2019

Conditional Dilated Attention Tracking Model - C-DATM

Tyler Clayton Highlander
Wright State University

Conditional Dilated Attention Tracking Model - C-DATM

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

by

Tyler Clayton Highlander
B.S.C.S., Wright State University, 2014
M.S., Wright State University, 2015

2019
Wright State University

WRIGHT STATE UNIVERSITY

GRADUATE SCHOOL

July 15, 2019

I HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER MY SUPERVISION BY Tyler Clayton Highlander ENTITLED Conditional Dilated Attention Tracking Model - C-DATM BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Mateen Rizki, Ph.D.
Dissertation Director

Michael Raymer, Ph.D.
Director, Computer Science
and Engineering Ph.D. Program

Barry Milligan, Ph.D.
Interim Dean of the Graduate School

Committee on
Final Examination

Mateen Rizki, Ph.D.

John Gallagher, Ph.D.

Michael Raymer, Ph.D.

Fred Garber, Ph.D.

Bernard Abayowa, Ph.D.

ABSTRACT

Highlander, Tyler Clayton . Ph.D., Department of Computer Science and Engineering, Wright State University, 2019. *Conditional Dilated Attention Tracking Model - C-DATM.*

Current commercial tracking systems do not process images fast enough to perform target-tracking in real- time. State-of-the-art methods use entire scenes to locate objects frame-by-frame and are commonly computationally expensive because they use image convolutions. Alternatively, attention mechanisms track more efficiently by mimicking human optical cognitive interaction to only process small portions of an image. Thus, in this work we use an attention-based approach to create a model called C-DATM (Conditional Dilated Attention tracking Model) that learns to compare target features in a sequence of image-frames using dilated convolutions. The C-DATM is tested using the Modified National Institute of Standards and Technology handwritten digits. We also compare the results achieved by C-DATM to the results achieved by other attention-based networks like Deep Recurrent Attentive Writer and Recurrent Attention Tracking Model that appear in the literature. C-DATM builds on previous attention principles to achieve generic, efficient, and recurrent-less object tracking. The GOTURN(General Object Tracking using Regression Networks) model which won the VOT 2014 dataset challenge contains similar operating principles to C-DATM and is used as an exemplar to explore the advantages and disadvantages C-DATM. The results of this comparison demonstrate that C-DATM has a number of significant advantage over GOTURN including faster processing of image sequences and the ability to generalize to tracking new targets without retraining the system.

Contents

List of Figures

List of Tables

Acknowledgment

I would like to take this opportunity to extend my thanks to my advisor Dr. Bernard Abayowa and Dr. Mateen Rizki for the years of mentorship and wisdom.

I would also like to express my great thanks to Dr. Michael Raymer, Dr. John Gallagher, and Dr. Fred Garber. I grew a large amount dealing with your off topic and ”off-ramp” questions.

Dedicated to

You, yes you, you know who you are. Always.

1 Introduction

Many of the top neural network tracking models use brute force to track targets [33] [8] [22]. Every new frame is fully explored in order to find the target once again. Convolution is a popular deep learning architecture used in top tracking models. Convolution has been empirically shown to achieve accurate results in visual classification tasks [23]. However, a major limitation of convolution-based techniques is the speed at which the networks perform their calculations. Due to the nature of a traditional convolution, models employing them compute at less than a frame per second [33] [8]. Work has been done on speeding up these calculations[15] and these techniques can be used in parallel with the techniques developed in this research.

If tasked with tracking a shooting star through the night sky a human would give visual attention to the shooting star. Once focused on the star, the only task is to keep that attention on the star as it moves. Attention-based methods attempt to implement this biological concept. By focusing attention on small regions, the attention-based neural networks are able to perform at higher speeds without losing benefits of convolutions. This research combines attention-based methodologies with an alternative approach to implementing convolutions to create a time efficient, accurate model.

1.1 Background of the Problem

Tracking is an age old problem and continues to have important applications. Currently, tracking applies to fields from national security to entertainment. Technology is evolving to power self-driving cars [16] and artificially intelligent systems to solve many problems. A realistic form of tracking is based on kinematics [28], which predicts the movement of objects in a scene based on a series of physics related equations.

Advancement in technology allowed neural network based techniques to be developed over standard kinematic models. Some of the best-performing tracking systems [33] [8] currently are neural network based [23]. Major downsides to these methods are their complexity, need to analyze entire images, and lack of ability to store track history(*i.e.* lack of memory). Neural network-based tracking systems are the state-of-the-art in terms of tracking performance but are computationally expensive and often process image sequences at less than one frame per second [33] [8].

Attention mechanisms attempt to mimic the human vision system. By only focusing on a small portion of the image the complexity and amount of data processed by the network is reduced so the system can process image-sequences at a higher frame rate. Attempting to recreate human systems attention mechanisms addresses many of the current limitations of the top tracking methods.

This work improves on the state-of-the-art attention-based tracking systems applied to moving MNIST targets[27]. MNIST consists of a collection of hand-written numeric digits from zero to nine. For our model evaluations these digits are randomly put in a larger scene where they move from frame-to-frame. There are tests containing multiple digits, where each digit travels independently. At the boundaries, the digits are reflected back into the scene simulating the sense of an object bouncing off a wall. In multi-object scenes, objects do not interact (*i.e.* object do not appear to bounce of each other). This data allows for direct comparisons to current state-of-the-art attention-based tracking models presented in the literature[18].

Our research uses components from existing models [13] and applies other techniques not found in current attention-based models[14][2] to improve both the tracking performance and the computational complexity compared to prior work[18][13][14]. Kahou *et al.* [18] makes the progression from attention mechanisms to tracking as described previously. Held *et al.* [14] creates a generic model that is non-existent in attention-based model literature, until this work. Bai *et al.* [2] uses dilated convolutions to increase the receptive fields of the convolutions, and can be viewed as a new type of memory that does not use recursive networks.

1.2 Problem Statement

The shortcomings of current neural network tracking systems are three-fold. The first major obstacle with state-of-the-art neural network tracking systems is speed. Neural network approaches track objects, however; they typically process video at less than a frame per second[33] [8]. The lack of speed of these networks is attributed to the number of free parameters in the system, *i.e.* complexity. Convolutional neural networks scale linearly with the size of the scene being analyzed and process the entire scene in order to predict the next bounding box. As size of the scenes grow, the complexity of these systems also grow linearly. Slow speeds make these models non-realistic in practice. An important contribution would be to significantly improve the processing speed of neural network tracking systems so they can be applied to real-world problems.

The second potential drawback with current models is the lack of memory in the system can lead to inferior performances to models that include memory. Memory can have a great impact on future decisions when tracking objects. Having a model that contains memory will allow for previous decisions to influence future decisions. Humans have the ability to remember how their past decisions affect their current state. Traditional neural network memory uses long short-term memory (LSTM) or a gated recurrent units (GRU).

This system will use dilated convolutions to learn correlation filters over a given time series so that an initial state can be pre-defined. Using memory in general will allow the network to be more robust and make better predictions of tracked object locations.

The third shortcoming can be found in the data. Environments are not clear images with an object that is easily identified compared to the background of the scene. Most real world situations come with clutter, including other objects in the scene or a background that blends with the object. Clutter currently is processed by convolution models due to their requirement to analyze the whole scene. Attempts can be made to deal with the clutter in convolutional systems, but this increases the complexity of the system. Either way, the system is hindered due to clutter in scenes. Occlusion is also a very prominent issue within the tracking community. Lack of generality can cause poor performance when the target is occluded. Generality is the system's ability to extract features to measure object similarity. Convolutional networks have a built-in ability to generalize using pooling layers, which has led to the success of these networks for feature generation. However; when object features are mixed with features present in the background of the scene the convolution layers may not be enough to discern the object features from the features in the scene.

1.3 Purpose of the Study

In this study we introduce the Conditional Dilated Convolution Attention Tracking Model (C-DATM). The C-DATM improves on state-of-the-art tracking systems in three different aspects. The first aspect of this study is to speed up the tracking process compared to other neural network models. Our goal is to process between 30 and 100 frames per second while using standard attention-mechanisms. Secondly, this study will use memory to allow for previous decisions to affect future ones in an attempt to create a more robust tracker. Lastly, the effects of occlusion on attention-based models will be a explored as part of this study. Previous research [13] [44] points towards attention models having a better ability to deal

with occlusion.

1.4 Research Questions

This study will answer three major questions:

1. By adding dilated convolutions as a format of memory and reformatting to an attention problem, can a more robust model be created, removing the need for recurrent systems? These questions will be answered by comparing the performance of our model to the past models, and through data-manipulation experiments. The memory layer will generate kernels, or filters, that will be visualized to understand our methods further.

2. Can the use of an attention-based model lead to acceptable performance speeds (At least 30 frames per second)? Speed will be judged based on frames per second(FPS) during training and testing as others [14] have.

3. Can an attention-based model be adjusted so that given the first frame's object bounding box, the model can generically track the desired object in a scene? Attention-based models are currently class based and do not transfer to new classes. MNIST contains ten different classes, and using a single model for testing the entire dataset fundamentally means the model is not single class based.

1.5 Significance of the Study

Dilated convolutions as a form of memory allow the system to better perform, aiding in the robustness of the features extracted. A more robust model needs less interaction with an advisor allowing real-world applications. With better performing networks real-world end-to-end systems are more likely to be developed, and applicable results is a major driving factor in this study.

Improving the processing speed of the model will facilitate development of practical systems that are applicable to many real-world problems. These systems will perform at

speeds that allow for their integration into existing security networks, self-driving cars, or even entertainment hardware like the Kinect. Creating an end-to-end system that tracks targets of interest at over 30 frames per second will allow these systems to be used for security with less overhead. Security networks can be developed at travel locations, such as airports, that require no human advisor, making the environment safer and more convenient for the travelers.

To date, there are no practical attention-based systems capable of generalization to track an arbitrary target without re-training the system. C-DATM utilizes a generic model capable of tracking a set of related classes of targets thus, saving time on training, human experts, and redeployment. Generality is a must for real-world applications as these systems will constantly be introduced to unique but similar objects to those previously seen. Adding the ability for the network to generalize will allow for tracking of new objects and will create systems that have a wide applicability.

1.6 Definition of Terms

The goal of C-DATM is to track macro-objects. Macro-objects are defined as objects that do not contain micro-movements. For example, C-DATM is designed to track objects like entire human beings moving through a scene. The model does not focus on learning how to track micro-movements such as hand gestures. A scene refers to the entire image-frame within a video sequence and a down-sample is a region of interest or focus of attention, sometimes at a lower resolution. The tracking performance of C-DATM was tested in a series of experiments using image-sequences containing MNIST digits. The accuracy of the C-DATM was measured taking the average intersection over union of the object and the bounding box containing the object.

1.7 Assumptions, Limitations, and Delimitations

The C-DATM is based on certain assumptions. By attempting to create a model that is as fast as possible the network will use only information from the previous bounding box to make a decision about the next bounding box in the frame. Using attention-based models the bounding box size can fluctuate, and the bounding box must contain some of the object being tracked to successfully predict the next bounding box. Without partial information of the object present in the next frame, the tracker will most likely lose its track. This principle also applies to occlusion, if the object is completely occluded the model will most likely lose the track. Another assumption forces the confusers in the scene to at least be somewhat discernible from the object of importance. If the object being tracked and the confuser enter the area of the bounding box, the confuser must contain some information that allows the model to tell it apart. The model may switch its track if the objects are the same after some occlusion of the two objects.

1.8 Conclusion

This study will work to improve on current tracking models by using attention-based mechanisms. Attention mechanisms will allow for quick processing of the scene, better handling of occlusion, and memory based tracking. The goal of this study is to create systems that can be used to solve real-world tracking problems such as those that arise in security applications.

II. Literature Review

2.1 Historical Background

This literature review discusses the science supporting attention-based mechanisms, what attention models are, and how they process data. Also discussed; current applications of attention that are, and are not, localization or tracking, and a non-attention-based neural network approach that gives insight to neural network kinematic tracking. These topics are foundation of the design decisions that led to the development of C-DATM. The literature review focuses on attention-based models' ability to process scenes quickly, track, deal with problems such as occlusion, and scale to larger problems. The model will answer three major questions: (1) Will adding memory to a state-of-the-art neural network create a more accurate system? (2) Can a smaller region of the image be analyzed leading to faster computations than current neural network tracking systems? (3) Can a generic attention-based model be created?

Attention mechanisms are founded on the principle that humans scan scenes by focusing their attention on particular regions until the desired object is found. Ranzato [35] states that convolutional networks process the entire scene, not putting emphasis on any particular region, much like the way a bee processes its surroundings. The systems are also equivariant rather than invariant, which means spatial transformations and scale changes are reflected in the low-level features. Some methods have achieved invariance by computing histograms of local features over large regions at different locations and scales [25].

Mnih [31] quotes references [37] suggesting humans are task based in their observations and must focus on particular regions for the best recall.

Erichsen proposed there are about 40 types of vision systems in nature [10]. All are designed through nature-based constraints and evolution. These systems are optimized to deal with physical limitations of the environment. However, human vision does not seem to have constraints based on classification of objects due to human's ability to work in unrestricted class domains. The human visual system's performance on unrestricted domains has prompted the design of artificial systems using human traits. The human visual system is foveated, the density of the cones around the fovea is much larger than other areas. Rensink [37] also states the gathering of visual information is done using a retina that has a high resolution only over a few degrees of visual angle. A complete representation of a scene, therefore; requires the contents of individual eye fixations to be integrated via a high-capacity visual buffer. According to Rensink, focused attention is intimately involved with the perception of objects. Ranzato[35] set up learning experiments focussing on regions of attention, which is the backbone of this research. Gregor [13] agrees on the importance of deciding where to look, and states that learning where to look is the main challenge of sequential attention models.

2.2 Attention-Based Works

The experiments done by Caicedo [4] show that attention models have the ability to localize objects in an image while only being privy to the object's class. The principles of Caicedos model are very similar to the principles being used throughout the attention community. A scene is analyzed as a whole to find a starting location and the systems narrows down the size of the scene and the location of its focus until an object is localized. This process has been extended to tracking by Kahou *et al.* [18].

Attention-based methods attempt to reduce the amount of data evaluated at critical mo-

ments to produce faster models. Gonzalez-Garcia *et al.* [12] accelerated category-specific R-CNN detectors. Lampert *et al.* [24] found high scoring regions only evaluating a few locations using their branch-and-bound algorithm. Attention-based methods are also being used for recognition tasks. Xu *et al.* [44] generates captions for images using attention-based recurrent neural networks. The captions interestingly are associated with regions of attention giving extra context and adding information to the understanding of the problem. Mnih [31] and Ba [1] also used attention-based methods to find important regions to analyze at a higher resolution due to their model's ability to quickly process data.

Analyzing the entire scene to start can be very costly when considering the scale of the system. Two general methods have been developed for initialization of these systems that are more suited to scaling objects. The first idea has become popular in facets of neural networks over the past few years and is usually referred to as transfer learning. Sermanet [39] and Tang [41] use transfer learning, which is the use of features from already developed neural networks, that have been successful, to initialize their networks. Transfer learning has been shown to have success like in Sermanets and Tangs works. Another method has been used and may be more applicable to this problem and future real world applications. The initial state of the model developed by Ranzato [35] uses a lower resolution version of the original image in order to allow scalable calculations. After the initialization is done via the down-sampled image the original resolution is used when analyzing future smaller glimpses created by the network.

A major impact of Mnih and Xu [44] in the recurrent model field relates directly to scalability. Many researchers have been using convolutional neural networks due to their recent success on visual tasks [23]. Ranzato states the best image recognition systems are convolutional neural networks [23][26]. Convolutional neural networks are trained on small image patches and then unrolled over larger images at test time [42]. Mnih states that convolutional models scale linearly with the amount of pixels used in the data. This scaling factor prevents convolutional neural networks from processing large images efficiently.

Recurrent models do not suffer from this problem which is the reason attention-based networks currently use them. Further developments of the idea of attention networks improve as they are not dependent on the size of the image, as confirmed by Ba [1].

Sermanet [39] makes a statement that most problems solved using attention-based methods have been toy problems. However, Sermanet's system produced state-of-the-art results on fine-grained categorization. Sermanet claims that using the attention-based model allowed for simultaneous localization and classification of objects within the scene. This is accomplished with the fundamentals of attention methods. The network learns to localize a certain class, subsequently classifying the localized object. Sermanet claims to be successful with attention-based methods even though the data has difficult class boundaries, large variations in pose and lighting, varying and cluttered backgrounds, and occlusion. Sermanet predicts an attention model could learn to focus processing power on the specific features of the objects that help to tell them apart, for example, the face, ears, and particular fur patterns for dogs. Future versions of this model could potentially also choose the scale at which to examine details. Sermanet's work[39] demonstrates the potential of attention-based networks applied to difficult problems.

Ba [1] explored the task of recognizing multiple objects in a scene using convolutional networks. Ba also agrees that convolutional networks have poor scalability with increasing image sizes. The attention network in Ba's work is used to extract useful features from a region and uses transfer learning for initialization. Multiple object detection is important in the field of visual systems for practicality and usefulness and shows the power of attention-based mechanisms. With the success of Ba's work, empirical evidence dictates that attention networks have the power and scalability to be used in state-of-the-art systems.

A less related problem, but interesting due to difficulty: Tang [41] has done work in generative models with attention. Tang uses a convolutional network for transfer learning to initialize the network. The importance of generative models is high due to occlusions in many sets as well as missing information. Occlusions are very common in realistic settings

and have been largely ignored in recent literature of deep learning according to Tang.

2.3 Attention-based Tracking

Attention-based networks have shown an ability to focus on salient objects in an image. Xu [44] developed a model that automatically learns to describe content in images. The important advancement in this paper is the deterministic quality of the learning algorithm which allows back-propagation to be used. Back-propagation gives credit to or punishes specific subsets of neurons within the network. Xu states that attention allows salient features to dynamically come to the forefront as needed. This model has a unique ability to track what the system is looking at when captions are being generated.

In addition to attention-based networks showing promise in localization and classification, the networks have also recently been used in generative model creation. Gregor [13] created a network, Deep Recurrent Attentive Writer(DRAW), that mimics the foveation of the human eye, like all other attention networks, and improved performance for generative models when applied to the Mixed National Institute of Standards and Technology (MNIST)[27] dataset. This network uses a differentiable attention mechanism that benefits from gradient descent learning. Gregor *et al.* [13] algorithm, coined Read, translates neural network outputs to an attention mechanism. A recurrent neural network was used to learn 5 parameters for a grid of Gaussians to down-sample the image; x, y, σ, δ, and γ. The x and y are distances from the edge of the image to the center point of the attention region in either direction. The δ is the distance between Gaussians in the grid, and the σ is the standard deviation of the Gaussians in the grid. The γ is a shading factor only needed for generative models. The down-sampled image is the region of focus for the network.

The first glimpse is always the same with DRAW by using parameter values of zero (0). After which, the network uses the glimpse as the only input to the recurrent neural network (RNN) to generate the next set of attention parameters. In training, a set number

of glimpses are used. In the case of DRAW's localization experiments, the final glimpse is compared to the truth and the error is back-propagated through the network. Using only the final position of the attention parameters the network can learn to accurately localize a class over a set of four glimpses.

Along with classification success, the attention mechanism allows for only a small portion of the image to be processed at any given time. The localization found on cluttered MNIST data in DRAW can be processed in real time. Gregors experiments showed promising results on localization in cluttered environments, as well as creating accurate generative models by aggregating multiple glimpses together, much like many psychologists believe our eye and brain communicate [37]. DRAW does not extend strictly to tracking problems and the localization tasks are class-based models.

2.4 Recurrent Attention Tracking Model

Tracking can be extended from the attention model. By simulating video via a fixed number of image frames the attention mechanism will localize the object of interest. As the object moves from frame-to-frame the attention mechanism keeps the focus on the object, effectively tracking the object. This was first published by Kahou[18] when they presented the RATM: Recurrent Attentive Tracking Model.

RATM used a very similar method to the READ function for their attention model with a few minor changes. One of the most interesting changes is RATM learns δ and σ in both the x and y directions. By learning the extra parameters RATM is allowing the network to find rectangular attention fields rather than the square regions of DRAW. RATM also used a different normalization from DRAW. The parameters δ and σ are forced to be positive values by using the absolute values of the learned parameters. The specific motivation for this modification was discovered during experimentation when it was observed the δ and σ parameters often saturated at low values, causing the attention window to zoom in on a

single pixel. RATM also used piece-wise linear activation functions have been shown to benefit optimization [32] and the absolute value function is a convenient trade-off between the harsh zeroing of all negative inputs of the rectified linear unit (ReLU) and the extreme saturation for highly negative inputs of the exponential function.

RATM experimented on different data sets, from tracking a bouncing ball to the recognition of human actions on the KTH dataset [38]. Using attention methods allows for all of these experiments to perform in real time. A number of important observations were made using RATM including the model can be trained on a noisy bounding box annotation of videos and recover from this noisy initialization during testing. Kahou suggested the recovery might be related to the pre-training of the feature-extraction module on static images. The information about the appearance of humans is transferred to the attention module, which learns to adapt the horizontal and vertical strides among other parameters of the glimpse to match this appearance. Another observation; the trained human tracker seems to generalize to related but more challenging data. This is an interesting aspect of the applicability of the network in real world situations. The generalization to harder data is a key point that this research will build upon when creating a general attention model. RATM however, does not perform at greater than 50% performance, measured as the intersection over union, for 2-digit MNIST tracking. Meaning, the network may not be able to differentiate between two classes. To accurately track multiple objects the model would have to be trained for each specific class, making the network non-general. RATM also uses harsh absolute values to battle vanishing bounding boxes. A simple minimum box size can solve the problem and give the parameters a larger search domain.

2.5 Generic Object Tracking Using Regression Networks (GOTURN)

Held *et al.* [14] proposed a deep regression network that performs at 100 frames per second (FPS), coined GOTURN. Held states that most of the community use on-line trackers trained during tracking execution; this does not benefit from large amounts of data available for offline training. This model is a neural network approach to learning in kinematic tracker fashion via learning a generic relationship between object motion and future position. According to Held, their approach is the first neural network tracker that learns to track generic objects at 100 FPS. Generic trackers are not trained for tracking a specific class. This tracker should be able to track any object given the initial starting location of the object. However, if the experiments have multiple objects that occlude each other over multiple frames the network will lose the ability to know which of the objects is the desired target.

Having the benefit of offline training the tracker can learn to handle rotation, changes in viewpoint, lighting changes, and other complex challenges according to Held. 100 FPS is also a benefit to using offline training because neural networks train slowly making on-line trackers extremely slow [33] [8] [22]. Held's [14] tracker uses a regression-based approach, requiring just a single feed-forward pass through the network to regresses directly to the location of the target object. The network uses CNNs to extract features from the first frame and compares those features with a cropped portion of the image. Fully connected layers learn the relationship between the first frame features and the cropped portion to produce a bounding box for the next frame. This method does not follow current attention-based principles and without memory can struggle with multiple object occlusions.

2.6 Past Work in Kinematic Tracking

Kinematics revolve around teaching an attention-based model to learn the acceleration and objects direction motion. Most models currently use class based trackers, however; kinematic trackers can be generic. Li's [28] survey paper analyzes various mathematical models of target dynamics proposed for maneuvering target tracking, including 2D and 3D maneuver models, as well as coordinate-uncoupled generic models for target dynamics. The principles of kinetic tracking are included in Held's work [14] as the network is learning how to compare two sets of features to estimate future bounding boxes. Both Held's model and this work will be subject to kinematic experiments to measure importance of the different types of features.

2.7 Past work in Dilated Convolutions

Dilated convolutions increase the receptive field of standard convolutions without increasing the number of free parameters. Larger receptive fields allow for faster application of the kernels to the image, producing a less complex system that forward propagates at a desirable rate. If the dilation is taken over the time dimension Bai *et al.* [2] empirically determined that dilated convolutions functions as a form of memory. Dilated convolutions can be ran in parallel, have flexible receptive fields, create stable gradients, have a low memory requirement during training, and support variable length inputs. These properties make dilated convolutions a viable alternative to standard recurrent memory. This work will compare the results of dilated convolutions for memory to RATM, which uses recurrent methods like previous attention-based models.

2.8 Joint Inference and Control

In this research Joint Inference and Control (JIC) [11] is used. JIC uses information from a tracking model to control the degrees of freedom of an observation system. Attention-based mechanisms lend themselves extremely well to JIC. The attention mechanisms work as the observation technique, typically a camera, on the scene. The scene is the maxium viewable space of the camera which is simulated using the entire image in these experiments. The attention region, or bounding box, is the area the camera is currently focused on.

Visual systems are becoming more abundant each year making JIC an emerging field. Systems are being developed in order to track targets of interest automatically, track over large areas, or even control self-driving cars. A major hindrance to the development of these systems, mentioned by Kamal [19], is the lack of communication in current methodology. To develop a system that can monitor a target over a large area the cameras must communicate in some fashion to continuously track. Kamal states that most systems are linear and not connected, while the reality of scene analysis is non-linear and therefore connectivity is important. The largest concern with communication, and why most systems are not able to incorporate communication, is the issue of scalability. With multiple cameras and a communication infrastructure, the amount of information being processed is great. Current methods that use probabilistic models or other non-neural network implementations, like Kamal [19] and Marques [30], require large amounts of information to be processed to achieve a working solution. This takes time, and as the complexity grows so does the computation needed. This lack of communication mechanisms and needed to process massive amounts of data limit the applicability of most current systems, as they would take too long to process the scene and the target would move too quickly for the system to maintain an accurate track. The model created in this work is scalable, and has built in communication with the previous position of the bounding box being an input to the network.

Learning a degree of freedom control parameter has spread throughout the visualiza-

tion community. Once JIC was embraced by the neural network community, reinforcement learning began to become popular. Mnih [31] developed a model using recurrent neural networks that have a degree of translation invariance parameter, and use reinforcement learning. This particular model is not differentiable. The learning works on a reward and punishment system. More recent work has been done in reinforcement learning using differentiable models [29]. Reinforcement learning is novel and many argue that it functions how humans learn, which is a primary goal among the neural network community; however like humans, Mnih's model is not calculus based and does not use mathematics to estimate error propagations using back-propagation. C-DATM is based in rigorous mathematics and the learning algorithm is differentiable, meaning the learned network is mathematically sound.

2.9 Performance Measures

The performance measure for popular tracking competitions [22] is accuracy based on the overlap of bounding boxes with a truth bounding box. The accuracy measure is calculated as the intersection over union (IOU)[5]. IOU is computed as the intersection between the predicted bounding box and the ground truth divided by the union of the same. This measure gives a percent track. If the bounding boxes do not intersect then the measure is zero divided by a large number, which is the desired response. If the intersections is one hundred percent but the union is large, a prediction of a bounding box as large as the scene, the measure will be below one hundred percent. With an intersection that is the same as the union a perfect track is observed, giving one hundred percent IOU. Taking the average IOU over a sequence creates a meaningful comparisons even when sequences of the data are different lengths.

2.10 Conclusion

Held noted that some visual attention models [3] [31] proven to be non-competitive with other state-of-the-art trackers. The C-DATM aims to change this outcome. First, through combining the fundamental principles of attention mechanisms with the generic approach of Held's work. Secondly, by adding attention this research adds memory to a model that currently does not have any. Memory allows the network to remember past movements and how well those movements tracked the targets influences future movements. Lastly, attention mechanisms allow for a smaller portion of the scene to be analyzed as compared to Held's work. By analyzing a smaller region of the scene less information is processed during tracking leading to the potential for processing speeds exceeding 100 FPS. Lastly, this research demonstrates the first generic attention-based model. The tracker is trained offline, like [14], and not using a single class of object. The attention mechanism is trained by providing the first bounding box around the object, which is the standard protocol for most popular tracking competitions [22], and predicts bounding boxes for later scenes by comparing the features of the original bounding box to that area of the next frame, while incorporating kinematics.

III. Methodology

There are three research contributions from this study. First, non-recurrent memory is added to a state-of-the-art neural network system to create a more robust system. Secondly, a smaller region of the image is analyzed leading to faster computations than current neural network tracking systems. Lastly, a novel generic attention-based model is created. The contributions are achieved using the methodology described below.

3.1 Design

A single attention-based model is applied to all the experiments done in this study. The model is considered attention-based because only a small portion of the entire scene is processed to track an object. Features are extracted from the region of attention using convolutional neural network layers. These features are processed through five dilated(over time) convolution layers. Lastly, three fully connected layers use regression to create the attention parameters. The modified READ algorithm (equations 3.9 - 3.15) uses the attention parameters to predict the location and size of future bounding boxes. Generality is incorporated into the system because the network is only given the location of the object in the first frame, and future information is derived from that initial location. Using the original features and comparing the next frames features to the original condition, the network predicts the best bounding box for the following frame. Because the process is only based on the initial condition it is general and applicable to any object as long as the object is

identified at the start.

3.2 Attention Mechanisms

In the past [13], a 28 by 28-pixel image had an attention area of 12 by 12. The attention area is only 144 pixels of the 784, processing about 18 percent of the image at a time. The bouncing ball experiment done by RATM[18] reduces the image size of 20 by 20 pixels to a focus area of 5 by 5 pixels. Using only $\sim$6% or 25 of the 400 pixels to process, greatly reduces the amount of information computed by the model. In this work, 64$\times$64 MNIST image-scenes are down-sampled to a 28$\times$28 region. This down-sampled region is $\sim$20% of the original scene, which is more information than processed in past attention-based works. A medium must be found between the amount of data processed and the time needed to compute a solution. Convolutions need large amounts of information to find the accurate correlations filters. The 28$\times$28 attention-region allows the model to fit an entire MNIST digit in the attention-region without any loss of resolution. Having such a small portion of the scene being processed by the network gives the potential for attention models to compute results very quickly.

Attention models are not limited to predicting the location of a fixed-size bounding box. The DRAW[13] algorithm introduced learned parameters to control the size of the bounding box. Using a slight modification from RATM[18] the attention region can be defined as a rectangular area rather than a square area by learning two more parameters than DRAW[13]. This study takes one more step by eliminating the parameters not needed for tracking. In previous works σ and γ values were learned to adjust the level of contrast in the image. For tracking σ is set to e^1, the largest possible value, creating the most contrast in the image making the down-sampled region well-defined. The γ value is pnly used for generative models so it is removed in C-DATM.

The READ operation found in DRAW[13] is expressed through equations 3.1-3.3.

The parameters in equations 3.1 are translated from the network using a fully connected layer W applied to the previous output of the recurrent model h^{dec}. In equation 3.2 the variable A represents the width of the scene, and B is the height. N, in equation 3.3, is the dimension of the Gaussian filters in the grid which equates to the square area size of the down-sampled image. The $\sim$ variables in all equations are the direct results of the neural network. The equations are the translations done to these results to create the attention mechanism of the models.

The parameters learned in DRAW, RATM, and C-DATM to control the bounding box are g_x, g_y, δ_x, and δ_y where g_x and g_y are the coordinates for the middle of the attention area with respect to the edges of the scene, and δ_x and δ_y are the distances between the center of each Gaussian in the grid in the horizontal and vertical directions. The model used in order to calculate these parameters is a combination of the mechanisms used in DRAW[13] (equations 3.1-3.3) and RATM[18] (equations 3.4-3.7). Specifically, DRAW only creates a square region of attention and contains a parameter, γ, that is not needed for tracking (see equation 3.1 in comparison to equations 3.2-3.3). While RATM uses absolute values which are very restrictive and limit the domain of the network's predictions to half that of using a hyperbolic tangent (see equations 3.4-3.7). Using the non-linearity associated with DRAW's READ operation results in accurately defined regions of the image without the harshness of the absolute value. To remove the saturation of the δ values, which can cause the bounding box to become smaller than a pixel, a minimum bounding box size is implemented. The details of which can be found in equations 3.10 and 3.11. Experiments shown in Figure 3.1 show proof of concept for the designed parameters.

$$(\widetilde{g}_x, \widetilde{g}_y, log\ \sigma^2, log\ \widetilde{\delta}, log\ \widetilde{\gamma} = W(h^{dec}) \tag{3.1}$$

$$g_x = \frac{A+1}{2}(\widetilde{g}_x + 1) \qquad g_y = \frac{B+1}{2}(\widetilde{g}_y + 1) \tag{3.2}$$

$$\delta = \frac{max(A,B)-1}{N-1}\widetilde{\delta} \tag{3.3}$$

The READ operation found in RATM[18] is expressed through equations 3.4-3.7. The parameters in equations 3.4 are translated from the network using a fully connected layer W applied to the previous output of the recurrent model h^{dec}. M and N, in equation 3.3, are respectively the horizontal and vertical dimensions of the Gaussian filters in the grid which equates to the rectangular area size of the down-sampled image.

$$(\widetilde{g}_x, \widetilde{g}_y, log\ \widetilde{\delta}_x, log\ \widetilde{\delta}_y, log\ {\sigma_x}^2, log\ {\sigma_y}^2) = W(h^{dec}) \tag{3.4}$$

$$g_x \frac{= (\widetilde{g}_x + 1)}{2} \qquad g_y \frac{= (\widetilde{g}_y + 1)}{2} \tag{3.5}$$

$$\delta_x = \frac{W-1}{M-1}\,|\widetilde{\delta}_x| \qquad \delta_y = \frac{H-1}{N-1}\,|\widetilde{\delta}_y| \tag{3.6}$$

$$\sigma_x = |\widetilde{\sigma}_x| \qquad \sigma_y = |\widetilde{\sigma}_y| \tag{3.7}$$

The READ operation found in C-DATM is expressed through equations 3.8-3.11. The parameters in equations 3.8 are translated from the network using a fully connected layer FC applied to the output of the dilated convolutions DCNN. W and H in equations 3.9, 3.10, and 3.11 are the height and width of the scene, respectively. M and N, in equations 3.10 and 3.11, are respectively the horizontal and vertical dimensions of the Gaussian filters in the grid which equates to the rectangular area size of the down-sampled image. bb_m in equations 3.10 and 3.11 is the minimum bounding box size, explained in detail below.

$$(\widetilde{g}_x, \widetilde{g}_y, \widetilde{\delta}_x, \widetilde{\delta}_y) = FC(DCNN) \tag{3.8}$$

$$g_x = \frac{W+1}{2}(\widetilde{g}_x + 1) \qquad\qquad g_y = \frac{H+1}{2}(\widetilde{g}_y + 1) \tag{3.9}$$

$$\delta_x = \frac{M-1}{bb_m} + \frac{W - bb_m - 1}{M-1}(\frac{\widetilde{\delta}_x + 1}{2}) \tag{3.10}$$

$$\delta_y = \frac{N-1}{bb_m} + \frac{H - bb_m - 1}{N-1}(\frac{\widetilde{\delta}_y + 1}{2}) \tag{3.11}$$

Equation 3.9 is used to calculate the center coordinates for the attention region. The range of $\widetilde{g}$ is restrained using a hyperbolic tangent non-linearity. A hyperbolic tangent is used over a sigmoid function because a sigmoid ranges from 0 to 1 while a hyperbolic tangent ranges from -1 to 1. The larger possible range of the hyperbolic tangent benefit the network when training. A hyperbolic tangent forces $\widetilde{g}$ to be between -1 and 1. If $\widetilde{g}$ were to become -1 then the value of g would 0, the minimum bounding box value center coordinate.

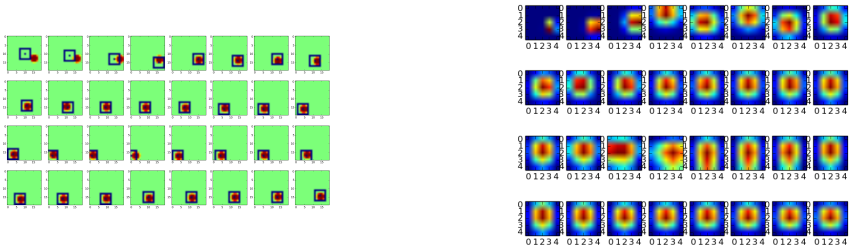

Figure 3.1: The frames of experiments done to replicate findings in RATM[18] for bouncing ball tracking. The left image is is the tracking done over the 32 frames in the sequence. The right image is the down-sampled image generated by the Read operation.

Opposite, if $\widetilde{g}$ becomes 1 then g is the scene length or height. Using a hyperbolic tangent $\widetilde{\delta}$ is in the range of -1 to 1. If the value from the fully connected layer is -1, then δ is evenly split over bb_m to cause the bounding box to be bb_m. A value of 1 causes the bounding box to cover the whole scene.

The parameters presented in equations 3.8 - 3.15 were used to replicate a toy experiment originally presented in RATM[18]. The experiment involved tracking a single bouncing ball through a scene. Figure 3.1 shows the sequence of images used in the experiments. The attention tracker successfully tracks the ball using the parameters shown in Equations 3.8-3.15 instead of the parameters used in RATM [18] or DRAW [13]. This result provides a proof-of-concept for the experiments presented in Chapter 4.

$$\mu^i{}_x = g_x + (i - M/2 - .5)\delta_x \tag{3.12}$$

$$\mu^j{}_y = g_y + (j - N/2 - .5)\delta_y \tag{3.13}$$

The READ operation that creates the attention area is found in DRAW[13]. Equations

3.12 and 3.13 are the calculation for the mean locations of filter rows located at pixels i, j. The parameters μ^i and μ^j are lists of coordinates representing the center locations of the filters in the scene used to produce the down-sampled image. Equations 3.14 and 3.15 calculate the vertical and horizontal filter banks where C_x and C_y are normalization factors so that $\sum_a F_x[i,a] = 1$ and $\sum_b F_y[j,b] = 1$. These filter banks are matrix multiplied with each other and then with the scene to create the down-sampled attention region.

$$F_x[i,a] = \frac{1}{C_x} e^{-\frac{a-{\mu^i}_x{}^2}{2\sigma^2}} \tag{3.14}$$

$$F_y[j,b] = \frac{1}{C_y} e^{-\frac{b-{\mu^j}_y{}^2}{2\sigma^2}} \tag{3.15}$$

The attention model used to generate the results shown in Figure 3.1 follows the structure of the model used in RATM[18]. The entire 20×20 pixel scene is down-sampled to 5×5 using the initial parameters in equation 3.8 of zero. This sets the initial attention area as most of the scene. As the scene changes, frame-by-frame, the previous attention area is processed through the dilated model to generate new parameters for equation 3.8. The new parameters are then used to create the updated attention region for the current frame and the down-sampled portion to be used in calculating the next frame. The model learns using the root mean squared error of the attention region compared to the true object locations from the previous frame only. The model uses previous scene parameters to successfully predict the bounding boxes of the bouncing ball. Using this method, only a few frames are needed to accurately localize the target. Zero values are necessary because the network does not start with any information from the scene so it must classify and localize instead of just keeping a target track.

3.3 Research Questions and Hypotheses

This study will answer three major questions:

1. By adding memory as found in attention-based models and reformatting to an attention problem, can a more robust model be created? Can we use dilated convolutions as a format of memory? Removing the need for recurrent systems. These questions will be answered by comparing the performance of our model to the past models, and through data-manipulation experiments. The memory layer will generate kernel, or filters, that will be visualized to understand our methods further.

2. Can the use of an attention-based model lead to acceptable performance speeds (At least 30 frames per second)? Speed will be judged based on frames per second(FPS) during training and testing as others [14] have.

3. Can an attention-based model be adjusted so that given the first frame's bounding box the model can generically track salient objects in a scene? Attention-based models are currently class based and do not transfer to new classes. MNIST contains ten different classes, and using a single model for testing the entire dataset fundamentally means the model is not class based.

Question one deals with adding memory into the system. As seen in Chung's work[7], having memory units such as the LSTM or GRU added to the model over a standard hyperbolic tangent lead to a better performing model. Building on that, dilated convolutions have performance increases from standard RNN memory units on sequences[2]. Current fast performing models in the neural network community [14] do not use memory in decision making, there is only a kinematic relationship learned though fully connected layers by comparing two different scenes. Instead of comparing two different regions of interest together, a single network will use memory of previous regions to learn the kinematic relationship and generate bounding boxes for the current scene.

Memory units both have the ability to regulate the amount previous decisions have on future ones, this work keeps 8 previous frames in memory and uses different levels of dilations to explore different memory length subsets. Having a regulated memory will allow the network to become more robust. Memory creates accuracy and robustness by allowing previous information to percolate through all decisions. For instance, if an object becomes occluded in a certain frame the previous information gathered through past frames will allow the network to accurately judge the partially visible object and make an educated conjecture on the location of the rest of the hidden object, creating an accurate bounding box through the occlusion. If the occlusion lasts longer than the memory length the model may struggle more with the occlusion. When the memory does not contain the original non-occluded target the network may have to keep a larger bounding box until the occlusion halts. Memory will allow the network to track targets in the difficult scenes found in the moving MNIST dataset and will allow the translation of attention models onto challenging problems.

Question two pertains to the speed of the network. Attention-based models only process a single small portion of the scene at a time, the attention region. By processing small portions of the image the network propagates quickly. By comparison, convolutional networks must interpret the entire scene and as problems are becoming more challenging those networks are continuing to grow in size making them very slow. Well performing convolutional networks[33] [8] [22] used for tracking do not even process at a frame per second. Held[14] takes some features of attention-based mechanics by only processing a portion of the image at a time, and through that achieves 100 frames per second on VOT 2014. This model will have similar, or faster speeds. Having dilated convolutions allows for larger receptive fields without more free parameters. This form of memory will lend itself well to the speed goal. The memory will also help to prevent the need for a larger comparison region. The constant smaller region of interest in this model gives the ability for quick computational speeds, which is important in the development of non-research

based systems. This model will have the ability to compute a challenging dataset at much faster computational speeds than current state-of-the-art neural network tracking systems.

Current attention-based models are class based, and question three will address this. Each model for the tracking applications must be trained to track a specific target. Class-based is necessary when multiple objects are in the scene for traditional attention models. This gives the unique ability to localize a target in a scene and then track it. However, the goal is to create a generic attention-based network. Popular tracking datasets[22] give the object's initial location at the start of the sequence. This information removes the need of the unique capability of attention-based models to localize. If the model principles are slightly modified attention can use the starting location of the object to its advantage. With the starting bounding box, the network can understand the features of the object, stored as the condition, for the particular scene without ever have been privy to that information before. This allows the network to then learn the relationship between features from frame to frame rather than the features of a distinct object class. This will give the novel ability to track any class of object.

3.4 Data Generation

Previous works [13][18] used recurrent neural network models to learn the parameters in the equations 3.8 - 3.15. To extend this work to attention-based models, this study presents experiments applied to moving targets taken from the Moving MNIST data set.

Moving MNIST is a data set generated from MNIST digits. These digits travel frame-to-frame over a sequence of frames. A frame starts as a 64×64 black space. A random digit from the MNIST set is chosen and placed randomly in the first frame and the digit is saved as the desired object to be tracked. The digit is given a random direction of travel and all digits move with the same velocity and reflect direction when the edge of the frame is reached. The location of the 28×28 image is saved as the object truth for each frame.

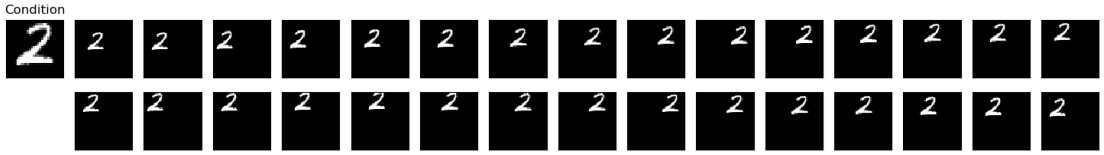

Figure 3.2: A moving MNIST sequence of 30 frames containing one-digit. The first row first frame is the object that is desired to be tracked. The next 15 frames in the first row are the first 15 frames in the sequence. The second row contains the last 15 frames of the sequence. The 2 digit travels frame-to-frame at a set velocity and direction, reflecting the direction when an edge is met to simulate bouncing off a wall.

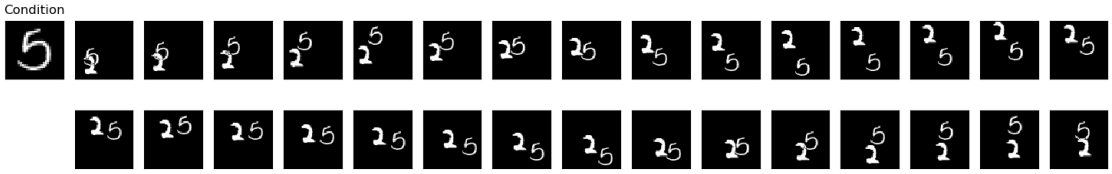

Figure 3.3: A moving MNIST sequence of 30 frames containing two-digit.The first row first frame is the object that is desired to be tracked. The next 15 frames in the first row are the first 15 frames in the sequence. The second row contains the last 15 frames of the sequence. The 2 and 5 digits travel frame-to-frame at a set velocity and separate direction, reflecting the direction when an edge is met to simulate bouncing off a wall. When digits overlapping results an element-wise addition of the two digits, normalized between zero and one.

An example of a single-digit traveling frame-to-frame is found in Figure 3.2.

To experiment on performance when occlusion is introduced the study uses a 2-digit set. Extending the parameters of the single-digit set, the two digits cannot be the exact same, but they can be from the same MNIST class. The digits occlude each other rather than colliding, and both digits have different trajectories. The occlusion found in the experiments is an element-wise addition of the two digits, normalized between zero and one. Digits to not leave the frame, or get covered completely by background clutter. An example of 2-digit sets are shown in Figure 3.3.

All networks are trained using 10,000 sequences with 30 frames each. All experiments are conducted using 1,000 sequences with 30 frames.

3.5 C-DATM

The input of the model is a down-sampled attention region of the current frame created from the object location in the previous frame. From the attention region C-DATM predicts a new bounding box with equations 3.9 - 3.15 using the activations of the final neural network layer. The output of C-DATM is the predicted object location for the current frame. This means the model is not learning to predict where the object is located based on the locations from previous frames; the network is judging the new position based on the visual information at past object locations.

The initial position of the attention region is the location of the object in the first frame. To initialize a location with a recurrent model an inverse matrix calculation must be done. The inverse calculation is only applicable to a square matrix, forcing the last hidden layer of the recurrent model to match the size of the output parameters. In C-DATM the square matrix would be 4x4 causing too small of a hidden layer to accurately learn the needed trends. Switching to a non-recurrent form of memory C-DATM is successfully initialized at the location of the object in the first frame.

Figure 3.4 illustrates the methodology used C-DATM to create attention-based tracking. The given location of the object in frame 1 is used to predict the location of the object in the second frame. In Figure 3.4, the black box represents the location of the object in the previous frame. That location is applied to the second frame where the object has moved slightly, this causes the object to no longer be centered in the attention-region. To track the object the information in the black box is processed by C-DATM. First, convolutional neural network layers will process the down-sampled region to extract features. Those features are fed through dilated convolutions that via a fully connected layer will produce the attention parameters (see equation 3.8). Using these attention parameters C-DATM uses the READ operation defined in equations 3.9 - 3.15. READ generates the down-sampled attention region (the red box in Figure 3.4) for the current frame.

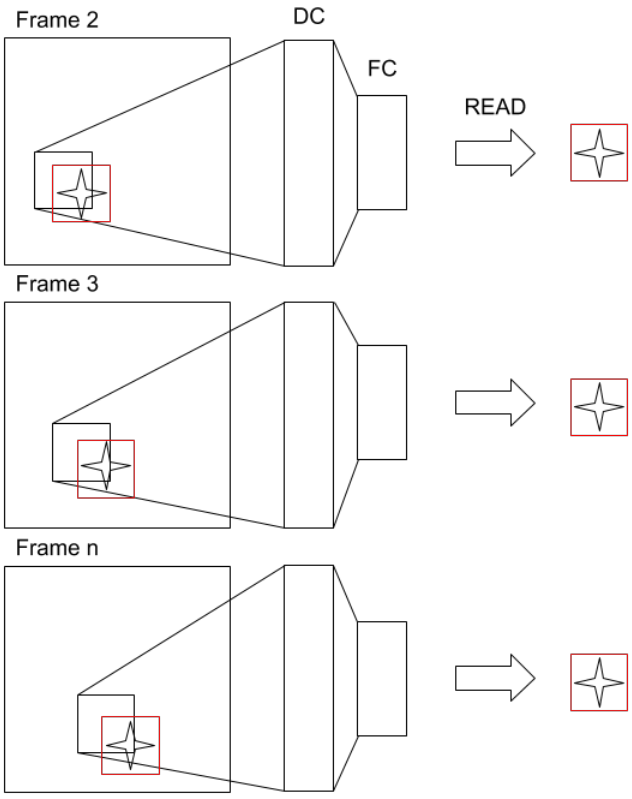

Figure 3.4: The overall model that illustrates how attention methods translate to tracking. The previous frame's object location is used on the current frame as input to the dilated convolutions(DC). Those activations are regressed through fully connected layers(FC). The 4 outputs of the fully connected layers are given to the READ operation to translate them to attention mechanisms. The attention mechanisms are described in equations 3.9 - 3.15

In short, the previous frames object location is applied to the current frame and the attention mechanism predicts the current object location. C-DATM predicts the new object location over all frames with only the condition, or desired object features, to compare to creating a tracking model. C-DATM learns to compare features over a set number of frames resulting in learning the kinematics of the objects movements.

C-DATM follows current community standards for attention-based mechanisms[18][13]. C-DATM uses less information by having a smaller region of comparison, learns kinematic information, and uses the condition object to track generally.

3.5.1 Network Layers

C-DATM is composed of three main layer types. The first type is the convolutional neural network layer. Convolutional layers learn a set of correlations filters. AlexNet[23] trained a networking containing convolutional layers on the Imagenet[9] dataset and improved significantly on the previous state-of-the-art. Imagenet is a dataset comprised of millions of images, the features learned from this set have been transferred, via transfer learning, to other research including Held's GOTURN[14]. Convolutional features are suggested to describe the important qualities of objects given enough training data and time.

The features generated by the convolution layers are fed to dilated convolution layers. If the past sequence frames are stacked across time then dilated convolutions can be used as memory units. If the sequence length is 30 frames and the model has 8 frames of memory the stacked features are visualized in Figure 3.7. Dilated convolutions are the same as regular convolutions except for the sparsity in application of the kernels. A dilation factor of one creates a standard convolution layer because the stride of the sparsity is the dilation factor minus one(creating a sparsity of zero). Increasing the dilation factor increases the sparsity of the kernel when applied. If the kernel is spread over a larger area, more sparse, then the affected area of the kernel, or receptive field, grows. Dilated convolutions increase receptive field sizes without increasing the number of free parameters(size of the kernels),

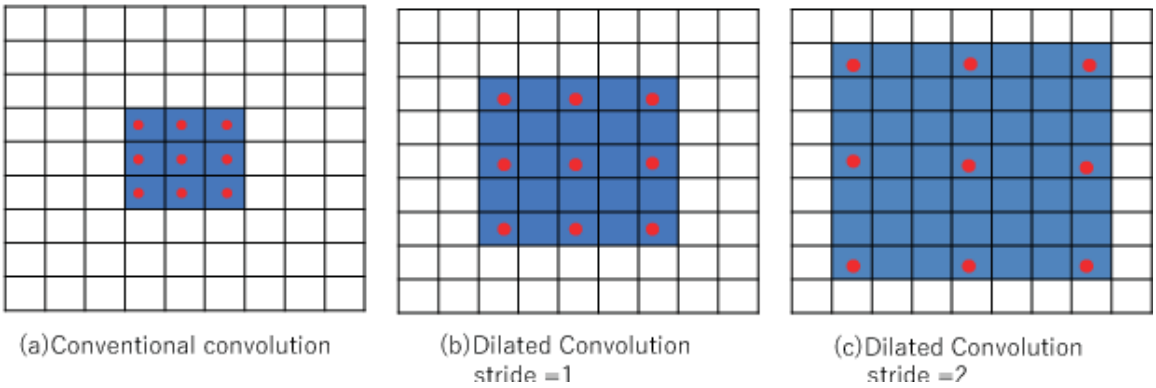

Figure 3.5: The increase in dilation factor causes the kernel to have a larger receptive field (blue area). This does not increase the free parameters(red dots) of the convolutions. (a) is a convention convolution with stride 0 or dilation factor 1. (b) is a dilated convolution with stride 1 or dilation factor 2. (c) is another dilated covolution with stride 2 or dilation factor 3. This visual representation is found in Yashmashita *et al.* [45].

as seen in Figure 3.5.

By having the last n number of frames stacked horizontally and feeding it to multiple layers of dilated convolutions a sparse representation of the past can be made and applied to the next predictions. Figure 3.6 illustrates a representation where n is eight and there are three dilation layers. The first has a dilation factor of 1, a normal convolution. The second layer has a dilation factor of 2, skipping a neuron. The last layer has a dilation factor of 4, skipping 3 neurons. Each of these dilated convolution layers have the same number of free parameters and learn correlation filters to represent moving targets over different time steps.

C-DATM ends with three fully connected layers of neurons. Fully connected layers perform a linear translation to map the information from the dilated convolutions to the READ operation found in equation 3.8. This allows the activations found through the dilated convolutions to generate the object location via the down-sampled attention-region generated by the attention mechanisms. Dropout[40] is used in literature to help generalize and reduce over-fitting. This model is not trained to a level of over-fitting and generalizes

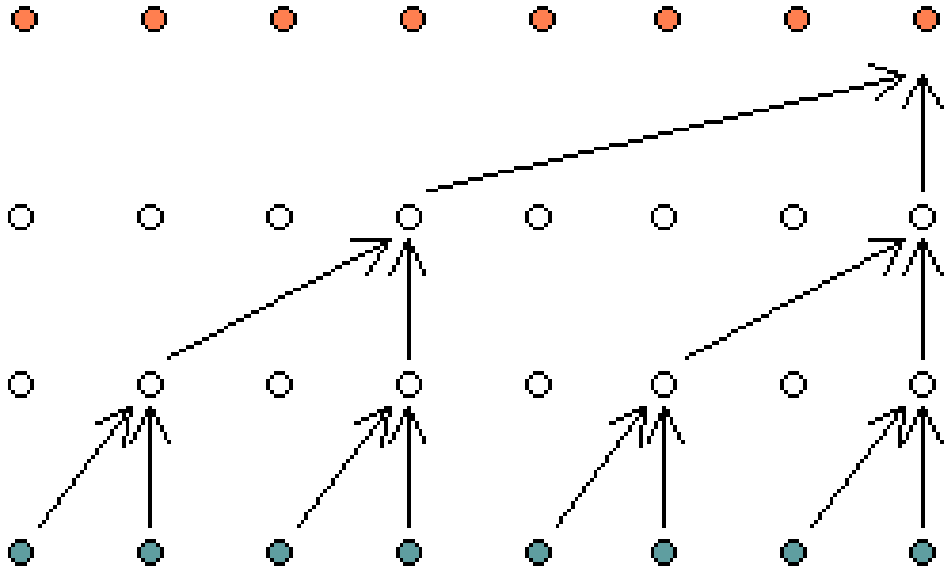

Figure 3.6: Each new prediction (orange) is a result of multiple layers of dilation(white) on the memory(blue). The first dilated layer from memory has a dilation factor of 1, the second has a dilation factor of 2, and the last have a dilation factor of 4.

well without dropout, so dropout is not used.

3.5.2 Non-Linearities

Rectified linear units (ReLU) are used in order to efficiently reach the global minimum of the loss function in the neural network[32]. Kahou[18] suggested their use of absolute values is to combat the harshness of the zeroing of negative values in an ReLU. Without the absolute values the effects of training with an ReLU can lead to vanishing or exploding gradients in this model. Also, literature [17] suggests batch normalization improves performance of dilated convolutions. To gain desired normalization and combat the harsh ReLUs without absolute values, C-DATM use scaled exponential linear units (SeLU)[21]. Instead of zeroing negative values they are scaled exponentially. The piece-wise activation function is in Figure 3.8. A visual representation of the activation function is in Figure 3.9.

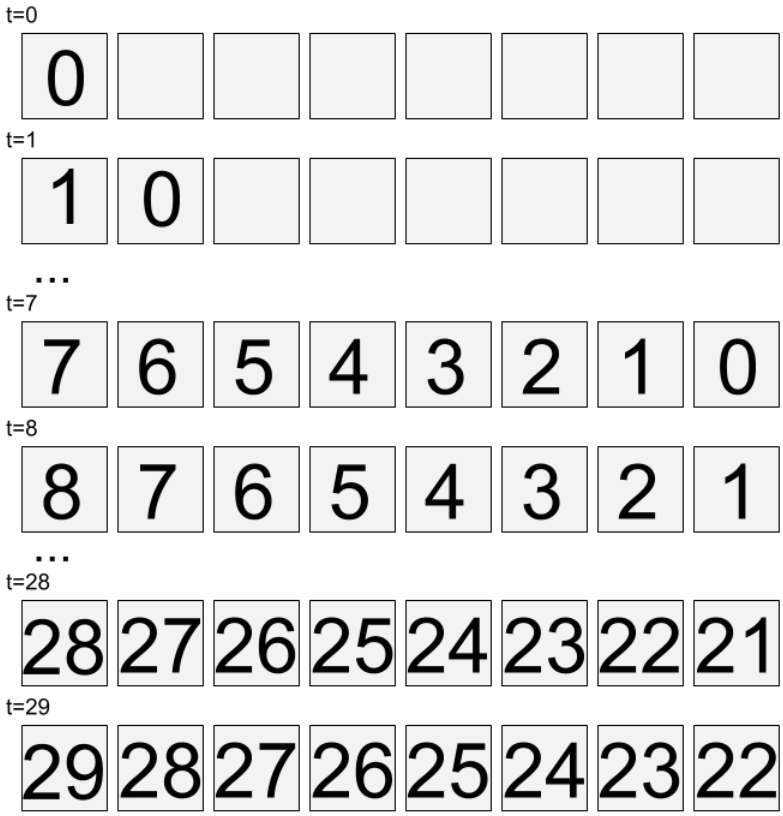

Figure 3.7: At time(t) of zero the model only holds the first frame in memory. As t reaches 8 the features are stacked. Once the time past is larger than the length of the memory old features are dropped and new are added.

$$\text{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leqslant 0 \end{cases}.$$

Figure 3.8: The piece-wise function of the scaled exponential linear unit[21]. The negative values are scaled exponentially in terms of α, and the positive values are by a factor of λ.

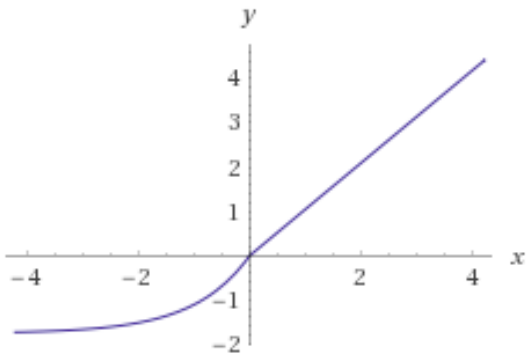

Figure 3.9: A visual representation of the piece-wise function of the scaled exponential linear unit[21]. In this visualization α and λ are one.

3.5.3 Layer Parameters

The convolution layers are designed to extract object features from the training set. The first convolution layer has 6 kernels, each being 5×5 in dimension. The activation from those correlation filters are fed to a 2×2 pooling layer down-sample the activations. The pooling outputs are fed to another convolutional layer that has 16 kernel, each kernels is 3×3. The visualization of the learned kernels is observed and discussed in Chapter 5. These layers are observed in Figure 3.10's first row.

Eight results of the feature extracting first two layers are stacked horizontally over time. If the sequence frame is less than t_7 the right empty memory slots are set to zero, as seen in Figure 3.7. This creates a 16×80×10 layer of memory. This is visualized in Figure 3.10's second row.

From the memory five dilated layers are used. Each layer, besides the last, has residual connections. Residual connections add the input to the output so that the sparsity of the dilated convolutions does not cause loss of visual information. Residual connections also allow for the dilated convolutions to apply to different time periods. In Figure 3.10 the dilated convolutions have a p value. That p value is for padding and it is needed in order to keep the outputs the same size as the inputs so residual connections can be used.

The first dilated layer has 24 kernels that are 5×5 with a dilation factor of 1. The

resulting matrix is 24×80×10. The second has 32 kernels, size 5×5 with a dilation factor of 2, resulting in a matrix of 32×80×10. The only dimension that grows applies to the number of kernels used in each layer. The third dilated layer has 40 kernels, size 5×5 with a dilation factor of 4. The fourth layer has 48 kernels with size 5×5 and a dilation factor of 8. Lastly, the network uses 56 kernels with dimension 5×5 with a dilation factor of 16. The last layer does not have any padding so the resulting matrix size is reduced to 56×16×6.

The input to the network and the condition are put through these layers simultaneously. The results are combined and put through three fully connected layers. The first reduces the siamese networks from 2×56×16×6 to 2056. The second layer keeps the size of 2056, to regress. The last layers format the output to the attention parameters, from 2056 to 4, to satisfy equation 3.8.

3.5.4 Training Methods

C-DATM is trained offline using dictated sequences and images defined as Moving MNIST. The lack of on-line training greatly speeds up the model during testing. Convolutional Neural Networks are slow to train, by freezing the weights during testing only forward-propagations are needed which less computationally expensive. C-DATM learns using the mean-squared error of each frames object location compared to the truth object. GOTURN[14] uses a L1 Norm with success, and mean-squared error allows the network to learn a single stable solution.

3.5.5 Pre-training/Transfer Learning

Previous models[14] use transfer learning though pre-trained neural network layers to extract features from the data before being passed through the model. This study uses MNIST with GOTURN, and pre-trained layers are created from LeNet5 [26] trained to classify MNIST images to digit classes. When C-DATM attempts to use pre-trained layers the

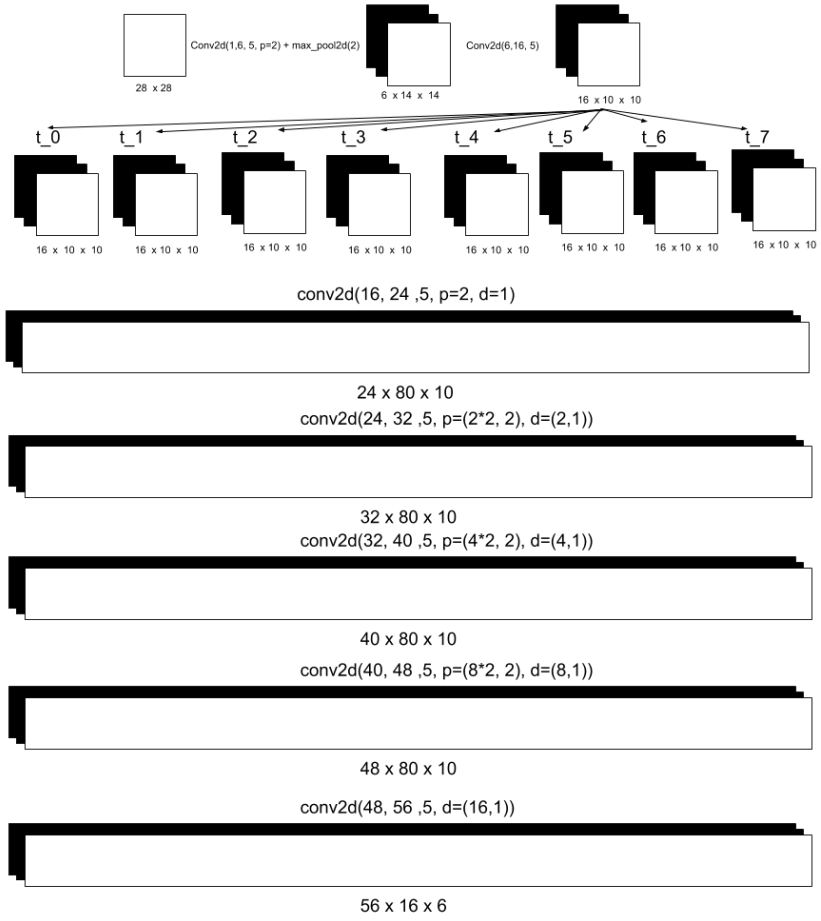

Figure 3.10: The first tow sections of the model used in this work. The first convolution layer has 6 kernels, each being 5x5 in dimension. The activation from those correlation filters are fed to a 2×2 pooling layer to effect a larger area of the data. That is then fed to another convolutional layer that has 16 kernels. Each of those kernels is 3×3. Eight results of the feature extracting first two layers are stacked horizontally over time. If the sequence frame is less than t_7 the right empty memory slots are set to zero. This creates a $16\times80\times10$ layer of memory. From the memory five dilated layers are used. Each layer, besides the last, has residual connection. The first dilated layer has 24 kernels that are 5x5 with a dilation factor of 1. The resulting matrix is $24\times80\times10$. The second has 32 kernels, size 5×5 with a dilation factor of 2, resulting in a matrix of $32\times80\times10$. The only dimension that grows applies to the number of kernels used in each layer. The third dilated layer has 40 kernels, size 5×5 with a dilation factor of 4. The fourth layer has 48 kernels with size 5×5 and a dilation factor of 8. Lastly, the network uses 56 kernels with dimension 5×5 with a dilation factor of 16. The last layer does not have any padding so the resulting matrix size is reduced to $56\times16\times6$.

model cannot produce successful tracks. Therefore, the experiments use a model trained end-to-end, and an explanation for limitations is discussed in Chapters 4 and 5.

3.6 Data Collection and Analysis

The results of all experiments on Moving MNIST are done on a single computer with a Nvidia Titan X to assure that all runs accurately measure the execution time (wall-clock time) using Python library functions.

The accuracy of the model is tested using a measure of "intersection over union". The calculations assume that both the truth object and the predicted object are represented by a rectangular shape. Equations 3.16 and 3.17 take two 4 coordinate systems representing rectangles, A and B, and a coordinate axis z. Equation 3.16 finds the minimum coordinate after locating the maximum in a direction, x or y, and calculating the largest x-coordinate of both object locations then taking the smallest of those. Equation 3.17 performs similar except the minimum is found first, then the maximum. This finds the smallest x-coordinate of both object locations, then the maximum of those. Equation 3.18 refers to a clamp function used to limit the minimum possible value to be 0. This ensures the calculation is correct even when the object locations do not intersect. Equation 3.20 explains the calculation of the union of object location boxes A and B, the area of A plus the area of B minus the intersection of A and B. Finally, equation 3.21 is the calculation of intersection over union (IOU) of two coordinate systems A and B.

$$minMax_z(A, B) = min(max(A_z, B_z)) \tag{3.16}$$

$$maxMin_z(A, B) = max(min(A_z, B_z)) \tag{3.17}$$

$$clamp(A) = max(A, 0)) \tag{3.18}$$

$$\begin{aligned} intersection(A, B) \ = \ & clamp(minMax_x(A, B) - maxMin_x(A, B)) \\ * \quad & clamp(minMax_y(A, B) - maxMin_y(A, B)) \end{aligned} \tag{3.19}$$

$$union(A, B) \ = \ area(A) \ + \ area(B) \ - \ intersection(A, B) \tag{3.20}$$

$$IOU(A, B) \ = \frac{intersection(A, B)}{union(A, B)} \tag{3.21}$$

The learned parameters of the network can be translated to object location coordinates. Equation 3.22 calculates the minimum location coordinate in a direction, usually x or y, represented by z from the attention parameters discussed in equations 3.9 - 3.11. Subsequently, equation 3.23 calculates the maximum coordinate in the z direction. Equation 3.24 represents a 4 coordinate object location used in this system.

$$min_z = g_z - \frac{(\delta_z * (D_z - 1))}{2} \tag{3.22}$$

$$max_z = g_z + \frac{(\delta_z * (D_z - 1))}{2} \tag{3.23}$$

$$[min_x, min_y, max_x, max_y] \tag{3.24}$$

3.7 Fashion MNIST

To further test the overall ability of C-DATM to track different types of objects, an extra dataset is used. With the importance of creating a model that has the ability to generalize over multiple sets and objects, extra experiments on different datasets are important to empirically show the potential of C-DATM. A moving fashion MNIST[43] set was created.

3.8 Experiment Definitions

A series of different experiments were conducted using C-DATM to understand what information the model uses to track objects. The first experiment focuses on kinematics. By creating random directional changes in the test set the reliance on kinematic information can be measured. For instance given a sequence of frames, F_1, F_2, ..., $F_{n/2}$, ..., F_2 , F_1 where the objects abruptly reverse direction at frame $F_{\frac{n}{2}}$, if the network continues to perform well when the object changes direction, kinematic features of the objects are not the

only features needed to track an object. We test the dilated convolution memory by creating erratic movement in the data. We also experiment with the memory by moving the objects faster than expected by skipping frames. C-DATM's operating condition is tested to determine the reliance C-DATM has on the condition to track objects. The condition experiments denote how well the network generalizes. By blurring the image that is the operating condition we can empirically understand how much focus the network has on specific features in the model. Visualization of the components of the networks help us understand how C-DATM works.

3.9 Development Setting

The use of PyTorch allows for rapide prototype of the system. This project started using Keras [6], however Keras is designed to use a graph to build the system. The whole network must be designed and compiled as one unit. This proved problematic because it limited our ability to visualize the behaviors of internal components of the network. PyTorch allows each part of the system to be displayed at anytime. PyTorch is built around four major cornerstones modularity, minimalism, easy extendabil- ity, and the ability to work with Python. Modularity is the key to developing many models, as the pieces can easily be interchanged, and this helps with the development of new models targeted for specific research tasks. Minimalism is key to the dissemination of research, the code is de- veloped to be short and simple so that others can use and improve on the solution. Extendability means that new modules are easy to add and the primary reason we selected PyTorch.

3.10 Summary

This research focuses on answering three questions about speed, memory, and generality of attention models. Moving MNIST is used to demonstrate the advantages of C-DATM.

Attention-based mechanisms have not been widely used in object tracking and this research extends our knowledge of attention-based tracking models. By creating a general, attention-based model that can process image-sequence at speeds greater than 100 frames per second, C-DATM represents an advancement in the state-of-the-art tracking systems. In the next section, the results of experiments are presented to explore and explain the contributions of C-DATM.

IV. Results

4.1 Baseline

All baseline results for this work are derived using a single conditional dilated convolution attention model, C-DATM, described in detail in Chapter 3, is trained using the single-digit Moving MNIST set. Visualizations of the training data cis shown in Figures 4.1-4.5. The condition, or an image of the truth object, is the desired tracking target for C-DATM. C-DATM is trained for 25 epochs. An epoch is defined as the use of every sequence in the training set once. The sequences are stochastically ordered during training for each epoch. The learning rate used is 1e-4 for the first 15 epochs, the last 10 use 1e-5. The Adam optimizer[20] is used with amsgrad[36], beta values of .9 and .999, an ϵ of 1e-3, and a weight decay of 5e-4.

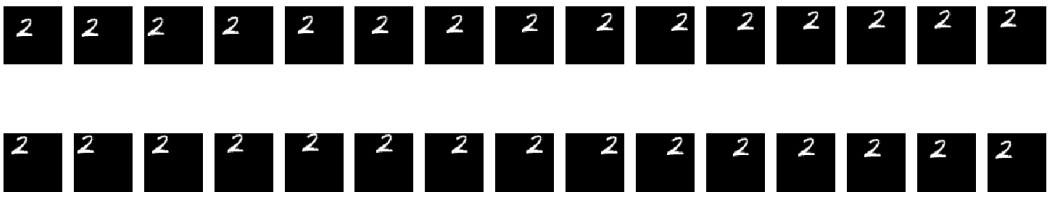

Figure 4.1: A visual representation of a 30 frame training sequence with a handwritten 2 digit. The first row is the first 15 images in the sequence, and the second row is the last 15.

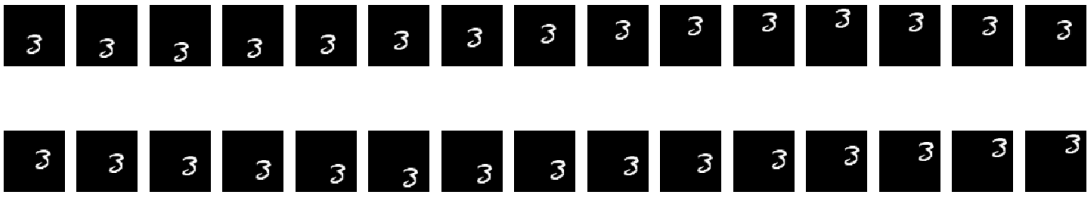

Figure 4.2: A visual representation of a 30 frame training sequence with a handwritten 3 digit. The first row is the first 15 images in the sequence, and the second row is the last 15.

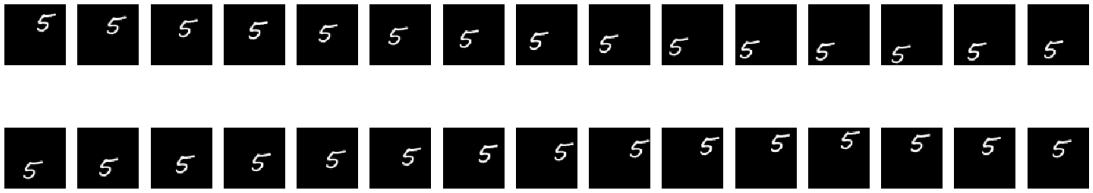

Figure 4.3: A visual representation of a 30 frame training sequence with a handwritten 5 digit. The first row is the first 15 images in the sequence, and the second row is the last 15.

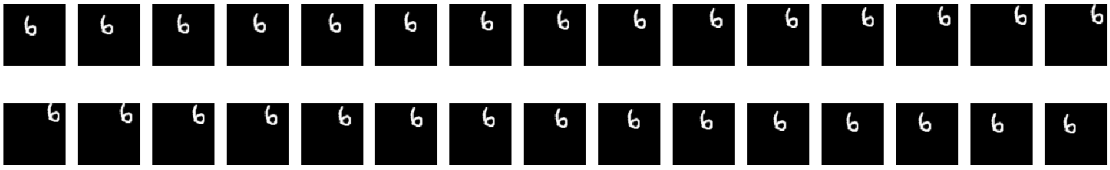

Figure 4.4: A visual representation of a 30 frame training sequence with a handwritten 6 digit. The first row is the first 15 images in the sequence, and the second row is the last 15.

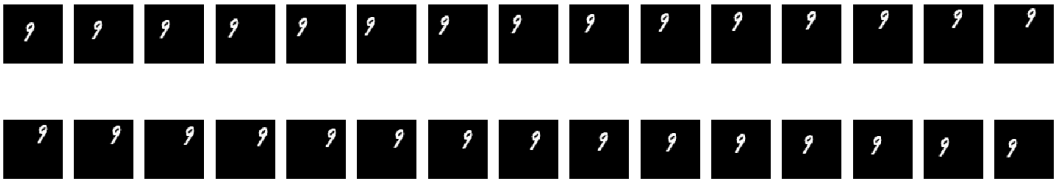

Figure 4.5: A visual representation of a 30 frame training sequence with a handwritten 9 digit. The first row is the first 15 images in the sequence, and the second row is the last 15.

After 25 epochs the total mean squared error of the truth object location compared to the predicated is 68,241. The average time per epoch is 212.0 seconds. Using 10,000

sequences with 30 frames each the network trains at 1,415 frames per second. Figure 4.6 is a visual representation of the training process of the network.

To compare C-DATM to GOTURN [14] we train a GOTURN model with pre-trained LeNet5 [26] feature extraction layers, much like AlexNet being used in GOTURN's VOT 2014 experiments. The pre-trained LeNet5 feature extraction is fine-tuned on the moving MNIST data set by retraining the regression layers.

GOTURN defines a bounding box initially located at the center of the image. The bounding box is a fixed size of 2×28 pixels. GOTURN then takes a sample from the previous frame in the sequence that is twice the size of the original bounding box. The larger bounding box from the previous frame is applied to the current frame after the object has moved slightly. That image and a down-sampled image of the correct object location are fed simultaneously to 3 layers of regression that predict the correct center position of the object location in the current frame.

The first layers of GOTURN are two convolutional layers. The input to GOTURN is a down-sampled version of the scene at the initial object location. The first convolution layer has 6 kernels, each being 5×5 in dimension. The activation from these correlation filters are fed to a 2×2 pooling layer. The pooling layer is fed to another convolutional layer that has 16 kernels, each is 3×3. The features produced from the second convolution layer are fed into 3 fully connected layers that are 2048 in size. A final layer translates the last 2048 fully connected layer to 2 outputs. Those outputs are scaled with the image size and are the center coordinates for the predicted object location. The size of the bounding box is 28×28 to follow the design choices of GOTURN. The best performance for GOTURN is to used pre-trained LeNet5 features.

GOTURN is trained on single-digit Moving MNIST data consisting of 10,000 sequences. GOTURN is trained for 275 epochs resulting in an ending L2 loss of 123,486. The first 25 epochs have a learning rate of 1e-5 and the last 250 have a learning rate of 1e-6.

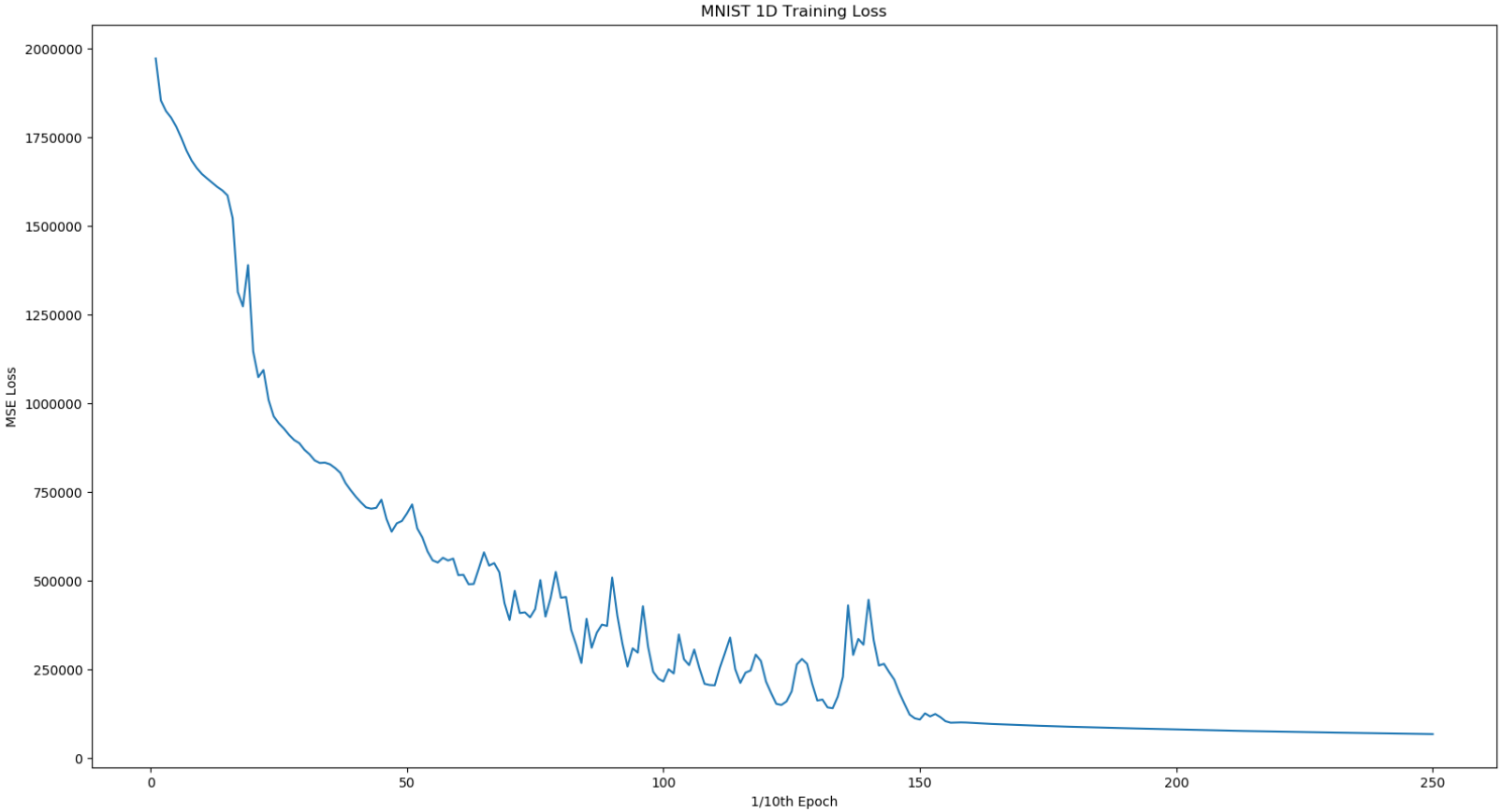

Figure 4.6: The training loss of the C-DATM baseline model.

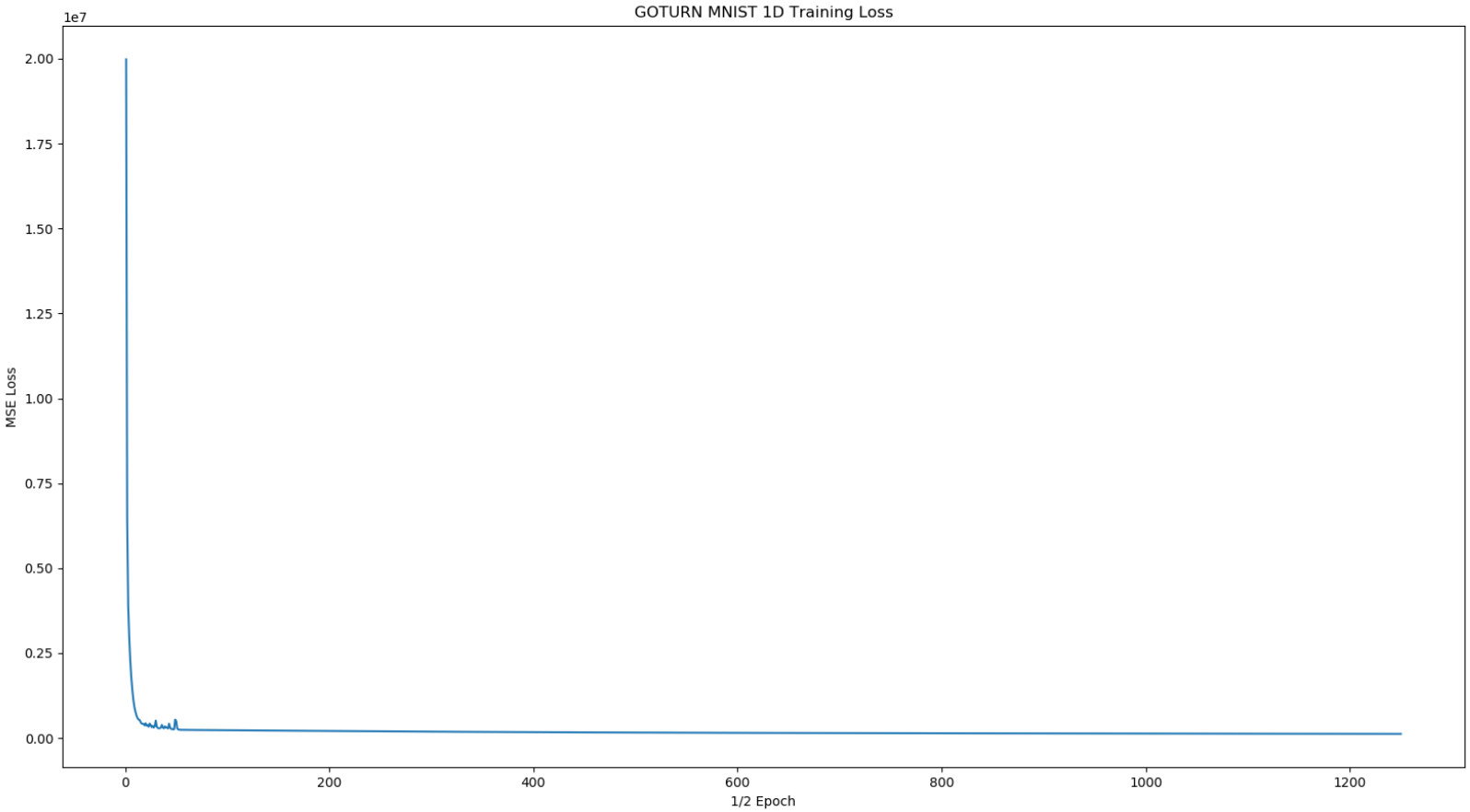

Figure 4.7: The training loss of the GOTURN model.

The training of this model is visualized in Figure 4.7.

C-DATM produces a tracking accuracy of 91.30% IOU for single-digits and 62.23% IOU for 2-digits. GOTURN's accuracy is 62.71% IOU for single-digits and 47.70% IOU for 2-digit data. These tracking statistics are used as the baseline for comparing the performances during the remaining experiments.

4.1.1 Single Digit MNIST

C-DATM produces a tracking accuracy of 91.30% IOU for single-digits. Figures 4.8-4.13 represent C-DATM tracking a single-digit. GOTURN's accuracy is 62.71% IOU when tracking a single-digit. Figures 4.14-4.19 are tracking a single-digit using GOTURN.

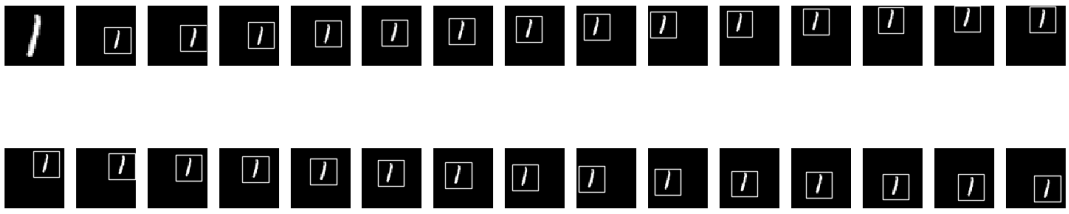

Figure 4.8: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten 1 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

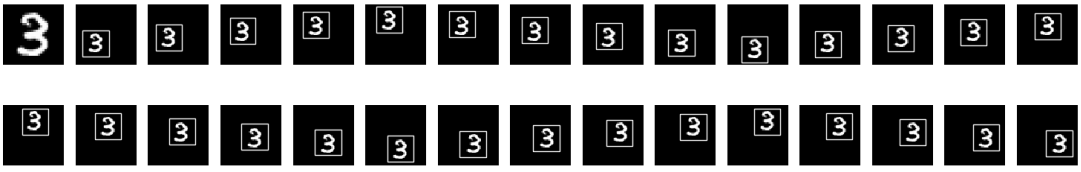

Figure 4.9: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten 3 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

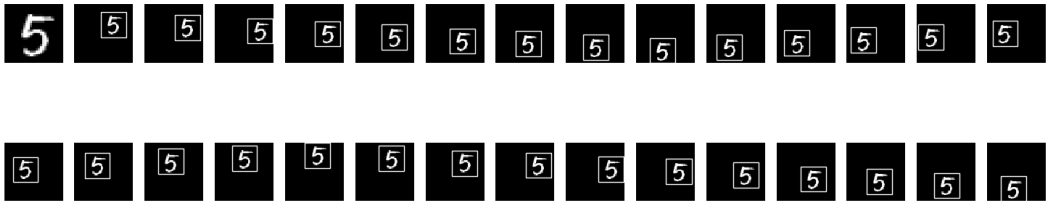

Figure 4.10: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten 5 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

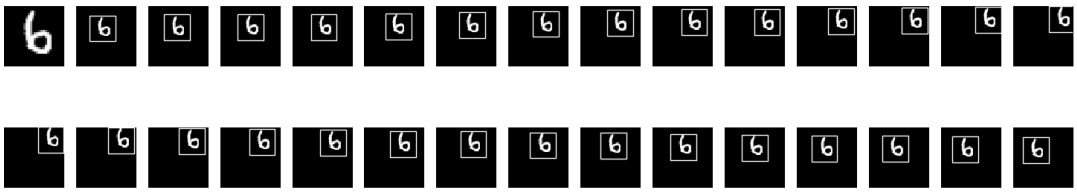

Figure 4.11: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten 6 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

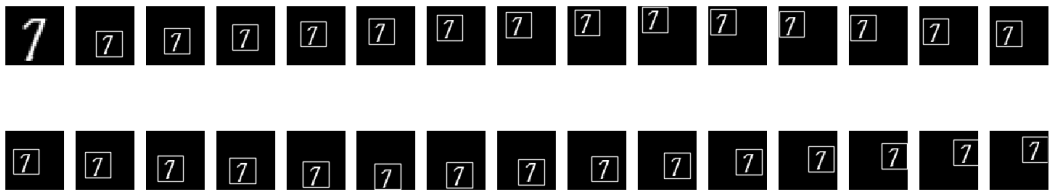

Figure 4.12: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten 7 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

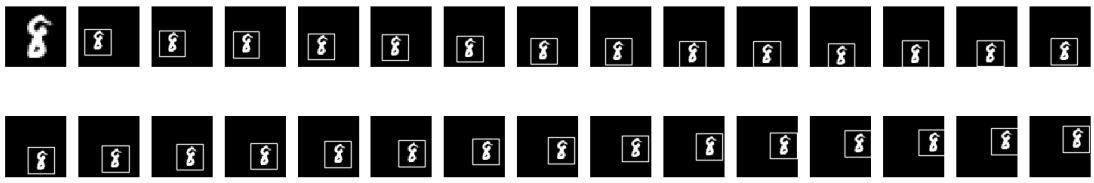

Figure 4.13: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten 8 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

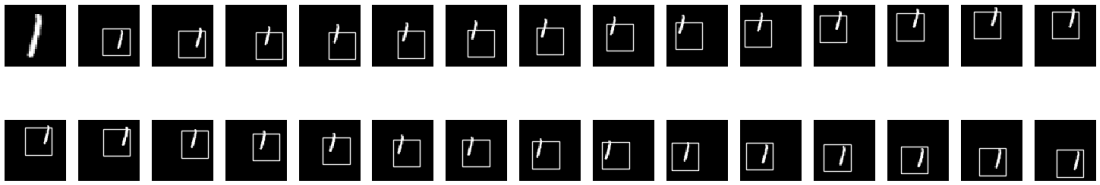

Figure 4.14: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten 1 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

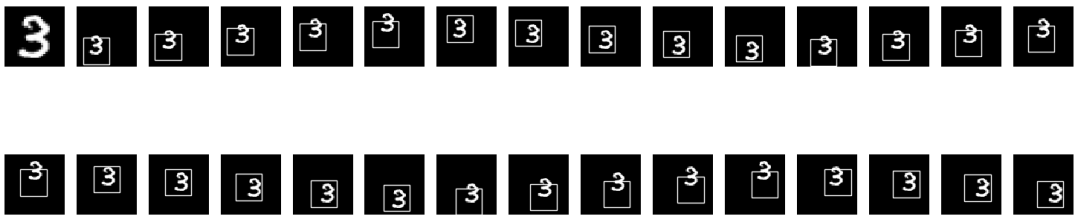

Figure 4.15: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten 3 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

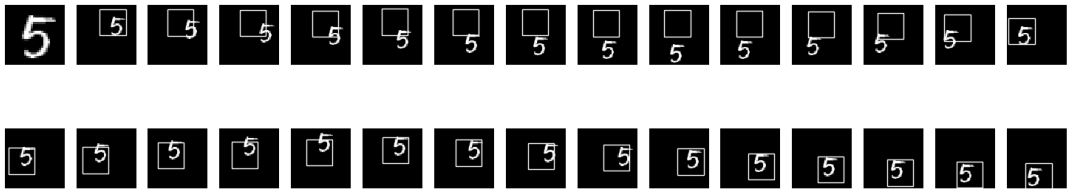

Figure 4.16: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten 5 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

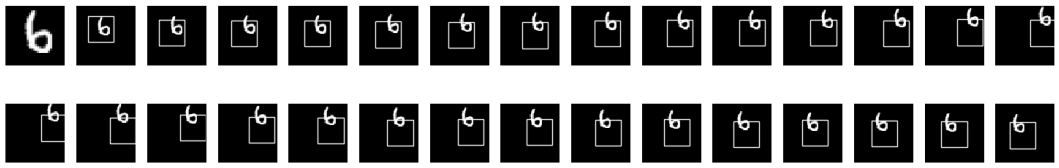

Figure 4.17: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten 6 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

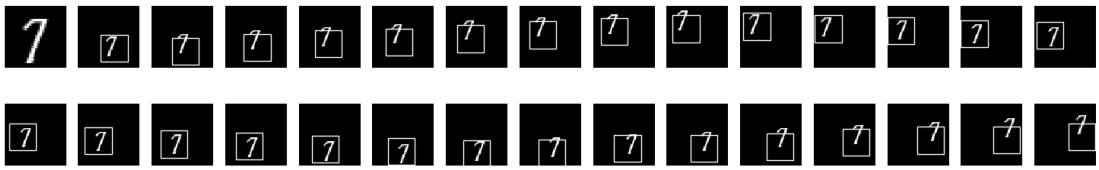

Figure 4.18: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten 7 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

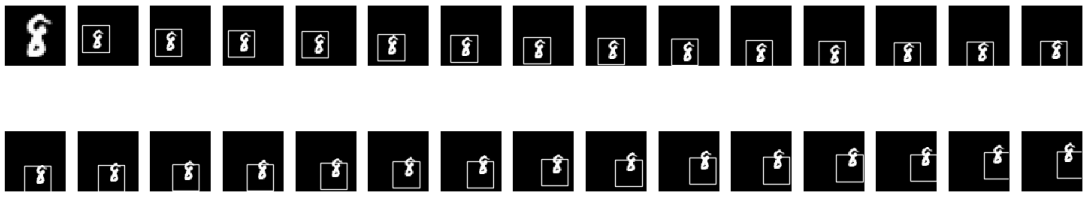

Figure 4.19: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten 8 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

4.1.2 Two Digit MNIST

C-DATM produces a tracking accuracy of 62.23% IOU for 2-digits. Figures 4.20-4.25 visualize tracking a digit when two digits are present. GOTURN's accuracy is 47.70% IOU for 2-digit data. Figures 4.26-4.31 represent tracking of a single-digit when two digits are in the scene.

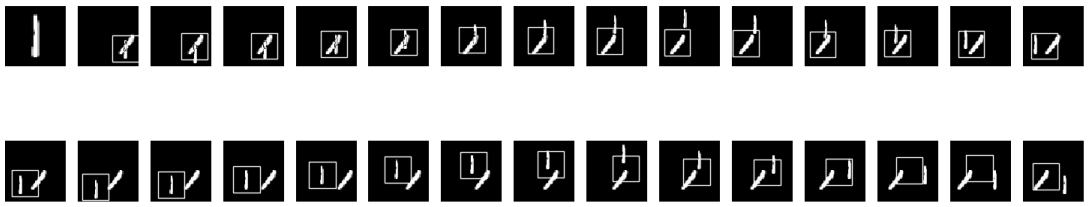

Figure 4.20: A visual representation of C-DATM tracking a 30 frame sequence with two handwritten digits 1 and 1. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

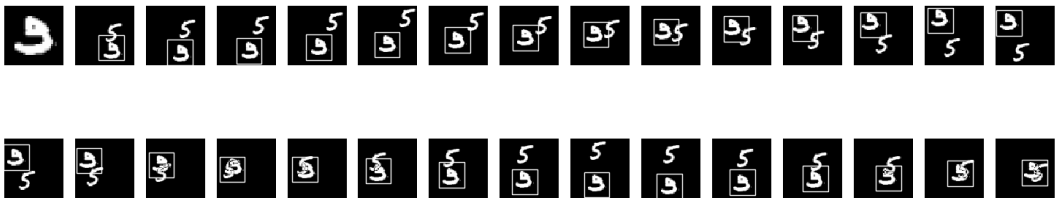

Figure 4.21: A visual representation of C-DATM tracking a 30 frame sequence with two handwritten digits 3 and 5. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

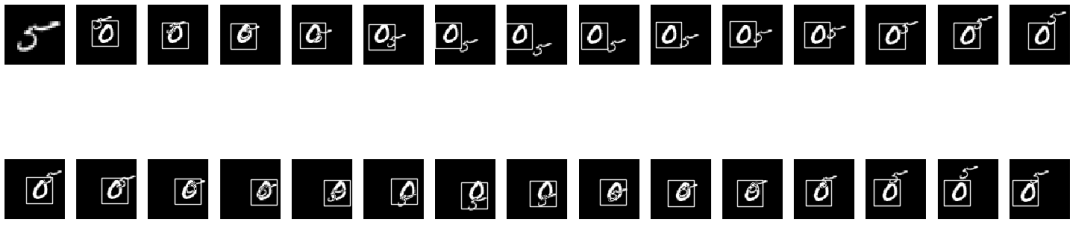

Figure 4.22: A visual representation of C-DATM tracking a 30 frame sequence with two handwritten digits 5 and 0. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

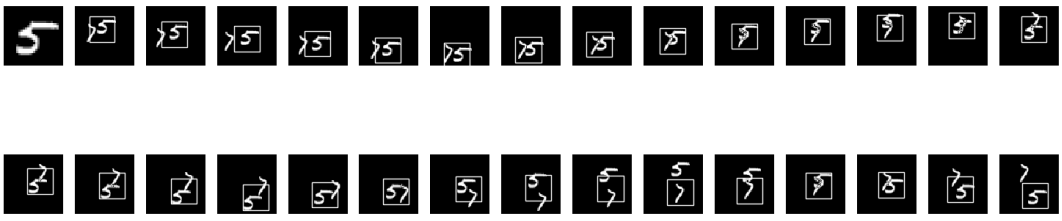

Figure 4.23: A visual representation of C-DATM tracking a 30 frame sequence with two handwritten digits 5 and 7. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

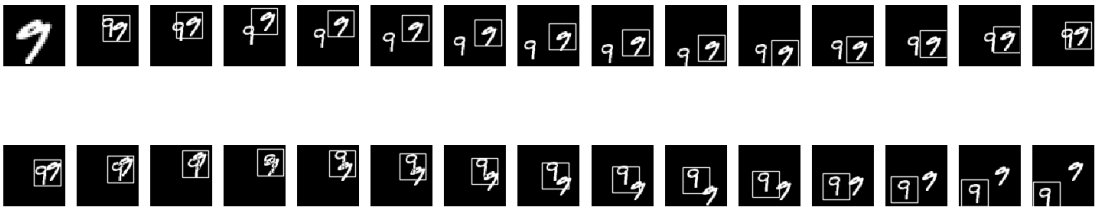

Figure 4.24: A visual representation of C-DATM tracking a 30 frame sequence with two handwritten digits 9 and 9. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

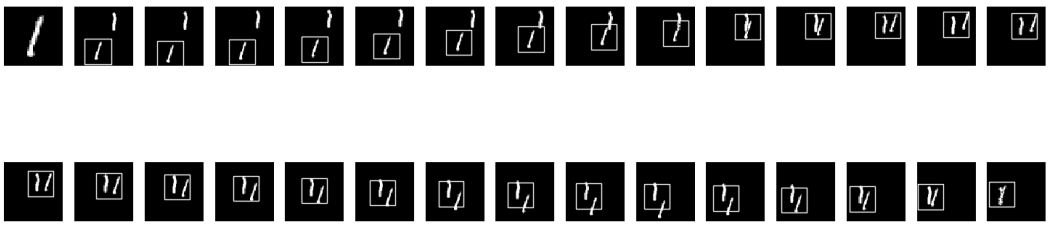

Figure 4.25: A visual representation of C-DATM tracking a 30 frame sequence with two handwritten digits 1 and 1. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

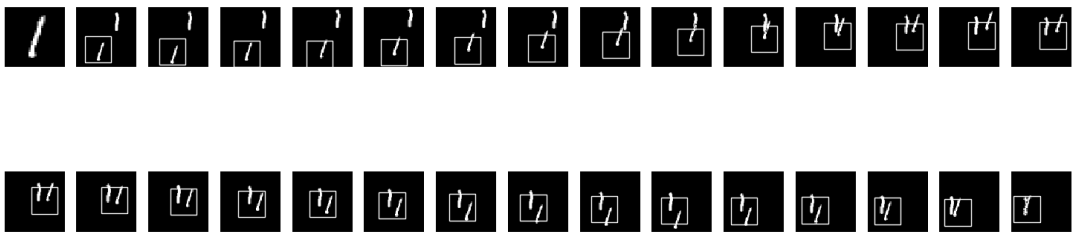

Figure 4.26: A visual representation of the GOTURN model tracking a 30 frame sequence with two handwritten digits 1 and 1. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

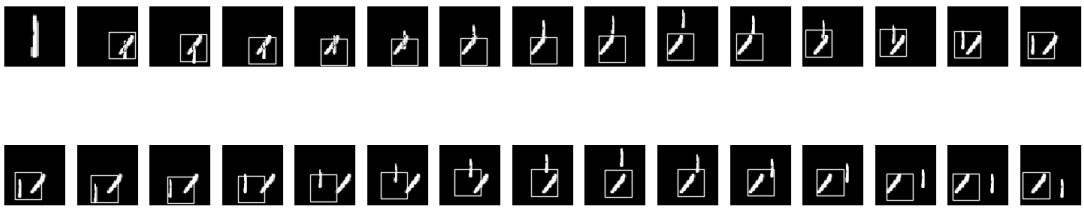

Figure 4.27: A visual representation of the GOTURN model tracking a 30 frame sequence with two handwritten digits 1 and 1. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

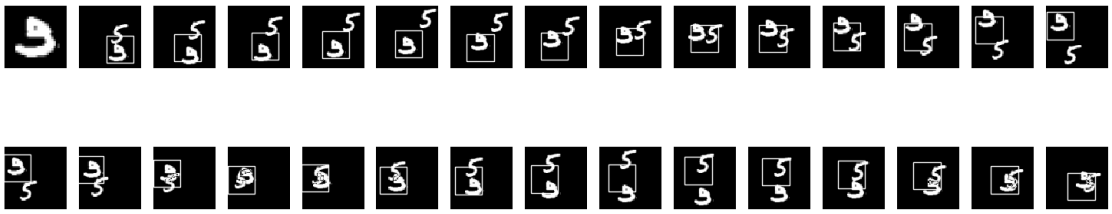

Figure 4.28: A visual representation of the GOTURN model tracking a 30 frame sequence with two handwritten digits 3 and 5. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

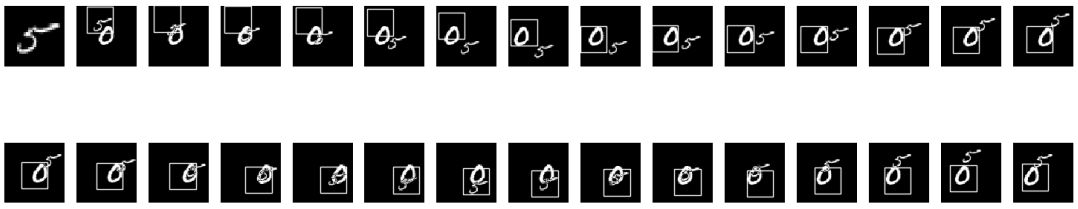

Figure 4.29: A visual representation of the GOTURN model tracking a 30 frame sequence with two handwritten digits 5 and 0. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

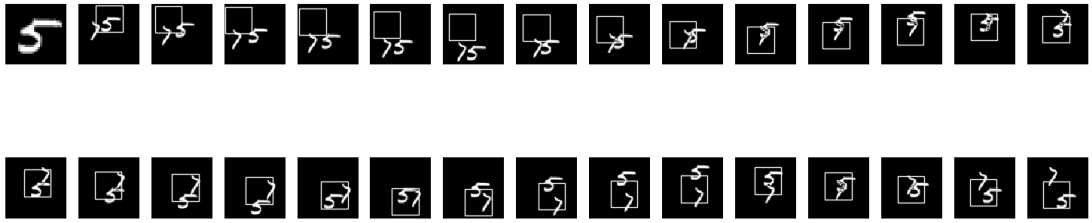

Figure 4.30: A visual representation of the GOTURN model tracking a 30 frame sequence with two handwritten digits 5 and 7. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

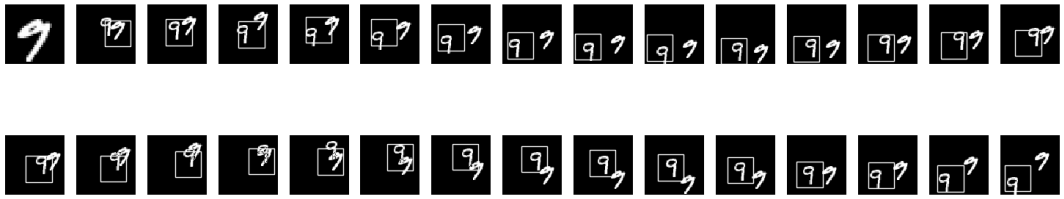

Figure 4.31: A visual representation of the GOTURN model tracking a 30 frame sequence with two handwritten digits 9 and 9. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

4.2 Kinematic Test

The kinematic experiment tests the network's reliance on the smooth kinematics of the objects. To perform this test, we took our 30 image long sequence and split it into two sub-sequences. The first half of the sequence is not changed from the original sequence and the images in the second half of the sequence mirrors the first half. This effectively created a sequence of image where the object abruptly reverses direction. We also create a sequence where the object reverses direction twice by splitting the sequence into thirds and mirror the middle third of the sequence.

All results from the kinematics experiment can be found in Table 4.1 and Table 4.2. Bar graph representations of the kinematics experiment is found in Figures 4.56 and 4.57.

Results of a single reverse tracking sequence are visualized in Figures 4.32-4.43. Results of a double reverse are in Figures 4.44-4.55.

C-DATM performs well when processing a sequence of images containing an object that turns once. For the single turn test we achieve 90.98% IOU which is only a minor decrease of 0.35% relative to the baseline performance. GOTURN also does well during the single turn test with an IOU of 64.02%, an increase of 2.09% relative to the baseline. Turning twice yields similar results, C-DATM achieves 90.80% IOU, which is a decrease of 0.55%, relative to the baseline. GOTURN reaches 64.42% IOU, a positive increase of 2.73% relative to the baseline.

For two-digit sequences and a single turn, C-DATM reaches 62.94% IOU, an increase of 1.14% compared to the baseline performance. GOTURN reaches 49.32%, an increase of 3.40% relative to the baseline. Turning twice follows the same increasing pattern. C-DATM achieves 64.10% IOU, an increase of 3.00%. GOTURN hits 51.02%, an increase of 6.96%.

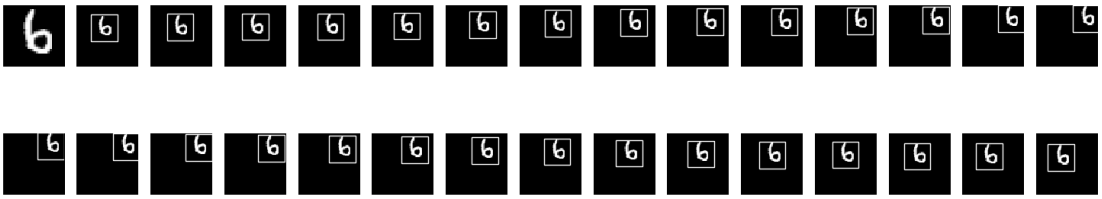

Figure 4.32: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 6. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second row is a mirror of the first.

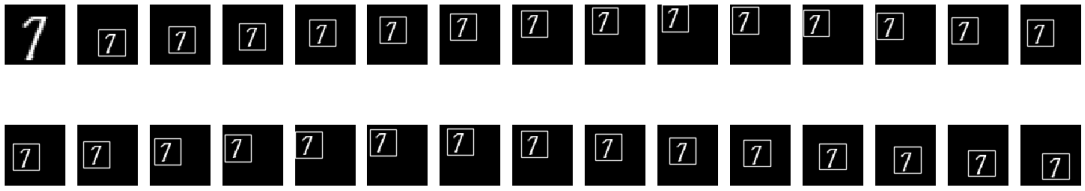

Figure 4.33: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 7. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second row is a mirror of the first.

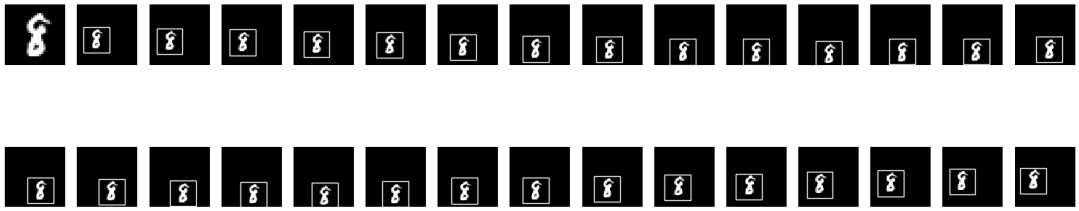

Figure 4.34: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 8. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second row is a mirror of the first.

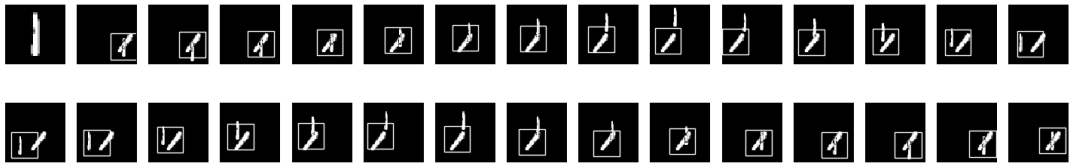

Figure 4.35: A visual representation of C-DATM tracking a 30 frame sequence with two handwritten digits 1 and 1. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second row is a mirror of the first.

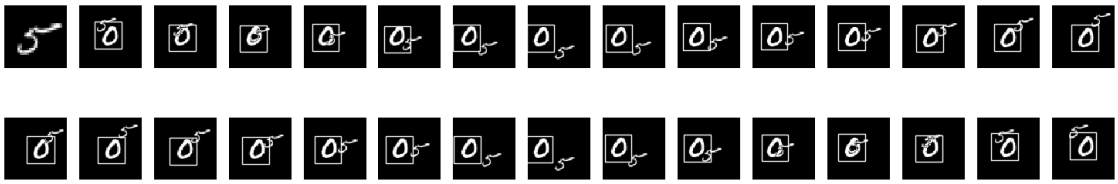

Figure 4.36: A visual representation of C-DATM tracking a 30 frame sequence with two handwritten digits 5 and 0. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second row is a mirror of the first.

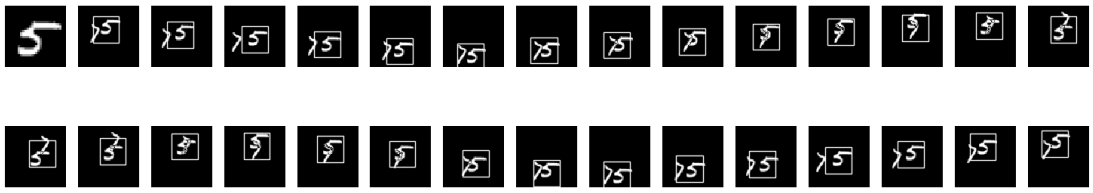

Figure 4.37: A visual representation of C-DATM tracking a 30 frame sequence with two handwritten digits 5 and 7. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second row is a mirror of the first.

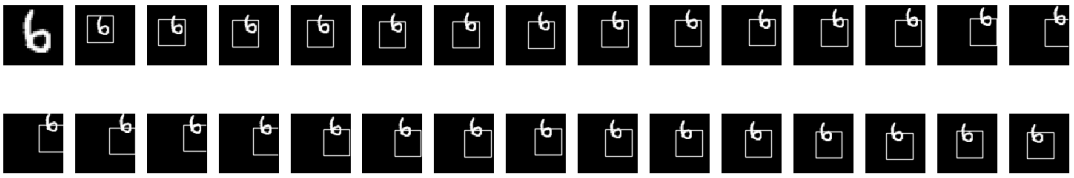

Figure 4.38: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 6. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second row is a mirror of the first.

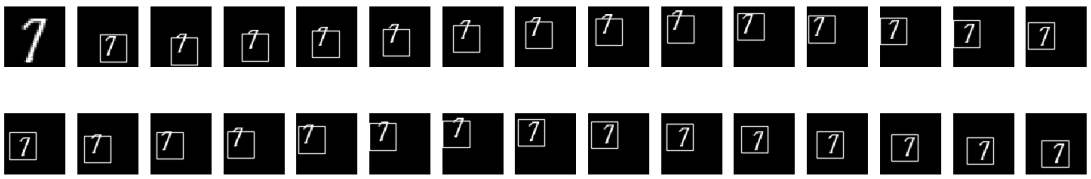

Figure 4.39: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 7. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second row is a mirror of the first.

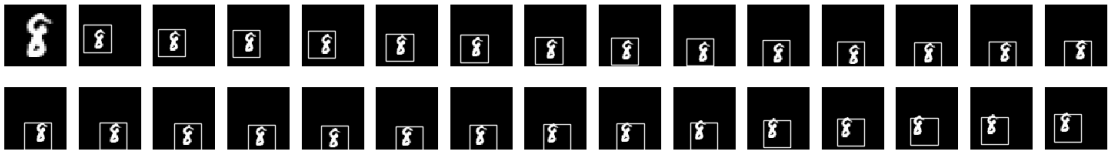

Figure 4.40: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 8. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second row is a mirror of the first.

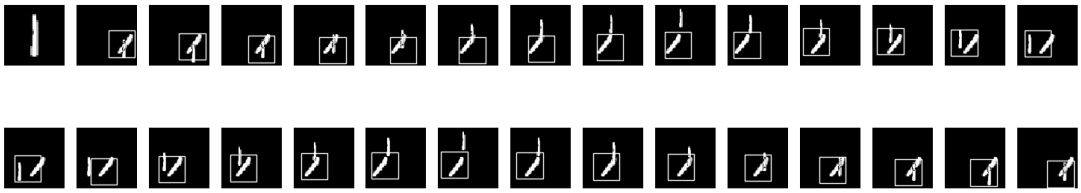

Figure 4.41: A visual representation of the GOTURN model tracking a 30 frame sequence with two handwritten digits 1 and 1. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second row is a mirror of the first.

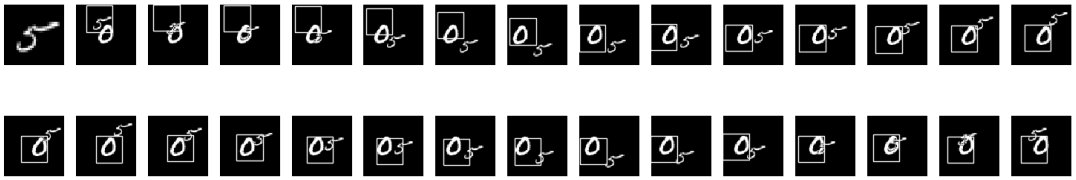

Figure 4.42: A visual representation of the GOTURN model tracking a 30 frame sequence with two handwritten digits 5 and 0. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second row is a mirror of the first.

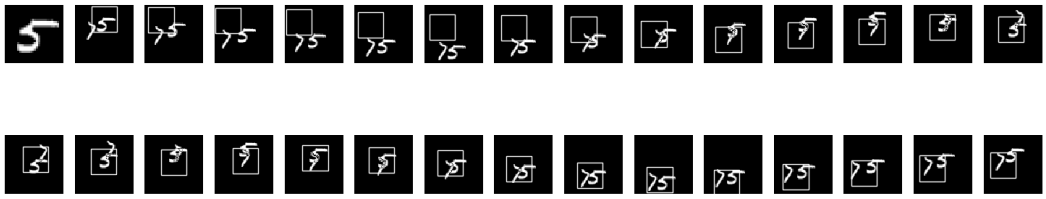

Figure 4.43: A visual representation of the GOTURN model tracking a 30 frame sequence with two handwritten digits 5 and 7. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second row is a mirror of the first.

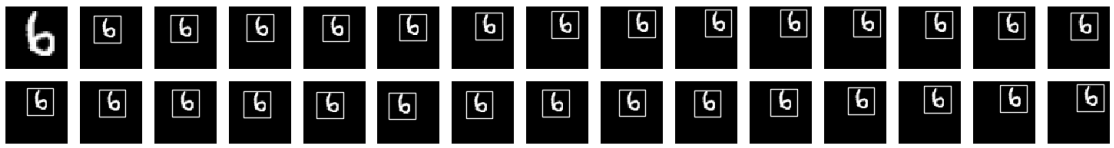

Figure 4.44: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 6. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second set of 10 frames is a mirror of the first and the last set of 10 frames is a copy of the first.

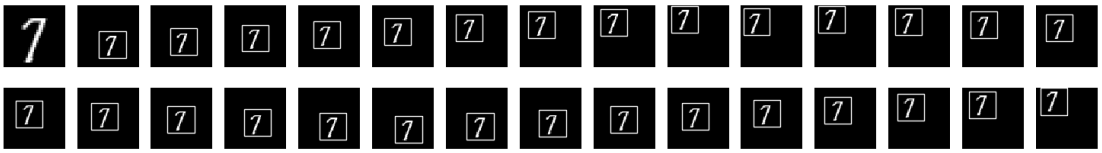

Figure 4.45: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 7. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second set of 10 frames is a mirror of the first and the last set of 10 frames is a copy of the first.

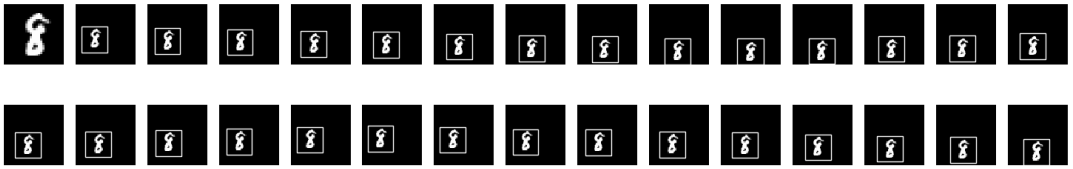

Figure 4.46: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 8. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second set of 10 frames is a mirror of the first and the last set of 10 frames is a copy of the first.

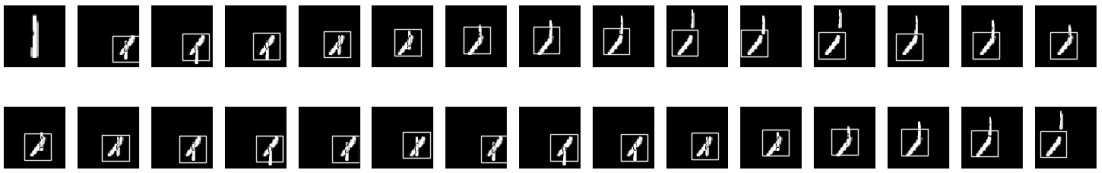

Figure 4.47: A visual representation of C-DATM tracking a 30 frame sequence with two handwritten digits 1 and 1. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second set of 10 frames is a mirror of the first and the last set of 10 frames is a copy of the first.

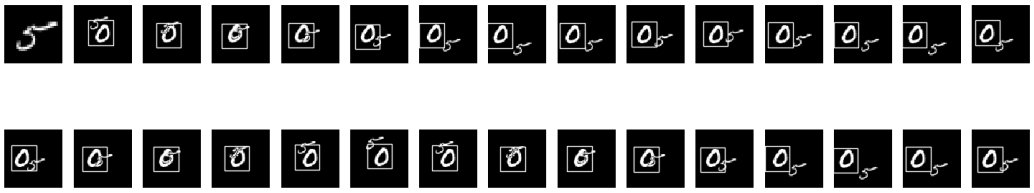

Figure 4.48: A visual representation of C-DATM tracking a 30 frame sequence with two handwritten digits 5 and 0. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second set of 10 frames is a mirror of the first and the last set of 10 frames is a copy of the first.

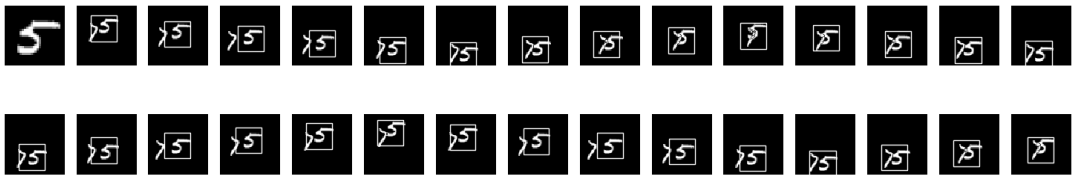

Figure 4.49: A visual representation of C-DATM tracking a 30 frame sequence with two handwritten digits 5 and 7. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second set of 10 frames is a mirror of the first and the last set of 10 frames is a copy of the first.

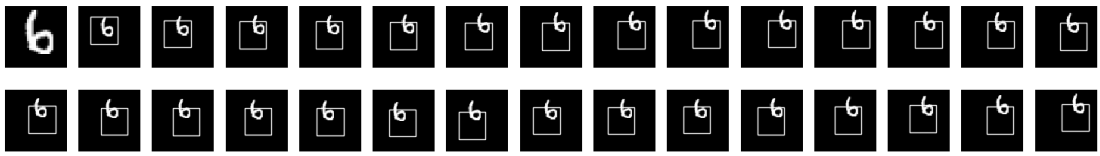

Figure 4.50: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 6. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second set of 10 frames is a mirror of the first and the last set of 10 frames is a copy of the first.

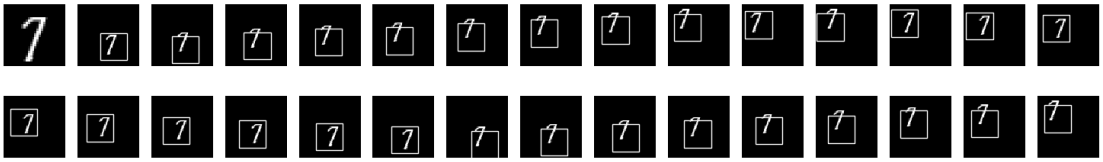

Figure 4.51: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 7. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second set of 10 frames is a mirror of the first and the last set of 10 frames is a copy of the first.

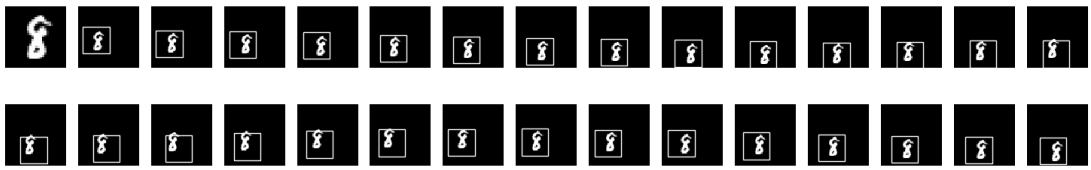

Figure 4.52: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 8. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second set of 10 frames is a mirror of the first and the last set of 10 frames is a copy of the first.

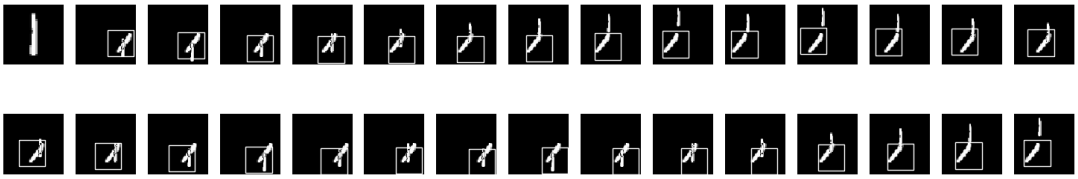

Figure 4.53: A visual representation of the GOTURN model tracking a 30 frame sequence with two handwritten digits 1 and 1. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second set of 10 frames is a mirror of the first and the last set of 10 frames is a copy of the first.

Reversals	C-DATM 1D	C-DATM 1D Δ	GOTURN 1D	GOTURN 1D Δ
1	90.98%	-0.35%	64.02%	+2.09%
2	90.80%	-0.55%	64.42%	+2.73%

Table 4.1: The Kinematics test results using the one-digit MNIST tracking set.

Reversals	C-DATM 2D	C-DATM 2D Δ	GOTURN 2D	GOTURN 2D Δ
1	62.94%	+1.14%	49.32%	+3.40%
2	64.10%	+3.00%	51.02%	+6.96%

Table 4.2: The Kinematics test results using the two-digit MNIST tracking set.

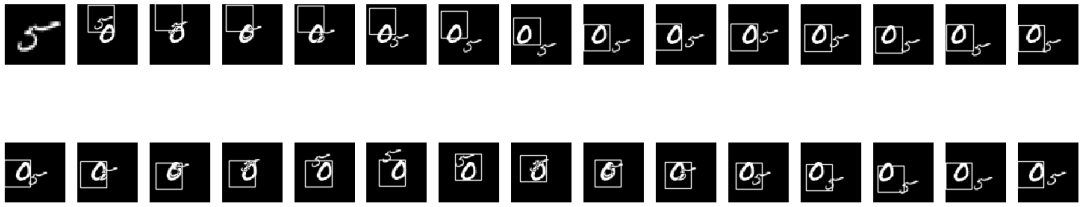

Figure 4.54: A visual representation of the GOTURN model tracking a 30 frame sequence with two handwritten digits 5 and 0. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second set of 10 frames is a mirror of the first and the last set of 10 frames is a copy of the first.

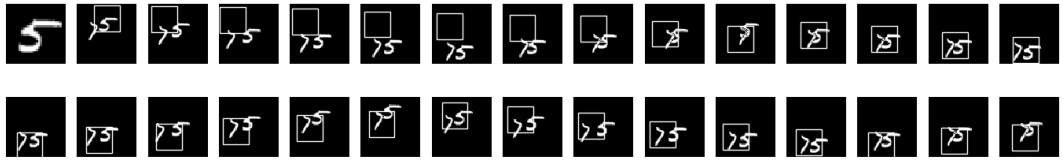

Figure 4.55: A visual representation of the GOTURN model tracking a 30 frame sequence with two handwritten digits 5 and 7. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track. The second set of 10 frames is a mirror of the first and the last set of 10 frames is a copy of the first.

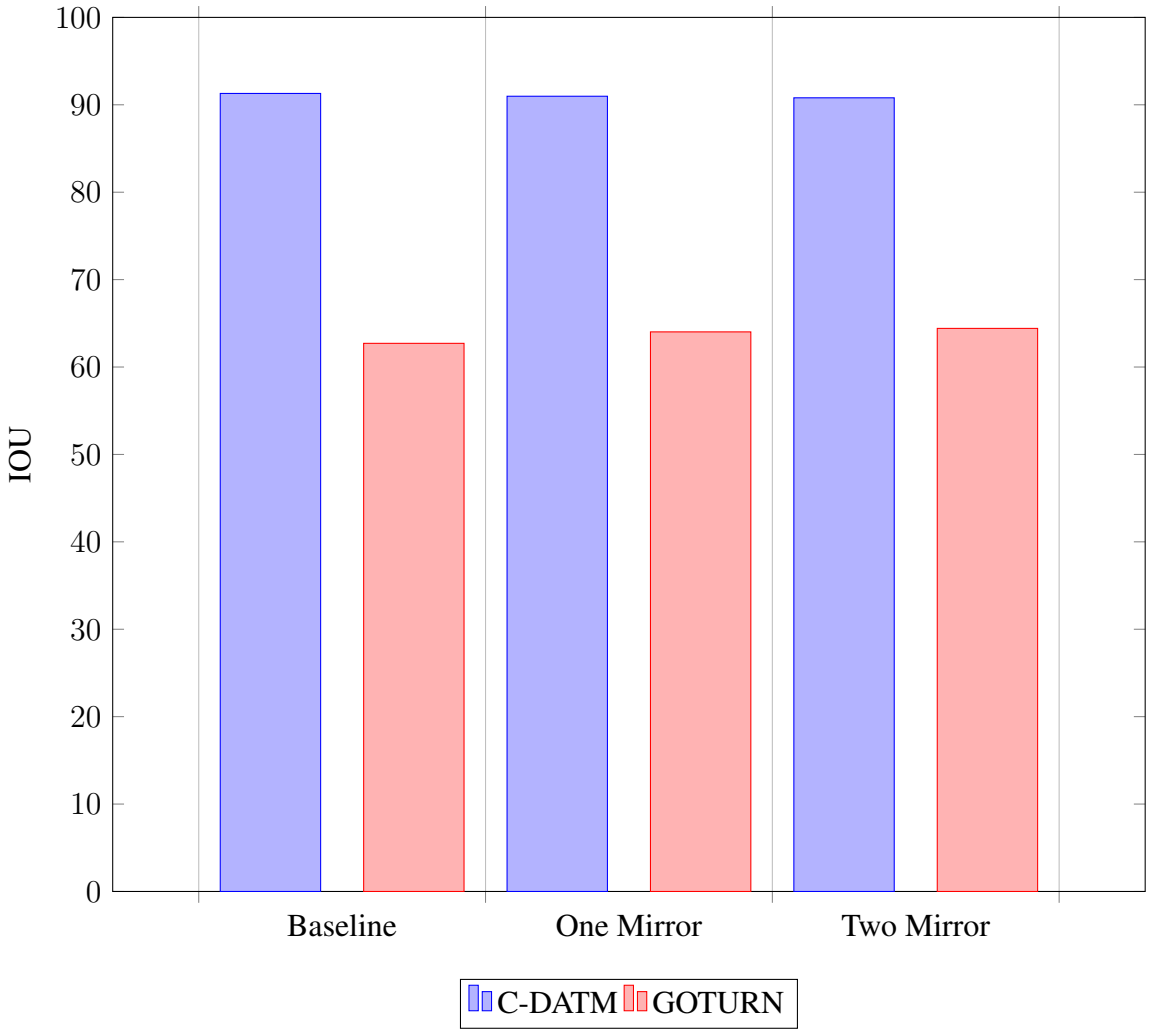

Figure 4.56: Bar graph representation of the kinematics experiments on single-digit data.

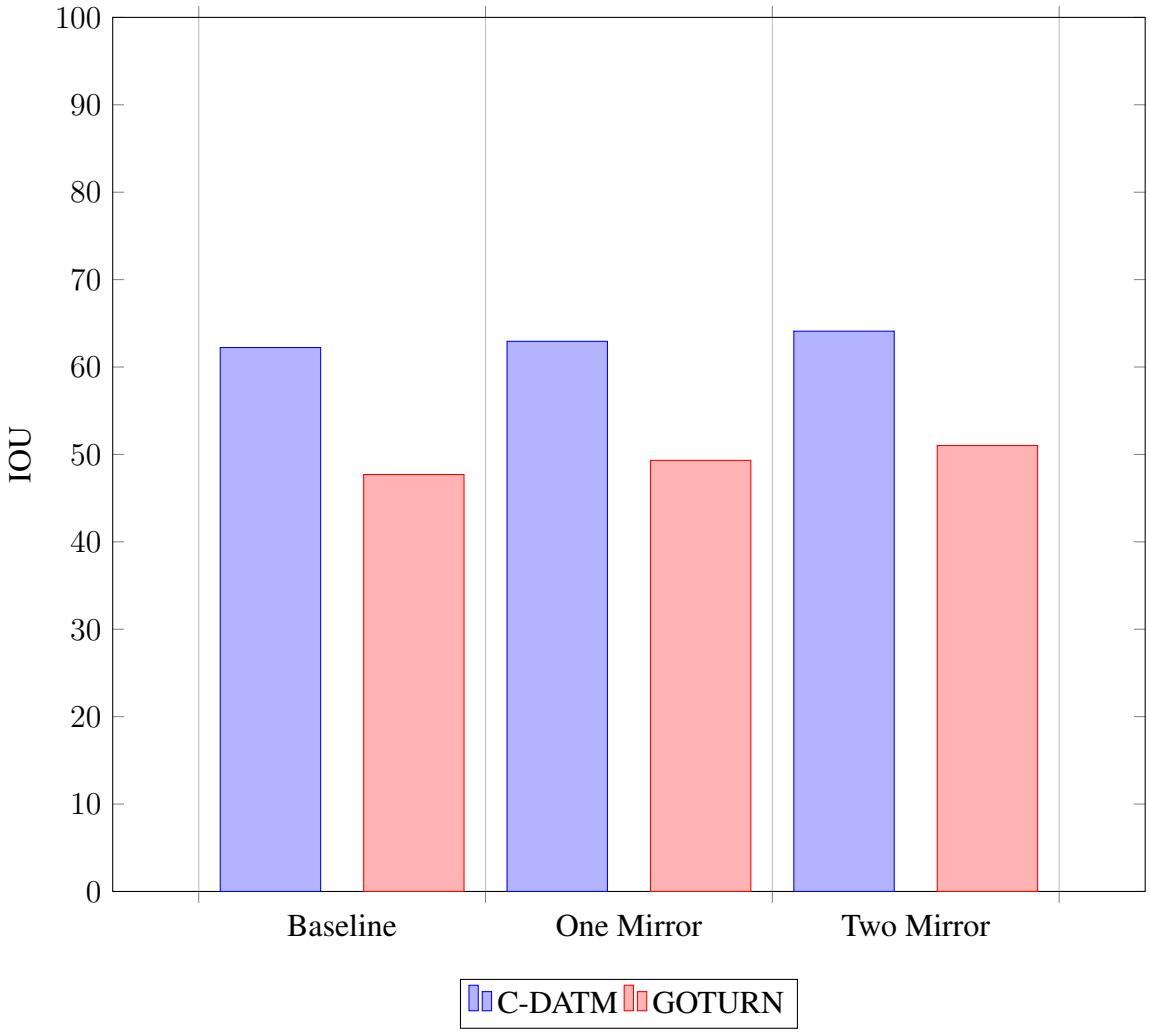

Figure 4.57: Bar graph representation of the kinematics experiments on two-digit data.

4.3 Object Speed Test

Attention methods are dependent on having the target at least partially present in the area of view in order to predict the movement of the bounding box and to be able to re-center the bounding box. This is a fundamental limitation to attention-based models. Because of this limitation, it is important that the object does not travel faster than the network can adjust the bounding box. To test how quickly the bounding box can move, and show how well each model deals with faster moving objects we introduce a data set where we skip frames, effectively simulating a faster moving object. A sample of this type of data is shown in Figure 4.58. All data from this experiment can be found in Table 4.3 and Table 4.4. A bar graph representations of the Tables are in Figures 4.83 and 4.84.

Figure 4.58: This Figure represents a single frame skipped shown in rows 1 and 2 (left). Two frames skipped is visualized in rows 1 and 2 (right).

When skipping one frame using the single-digit data C-DATM achieves an IOU of 73.10%, a decrease in 19.93% relative to the baseline. GOTURN achieves 44.18% IOU, a decrease in 29.55% relative to the baseline. For two-digit tracking and a single skipped frame C-DATM has an IOU of 59.30%, a decrease of 4.71%. GOTURN has an IOU of 38.09%, a decrease of 20.15%. A visualization of these results is shown in Figures 4.59-4.70.

Skipping two frames for a single-digit sequence results in an IOU of 60.95% for C-DATM, a decrease of 33.24%. GOTURN results in an IOU of 35.26%, a decrease of 43.77%. Skipping two frames for two-digits sequences results in an IOU of 54.80% for C-

DATM, a decrease of 11.94%. GOTURN results in a 32.42% IOU, a decrease of 32.03%.

Visualization of these results is found in Figures 4.71-4.82.

Figure 4.59: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 8. The first image in the first row represents the object the model is attempting to track. A frame was skipped between each.

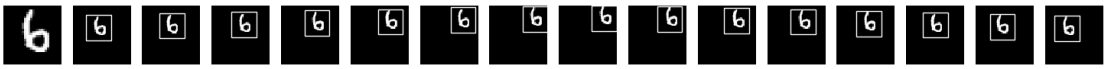

Figure 4.60: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 6. The first image in the first row represents the object the model is attempting to track. A frame was skipped between each.

Figure 4.61: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 7. The first image in the first row represents the object the model is attempting to track. A frame was skipped between each.

Figure 4.62: A visual representation of C-DATM tracking a 30 frame sequence with handwritten digits 5 and 7. The first image in the first row represents the object the model is attempting to track. A frame was skipped between each.

Figure 4.63: A visual representation of C-DATM tracking a 30 frame sequence with handwritten digits 1 and 1. The first image in the first row represents the object the model is attempting to track. A frame was skipped between each.

Figure 4.64: A visual representation of C-DATM tracking a 30 frame sequence with handwritten digits 5 and 0. The first image in the first row represents the object the model is attempting to track. A frame was skipped between each.

Figure 4.65: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 8. The first image in the first row represents the object the model is attempting to track. A frame was skipped between each.

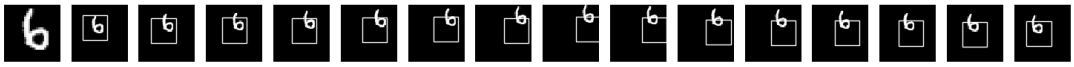

Figure 4.66: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 6. The first image in the first row represents the object the model is attempting to track. A frame was skipped between each.

Figure 4.67: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 7. The first image in the first row represents the object the model is attempting to track. A frame was skipped between each.

Figure 4.68: A visual representation of the GOTURN model tracking a 30 frame sequence with handwritten digits 5 and 7. The first image in the first row represents the object the model is attempting to track. A frame was skipped between each.

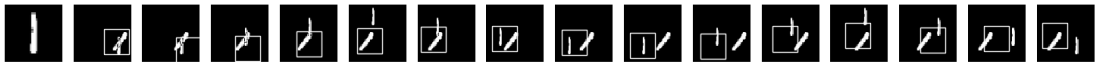

Figure 4.69: A visual representation of the GOTURN model tracking a 30 frame sequence with handwritten digits 1 and 1. The first image in the first row represents the object the model is attempting to track. A frame was skipped between each.

Figure 4.70: A visual representation of the GOTURN model tracking a 30 frame sequence with handwritten digits 5 and 0. The first image in the first row represents the object the model is attempting to track. A frame was skipped between each.

Figure 4.71: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 8. The first image in the first row represents the object the model is attempting to track. Two frames were skipped between each.

Figure 4.72: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 6. The first image in the first row represents the object the model is attempting to track. Two frames were skipped between each.

Figure 4.73: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 7. The first image in the first row represents the object the model is attempting to track. Two frames were skipped between each.

Figure 4.74: A visual representation of C-DATM tracking a 30 frame sequence with handwritten digits 5 and 7. The first image in the first row represents the object the model is attempting to track. Two frames were skipped between each.

Figure 4.75: A visual representation of C-DATM tracking a 30 frame sequence with handwritten digits 1 and 1. The first image in the first row represents the object the model is attempting to track. Two frames were skipped between each.

Figure 4.76: A visual representation of C-DATM tracking a 30 frame sequence with handwritten digits 5 and 0. The first image in the first row represents the object the model is attempting to track. Two frames were skipped between each.

Figure 4.77: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 8. The first image in the first row represents the object the model is attempting to track. Two frames were skipped between each.

Figure 4.78: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 6. The first image in the first row represents the object the model is attempting to track. Two frames were skipped between each.

Frame Skip	C-DATM 1D	C-DATM 1D Δ	GOTURN 1D	GOTURN 1D Δ
1	73.10%	-19.93%	44.18%	-29.55%
2	60.95%	-33.24%	35.26%	-43.77%

Table 4.3: The speed test results using the one-digit MNIST tracking set.

Figure 4.79: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 7. The first image in the first row represents the object the model is attempting to track. Two frames were skipped between each.

Figure 4.80: A visual representation of the GOTURN model tracking a 30 frame sequence with handwritten digits 5 and 7. The first image in the first row represents the object the model is attempting to track. Two frames were skipped between each.

Figure 4.81: A visual representation of the GOTURN model tracking a 30 frame sequence with handwritten digits 1 and 1. The first image in the first row represents the object the model is attempting to track. Two frames were skipped between each.

Figure 4.82: A visual representation of the GOTURN model tracking a 30 frame sequence with handwritten digits 5 and 0. The first image in the first row represents the object the model is attempting to track. Two frames were skipped between each.

Frame Skip	C-DATM 2D	C-DATM 2D Δ	GOTURN 2D	GOTURN 2D Δ
1	59.30%	-4.71%	38.09%	-20.15%
2	54.80%	-11.94%	32.42%	-32.03%

Table 4.4: The speed test results using the two-digit MNIST tracking set.

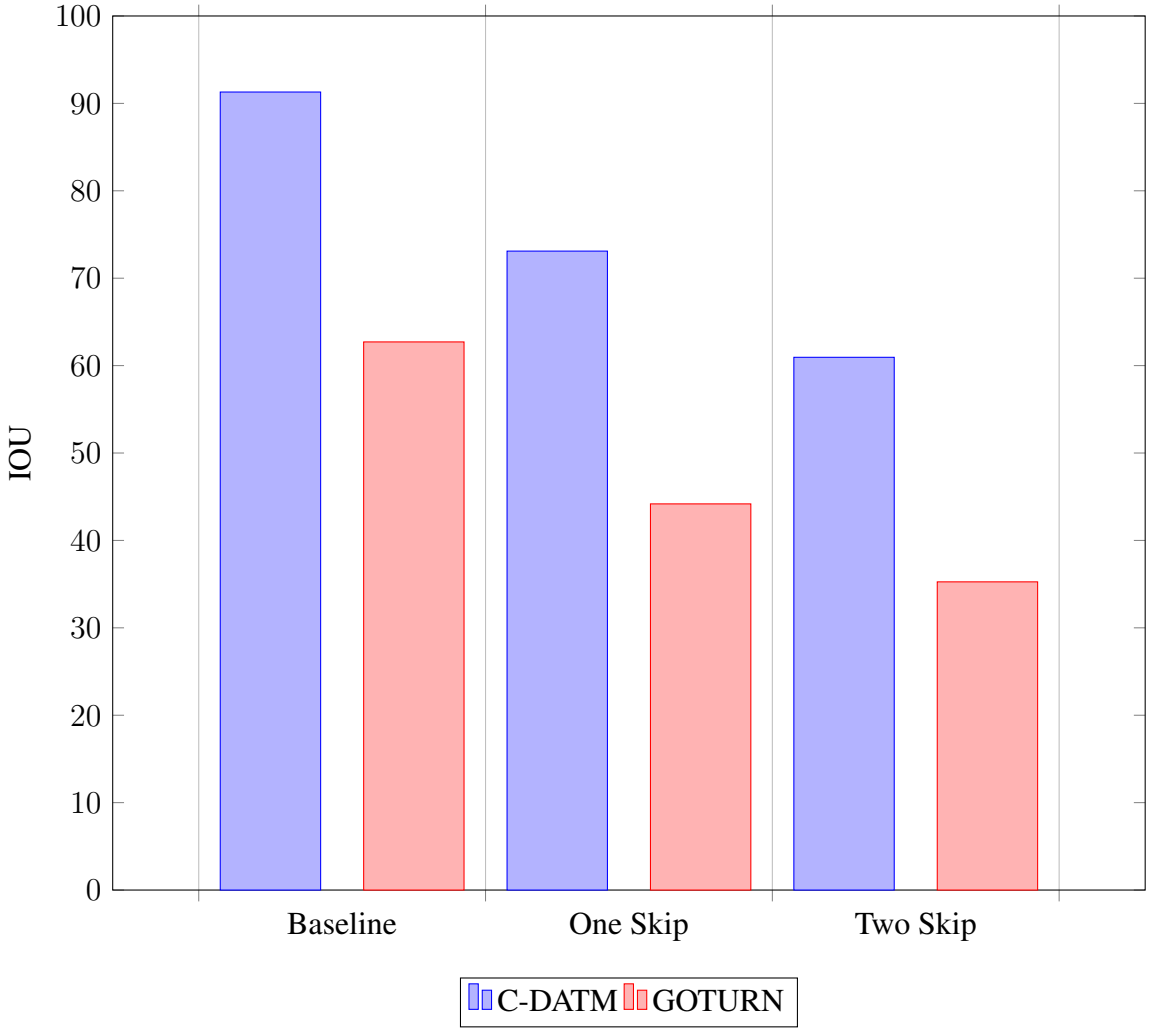

Figure 4.83: Bar graph representation of the speed experiments on single-digit data.

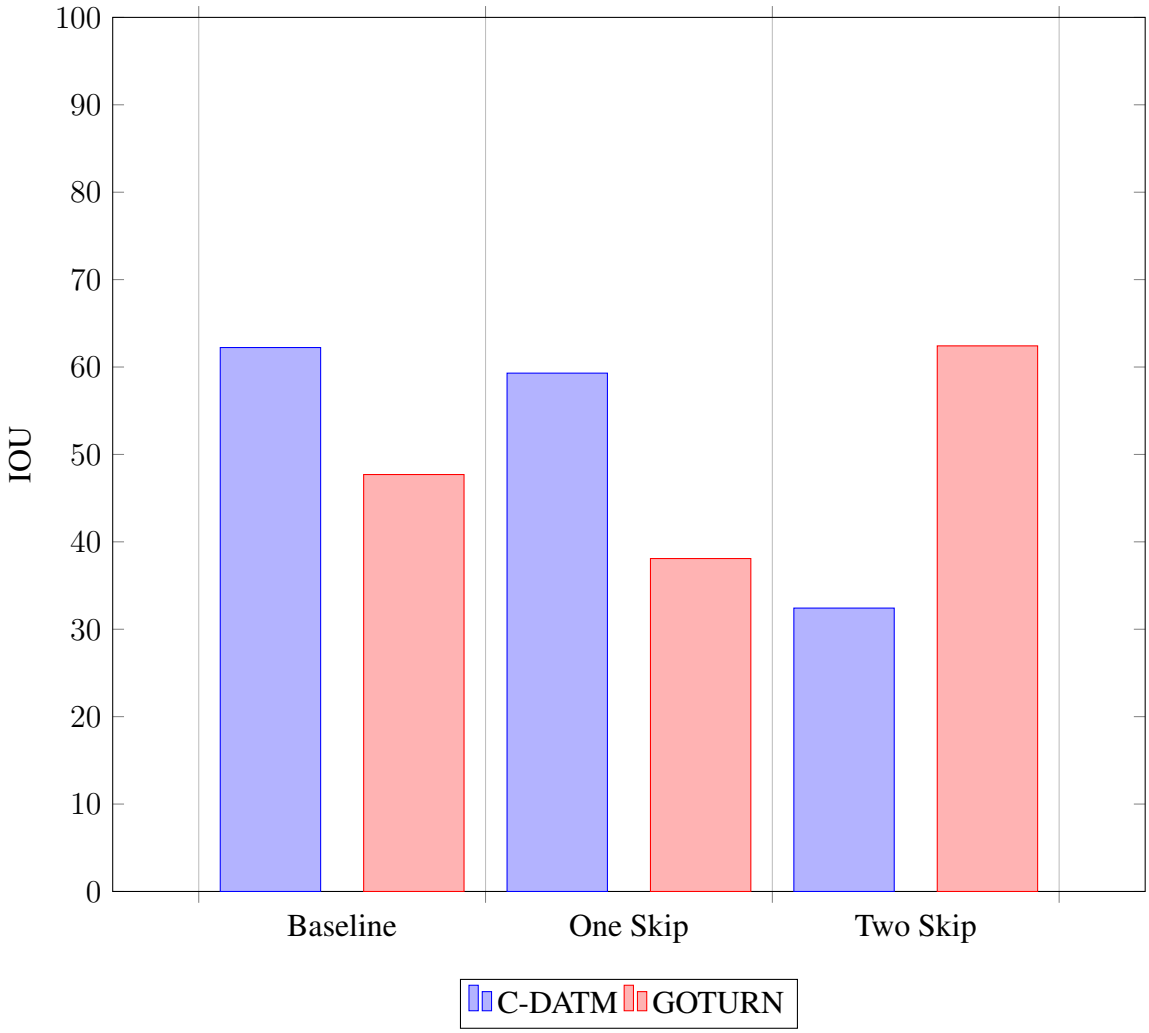

Figure 4.84: Bar graph representation of the speed experiments on two-digit data.

Blur Rate	C-DATM 1D	C-DATM 1D Δ	GOTURN 1D	GOTURN 1D Δ
2	92.62%	+1.45%	69.65%	+11.07%
3	91.23%	-0.077%	70.18%	+11.91%
4	89.86%	-1.58%	70.22%	+11.98%
5	88.34%	-3.24%	70.23%	+11.99%

Table 4.5: The blur rate test results using the one-digit MNIST tracking set.

Blur Rate	C-DATM 2D	C-DATM 2D Δ	GOTURN 2D	GOTURN 2D Δ
2	62.25%	+0.032%	49.86%	+4.53%
3	62.13%	-0.16%	49.99%	+4.80%
4	61.96%	-0.43%	50.11%	+5.05%
5	61.57%	-1.06%	50.09%	+5.01%

Table 4.6: The blur rate test results using the two-digit MNIST tracking set.

4.4 Blur Test

In this experiment we blur the images to make it more difficult for the network to focus on specific object-features.

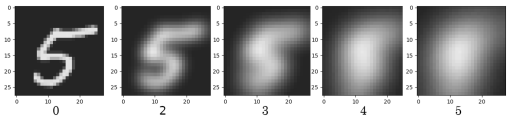

Figure 4.85: This Figure represents the blur levels used in the experiments.

To explore how important features are to the model's ability to track objects, experiments were performed that blur the object thus providing less information to the network. For C-DATM, the condition, or object used for comparisons, is blurred. For GOTURN, the information from the previous bounding box in the sequence is blurred. There were four different Gaussian filters used to blur the images (see Figure 4.85). The results of these tests are given in Table 4.5 and Table 4.6. Bar graphs for these tables are in Figures 4.86 and 4.87. Figures 4.88-4.119 visualize assorted blur rates using both C-DATM and GOTURN.

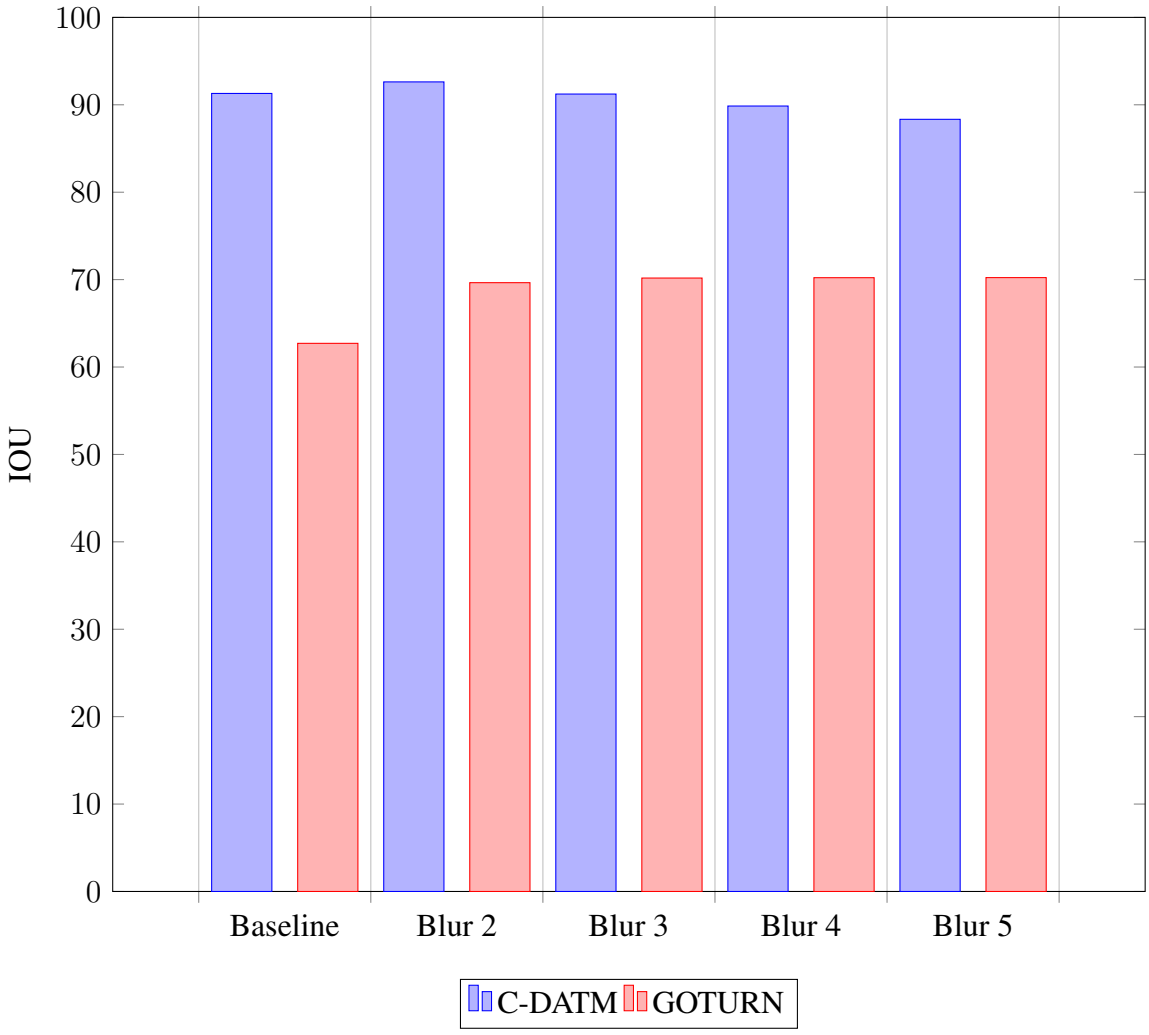

Figure 4.86: Bar graph representation of the blur experiments on single-digit data.

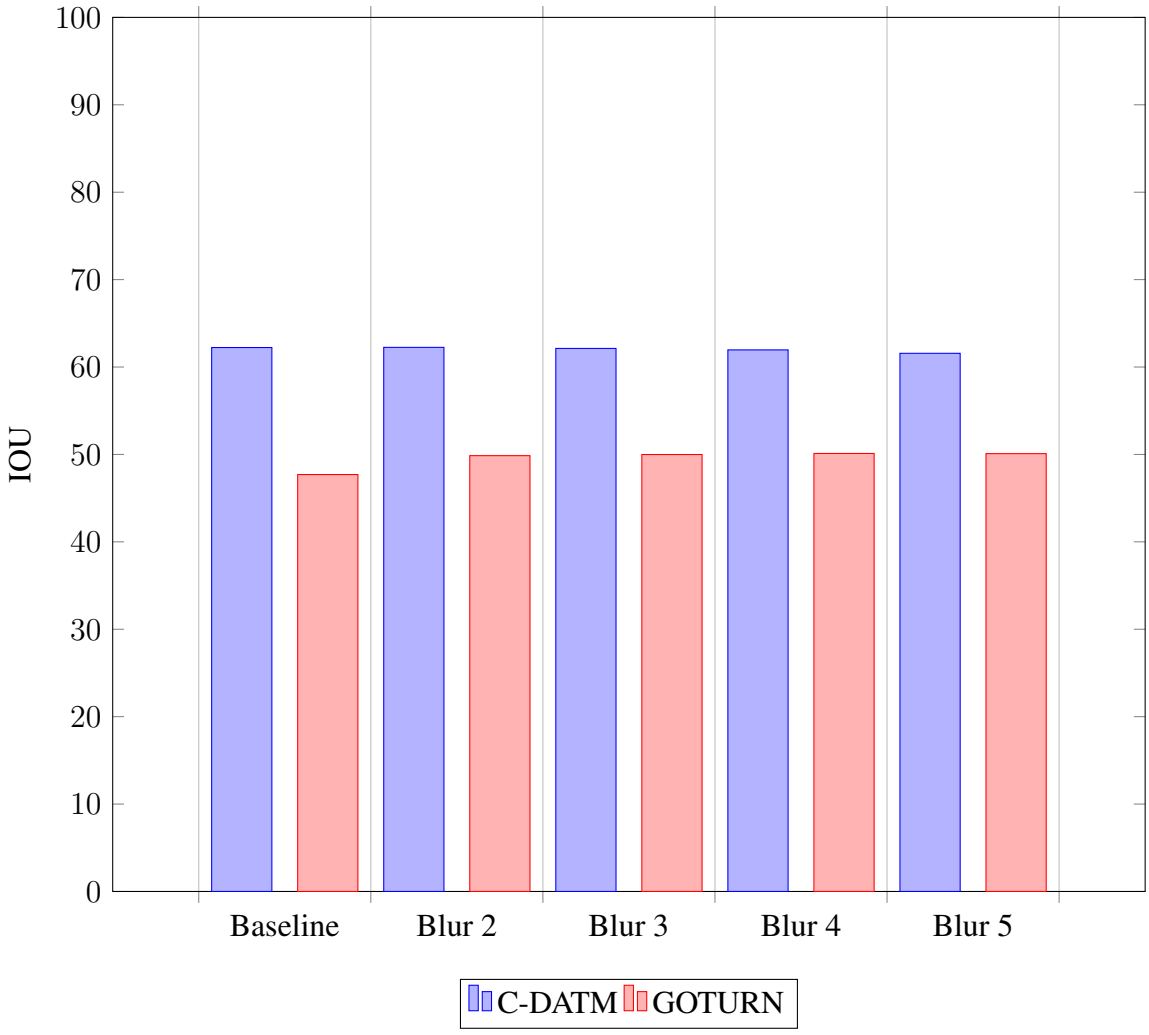

Figure 4.87: Bar graph representation of the blur experiments on two-digit data.

Figure 4.88: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 8. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 2 was applied to the condition.

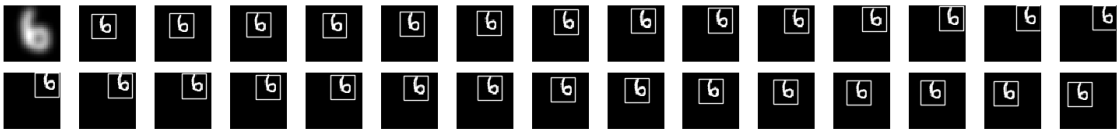

Figure 4.89: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 6. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 2 was applied to the condition.

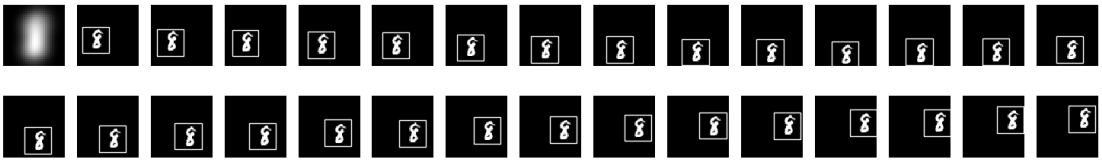

Figure 4.90: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 8. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 3 was applied to the condition.

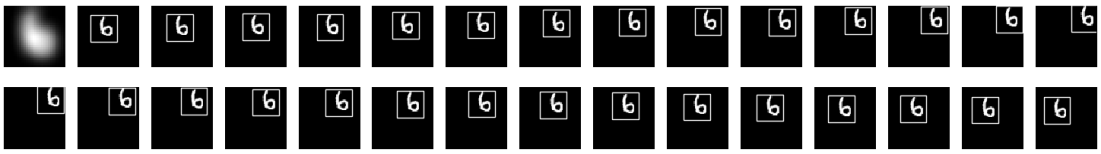

Figure 4.91: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 6. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 3 was applied to the condition.

Figure 4.92: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 8. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 4 was applied to the condition.

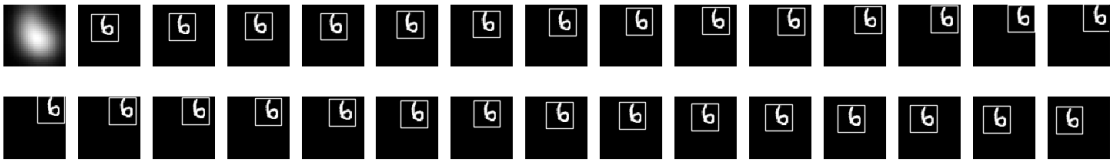

Figure 4.93: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 6. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 4 was applied to the condition.

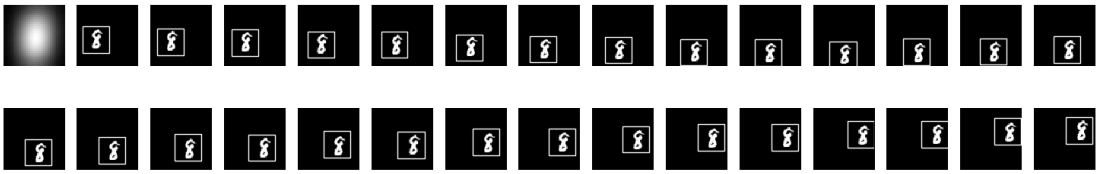

Figure 4.94: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 8. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 5 was applied to the condition.

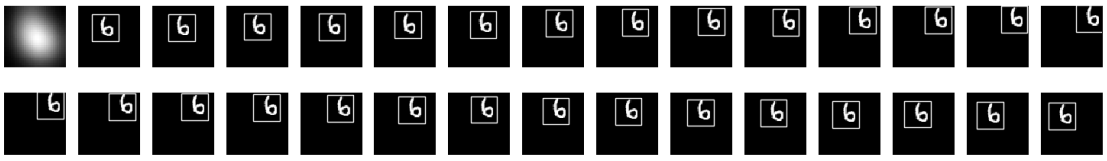

Figure 4.95: A visual representation of C-DATM tracking a 30 frame sequence with a handwritten digit 6. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 5 was applied to the condition.

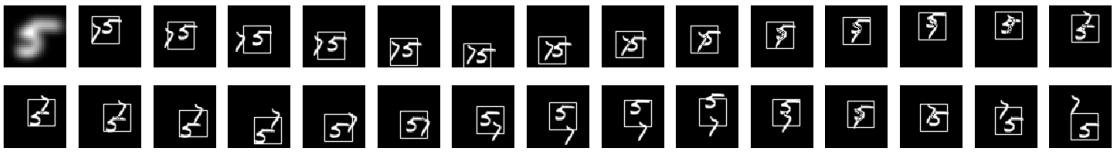

Figure 4.96: A visual representation of C-DATM tracking a 30 frame sequence with handwritten digits 5 and 7. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 2 was applied to the condition.

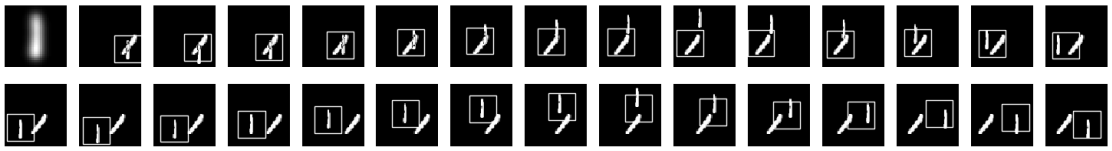

Figure 4.97: A visual representation of C-DATM tracking a 30 frame sequence with handwritten digits 1 and 1. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 2 was applied to the condition.

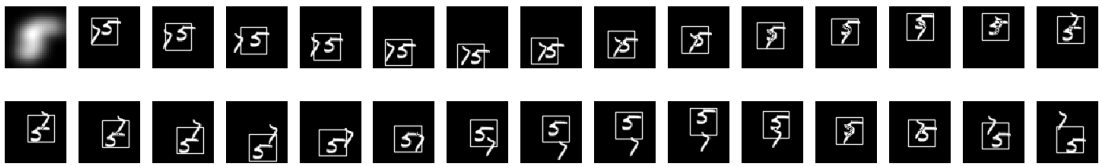

Figure 4.98: A visual representation of C-DATM tracking a 30 frame sequence with handwritten digits 5 and 7. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 3 was applied to the condition.

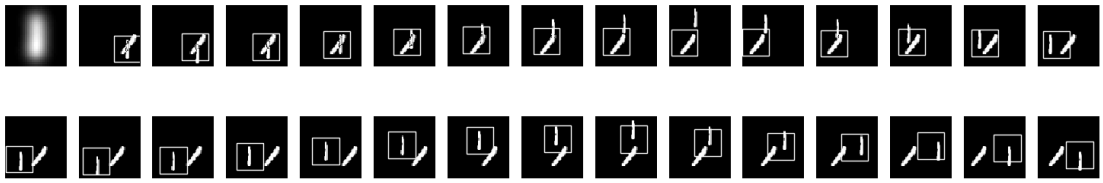

Figure 4.99: A visual representation of C-DATM tracking a 30 frame sequence with handwritten digits 1 and 1. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 3 was applied to the condition.

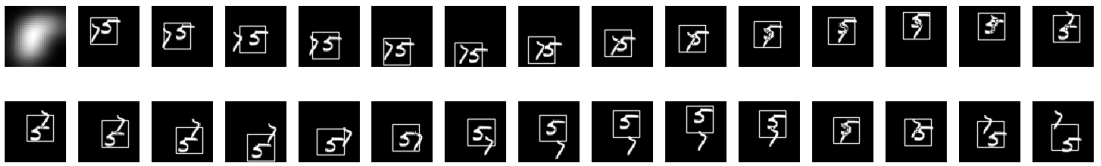

Figure 4.100: A visual representation of C-DATM tracking a 30 frame sequence with handwritten digits 5 and 7. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 4 was applied to the condition.

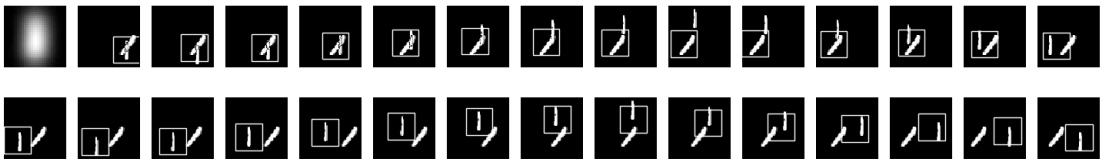

Figure 4.101: A visual representation of C-DATM tracking a 30 frame sequence with handwritten digits 1 and 1. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 4 was applied to the condition.

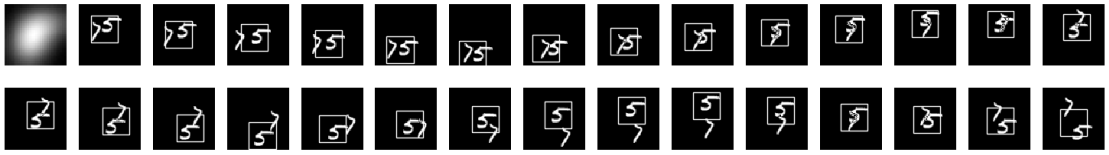

Figure 4.102: A visual representation of C-DATM tracking a 30 frame sequence with handwritten digits 5 and 7. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 5 was applied to the condition.

Figure 4.103: A visual representation of C-DATM tracking a 30 frame sequence with handwritten digits 1 and 1. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 5 was applied to the condition.

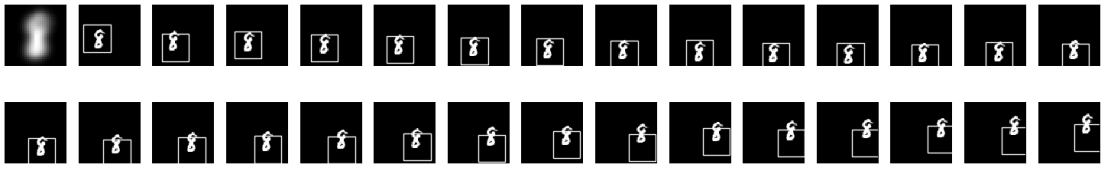

Figure 4.104: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 8. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 2 was applied to the condition.

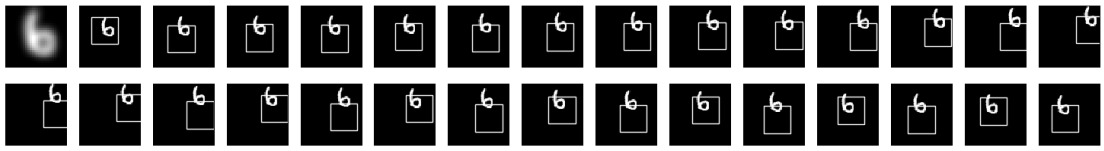

Figure 4.105: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 6. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 2 was applied to the condition.

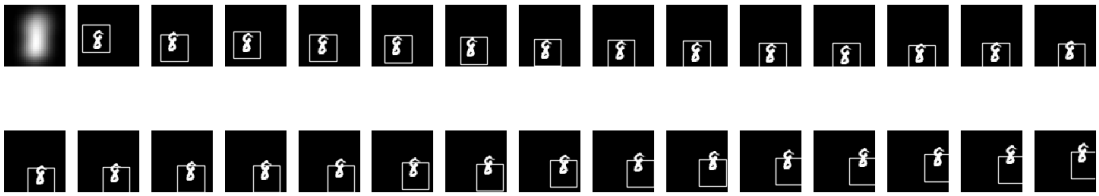

Figure 4.106: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 8. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 3 was applied to the condition.

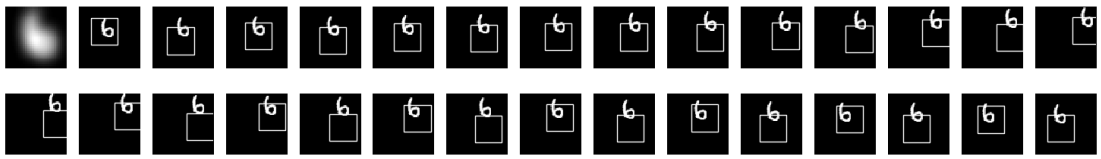

Figure 4.107: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 6. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 3 was applied to the condition.

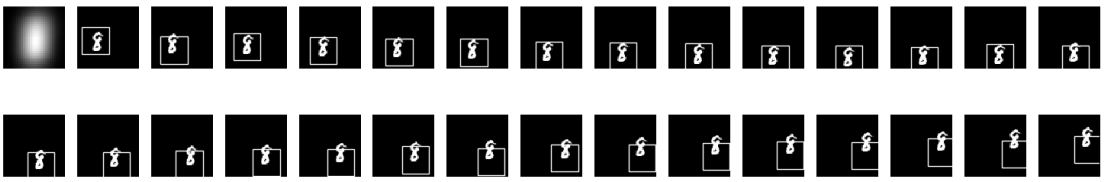

Figure 4.108: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 8. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 4 was applied to the condition.

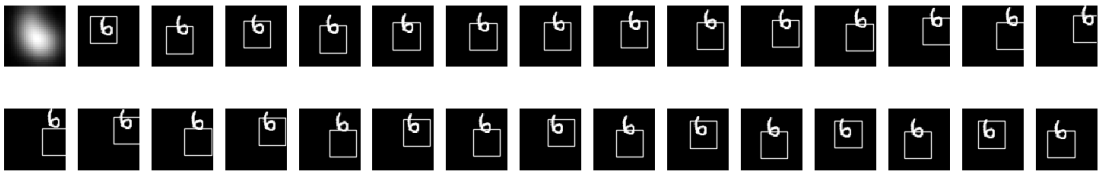

Figure 4.109: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 6. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 4 was applied to the condition.

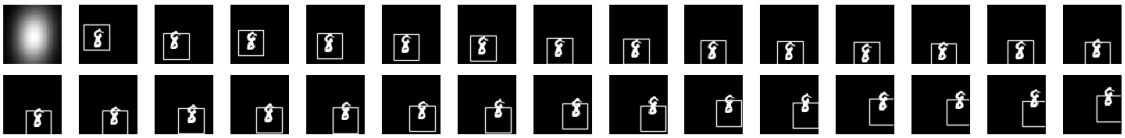

Figure 4.110: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 8. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 5 was applied to the condition.

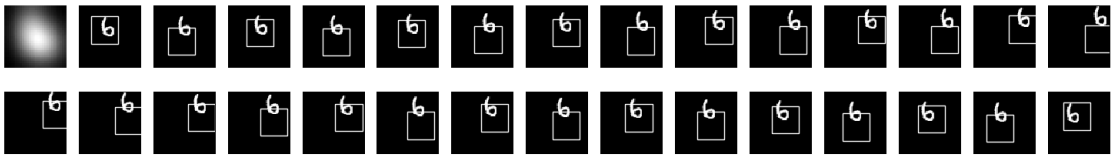

Figure 4.111: A visual representation of the GOTURN model tracking a 30 frame sequence with a handwritten digit 6. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 5 was applied to the condition.

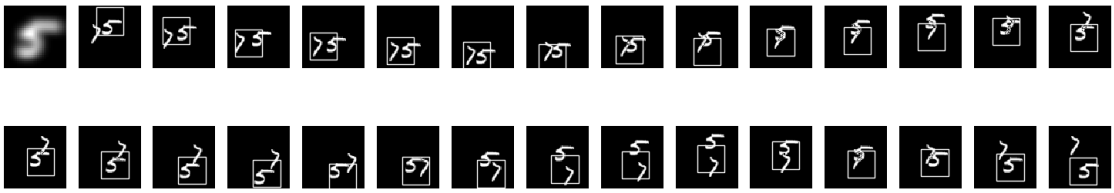

Figure 4.112: A visual representation of the GOTURN model tracking a 30 frame sequence with handwritten digits 5 and 7. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 2 was applied to the condition.

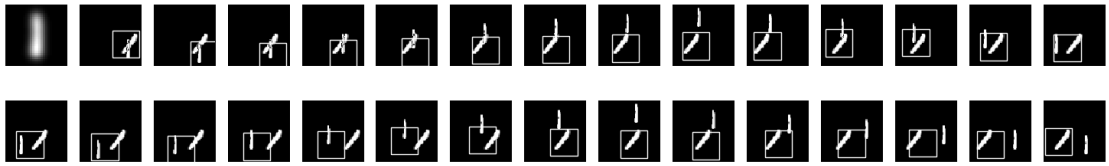

Figure 4.113: A visual representation of the GOTURN model tracking a 30 frame sequence with handwritten digits 1 and 1. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 2 was applied to the condition.

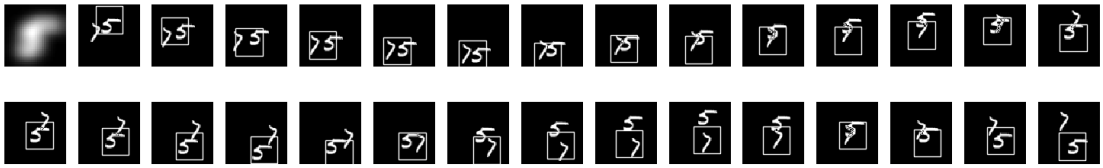

Figure 4.114: A visual representation of the GOTURN model tracking a 30 frame sequence with handwritten digits 5 and 7. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 3 was applied to the condition.

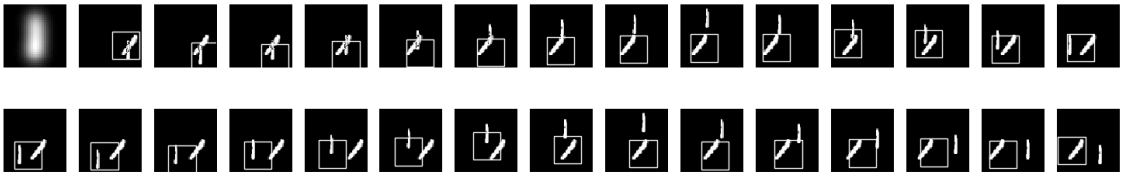

Figure 4.115: A visual representation of the GOTURN model tracking a 30 frame sequence with handwritten digits 1 and 1. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 3 was applied to the condition.

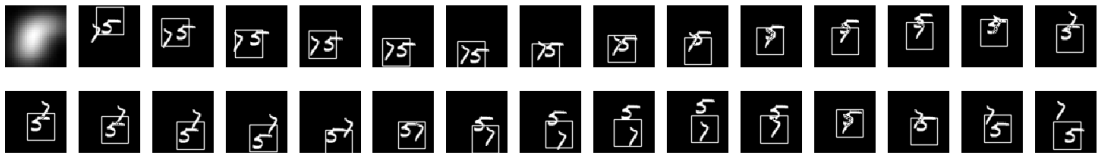

Figure 4.116: A visual representation of the GOTURN model tracking a 30 frame sequence with handwritten digits 5 and 7. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 4 was applied to the condition.

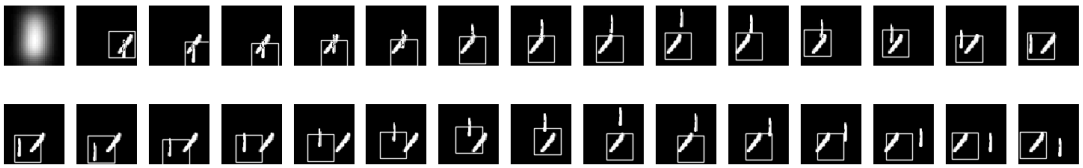

Figure 4.117: A visual representation of the GOTURN model tracking a 30 frame sequence with handwritten digits 1 and 1. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 4 was applied to the condition.

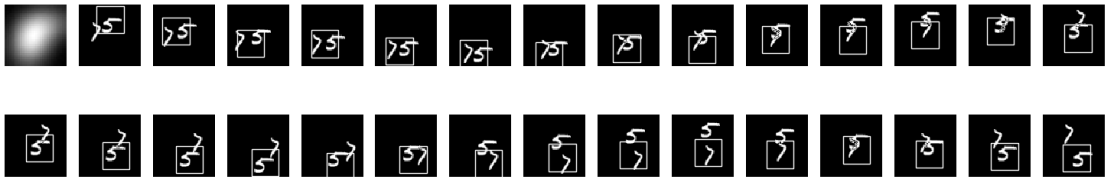

Figure 4.118: A visual representation of the GOTURN model tracking a 30 frame sequence with handwritten digits 5 and 7. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 5 was applied to the condition.

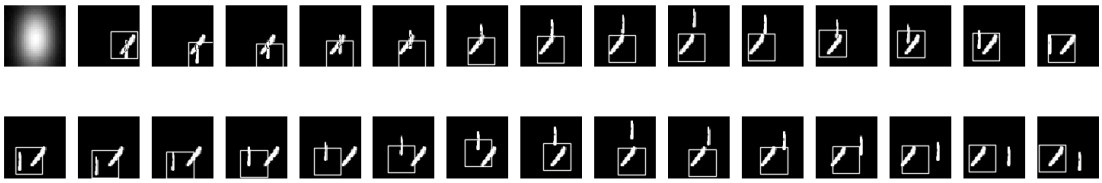

Figure 4.119: A visual representation of the GOTURN model tracking a 30 frame sequence with handwritten digits 1 and 1. The first image in the first row represents the object the model is attempting to track. A Gaussian blur rate of 5 was applied to the condition.

4.5 Salt and Pepper Noise

Salt and pepper experiments are done for two reasons, to show C-DATM is better than a standard correlation test and to subject the model to noise that is not just a confuser. The salt and pepper noise is created by defining a percent amount of pixels affected, and changing that number of random pixels in the scene to either 1(salt) or 0(pepper). The ratio of salt to pepper for this experiment is 50/50. For single object tracking, compared to baseline

experiments 1% salt and pepper (Figure 4.120) results in an 91.11 % IOU. 5% salt and pepper (Figure 4.121) results in 90.02% IOU. 10% salt and pepper (Figure 4.122) results in 89.86% IOU. 50% salt and pepper (Figure 4.123) results in 61.85% IOU.

For two object tracking, compared to baseline experiments 1% salt and pepper results in an 61.46 % IOU. 5% salt and pepper results in 61.22% IOU. 10% salt and pepper results in 60.65% IOU. 50% salt and pepper results in 50.52% IOU.

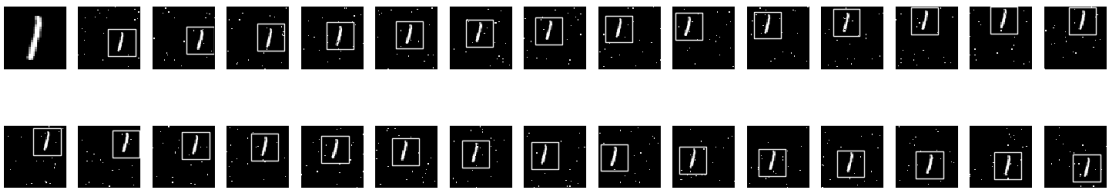

Figure 4.120: A visual representation of the C-DATM model tracking a 30 frame sequence with a handwritten digit 1. The first image in the first row represents the object the model is attempting to track. A 1% Salt and Pepper noise filter is applied to the scene.

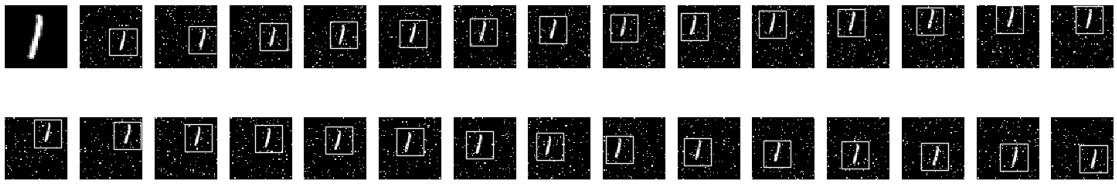

Figure 4.121: A visual representation of the C-DATM model tracking a 30 frame sequence with a handwritten digit 1. The first image in the first row represents the object the model is attempting to track. A 5% Salt and Pepper noise filter is applied to the scene.

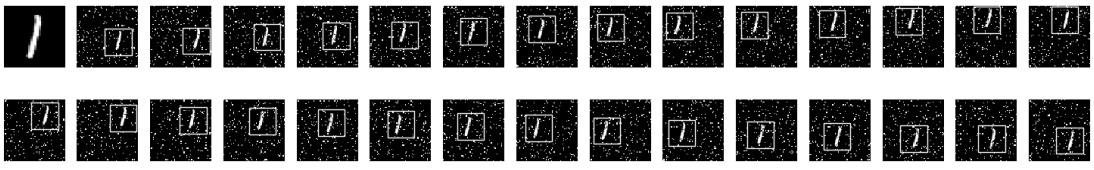

Figure 4.122: A visual representation of the C-DATM model tracking a 30 frame sequence with a handwritten digit 1. The first image in the first row represents the object the model is attempting to track. A 10% Salt and Pepper noise filter is applied to the scene.

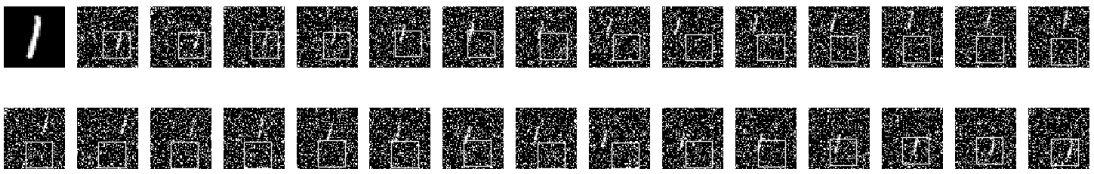

Figure 4.123: A visual representation of the C-DATM model tracking a 30 frame sequence with a handwritten digit 1. The first image in the first row represents the object the model is attempting to track. A 50% Salt and Pepper noise filter is applied to the scene.

4.6 Training with 2-Digits

All prior experiments only use single-digit MNIST tracking data to train the models. This allows us to examine C-DATM to generalize tracking results to more complex problems that are not present in the training set. However, if we add Moving MNIST data with two digits in the scene to our training data we can improve the overall tracking performance of C-DATM. If do not use any single-digit Moving MNIST data to train our model a solution cannot be reached, because the data is too complicated to optimize the network. However, if the data is mixed so that a portion the sequences contain one-digit and others contain two-digits, an acceptable solution is found. Through experimentation the best mix of training data contained 70% one-digit and 30% two-digit. Added two-digit training data is visualized in Figures 4.124-4.128.

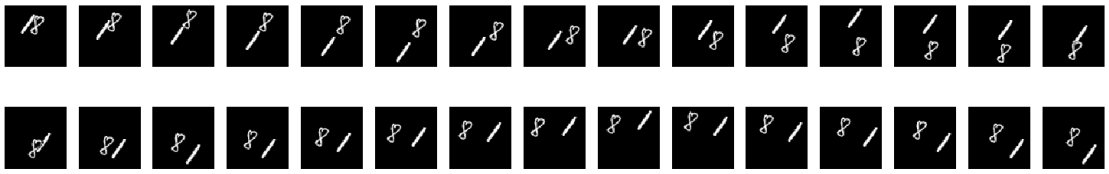

Figure 4.124: A visual representation of a 30 frame training sequence with handwritten digits 1 and 8. The first row is the first 15 images in the sequence, and the second row is the last 15.

Figure 4.125: A visual representation of a 30 frame training sequence with handwritten digits 3 and 5. The first row is the first 15 images in the sequence, and the second row is the last 15.

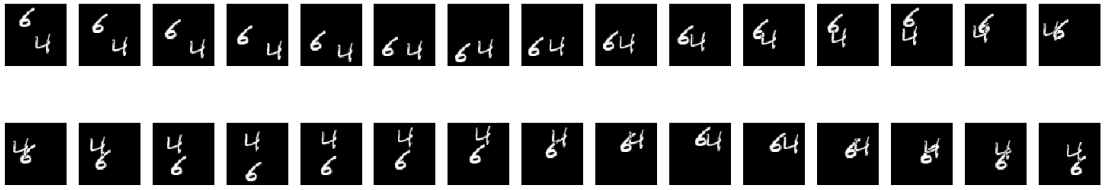

Figure 4.126: A visual representation of a 30 frame training sequence with handwritten digits 6 and 4. The first row is the first 15 images in the sequence, and the second row is the last 15.

Figure 4.127: A visual representation of a 30 frame training sequence with handwritten digits 7 and 7. The first row is the first 15 images in the sequence, and the second row is the last 15.

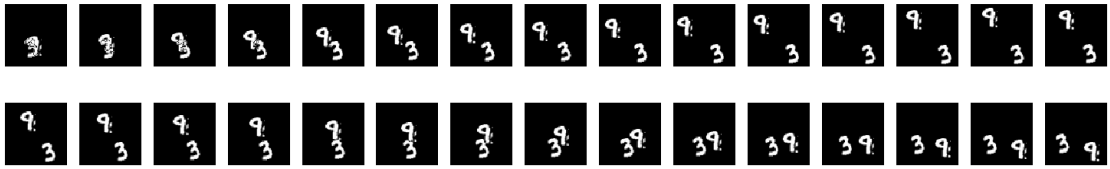

Figure 4.128: A visual representation of a 30 frame training sequence with handwritten digits 9 and 3. The first row is the first 15 images in the sequence, and the second row is the last 15.

Using 10,000 sequences with a 70/30 split we use the same parameters for C-DATM as in our baseline training. The network is trained for a total of 20 epochs. The first 10

epochs have a learning rate of 1e-4 and the last 10 a rate of 1e-5. The total mean squared error of the final model is 340,348. The training of this model is visualized in Figure 4.129. C-DATM produces an IOU of 94.11% while tracking a single MNIST digit, and an IOU of 69.35% for two-digit MNIST tracking. This is an improvement of our previous baseline results by 2.81% and 7.12% respectively. Results of this training set mixture on C-DATM are visualized in Figures 4.130-4.135.

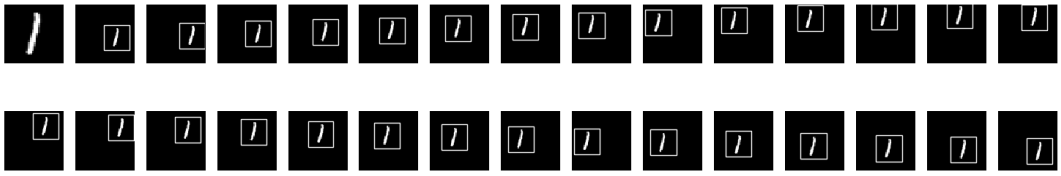

Figure 4.130: A visual representation of C-DATM trained with a mixture of 1 and 2-digit training data, tracking a 30 frame sequence with a handwritten 1 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

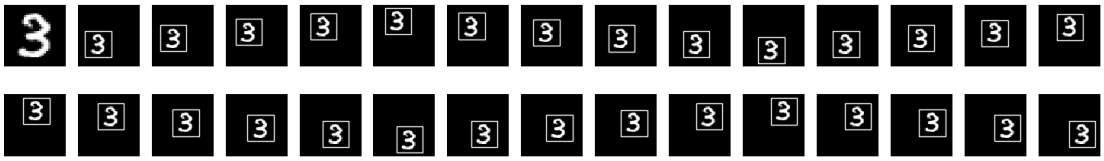

Figure 4.131: A visual representation of C-DATM trained with a mixture of 1 and 2-digit training data, tracking a 30 frame sequence with a handwritten 3 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

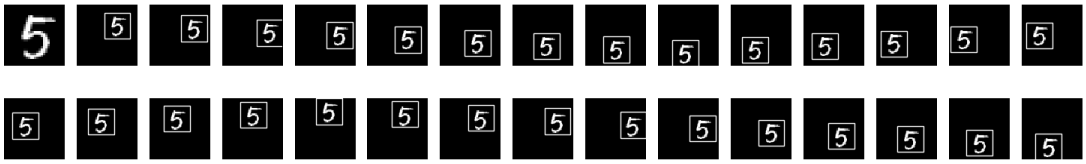

Figure 4.132: A visual representation of C-DATM trained with a mixture of 1 and 2-digit training data, tracking a 30 frame sequence with a handwritten 5 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

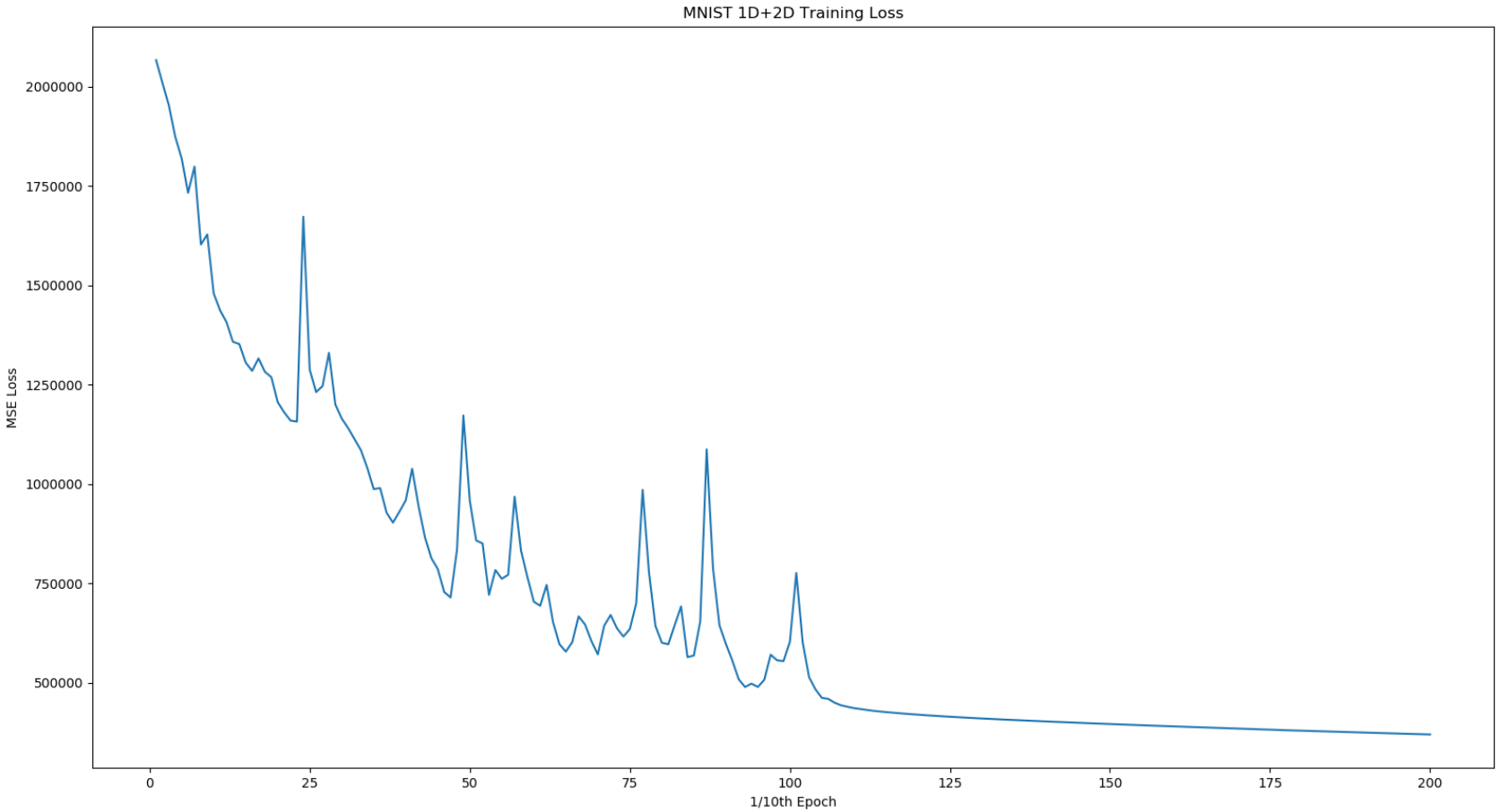

Figure 4.129: The training loss of C-DATM trained using MNIST 1D and 2D.

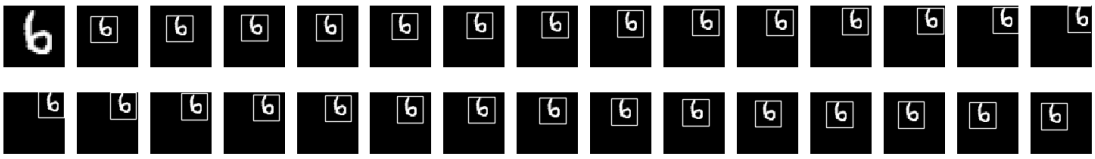

Figure 4.133: A visual representation of C-DATM trained with a mixture of 1 and 2-digit training data, tracking a 30 frame sequence with a handwritten 6 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

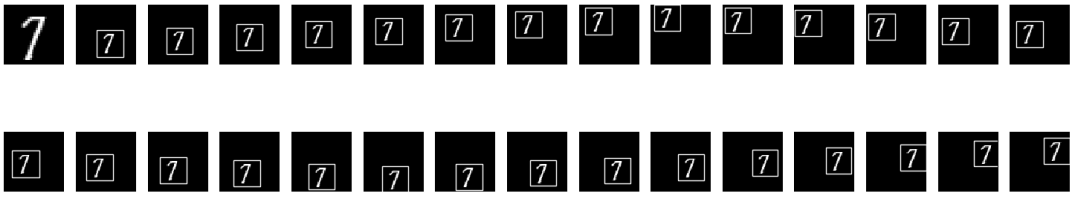

Figure 4.134: A visual representation of C-DATM trained with a mixture of 1 and 2-digit training data, tracking a 30 frame sequence with a handwritten 7 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

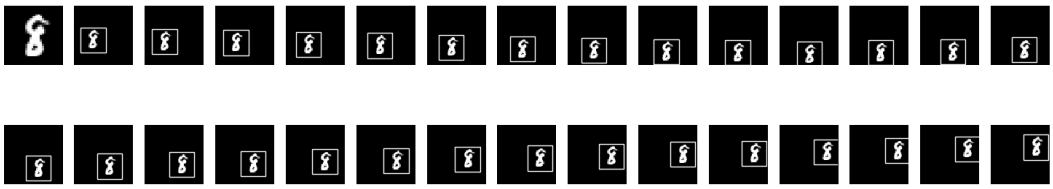

Figure 4.135: A visual representation of C-DATM trained with a mixture of 1 and 2-digit training data, tracking a 30 frame sequence with a handwritten 8 digit. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

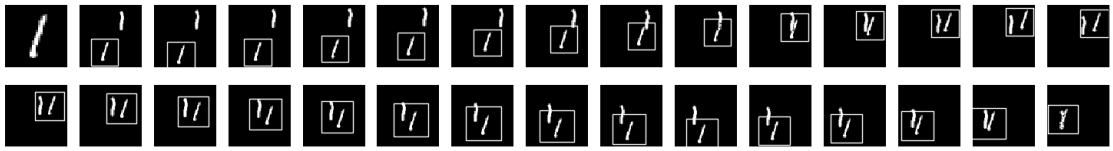

Figure 4.136: A visual representation of C-DATM trained with a mixture of 1 and 2-digit training data, tracking a 30 frame sequence with handwritten digits 1 and 1. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

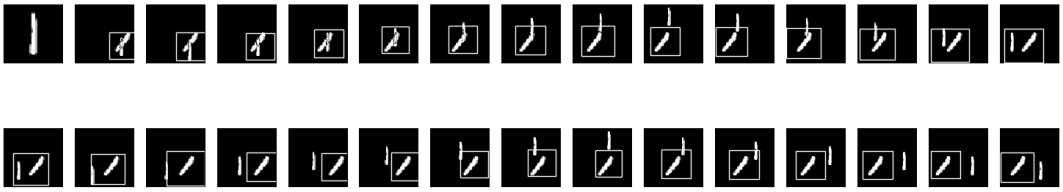

Figure 4.137: A visual representation of C-DATM trained with a mixture of 1 and 2-digit training data, tracking a 30 frame sequence with handwritten digits 1 and 1. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

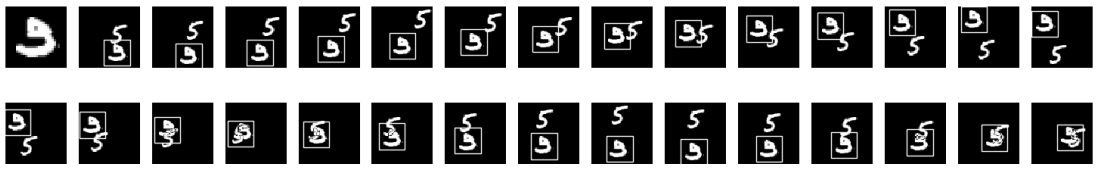

Figure 4.138: A visual representation of C-DATM trained with a mixture of 1 and 2-digit training data, tracking a 30 frame sequence with handwritten digits 3 and 5. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

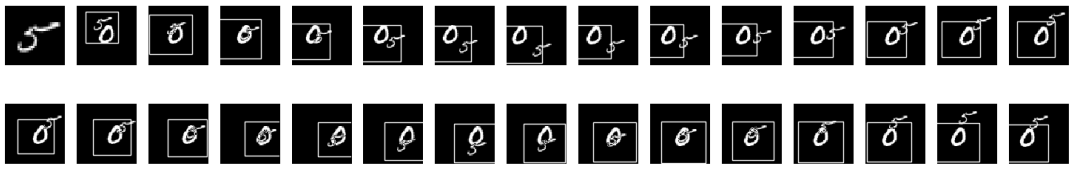

Figure 4.139: A visual representation of C-DATM trained with a mixture of 1 and 2-digit training data, tracking a 30 frame sequence with handwritten digits 5 and 0. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

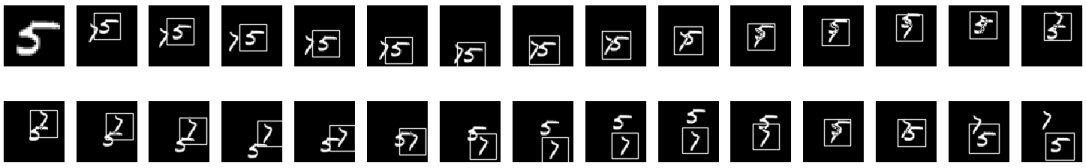

Figure 4.140: A visual representation of C-DATM trained with a mixture of 1 and 2-digit training data, tracking a 30 frame sequence with handwritten digits 5 and 7. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

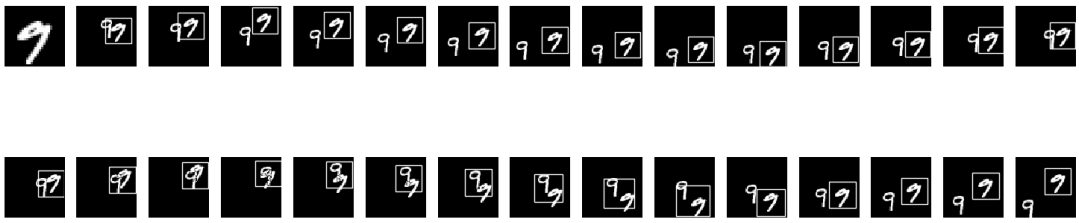

Figure 4.141: A visual representation of C-DATM trained with a mixture of 1 and 2-digit training data, tracking a 30 frame sequence with handwritten digits 9 and 9. The first row is the first 14 images in the sequence, excluding the first frame, and the second row is the last 15. The first image in the first row represents the object the model is attempting to track.

4.7 Fashion MNIST

The Fashion MNIST[43] set contains fashion objects, but still has the same properties of the standard MNIST set. Training and test data sets created using the Fashion MNIST follow the same pattern used for the standard MNIST moving data set. A collection of 10,000 sequences were created of length of 30 frames for training C-DATM, and 1,000 sequences were generated for testing. There are versions of these sets, one containing a single fashion object, and another containing two fashion objects. The set containing two objects have the type of objects randomly chosen from the available classes, meaning their class can be the same, but the specific object is different. The objects move through the frame independently of each-other, but do occlude each other via summation of the pixel

intensities of both objects. Training data is visualized in Figures 4.142-4.144.

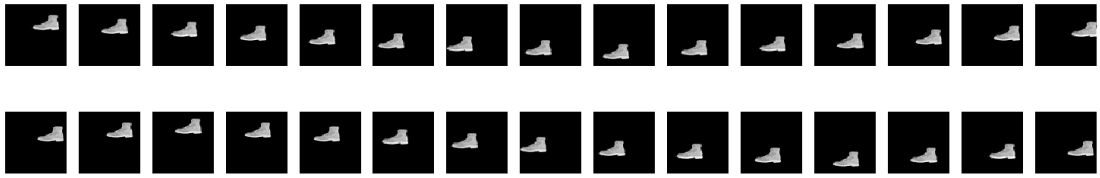

Figure 4.142: A visual representation of a 30 frame training sequence with the fashion object boots. The first row is the first 15 images in the sequence, and the second row is the last 15.

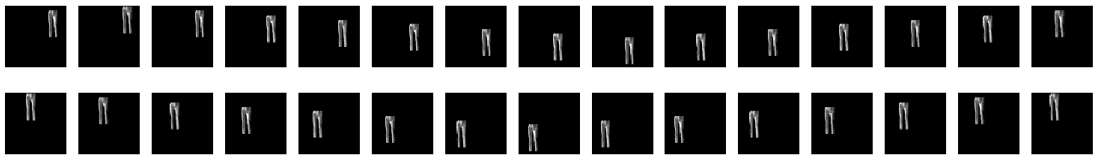

Figure 4.143: A visual representation of a 30 frame training sequence with the fashion object pants. The first row is the first 15 images in the sequence, and the second row is the last 15.

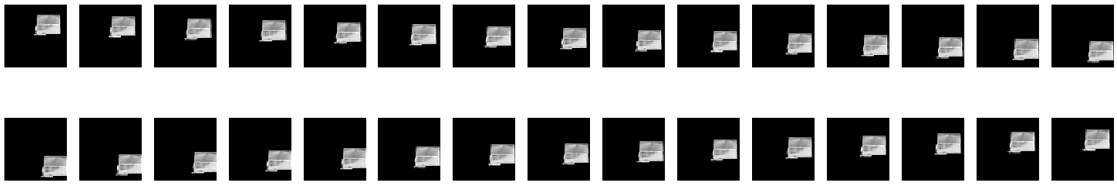

Figure 4.144: A visual representation of a 30 frame training sequence with the fashion object purse. The first row is the first 15 images in the sequence, and the second row is the last 15.

The C-DATM parameters used for Fashion MNIST are the same as those used for the baseline training using MNIST data. The C-DATM is trained using a single fashion object, and a mix of a single objects and two object data sets. Both methods use a total of 10,000 sequences. For single object training there are a total of 5 epochs at a learning rate of 1e-4 resulting in a total mean squared error of 194,652. The IOU for a single fashion object of C-DATM is 94.11% and two fashion objects track at an IOU of 60.98%. Training of C-DATM

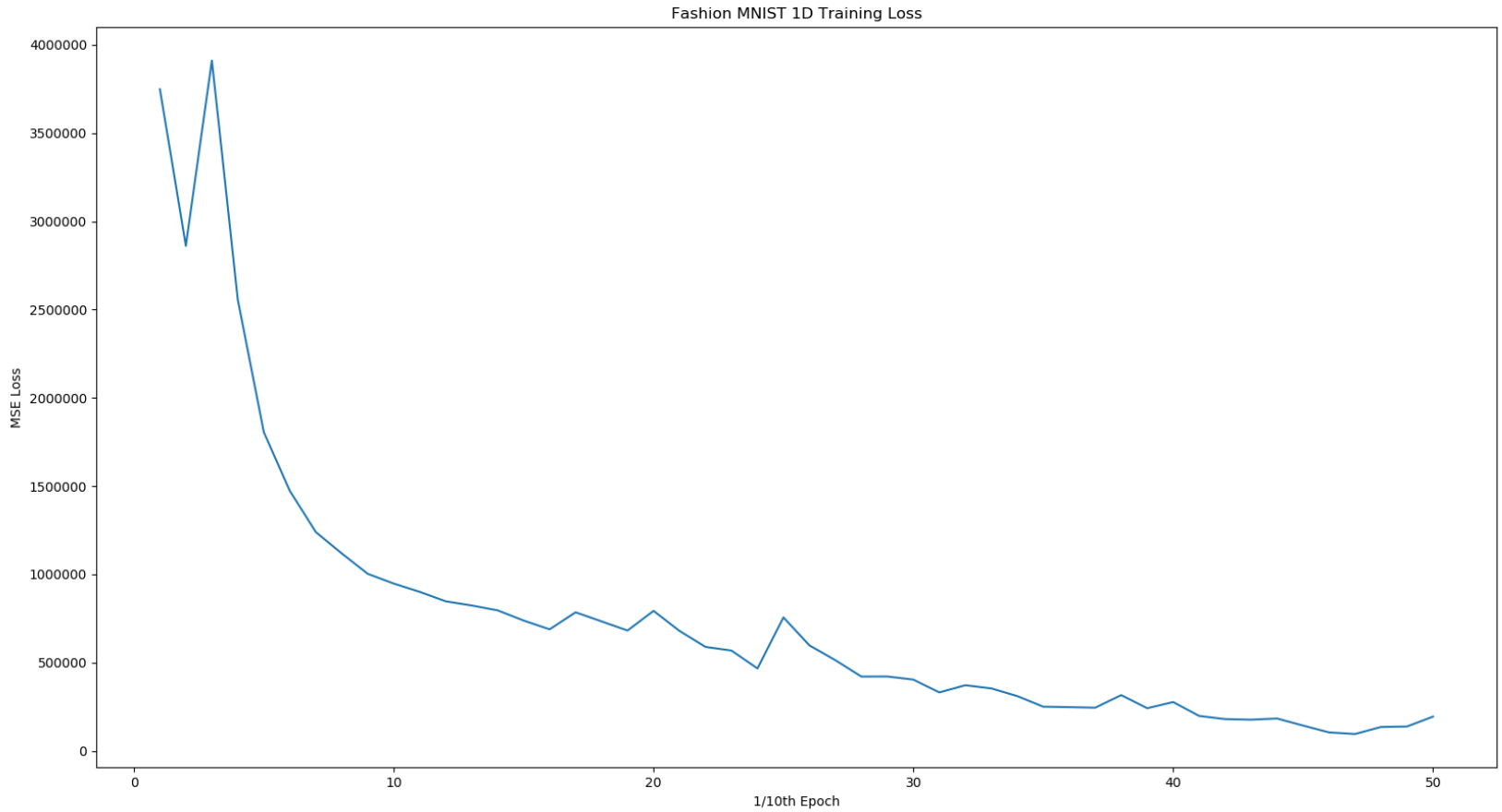

Figure 4.145: The training loss of the fashion MNIST C-DATM.

is visualized in Figure 4.145. Visualization of the results are in Figures 4.146-4.151.

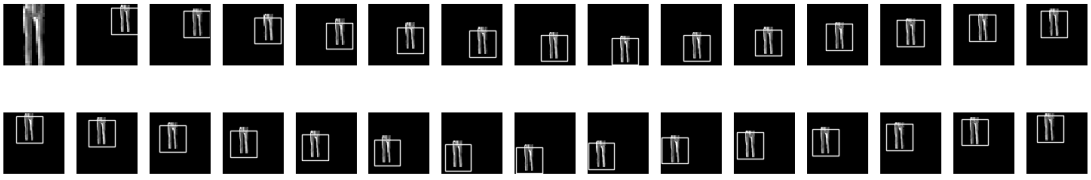

Figure 4.146: A visual representation of C-DATM tracking a 30 frame sequence with the fashion object pants. The first row is the first 15 images in the sequence, excluding the first frame, and the second row is the last 15. The first frame in the first row is the object the model is attempting to track.

Figure 4.147: A visual representation of C-DATM tracking a 30 frame sequence with the fashion object shoes. The first row is the first 15 images in the sequence, excluding the first frame, and the second row is the last 15. The first frame in the first row is the object the model is attempting to track.

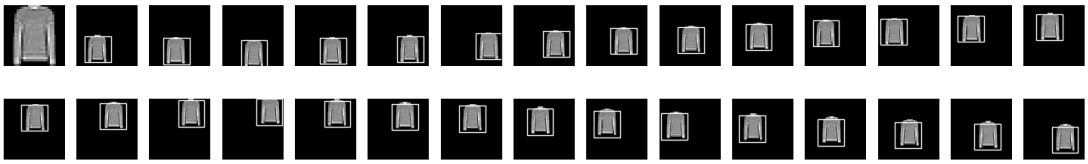

Figure 4.148: A visual representation of C-DATM tracking a 30 frame sequence with the fashion object sweater. The first row is the first 15 images in the sequence, excluding the first frame, and the second row is the last 15. The first frame in the first row is the object the model is attempting to track.

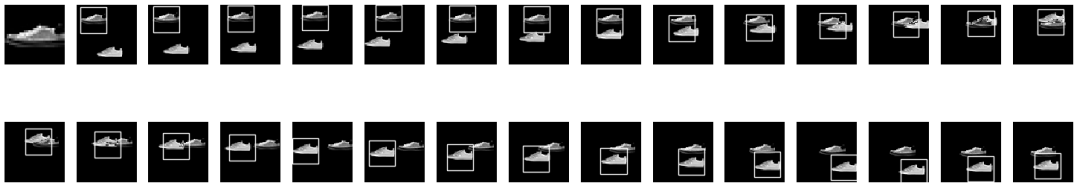

Figure 4.149: A visual representation of C-DATM tracking a 30 frame sequence with two fashion objects, shoes and shoes. The first row is the first 15 images in the sequence, excluding the first frame, and the second row is the last 15. The first frame in the first row is the object the model is attempting to track.

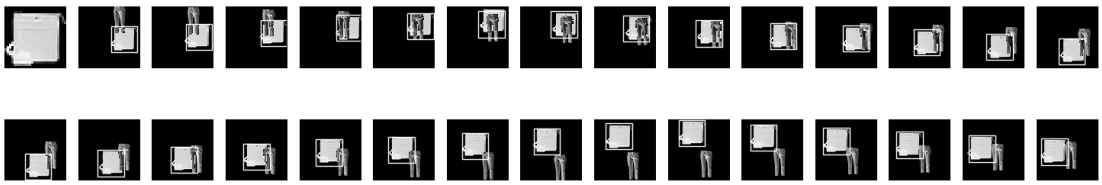

Figure 4.150: A visual representation of C-DATM tracking a 30 frame sequence with two fashion objects, a purse and pants. The first row is the first 15 images in the sequence, excluding the first frame, and the second row is the last 15. The first frame in the first row is the object the model is attempting to track.

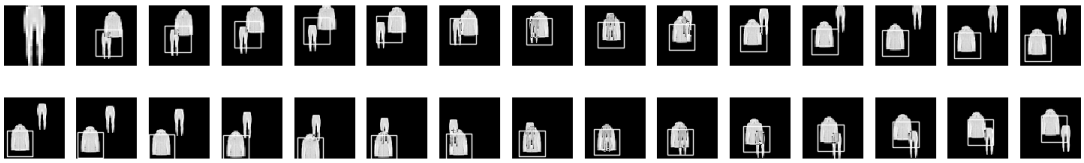

Figure 4.151: A visual representation of C-DATM tracking a 30 frame sequence with two fashion objects, pants and a sweater. The first row is the first 15 images in the sequence, excluding the first frame, and the second row is the last 15. The first frame in the first row is the object the model is attempting to track.

Experiments suggest that again a 70% single object to 30% two object data give the best results for mixed data training. The added training data with two objects is visualized in Figures 4.152-4.154. Training for a total of 20 epochs allows the network to reach a mean squared error of 435,605. The first 10 epochs have a learning rate of 1e-4 and the last 10 have a rate of 1e-5. C-DATM's training is visualized in Figure 4.155. The IOU for

a single fashion object of C-DATM is 93.95% and two fashion objects track at an IOU of 69.35%. Visualizations of results using the mixed training data are in Figures 4.156-4.161.

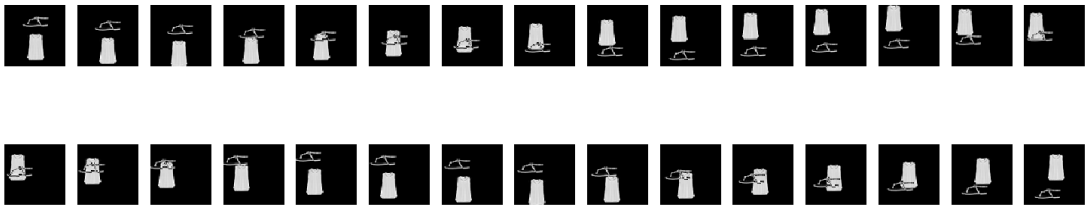

Figure 4.152: A visual representation of a 30 frame training sequence with fashion objects sandals and a shirt. The first row is the first 15 images in the sequence, and the second row is the last 15.

Figure 4.153: A visual representation of a 30 frame training sequence with fashion objects shorts and shoes. The first row is the first 15 images in the sequence, and the second row is the last 15.

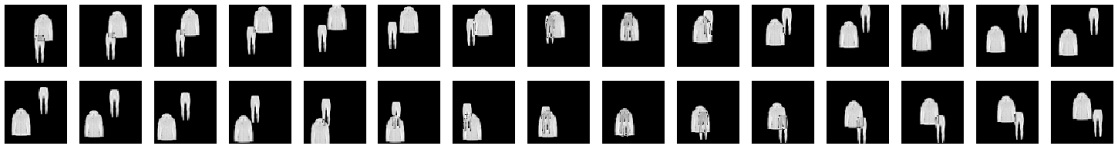

Figure 4.154: A visual representation of a 30 frame training sequence with fashion objects pants and jackets. The first row is the first 15 images in the sequence, and the second row is the last 15.

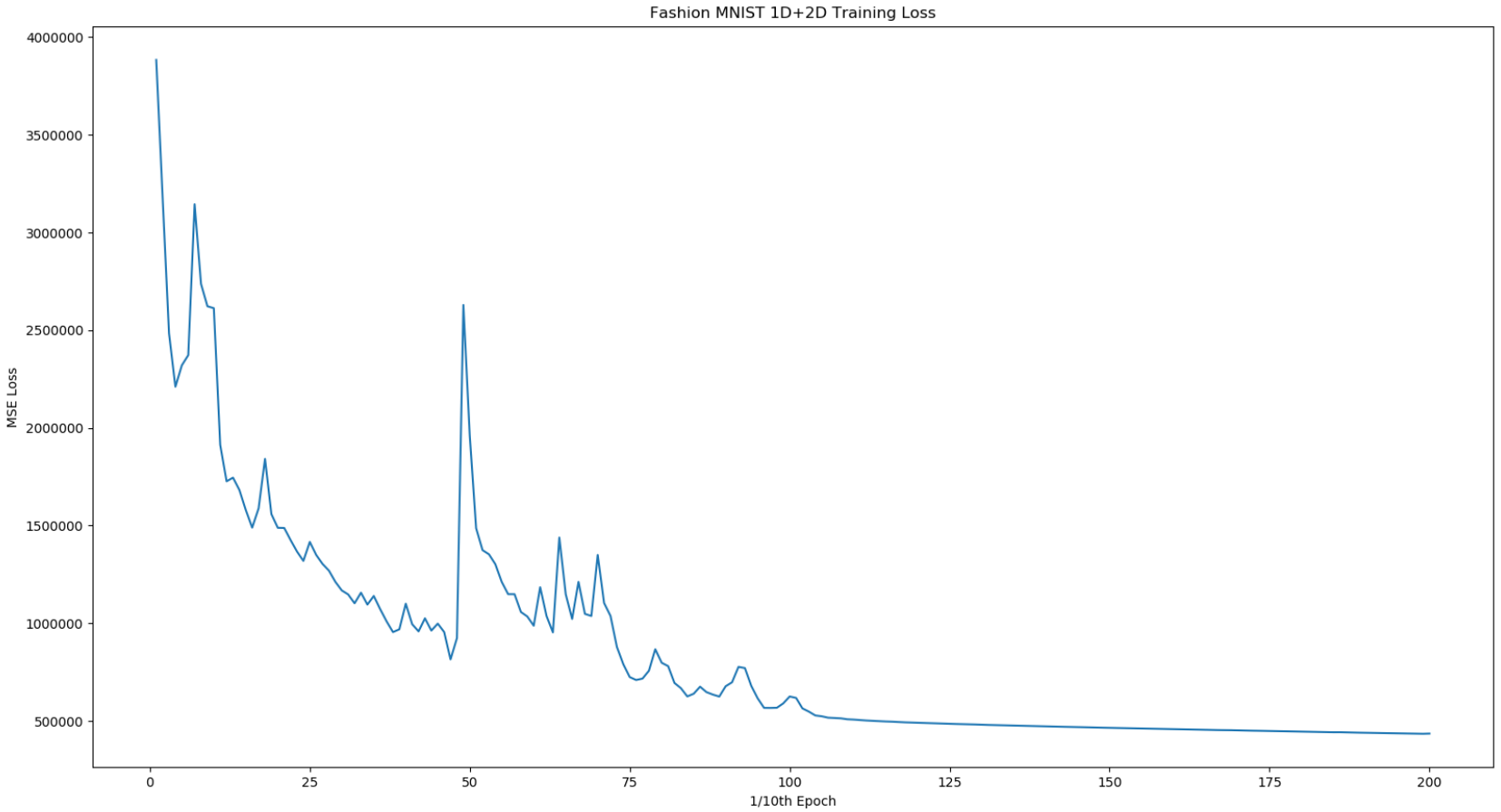

Figure 4.155: The training loss of C-DATM trained using Fashion MNIST 1D and 2D.

Figure 4.157: A visual representation of C-DATM tracking a 30 frame sequence with the fashion object shoes. C-DATM is trained with a mixture of training data containing both one object and two. The first row is the first 15 images in the sequence, excluding the first frame, and the second row is the last 15. The first frame in the first row is the object the model is attempting to track.

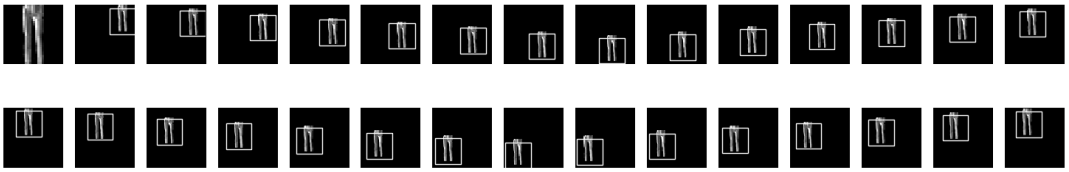

Figure 4.156: A visual representation of C-DATM tracking a 30 frame sequence with the fashion object pants. C-DATM is trained with a mixture of training data containing both one object and two. The first row is the first 15 images in the sequence, excluding the first frame, and the second row is the last 15. The first frame in the first row is the object the model is attempting to track.

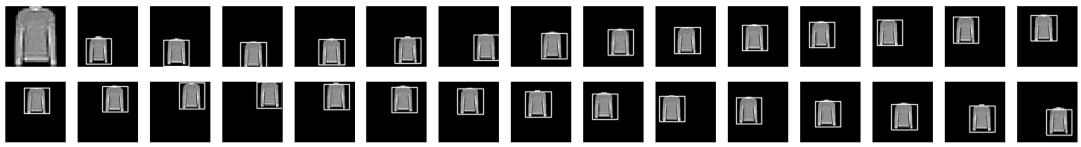

Figure 4.158: A visual representation of C-DATM tracking a 30 frame sequence with the fashion object sweater. C-DATM is trained with a mixture of training data containing both one object and two. The first row is the first 15 images in the sequence, excluding the first frame, and the second row is the last 15. The first frame in the first row is the object the model is attempting to track.

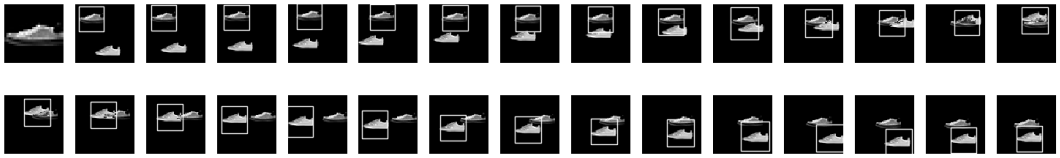

Figure 4.159: A visual representation of C-DATM tracking a 30 frame sequence with two fashion objects, shoes and shoes. C-DATM is trained with a mixture of training data containing both one object and two. The first row is the first 15 images in the sequence, excluding the first frame, and the second row is the last 15. The first frame in the first row is the object the model is attempting to track.

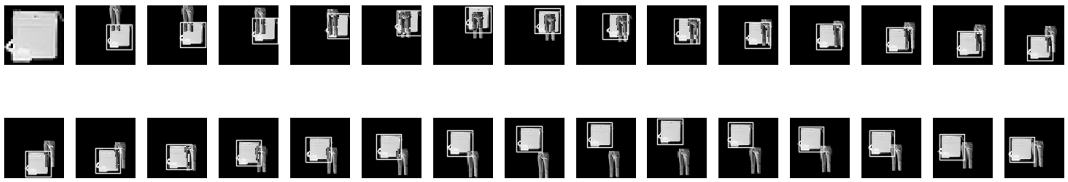

Figure 4.160: A visual representation of C-DATM tracking a 30 frame sequence with two fashion objects, a purse and pants. C-DATM is trained with a mixture of training data containing both one object and two. The first row is the first 15 images in the sequence, excluding the first frame, and the second row is the last 15. The first frame in the first row is the object the model is attempting to track.

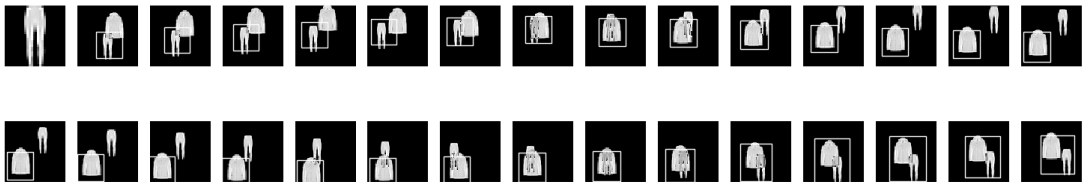

Figure 4.161: A visual representation of C-DATM a 30 frame sequence with two fashion objects, pants and a sweater. C-DATM is trained with a mixture of training data containing both one object and two. The first row is the first 15 images in the sequence, excluding the first frame, and the second row is the last 15. The first frame in the first row is the object the model is attempting to track.

V. Discussion

Overall C-DATM outperforms GOTURN in terms of tracking accuracy for the moving MNIST set. The 91.30% IOU of C-DATM compared to the 62.71% IOU of GOTURN, for tracking a single-digit, is very clear in the visualizations of the bounding boxes. Poor performing tracks by GOTURN include Figures 4.16 and 4.17. Figure 4.16 is a track of the 5 digit, over multiple frames GOTURN loses the five completely. Figure 4.17 is the track of a 6 digit that GOTURN's track is always below the number. Figures 4.10 and 4.10 are consistent tracks, by C-DATM, of a 5 and 6 digit, respectively. The comparison of these figures is the clearest example of the difference in tracking accuracy between C-DATM and GOTURN. Figures 4.14 and 4.18 are tracking of a 1 and a 7 respectively. GOTURN performs relatively well in these figures keeping the track through all frames but not always centered on the object. Figures 4.8 and 4.12 show the results of C-DATM object tracking and we can clearly see the C-DATM improves on GOTURN by keeping the object more centered in the bounding box. In Figures 4.15 and 4.19 we can see that GOTURN tracks well, only losing object center in a few frames. C-DATM outperforms GOTURN (see Figures 4.9 and 4.13) by keeping the object centered in every frame.

The lower IOU of C-DATM(62.23%) and GOTURN(47.70%) when applied to tracking two-digit scenes is not unexpected since tracking in a clutter scene is a more difficult task. Figures 4.29 and 4.31 sho an example where GOTURN tracks the wrong object. Figures 4.22 and 4.24 shows an example where C-DATM also track the wrong object. The only difference between these examples is C-DATM maintains a slightly tighter track when

the correct digit is being tracked. Figures 4.27 and 4.28 also do not always track the correct object using GOTURN. GOTURN improves and tracks the correct object in 4.28 22 of the 29 frames. Figures 4.20 and 4.21 perform slightly better using C-DATM. C-DATM, in Figure 4.21 maintains a correct track, while GOTURN does not. Figure 4.20 tracks correctly over most of the frames, but loses the correct object at the end. In Figures 4.26 and 4.30, GOTURN tracks the desired object through the entire sequence. However, in Figures 4.23 and 4.25 C-DATM keeps a tighter track and does not lose the target.

We further analyzed the data to explore the results of our tests. Table 5.1 and Figure 5.1 contain the average between-class difference of the center of mass in the data. Each combination is considered. As the blur rate increases the average increases. This means that if the networks are using the center of mass to track objects as the blur rate increases it is easier to discriminate between the classes. Table 5.2 and Figure 5.2 contain the difference between the average center of mass in a class, at each blur rate. As the blur rate increases the average differences decrease, especially with a blur rate of 4 or 5. This suggests that as each class is blurred, the different members of each class look more similar. If the network uses the center of mass for tracking and the blur rate is high, then the network does not need crisp features to track well.

Based on the center of mass information, we can empirically state that both models use the center of mass as at least part of their feature set. Even though a blur rate with a radius of 5 looks to have no information, features still exist in the grey scale gradients and exploited by the networks. However, due to the larger impact that blurring has on GOTURN versus C-DATM, it is clear it is more dependent on the center of mass. With blur having such a small impact on C-DATM it suggests that the model is already able to generalize between classes and in classes before the data augmentation. Thus, C-DATM is able to generalize when there is a loss of crisp object features.

Blur Rate	Average Between Class Difference of Center of Mass
0	0.011, 0.008
2	0.012, 0.011
3	0.024, 0.013
4	0.050, 0.020
5	0.075, 0.030

Table 5.1: The average difference between all classes center of mass with varying blur rate. All combinations of 2 way class comparisons are computed, excluding repetition.

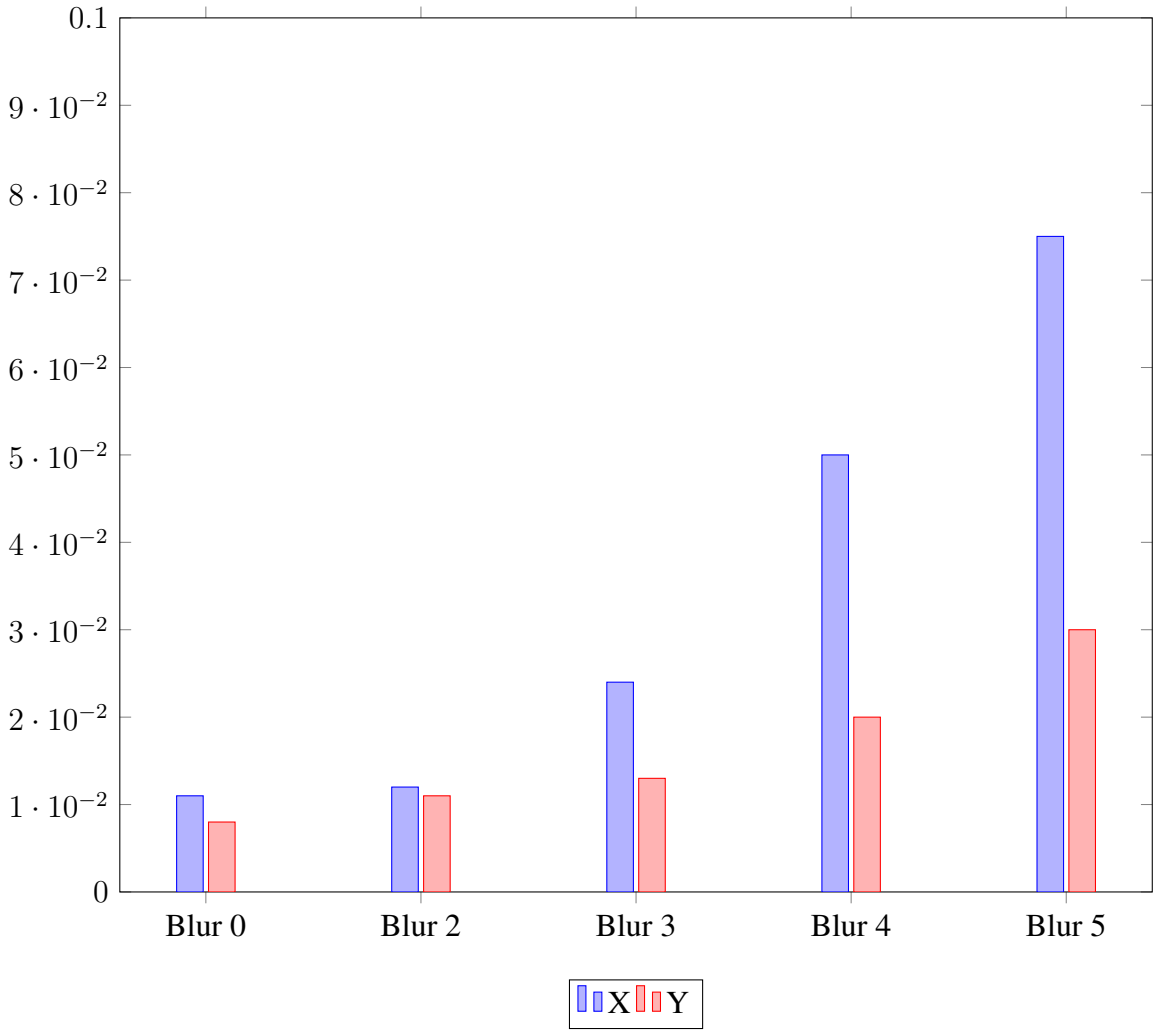

Figure 5.1: Bar graph representation of the average difference between all classes center of mass with varying blur rate. All combinations of 2 way class comparisons are computed, excluding repetition.

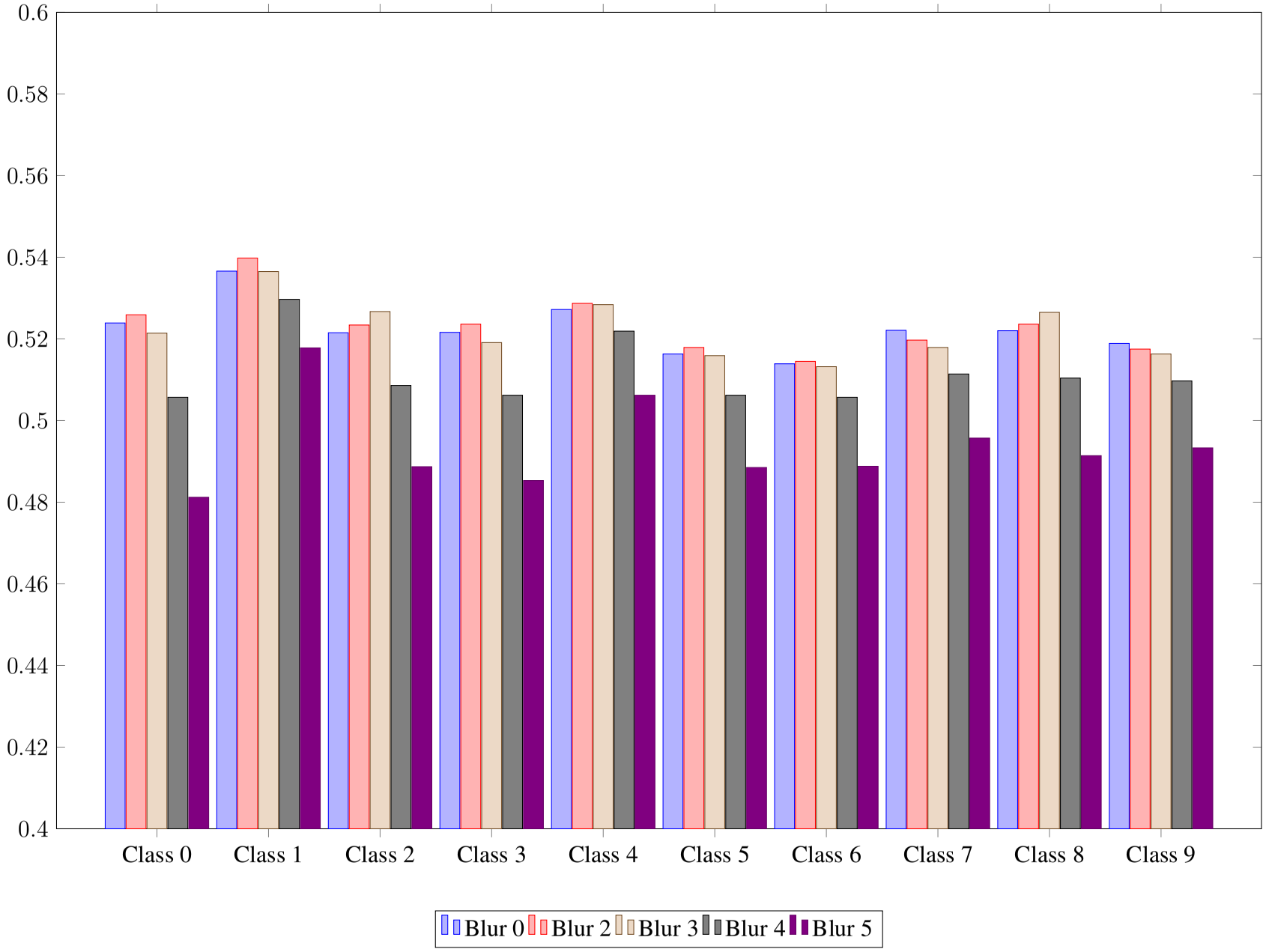

Figure 5.2: Bar graph representation of the average in class center of mass with varying blur rate.

Class	Blur 0	Blur 2	Blur 3	Blur 4	Blur 5
0	.335, .334	.336, .334	.332, .333	.320, .324	.303, .310
1	.334, .351	.337, .351	.334, .351	.323, .353	.307, .353
2	.335, .330	.337, .331	.334, .330	.324, .326	.309, .353
3	.336, .330	.337, .331	.332, .332	.318, .329	.301, .319
4	.337, .337	.338, .339	.337, .339	.331, .336	.320, .326
5	.330, .329	.332, .329	.328, .329	.319, .327	.305, .318
6	.328, .330	.329, .331	.327, .330	.319, .329	.305, .320
7	.333, .335	.329, .335	.326, .336	.319, .334	.307, .326
8	.329, .337	.330, .337	.326, .338	.315, .336	.299, .327
9	.331, .332	.328, .332	.323, .334	.319, .332	.306, .323

Table 5.2: The average in class center of mass difference with varying blur rate.

5.1 Kinematics Test

Comparing Figures 4.32-4.34 to Figures 4.38-4.40 it is clear that C-DATM outperforms GOTURN for tracking single objects, even when the kinematics of the network are inconsistent. This is shown through the tracking IOU in Table 4.1. Creating more inconsistencies, by changing directions twice, have similar results as shown in Table 4.1. Comparing the bounding boxes in Figures 4.44-4.46 to Figures 4.50-4.52 confirms the existence of these inconsistencies caused by changing the kinematics of the object. Specifically, in Figure 4.38 GOTURN struggles to keep track of the digit 6, constantly having the bounding box below the digit. However, in Figure 4.32 the digit 6 is tracked accurately by C-DATM.

Similar patterns exists when examining the results of the kinematics experiments on two-digit tracking sequences. Our attention based method consistently outperforms GOTURN as shown in Figures 4.35-4.37 compared to Figures 4.41-4.43 and Figures 4.47-4.49 compared to Figures 4.53-4.55. However, both models struggle with some of the two-digit tracking seen in Figures 4.36, 4.48, 4.42, and 4.54, just as the models did without data augmentation to experiment with kinematics (Figures 4.22 and 4.29). Overall the performance of C-DATM in the kinematics tests are very similar to the baseline test.

These results show one aspect of the network very clearly, overall the networks do not rely on kinematic data to track the objects. The memory used in C-DATM, however,

does use some kinematic information as evidenced by a slight decrease in performance. Both networks, C-DATM and GOTURN, increase in performance for at least some of the tests, this can be attributed to less data being available in the training sequences making it an easier task than the original. The sequences become easier to track due to having the repetitions in the sequences. The two-turn kinematic experiments further suggest these conclusions due to their higher increase in performance and more repetitions.

5.2 Object Speed Test

When skipping frames to simulate faster moving objects both methods have reduced tracking accuracy. GOTURN struggles to keep tracks through the skipped frames. Figures 4.67 and 4.66 show that GOTURN does not keep the object centered in the bounding box. The object is not lost, and C-DATM keeps a tighter track as shown in Figures 4.60 and 4.61. Both models track well with a single skipped frame on the 8-digit as shown in Figures 4.65 and 4.59. Skipping two frames instead of one decreases performance, even when applied to single-digit tracking. The results shown in Figures 4.78 and 4.79 are similar to the results produced for the single frame skip experiment, but the problems maintaining an accurate track are increased in severity. C-DATM struggles to track the 7-digit (Figure 4.73) when skipping multiple frames, but keeps a tight track in Figures 4.72 and 4.71. GOTURN performs well in Figure 4.77 as well. These results suggest that both models are better at tracking objects with loops, even when skipping frames.

The test with 2-digits in the frames lead to similarly decreased performance when skipping frames. The networks perform worse when skipping frames. GOTURN fails to track the correct object (Figures 4.68 and 4.80) where it accurately tracked objects when no frames were skipped. Both models continue to fail to track objects as shown in Figures 4.69, 4.63, 4.81, and 4.75. As well as, Figures 4.70, 4.64, 4.82, and 4.76. Both the models struggle to maintain tight of tracks throughout the speed experiments. The visualizations

of object tracking during the speed tests accentuate the need for the object to be kept in the view of the previous bounding box, the core limitation of attention-based models.

The speed test illustrates that C-DATM is better at adapting to faster moving targets compared to GOTURN. This is due to the attention-based mechanisms used in C-DATM that allow the network to adapt the size of the bounding box. GOTURN's bounding box is always a fixed size (28 x 28 for our experiments) while C-DATM's average bounding box area changes depending on how fast the object moves. The normal test produces an average bounding box area of 781.13, one skip 792.30, and two skips 799.92. The network is not trained with objects moving as fast as they do in the experiments, yet the network learns to expand the attention area to attempt to retain the object in the attention region.

The performance of both methods decreases relative to the baseline when frames are skipped. This is consistent with the design of the methods and is a limitation of these type of networks. However, our method adapts to the faster moving objects better than GOTURN. This is due to our attention-based network's ability to dynamically change the size of the bounding box while GOTURN has a fixed size bounding box which is twice as large as the size of the bounding boxes found in the truth image.

5.3 Blur Test

Blurring the image does not change the results of C-DATM when tracking a single-digit object. Figures 4.88-4.95 show that C-DATM accurately tracks the object centered even with a blur radius of 5. The performance of GOTURN actually improves as the blur rate increases (see Figure 4.104 to Figure 4.110). Even with the increased tracking accuracy C-DATM outperforms GOTURN(see Figure 4.94 compared to Figure 4.110).

When tracking in a 2-digit environment blurring does not decrease the results of C-DATM. Figures 4.96-4.103 shows that C-DATM tracks with and without the blurring. In fact, C-DATM keeps a tight track on the object even with the highest blur rate ap-

plied(Figure 4.102). GOTURN also has similar results, however performance improved with the introduction of blurring. Figure 4.118 has a tighter track than Figure 4.112. This visualization confirms the performance differences shown in Tables 4.5 and 4.6.

The results of this test were somewhat unexpected. The GOTURN model, which in previous experiments show empirical dependence on the features, performs better when the object is blurred. This gives us insight to which features the network finds most important, and those features are not hard edges and parts of the numbers, but the center of mass and distribution of the mass. The center of mass of blurred images are more similar than when thay are not blurred. The improved results observed when using blurred images suggests the GOTURN model is not able to generalize. C-DATM's performance decreases slightly when the image is blurred. This empirically suggests C-DATM is dependent on the features lost when the object is blurred. This also suggests that C-DATM generalizes well without blurring , which pre-conditions the data to improve diversity between class and decreases diversity in the classes.

A smaller increase in the performance of GOTURN's two-digit tests makes it even more likely that the center of mass is highly important. When two-digits are in the bounding box the center of mass is arbitrary, there is no reference point for the calculation. GOTURN does improve tracking when the data is blurred for two-digits as well. This is most likely due to GOTURN's ability to track a single target when the data is blurred when the network chooses the correct target randomly. The 50% average IOU suggests the system is randomly guessing between the two classes.

5.4 Salt and Pepper Noise

The results of the salt and pepper noise show that the network is able to handle small amounts of noise in the environment, with the network tracking almost as well as the baseline up until 50% noise. Both the single and two-digit experiments empirically show the

network is better than a standard correlation matching function due to the tracking ability even when the noise is present.

5.5 Fashion MNIST

C-DATM transfers well to the Fashion MNIST set. With similar performances to the regular moving MNIST data set. Figures 4.146-4.148 show how the model performs when tracking objects from pants to shoes. The two object set is again a more difficult problem, which is reflected in the IOU performance. Figure 4.149 shows the system's behavior when tracking a shoe when there are two similar looking shoes in the scene. The network tracks the correct shoe through most frames but gets confused when the shoes occlude each other. Figure 4.150 shows an example sequence containing a purse and pants. C-DATM holds a tight track on the correct object all the way through the sequence. Figure 4.151 contains pants and a sweater. Through the first and last portions of the track the correct object is tracked, but during the middle portion of the sequence the network loses the pants when occluded by the sweater.

The tracking accuracy on Fashion MNIST shows that C-DATM is extendable to more difficult problem spaces, without the need to change the structure of the model.

5.6 Two-Digit Training

By using some two-digit or two object data when training C-DATM the performance is increased. The IOU rises for both single and double object sets. The larger increase in two-digit performance is logical due to the network now including such data when training. Figures 4.130-4.135 illustrate that C-DATM does suffer a loss in performance with single objects due to the change of training data.

Figure 4.136 shows the model tracks the correct target as it did in Figure 4.25, however

this figure show the ability for the network to change the bounding box size, as the two 1's are close together the bounding box grows so the network can decide which object is the correct object to track. Even with the improved accuracy Figure 4.137 still does not follow the correct object. This suggests the network struggles with rotation. The network continues to track correctly (see Figure 4.138) and improves the tightness of the track, even when objects are occluded. Figure 4.139 shows and example where the system struggles to track the correct digit, however again the network expands the bounding box to try to locate the correct object, which spends most of the sequence overshadowed by a much larger 0 digit. The most noteworthy improvement is Figure 4.141, which is a very bad track for GOTURN and not a great track for the original C-DATM. However, by including two-digit sequences in the training data the correct 9-digit is now tracked through the entire sequence.

5.7 Experiment Variances

To understand the significance of the data presented in all experiments the variance is calculated. All experiments are done on 1,000 sequences of 30 frames. Over all frames and sequences in the test set the variance is calculated for the IOU. The variance found in Tables 5.3 - 5.5 is based on the variance of the percentile of the IOU. The variances in all experiments are extremely low and the support the results suggested by the experiments.

Reversals	1D CDATM	2D CDATM	1D GOTURN	2D GOTURN
1	.056	1.77	.624	1.29
2	.049	1.67	.492	1.16

Table 5.3: This table contains the Intersection Over Union variance in our kinematics experiments.

Frames Skipped	1D CDATM	2D CDATM	1D GOTURN	2D GOTURN
1	1.20	1.74	1.34	1.34
2	1.63	1.58	1.22	1.39

Table 5.4: This table contains the Intersection Over Union variance in our object speed experiments.

Blur Radius	1D CDATM	2D CDATM	1D GOTURN	2D GOTURN
0	.104	1.90	.592	1.28
2	.089	1.80	.421	1.08
3	.076	1.79	.444	1.10
4	.063	1.78	.484	1.13
5	.042	1.75	.519	1.16

Table 5.5: This table contains the Intersection Over Union variance in our object blurring experiments.

5.8 Feature visualizing

Both models start with the same convolution layers for feature extraction. GOTURN uses pre-trained features from Lenet5 [27] which can be seen in Figure 5.3. Our features are from the same structure but trained end-to-end and can be found in Figure 5.4.

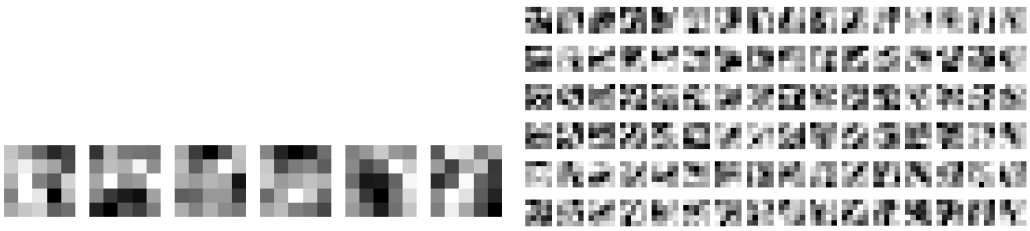

Figure 5.3: This Figure is the pre-trained kernels of the feature extractor used in the GOTURN model.

There are two large differences in the kernels. First, C-DATM kernels seem to have more information, i.e. the features are larger. Second, our features are more distinct (larger contrast) than GOTURN. These differences are due to C-DATM having the dilated convolutions to further extract information. The more activation the kernels provide (in these figures more black portions) the more information available to the dilated convolutions

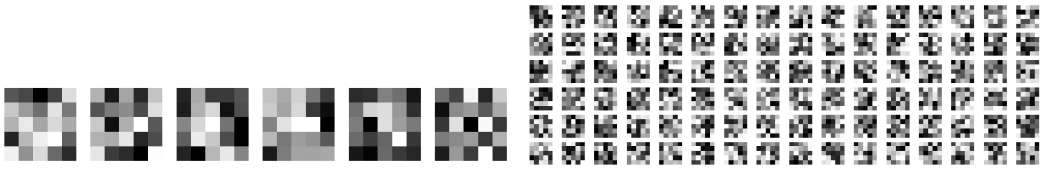

Figure 5.4: This Figure is the end-to-end trained kernels of the feature extractor used in C-DATM.

have to create robust features. Figure 5.5 illustrates how the features in C-DATM's first two convolution layers mimic portions of MNIST digits.

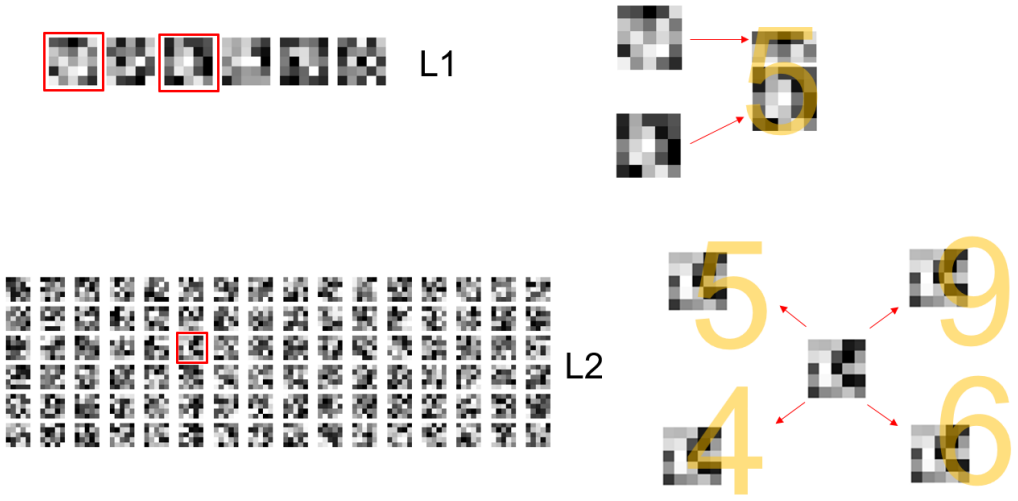

Figure 5.5: This Figure is the end-to-end trained kernels of the first two convolution layers (L1 and L2) in our tracker. Showing kernels that contain parts of 6, 5, 9, and 4 MNIST digits.

Figures 5.6 and 5.7 show that C-DATM is learning features to complete the tracking task. The features found on the left side of each figure show random initialization and on the right the trained kernels represent features learned by the network. Figure 5.11 shows close up important features for the task.

The most interesting features come from the first, second, and last dilated layers. The first and second dilated layer features are found in Figure 5.9. These kernels contain edge features, repeated in a direction. Some repeat curves to the right or above, others repeat

lines. These kernels are dilated over time, so with a small dilation, a small amount of time has passed. These repetitions are patterns of edges moving through space. Figure 5.8 contains examples from C-DATM's trained second dilation layer where digits are superimposed over features to illustrate the network learning how the digits move over time.

The last dilated layer has the largest amount of information, like deep layers of convolutions with pooling applied. That is why similar features are found. For example, a face tracker will create kernels that contain general face images. With C-DATM generalized versions of numeric digits appear in the features. These kernels are found in Figure 5.10. Figure 5.11 zooms in on a few of the kernels mentioned. The major difference between these kernels and the layers in a standard deep convolution network is that the generalization happens over time as well as the data. The features will combine all the data found in a class to a kernel, as well as all the data found for that class throughout the time sequences. This adds robustness to the tracker.

5.9 Conclusions

Dilated convolutions have characteristics, convolutional features, large receptive fields, and low free parameters, that are sought after for improving attention based models. Stacking inputs over time and using exponentially growing receptive fields to accurately learn changes in features with few free parameters provide fast and accurate attention models. Our experiments show that this is still true in two dimensional spatial data. The feature learning done in the single MNIST digit tracking experiments is robust enough to achieve SOTA attention based tracking on two MNIST digits without ever being trained using data containing more than one MNIST digit.

Training C-DATM conditionally allows the network to learn how to track the change

of features over time, rather than learning which features best describe the object over time. This allows for the single model to generically track across a wide range of targets. The network also processes training data at 1,415 frames per second and test data at 5,000 frames per second, exceeding the real-time efficiency requirement.

Based on the center of mass information hidden in the data we can empirically state that both models use the center of mass as at least part of their feature set. Even though a blur rate with a radius of 5 looks to have no information, features still exist in the grey scale gradients and are used by the networks.

However, due to the larger impact that blurring has on GOTURN versus C-DATM, it is clear it is more dependent on the center of mass. With blur having such a small impact on C-DATM it suggests that it does not need the problem space to become more separated to perform well. Thus, C-DATM is able to generalize across the data.

Through multiple experiments, we compared our most recent work to GOTURN, a SOTA tracking model for VOT 2014. C-DATM always out-performs GOTURN, and is less sensitive to conditions in the data; i.e. blurred object, speed changes, and random trajectory changes. We empirically demonstrate that C-DATM generalizes well, that neither model is dependent on kinematics, and that an attention-based methodology lends itself better to non-fluid targets.

C-DATM trains at 212 seconds an epoch, while GOTURN trains at 91 seconds an epoch. However, C-DATM tests at 6 seconds while GOTURN tests at 22 seconds (1,000 sequences). The slightly slower training speeds of C-DATM are minimal to the time saved during testing. This time savings is due to the ability for our network to generalize over different data sets. All tests were done using PyTorch [34].

Figure 5.6: This Figure is the end-to-end trained kernels of the second dilated convolution layer(right) in our tracker compared to it's non-trained state(left).

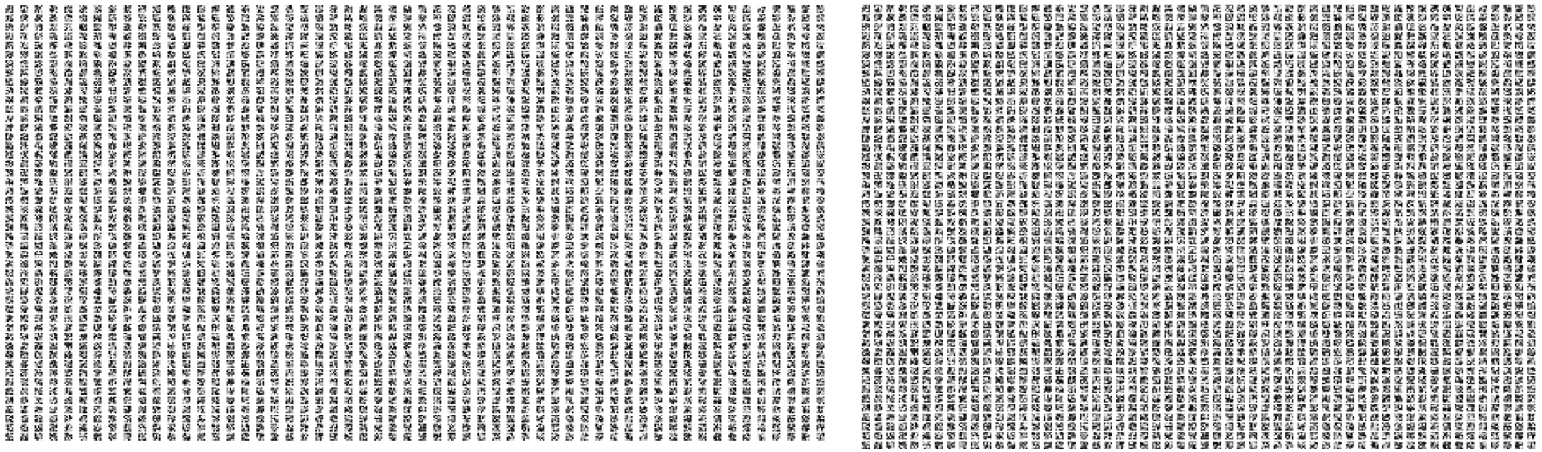

Figure 5.7: This Figure is the end-to-end trained kernels of the fifth dilated convolution layer(right) in our tracker compared to it's non-trained state(left).

Figure 5.8: This Figure is the end-to-end trained kernels of the second dilated convolution layer in our tracker. Showing kernels representing moving edge features, with examples of 6, 9, 8, 3, 7, and 5 moving across the features.

Figure 5.9: This Figure is the end-to-end trained kernels of the first and second dilated convolution layers in our tracker.

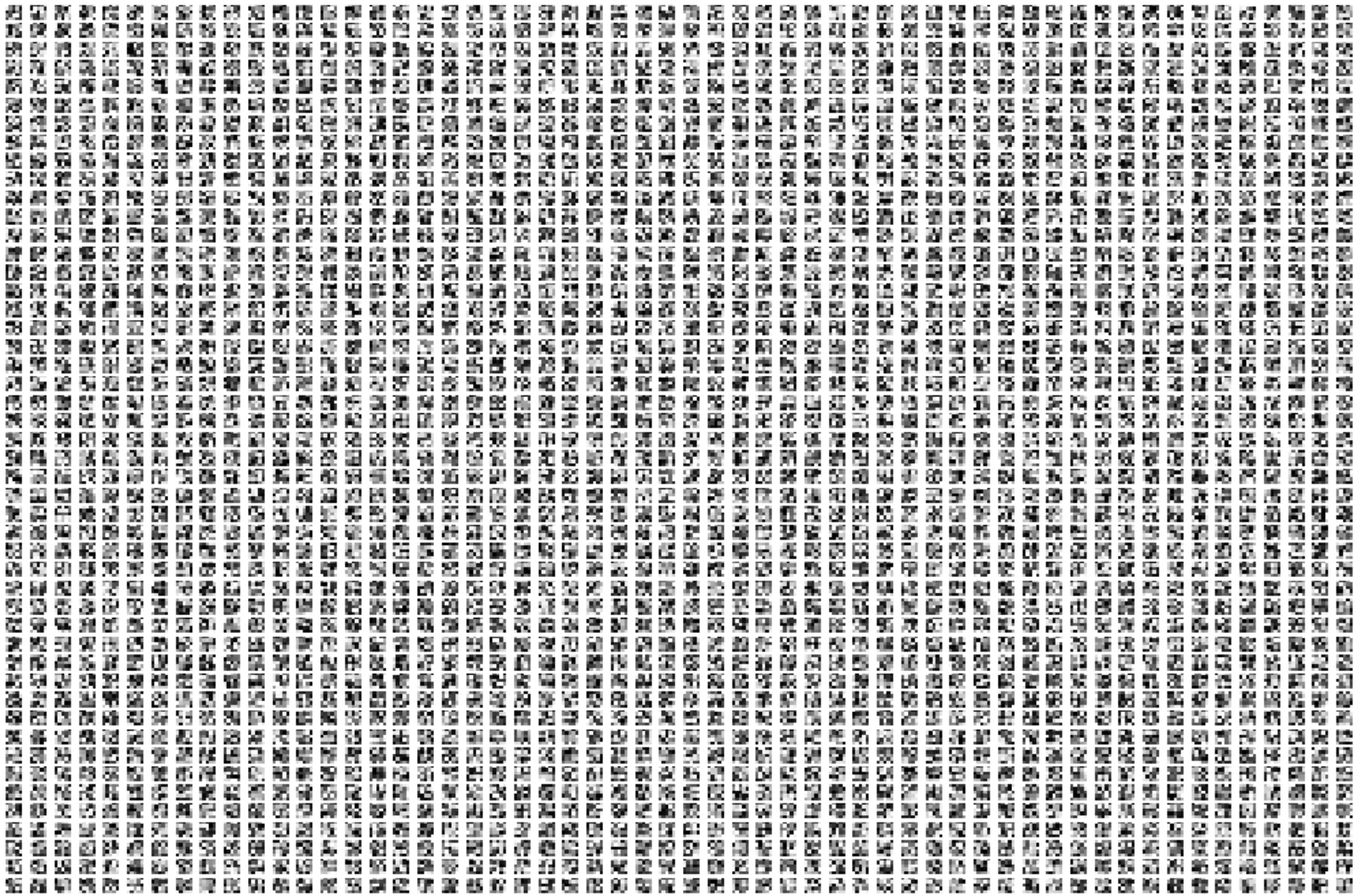

Figure 5.10: This Figure is the end-to-end trained kernels of the last dilated convolution layer in our tracker.

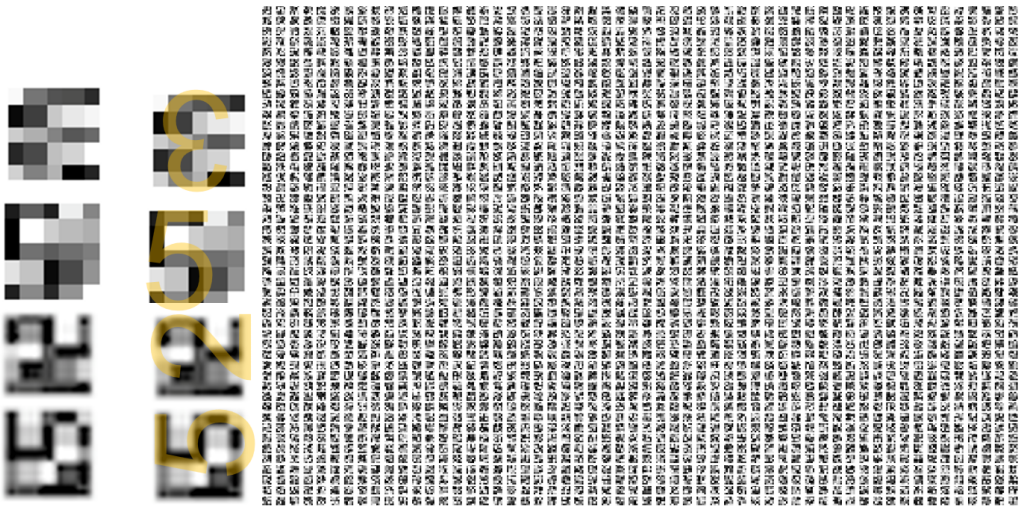

Figure 5.11: This Figure is the end-to-end trained kernels of the last dilated convolution layer in our tracker. Showing kernels for 3, 5, and 2 MNIST digits.

Bibliography

[1] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[3] Loris Bazzani, Hugo Larochelle, Vittorio Murino, Jo-anne Ting, and Nando D Freitas. Learning attentional policies for tracking and recognition in video with deep networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 937–944, 2011.

[4] Juan C Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2488–2496, 2015.

[5] Luka Čehovin, Aleš Leonardis, and Matej Kristan. Visual object tracking performance measures revisited. *IEEE Transactions on Image Processing*, 25(3):1261–1274, 2016.

[6] Franois Chollet. keras. https://github.com/fchollet/keras, 2015.

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[8] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4310–4318, 2015.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[10] Jonathan T Erichsen and J Margaret Woodhouse. Human and animal vision. *Machine Vision Handbook*, pages 89–115, 2012.

[11] Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):267–282, 2008.

[12] Abel Gonzalez-Garcia, Alexander Vezhnevets, and Vittorio Ferrari. An active search strategy for efficient object class detection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3022–3031. IEEE, 2015.

[13] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1462–1471, 2015.

[14] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[15] Tyler Highlander. Very efficient training of convolutional neural networks using fast fourier transform and overlap-and-add. *Brisish Machine Vision Conference*, 2016.

[16] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, et al. An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*, 2015.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[18] Samira Ebrahimi Kahou, Vincent Michalski, and Roland Memisevic. RATM: recurrent attentive tracking model. *CoRR*, abs/1510.08660, 2015.

[19] Ahmed T Kamal, Jawadul H Bappy, Jay A Farrell, and Amit K Roy-Chowdhury. Distributed multi-target tracking and data association in vision networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1397–1410, 2016.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

[22] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2015 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1–23, 2015.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[24] Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[25] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2169–2178. IEEE, 2006.

[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[27] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[28] X Rong Li and Vesselin P Jilkov. Survey of maneuvering target tracking: dynamic models. In *AeroSense 2000*, pages 212–235. International Society for Optics and Photonics, 2000.

[29] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[30] Tiago Marques, Luka Lukic, and José Gaspar. Observation functions in an information theoretic approach for scheduling pan-tilt-zoom cameras in multi-target tracking applications. In *Robot 2015: Second Iberian Robotics Conference*, pages 503–515. Springer, 2016.

[31] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014.

[32] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[33] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. *arXiv preprint arXiv:1510.07945*, 2015.

[34] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Pytorch, 2017.

[35] Marc'Aurelio Ranzato. On learning where to look. *arXiv preprint arXiv:1405.5488*, 2014.

[36] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. 2018.

[37] Ronald A Rensink. The dynamic representation of scenes. *Visual cognition*, 7(1-3):17–42, 2000.

[38] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 3, pages 32–36. IEEE, 2004.

[39] Pierre Sermanet, Andrea Frome, and Esteban Real. Attention for fine-grained categorization. *arXiv preprint arXiv:1412.7054*, 2014.

[40] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[41] Yichuan Tang, Nitish Srivastava, and Ruslan R Salakhutdinov. Learning generative models with visual attention. In *Advances in Neural Information Processing Systems*, pages 1808–1816, 2014.

[42] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[43] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[44] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2048–2057, 2015.

[45] Takayoshi Yamashita, Juliana Yamashita, Hinonori Furukawa, and Yuji Yamauchi. Multiple skip connections and dilated convolutions for semantic segmentation. 2017.