NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

MULTI-FRAME CONVOLUTIONAL NEURAL NETWORKS FOR OBJECT DETECTION IN TEMPORAL DATA

by

Justin Downs

March 2017

Thesis Advisor:	Mathias Kölsch
Second Reader:	Lyn Whitaker

Approved for public release. Distribution is unlimited.

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY *(Leave Blank)*	**2. REPORT DATE** March 2017	**3. REPORT TYPE AND DATES COVERED** Master's Thesis 07-06-2015 to 03-31-2017

4. TITLE AND SUBTITLE MULTI-FRAME CONVOLUTIONAL NEURAL NETWORKS FOR OBJECT DETECTION IN TEMPORAL DATA	**5. FUNDING NUMBERS**
6. AUTHOR(S) Justin Downs	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School, Monterey, CA 93943	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES

The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(maximum 200 words)*

Given the problem of detecting objects in video, existing neural-network solutions rely on a post-processing step to combine information across frames and strengthen conclusions. This technique has been successful for videos with simple, dominant objects but it cannot detect objects if a single frame does not contain enough information to distinguish the object from its background. This problem is especially relevant in the maritime environment, where a whitecap and a human survivor may look identical except for their movement through the scene. In order to evaluate a neural network's ability to combine information across multiple frames of information, we developed two versions of a convolutional neural network: one version was given multiple frames as input while the other version was only provided a single frame. We measured the performance of both versions on the benchmark 3DPeS Dataset and observed a significant increase in both recall and precision when the network was given 10 frames instead of just one. We also developed our own "noisy" dataset consisting of small birds flying across the Monterey Bay. This dataset contained many instances where, in a single frame, the objects to be detected were indistinguishable from the surrounding waves and debris. For this dataset, multiple frames were essential for reliable detections. We also observed a greater improvement in the false negative rate compared to the false positive rate in this "noisier" dataset, suggesting that the additional frames were especially useful for improving the detection of hard-to-detect objects.

14. SUBJECT TERMS Convolutional neural networks, machine learning, object detection, computer vision	**15. NUMBER OF PAGES** 73
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

THIS PAGE INTENTIONALLY LEFT BLANK

MULTI-FRAME CONVOLUTIONAL NEURAL NETWORKS FOR OBJECT DETECTION IN TEMPORAL DATA

Justin Downs
Lieutenant, United States Navy
B.S., United States Naval Academy, 2009

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2017

Approved by: Mathias Kölsch
Thesis Advisor

Lyn Whitaker
Second Reader

Peter Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Given the problem of detecting objects in video, existing neural-network solutions rely on a post-processing step to combine information across frames and strengthen conclusions. This technique has been successful for videos with simple, dominant objects but it cannot detect objects if a single frame does not contain enough information to distinguish the object from its background. This problem is especially relevant in the maritime environment, where a whitecap and a human survivor may look identical except for their movement through the scene. In order to evaluate a neural network's ability to combine information across multiple frames of information, we developed two versions of a convolutional neural network: one version was given multiple frames as input while the other version was only provided a single frame. We measured the performance of both versions on the benchmark 3DPeS Dataset and observed a significant increase in both recall and precision when the network was given 10 frames instead of just one. We also developed our own "noisy" dataset consisting of small birds flying across the Monterey Bay. This dataset contained many instances where, in a single frame, the objects to be detected were indistinguishable from the surrounding waves and debris. For this dataset, multiple frames were essential for reliable detections. We also observed a greater improvement in the false negative rate compared to the false positive rate in this "noisier" dataset, suggesting that the additional frames were especially useful for improving the detection of hard-to-detect objects.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

3DPeS	3D People Surveillance
ILSVRC	Imagenet Large Scale Visual Recognition Challenge
ROC	Receiver Operating Characteristic
SAR	Search and Rescue
VATIC	Video Annotation Tool from Irvine, California

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

Nobody believed in this project like Dr. Mathias Kölsch, and nobody could have provided the guidance it needed as well as he did. Thank you for all your time, which was always extremely precious.

Dr. Lyn Whitaker, thank you for saying yes after so many nos.

Tom Batcha never knew how to be unhelpful. Even after 2 hours spent compiling dependencies in userspace he never gave up, and I'm pretty sure he missed lunch doing it.

All of this would have been impossible without the generous support provided by Dr. Kozdon and Dr. Wilcox and their wonderful `beards` collective.

And Marie gave up more than she planned for, but with a patience and understanding that I would not have survived without.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Detecting interesting and suspicious objects in a maritime environment is both extremely important to the United States Navy and very difficult to accomplish well without human involvement. Relying on human experts to detect maritime objects, however, is expensive and often unfeasible. The problem of finding a human survivor in the water is a perfect example where relying on humans for object detection becomes problematic: the rotary and fixed wing platforms employed in typical search and rescue (SAR) have high operating costs and put additional human life at risk. Humans are also notoriously susceptible to fatigue and must limit on-station time to account for human limitations [1].

While computer vision algorithms have historically been inferior to human vision capabilities for object detection, the past several years have seen huge leaps in computer vision performance [2]. Many computer vision algorithms have even surpassed human capabilities for some tasks [3], [4]. If this trend continues (and there is no sign of it slowing) it is reasonable to expect computer vision to become a viable replacement (or a crucial tool) for humans in many jobs requiring visual analysis.

The inherent advantages of replacing humans with computers for detection tasks are huge. The high-risk and high-cost human-piloted platforms can be replaced by a larger swarm of low-cost autonomous drones. The on-station time will no longer be dictated by human factors, but instead by the platforms' capabilities. A computer's performance can be much more predictable than a human crew's and will not depend on training differences and personality.

1.1 Problem Statement

Imagine the task of detecting only moving cars but ignoring stationary cars. An object detector could probably do very well by looking for clues in a single frame of video: cars in parking spots are usually stationary, moving cars may have a motion blur, and if it had an infrared sensor it could even detect hot engines. But such approaches can only be so helpful: some cars stop in the middle of the road, some are moving slow enough that there is no blur, and cars that have parked recently have warm engines. While algorithms can be designed to overcome these obstacles they will not generalize to other tasks, such as detecting pedestrians or butterflies.

The usefulness of multiple frames does not stop at detecting moving objects, either. Videos are often collected at lower quality than still images, and are often plagued by sensor noise and compression artifacts. Differentiating between a dog and a fox may involve collecting different clues from different video frames. (Perhaps the ears are more visible in one frame and the tail is more visible in a different one.) Multiple frames, then, can conceivably improve classification performance.

Detecting objects at sea requires overcoming all of these problems. Instead of distinguishing between a dog and fox, a common maritime detection task is to differentiate between a human survivor and a whitecap. This task combines both problems from before: A human survivor and a whitecap look identical, except that a survivor “remains the same while the whitecaps blink on and off” [5]. Furthermore, typical surveillance systems have wide angles of view and potential survivors occupy only a small fraction of the entire image (sometimes only a few pixels). The noise and artifacts present in the image make detecting any distinguishing details often impossible with just a single frame. It is clear that if a computer vision algorithm is to satisfactorily detect a human survivor, it must fuse information from multiple video frames.

Even as researchers develop better algorithms for classification and detection of simple objects, there has not been very much research into the problem of detecting objects in the maritime environment, a problem with unique challenges that must be addressed directly. Most current computer vision benchmarks focus on images with “dominant objects,” or objects that occupy a significant fraction of the image [2], [6]. (A typical image is shown in Firgure 1.1.) Solving the problem of object detection at sea must include developing a dataset that better models the problem.

1.2 Research Questions

Our contribution to the field of object detection is to advance object detection at sea by developing a benchmark that simulates the task of detecting a survivor at sea. This

Figure 1.1. Typical Imagenet sample. Source: [2].

benchmark will encapsulate the problems of Since actual SAR sensor footage has not been approved for public release,

We introduce a method of combining multiple frames of information to improve the performance of a neural network for the task of maritime object detection. Specifically, we attempt to answer:

1. Can providing information from multiple frames of video improve the detection performance of a convolutional neural network compared to an identical network given only a single frame of information? How much of an improvement does adding temporal information provide?
2. Can a neural network network be designed to reliably detect objects in a maritime environment?

1.3 Thesis Organization

Chapter 2 describes the current landscape of computer vision, with particular attention paid to object detection and neural networks. It explains historical and contemporary approaches to relevant problems and explores the advantages each approach introduced, as well as the pitfalls.

Chapter 3 describes our experimental setup. The frameworks, architectures, and parameters used during our experiments are explained in depth.

Chapter 4 lists the results of our exploration in adding multiple frames to convolutional neural networks. We describe the improvements made by adding the multiple frames during training for both an existing benchmark and our new maritime dataset. We pay particular attention to the ways in which the networks fail and how the network is able to leverage the additional frames of information.

Chapter 5 discusses the possible implications of our results and suggests directions for future research, with an emphasis on the importance and uniqueness of maritime object detection. We also provide some insight and suggestions for driving interest in the problem.

CHAPTER 2: Background

The problem of object detection is central to computer vision and is the foundation upon which other computer vision problems rely. Before a system can classify an object, it must first know that an object exists at all. Detecting an object in a single image is often easy; photographs are usually taken so that a single "dominant object" is present and obvious. Many of the techniques described in this chapter are designed to operate on a single image. Video sequences, however, provide additional information that can improve detections. How an algorithm processes this additional temporal information can be categorized as either *detect-first* or *track-first*.

A detect-first approach tries to detect objects in each frame individually and then combine the detections to strengthen the predictions. A spurious false-positive in one frame can be thrown out if adjacent frames do not find an object nearby. The advantage of detect-first algorithms is that they can utilize algorithms designed for single images which are often more mature and reliable than algorithms designed for video.

A track-first approach analyzes the differences between adjacent frames and extracts temporal information first (such as optical flow), then makes detections based on that information. A dark shape may not look like an object in a single frame, but if over several frames it moves in a different direction than the rest of the scene it can be separated from the background and classified as an object. While more work has to go in to developing algorithms designed specially for video sequences, track-first approaches can better utilize the temporal information present in video.

2.1 Traditional Object Detection

Background Subtraction [7] is a very simple example of a track-first algorithm. As its name implies, a "background image" is subtracted from an image under investigation in order to find a difference. If the background image is sufficiently representative of the scene being analyzed, the difference between the two images will be any objects not part of the

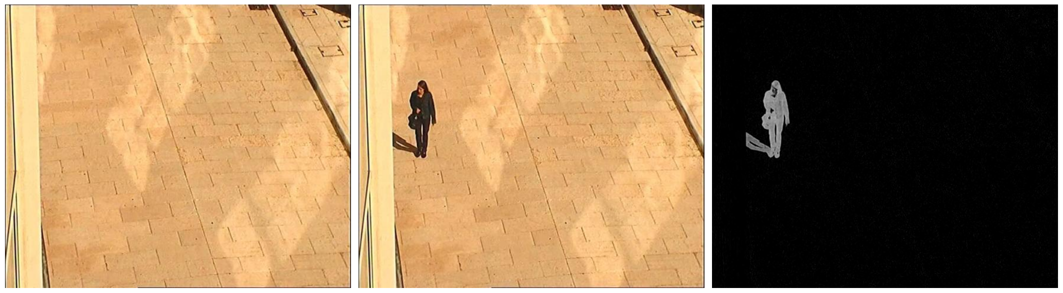


Figure 2.1. Background Subtraction. Adapted from [8].

The left image is the background image, while the center image is under investigation. Subtracting the pixel values of each image results in the image on the right.

background, or put another way, the foreground. Figure 2.1 illustrates a simple version of this concept.

While this technique has the advantage of being simple, it requires that a background model be defined. The most basic solution is to provide the algorithm with an image of the scene with no foreground objects. But this solution is rarely effective: a slight movement of the camera will misalign the background model and result in the wrong pixels being subtracted from each other (see Figure 2.2). This solution does not account for lighting changes or object movement: if a person moves a book a few inches and leaves, the book will be highlighted until a new background model is chosen. Furthermore, this technique assumes that an image with no foreground objects is available. This might not be the case for a busy scene.

Most implementations are more sophisticated and continuously update a background model by keeping a recent history of images. While there are many ways of creating this model [9], they suffer from the limitation of only "seeing" moving objects. After a stationary object is introduced to the scene it will initially be detected but slowly fade into the background as long as it remains stationary.

Template matching was another early (and still commonly used) technique to detect objects in a scene. Template matching algorithms seek to develop a model for the object they are designed to detect, and then search for regions in an image that match the template. One simple template-matching technique involves simply developing image kernels that

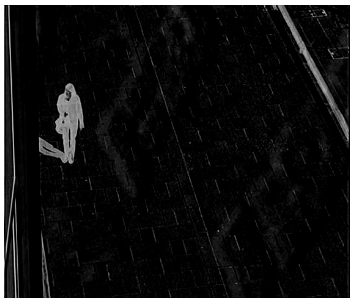


Figure 2.2. Background Subtraction with a 20 pixel shift of the background. Adapted from [8].

If the background image is shifted only slightly, the difference between the two images includes more than just foreground objects and becomes less useful for object detection tasks.

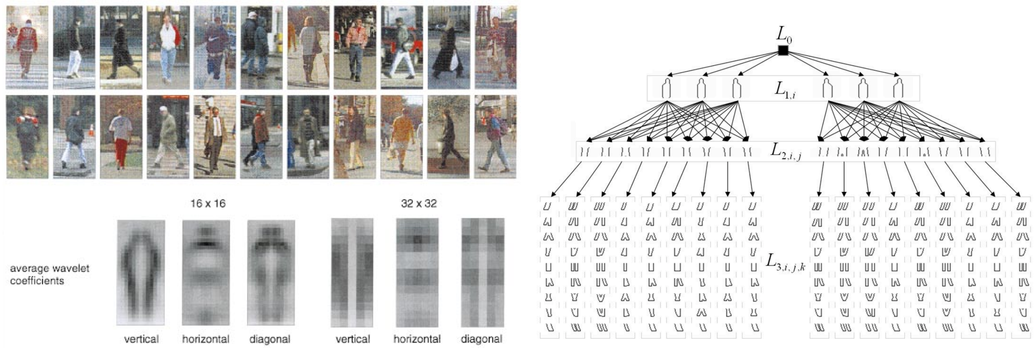


Figure 2.3. Template matching with Haar Wavelets (left) and a Hierarchical Part-Based Approach (right). Source: [10], [11].

look similar to objects of interest. Convolving these kernels with an image results in high responses in regions that look similar to the kernel. This method involves developing kernels for many aspects of each object the detector must operate on. While this method completes both object detection and classification in one step, it is very sensitive to lighting changes, visual obscurations, and unexpected transformations. Furthermore, a template must be developed in advance for every object that the algorithm is designed to detect; a template matching algorithm must either be extremely generic (such as Haar Wavelets [10]) or extremely composable (such as part-based hierarchical template matching [11]) to be useful in practice (see Figure 2.3).

Another template-matching approach is Histogram of Oriented Gradients (HOG) feature matching (with SVM classification) [12]. Instead of relying on convolutional kernels to detect and classify objects, HOG feature matching creates a "histograms of gradients feature vector." Histograms of gradients are intended to be more general than strict pixel comparisons by representing a template not in pixels but by the relationships between object edges. (Scale-invariant feature transform, or SIFT, is another way of generating a feature vector from a subimage [13].) This feature vector achieves rotational, lighting, and size invariance, solving many problems faced by other template-matching approaches. Still, these feature vectors cannot handle occlusions or generalize to recognize shapes that haven't been explicitly introduced to the algorithm. Furthermore, they only operate on a single image; these algorithms by themselves cannot fuse information from multiple frames to improve their performance.

2.2 Object Detection in Neural Networks

Until recently, the best general image classification and detection algorithms relied on various forms of the previous techniques. However, in 2012 a team of researchers from the University of Toronto won the Imagenet Large Scale Visual Recognition Challenge (ILSVRC) by utilizing a concept known as "convolutional neural networks" [14], [2]. Convolutional neural networks had long been enlisted for simple tasks like processing checks and reading envelopes [15], but improvements in datasets and computational resources have allowed researchers to apply them to more complex problems such as general image classification and object localization. Put simply, a neural network is a series of linear classifiers, where the output vector of one classifier is fed into the input of the next. By representing data points as a mathematical vector of features, neural networks optimize a set of "weights" that can be multiplied with the data to derive in a vector of predictions. In computer vision, the feature vector is usually just the image pixel values. While applying a simple linear classifier to pixel values would seem to be a naive approach to image classification, chaining multiple classifiers together allows each "layer" to learn different concepts about images.

One special-purpose layer used commonly in computer vision is known as a "convolutional" layer. Instead of deriving optimal weights to multiply with each data point in the input vector, a convolutional layer derives an image kernel that it convolves with the input vector. By stacking these layers together a convolution neural network effectively implements a template-matching approach to recognize objects in an image, except that it creates hundreds

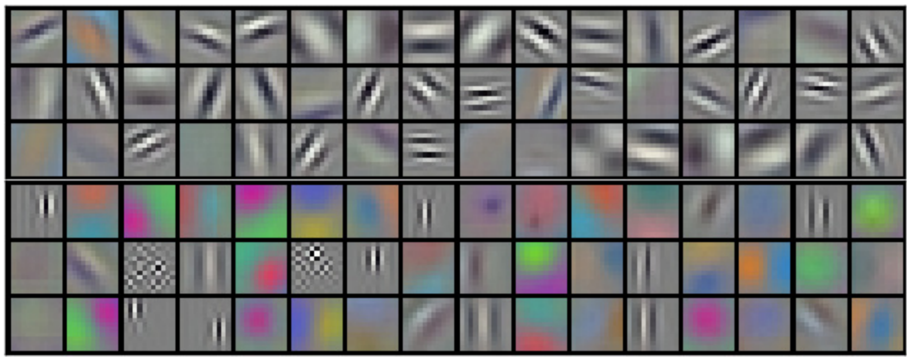


Figure 2.4. Learned weights from a convolutional layer in AlexNet. Source: [14].

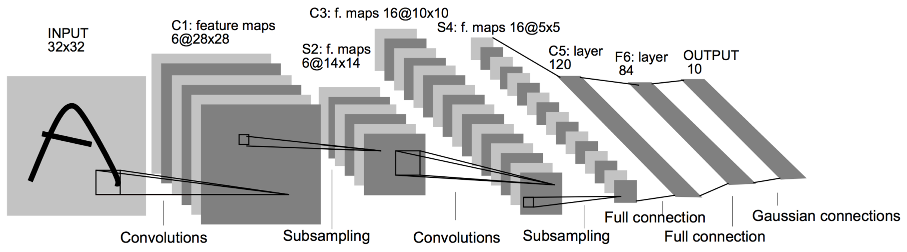


Figure 2.5. The LeNet-5 convolutional neural network was developed to classify handwritten characters. Source: [15].

of general templates and usually stacks multiple convolutional layers together. Early layers usually learn to recognize simple edges and shapes (such as those pictured in Figure 2.4) while later layers synthesize these results to recognize more complex concepts, such as car tires or faces. An early successful convolutional neural network, called LeNet-5, is shown in Figure 2.5.

At first, these convolutional neural networks simply provided image classification. The LeNet-5 network, for example, classified input images as one of ten digits. *Localization* is a different task closely linked to object detection. The localization task requires an algorithm to identify the subset of an image in which an object can be found. Put another way, it requires that the *location* of objects within an image are determined. One of the first successful methods developed for localizing objects with neural networks is known as the "sliding window" technique, used most effectively by the Overfeat network to win the 2013 Large Scale Visual Recognition Challenge [16]. Instead of simply outputting a classification of the dominant object in an image, the network was equipped to output bounding-box

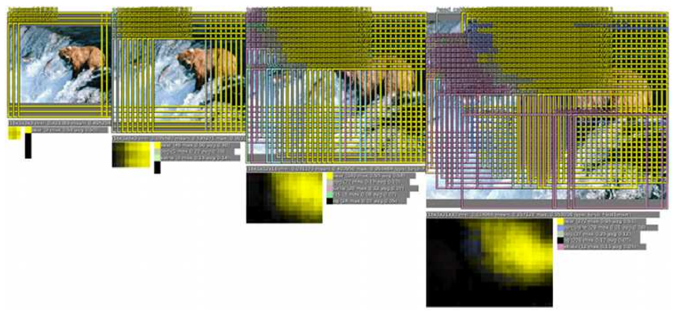


Figure 2.6. Sliding window algorithm developed for the Overfeat network. Source: [16].

In order to localize the bear in the image, the Overfeat network classified thousands of subimages and built a model of what was in the image. A post-processing step combined the classifications and returned a bounding box. This approach to object detection is conceptually simple but time-intensive.

coordinates through regression. Overfeat improved the technique by applying the bounding-box regression and classification to many "windows" in an image and aggregating the results (pictured in Figure .) While this technique was successful it was also inefficient: a single image typically required hundreds of passes through the network.

The idea of "region proposal" was introduced in 2013 as "selective search," [17] an extension of existing bottom-up segmentation techniques. What made selective search novel was that it passed image segments to a classifier "after" splitting an image into discrete sections. (The sliding-window technique, as previously explained, classifies every part of an image and localizes objects by finding the classifications with the highest scores.) The selective search algorithm first segments an image using a graph-based approach described by Felzenszwalb and Huttenlocher [18], and passes the segments to a neural network for classification. This method reduces many of the extraneous classifications made by the sliding-window technique but cannot adapt its segmentation algorithm to best fit different datasets. If its segmentation algorithm performs poorly over a dataset, it cannot be improved with additional training. In other words, machine learning starts and stops with the classification.

In 2015 Microsoft introduced a neural network capable of proposing regions of interest in its "Faster R-CNN" paper [19]. This advance allowed the network to not only more accurately and quickly detect objects in an image (since the network did not require a long pre-precessing stage), it gave the network the capability to "learn how to better detect objects." Instead of relying on a general object-detection or segmentation algorithm to find object-like things, Faster R-CNN learned what "objects" in the training dataset looked like. Furthermore, since the network is trained end-to-end, gradients from the classification section of the network flow up through the region-proposal section. When the network gets a classification wrong, not only does the classifier adjust its weights; the region-proposal network also adjusts its weights so that it can propose better regions (regions that will result in better classifications, that is) in the future. This architecture still beats out most others for object detection [2].

2.3 Object Detection in Video Using Neural Networks

The idea of feeding multiple frames of data to a neural network is not unprecedented: in 2013 a team of researchers developed a convolutional neural network to classify similar human actions such as jogging, running, handwaving and hand-clapping [20]. These activities are usually impossible to classify without multiple frames of data. The team utilized 3-dimensional convolution layers to learn temporal features and applied the algorithm to busy scenes of people moving. While they were successful in classifying human actions that could not easily be classified in a single frame, their solution was strictly a classifier. They relied on a face-detector to first detect all the humans in a video, then inferred their positions. (See Figure 2.7.) After determining the regions of the video that contained humans they passed cropped video sequences to their neural network for classification. Since their 3D convolutional neural network was part of the region proposal process, their algorithm depended on the strength of the face detector; if the face detector failed to detect a human, their 3D convolutional neural network would never have the opportunity to make a classification.

There have been several projects since that extend the idea of temporal features, introducing techniques such as a peripheral-fovea concept, external visual flow information, and Long-Short Term Memory (LSTM) layers to improve performance. [21], [22] These papers describe several methods, illustrated in Figure 2.8, for combining information across frames for video classification. The first method, called Early Fusion, introduces a higher-dimensional convolutional layer to collect and process information. While a normal CNN employs 3-dimensional kernels (2 dimensions to cover the image height and width and a third to cover each color channel), these algorithms extend the kernels into a fourth di-

Figure 2.7. Detecting human actions in video; face-detection pre-processing step. Source: [20].

Shuiwang Ji et al. used 3D convolutions in their neural network to detect human actions through time. They did not incorporate a region proposal network into their solution, though; a face detector extracted humans from the videos and passed those subimages to the 3D convolutional neural network.

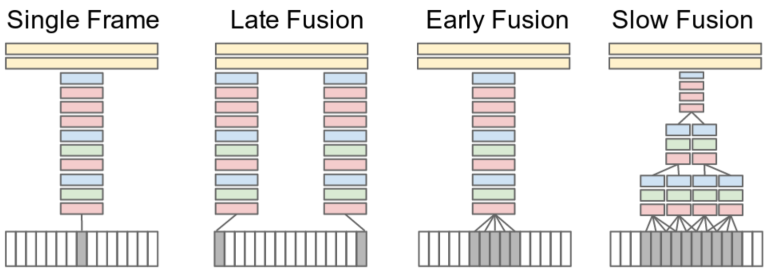


Figure 2.8. Different approaches for combining information from multiple frames in a neural network. Source: [22].

While these methods are sufficiently general to inform any neural network architecture that combines multiple frames, both papers describing these approaches tested them on whole-frame classification.

mension which was covered multiple frames. Late Fusion combines information from two frames spaced several frames apart. Slow Fusion employs several locally connected layers to allow the network to slowly combine frame data. All of these projects tackled the problem of *classification*, not region proposal. While it is tempting to apply their results to all neural network video tasks, classification and region proposal are complementary tasks and the interaction between them is not yet sufficiently understood.

Recurrent networks [23] are a promising alternative to these “fusion” approaches. Recurrent networks receive the same information as a traditional network (in the case of imagery, a vector of pixels usually) but keep track of their state between iterations. A specialized neuron called a Long Short-Term Memory neuron [24] allow recurrent networks to selectively “remember” details that they deem important, and recall those details in later iterations. Common state information carried forward includes words in a sentence (to predict the next word), the last location analyzed (in the case of the cited work), and the location in a previous frame that an object was found. These networks have the advantage that they do not depend on their architecture to determine how far back their memory goes. If it turns out that it is important to remember a small detail seen 50 frames ago, the Long Short-Term Memory neurons are capable of keeping that information available. The “fusion” techniques, on the other hand, require that the number of frames fused together be defined as part of the network’s architecture. As exciting as recurrent networks are, however, they have not yet been employed for region-proposal and classification tasks.

In 2015 and 2016 the ILSVRC included an “object detection from video” competition alongside its traditional object detection challenge. The competition involved localizing “dominant objects” from 30 categories in short video sequences. Most entries followed the same technique: run an object detection algorithm on each frame individually (such as Faster R-CNN [19] or ResNet101 [25]) and aggregate the results across frames. The most successful teams have been those with the most sophisticated post-processing steps. The most recent winning team described a way of aggregating the detections in an algorithm they dubbed “Tubelets” [26]. This technique allowed high confidence in one frame to influence the next (since if an object is present in one frame it will probably be present in the next frame) and relied on an external motion model to propagate predictions through time. Still, this technique only provides its region proposal network a single frame of information. These detect-and-aggregate approaches work well for the ILSVRC Dataset where there is a dominant object that can easily be detected in a single frame. If their region-proposal networks cannot find the object by looking at a single frame (such as the scenario presented in Section 1.1), their post-processing aggregation step will not be able to compare any information across frames.

2.4 Using Multiple Frames in a Region Proposal Network

Instead of following the detect-then-track techniques employed by the latest ILSVRC winners, we designed a region proposal network that accepts multiple video frames as input. By giving the neural network direct access to several frames, it has the opportunity to learn temporal features, such as motion and changes in shape. Access to multiple frames also gives it the information necessary to strengthen its detection confidence during especially noisy moments, as described in Section 1.1. We did not conduct any pre-processing except for data normalization and augmentation, nor did we rely on a post-processing step to fuse predictions. We are introducing a completely end-to-end region proposal network that is capable of utilizing temporal information to make object detections. To ensure that the network is in fact using the multiple frames provided to it, we test its performance against an identical network given only a single frame of information. We then test both networks on a new maritime dataset meant to simulate the problems outlined in Section 1.1.

CHAPTER 3: Methodology

We designed a convolutional neural network to test and measure the performance gains of object detection when using multiple frames from a video versus a single frame. We designed a data layer that provided any number of frames of information at an arbitrary frame rate. By changing only the data provided to the network and keeping the rest of the architecture constant we were able to observe how varying the number of input frames affected the performance of the network.

3.1 Data Augmentation and Pre-processing

In order to prevent the network from overfitting and learning common paths of motion (for example, a commonly used corridor in the 3DPeS dataset) we implemented several data augmentation techniques that are now standard practice in neural network training [27].

Every series of video frames, before being processed by the network for training, was randomly flipped horizontally or vertically, and/or transposed. Every frame in a 10-frame clip was transformed thew same way. This pre-processing step avoided orientation bias by ensuring an equal chance of orienting the image any of the 8 possible ways to flip and/or rotate an image. Images were not transformed during testing.

In both the training and the testing steps, a constant was subtracted from each pixel value before providing the image sequence to the network. This constant was derived from the mean value of all the pixel values in the training set and was different for every corpus. While we derived our mean pixel values by analyzing every image in each training corpus (listed in Table 3.1), we discovered that the pixel means of our datasets had very low variance. The pixel means for every frame in any given camera angle in the 3D People Surveillance (3DPeS) Dataset had a standard deviation of less than 1.8; the frame means in the Monterey Bay Webcam Dataset had a standard deviation of 4.0. This observation allows future implementations to analyze much fewer training frames to derive a representative mean value. If you can assume that the mean of the frames in a "surveillance-style" video corpus (where the videos are captured at a fixed camera angle in consistent lighting) has a standard deviation of 4.0, the mean of a single frame has a 99% chance of being within 10 of the entire corpus's mean.

Dataset	Constant Mean Value
3DPeS [8]	160
Monterey Bay Webcam	125

Table 3.1. Constant mean subtracted from each image in both the training set and the test set, by dataset.

Since our Monterey Bay Webcam Dataset contained many frames with no object present, we added logic to our data layer to ensure that at least 50% of the sequences introduced to the network during training contained an object. While this ensured that whole-image classifications would be be split between positive and negative examples, it did not ensure an even split for each *region* of an image. Section 4.4.1 goes into more detail about the data imbalance for the Monterey Bay Webcam Dataset.

3.2 Network Architecture

In order to measure the effects of additional temporal data, we designed two networks and compared their results. The two networks are identical except for the number of frames they accept. Figure 3.1 provides an illustration of the general acrchitecture, where f is the number of frames provided to the network.

3.2.1 Data Input

Our network was loosely based on the well-known LeNet CNN [15]. Due to memory constraints, we limited batch size to 1 during training. Since such a small batch size can result in inaccurate estimates of the gradient, we reduced the learning rate to $\alpha = 0.0001$ as described in Section 3.3.

The network received image sequences at a dimensionality of $f \times 300 \times 300$, where f is the number of frames in the sequence. Each frame, then, contained 300×300 pixels and a single grayscale channel.

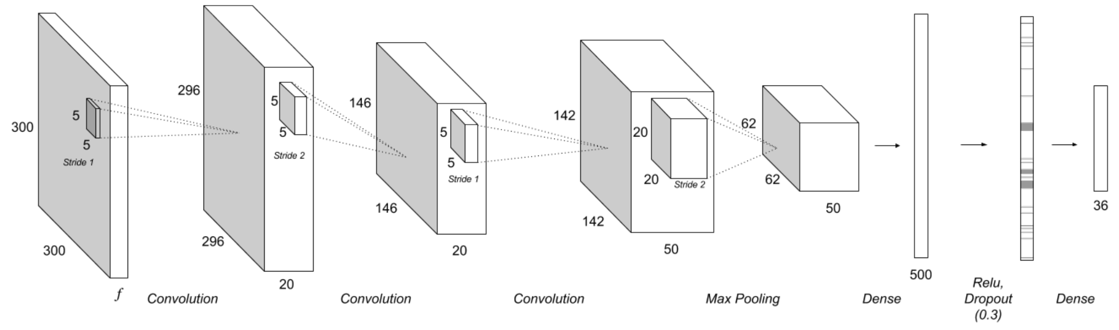


Figure 3.1. Convolutional neural network architecture developed for this project.

We tested the network with different numbers of frames as input. In every case, f is the number of frames provided as input.

3.2.2 Single-frame architecture

To provide a baseline performance level, we trained the network to identify objects given only a single frame of input. This was achieved by setting our custom data layer to provide a single frame of input, depicted in Figure 3.1 where $f = 1$.

3.2.3 Multiple-frame architecture

To test the network's ability to utilize multiple frames of information, we set the data layer to provide 10 frames to the network in chronological order. Empirically, we found the most success when providing the network with 10 frames at a frame rate of 4.8 frames per second. (This is the result of dropping 4 out of every 5 frames in a 24 fps video clip.) The data layer provided the frame data to the network as a $10 \times 300 \times 300$ array, depicted in Figure 3.1 when $f = 10$.

When training with multiple frames, the network was trained to detect objects located in the 6th frame provided to it. This allowed the network to look 5 frames (0.75 seconds) into the past and 3 frames (0.45 seconds) into the future to assist with detection. While this "look-ahead" does introduced a 3-frame "lag" into the system, this network architecture can still operate on a "live" video feed; albeit with a 0.45 second minimum detection delay.

3.2.4 Ground truth labels

The labels were provided at the input layer as an array with a length of N^2. The image was divided into a grid with dimensions $N \times N$; if a bounding box was present in a specific grid-box, the corresponding label array element was set to 1. If no object bounding box was present in a grid-box, the corresponding label array element was set to 0. (See Figure 4.1 for an illustration of how images were divided into grids.) Since bounding boxes did not perfectly segment objects (box corners almost never actually contain the object being annotated), grid-boxes were only labeled as containing a bounding box if a certain percentage of the grid-box overlapped a bounding box. We achieved the best results if we only counted a grid-box as containing a bounding box if at least 3% of the grid-box's total area was overlapped. We found the best balance of network performance and region-proposal resolution when $N = 6$, resulting in a a 36-element label array.

3.2.5 Convolutional Layers

Description

Similar to the LeNet model, we followed the data layer with a 20-filter convolutional layer using 5×5 pixel filters and a stride of 1.

We departed from LeNet on the next layer, replacing the max-pooling layer with another convolutional layer. This layer was also 20 filters deep and employed 5×5 filters, but compressed the data with a stride of 2.

The next convolutional layer was identical to the LeNet model: 5×5 pixel filters and a depth of 50.

The colvolutional layers were followed by a pooling layer with a spacial extent of 20×20 and a stride of 2. While this is an unusually large pooling extent, this setting achieved a better result than smaller kernel sizes.

Motivation

While grayscale images have an input depth of 1 and color images have an input depth of 3, our multi-frame network has an input depth of 10. This means that instead of learning normal visual features like those pictured in Figure 2.4, the first convolutional layer will be learning features with 10 channels, where each channel corresponds to a different slice of time (instead of different colors like a typical image). The features learned in this layer will be temporal features, since the convolutions will be performed over several frames. While a convolutional neural network operating on normal 3-channel single color images is able to distinguish when a green area is to the left of a red area, this 10-channel "temporal" convolutional layer will be able to learn when an shape in one frame moves to the left in the next frame. This first layer might also learn about how an object changes through time, such as a bird flapping its wings.

After this first layer learns temporal features, the following convolutional layers operate on these features in more complex ways. The same way that typical convolutional neural networks combine edge detectors features to detect truck wheels and cat ears, a temporal convolutional neural network can combine relative motion (movement through the scene) with shape transformation (a bird flapping its wings) to distinguish a bird flying through the air and a bird flapping its wings stationary in the water.

3.2.6 Fully Connected Layers

The result of the max-pooling was fed into a 500-element fully-connected layer using Rectified Linear Units (ReLU) [14] as the activation function. DropOut [28] was applied to this layer with a ratio of 0.3 (70% of the activations were kept). A final fully-connected layer was connected, providing a 36-element output. This layer utilized the sigmoid function for its activations.

3.3 Training

Since the output of the network was simply an array in which 0 indicated "no object detected" and 1 indicated "object detected," we treated the region-proposal problem as a multi-label classification problem and used the cross-entropy loss as our cost function.

For optimization we used stochastic gradient descent with momentum $\mu = 0.9$. We found the best base learning rate to be $\alpha = 0.0001$. Since our datasets were relatively small we set weight decay to a fairly high setting of 0.05 to avoid over-fitting and promote generalization. We found that this higher-than-normal setting resulted in the best testing performance. We tested the network after 300,000 iterations of training. This resulted in 7 epochs of training for the Monterey Bay Webcam Dataset and 23 epochs of training for the 3DPeS dataset.

CHAPTER 4: Results and Analysis

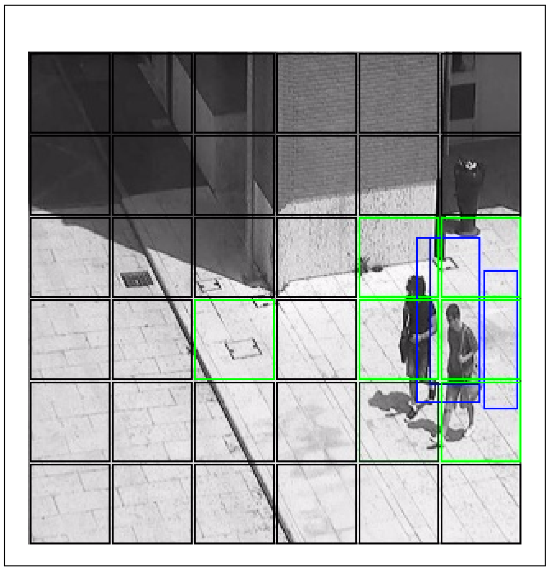
4.1 Implementing the Network

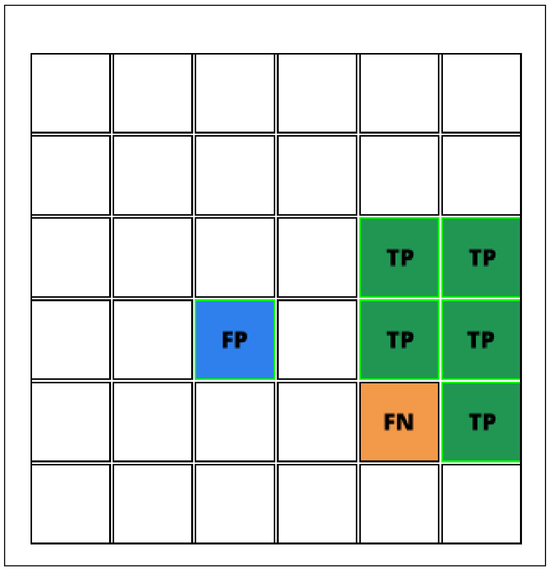
We implemented our network using the Caffe Deep Learning Framework, designed primarily by the Berkeley Vision and Learning Center [29]. Our network followed the architecture described in Section 3.2 and we relied on the built-in layer implementations whenever possible. The one exception to this was the custom data layer we designed in Python to load video sequences into the network. We designed the data layer to combine multiple frames from a video sequence into a single training (or testing) example that the network processed in a single pass. The custom data layer also supported the data augmentation techniques described in Section 3.1. Every other layer of the network was implemented by the Caffe project.

4.2 Measuring Performance

The network's output is a 36-element array in which values close to 0 indicate "no object detected" and values close to 1 indicate "object detected" for the region of the input corresponding to each element of the array. One pass through the network produces 36 of these classifications. These classifications were separated into true positives, false positives, true negatives, or false negatives (see Figure 4.1). We then plotted the performance at different confidence thresholds and measure the area under the receiver operating characteristic (ROC) curve.

We also produced a graphical representation of the network output, shown in Figure 4.1a. In this representation, the blue bounding boxes represent the human-annotated ground truth. Each cell of the grid laid over the image corresponds to a single element of the output vector. The intensity of the green border of each cell indicates the value of its corresponding element in the output array; a value close to 0 is represented by a black border while a value close to 1 is represented by a bright green border. This representation is used throughout this thesis.


(a)



(b)

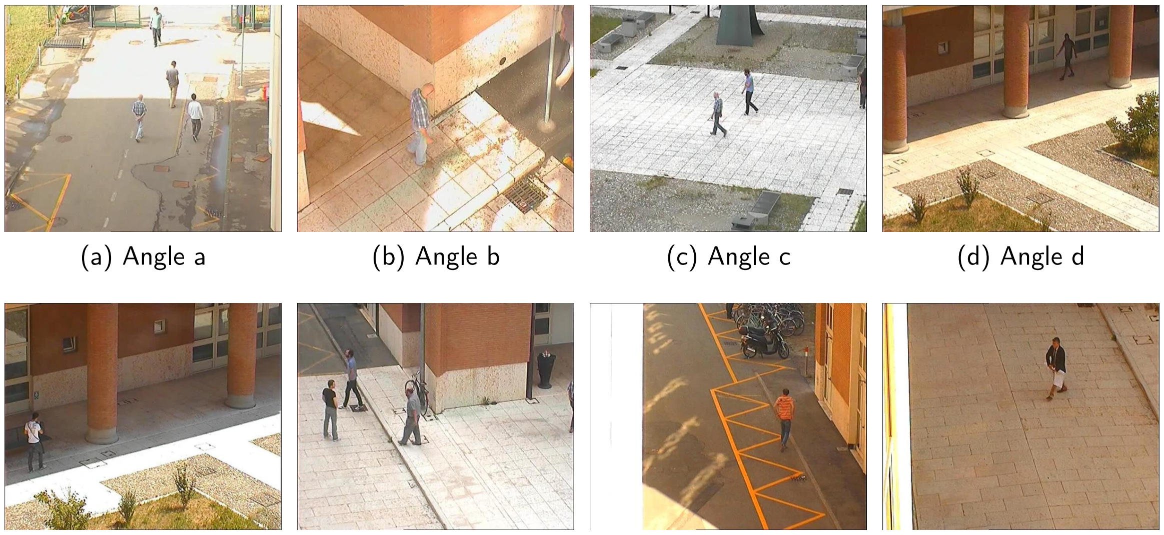
Figure 4.1. Classification metric. Adapted from [8].

A grid box counts as a positive example if it intersects a ground truth bounding box by at leat 3%. Using this metric, the network essentially provides 36 predictions for every image; each prediction counts as either a true positive, false positive, true negative, or false negative.

4.3 Identifying Pedestrians in the 3DPeS Dataset

4.3.1 Dataset Description

The 3DPeS Dataset is a collection of video sequences provided by the ImageLab at the University of Modena and Reggio Emilia [8]. Each sequence is taken by one of eight fixed-position cameras (pictured in Figure 4.2) and depicts various pedestrians walking, sitting, and standing. The sequences include diverse lighting and shadowing conditions and a range of pedestrian activities. Every frame contains at least part of one person; frames with no pedestrians were trimmed from the sequences. Although the dataset is designed to measure person re-identification (the same pedestrians appear in different scenes) and 3D scene reconstruction, it can easily be repurposed as a benchmark for simple object detection. The fixed camera angles and large pedestrians make this particular dataset a rather easy object detection task. We used this dataset to establish a baseline performance measurement for our network, expecting a maritime object detection task to be much harder.


(a) Angle a (b) Angle b (c) Angle c (d) Angle d

(e) Angle e (f) Angle f (g) Angle g (h) Angle h

Figure 4.2. Typical 3DPeS frames. Source: [8].

Since the 3DPeS Dataset was not intended for object detection, it did not include ground truth bounding boxes. We used the Video Annotation Tool from Irvine, California (VATIC) to annotate the 3DPeS Dataset, labeling each pedestrian with a bounding box. Although our annotations include a unique label for each pedestrian, for the purposes of object detection we treated each pedestrian simply as "an object." We labeled a total of 45 sequences that included all 8 camera positions. Each sequence included, on average, 250 frames. Since camera positions "d" and "e" are so similar, we grouped those sequences together, resulting in 7 discrete camera angles for our experiments.

4.3.2 Experimental Setup

We conducted two experiments using our network and the 3DPeS Dataset.

In the first experiment we trained a version of our network given 10 frames of information (described in Section 3.2.3) and a version given only a single frame of information (described in Section 3.2.2). We implemented cross-validation by training the network 45 different times, each time leaving out a single sequence. The network was tested on the sequence that was left out of training.

The in second experiment we trained the same two versions of the network 7 separate times. Each time we left out all the sequences captured by a specific camera angle. We tested the networks on the sequences left out of each training session. While the objects being detected were similar across camera angles (they were are pedestrians), the camera angles were different enough that they could be called different “scenes.” This experiment was designed to measure the networks’ ability to detect objects in “novel” scenes never seen before. Success in this experiment is evidence that the networks are not simply memorizing the background scenes and comparing them with the test data.

4.3.3 3DPeS Experiment 1: Training the Network on All Scenes

Results

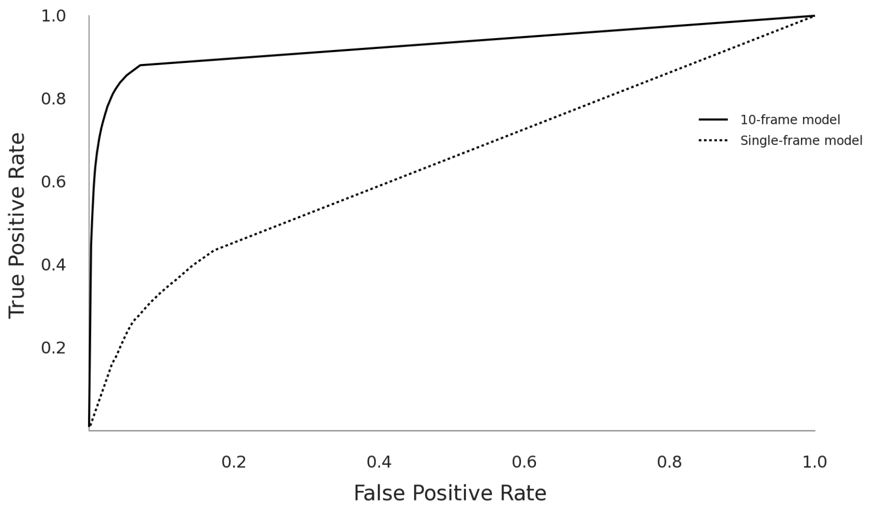


Figure 4.3. ROC curve comparison of a 10-frame model and a single-frame model trained on the 3DPeS Dataset. Cross-validation across all camera angles.

Figure 4.3 and Table 4.1 show the results of the first experiment, where the networks were exposed to every camera angle through cross-validation. The 10-frame model performed much better than a network given only a single frame. The additional frames improved both the false positive rate and the false negative rate for the network. Furthermore, these results indicate that providing 10 frames to the network reduced both error rates by about two-thirds.

Model	Best F1-Score	False Positive Rate	False Negative Rate
Single-frame model	0.34	0.10	0.66
10-frame model	0.82	0.03	0.20

Table 4.1. Performance comparison, familiar scenes in the 3DPeS Dataset. Cross-validation across all camera angles.

3DPeS Studying the Failures

While our 10-frame model worked fairly well identifying pedestrians in the 3DPeS Dataset, it certainly wasn't perfect. That said, we noticed that most of its mistakes could be fit into the following categories:

1. Groups were easier to identify than individual pedestrians
2. Overfitting to scenes
3. Distractors
4. Objects falling between grid-squares

One of the most conspicuous category of errors was the tendency for the network to fail to identify individual pedestrians but localize groups of people. A fairly common scenario would be for several pedestrians to enter the scene independantly and rendevous. Often the network would fail to identify the pedestrians as they entered but as soon as a group formed, it would highlight them. Figure 4.4 shows this happening.

Figure 4.5 and Figure 4.6 show a similar phenomenon, where certain pedestrians were not detected until another pedestrian approached. In this situation, a person or group of people standing still were not identified until a pedestrian walked past them. The network usually had no problem identifying the moving pedestrian. Once the moving pedestrian got close enough to the previously ignored group of people, the network expanded its detections to include the group standing still. Once the moving pedestrian continued past the group, the network stopped detecting the people standing still.

While this seems strange, it also seems fairly easy to explain. The 3DPeS Dataset contained many examples of groups of people. It also contained many objects that could make

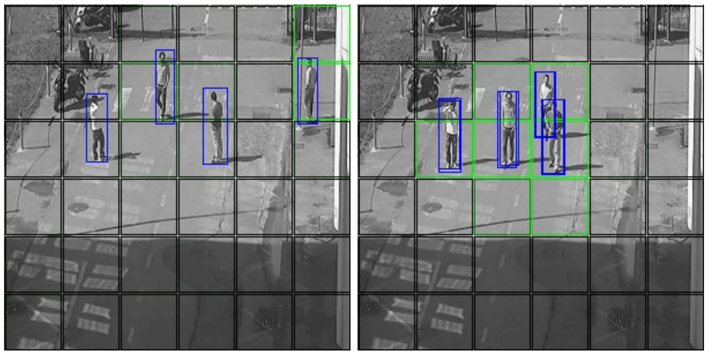


Figure 4.4. Groups of people were more easily identified than individuals. 10-frame model. Adapted from [8].

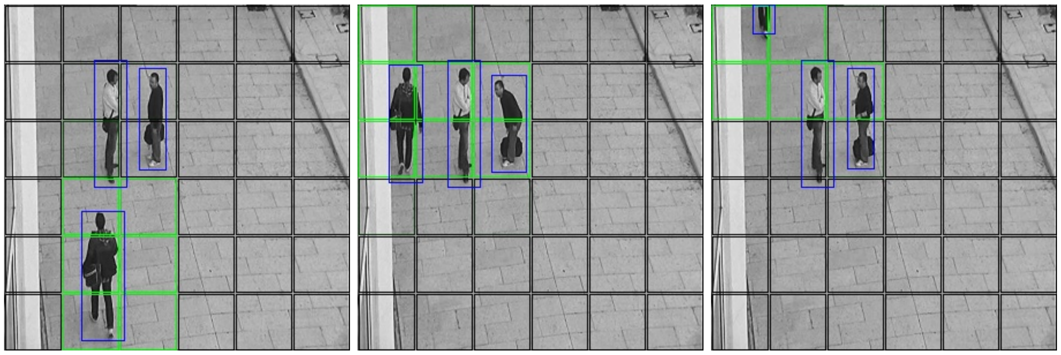


Figure 4.5. Pedestrians were more easily identified when they appeared together. 10-frame model. Adapted from [8].

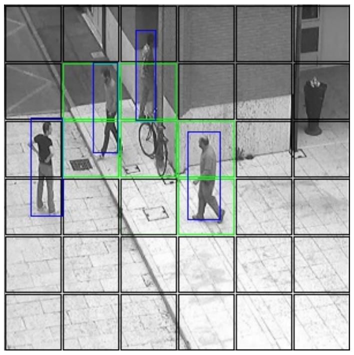


Figure 4.6. Sometimes pedestrians were not recognized when they appeared outside a group. 10-frame model. Adapted from [8].

detecting an individual person difficult: parked bikes, an open truck bed, and parked cars for example. While these things might be mistaken for an individual person, a group of people contain a lot more information to aid in classification. Furthermore, the idea that a

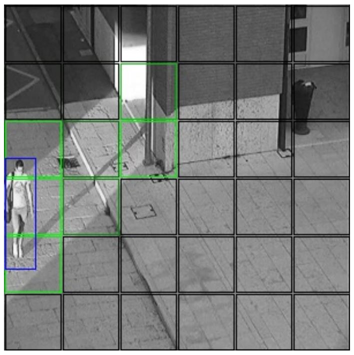


Figure 4.7. False positive example: The network sometimes overfit to a scene. Adapted from [8].

"moving" person is easier to identify than a small group of people standing still should be apparent: the network was given multiple frames to analyze; the ability to detect moving objects is an obvious implication.

What this observation means for neural networks overall is that an end-to-end network trained for object detection will utilize non-local information to make detections. A pedestrian is easier to recognize in a group: that means pixels from elsewhere in the image are being used to classify the region with that person. Even more exciting is the fact that a person standing still is easier to identify when a moving person walks by. In that case the network is pulling information from elsewhere in the image, and from "elsewhere in time." A convolutional neural network given temporal information is able to fuse non-local information both spatially and temporally to make its classifications.

Sometimes relying on non-local information to make classifications worked against the network. Figure 4.7 shows an example of a sequence where the network returned many false positives in a region where people often assemble. It could be the case that a combination of unexpected light effects cause the network to make an inaccurate prediction, but we observed that for several scenes it was much more common for the network to incorrectly detect the existence of a pedestrian in areas where people commonly gather. Of course, biasing toward a prior probability is expected, given that neural networks are statistical models. While we did observe this error in a few of our tests, it did not dominate our results so we decided that additional regularization was not necessary.

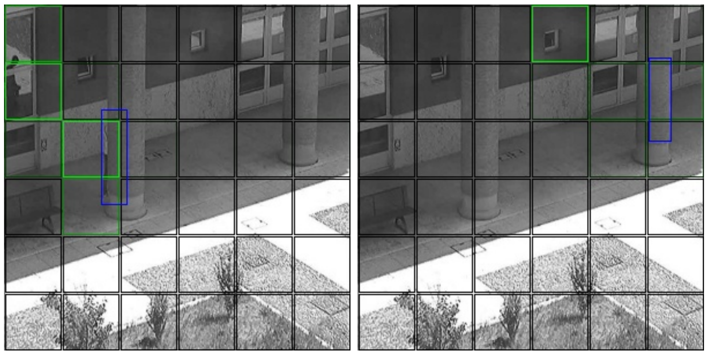


Figure 4.8. False positive examples: The network sometimes confused reflections (left) or suspicious objects (right) with pedestrians. 10-frame model. Adapted from [8].

The 3DPeS Dataset contained some sequences that caused confusion for our network. Figure 4.8 shows two examples. In the first, the reflection of a pedestrian can be seen walking past a window; the network classified the reflection as a pedestrian. While this counted against the network it is important to note that this is actually a good result! This is a case where the human annotator failed to find a person in an image that the computer was able to identify. While the human may have dismissed the reflection as "not a real person," it is likely that in a real-life application, finding a reflection of a person is just as important as finding the actual person. A human searching for a pedestrian might be looking for something specific (scanning the walkway, for example, since that's where a person is likely to be) and miss evidence where it is not expected.

By far the most common reasons for a false positive classification was a shortcoming of the experiment rather than the network's inability to learn. Two factors interacted to cause these errors: First, the network is designed to output an array of classifications corresponding to non-overlapping regions of an image. As objects move between regions there are consecutive frames with minute changes in pixel values that result in different classification arrays. It is understandable that a network would have trouble classifying the frames on either side of that transition. Second, the ground truth bounding boxes do not perfectly model the objects. Even if the human annotations were perfect, the pedestrians being tracked are not rectangles and so there must be regions of the bounding box that contain pixels not belonging to any object of interest. These factors work together to create "regions of uncertainty" near the edges of each grid-square. Since the network must make

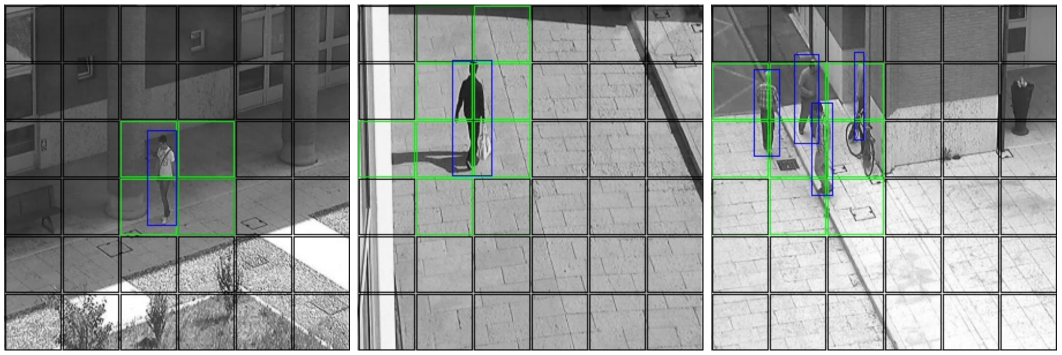


Figure 4.9. False positive examples: When the ground truth bounding boxes fall near the edge of a grid-square, the network often classifies surrounding regions as containing pedestrians. 10-frame model. Adapted from [8].

a classification for every region (assigning a low score is akin to classifying a region as empty) it tended to overshoot its prediction and highlight the regions adjacent to the object (see Figure 4.9).

4.3.4 3DPeS Experiment 2: Testing Novel Scenes

Results

Figure 4.10 and Table 4.2 show the results of the second experiment, where the 10-frame model and the single-frame model were each tested on images captured by a camera angle not included in the training set.

Model	Best F1-Score	False Positive Rate	False Negative Rate
Single-frame model	0.29	0.101	0.701
10-frame model	0.78	0.036	0.218

Table 4.2. Performance comparison, novel scenes in the 3DPeS Dataset. (Combined results across all categories.)

Both models performed worse when tested on novel scenes than when tested on scenes previously introduced, although the 10-frame model still performed reasonably well with an F1-score of 0.78 (Table 4.2). The single-frame model suffered slightly more, achieving an F1-score of only 0.29. This slightly poorer performance of the single-frame model suggests that the network was learning scene-specific information; i.e., a fixed background

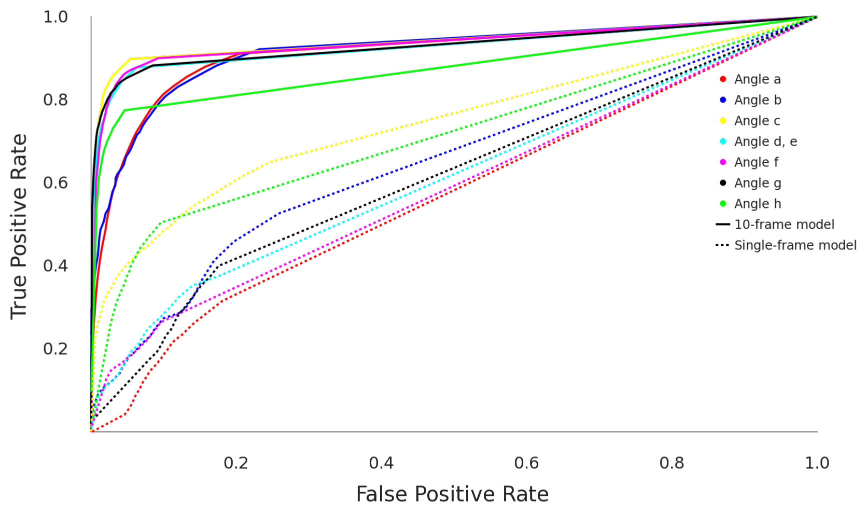


Figure 4.10. ROC curve comparison of a 10-frame model and a single-frame model trained on the 3DPeS Dataset, by scene. Solid line series correspond to the 10-frame model for specific camera angles while dashed line series correspond to the single-frame model for each camera angle.

model. When given multiple frames of information, the network learned to detect moving objects. This additional information allows the network to generalize its detection to novel scenes.

Studying the Failures

The 10-frame model made similar mistakes when tested on novel scenes as it did when tested on familiar scenes (outlined in Section 4.3.3) with one addition: it had more trouble identifying pedestrians that did not move. Figure 4.11 shows a common example of the 10-frame network missing a pedestrian who did not move in the 10 frames given to the network. The crowd of walking people in Figure 4.12, however, was classified with relatively high confidence. The 10-frame network was clearly relying on motion as a clue for detecting the pedestrians, and it was most useful when evaluating scenes never seen before.

The single-frame model could obviously not analyze motion to aid in detection, and this handicap showed clearly in the results. A survey of the detection errors showed several scenes where the network completely failed to generalize to the novel perspectives and produced seemigly random results. (Figure 4.13 shows two such instances.) Even in scenes where the network performed reasonably well, its mistakes were not in the "objects falling between grid-squares" category from Section 4.3.3; they were a failure to even detect the pedestrians. Often the single-frame model would detect the legs of a pedestrian but not the torso or head.

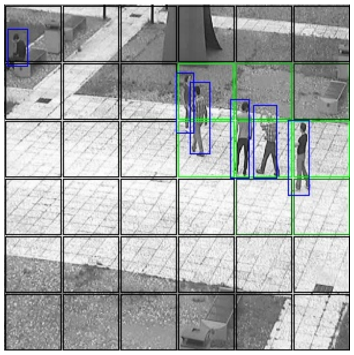


Figure 4.11. The 10-frame model had trouble finding pedestrians who did not move. Adapted from [8].

Note the person sitting and reading a book in the top-left corner of the image.

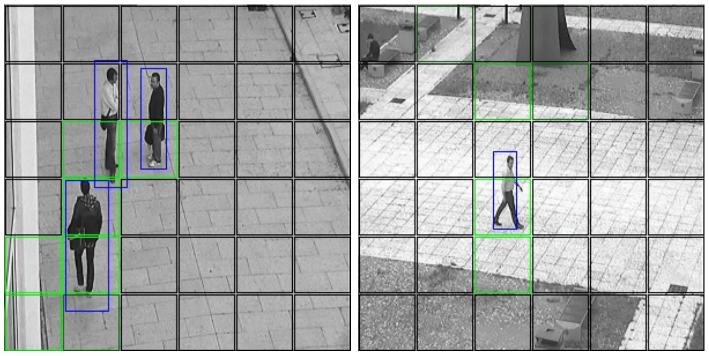


Figure 4.12. The single-frame model did learn to find pedestrians, although poorly. Adapted from [8].

The results of the two 3DPeS experiments were clear: a neural network is better able to detect pedestrians in the 3DPeS Dataset when given multiple frames instead of a single image. This suggests that the network is able to utilize information across frames to make classifications. While these are conclusive and exciting results, we wanted to apply these findings to a more difficult problem: identifying small objects at sea.

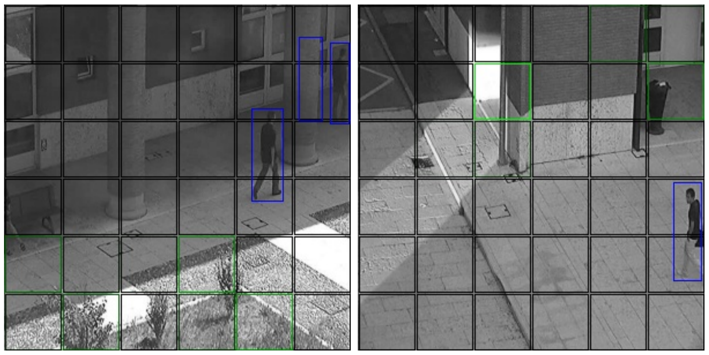


Figure 4.13. The single-frame model failed to generalize to some novel scenes in the 3DPeS Dataset. Adapted from [8].

4.4 Finding Birds in the Monterey Bay Webcam Dataset

4.4.1 Dataset Description

We chose webcam footage of the Monterey Bay as a "noisy" dataset, and trained the network to detect birds flying across the frame [30]. While the camera position was static in these sequences, the background was continuously changing due to sea state and surface debris. These video sequences often contain objects that are not annotated; we trained the network only to find birds in flight. We left out otters and buoys, as well as birds swimming but not flying. This was due to an uncertainty about what should constitute an "object" in the dataset: the human annotators usually had trouble distinguishing between sea life and general debris. To eliminated "annotation false positives" we only annotated the category of objects that humans can recognized without error: flying birds. While the target objects were moving in every sequence, a single frame rarely had enough information to make an accurate detection. Each frame had an original resolution of 640×360 pixels and was down-sampled to the network resolution of 300×300 for training and testing. Figure 4.14 shows three frames from the dataset; each frame has a positive object in it.

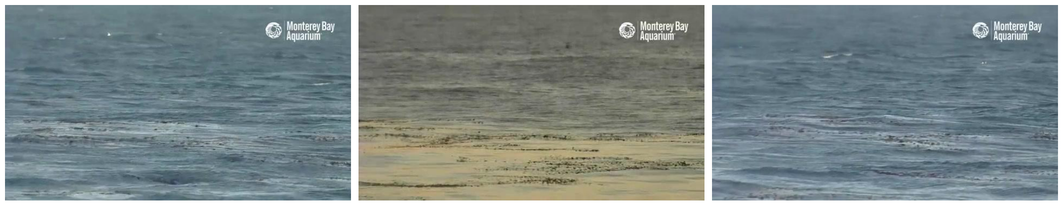


Figure 4.14. Still frames from the Monterey Bay Dataset. Each frame has a positive object in it. Source: [30].

The Monterey Bay Webcam Dataset is very unbalanced; only 14% of the frames have any objects at all. Section 3.1 describes our attempt to balance the dataset by ensuring a 50% split between frames containing an object and frames with no object. This attempt does not completely balance the data, however. Since the network makes 36 predictions for every frame, and most frames that contain any object only contain a single object; frames considered "positive" examples during pre-processing usually resulted in a single positive classification and 35 negative classifications when split into grid-boxes. The implication is that even when pre-processing ensures a 50% split between "positive" and "negative" frames, only 2% of the grid-box classifications were positive. (Section 4.4.3 describes our observation of the effect this skew had on network performance.)

4.4.2 Weight Initialization

Training the network on this dataset directly resulted in poor results (covered in section 4.4.3). While the network was able to overfit to the data and successfully identify objects in sequences previously seen, it was unable to generalize to new sets, even with a high weight decay. Since the objects being detected are so small and are often completely masked by sensor noise, the early layers of the network never learn the typical edge-detector and simple convolution concepts that are useful to most visual problems [31]. To overcome this obstacle we trained the network on two "easier" synthetic datasets and initialized the training for the Monterey Bay Webcam Dataset with the resulting weights. Starting with weights trained on larger, more visible objects allowed the early layers of the network to learn the basic convolutional filters before learning the more complex features specific to the Monterey Bay Webcam Dataset.

Detecting Large Moving Objects

In the first pretraining step we trained the network to detect a large object moving against a background of ocean waves, as seen in Figure 4.15. The object was colored either black or white and moved at a random constant speed following a linear path in a random direction. Throughout each sequence the object was only rendered 50% of the time; the network had to learn to collect information across multiple frames to determine the location of the object. This dataset was generated synthetically and included 400 sequences, averaging 100 frames each.

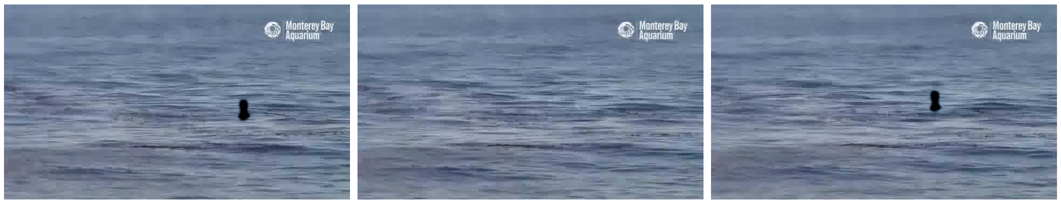


Figure 4.15. Sequence of frames from the synthetic "Large Object" Dataset. The object was only rendered 50% of the time, and can be seen missing in the middle image. Adapted from [30].

Detecting Small Moving Objects

The weights trained on the "Large Object" Dataset were used as a initial weights when training the network on a synthetic "Small Object" Dataset (Figure 4.16). This dataset was generated identically to the "Large Object" Dataset expect that the shapes were much smaller. A small point of either black or white, only a few pixels across, was rendered moving in a random direction at a constant speed against a background of ocean waves. Like the "Large Object" Dataset, the "Small Object" shapes were only rendered in 50% of the frames; the network had to utilize multiple frames to determine the precise location of the object. Table 4.3 shows that the single-frame network had trouble even on this pretraining task. While the 10-frame model achieved a respectable F1 score of 0.76, the single-frame model only scored 0.07.

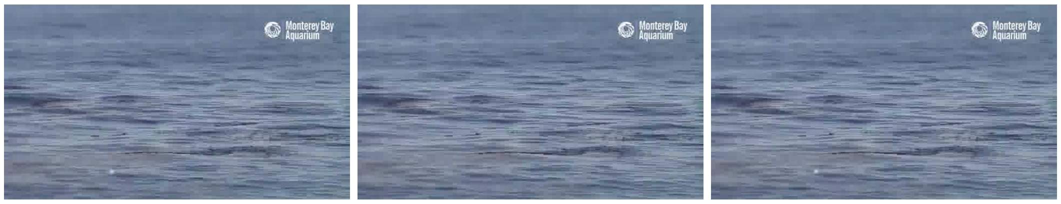


Figure 4.16. Sequence of frames from the synthetic "Small Object" Dataset. The network was trained to find the small white "dot" in the lower left of the left and right images. The center image is an instance where the object was not rendered. Adapted from [30].

4.4.3 Results

The Monterey Bay Webcam Dataset is a very challenging dataset, and it is clear that providing temporal information to the network improves performance significantly. Figure 4.17 and Table 4.4 show that a single-frame model performs only slightly better than random.

Model	Pretraining Dataset	Best F1-Score
Single-frame model	“Large Objects”	0.85
10-frame model	“Large Objects”	0.96
Single-frame model	“Small Objects”	0.07
10-frame model	“Small Objects”	0.76

Table 4.3. Network performance during pretraining.

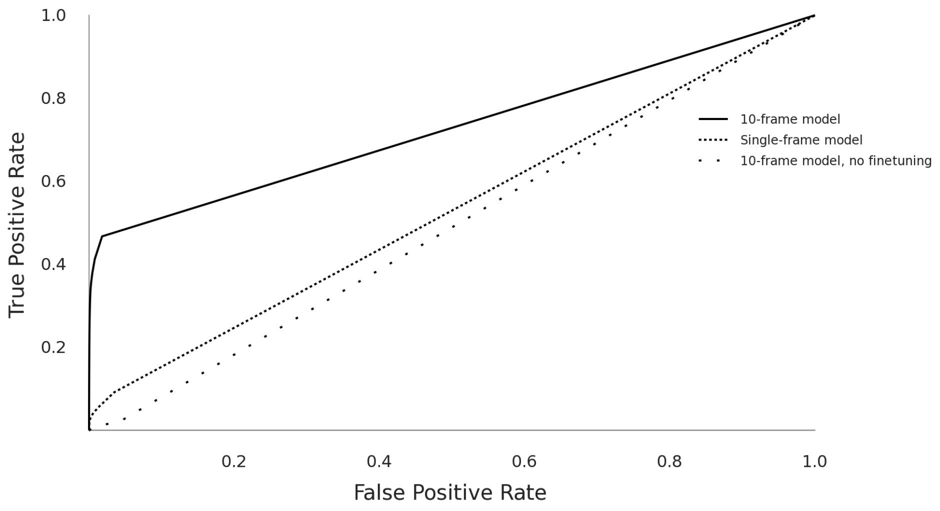


Figure 4.17. ROC curve comparison of a 10-frame model, and a single-frame model trained on the Monterey Bay Dataset. Also included is a “no finetuning” model, for which the “weight initialization” step described in Section 4.4.2 was omitted.

The multiple-frame models, on the other hand, successfully learned to detect objects in the videos.

Model	Best F1-Score	False Positive Rate	False Negative Rate
Single-frame model	0.04	0.0019	0.97
10-frame model, no pretraining	0.01	0.01	0.993
10-frame model	0.38	0.0011	0.70

Table 4.4. False positive and false negative rate comparison, Monterey Bay Webcam Dataset.

We expected that making additional frames available to the network would result in improving both the false positive rate (the network would better filter out background clutter) and improve the false negative rate (the network would be able to identify objects previously not identified.) We found, however, that while additional frames did significantly decrease the false negative rate, the additional information had a lesser effect on the false positive rate.

Figure 4.18 is a typical example of where the 10-frame model outperformed the single-frame model. These examples are of objects clearly visible but easily mistaken for waves and background clutter. The 10-frame model, however, has the advantage of knowing how the object moves. The pictured examples are of birds with a very recognizable "wing-flapping" pattern.

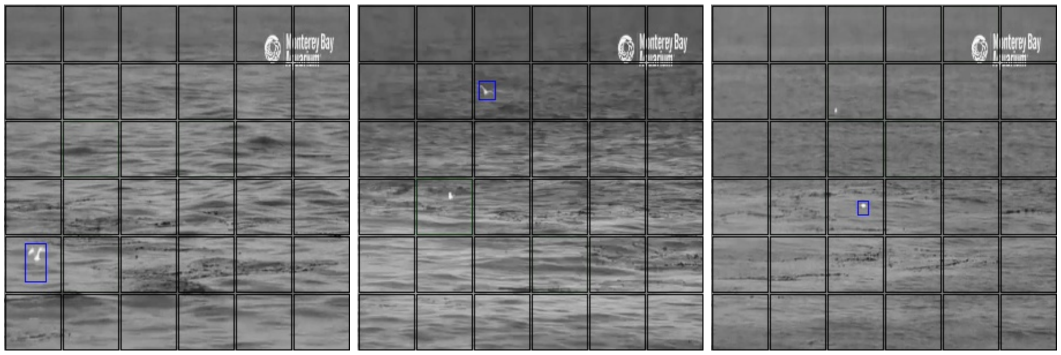


Figure 4.18. Examples of the single-frame model's false negative classifications. Adapted from [30].

Since the single-frame model had so much less information, it was unable to be certain about many objects. We observed that, as a result, it developed a bias toward "no object," or classifying a grid-square as empty. This bias resulted in a significant increase in false negatives but only a modest increase in false positives compared to the 10-frame model (see Table 4.4). The bias toward "no object" is almost certainly a result of the skewed dataset described in Section 4.4.1. This is an interesting result and an important implication of skewed data: if classification uncertainty is high, a classifier will usually "default" toward the skew. This is not necessarily a bad thing if the classifier is meant to maximize its F1 Score. Since only 2% of all regions contained objects, in the absence of compelling evidence it is most sensible to classify a region as "no object."

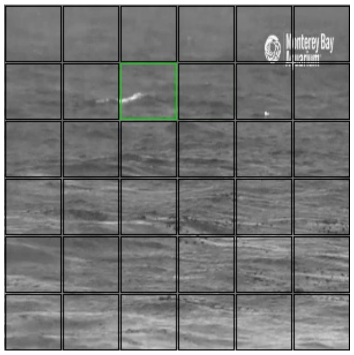


Figure 4.19. False positive example where the network misidentified a wave as a bird. 10-frame model. Adapted from [30].

Studying the Failures: False Positives

We collected the test images on which the network performed the worst. To collect these images, we analyzed the false positives in which the network assigned a score of greater than 0.9, and false negatives where the network assigned a score less than 0.05. We empirically found that these thresholds worked well to filter the results to a few hundred images in each error category. By studying these images we were able to categorize the errors into "common mistakes" that the network made on the test set.

False positives fell into one of four categories:

1. Misidentification of waves
2. Objects falling between grid-squares
3. Detecting an object "too early"
4. Other errors

A major motivation for adding multiple frames to a neural network was to give the network enough information to distinguish between interesting objects and background clutter in a maritime environment. Even if a wave looks like a bird in one frame, observing it through several frames is usually enough for a human to correctly identify it. While our 10-frame network was successfully able to ignore most of the background clutter (see Figure 4.20) it still sometimes misclassified waves as birds (as in Figure 4.19). Interestingly, there were other sea creatures present in some sequences (sea otters were common, for example) but the 10-frame network never falsely classified them as birds.

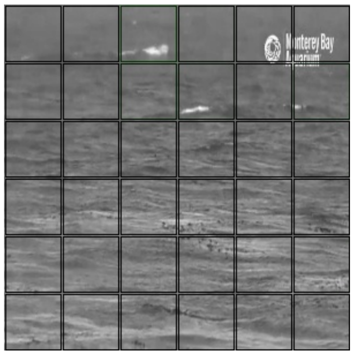


Figure 4.20. True negative example where the network correctly ignored several waves. 10-frame model. Adapted from [30].

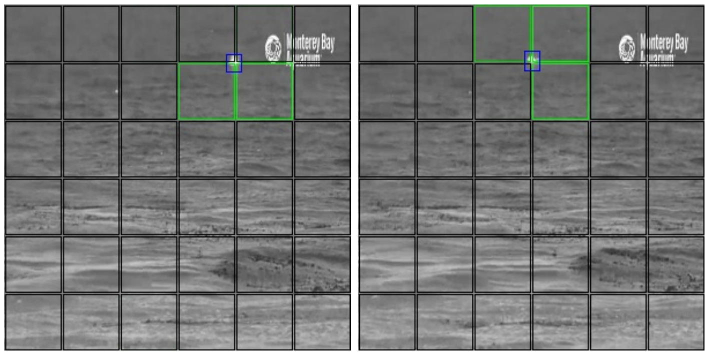


Figure 4.21. False positive example where the object of interest fell on the boundary of grid-squares. 10-frame model. Adapted from [30].

The "regions of uncertainty" described in Section 4.3.3 are even more present in the Monterey Bay Webcam Dataset. Since objects usually appear by themselves and occupy only a small fraction of a grid-square, and since the network often has to fuse information across frames to localize the objects, the uncertainty surrounding these small objects is even greater (see Figure 4.21).

A surprising source of false positive errors is shown in Figure 4.22, where the network identified an object in a region that, while empty in the frame with the error, would be occupied by an object in the next few frames. Figure 4.23 shows two examples where a bird is correctly identified, but the adjacent region is incorrectly identifies as containing an object of interest. What makes this error interesting is that the network has made a *temporal* mistake. It did not misidentify an object (the object would indeed exist), it simply classified the region as containing an object too early. While it is exciting to imagine that the network

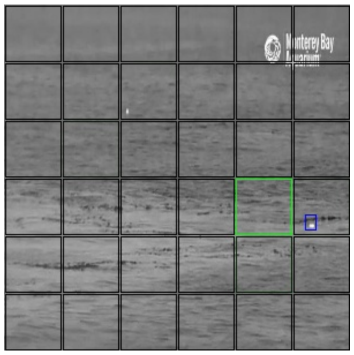


Figure 4.22. False positive example where the network highlighted where a bird would exist in the future. 10-frame model. Adapted from [30].

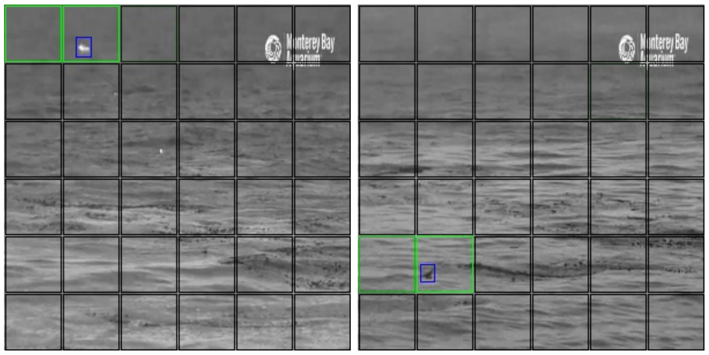


Figure 4.23. False positive examples where the network highlighted a region where objects will be in the future. In both images the birds are flying from right to left. 10-frame model. Adapted from [30].

is “predicting” where an object will be in the future, it is important to remember that our 10-frame model was able to look 0.45 seconds into the future, and thus had the information necessary to know that the object would eventually be in that region.

One possible explanation for these errors is that they are extreme cases of the previous class of error: where the object falls on the boundary of grid-squares. As a ground truth bounding box passes from one grid-square to the next, there is usually a moment when two adjacent grid-squares are classified as containing the object. The network seems to be using its knowledge of the “future” to make that determination.

With the complexity of neural networks comes a certain level of opacity in regards to their operation, and so we categorized some of our network’s mistakes as simply “other.”

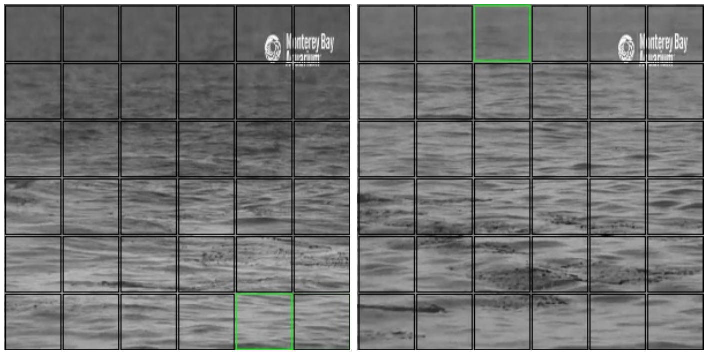


Figure 4.24. False positive examples where the network made silly mistakes. 10-frame model. Adapted from [30].

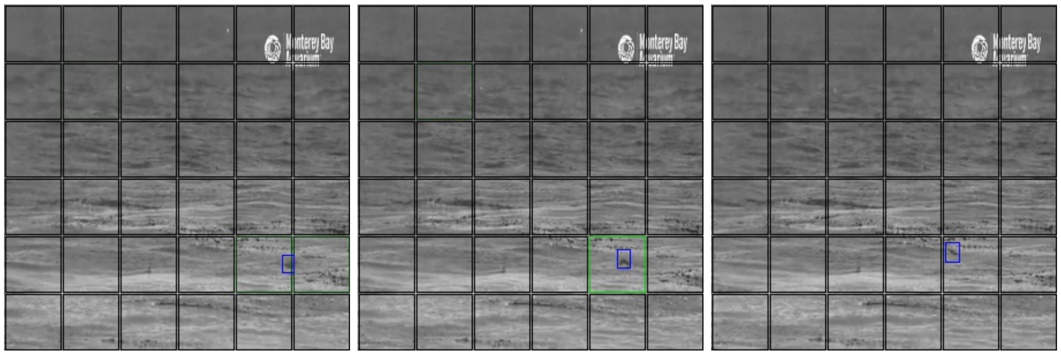


Figure 4.25. False negative examples. The network is sometimes able to identify an object. 10-frame model. Adapted from [30].

In these instances the network misclassified a region for no obvious reason; they are mistakes unrelated to nearby objects or similar-looking objects. As an example, Figure 4.24 shows two instances where the network classified a region as containing an object even though there clearly (to a human) is no object present.

Studying the Failures: False Negatives

Many of the network's false negative classifications were intermittent for a specific object; as an object passed through the frame the network would sometimes identify it, and then fail to identify it a few frames later as seen in Figure 4.25. There are two likely explanations for this category of error.

First, as has been stated before, the objects of interest in the Monterey Bay Webcam Dataset are often very tiny and there is often simply very little information available to identify an object. The little information that *is* available is usually difficult to separate from the background clutter. All of this is to say that it is a hard problem. We attempted to mitigate this difficulty by providing the network with several frames of information; if one of the frames is especially poor there is a good chance that there is a better one available. Of course, relying on past and future frames for classification adds some uncertainty: if an object can be identified in a frame 0.25 seconds earlier, it is not always clear what that says about that object's *current* location. In the end, those 10 frames are not always sufficient to make a correct classification.

Second, the network performs best when the objects of interest are in the middle of a grid-square. As an object moves toward the edge of a grid-square, the network often lowers the score for that region. This is almost certainly a result of the experimental setup covered in Section 4.3.3. These errors might simply be the "false negative" counterpoints to the network's otherwise overly-optimistic classification of objects falling on grid-square boundaries. The fact that the network will sometimes classify the surrounding regions as containing an object and other times classify the regions as empty reinforces the idea of "regions of uncertainty."

Sometimes an object is simply too difficult to identify. Figure 4.26 shows two examples of objects that would be challenging for a human to locate, assuming the task is even possible! Since the images provided to the network were down-sampled from 640×360 to 300×300 pixels, it may be the case that the objects were completely undetectable in the images given to the network even if multiple frames were available.

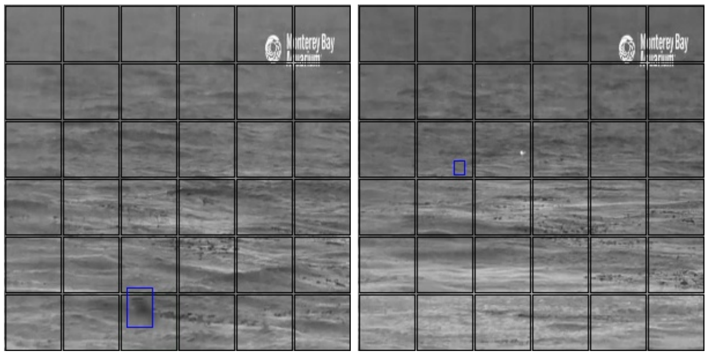


Figure 4.26. False negative examples. 10-frame model. Adapted from [30].

Note that both examples are correctly annotated; there is a bird inside each blue rectangle, and the small white object in the right image is not a flying bird.

Figure 4.27 shows an example of a “novel” object that the 10-frame version of the network was able to partially identify. While there is one other instance of a flock of birds in the training corpus, flocks appear so infrequently and look so different from every other object that the network has trouble identifying the entire flock when it does appear.

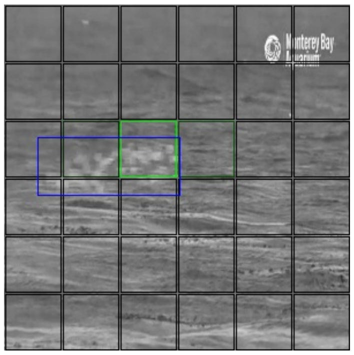


Figure 4.27. The birds (blue) cover a larger area than was detected (green). Only one other sequence contained a similar flock of birds. 10-frame model. Adapted from [30].

If an object is difficult to detect in one frame, it is likely to be difficult to detect in the other frames in the sequence. This means that if the network was unable to detect an object at all, and the object appeared in 100 video frames (about 4 seconds), the network would generate 100 false negatives. The difficulty of many objects in the dataset combined with the challenges described above resulted in a relatively high 70% false negative rate for the 10-frame model (seen in Figure 4.4).

4.5 Comparing Dataset Results

Every model performed much better on the 3DPeS Dataset than the Monterey Bay Webcam Dataset, although that result was expected. The 3DPeS Dataset provides a problem that is normally solved without mistakes by a human, while the Monterey Bay Webcam Dataset provides a problem that many humans find difficult. One of the key differences in the results was found in the nature of the errors made by the network. The 3DPeS Dataset contained larger objects that often formed groups. These objects were also relatively homogeneous in shape and contrast. The 10-frame network leveraged these characteristics by using information from surrounding regions to inform classifications (see Section 4.3.3). The Monterey Bay Webcam Dataset, on the other hand, did not contain groups of objects spanning multiple grid-boxes. We also did not observe the same cross-region information processing that we observed in the 3DPeS Dataset. Instead, we found that the network sometimes mispredicted where an object was located by predicting a location where the object would be in the future. We suspect that these errors occurred because the data in a single frame is often insufficient to locate an object; the network must combine information from several frames to localize an object in the "present." The network simply made some mistakes when combining that information. The ac3DPeS Dataset, on the other hand, usually had enough information in a single frame to localize every pedestrian. The network could depend more on information from the "present" frame to make its classifications and did not need to project object positions across frames.

The 10-frame model significantly outperformed the single-frame model in both datasets. These results show that additional frames of information can improve neural network detection performance. We chose datasets containing sequences of moving objects from fixed vantage points, i.e., surveillance video, but it is easy to imagine that providing a network with additional frames can improve performance in other classification and localization tasks. We showed that given the data, neural networks will combine information across both space and time to improve their performance.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5: Conclusion

While several of the approaches to object detection described in Chapter 2 performed extremely well for many applications, none employed neural networks capable of learning temporal features across frames. In Chapter 4 we observed and measured performance gains when providing a simple neural network with multiple video frames. We found that providing a convolutional neural network with multiple frames from a video sequence can significantly improve detection performance for the problem domains we tested.

We have been careful to separate our usage of the terms "temporal features" and "multiple frames of information." While these two terms might seem analogous to a human, it is only because extracting temporal features (such as movement and its direction) from multiple frames is nearly automatic for us. This was not guaranteed to be true for neural networks! Many of the algorithms introduced in Section 2.3 provided temporal features directly: the "Tubelets" approach, for example, calculated optical flow before training and provided that *processed* information to the network [26]. Our experiment provided only the raw pixel data to the network; the network had to learn its own motion model.

Of course, developing an operational-grade computer vision algorithm for maritime object detection cannot happen before the proper dataset becomes available. This project simulated the task with a webcam video feed of the Monterey Bay; the resulting network can only be considered a proof of concept. We used the webcam footage because actual maritime SAR footage was unavailable. While the results of the project strongly suggest that neural networks are up to the task of maritime object detection, such an algorithm will not exist until a corpus of maritime SAR footage is made available. The pace of computer vision research today and its seemingly insatiable appetite for novel problems means that once such a corpus is made publicly available, it will not be long before computer algorithms are developed that outperform their human counterparts.

One key difference between actual SAR footage and the Monterey Bay Webcam Dataset that deserves more research is the problem of a "moving camera perspective." The results from our 3DPeS "novel scene" experiment suggest that multi-frame neural networks are able to learn about movement and generalize to different backgrounds. Our Monterey Bay Webcam experiment results showed that multi-frame neural networks can adapt to dynamic backgrounds. However even in these “novel” and “dynamic” scenes the camera angle remained static. An obvious next step would be to test a network’s ability to detect an object when the camera angle changes. In this scenario there would be a difference between an object moving relative to the video frame and an object moving relative to the environment. This is a vision task that must be addressed before autonomous SAR drones can be reasonably effective.

Once computer vision is better studied in the maritime environment, new possibilities will open up: cheap SAR drone swarms, surface mine detection, force protection, and satellite surveillance are some obvious examples. Computer vision analysis of satellite imagery might even be improved by capturing several frames of information over a short period of time, possibly improving detection and classification. The idea that temporal information can improve the detection performance of a neural network can be extended beyond the realm of video data and change the way we collect data in the first place. While we attempt to improve detection capabilities by increasing the spectrum coverage and image fidelity of our sensors, we may be able to achieve similar gains simply by analyzing short sequences of images in applications where we were previously looking at single images on their own.

While our research focused on “object detection,” maritime object classification is an obvious next step. For objects in the water, does providing multiple frames of information to a neural network improve classification performance? Is there even enough information available to classify a bird separate from a human survivor? The fact that our network successfully distinguished birds from otters and ocean waves suggests that such classifications are possible. The improvements in recall and precision hint that additional frames can strengthen classification performance as well. A network with more capacity than the one designed for this project could almost certainly perform the task of maritime object classification.

Our overall conclusion is that neural network object detection performance can be improved by providing the network with multiple frames from a video sequence. The 3DPeS experiments in Section 4.3 showed a performance increase when detecting pedestrians from a fixed-angle overhead camera. In this case the network was able to collect and connect information across frames (and from different locations within a frame) to localize objects. When the network was tested on new scenes, the information from the additional frames became invaluable. The temporal information became especially important when detecting objects in the Monterey Bay Webcam Dataset; a single-frame model did only slightly better than guessing. Given multiple frames, however, the network was able to synthesize information across frames to locate objects even when they were partially obscured or missing. When detecting objects in a noisy environment, such as at sea, a multi-frame neural network has the ability to overcome some of the obstacles that make such a problem difficult.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] Office of the Chief of Naval Operations, "Navy search and rescue tactical information document," Department of the Navy, Washington, D.C., Tech. Rep. NWP 3-22.5-SAR-TAC, Sep 1997.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[3] A. Linn. (2015, December). Microsoft researchers win ImageNet computer vision challenge. [Online]. Available: https://blogs.microsoft.com/next/2015/12/10/microsoft-researchers-win-ImageNet-computer-vision-challenge

[4] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, 2017. [Online]. Available: http://www.nature.com/nature/journal/vaop/ncurrent/full/nature21056.html

[5] National Search and Rescue Committee, "National search and rescue supplement to the international aeronautical and maritime search and rescue manual," United States Coast Guard, Washington, D.C., Tech. Rep. NSS, May 2000.

[6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.

[7] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: principles and practice of background maintenance," in *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1. IEEE, 1999, pp. 255–261.

[8] D. Baltieri, R. Vezzani, and R. Cucchiara, "3DPeS: 3D people dataset for surveillance and forensics," in *Proceedings of the 2011 Joint ACM Workshop on Human Gesture and Behavior Understanding*. ACM, 2011, pp. 59–64.

[9] M. Piccardi, "Background subtraction techniques: A review," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 4. IEEE, 2004, pp. 3099–3104.

[10] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *International Journal of Computer Vision*, vol. 38, no. 1, pp. 15–33, 2000.

[11] Z. Lin, L. S. Davis, D. Doermann, and D. DeMenthon, “Hierarchical part-template matching for human detection and segmentation,” in *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007, pp. 1–8.

[12] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1. IEEE, 2005, pp. 886–893.

[13] D. G. Lowe, “Object recognition from local scale-invariant features,” in *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2. IEEE, 1999, pp. 1150–1157.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based Learning Applied to Document Recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[16] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “Overfeat: Integrated recognition, localization and detection using convolutional networks,” *CoRR*, vol. abs/1312.6229, 2013. [Online]. Available: http://arxiv.org/abs/1312.6229

[17] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[18] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient graph-based image segmentation,” *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[19] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.

[20] S. Ji, W. Xu, M. Yang, and K. Yu, “3D convolutional neural networks for human action recognition,” *Ieee Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[21] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4694–4702.

[22] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[23] V. Mnih, N. Heess, A. Graves *et al.*, "Recurrent models of visual attention," in *Advances in Neural Information Processing Systems*, 2014, pp. 2204–2212.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[26] K. Kang, W. Ouyang, H. Li, and X. Wang, "T-CNN: Tubelets with convolutional neural networks for object detection from videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 817–825.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[28] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving Neural Networks by Preventing Co-adaptation of Feature Detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[29] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[30] Monterey Bay Aquarium. (2016). Monterey Bay live web cam at the Monterey Bay Aquarium. [Online]. Available: http://www.montereybayaquarium.org/animals-and-experiences/live-web-cams/monterey-bay-cam

[31] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition," in *Icml*, vol. 32, 2014, pp. 647–655.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California