65, 39, 40] and train our model on the combination of ImageNet-VID and ImageNet-DET datasets using the 30 overlapping

Table 4.1: Inference time comparisons of StreamDEQ with differing number of iterations on Cityscapes and ImageNet-VID datasets.

		Cityscapes		ImageNet-VID	
Model	# iterations	mIoU	FPS	mAP@50	FPS
StreamDEQ	1	45.5	4.3	9.1	10.3
StreamDEQ	2	57.9	2.9	39.5	9.2
StreamDEQ	4	71.5	1.9	50.4	6.2
StreamDEQ	8	78.2	1.1	54.8	3.5
MDEQ (Baseline)	27/26	79.7	0.3	55.0	1.2

Note: StreamDEQ model for the ImageNet-VID dataset is the StreamDEQ-26 model.

classes. We use a mini-batch size of 4, distributed to 4 NVIDIA A100 GPUs. We resize each image to have a shorter side of 600 pixels and train the model for a total of 7 epochs in 3 stages. We initialize the learning rate to 0.01 and divide it by 10 after epochs 2 and 5. We test the model on ImageNet-VID `val` and report mAP@50 scores following the common practice.

We adopt Faster R-CNN [66] by replacing its ResNet backbone with the MDEQ-XL model. To incorporate multi-level representations, we also use a Feature Pyramid Network (FPN) [67] module after MDEQ-XL. Without any additional modifications, we directly utilize the model while keeping the model hyperparameters and remaining architectural details the same as other Faster R-CNN models with ResNet backbones. Exceptionally, we only modify the number of channels for the FPN module to match that of MDEQ-XL. We start training with the ImageNet pretrained MDEQ-XL model from MDEQ [14].

We propose two numbers to be used as the number of Broyden iterations to train StreamDEQ. First, we pick 26, following the ImageNet classification experiments in MDEQ [14]. We name this model StreamDEQ-26. However, we find that with more iterations, we can obtain a more stable fixed-point and better performance on this challenging video object detection task. Therefore, we use 50 Broyden iterations for our second version (StreamDEQ-50).

UR-StreamDEQ uses the same architecture and training details. Again we have two schemes to train UR-StreamDEQ. Similar to StreamDEQ, we use a constant number of unrollings for the first scheme, which we set to 20. This is equivalent to applying the same layer 20 times to its output while injecting the input into the model every time. Secondly, we propose setting the number of unrollings stochastically. To clarify, for each training frame, we set the number of unrollings uniformly and at random between 1 and 20. We call this variant of UR-StreamDEQ, Stochastically Unrolled StreamDEQ (SUR-StreamDEQ).

Unlike many video object detection models [37, 65, 40], we train our models in the causal single-frame setting, meaning we do not use any temporal information for improved training.

4.2.2 Results

To the best of our knowledge, this is the first time an implicit model (MDEQ) has been used for a video object detection task. We achieve 55.0, 60.7, 70.8, and 70.2 mAP@50 with StreamDEQ-26, StreamDEQ-50, UR-StreamDEQ, and SUR-StreamDEQ, respectively on ImageNet-VID `val`. We are aware that Faster R-CNN with ResNet-50 backbone yields 70.7 mAP@50 off-the-shelf; however, Faster R-CNN is highly optimized to perform well with ResNet backbones. Yet, we use this same setting with an MDEQ without any parameter optimization, as our focus is not on constructing an MDEQ-based state-of-the-art video object detector. We believe there is room for improvement in detector design and tuning details, which we leave for future work.

Similar to the video segmentation task, we run StreamDEQ models with different numbers of iterations. We present the results of this experiment with StreamDEQ-26 in Figure 4.6 and StreamDEQ-50 in Figure 4.7. We observe the same trends with the segmentation task. Over time, detection performance increases and stabilizes at a value proportional to the number of Broyden iterations.

We note that, neither StreamDEQ-26 nor StreamDEQ-50 can produce any detection results in the non-streaming mode with 1 or 2 iterations. Instead, in streaming mode, if we perform 2 iterations with StreamDEQ-26, we improve the performance from

Table 4.2: Wall clock training time and single-frame performance comparisons of StreamDEQ models and Faster R-CNN with ResNet-50 backbone on the ImageNet-VID dataset.

Model	# iterations*	Training time (in hours)	mAP@50
StreamDEQ-26	26	114	55.0
StreamDEQ-50	50	130	60.7
UR-StreamDEQ	20	46	70.8
SUR-StreamDEQ	[1, 20]†	34	70.2
Faster R-CNN (ResNet-50)	-	9	70.7

$*$ denotes the number of Broyden iterations for StreamDEQ and the number of unrollings for (S)UR-StreamDEQ.
† indicates that the number of unrollings for each frame during the training of SUR-StreamDEQ is selected uniformly and at random between 1 and 20.

0 to 39.5 mAP@50 in 20 frames. We observe a similar pattern with StreamDEQ-50, *i.e.* the performance improves as the video progresses. However, even though the single-frame performance of StreamDEQ-50 is superior to StreamDEQ-26, the streaming performance for the lower number of iterations is worse. We believe this comes from performing more iterations during training. The model has more *time* to find a stable point during training that, with few iterations, it does not need to reach a near-optimal solution. Therefore, the step size per iteration gets smaller, but the overall quality of the fixed-point improves with StreamDEQ-50. Still, with more iterations per frame, *e.g.* 4 or 8, StreamDEQ-50 surpasses the streaming performance of StreamDEQ-26. While performing 8 iterations per frame, StreamDEQ-26 obtains 54.8 mAP@50, whereas, StreamDEQ-50 boosts this number to 58.9 in 20 frames. This allows us to choose between better performance in the short term or better performance over a longer video.

In addition, we also compare the inference speed of StreamDEQ with our baseline for different number of iterations in Table 4.1. In the 8-iteration case, we obtain a score of 54.8 with StreamDEQ-26, which is only 0.2 lower than our baseline model with 26 iterations while being almost 3 times faster. Likewise, with StreamDEQ-50 using 8 iterations per frame, we obtain a score only 1.8 lower than its computationally expensive single-frame counterpart.

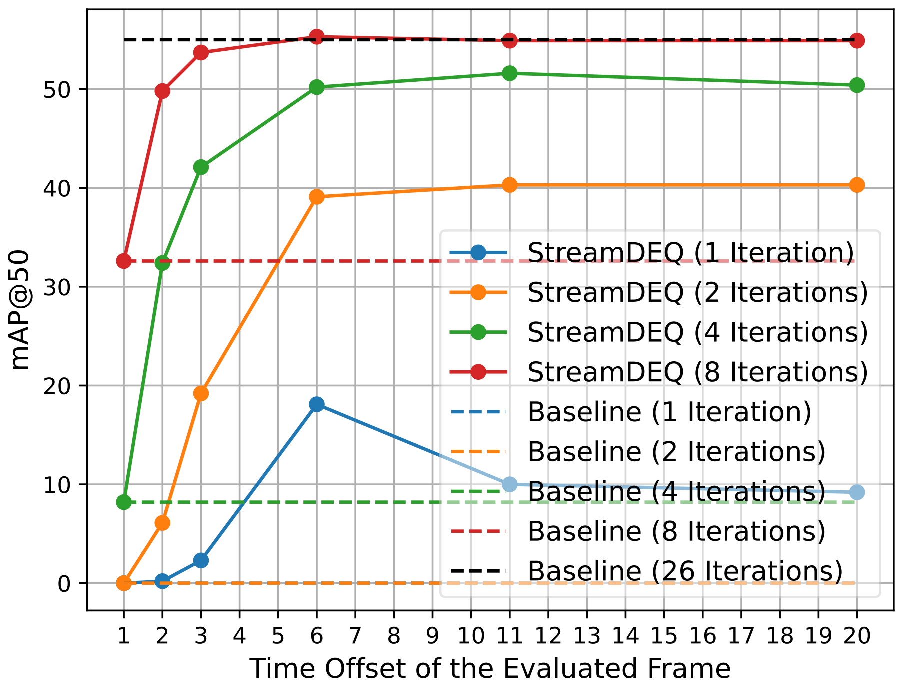

Figure 4.6: mAP@50 results of StreamDEQ-26 for various number of iterations after initialization with zeros from the beginning of a clip on the ImageNet-VID dataset.

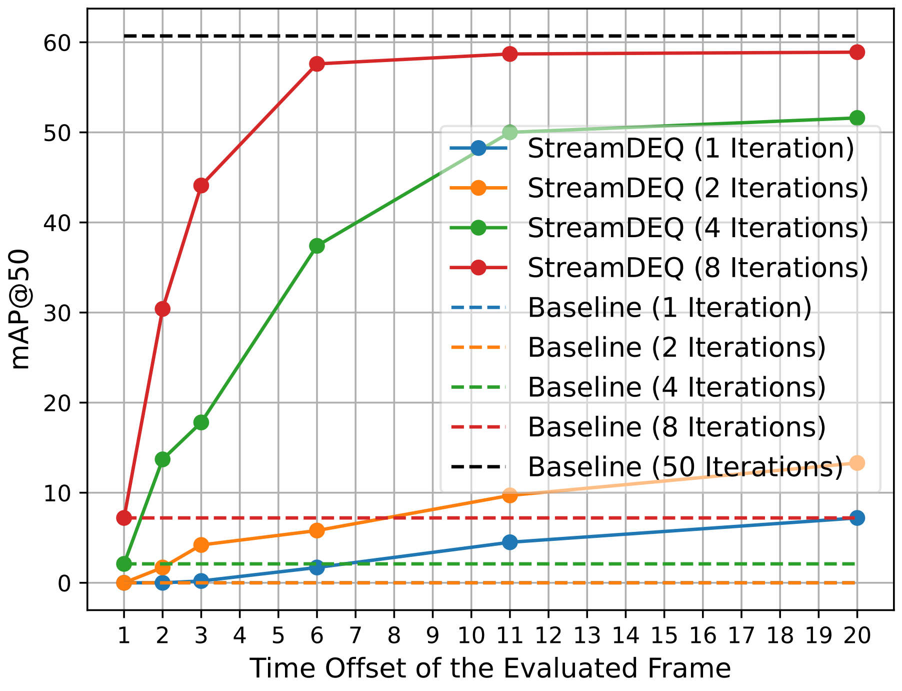

Figure 4.7: mAP@50 results of StreamDEQ-50 for various number of iterations after initialization with zeros from the beginning of a clip on the ImageNet-VID dataset.

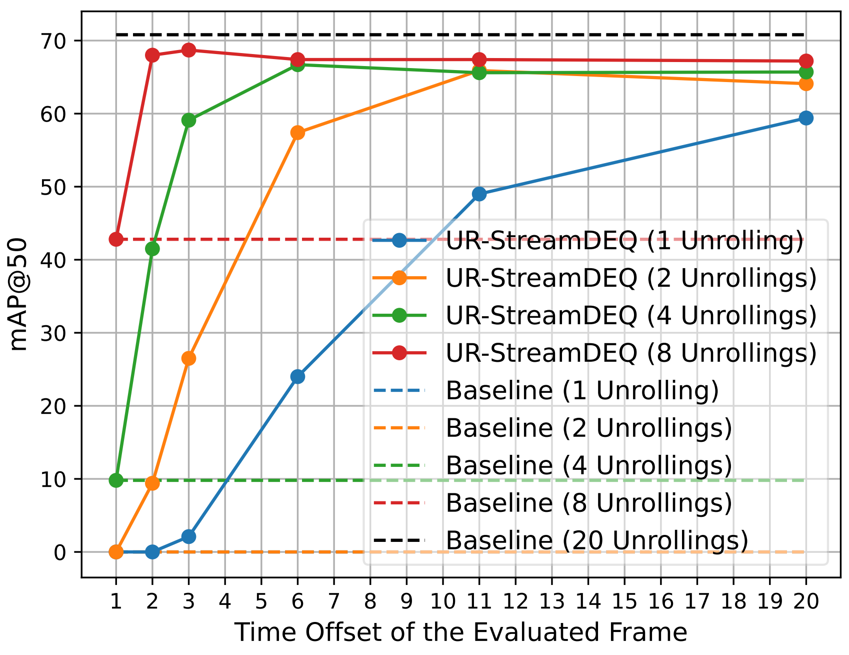

Figure 4.8: mAP@50 results of UR-StreamDEQ for various number of iterations after initialization with zeros from the beginning of a clip on the ImageNet-VID dataset.

We conduct these experiments for UR-StreamDEQ and SUR-StreamDEQ as well for the settings described in our experimental setup, the results of which we present in Figure 4.8 and Figure 4.9, respectively.

Our first observation is that the baseline single-frame performances of UR-StreamDEQ and SUR-StreamDEQ are higher than both StreamDEQ-26 and StreamDEQ-50. With UR-StreamDEQ, we achieve 70.8 mAP@50, which is 15.8 more than StreamDEQ-26 and 10.1 more than StreamDEQ-50. Furthermore, even with 1 unrolling per frame, both UR-StreamDEQ and SUR-StreamDEQ attain about 60 mAP@50 after 20 frames. We again discover that the scores stabilize at a value proportional to the number of unrollings.

One interesting matter is the performance comparison of UR-StreamDEQ and SUR-StreamDEQ, especially for the low number of unrollings. UR-StreamDEQ does not perform well initially for 1 unrolling per frame and is only able to produce good representations after $\sim$10 frames. This is closer to what we see with StreamDEQ-26 and StreamDEQ-50 in terms of performance. On the other hand, SUR-StreamDEQ starts with a strong 15.5 mAP@50 even after one frame with 1 unrolling. We attribute this

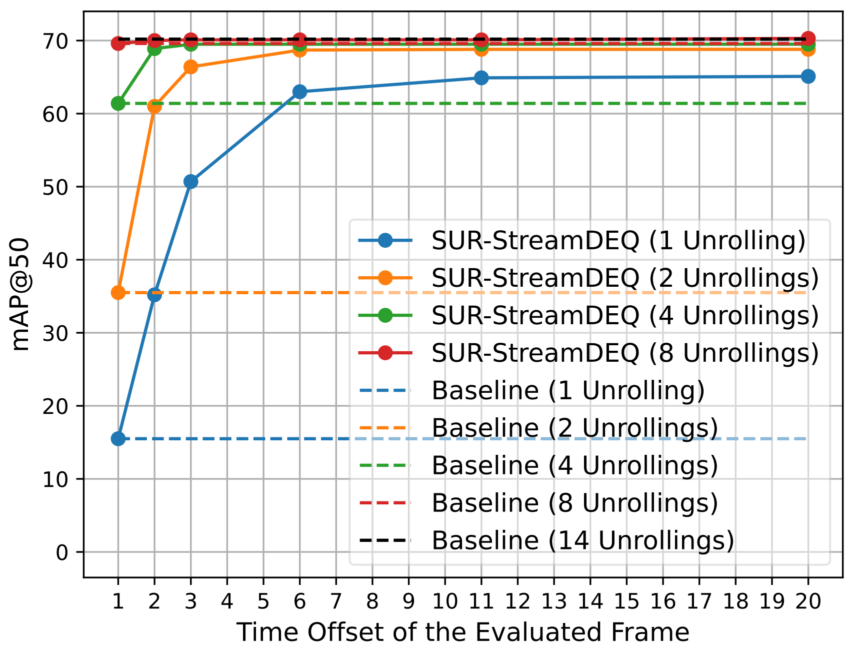

Figure 4.9: mAP@50 results of SUR-StreamDEQ for various number of iterations after initialization with zeros from the beginning of a clip on the ImageNet-VID dataset.

to the model's robustness to the noise introduced by the lower number of iterations as well as to the accumulation of information through time.

In light of this, we also try to simulate this behavior of SUR-StreamDEQ with UR-StreamDEQ by using again a constant but lower number of unrollings, *e.g.* 4, 8, to train the model. We find that the performance improves for the first few frames over that of UR-StreamDEQ, but after the first few frames, the scores diminish radically. For example, consider the case where the model is unrolled 8 times per frame during the entire training. When evaluated with 2 unrollings per frame, the model starts with good predictions and obtains over 65 mAP@50 in 4 frames. However, with the 5^{th} frame, the performance drops to $\sim$30 mAP@50, which means that the model cannot maintain high levels of accuracy for the whole video. Similarly, performance degrades after 8 frames if we employ 1 unrolling per frame during evaluation. Note that the point where the performance drops significantly coincides with the number of training unrollings. That is to say, if the model is trained with 8 unrollings, the streaming performance drops after the model unrolls 8 times (1 unrolling $\times$ 8 frames or 2 unrollings $\times$ 4 frames). We think this is because the model cannot learn more complicated structured representations, which require more unrollings. As a result, the model produces inaccurate representations and cannot find a precise fixed-point. If the model could reach a "good enough" fixed-point, then it would not have deviated from that equilibrium with minor changes to the input (*i.e.* successive frames in a streaming video).

On the contrary, SUR-StreamDEQ is able to sustain great performance with any number of unrollings once the initial frames are processed. We explain this huge spike in performance with the randomness during training. UR-StreamDEQ only learns about 20 unrollings per frame, whereas SUR-StreamDEQ has knowledge about a frame's representation after any number of unrollings. Therefore, it is able to adapt to more challenging situations (noisy representations), and its performance improves drastically. This also explains why the accuracy of UR-StreamDEQ starts to show a minor decline after $\sim 5 - 10$ frames, whereas SUR-StreamDEQ maintains its performance even after 20 frames with as low as 2 unrollings per frame.

Notice that the single-frame performance of SUR-StreamDEQ uses 14 unrollings. Other StreamDEQ variants use the maximum number of iterations/unrollings that they use during training to make full use of their potential. However, since the number of unrollings per frame is not deterministic with SUR-StreamDEQ, we observe its best single-frame performance using 14 unrollings, achieving 70.2 mAP@50. For example, with 20 unrollings, the score drops to 69.9 mAP@50. The variations between different numbers of unrollings are minor but still, this shows the effectiveness of the stochasticity by saving 6 unrollings per frame with a 0.3 performance improvement in the single-frame setting.

Additionally, SUR-StreamDEQ catches up to its single-frame performance with both 4 and 8 unrollings per frame when we evaluate it on streaming videos. Usually, the single-frame performance of any Stream-DEQ model acts as an upper bound to its streaming video performance. This makes sense since the models become accustomed to still images from the training phase, yet, we introduce motion during streaming evaluation. Thus, the performance should degrade slightly due to the injection of moving pictures at each time step. It is interesting to note that SUR-StreamDEQ manages to optimize in such a way that it can tolerate the "noise" coming from the shifting objects between successive frames and achieves 70.3 mAP@50 after 20 frames. This is 0.1 more than its single-frame baseline. This demonstrates that even though the performance only marginally improves, with the stochasticity we add to the model during training, the model becomes more effective in the low unrolling scenarios by learning diverse levels of representation complexities.

Finally, we compare the training and inference speeds of different StreamDEQ models in Table 4.2 and Table 4.3, respectively. We observe in Table 4.2 that explicit models obtain better performances and also train faster. The additional cost induced by the calculation of the inverse Jacobian during the backward pass slows the training of implicit models. For example, StreamDEQ-26 trains in approximately 5 days of wall clock time on 4 GPUs. On the other hand, the explicit variants of StreamDEQ train within 2 days of wall clock time. Table 4.3 shows that the explicit versions of StreamDEQ are almost twice as fast during inference with the same number of iterations/unrollings while demonstrating better accuracy. Furthermore, when we use SUR-StreamDEQ, we can achieve 65.1 mAP@50 with 23.2 FPS which is enough for most real-time applications.

Table 4.3: Inference time and streaming video performance comparisons of StreamDEQ models with differing number of iterations on the ImageNet-VID dataset.

Model	# iterations*	mAP@50	FPS
StreamDEQ-26	1	9.1	10.3
StreamDEQ-50		7.2	
UR-StreamDEQ		59.4	23.2
SUR-StreamDEQ		65.1	
StreamDEQ-26	2	39.5	9.2
StreamDEQ-50		13.3	
UR-StreamDEQ		64.1	17.4
SUR-StreamDEQ		68.8	
StreamDEQ-26	4	50.4	6.2
StreamDEQ-50		51.6	
UR-StreamDEQ		65.7	11.6
SUR-StreamDEQ		69.5	
StreamDEQ-26	8	54.8	3.5
StreamDEQ-50		58.9	
UR-StreamDEQ		67.2	6.8
SUR-StreamDEQ		70.3	

* denotes the number of Broyden iterations for StreamDEQ and number of unrollings for (S)UR-StreamDEQ.

CHAPTER 5

CONCLUSIONS

In this thesis, we proposed StreamDEQ, an efficient streaming video application of the multiscale implicit deep model, MDEQ. To the best of our knowledge, this is the first large-scale video application of DEQs. Furthermore, we introduced two explicit variants of StreamDEQ. We showed that our models could start from scratch (*i.e.* all zeros) and efficiently update their representations to reach near-optimal representations as the video streams. We validated this claim on video semantic segmentation and video object detection tasks with thorough experiments.

StreamDEQ models present a viable approach for both real-time video analysis and off-line large-scale methods. StreamDEQ is not specific to segmentation or object detection, and can be used as a drop-in replacement for most other structured prediction problems on streaming videos (action recognition, depth estimation etc.) as long as the prediction task involves an iterative fixed-point solution procedure. In addition, StreamDEQ is architecture-agnostic, meaning its architecture (f_θ) can be tuned to fit any criteria. The approach proposed in this thesis can easily be used in conjunction with current and future developments.

Future work directions include extending the theoretical analysis of MDEQ [14] from single images to videos and experimenting with different architectures to obtain better performances. Finally, as all variants of StreamDEQ are trained without temporal information, incorporating video training with the StreamDEQ scheme in order to improve performance is a promising future work direction.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[2] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[3] S. Zagoruyko and N. Komodakis, “Wide residual networks,” in *Proceedings of the British Machine Vision Conference (BMVC)* (E. R. H. Richard C. Wilson and W. A. P. Smith, eds.), pp. 87.1–87.12, BMVA Press, September 2016.

[4] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, “Deep feature flow for video recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2349–2358, 2017.

[5] X. Zhu, J. Dai, L. Yuan, and Y. Wei, “Towards high performance video object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7210–7218, 2018.

[6] Y.-S. Xu, T.-J. Fu, H.-K. Yang, and C.-Y. Lee, “Dynamic video segmentation network,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6556–6565, 2018.

[7] M. Liu, M. Zhu, M. White, Y. Li, and D. Kalenichenko, “Looking fast and slow: Memory-guided mobile video object detection,” *arXiv preprint arXiv:1903.10172*, 2019.

[8] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, *et al.*, “T-cnn: Tubelets with convolutional neural networks for object detection from videos,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2896–2907, 2017.

[9] S. Wang, Y. Zhou, J. Yan, and Z. Deng, "Fully motion-aware network for video object detection," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 542–557, 2018.

[10] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[11] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," in *ICLR*, 2015.

[12] J.-B. Cordonnier, A. Mahendran, A. Dosovitskiy, D. Weissenborn, J. Uszkoreit, and T. Unterthiner, "Differentiable patch selection for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2351–2360, 2021.

[13] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[14] S. Bai, V. Koltun, and J. Z. Kolter, "Multiscale deep equilibrium models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[15] C. G. Broyden, "A class of methods for solving nonlinear simultaneous equations," *Mathematics of computation*, vol. 19, no. 92, pp. 577–593, 1965.

[16] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[17] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang, "Seq-nms for video object detection," *arXiv preprint arXiv:1602.08465*, 2016.

[18] K. Kang, W. Ouyang, H. Li, and X. Wang, "Object detection from video tubelets with convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 817–825, 2016.

[19] J. Carreira, V. Patraucean, L. Mazare, A. Zisserman, and S. Osindero, "Massively parallel video networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 649–666, 2018.

[20] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons, and M. Zaharia, "Pipedream: generalized pipeline parallelism for dnn training," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pp. 1–15, 2019.

[21] M. Li, Y.-X. Wang, and D. Ramanan, "Towards streaming perception," in *European Conference on Computer Vision*, pp. 473–488, Springer, 2020.

[22] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[23] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

[24] M. Liu and M. Zhu, "Mobile video object detection with temporally-aware feature maps," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5686–5695, 2018.

[25] B. Zhao, B. Zhao, L. Tang, Y. Han, and W. Wang, "Deep spatial-temporal joint feature representation for video object detection," *Sensors*, vol. 18, no. 3, p. 774, 2018.

[26] H. Zhu, H. Wei, B. Li, X. Yuan, and N. Kehtarnavaz, "A review of video object detection: Datasets, metrics and methods," *Applied Sciences*, vol. 10, no. 21, p. 7834, 2020.

[27] R. Gadde, V. Jampani, and P. V. Gehler, "Semantic video cnns through representation warping," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4453–4462, 2017.

[28] P.-Y. Huang, W.-T. Hsu, C.-Y. Chiu, T.-F. Wu, and M. Sun, "Efficient uncertainty estimation for semantic segmentation in videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 520–535, 2018.

[29] S. Jain, X. Wang, and J. E. Gonzalez, "Accel: A corrective fusion network for efficient semantic segmentation on video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8866–8875, 2019.

[30] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell, "Clockwork convnets for video semantic segmentation," in *European Conference on Computer Vision*, pp. 852–868, Springer, 2016.

[31] Y. Li, J. Shi, and D. Lin, "Low-latency video semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5997–6005, 2018.

[32] F. Liang, T.-W. Chin, Y. Zhou, and D. Marculescu, "Ant: Adapt network across time for efficient video processing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2603–2608, 2022.

[33] P. Hu, F. Caba, O. Wang, Z. Lin, S. Sclaroff, and F. Perazzi, "Temporally distributed networks for fast video semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8818–8827, 2020.

[34] Y. Liu, C. Shen, C. Yu, and J. Wang, "Efficient semantic video segmentation with per-frame inference," in *European Conference on Computer Vision*, pp. 352–368, Springer, 2020.

[35] Y. Liu, C. Shen, C. Yu, and J. Wang, "Efficient video segmentation models with per-frame inference," *arXiv preprint arXiv:2202.12427*, 2022.

[36] A. Habibian, H. B. Yahia, D. Abati, E. Gavves, and F. Porikli, "Delta distillation for efficient video processing," *arXiv preprint arXiv:2203.09594*, 2022.

[37] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-guided feature aggregation for video object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 408–417, 2017.

[38] G. Bertasius, L. Torresani, and J. Shi, "Object detection in video with spatiotemporal sampling networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 331–346, 2018.

[39] H. Wu, Y. Chen, N. Wang, and Z. Zhang, "Sequence level semantics aggregation for video object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9217–9225, 2019.

[40] Y. Chen, Y. Cao, H. Hu, and L. Wang, "Memory enhanced global-local aggregation for video object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10337–10346, 2020.

[41] Z. Liu, H.-J. Wang, Z. Xu, T. Darrell, and E. Shelhamer, "Confidence adaptive anytime pixel-level recognition," in *International Conference on Learning Representations*, 2022.

[42] L. Bazzani, N. de Freitas, H. Larochelle, V. Murino, and J.-A. Ting, "Learning attentional policies for tracking and recognition in video with deep networks," in *ICML*, 2011.

[43] M. Denil, L. Bazzani, H. Larochelle, and N. de Freitas, "Learning where to attend with deep architectures for image tracking," *Neural computation*, vol. 24, no. 8, pp. 2151–2184, 2012.

[44] H. Rhee, D. Min, S. Hwang, B. Andreis, and S. J. Hwang, "Distortion-aware network pruning and feature reuse for real-time video segmentation," *arXiv preprint arXiv:2206.09604*, 2022.

[45] A. Habibian, D. Abati, T. S. Cohen, and B. E. Bejnordi, "Skip-convolutions for efficient video processing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2695–2704, 2021.

[46] Y. Chai, "Patchwork: A patch-wise attention network for efficient object detection and segmentation in video streams," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3415–3424, 2019.

[47] B. E. Bejnordi, A. Habibian, F. Porikli, and A. Ghodrati, "Salisa: Saliency-based input sampling for efficient video object detection," *arXiv preprint arXiv:2204.02397*, 2022.

[48] Z. Huang, S. Bai, and J. Z. Kolter, "(Implicit)^2: Implicit layers for implicit representations," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[49] S. Bai, V. Koltun, and J. Z. Kolter, "Neural deep equilibrium solvers," in *International Conference on Learning Representations*, 2021.

[50] A. Pal, A. Edelman, and C. Rackauckas, "Mixing implicit and explicit deep learning with skip deqs and infinite time neural odes (continuous deqs)," *arXiv preprint arXiv:2201.12240*, 2022.

[51] S. Bai, V. Koltun, and J. Z. Kolter, "Stabilizing equilibrium models by jacobian regularization," in *International Conference on Machine Learning (ICML)*, 2021.

[52] Z. Geng, X.-Y. Zhang, S. Bai, Y. Wang, and Z. Lin, "On training implicit models," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24247–24260, 2021.

[53] S. W. Fung, H. Heaton, Q. Li, D. McKenzie, S. Osher, and W. Yin, "Jfb: Jacobian-free backpropagation for implicit networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[54] S. Bai, Z. Geng, Y. Savani, and J. Z. Kolter, "Deep equilibrium optical flow estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 620–630, 2022.

[55] C. Lu, J. Chen, C. Li, Q. Wang, and J. Zhu, "Implicit normalizing flows," in *International Conference on Learning Representations*, 2021.

[56] L. Ma, T. Wang, B. Dong, J. Yan, X. Li, and X. Zhang, "Implicit feature refinement for instance segmentation," in *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 3088–3096, 2021.

[57] T. Wang, X. Zhang, and J. Sun, "Implicit feature pyramid network for object detection," *arXiv preprint arXiv:2012.13563*, 2020.

[58] S. Bai, J. Z. Kolter, and V. Koltun, "Trellis networks for sequence modeling," in *International Conference on Learning Representations*, 2019.

[59] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser, "Universal transformers," in *International Conference on Learning Representations*, 2019.

[60] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[61] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[62] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[63] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "MMDetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.

[64] M. Contributors, "MMTracking: OpenMMLab video perception toolbox and benchmark." `https://github.com/open-mmlab/mmtracking`, 2020.

[65] J. Deng, Y. Pan, T. Yao, W. Zhou, H. Li, and T. Mei, "Relation distillation networks for video object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7023–7032, 2019.

[66] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[67] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature

pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.