Deep Automatic Threat Recognition: Considerations for Airport X-Ray Baggage Screening

by

Kevin J Liang

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____

Lawrence Carin, Advisor

_____

Arthur Robert Calderbank

_____

Guillermo Sapiro

_____

Henry Pfister

_____

Hai Li

Dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
in the Graduate School of
Duke University

2020

ABSTRACT

Deep Automatic Threat Recognition: Considerations for Airport X-Ray Baggage Screening

by

Kevin J Liang

Department of Electrical and Computer Engineering
Duke University

Date: _____

Approved:

_____

Lawrence Carin, Advisor

_____

Arthur Robert Calderbank

_____

Guillermo Sapiro

_____

Henry Pfister

_____

Hai Li

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering in the Graduate School of Duke University

2020

Abstract

Deep learning has made significant progress in recent years, contributing to major advancements in many fields. One such field is automatic threat recognition, where methods based on neural networks have surpassed more traditional machine learning methods. In particular, we evaluate the performance of convolutional object detection models within the context of X-ray baggage screening at airport checkpoints. To do so, we collected a large dataset of scans containing threats from a diverse set of classes, and then trained and compared a number of models. Many currently deployed X-ray scanners contain multiple X-ray emitter-detector pairs arranged to give multiple views of the scanned object, and we find that combining predictions from these improves overall performance. We select the best-performing models fitting our design criteria and integrate them into the X-ray scanning machines, resulting in functional prototypes capable of simulating live screening deployment.

We also explore a number of subfields of deep learning with potential to improve these deep automatic threat recognition algorithms. For example, as data collection efforts are scaled up and the number threat categories are expanded, the likelihood of missing annotations will also increase, especially if this new data is collected from real airport traffic. Such a setting is actually common in object detection datasets, and we show that a positive-unlabeled learning assumption better fits the characteristics of the data. Additionally, real-world data distributions tend to drift over time or evolve cyclically with the seasons. Baggage scan images also tend to be sensitive, meaning storing data may represent a security or privacy risk. As a result, a continual learning setting may be more appropriate for these kinds of data, which we examine in the context of generative adversarial networks. Finally, the sensitivity of security applications makes understanding models especially important. We thus spend some

time examining how certain popular neural networks emerge from assumptions made starting from kernel methods. Through these works, we find that deep learning methods show considerable promise to improve existing automatic threat recognition systems.

Acknowledgements

To start, I would like to express deep thanks to my Ph.D. adviser, Lawrence Carin. I first came to him as a complete stranger, an undergraduate student unsure if I wanted to pursue graduate research. He saw potential, took the chance, and offered me a position in his group, an opportunity whose value I didn't fully comprehend at first but have become more appreciative of with each passing year. His encouragement and belief in my abilities played a key role in me becoming the person I am today. I especially appreciate his willingness to let a junior, untested graduate student give a software tutorial to over 150 students at the fledging Duke Machine Learning School program. I had no idea I would eventually play the role I have had in shaping the Duke Machine Learning Schools, +DataScience, and the Duke Introduction to Machine Learning Coursera over the past 4 years, and I thank Larry for helping me discover teaching as a passion.

In addition to my adviser, I am appreciative of my dissertation committee: Robert Calderbank, Guillermo Sapiro, Henry Pfister, and Hai Li. It is an honor to have such a distinguished group providing feedback and encouragement during my graduate studies. Special thanks to Guillermo for being the first to introduce me to research, so many years ago.

I am also grateful for my many collaborators and mentors over the years: David Carlson, John Sigman, Greg Spell, Dan Salo, Xuejun Liao, Chunyuan Li, Guoyin Wang, Nikhil Mehta, Nathan Inkawhich, Yuewei Yang, Liqun Chen, Yitong Li, Ricardo Henao, Yunchen Pu, Chenyang Tao, Kyle Ulrich, and the many others from the Carin group. Also, thanks to Boyla Mainsah, Sandy Throckmorton, and Leslie Collins for being research mentors during my undergraduate career, Katherine Heller for providing guidance during the first year of my Ph.D. career, and Lisa Huettel for

shaping much of my undergraduate career with her indispensable advising. Outside of Duke University, I have also been fortunate to work with Chris Gregory, Geert Heilmann, and Souleymane Diallo from Smiths Detection; Dan Strellis, William Chang, Felix Liu, and Tejas Mehta from Rapiscan Systems; and Armita Soroosh and Suriyun Whitehead from the Transportation Security Administration. None of my research would have been possible without all of your outstanding efforts, stimulating conversations, friendly encouragement, and enlightening mentorship.

The internships I have had over the years have also been instrumental in my development. It has been a privilege to work with so many talented people. Thank you Roy Williams from Microsoft; Wei Hua, Mona Mahmoudi, and Ting Yu from Google; Robbie Allen, Ya Xue, and Ikenna Odinaka from Infinia ML; and Yunchen Pu (again) and Wenlin Chen from Facebook. After each of these positive experiences, I always returned to Duke with fresh perspectives and renewed vigor.

To Duke University, I first stepped on your beautiful campus almost 9 years ago, with no idea of the journey I had ahead of me. Thank you for taking me in and being a welcoming home for all these years. I have met so many amazing people here, many of them lifelong friends and continuing sources of inspiration, and I have been given opportunities I would have never had anywhere else.

And finally, to my family: my parents and my sister, Sarah. Thank you for your love and unending support. Wouldn't have been able to do it without you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past decade, deep learning [LBH15, GBC16] has revolutionized a number of fields. Instead of traditional machine learning approaches of utilizing hand-engineered features, deep learning leverages neural networks to learn hierarchical, abstract representations from the data itself. In many settings, these learned representations have far outperformed human-selected ones, leading to advances in a number of applications, including computer vision [KSH12, RHGS15, RFB15, ALA+15, PGH+16], natural language processing [HS97, WSC+16, VSP+17, DCLT19], audio signal processing [FLTZ10, ODZ+16], and reinforcement learning [LHP+15, MKS+15, SHM+16]. While neural networks can be computationally intensive to train and require a large set of data samples [SSSG17], Moore's Law and the adaptation of graphical processing units (GPUs) for scientific computing [KW05, KSH12] coupled with the rise of big data have reduced these barriers, making deep learning increasingly feasible. No longer just an academic curiosity, neural networks have shown enough promise on traditional benchmark datasets that deep methods are being applied to or considered for an increasingly wide array of real-world applications.

One such application of interest is automatic threat recognition. Deep computer vision algorithms capable of automatically detecting objects of interest have the potential to greatly improve existing methods across many security applications. In this dissertation, we begin by focusing on the security application of bag screening. X-ray scanners are deployed at a variety of security checkpoints around the world, with the goal of preventing dangerous items from being brought into certain vulnerable areas. These security checkpoints feature prominently in air travel, aiming to

prevent potential bad actors from bringing weapons into airport terminals and, ultimately, onto airplanes. These scanners use X-ray technology to provide an internal view of carry-on baggage and personal belongings. This allows human operators to "see" inside items without having to physically open each one, thereby minimizing intrusiveness and speeding up inspections.

While these machines employ sophisticated technology to construct images of internal contents, it remains up to human operators to visually locate and identify any prohibited items should they be present. This can be a challenging task. Because of the transmission nature of X-ray imaging, scans are the result of a 3-dimensional volume being collapsed into a 2-dimensional image. While machines often position multiple X-ray emitter-detector pairs around the scanning tunnel to provide multiple views, each image can still be quite cluttered, with objects occurring stacked in varying orientations or positions. Threat items are also rare, diverse, and constantly evolving; operators must remain vigilant for many classes of items at all times, some of which they may only see once or look very different the next time it appears. To maintain throughput, decisions must be made quickly as well, often on the order of seconds. Finally, the human factor of the decision process means variability between operators, and fatigue can lead to performance deterioration over time. A computer algorithm that never tires or bores, standardized across all machines operating in all airports, and capable of making predictions quickly can provide significant assistance to human operators at existing checkpoints, perhaps taking the first steps toward rethinking or automating security checkpoints.

As a result, a number of agencies around the world have turned to computer vision algorithms using deep learning as a possible approach to automatic threat detection at airport checkpoints [AB20]. In particular, the United States Transportation Security Administration (TSA) approached Duke University to explore the potential of such

an application. We also partnered with Smiths Detection and Rapiscan Systems, whose X-ray scanners make up the vast majority of the fleet currently deployed at American airports. Smiths and Rapiscan provided much of the technical domain knowledge of airport X-ray security screening and collected much of the data needed to train our models. Since the goal of the project was to build functional prototypes to test the feasibility of performing automatic threat recognition live, it was also necessary to integrate trained model trained models into the machines themselves. Rapiscan and Smiths provided significant assistance in this regard as well.

We start with standard fully supervised deep learning methods. Specifically, we investigated the performance of popular convolutional object detection models [RHGS15, DLHS16, LAE+16] on X-ray baggage scans. Given that this kind of data is not publicly available, the first step was to collect large datasets of threats in varied environments with both the Smiths HI-SCAN 6040aTiX (Chapter 2) and Rapiscan 620DV (Chapter 3) machines [LHG+18, LSS+19]. Each of these data samples had at least one threat, with the threat identity saved as a classification label and bounding boxes indicating location drawn by hand. We then trained several models in a fully supervised manner on these datasets and compared performance across models on each target class. Given the multiview nature of X-ray baggage scanners, we explored a number of schemes for combining information from 2 or 4 images of the same bag from different perspectives, but ultimately found that a simple OR-gate style decision-making process was sufficiently effective, significantly boosting performance over using just a single view or image. Finally, having achieved promising performance on held out test sets, we then proceeded to integrate trained algorithms directly into Smiths and Rapiscan machines, projecting our model's predictions directly onto the viewing screens a human operator might see at a checkpoint.

While we were able to collect sizable datasets for supervised training, a major

weakness with this approach is how expensive data collection is: despite strategies to accelerate bag assembly and threat annotation, it still took about 400 worker-hours to collect each dataset. On the other hand, millions of scans are conducted at US airports everyday. This type of data is commonly referred to as Stream-of-Commerce (SOC), and our collaboration with the TSA opens up the possibility of tapping into this near limitless source of data, including threat examples: for example, the TSA regularly finds thousands of loaded firearms in carry-on baggage every year [Wag20, TSA20]. Under current operational procedures, this data remains unlabeled and uncollected. However, given collaboration with the TSA or other security agencies, it is possible to implement a system where the many items caught at checkpoints daily are annotated and saved. This would make the SOC data a valuable data source. Unfortunately, it is highly probable that not every threat in the SOC is caught, leading to underlabeled data. In Chapter 4, we examine the impacts of such data during training [YLC20]. Notably, we find that many popular object detection datasets [EVW+10, LMB+14, YWLS17, KZG+17] are missing annotations for many object instances, as the complexity of object detection scenes make an exhaustive labeling difficult. We find that recasting object detection from the implicitly assumed positive-negative (PN) classification problem to a positive-unlabeled (PU) one can improve detection performance for many object detection datasets.

Real-world environments tend to change over time, and as a result, data distribution shift is a natural tendency. On the other hand, many machine learning models are trained with static datasets that represent a snapshot in time. Over time, a model trained with such a dataset may become stale and unable to adapt to new kinds of inputs. This may present a challenge for deployment of automatic threat recognition systems at airports, as passenger bags, electronics, fashion, and the threats themselves are likely to evolve. The changing of the seasons are also likely to inject an

annual cyclical nature to the types of items carried. Model's that are able to learn in a continual manner are more likely to perform well over time. Given that it is a common concern for machine learning, continual learning [PKP+19] has garnered considerable interest in recent years. In Chapter 5, continual learning within the context of generative adversarial networks [GPAM+14] are explored [LLWC18a]. We find that the dynamics of the adversarial game played by the generator and discriminator tend to lead to oscillations due to catastrophic forgetting [MC89, Rat90], which can be combated with continual learning techniques.

Especially for sensitive applications like security screening, we often care about model interpretability, so we can identify how models come to the conclusions they make. On the other hand, neural networks are commonly referred to as "blackbox." While this characterization may be a little strong, it is fair to say that how deep learning methods make decision is less clear than other types of machine learning, like kernel methods. Being able to understand how neural networks work go a long way towards building trust in such systems, especially when the consequences of a wrong answer are dire. In Chapter 6, we explore connections between kernel methods and sequential neural methods [LWL+19]. In particularly, we see how certain assumptions can allow one to derive popular sequential neural networks like Long short-term memory (LSTM) [HS97] and the convolutional neural network [LB95] emerge naturally from kernel methods.

We conclude in Chapter 7 with a summary of our contributions and point out some next steps to further improve the model and prepare for full-scale deployment. We also identify some interesting future research directions relevant to automatic threat recognition. Through these works, we have demonstrated the potential for deep convolutional object detection models within the context automatic threat recognition at airport checkpoints.

Chapter 2

Automatic Threat Recognition of Prohibited Items at Aviation Checkpoints with X-ray Imaging: A Deep Learning Approach

2.1 Introduction

It is the responsibility of the Transportation Security Administration (TSA) to ensure the safety of the traveling public within the US, including the over 2.5 million passengers passing through American airports each day [Adm19]. As such, before boarding an airplane, every passenger must pass through a security checkpoint, where the TSA screens carry-on baggage and personal belongings for dangerous and prohibited items. To facilitate screening, the TSA employs dual-energy multi-view X-ray scanners produced by Smiths Detection and other vendors that provide a non-intrusive internal view of bags. These scanners produce images color-coded to show material properties that are then displayed on screens for human Transportation Security Officers (TSOs) to examine. Bags or bins containing potential threats are removed for further inspection.

The current Concept of Operations (CONOPs) requires the TSO to visually inspect each image to pick out threats, which can be a challenging task. In this context, threats refer to items prohibited by the TSA, which can vary widely, including (but not limited to) firearms, sharps, blunt weapons, precursors, and explosives. Not only do these items come in many different and evolving forms, they are also often packed

in cluttered bag environments, and many can be confused with benign objects of similar material properties or shapes. Certain threats are quite rare as well, requiring TSOs to maintain vigilance over long shifts. Moreover, in order to maintain high passenger throughput at the checkpoints, TSOs must make their decisions quickly.

While challenging for humans alone, computer algorithms that analyze scans alongside human operators may boost overall performance. Current scanners already implement algorithms that calculate material properties from dual-energy multi-view scans, automatically highlighting objects or regions that might contain explosives or other prohibited items. It is of TSA interest to extend this automatic detection capability to support operator detection of other prohibited item classes, such as firearms or sharps. Such an algorithm must be accurate enough to be trusted, have a low enough false alarm rate so as to not be a distraction, and be fast enough to not slow down current operations–we aim for a rough cutoff of about a second per bag.

Within the greater field of computer vision, the task of localizing and classifying objects is a canonically studied problem, commonly termed "object detection." In this context, localization refers to determining the location of an object within an image, often by producing the coordinates of a box that tightly bounds it (a "bounding box"); classification refers to the selection of one of a pre-determined number of class labels for each such object. Locating and identifying threat objects is exactly what TSOs at security checkpoints do every day, so developing the ability to do so automatically is of value. In recent years, the emergence of deep learning, a subfield of machine learning, has resulted in an unprecedented leap in the performance of object detection models. In particular, methods based on convolutional neural networks [LBD$^+$89, KSH12, IS15, HZRS16, SIVA16, HZC$^+$17] have resulted in algorithms that have proven effective at detecting a wide range of object classes [GDDM14, Gir15, RHGS15, LAE$^+$16, DLHS16, HRS$^+$17].

The bulk of object detection research has focused on datasets of natural images [EVW+10, LMB+14, RDS+15], but X-ray scans of baggage possess certain unique aspects. X-ray scans are produced by transmission (photons pass completely through the target), as opposed to reflections off of surfaces. This means that individual items can appear superimposed on top of each other, while also appearing in any orientation. Additionally, multiple X-ray detectors are positioned within the scanner to provide multiple views of the same object, unlike single-perspective natural images typically considered in object detection. Nonetheless, deep learning has already demonstrated some success for X-ray image security screening [RJMG17, AB17, AKWB18]. The work shown in this paper, however, is part of the first effort to incorporate deep learning in real-world systems at US airport security checkpoints. The goal of this work is to detect and automatically highlight prohibited items in bags as well as determine the performance of these methods on datasets collected for this study. Although the list of classes to detect for this research effort is quite extensive, discussion will be limited to the detection of firearms (e.g. guns) and sharps (e.g. knives).

In this paper, we describe methods and present results from an ongoing research effort funded by the TSA to develop a deep learning based Automatic Threat Recognition (ATR) system for airport checkpoint scanners. In Section 2.2, we describe (1) the Smiths Detection platform used to collect the X-ray images and (2) the data collection and labeling protocols. In Section 2.3, we introduce five deep object detection models which we apply to the labeled data, as well as metrics for evaluating system performance and the hardware set-up of the deployable prototype. For results (Section 2.4), we show object detection metrics on firearms and sharps datasets, and we demonstrate improvement by combining detection results from the multiple views of a single scan. We also describe a test to extrapolate how such a prototype system

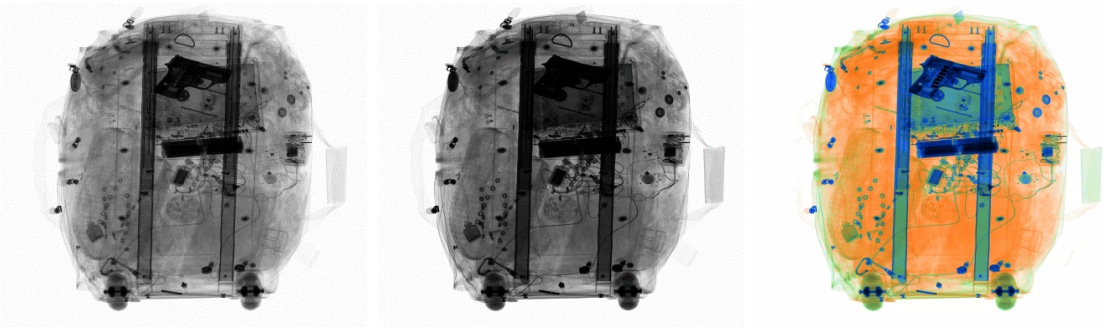
might perform when deployed in the field, showing threat detection alongside false alarm rates.

2.2 Data Collection

2.2.1 Smiths Detection X-ray System

Data used for the training and testing of the ATR methods were collected by Smiths Detection personnel using a Smiths Detection host X-ray system. The platform is a cabinet X-ray security system that contains four separate pairs of 160 KeV X-ray sources and detector arrays, arranged opposite each other around a tunnel. Passenger bags and other belongings are carried through the tunnel by a conveyor belt at a rate of 240 mm/sec. Each X-ray beam is collimated to a narrow width for optimal resolution, while the X-ray line detectors are arranged linearly; radiation emitted from the sources passes through the objects being scanned before reaching the detectors. The system is dual-energy: the X-ray detectors measure intensity at a high and low energy band. Image slices of a given view are assembled to produce two grayscale images corresponding to the two energies (see Figures 2.1a and 2.1b), yielding eight images in total across the four views. All training and evaluation were performed on these low and high energy images.

Although outside the scope of this paper, for purposes of illustration it is helpful to show color images, which make certain material characteristics easier to see with the human eye. The images that are displayed to the TSOs at airport checkpoints are post-processed color images that fuse the high and low energy scans to estimate the effective atomic number, Z-effective, of materials along the X-ray path from source to detector; Figure 2.1c shows the color image corresponding to the high and low images in Figures 2.1a and 2.1b, respectively. The colors have been selected to correspond to three different Z-effective bands. Orange corresponds to materials of



(a) (b) (c)

Figure 2.1: (a) High energy. (b) Low energy. (c) Color-mapped Material lumination. Various image types generated by the Smiths Detection host X-ray system. Images scanned with a laboratory prototype not in TSA configuration.

low Z-effective (e.g. organic materials), green corresponds to materials of medium Z-effective (e.g. ceramics), and finally blue corresponds to materials of high Z-effective (e.g. metals). The brightness (or intensity) of the color image reflects the absorption (and thus provides information on relative thickness) of the materials. The process for computing the colors is proprietary and uses a periodic calibration process to assure consistent results.

While visually helpful to humans, it was found that the material color-mapped RGB images do not provide any noticeable benefit to the deep learning algorithms. We conjecture that this is because the color mapping is simply a transformation of the high and low energy images, and deep learning is capable of effectively learning its own representations. Because of these considerations, we use the high and low energy images for training the model, but use color-mapped material lumination when presenting results.

2.2.2 Images and Labeling

Two types of image data were used in the training of the ATR algorithm: non-threat and threat. The non-threat data, referred to as Stream-of-Commerce (SOC) data,

were acquired over multiple days from real traffic at airport checkpoints. These data are generally assumed not to contain any threats and represent negative examples for training and false alarm evaluation. Positive training examples with threats were collected on multiple occasions at the Transportation Security Laboratory (TSL) and at a Smiths Detection laboratory. For this investigation, a total of 37 hand guns, 92 pocket knives, and 20 other mixed sharps were scanned in approximately 100 different pre-packed bags. Guns and sharps of several sizes were considered. Over the course of a few days, 2022 scans of guns, 1350 scans of pocket knives, and 706 scans of other mixed sharps were acquired.

A systematic process was performed to acquire the data. Sets of bags were filled with contents typical of passenger luggage, and a threat item was packed into each. These bags were then scanned in multiple positions and orientations with the X-ray system. Upon completion of a number a scans, the threat was typically moved to a new bag in the bag sequence, and the scanning process was repeated. Two measures were implemented to avoid overfitting on specific bags. First, bag contents were periodically altered: this involved adding materials to the bag to inject clutter into the scene or partially obscure the threat object in at least one view. Second, the bag sets were rotated so that no single bag was scanned excessively throughout the data collection. To ensure that the threat objects were scanned in multiple perspectives, their location and orientation within the bags were also varied. This occurred while moving the threats from one bag to the next during data collection. For instance, a small gun that was laid flat (horizontally) in the center of Bag 1 may be placed along the front of Bag 2 in a vertical orientation.

Most object detection algorithms are supervised, meaning that they require training data with labels in the format of the expected output. Within this context, this means that both image scans of threats as well as threat localizations need to be col-

(a) (b) (c)

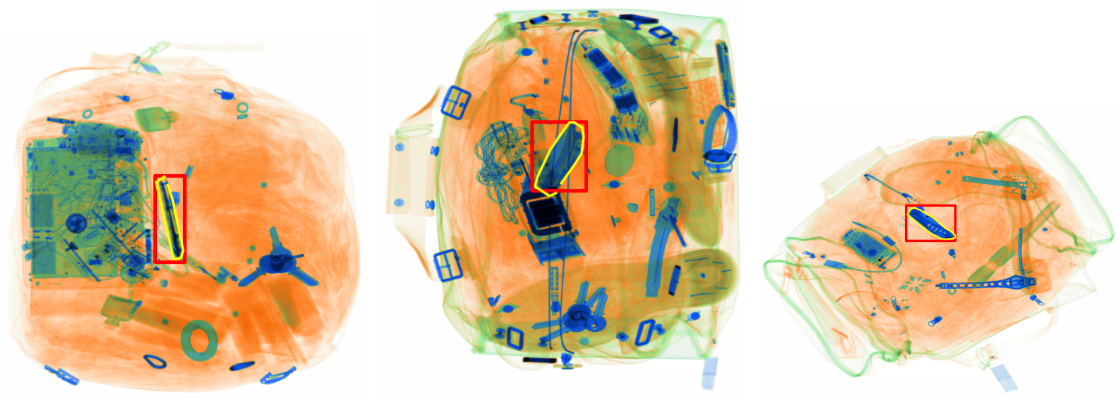
Figure 2.2: Example bounding boxes of firearms in red. Images scanned with a laboratory prototype not in TSA configuration.

lected. Labeling of the training set was performed manually using a Smiths Detection proprietary utility to draw an outline around each threat object. These outlines were then used to create a binary mask of the threat object's location. Since the methods explored in this paper rely on bounding boxes rather than pixel-wise segmentations, the min and max coordinates of these masks were used to generate bounding boxes to train the model; see Figures 2.2 and 2.3 for examples.

2.3 Methods

2.3.1 Models

Virtually every state-of-the-art object detection algorithm begins with some form of Convolutional Neural Network (CNN) operating as a feature extractor, followed by some form of specialized architecture for producing bounding box coordinates and classifications. The primary difference between these algorithms pertains to this latter part. While there have been many object detection models that have been proposed in recent years, we focus on a few popular models with high performance: Faster Regions with Convolutional Neural Networks (Faster R-CNN) [RHGS15], Single Shot



(a) (b) (c)

Figure 2.3: Example bounding boxes of sharps in red; segmentation outlines also visible in yellow. Images scanned with a laboratory prototype not in TSA configuration.

MultiBox Detector (SSD) [LAE+16], and Region-based Fully Convolutional Networks (R-FCN) [DLHS16].

For the convolutional feature extractor, many CNNs have been developed, primarily designed for image classification; however, it has been shown that such CNNs do indeed also work well for object detection [GDDM14] and that there is a positive correlation between image classification and object detection performance [HRS+17]. However, classification accuracy is not the only consideration; networks with larger numbers of parameters may have higher representational power, but are often computationally slower as a result (see Table 2.1 for a comparison). Since maintaining high throughput of bags at airport checkpoints is of interest, the algorithm cannot take an arbitrarily long time to deliberate.

Each of the aforementioned object detection models, termed meta-architectures, along with several different CNNs have been implemented in TensorFlow [AAB+15] as part of Google's Object Detection API [HRS+17], and we leverage these implementations as the basis for our experiments. Models vary from approximately 5 to 50 frames per second, depending on the CNN and detection meta-architecture. Note,

Table 2.1: Top-1 accuracy on ImageNet classification and model size for the CNNs used in experiments [HRS+17].

CNN	Top-1 Accuracy	Number of parameters
MobileNet [HZC+17]	71.1	3,191,072
Inception V2 [IS15]	73.9	10,173,112
ResNet-101 [HZRS16]	76.4	42,605,504
Inception ResNet V2 [SIVA16]	80.4	54,336,736

the model must process all four views and perform additional overhead to acquire and display the results, but even the slowest of these models still meets our target of delivering results within one second.

We train each model entirely on data from a single class (firearms or sharps) and report results individually. Each model can be run independently on a particular scan to look for threats of a particular category. However, the models were originally developed for multi-class discrimination, and nothing prevents us from doing the same. Training separate models was due to the order in which data was collected and annotated.

2.3.2 Evaluation Metrics

In order to quantify model performance, we borrow several common metrics from the object detection and information retrieval literature:

- *Intersection over Union (IoU)*: IoU is the ratio calculated by the intersection (overlap) of two sets divided by their union. A value of 0 implies no overlap, while a value of 1 means that the two sets are equal (Intersection = Union). See Figure 2.4 for examples.

- *True Positive (T_p)*: Defined as a correctly classified box that has an IoU above

a threshold (commonly 0.5) with a ground truth box[1].

- *False Positive (F_p)*: A proposed bounding box that either misses the classification or is not tight enough to achieve an IoU above the set threshold.

- *False Negative (F_n)*: A ground truth object that was not properly bounded and classified.

- *Precision ($\frac{T_p}{T_p+F_p}$)*: The proportion of proposed bounding boxes (algorithm-produced detections) that are correct.

- *Recall ($\frac{T_p}{T_p+F_n}$)*: The proportion of objects (ground-truth) that were correctly detected by the algorithm.

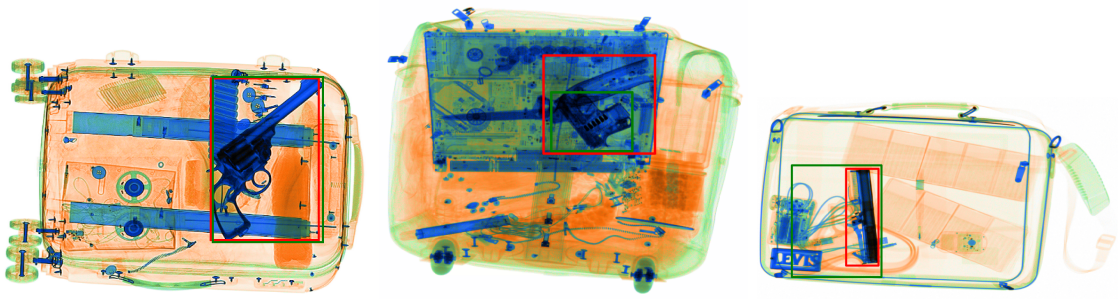
- *Average Precision (AP)*: The area under the curve (AUC) of the precision-recall curve for a single class.

- *mean Average Precision (mAP)*: The mean of the APs across all classes. For a single class problem (as we consider here), the mAP is equivalent to the class AP.

- *Percent Correctly Localized (CorLoc)*: Percentage of positives correctly identified as a T_p by the model. Equivalently, the recall value for the point on the precision-recall curve at which we choose to operate.

The latter two are reported for the quantitative results to compare performances. Additionally, precision-recall curves are shown to give a sense of performance for various operating thresholds.

[1]Often object detection algorithms produce many bounding boxes with high IoU with the ground truth object. While these are all technically correct, this kind of output is not as desirable, so often only one true positive is allowed per ground truth. Non-maximal suppression [NV06] is typically used to cut down the number of "repeat" detections for a single object.



(a) (b) (c)

Figure 2.4: Ground truth boxes shown in green; generated detection boxes shown in red. Images scanned with a laboratory prototype not in TSA configuration. (a) IoU = 0.92, alarm is above IoU threshold, resulting in a T_p. (b) IoU = 0.43, alarm is too big, resulting in a F_p and a F_n. (c) IoU = 0.31, alarm is too small, resulting in a F_p and a F_n.

2.3.3 Multi-View Evaluation

Many X-ray systems have several X-ray detector lines, providing orthogonal views of a bag. As alluded to in Section 2.2.1, the Smiths Detection host X-ray system used for data collection provides four views, meaning each scan results in four images. While these can be treated independently during ATR, this ignores spatial correlations of objects between views (see Figure 2.5). In the field, TSOs do not ignore threats which appear in only a single view, so we can consider the performance of the ATR if we apply an OR-gate to detections in scans. This means a T_p is only required in one out of four scans, while all F_ps are counted the same as in single-view evaluation.

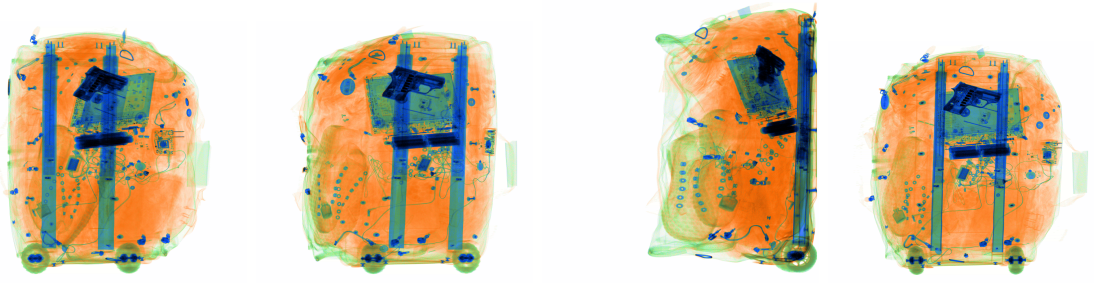


Figure 2.5: Four views generated during a scan. The Y-axis is the conveying direction. Images scanned with a laboratory prototype not in TSA configuration.

Multi-view performance is estimated through two perspectives: object detection and real-world threat recognition. For each, the deep learning object detection algorithm is first run on all views of each scan independently. Performance is then scored accordingly, depending on the metric. Deep learning object detection literature is primarily concerned with the concept of information retrieval: there are ground truth objects in each image, and the goal is to tightly bound as many as possible while avoiding false positives and mislabeled objects. mAP and CorLoc capture these objectives. For the implications of ATR in the field, we choose to analyze detection and false alarm rates on a per bag basis. In Section 2.4.2 we discuss results of multi-view to object detection metrics, and in Section 2.4.3, we show simulated real-world performance and the corresponding false alarm rates.

2.3.4 Hardware Implementation

The object detection models are trained on labeled data with a NVIDIA Titan X GPU. After training has converged, learned model weights are saved for inference. Training and evaluation of images can be conducted with a pre-scanned and labeled dataset on any workstation set-up with enough computational power. Results shown throughout the rest of this paper were evaluated offline on a held-out test set, for speed and reproducibility.

The end goal, however, is to deliver a system that can be deployed to airport security checkpoints. Therefore, initial laboratory prototypes pipe scanned images to a GPU and computer bolted to the exterior of the X-ray scanner to compute threat locations, which are then sent back and projected on top of the color images on the main display. As part of this effort, such a prototype system has been demonstrated to TSA sponsors in a setting and manner consistent with anticipated evaluation and possible field use.

2.4 Results

2.4.1 Object Detection Evaluation: Single View

ATR performance was first evaluated using object detection metrics (mAP, CorLoc), treating all four views generated from a single bag as independent images. Images were randomly shuffled into a roughly 70:10:20 training:validation:test split at the bag level, ensuring views of the same bag ended up in the same split.

Firearms

With a single view, all tested object detection models do well on a dataset of 2022 firearms scans (see Figure 2.6a and Table 2.2). An SSD model with a MobileNet V1 convolutional feature extractor, designed for speed and compactness, has the lowest mAP and CorLoc, but still achieves 0.9393 and 0.9295 respectively. Faster R-CNN with ResNet101 achieves the highest mAP of 0.9644, while R-FCN with ResNet101 has the highest CorLoc of 0.9550.

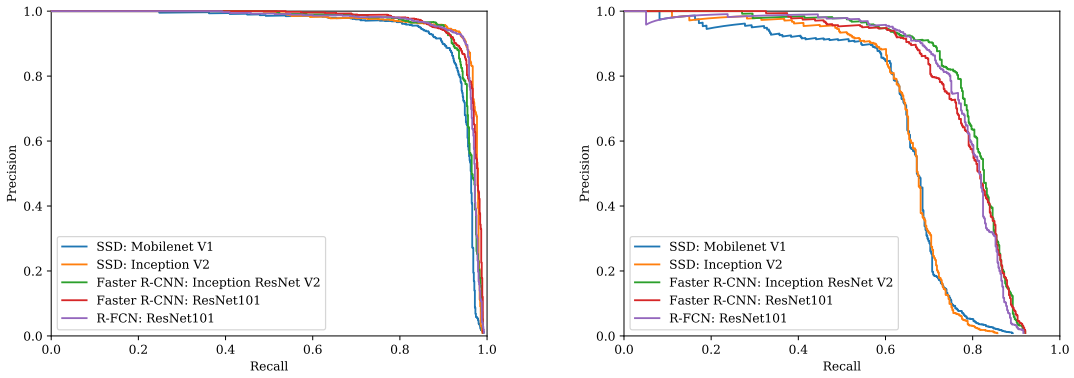


Figure 2.6: Precision-recall curve of each objection model on (a) firearms and (b) sharps data. Curves approaching the upper right corner imply better performance.

Table 2.2: Performance of each object detection model on firearms and sharps, treating all views as independent images. Best values for each class category and metric are bolded.

	Firearms		Sharps	
Model	mAP	CorLoc	mAP	CorLoc
SSD: MobileNet V1 [LAE+16, HZC+17]	.9393	.9295	.6463	.6872
SSD: Inception V2 [LAE+16, IS15]	.9621	.9514	.6575	.6828
Faster R-CNN: Inception ResNet V2 [RHGS15, SIVA16]	.9546	.9478	**.8003**	**.7775**
Faster R-CNN: ResNet101 [RHGS15, HZRS16]	**.9644**	.9514	.7863	.7621
R-FCN: ResNet101 [DLHS16, HZRS16]	.9591	**.9550**	.7884	.7753

Sharps

In Figure 2.3, we show results of single-view evaluation for all 5 object detection models on a dataset of 20 mixed open and closed blades in baggage. The dataset contains 274 scans of fixed-blade knives, 210 scans of pocket knives, 74 scans of scissors, and 148 scans of tools with sharp edges. The ground truths for all of these sharps were used to train a single-class sharps detector and then tested on held-out test scans.

Sharps pose several challenges different from firearms. By visually comparing Figures 2.2 and 2.3, it can be seen that knives provide a much smaller profile for the algorithm to find. Furthermore, depending on the sharp's orientation and surroundings, some views may be uninformative if the X-ray beam hits a knife edge-on or traverses through the handle and down the blade; knives, with their smaller size and thin aspect ratio, are also easily obscured by common opaque objects, such as metal ribbing or bottles. As such, we expect performance on par with firearms to require additional effort, and both Figure 2.6b and Table 2.2 illustrate this. In particular, it appears that the SSD models do much worse than the other models. We hypothesize that this is due to the smaller number of knives training samples and higher variation in aspect ratios of bounding boxes relative to firearms being exacerbated by the

way learned variables are arranged in the SSD architecture. More investigation is necessary to draw any concrete conclusions.

2.4.2 Object Detection Evaluation: Multi-View

At this time, ATR performance on firearms is much better than sharps. Part of the lower performance on sharps may be due to uninformative views of the threat being more likely for thinner objects. However, as noted in Section 2.2.1, the Smiths Detection host X-ray system provides four different angles of the same object, potentially offering clearer perspectives of an object occluded in one such view. By using the information combination scheme (OR-gate) outlined in Section 2.3.3, we see a substantial jump in performance (Figure 2.7, Table 2.3). A similar scheme can be employed for firearms, but we omit this here because of the already excellent performance with a single view.

Figure 2.7 shows the gain in precision-recall for multi-view evaluation applied to the same results which were presented with single-view evaluation in Section 2.4.1, while Table 2.3 shows the gain in mAP by considering multi-view evaluation. For the largest network, the Faster R-CNN with Inception ResNet V2, the mAP increased by 19.2% to .9347.

Table 2.3: Performance on knives, incorporating all views as described in Section 2.3.3. Better performance for each model bolded.

	mAP-Independent Views	mAP-Multi-View
SSD: MobileNet V1 [LAE+16, HZC+17]	.6425	**.7852**
SSD: Inception V2 [LAE+16, IS15]	.6541	**.8020**
Faster R-CNN: Inception ResNet V2 [RHGS15, SIVA16]	.7836	**.9347**
Faster R-CNN: ResNet101 [RHGS15, HZRS16]	.7837	**.9283**
R-FCN: ResNet101 [DLHS16, HZRS16]	.7976	**.9383**

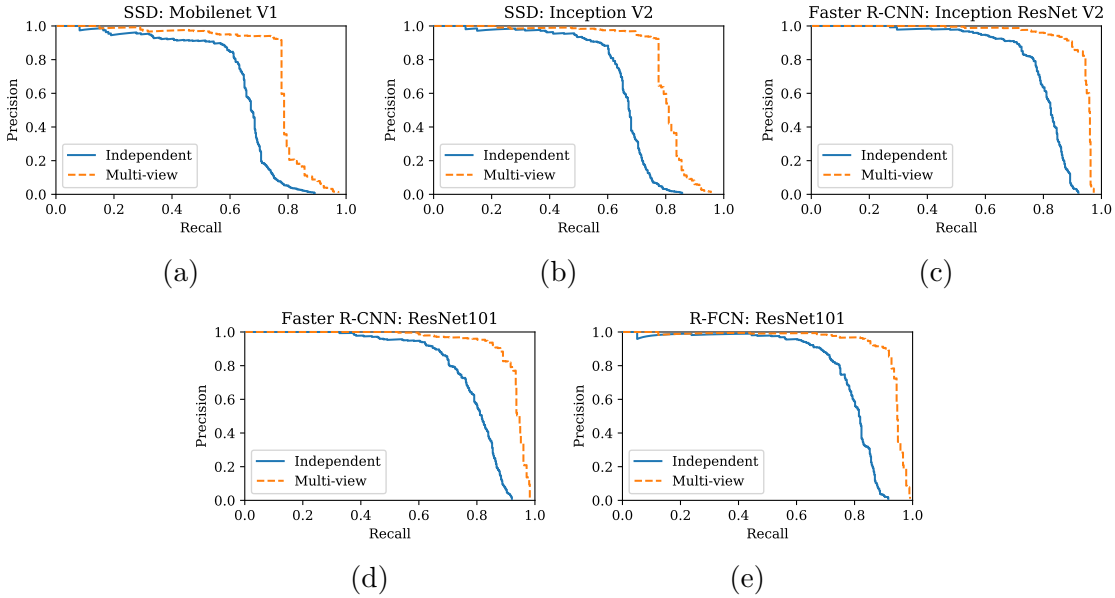


Figure 2.7: Performance gain of the precision-recall curve (orange dotted line) for each model by incorporating all four views for the sharps dataset. Each of the blue lines correspond to one of the colored lines in Figure 2.6b, which were computed using single-view evaluation. Curves further up and to the right indicate better performance.

2.4.3 Detection vs False Alarm Evaluation

As proof-of-concept and to demonstrate feasibility, a trained Faster R-CNN with ResNet101 was evaluated on a combined test set comprised of 260 pocket knife scans, 1300 firearm scans, and 15000 real SOC scans, which are assumed to not contain firearms or sharps. For this experiment, we considered detection and false alarm rates at the bag level. An alarm satisfying the T_p criteria in Section 2.3.2 in any of the four views resulted in the entire bag being labeled as a detection. False alarm rate was calculated as the proportion of bag scans where any of the four views had a F_p bounding box produced by the object detection algorithm; if there are multiple false alarm bounding boxes produced, the scan still only counts as a single false alarm. In some sense, this approach is more amenable to the security checkpoint application, as a single alarm in any views should be cause for a TSO to investigate further. For

the data collected in this study, the ATR captures 91.9% of views containing firearms at a 1% false alarm rate and 89.8% of sharps at a 3% false alarm rate with single-view evaluation. However, by utilizing all four views, the ATR algorithm is able to capture 95.5% of bags containing firearms at a 1% false alarm rate and 94.0% of bags containing sharps at a 3% false alarm rate.

2.5 Conclusions

Despite major advances in threat detection for aviation security, there are still challenges regarding both operator and algorithm efficiency detecting many prohibited items. Of special interest are weapons such as firearms and knives, which continue to be frequently encountered by TSOs. Results of applying deep learning techniques to this setting both in offline computer testing and live demonstrations indicate that this approach can detect prohibited items with high accuracy, minimal false alarm rates, and no adverse impact to passenger throughput, especially for firearms detection. A simple multi-view evaluation has also been demonstrated to improve performance, especially for items like sharps, which may be difficult to spot in just a single view. A more sophisticated algorithm that combines information between views may do even better, but we leave this to future work.

Ultimately, a goal of the TSA is to lessen the burden on the TSOs by instituting a computer-aided CONOPs where TSOs only have to review on alarm, or even transition to an entirely automated system. To achieve this, a wider range of prohibited items are being targeted in future work. These items, including blunt weapons, precursors, and flammable liquids, come in almost infinitely many variations in size, shape, texture, and materials, but given the promise shown by deep learning in this initial study, Smiths Detection and the TSA are well positioned to bring ATR systems able to detect such classes to airport checkpoints in the near future.

Chapter 3

Toward Automatic Threat Recognition for Airport X-ray Baggage Screening with Deep Convolutional Object Detection

3.1 Introduction

The Transportation Security Administration (TSA) oversees the safety of the traveling public in the United States of America. One of the most visible functions of the TSA is security screening of travelers and their personal belongings for potential threats. Handsearching each passenger's bag would be both time-consuming and intrusive, so X-ray scanner systems such as the Rapiscan 620DV are deployed to remotely provide an interior view of baggage contents. Many real threats are captured nationwide: in 2018, for example, 4239 firearms were found in carry-on bags, and more than 80% of these were loaded [Wag20]. These numbers have steadily grown in recent years as air traffic has continued to increase nationally. The capability of finding these objects effectively is an important concern for national security.

Currently, the detection of prohibited items relies on Transportation Security Officers (TSOs) to visually pick out these items from displayed image scans. This is challenging for several reasons. First, the set of prohibited items that TSOs must identify is quite diverse: firearms; sharp instruments; blunt weapons; and liquids, aerosols, and gels (LAGs) with volumes exceeding the TSA-established thresholds all pose security concerns. Second, the majority of scans are benign, yet TSOs must remain alert for long periods of time. Third, because X-ray scans are transmission images, the contents of a bag appear stacked on top of each other into a single, often

cluttered scene, which can render identification of individual items difficult. The Rapiscan 620DV provides dual perpendicular views to ameliorate this problem, but depending on the orientations, views can still be non-informative. Finally, given the need to maintain passenger throughput, evaluation of a particular scan should not take excessively long.

For the aforementioned reasons, an automatic threat detection algorithm to aid human operators in locating prohibited items would be useful for the TSA, especially if it can be readily integrated into the existing fleet of deployed scanners. Fundamentally, the TSOs both localize and identify dangerous items in an image, which are the same objectives of *object detection* [GDDM14, Gir15, RHGS15, LAE+16, DLHS16, HRS+17]. Object detection has long been considered a challenging task for computers, but advances in deep learning [GBC16] in recent years have resulted in enormous progress. Specifically, CNNs [LBD+89] have proven extremely useful at extracting learned features for a wide variety of computer vision tasks, including object detection. As a result, the TSA is interested in assessing the feasibility of deploying algorithms that can automatically highlight objects of interest to TSOs [TSA17].

Most deep learning methods require a large training dataset of labeled examples to achieve good performance [SSSG17]; for object detection, this means data comprising both images and bounding boxes with class labels. While many such datasets exist for Red-Green-Blue (RGB) natural scenes (*e.g.* [EVW+10, LMB+14, COR+16]), none contain threats in X-ray luggage, and so a sizable data collection effort was necessary for this endeavor. We assembled a large variety of cluttered bags (*e.g.* clothing, electronics, etc.) with hidden threats (firearms, sharps, blunts, LAGs), and scanned these with the Rapiscan 620DV. Each threat in the scans was then annotated with a tight bounding box and labeled according to class. This dataset was then used for training and evaluating object detection models.

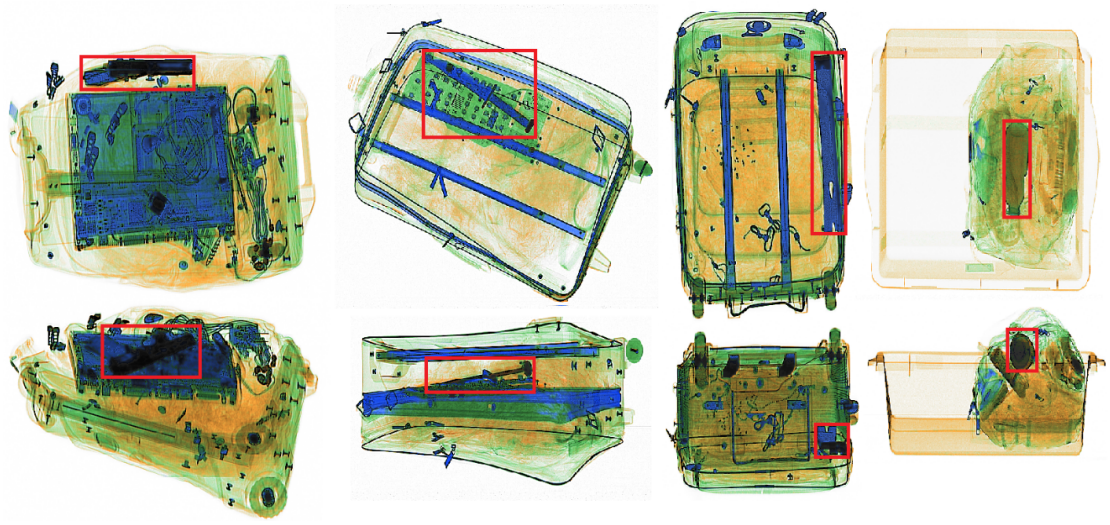


Figure 3.1: Example scans of bags in false color containing a firearm (handgun), sharp (knife), blunt (crow bar), and LAG (bottle of liquid), from the left to right, with (top row) top view and (bottom row) side view shown. Ground truth locations of threats in each image are bounded with a red box. Scans produced by a laboratory prototype not in TSA configuration.

In this work, we present the results of a research effort in collaboration with the TSA to develop a deep learning-based automated threat detection system. We first describe the Rapiscan 620DV scanner and the data collection process. We then introduce the deep learning algorithms we used to perform object detection and how we integrated them into a Rapiscan 620DV prototype, for live testing. Finally, we present experimental results on a number of models we tested on the collected data. The resulting prototype system has shown great promise, and technology like this may one day be deployed by the TSA to airports nationally.

3.2 Data Collection

3.2.1 Rapiscan 620DV X-Ray Scanning System

The Rapiscan 620DV X-ray screening system is designed for aviation and high-security applications. It comprises a tunnel 640 mm wide and 430 mm high, equipped

with a 160 kV / 1 mA X-ray source that achieves a steel penetration of 33mm and a wire resolution of about 80 micrometers (40 American Wire Gauge). The scanner produces two views through the near-orthogonal orientation of the fan-shaped beams from the X-ray sources. These projections generate a horizontal and vertical view of the object under inspection, both of which can be used to identify the contents of a bag. X-ray detectors collect both high and low X-ray energy data, which allows for material discrimination. Examples are shown in Figure 3.1.

While it is possible to use the high and low energy image scans as direct inputs to our model, we instead choose to use the pre-processed RGB coloration typically shown to human TSOs. This coloring uses the relationship between the linear attenuation coefficient and photon energy to estimate effective atomic number (Z), transforming the image into one where material properties can be more readily inferred: for example, organic materials tend to have low Z, while metallic materials tend to have higher Z. According to Rapiscan's proprietary coloring scheme, metallic objects are colored blue, organic materials are tinted orange, and materials with effective Z (Z_{eff}) between these two are shaded green. Using this false coloring as our input achieves two objectives: (i) encoding of additional human knowledge of material properties, which are highly informative for threat detection (firearms, sharps, and blunts, for example, often contain metallic components) and (ii) aligning the image input color distribution closer to the pre-trained weights, which were trained on RGB natural scenes.

3.2.2 Scan Collection and Annotation

Baggage scans were collected at various sites, occurring over multiple collection events. This data collection targeted several of the TSA's designated threat categories: firearms (*e.g.* pistols), sharps (*e.g.* knives), blunts (*e.g.* hammers), and

Table 3.1: Total number of unique threat items and number of images collected for each threat.

Threat Type	Total Threats	Total Images
Blunts	10	3366
Firearms	43 (assembled) + 19 (disassembled)	3480
LAGs	70	3456
Sharps	40	3484

LAGs (*e.g.* liquid-filled bottles). A diverse set of unique items from each class were selected to provide coverage for each threat type; for example, the firearms set included both assembled and disassembled guns. To simulate the diversity of real-world traffic, a variety of host bags was used, including roller, laptop, and duffel bags. Each was filled with diverse assortments of benign items, such as clothing, shoes, electronics, hygiene products, and paper products. Threats were added to each host bag in different locations and orientations, as well as with imaginative concealments, to simulate the actions of potentially malicious actors. Under the assumption that threat objects are typically rare, most bags contained only one threat, as in the examples shown in Figure 3.1.

Given the time-consuming nature of assembling bags for scanning, a single bag was used to host different unique threats for multiple scans, with a minor exchanging of benign clutter between insertions. Each bag was also scanned in several different poses (*e.g.* flipped or rotated). These strategies allow for more efficient collection of more scans and encourage our models to learn invariance to exact positioning within the tunnel. Total number of threats scanned are summarized in Table 3.1.

After the scans were collected, each image was hand-annotated by human labelers, where each label consisted of both the threat class-type, as well as the coordinates of the bounding box. Each box was specified to be as tight as possible in each view, while still containing the full object; in the case of objects like sharps and blunts,

this definition included the handle, for instances in which there was one. In total, the entire data collection effort of assembling, scanning, and labeling bags took over 400 worker hours.

3.3 Methods

3.3.1 Convolutional Neural Networks

The advent of deep convolutional neural networks (CNNs) [LBD+89] has resulted in a quantum leap in the field of computer vision. Across virtually all computer vision tasks, the incorporation of CNNs into model designs has resulted in significant performance gains; consequently, CNNs play a significant role in almost every recent computer vision algorithm. Unlike classical methods that rely upon carefully selected, human-engineered features, machine learning (and deep learning) methods learn these features from the data itself. CNNs in particular are designed to learn hierarchical representations [ZF14], resulting in a feature extractor that produces highly informative, abstract encodings that can be used for downstream tasks, such as classification [KSH12]. Additionally, the learned visual features are highly transferable: for example, CNN weights learned for the classification task of ImageNet [DDS+09] can serve as a good initialization for other datasets or even other related computer vision tasks [YCBL14, RASC14, GDDM14]. Doing so can considerably reduce the number of training examples needed for the desired task. In the setting of automatic threat detection at TSA checkpoints, this is especially significant, as we must assemble, scan, and label each training sample ourselves; pre-trained networks allow us to significantly cut down man-hours and costs.

There are several design considerations for CNNs. Most obvious is model performance: how good are the features the CNN extracts for the downstream task? In general, there is a positive correlation between the number of CNN layers (depth) and

Table 3.2: ImageNet classification accuracy and number of parameters for each of the CNN architectures considered in our experiments. Adapted from [HRS+17].

CNN Architecture	Top-1 Accuracy	Number of parameters
Inception V2 [IS15]	73.9	10.2 M
ResNet-101 [HZRS16]	77.0	42.6 M
ResNet-152 [HZRS16]	77.8	58.1 M
Inception ResNet V2 [SIVA16]	80.4	54.3 M

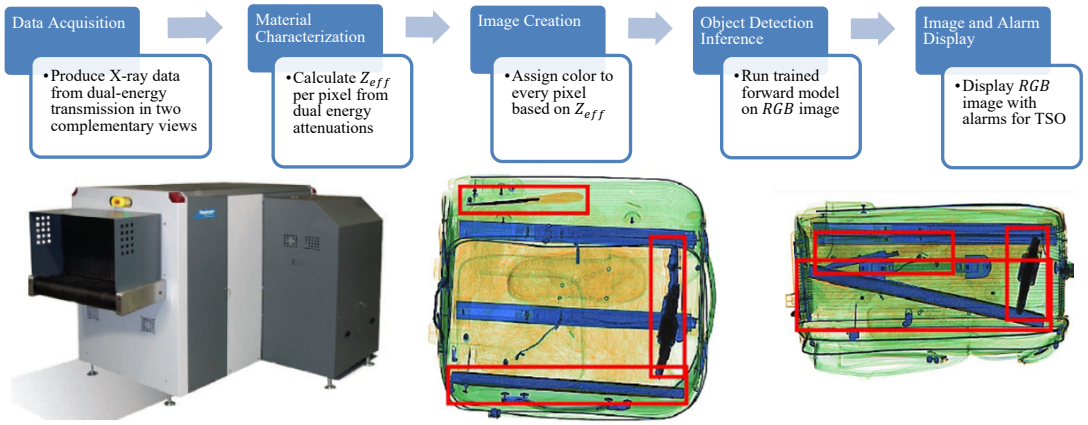


Figure 3.2: Diagram of the prototype Rapiscan 620DV X-ray screening system with threat recognition capability. Dual-energy X-Ray information yields false-color images of two views, which are fed to a trained deep convolutional object detector. Detections above threshold are displayed for the user. Scans produced by a laboratory prototype not in TSA configuration.

parameters with overall performance [SZ15, HZRS16], though architectural choices can play a significant role as well [ZVSL18]. However, finite hardware memory and processing time limit model size. We consider several popular CNN architectures in our experiments, summarized in Table 3.2.

3.3.2 Object Detection

Localizing and classifying objects in a scene is a canonical research area in computer vision. In this context, localization refers to the production of a *bounding box* which is as tight as possible while still containing the entire object, while classification is

the identification of which of a pre-determined set of classes the object belongs to. Formally, given an image X, the goal of object detection is to predict the class c_i of each object indexed by i, as well as the center and dimensions (x_i, y_i, w_i, h_i) of a bounding box.

Modern object detectors are almost exclusively built upon CNN backbones. The specific CNN architecture used is often readily interchangeable, with the choice of CNN depending on the trade-off between accuracy with speed and memory. How predictions are made from the features extracted by the CNN can vary, and various object detection *meta-architectures* [HRS+17] have been recently proposed, of which we highlight two notable ones here.

Faster R-CNN: Faster R-CNN [RHGS15] makes predictions in a two-stage process. In first stage, called the Region Proposal Network (RPN), a set of reference boxes of various sizes and dimensions (termed *anchor boxes*) are tiled over the entire image. Using features extracted by a CNN, the RPN assigns an "objectness" score to each anchor based on how much it overlaps with a ground-truth object, as well as a proposal of how each anchor box should be adjusted to better bound the object. The N_p box proposals with highest objectness scores are then passed to the second stage, where N_p is a hyperparameter controlling the number of proposals. In the second stage, a classifier and box refinement regressor yield final output predictions. Non-maximal suppression reduces duplicate detections.

SSD: Unlike Faster R-CNN, which performs classification and bounding box regression twice, Single-stage detectors like SSD [LAE+16] combine both stages. This eliminates the proposal stage to directly predict both classes and bounding boxes at once. This reduction tends to make the network much faster, though sometimes at the cost of accuracy.

Table 3.3: Inference speed and mAP of the considered feature extractor and meta-architecture combinations. Timing measured on a Nvidia GeForce GTX 1080 Graphical Processing Unit (GPU).

Model	Speed (s/scan)	mAP	Sharps	Blunts	Firearms	LAGs
SSD-InceptionV2	0.042	0.7523	0.408	0.918	0.757	0.907
Faster-RCNN-ResNet101	0.222	0.9166	0.766	0.976	0.944	0.973
Faster-RCNN-ResNet152	0.254	0.9244	0.786	0.980	0.947	0.976
Faster-RCNN-InceptionResNetV2	0.812	0.9410	0.818	0.983	0.962	0.985

3.3.3 Evaluation

The two-part nature of the object detection task–localization and classification–requires evaluation metrics that assess both aspects of detections. The quality of an algorithm-produced predicted box (B_p) with a ground-truth bounding box (B_{gt}) is formalized as the IoU = area($B_p \cap B_{gt}$)/area($B_p \cup B_{gt}$).

A T_p, F_p, and F_n are defined in terms of the IoU of a predicted box with a ground-truth box, as well as the class prediction. A true positive proposal is a correctly classified box that has an IoU above a set threshold (*e.g.* 0.5), a false positive proposal either misclassifies an object or does not achieve a sufficiently high IoU, and a false negative is a ground-truth object that was not properly bounded (with respect to IoU) and correctly classified.

At a particular IoU threshold, the *precision* and *recall* of the model may be computed as the proportion of proposed bounding boxes that are correct and the proportion of ground truth objects that are correctly detected, respectively. These quantities are: Precision = $T_p/(T_p+F_p)$, Recall = $T_p/(T_p+F_n)$. Precision-recall (PR) curves are constructed by plotting both quantities over a range of operating point thresholds. We present these curves in Section 3.5 to provide a sense for model performance. Additionally, we may quantitatively summarize model performance through mAP. AP is the AUC of the PR curve for a single class, and mAP is the mean of the APs across all classes.

3.3.4 Rapiscan 620DV Integration

In order to take a concrete step towards the TSA's goal of potentially deploying the deep learning-based automated threat detector, we also worked to integrate the algorithm with the Rapiscan 620DV. The Rapiscan 620DV has an onboard computer and monitors to construct and display images from the output of the X-ray photon detectors, as well as algorithms for explosives detection. We wish to leave these functionalities untouched, simply overlaying an additional detection output on screen. Therefore, we pipe the constructed scan images to our model, perform inference, and project the predictions to the display (see Figure 3.2).

To achieve threat recognition, we export a trained model and run it in parallel with existing software. The system computer hardware was upgraded to an Intel i7 CPU and a Nvidia GeForce GTX 1080 GPU in order to support the TensorFlow [AAB+15] implementation of the model graph. This allows for a single integrated machine to perform all of the computation for the 620DV, unlike previous implementations that require an additional auxiliary machine to perform the deep neural network computation [LHG+18]. While the resulting integrated system has been used for live demos, the experimental results we report in this paper were computed with a held-out test set.

3.4 Related Work

The development of computer-aided screening for aviation security has garnered much attention in the past two decades. We focus here specifically on efforts to locate and classify potential threats in X-ray images.

Initial work using machine learning to classify objects in X-ray images leveraged hand-crafted features fed to a traditional classifier such as a Support Vector Machine (SVM). In particular, [BYB11] used Bag-of-Visual-Words (BoVW) and an SVM to

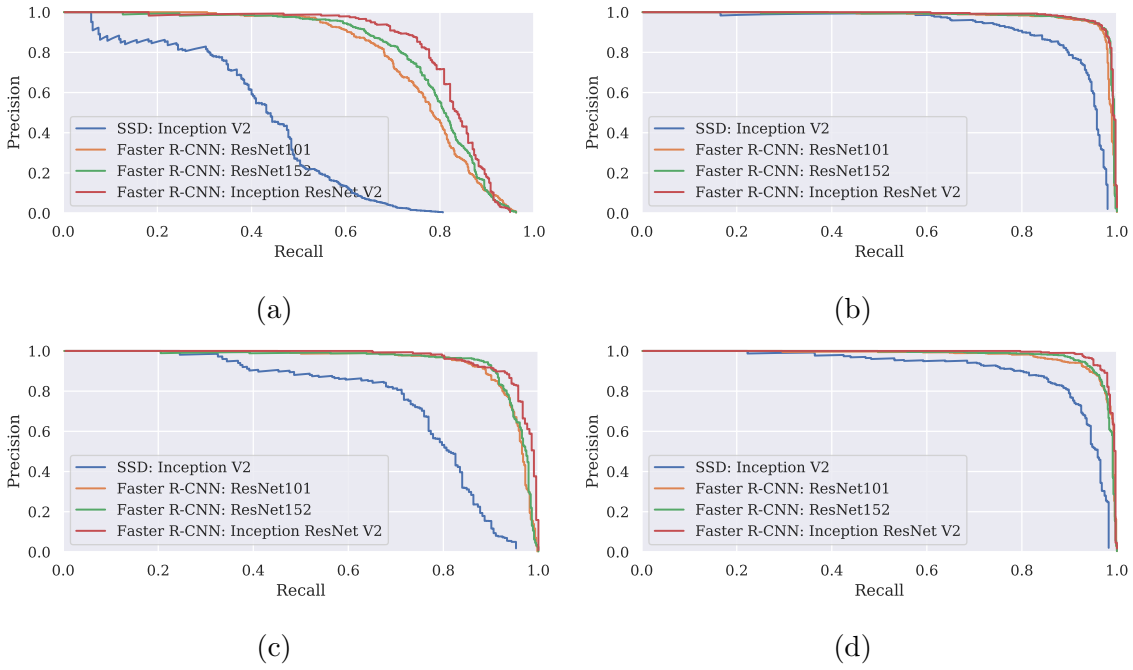


Figure 3.3: PR curves for four meta-architecture/feature extractor combinations. (a) Sharps, (b) Blunts (c) Firearms (d) LAGs.

classify X-ray baggage with feature representations such as Difference of Gaussians (DoG) in conjuction with scale-invariant feature transform (SIFT) [Low99]. Further BoVW approaches are used for classification in [TMB13], [BBB13], [MSA16], [KAD+16].

While deep learning has been applied to general image analysis for at least a decade, its adoption for X-ray security image analysis is relatively recent. Still, there are several works that apply deep learning to baggage screening. In [RJMG17], the authors provide a review of methods for automating X-ray image analysis for cargo and baggage security, pointing to the use of CNNs as a promising direction. The first application of deep learning to an X-ray baggage screening context was for classifying manually cropped regions of X-ray baggage images that contained different classes of firearms and knives, with additional benign classes of camera and laptop [AKDB16]. To perform classification, [AKDB16] fine-tuned a pre-trained

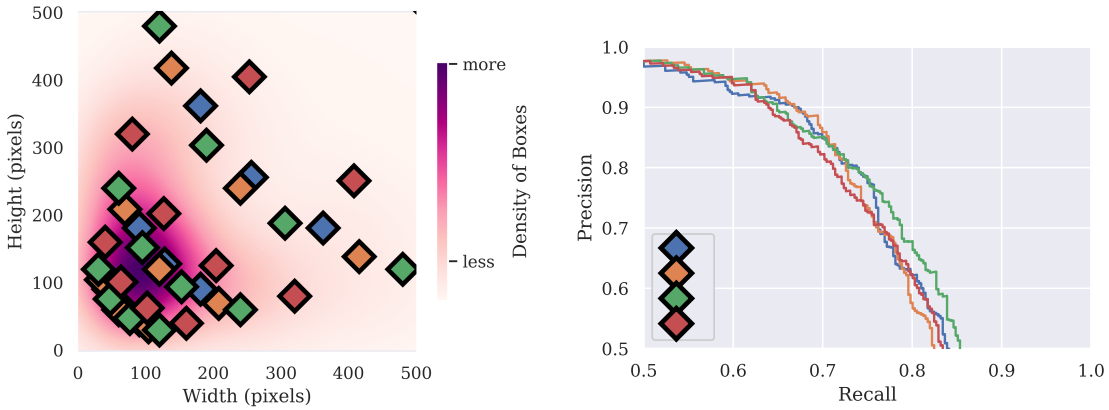
CNN to their unique datasets, leveraging transfer learning to improve training with a limited number of images compared to the size of datasets that CNNs are typically trained on. In [AKDB16], the authors compare their classification performance to the BoVW methods mentioned above.

The work of [AKDB16] is extended in [AB17, AKWB18] to examine the use of deep object detection algorithms for X-ray baggage scans. The authors address two related problems: binary identification of objects as firearms or not and a multiclass problem using the same classes as [AKDB16]. They expand the CNN classification architectures investigated to include VGG [SZ15] and ResNet [HZRS16], and they further adapt Faster R-CNN [RHGS15], R-FCN [DLHS16], and YOLOv2 [RF17] as CNN-based detection methods to X-ray baggage. However, these experiments were done in simulation on pre-collected datasets, without any integration into the scanner hardware. They also do not take advantage of the X-ray scanner's multiple views.

Concurrent with this work, the TSA has sought to incorporate deep learning systems at U.S. airport security checkpoints in other efforts. In [LHG+18], the authors present data collection efforts for firearms and sharps classes and compare the performance of five object detection models. Relative to [LHG+18], we also include blunt weapons and LAGs categories, and we train a single four-class detector, rather than training an individual detector for each category.

3.5 Experiments

For our experiments and in-system implementation, we use Google's code base [HRS+17] of object detection models, implemented in TensorFlow [AAB+15]. We initialize each model with pre-trained weights from the MSCOCO Object Detection Challenge [LMB+14] and then fine-tune them to detect each of the target classes (firearms, sharps, blunts, LAGs) simultaneously, which allows us to perform



(a) (b)

Figure 3.4: (a) Heatmap indicating density of bounding box dimensions for the training set. Various anchor box distributions are shown; each color indicates a different anchor box experimental setting. Natural image defaults in blue. (b) Precision-recall curves for default anchor boxes and engineered set on sharps. Colors correspond to the distributions shown in (a).

detection four times as fast as if we trained a separate algorithm for each. Since we initialize with weights pre-trained on MSCOCO, we pre-process each image by subtracting from each pixel the channel-means of the MSCOCO dataset; this aligns our pre-processing with that performed on images for the MSCOCO Challenge.

For all Faster R-CNN algorithms, we use a momentum optimizer [Qia99] with a learning rate of 0.003 for 130,000 steps, reducing it by a factor of 10 for 40,000 steps, and reducing by another factor of 10 for a final 30,000 steps. For the SSD model, we used 200,000 steps of an RMSprop optimizer [HSS12] with an exponential decay learning rate starting at 0.003, and decaying by 0.9 every 4000 steps. During training, a batch size of 1 was used for all Faster R-CNN models, and a batch size of 24 was used for SSD.

From the 13,786 images collected, we create a 70/10/20 train-validation-test split, which we use for all experiments. We take care to ensure the two images (views) of a particular bag remain in the same split.

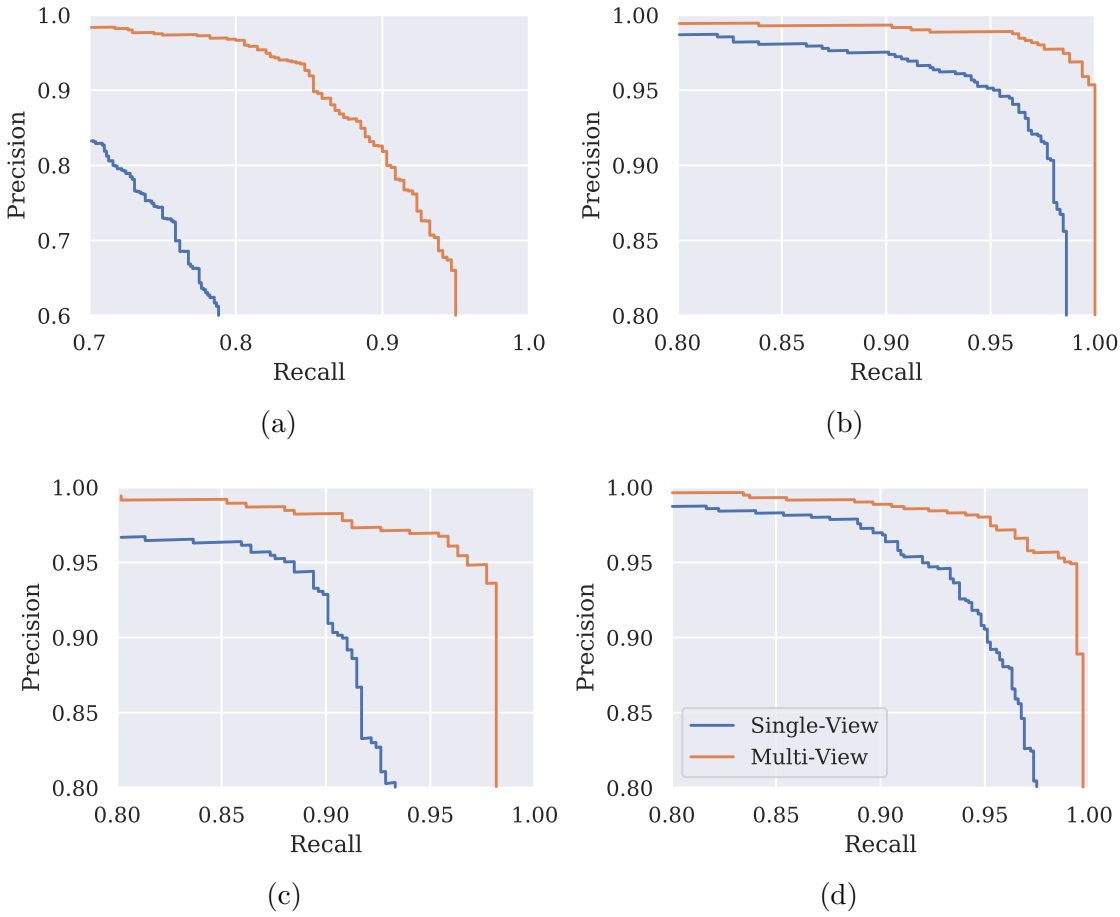


Figure 3.5: Precision-recall comparison of single view (blue) versus with multi-view (orange) detection for (a) sharps (c) blunts (c) firearms (d) LAGs . Note that the multi-view graphs shown here are a choice of analysis, and not a different technique. The training and inference of Faster R-CNN are the same in both traces.

3.5.1 Feature Extractor and Meta-architecture

As discussed in Section 3.3, there are many options for both CNN feature extractor and the object detection meta-architecture, each with its own advantages and disadvantages. See [HRS+17] for extensive comparisons on MS COCO [LMB+14]. For the collected X-ray scan dataset, we choose to analyze several high-performing combinations. Detection performance is measured in terms of AP for each of the classes of interest, and mAP for overall performance is also calculated. We also measure

processing time per scan to project practical passenger wait times.

We summarize the results in Table 3.3 and Figure 3.3. Overall, Faster R-CNN with Inception ResNet V2 has the highest mAP, while SSD with Inception V2 performed the worst. In general, faster models are less accurate, which may be seen in the "Speed" column of Table 3.3. Faster R-CNN with the two smaller feature extractors (ResNet101 and ResNet152) achieve nearly the same performance on sharps as ResNet Inception V2, but at more than three times the speed. While the speed of single-stage models is suitable for video frame rates, we found this to be unnecessary for checkpoint threat recognition and to sacrifice too much accuracy.

3.5.2 Anchor Boxes

As discussed in Section 3.3.2, bounding box predictions are typically made relative to anchor boxes tiled over the image. The object detection algorithms we have considered were primarily designed for finding common objects (*e.g.* people, animals, vehicles) in natural scenes, with datasets like PASCAL VOC [EVW+10] or MS COCO [LMB+14] in mind.

The anchor box distribution is commonly held to act as a kind of "prior" over the training data. In YOLO V2 [RF17], anchors are learned by k-means clustering, and some of the performance gains of this model are credited to this improvement. We chose several configurations of anchor boxes to better match the distribution of our training data, and display those configurations alongside training dataset bounding box dimension density in Figure 3.4a. The dataset used for these experiments was smaller than the dataset used for the main findings as described in Table 3.1. Training, test, and validation sets were drawn from a pool of images containing 2768 Sharps, 1788 LAGs, 1800 Blunts, and 3080 Firearms. This does not impact our conclusions stated in the next paragraph.

Table 3.4: Single View versus Effective Multiple View APs.

Threat	Single View AP	Multiple Views AP
Sharps	0.786	0.935
Blunts	0.980	0.995
Firearms	0.947	0.984
LAGs	0.976	0.994

During model validation, some of these configurations showed modest gains for sharps, but these did not generalize during testing. The sharps class PR curves for the anchor box distributions in 3.4a are shown in 3.4b. We find that performance is robust to different anchor configurations, showing that even with a different box size distribution, Faster R-CNN is able to learn accurate bounding box regressors.

3.6 Discussion

The results we have shown bear implications for a pilot real-world deployment of this technology. In Table 3.3, we showed test AP on sharps and timing for four feature extractor/meta-architecture pairs. In a possible real-world system, we strive for inference rates which would not impact screening time and security checkpoint throughput. Because of the long evaluation time of the Faster R-CNN with InceptionV2 model ($\sim$ 800 ms seconds per bag), we recommend use of Faster R-CNN with ResNet152 ($\sim$ 250 ms per bag) for its performance/speed tradeoff. For the remainder of the Discussion section, we will show results only from this model.

3.6.1 Multiple View Redundancy

Unlike typical object detection research benchmarks, the Rapiscan 620DV provides two views along nearly perpendicular axes of the same scanned object. Within the context of threat detection in X-ray images, this is especially important, as individual views may occasionally be uninformative due to perspective or clutter. Leveraging

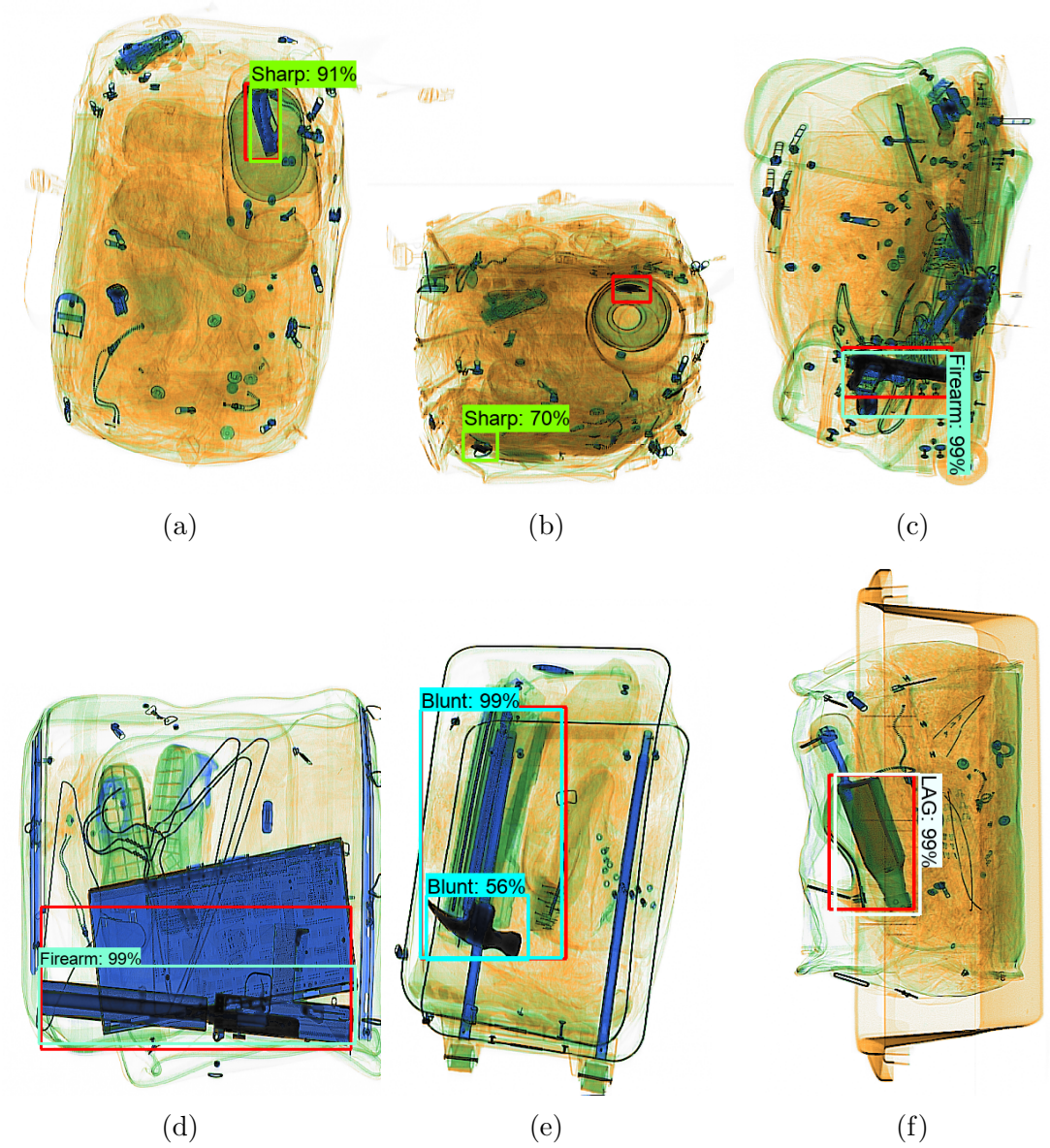


Figure 3.6: Sample detections from the Faster R-CNN model with ResNet152 as the feature extractor. Ground truth bounding boxes are shown in red. (a-b) Sharp detection. (c-d) Firearm detection. (e) Blunt weapon detection. (f) LAG detection. The color of predicted box and the label indicate the predicted class. Scans produced by a laboratory prototype not in TSA configuration.

the two separate views can improve overall performance.

In order to account for the multiple views, we consider a true positive in any view to be a true positive in all views. False positives are added independently across all

views. Note that this is not describing a change to the training of the algorithm, nor the inference process. Rather, by performing our analysis in this way, we hope to better represent how the system might work in a potential real-world deployment, when both views are available to a TSO. We show the improvements in PR between single-view and multi-view evaluation in Figure 3.5 and summarize the AP in Table 3.4.

3.6.2 Sample Detections

In Figure 3.6, we display selected detections from the fully trained Faster R-CNN with ResNet152 as the feature extractor, for a number of threat classes.

In Subfigures (3.6a-3.6b), we display two views of the same scan. The very small profile of the folding knife in one view (3.6b) makes detection challenging for the trained object detector (though there is a low-confidence false alarm). However, the knife presents a more clear profile in Subfigure 3.6a, and is detected there. This motivates what we call "multi-view" analysis, which we discuss further in Section 3.6.1.

Subfigure 3.6e shows a blunt threat which is detected twice. The larger detection, which encompasses the head and handle of a hammer, is a True Positive, because the IoU of this detection is greater than 0.5. The other detection in this image, however, only covers the hammer's head. While the presence of the hammer merits an alarm, the detection does not overlap enough with the ground truth, and is therefore a False Positive. Some of the training data included hammer heads disconnected from a handle. It may be harder for the CNN to learn to bound hammers with handles or hammer heads only.

To demonstrate detections of the remaining threat classes Subfigure 3.6f shows a detected LAG, and Subfigures 3.6c and 3.6d show scans with firearms. Note that the

machine pistol in Subfigure 3.6d is not as well localized, compared to the firearm in 3.6c, likely due to the obscuring presence of a laptop. However, such an alarm still makes the threat readily visible to a human operator.

3.7 Conclusions

We have investigated use of state-of-the-art techniques for the challenging task of threat detection in bags at airport security checkpoints. First, we collected a significant amount of data, assembling by hand many bags and bins which simulate everyday traffic. These concealed a wide variety of threats. We scanned each bag to produce X-ray images, and annotated both views of the scan. We then trained multiple modern object detection algorithms on the collected data, exploring a number of settings and engineering them for the task at hand. We have presented the results of evaluating the model on held-out validation and test data.

In general, we do not find single stage methods to be accurate enough as a security screening method, and their frame rate advantages are superfluous in this application. There are variants of the Faster R-CNN which can run on commercially available computer hardware, and still achieve accurate threat recognition.

In addition to the evaluation presented in Section 3.5, the TSA has also tested prototype Rapiscan 620DV systems with directly integrated trained models. These results have shown the promise of deep learning methods for automatic threat recognition. Further, they illustrate that the TSA, using X-ray scanners such as the Rapiscan 620DV, has the capability to bring these new technologies to airport checkpoints in the near future.

Chapter 4

Object Detection as a Positive-Unlabeled Problem

4.1 Introduction

The performance of supervised deep learning models is often highly dependent on the quality of the labels they are trained on [ZBB+17, VAC+17, JZL+18]. Recent work [TSdC+19] has implied the existence of "support vectors" in deep learning datasets: hard to classify examples that have an especially significant influence on a classifier's decision boundary. As such, ensuring that these difficult examples have the correct label would appear to be important to the final classifier.

Collecting completely accurate labels for object detection [GDDM14, Gir15, LAE+16, RHGS15, RDGF16, DLHS16, RF17], however, can be challenging, much more so than it is for classification data. Unlike the latter, where there is a single label per image, the number of objects in an image is often variable, and objects can come in a large variety of shapes, sizes, poses, and settings, even within the same class. Worse, object detection scenes are often crowded, resulting in object instances that may be occluded. Given the requirement for tight bounding boxes and the sheer number of instances to label, constituting annotations can be very time-consuming. For example, just labeling instances, without localization, required ∼30K worker hours for the 328K images of MS COCO [LMB+14], and the airport checkpoint X-ray dataset used in [LHG+18], which required assembling bags, scanning, and hand labeling, took well over 250 person hours for 4000 scans over the span of several months. For medical datasets [MAD+12, YWLS17, LGH+17, WPL+17],

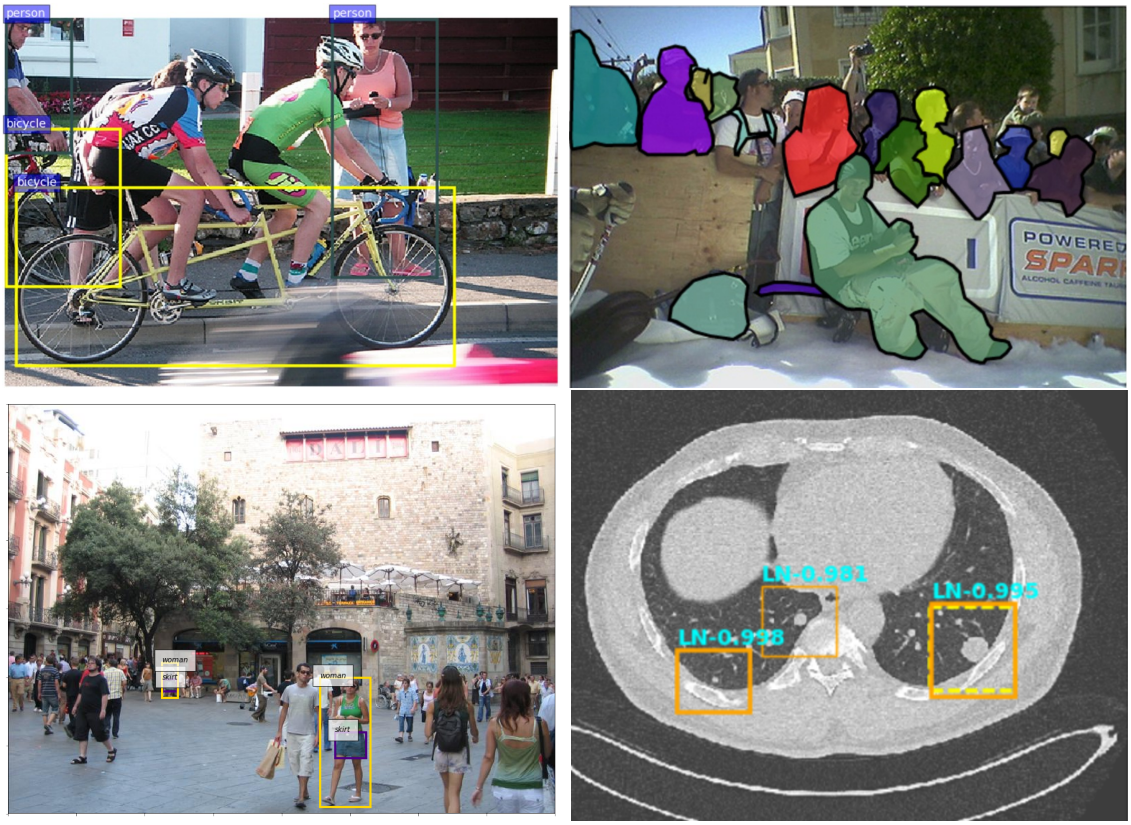


Figure 4.1: Because of inter- and intra-annotator inconsistencies and the inherent difficulty of instance labeling, the ground truth of object detection datasets can be incomplete. Example images and their ground truth labels shown for (clockwise from top left) PASCAL VOC [EVW+10] (missing people and bottles), MS COCO [LMB+14] (missing people), DeepLesion [YWLS17] (ground truth is the dotted line; two boxes on the left indicate two unlabeled nodules), and Visual Genome [KZG+17] (missing people, tree, clothing, etc.).

this becomes even more problematic, as highly trained (and expensive) radiologist experts or potentially invasive biopsies are needed to determine ground truth.

As a result of its time-consuming nature, dataset annotations are often crowdsourced when possible, either with specialized domain experts or Amazon's Mechanical Turk, significantly speeding up the data annotation process. In order to ensure consistency, dataset designers establish labeling guidelines and/or have multiple workers label the same image [EVW+10, LMB+14]. Regardless, tough judgment-call instances, inter- and even intra-worker variability, and human error can still result

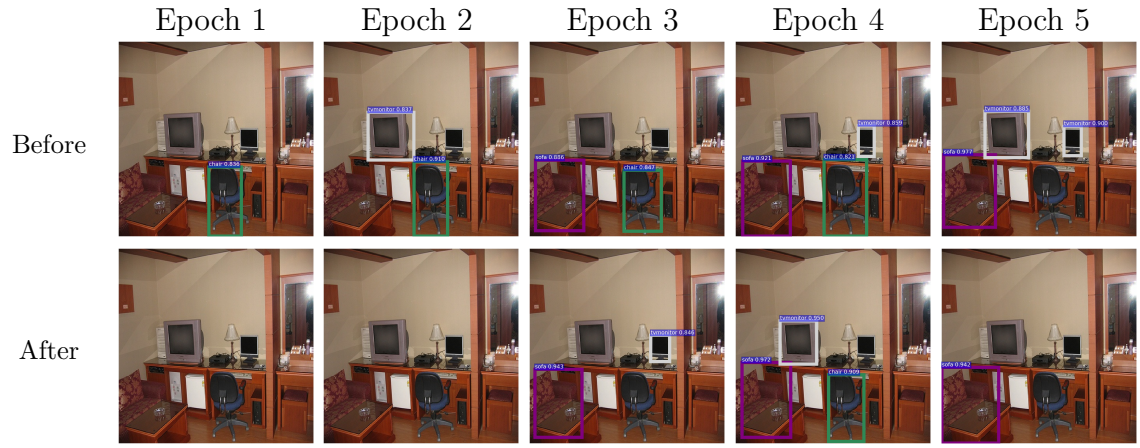


Figure 4.2: Detections on a PASCAL VOC train set image missing annotations throughout training: only the sofa in the lower left has a label. Each column shows detections directly before (top) and after (bottom) the model is trained on the image shown, for each epoch. While the sofa is consistently detected (purple box) after being learned, the unlabeled objects (2 monitors, a chair) are repeatedly found and then suppressed after being trained upon.

in overall inconsistency in labeling, or missing instances entirely. This becomes especially exacerbated when trying to form a larger dataset, like OpenImages [KRA+18], which while extremely large, is incompletely labeled.

On the other hand, object detection algorithms often use the standard cross-entropy loss for object classification. As a result, implicit to this loss function is the assumption that any region without a bounding box does not contain an object; in other words, classification is posed as a positive-negative (PN) learning problem. While such an assumption may be reasonable for an appropriately accurate ground truth for each image, despite best efforts, this is often not the case in practice due to the previously outlined difficulties of instance annotation. As shown in Figure 4.1 for a wide array of common datasets, the lack of instance label does not always mean the absence of a true object.

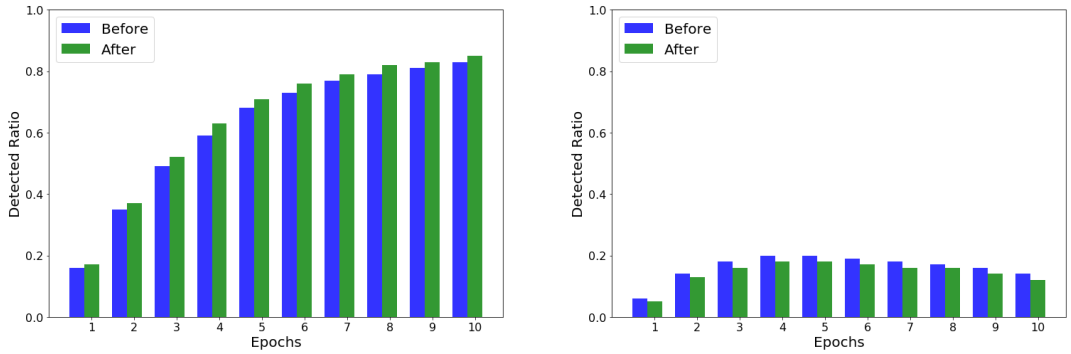
While the result of this characterization constitutes a noisy label setting, it is not noisy in the same respect as is commonly considered for classification problems

[ZBB+17, VAC+17, JZL+18]. The presence of a positive label in object detection datasets are generally correct with high probability; it is the *lack* of a label that should not be interpreted with confidence as a negative (or background) region. Thus, given these characteristics common to object detection data, we propose recasting object detection as a positive-unlabeled (PU) learning problem [Den98, DDGL99, LDG00, EN08, KNDS17]. With such a perspective, existing labels still implies a positive sample, but the lack of one no longer enforces that the region must be negative. This can mitigate the confusing learning signal that often occurs when training on object detection datasets.

In this work, we explore how the characteristics of object detection annotation lend themselves to a PU learning problem and demonstrate the efficacy of adapting detection model training objectives accordingly. We first illustrate with an empirical study the confusing effect missing labels have on the training process. We then perform a series of experiments to demonstrate the effectiveness of the PU objective on two popular, well-labeled object detection datasets (PASCAL VOC [EVW+10] and MS COCO [LMB+14]) across a range of label missingness, as well as two datasets with real incomplete labels (Visual Genome [KZG+17], DeepLesion [YWLS17]).

4.2 Example Forgetting in Object Detection

In a recent study of training dynamics of neural network classifiers, the authors of [TSdC+19] defined a "forgetting event" as a training example switching from being classified correctly by the model to being classified incorrectly during training. It was found that certain examples were forgotten more frequently than others while others were never forgotten (termed "unforgettable"), with the degree of forgetting for individual examples being consistent across neural network architectures and random seeds. When visualized, the forgotten examples tend to have atypical or uncommon



(a) Labeled objects (b) Unlabeled objects

Figure 4.3: Detection rates of objects before and after training on their corresponding images for (a) labeled instances and (b) instances with labels withheld during training.

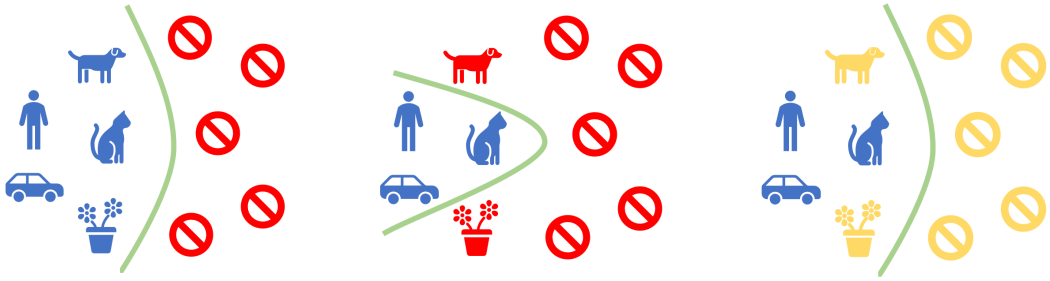
characteristics (e.g., pose, lighting, angle), relative to "unforgettable" examples. Interestingly, a significant number of "unforgettable" examples could be removed from the training set with only a marginal reduction in test accuracy, if the "hard" examples were kept. This implies that the "hard" examples play a role akin to support vectors in max-margin learning, while easier "unforgettable" examples have little effect on the final decision boundary.

Within the context of object detection datasets, we hypothesize that unlabeled object instances form a similar group of hard examples that are also learned and then forgotten throughout training. Unlike the inter-batch catastrophic forgetting in [TSdC+19], however, where hard examples are learned while part of the current minibatch and then forgotten while learning other examples, unlabeled samples in object detection are learned from other examples and then *suppressed* after incurring misclassification losses during training (see Figure 4.2). Unlabeled instances strongly resemble positive examples throughout the rest of the dataset and indeed should be considered as such, but their lack of labels mean that the typical PN classification objective incentivizes learning them as negatives. Given that hard examples have a strong influence on classifier boundaries, having unlabeled examples trained as

negatives may prove especially detrimental to training.

We perform a similar study as [TSdC+19] and investigate forgetting events on PASCAL VOC [EVW+10] by tracking detection rates of labeled and unlabeled instances in the training set throughout learning. In particular, an object is considered detected if the detector produces a bounding box with intersection over union (IoU) of at least 0.5 and the classifier is at least 80% confident in the correct class. We track whether or not an object was detected directly before the image it belongs to is trained upon, and then again after the gradients have been applied. These indicator variables are then combined across objects for each epoch and reported as a percentage. While PASCAL VOC does naturally have unlabeled instances, we do not have access to these without a re-labeling effort. As such, we remove 10% of object annotations during training, but use them to calculate detection rates for this experiment.

Detection rates for labeled and unlabeled objects over time are shown in Figure 4.3. As is expected, the model learns to detect a higher percentage of labeled instances over time, and objects are overall more likely to be detected immediately after the detector trains on them. Despite not having an explicit learning signal, unlabeled objects are still learned throughout training, but at a lower rate than labeled ones. In contrast with labeled objects, unlabeled object detections are discouraged with each PN gradient, leading to a dip in overall detection rates immediately after training. Despite this, overall detection rates of unlabeled objects grows through the first 5 epochs of training, implying a repeated cycle of learning unlabeled objects from other intra-class examples, forgetting them when explicitly trained against them, and then learning them again. Given the undesirability of this forced suppression of detected objects, we seek a method to remedy this behavior.



(a) Positive-Negative, assuming full labels

(b) Positive-Negative, with missing labels

(c) Positive-Unlabeled, with missing labels

Figure 4.4: A classifier (green) learns to separate proposals by "objectness". Models trained with a standard cross-entropy loss implicitly assume positive-negative (PN) learning: regions with bounding boxes are considered positive (blue), and any other proposed boxes are treated as negative (red). This is reasonable when labels are complete (a), but in reality, object detection datasets are inherently challenging to label, leading to missing annotations; this forces the classifier to exclude unlabeled objects from the positive class (b). We propose a positive-unlabeled (PU) approach (c), which considers non-labeled regions as unlabeled (yellow) rather than negative, allowing non-positive regions to be classified as positive. Best viewed in color.

4.3 Methods

4.3.1 Faster R-CNN

In principle, the observed problem is characteristic of the data and is thus general to any object detection framework. However, in this work, we primarily focus on Faster R-CNN [RHGS15], a popular 2-stage method for which we provide a quick overview here.

As with other object detection models, given an input image X, the desired output of Faster R-CNN is a bounding box $B^{(i)} \in \mathbb{R}^4$ and class probabilities $c^{(i)} \in \mathbb{R}^k$ for each object (indexed by i) present, where k is the number of classes and the final classification decision is commonly $\operatorname{argmax} c^{(i)}$. Faster R-CNN does this in a 2-stage process. First, a convolutional neural network (CNN) [LBD+89] is used to produce image features h. A Region Proposal Network (RPN) then generates bounding box proposals $\hat{B}^{(i)}$ relative to a set of reference boxes spatially tiled over h. At the same

time, the RPN predicts an "objectness" probability $\hat{c}^{(i)}$ for each proposal, learned as an object-or-not binary classifier. The second stage then takes the proposals with the highest scores, and predicts bounding box refinements to produce $B^{(i)}$ and the final classification probabilities $c^{(i)}$.

Of particular interest is how the classifier producing $\hat{c}^{(i)}$ is trained. Specifically, the cross-entropy loss $H(t, y)$ is employed, where $H(t, y)$ signifies the loss incurred when the model outputs t when the ground truth is y. In the RPN, this results in the following classification risk minimization:

$$R_{pn}^{RPN} = \pi_p \mathbb{E}[H(\hat{c}_p, +1)] + \pi_n \mathbb{E}[H(\hat{c}_n, -1)] \tag{4.1}$$

where π_p and π_n are the class probability priors for the positive and negative classes, respectively, and $\hat{c}_p^{(i)}$ and $\hat{c}_n^{(i)}$ are the predicted "objectness" probabilities for ground truth positive and negative regions. This risk is estimated with samples as:

$$\mathcal{L}_{pn}^{RPN} = \frac{\hat{\pi}_p}{N_p} \sum_{i=1}^{N_p} H(\hat{c}_p^{(i)}, +1) + \frac{\hat{\pi}_n}{N_n} \sum_{i=1}^{N_n} H(\hat{c}_n^{(i)}, -1) \tag{4.2}$$

where N_p and N_n are the number of ground truth positive and negative regions being considered, respectively, and the class priors are typically estimated as $\hat{\pi}_p = \frac{N_p}{N_p+N_n}$ and $\hat{\pi}_m = \frac{N_n}{N_p+N_n}$. Notably, this training loss treats all non-positive regions in an image as negative.

4.3.2 PU Learning

In a typical binary classification problem, input data $X \in \mathbb{R}^d$ are labeled as $Y \in \{\pm 1\}$, resulting in what is commonly termed a positive-negative (PN) problem. This implicitly assumes having samples from both the positive (P) and negative (N) distributions, and that these samples are labeled correctly (Figure 4.4a). However, in some instances, we only know the labels of the positive samples. The remainder of our

data are *unlabeled* (U): samples that could be positive or negative. Such a situation is commonly called a positive-unlabeled (PU) setting, where the N distribution is replaced by an unlabeled (U) distribution (Figure 4.4c). Such a representation admits a classifier that can appropriately include unlabeled positive regions on the correct side of the decision boundary. We briefly review PN and PU risk estimation here.

Let $p(x, y)$ be the underlying joint distribution of (X, Y), $p_p(x) = p(x|Y = +1)$ and $p_n(x) = p(x|Y = -1)$ be the distributions of P and N data, $p(x)$ be the distribution of U data, $\pi_p = p(Y = +1)$ be the positive class-prior probability, and $\pi_n = p(Y = -1) = 1 - \pi_p$ be the negative class-prior probability. In a PN setting, data are sampled from $p_p(x)$ and $p_n(x)$ such that $\mathcal{X}_p = \{x_i^p\}_{i=1}^{N_p} \sim p_p(x)$ and $\mathcal{X}_n = \{x_i^n\}_{i=1}^{N_n} \sim p_n(x)$. Let g be an arbitrary decision function that represents a model. The risk of g can be estimated from $\mathcal{X}_p$ and $\mathcal{X}_n$ as:

$$\hat{R}_{pn}(g) = \pi_p \hat{R}_p^+(g) + \pi_n \hat{R}_n^-(g) \tag{4.3}$$

$\hat{R}_p^+(g) = 1/N_p \sum_{i=1}^{N_p} \ell(g(x_i^p), +1)$ and $\hat{R}_n^-(g) = 1/N_n \sum_{i=1}^{N_n} \ell(g(x_i^n), -1)$, where ℓ is the loss function. In classification, ℓ is commonly the cross-entropy loss $H(t, y)$.

In PU learning, $\mathcal{X}_n$ is unavailable; instead we have unlabeled data $\mathcal{X}_u = \{x_i^u\}_{i=1}^{N_u} \sim p(x)$, where N_u is the number of unlabeled samples. However, the negative class empirical risk $\hat{R}_n^-(g)$ in Equation 4.3 can be approximated indirectly [DNS15, DNS14]. Denoting $R_p^-(g) = \mathbb{E}_p[\ell(g(X), -1)]$ and $R_u^-(g) = \mathbb{E}_{X \sim p(x)}[\ell(g(X), -1)]$, and observing $\pi_n p_n(x) = p(x) - \pi_p p_p(x)$, we can replace the missing term $\pi_n R_n^-(g) = R_u^-(g) - \pi_p R_p^-(g)$. Hence, we express the overall risk without explicit negative data as

$$\hat{R}_{pu}(g) = \pi_p \hat{R}_p^+(g) + \hat{R}_u^-(g) - \pi_p \hat{R}_p^-(g) \tag{4.4}$$

where $\hat{R}_p^-(g) = 1/N_p \sum_{i=1}^{N_p} \ell(g(x_i^p), -1)$ and $\hat{R}_u^-(g) = 1/N_u \sum_{i=1}^{N_u} \ell(g(x_i^u), -1)$.

However, a flexible enough model can overfit the data, leading to the empirical

risk in Equation 4.4 becoming negative. Given that most modern object detectors utilize neural networks, this type of overfitting can pose a significant problem. In [KNDS17], the authors propose a non-negative PU risk estimator to combat this:

$$\hat{R}_{pu}(g) = \pi_p \hat{R}_p^+(g) + \max\{0, \hat{R}_u^-(g) - \pi_p \hat{R}_p^-(g)\} \tag{4.5}$$

We choose to employ this non-negative PU risk estimator for the rest of this work.

4.3.3 PU Learning for Object Detection

PU Object Proposals

In object detection datasets, the ground truth labels represent positive samples. Any regions that do not share sufficient overlap with a ground truth bounding box are typically considered as negative background, but the accuracy of this assumption depends on every object within a training image being labeled, which may not be the case (Figure 4.1). As shown in Figure 4.4b, this results in the possibility of positive regions being proposed that are labeled negative during training, due to a missing ground truth label. Therefore, we posit that object detection more naturally resembles a PU learning problem than PN.

We recognize that two-stage detection naturally contains a binary classification problem in the first stage. In Faster R-CNN specifically, the RPN comprising the first stage assigns an "objectness" score, which is learned with a binary cross-entropy loss (Equation 4.2). As previously noted, the PN nature of this loss can be problematic, so we propose replacing it with a PU formulation. Combining Equations 4.2 and 4.5, we produce the following loss function:

$$\mathcal{L}_{pu}^{RPN} = \frac{\pi_p}{N_p} \sum_{i=1}^{N_p} H(\hat{c}_p^{(i)}, +1) + \max\left\{0, \frac{1}{N_u} \sum_{i=1}^{N_u} H(\hat{c}_u^{(i)}, -1) - \frac{\pi_p}{N_p} \sum_{i=1}^{N_p} H(\hat{c}_p^{(i)}, -1)\right\} \tag{4.6}$$

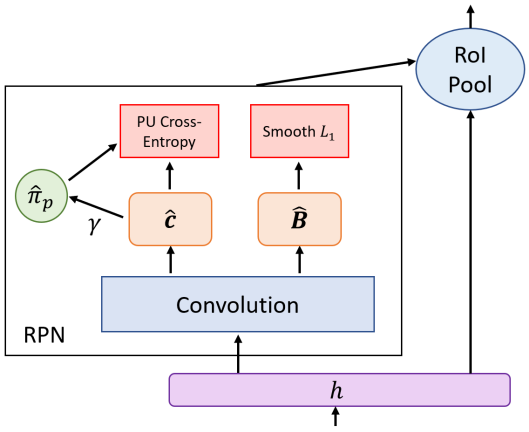
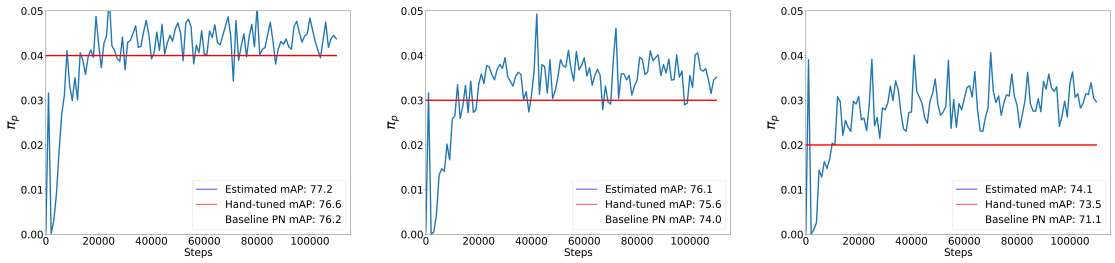


Figure 4.5: Faster R-CNN [RHGS15] Region Proposal Network (RPN) with the proposed positive-unlabeled cross-entropy loss. The estimate of the positive class prior $\hat{\pi}_p$ is updated with the objectness predictions $\hat{c}$, with momentum γ.



(a) $\rho = 0.4$ (b) $\rho = 0.5$ (c) $\rho = 0.6$

Figure 4.6: Positive class prior $\hat{\pi}_p$ estimated during training of Faster R-CNN on PASCAL VOC versus from hand-tuning π_p as a hyperparameter, for instance label missingness proportion $\rho = \{0.4, 0.5, 0.6\}$.

Such a loss function relaxes the penalty of positive predictions for unlabeled objects.

Estimating π_p

The PU cross-entropy loss in Equation 4.6 assumes the class-prior probability of the positive class π_p is known. In practice, this is not usually the case, so π_p must be estimated, denoted as $\hat{\pi}_p$. For object detection, estimating π_p is especially problematic because π_p is not static: as the RPN is trained, an increasing proportion of region proposals will (hopefully) be positive. While [KNDS17] showed some robustness to π_p

misspecification, this was only on a fairly narrow range of $\pi_p \in [0.8\pi_p, 1.2\pi_p]$. During object detection performance, π_p starts from virtually 0 and grows steadily as the RPN improves. As such, any single estimate $\hat{\pi}_p$ poses the risk of being significantly off the mark during a large portion of training.

To address this, we recognize that the RPN of Faster R-CNN is already designed to infer the positive regions of an image, so we count the number of positive regions produced by the RPN and use it as an estimator for π_p:

$$\hat{\pi}_p = \frac{N_p^{RPN}}{N^{RPN}} \tag{4.7}$$

where N^{RPN} is the total number of RPN proposals that are sampled for training, and N_p^{RPN} being those with classifier confidence of at least 0.5. Note that this estimation of π_p comes essentially for free. Given that Faster R-CNN is trained one image at a time and the prevalence of objects varies between images, we maintain an exponential moving average with momentum γ in order to stabilize $\hat{\pi}_p$ (see Figure 4.5). This estimate $\hat{\pi}_p$ is then used in the calculation of the loss $\mathcal{L}_{pu}^{RPN}$ and its gradients.

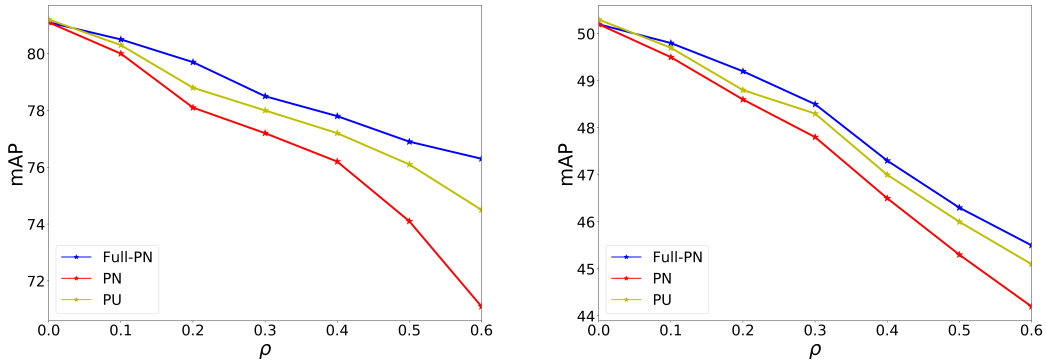
4.4 Related work

Like many machine learning problems, the formulation of most object detection frameworks are designed fully supervised [GDDM14, Gir15, LAE+16, RHGS15, RDGF16, DLHS16, RF17]: it is assumed that there exists a dataset of images where every object is labeled and such a dataset is available to train the model. However, as discussed above, collecting such a dataset can be expensive. Because of this, methods that can learn from partially labeled datasets have been a topic of interest for object detection. What "partially labeled" constitutes can vary, and many types of label missingness have been considered.

Weakly supervised object detection models [BV16, OBLS15, RW15] assume a dataset with image level labels, but not any instance labels. These models are somewhat surprisingly competent at identifying approximate locations of objects in an image without any object specific cues, but have a harder time with providing precise localization. This is especially the case when there are many of the same class of object in close proximity to each other, as individual activations can blur together, and the lack of bounding boxes makes it difficult to learn precise boundaries.

Other approaches consider settings where bounding boxes are available for some classes (e.g., PASCAL VOC's 20 classes) but not others (e.g., ImageNet [DDS+09] classes). LSDA [HGT+14] does this by modifying the final layer of a CNN [KSH12] to recognize classes from both categories, and [TWG+16] improves upon LSDA by taking advantage of visual and semantic similarities between classes. OMNIA [RGBYO18] proposes a method merging datasets that are each fully annotated for their own set of classes, but not each other's.

There are also approaches that consider a single dataset, but the labels are undercomplete across all classes. This setting most resembles what we consider in our paper. In [DZM+18], only 3-4 annotated examples per class are assumed given to start; additional pseudo-labels are generated from the model on progressively more difficult examples as the model improves. Soft-sampling has also been proposed to re-weight gradients of background regions that either have overlap with positive regions or produce high detection scores in a separately trained detector [WBS+19]; experiments were done on PASCAL VOC with a percentage of annotations discarded and on a subset of OpenImages [KRA+18].



(a) PASCAL VOC (b) MS COCO

Figure 4.7: mAP at IoU 0.5 (AP_{50}) on (a) PASCAL VOC and (b) MS COCO, for a range of label missingness ρ.

4.5 Experiments

4.5.1 Hand-tuning Versus Estimation of π_p

As discussed in Section 4.3.3, PU risk estimation requires the prior π_p. We experiment with two ways of determining π_p. In the first method (*Hand-Tuned*), we treat π_p as a constant hyperparameter and tune it by hand. In the second (*Estimated*), we infer π_p from our network as described in Equation 4.7, setting momentum γ to 0.9. We compare the estimate $\hat{\pi}_p$ inferred automatically with the hand-tuned π_p that yielded the highest mAP on PASCAL VOC. To see how our estimate changes in response to label missingness, when assembling our training set, we remove each annotation from an image with probability ρ, giving us a dataset with $1 - \rho$ proportion of the total labels, and then do our comparison for $\rho = \{0.4, 0.5, 0.6\}$ in Figure 4.6.

In all tested settings of ρ, the estimation $\hat{\pi}_p$ increases over time before stabilizing. Such a result matches expectations, as when an object detection model is first initialized, its parameters have yet to learn good values, and thus the true proportion of positive regions π_p is likely to be quite low. As the model trains, its ability to generate accurate regions improves, resulting in a higher proportion of regions being

positive. This in turn results in a higher true value of π_p, which our estimate $\hat{\pi}_p$ follows. As the model converges, π_p (and $\hat{\pi}_p$) stabilizes towards the true prevalence of objects in the dataset relative to background regions. Interestingly, the final value of $\hat{\pi}_p$ settles close to the value of π_p found by treating the positive class prior as a static hyperparameter, but consistently above it. We hypothesize that this is due to a single static value having to hedge against the early stages of training, when π_p is lower.

We use our proposed method of auto-inferring π_p for the rest of our experiments, with $\gamma = 0.9$, rather than hand-tuning it as a hyperparameter.

4.5.2 PU Versus PN on PASCAL VOC and MS COCO

We investigate the effect that incomplete labels have on object detection training for the popular datasets PASCAL VOC [EVW+10] and MS COCO [LMB+14], using Faster R-CNN [RHGS15] with a ResNet101 [HZRS16] convolutional feature extractor. In order to quantify the effect of missing labels, we artificially discard a proportion ρ of the annotations. We compare three settings, each for a range of values of ρ. Given that the annotations are the source of the learning signal, we keep the number of total instances constant between settings for each ρ as follows:

- *PN*: We remove a proportion of labels from every image in the dataset, such that the total proportion of removed labels is equal to ρ, and all images are included in the training set. We then train the detection model with a PN objective, as is normal.

- *Full-PN*: We discard a proportion ρ of entire images and their labels, resulting in a dataset of fewer images, but each of which retains its complete annotations.

- *PU*: We use the same images and labels as in *PN*, but instead train with our

Table 4.1: Detector performance on Visual Genome, with full labels, at various IoU thresholds.

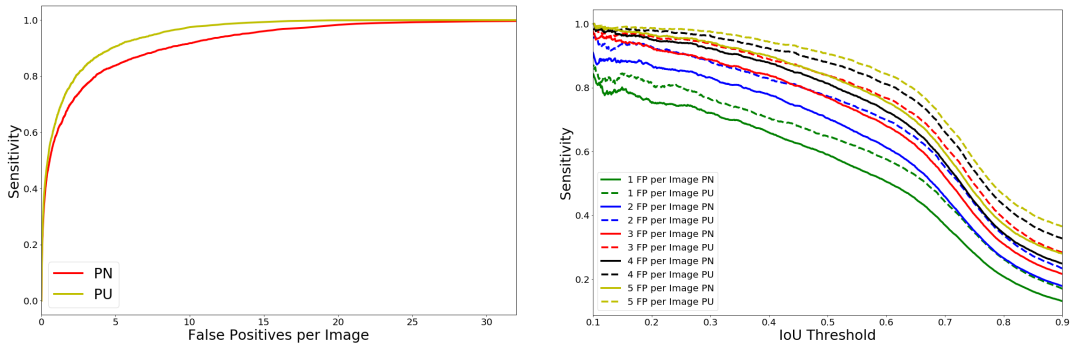
	AP_{25}		AP_{50}		AP_{75}	
Weighted?	Y	N	Y	N	Y	N
PN	12.09	22.79	9.11	17.35	2.46	9.98
PU	**13.83**	**25.56**	**10.44**	**19.89**	**4.52**	**11.79**

proposed PU objective.

A comparison of mean average precision (mAP) performance at IoU 0.5 for these 3 settings on PASCAL VOC and MS COCO is shown in Figure 4.7. As expected, as ρ is increased, the detector's performance degrades. Focusing on the results for *PN* and *Full-PN*, it is clear that for an equal number of annotated objects, having fewer images that are more thoroughly annotated is preferable to a larger number of images with less thorough labels. On the other hand, considering object detection as a PU (*PU*) problem as we have proposed allows us to improve detector quality across a wide range of label missingness. While having a more carefully annotated set (*Full-PN*) is still superior, the PU objective helps close the gap. Interestingly, there is a small gain (PASCAL VOC: +0.2, MS COCO: +0.3) in mAP at full labels ($\rho = 0$), possibly due to better learning of objects missing labels in the full dataset.

4.5.3 Visual Genome

Visual Genome [KZG+17] is a scene understanding dataset of objects, attributes, and relationships. While not as commonly used as an object detection benchmark as PASCAL VOC or MS COCO, Visual Genome is popular when relationships or attributes of objects are desired, as when Faster R-CNN is used as a pre-trained feature extractor for Visual Question Answering [ALA+15, AHB+18]. Given the large number of classes (33,877) and the focus on scene understanding during the



(a) (b)

Figure 4.8: Lesion sensitivity versus (a) false positive rate and (b) IoU threshold for different false positive (FP) allowances per image. We compare the baseline Faster R-CNN variant in [YWLS17] trained with a PN objective versus the proposed PU objective.

annotation process, the label coverage of all object instances present in each image is correspondingly lower than PASCAL VOC or MS COCO. In order to achieve its scale, the labeling effort was crowd-sourced to a large number of human annotators. As pointed out in [EVW+10], even increasing from 10 classes of objects in PASCAL VOC2006 to the 20 in VOC2007 resulted in a substantially larger number of labeling errors, as it became more difficult for human annotators to remember all of the object classes. This problem is worse by several orders of magnitude for Visual Genome. While the dataset creators implemented certain measures to ensure quality, there still are many examples of missing labels. In such a setting, the proposed PU risk estimation is especially appropriate, even with all included labels.

We train ResNet101 Faster R-CNN using both PN and the proposed PU risk estimation on 1600 of the top object classes of Visual Genome, as in [AHB+18]. We evaluate performance on the classes present in the test set and report mAP at various IoU thresholds $\{0.25, 0.50, 0.75\}$ in Table **??**. We also show mAP results when each class's average precision is weighted according to class frequency, as done in [AHB+18]. The PASCAL VOC and MS COCO results in Figure 4.7 indicate that

we might expect increasing benefit from utilizing a PU loss as missing labels become especially prevalent, and for Visual Genome, where this is indeed the case, we observe that PU risk estimation outperforms PN by a significant margin, across all settings.

4.5.4 DeepLesion

The recent progress in computer vision has attracted increasing attention towards potential health applications. To encourage deep learning research in this direction, the National Institutes of Health (NIH) Clinical Center released DeepLesion [YWLS17], a dataset consisting of 32K CT scans with annotated lesions. Unlike PASCAL VOC, MS COCO, or Visual Genome, labeling cannot be crowd-sourced for most medical datasets, as accurate labeling requires medical expertise. Even with medical experts, labeling can be inconsistent; lesion detection is a challenging task, with biopsy often necessary to get an accurate result. Like other datasets labeled by an ensemble of annotators, the ground truth of medical datasets may contain inconsistencies, with some doctors being more conservative or aggressive in their diagnoses. Due to these considerations, a PU approach more accurately characterizes the nature of the data.

We re-implemented the modified version of Faster R-CNN described in [YWLS17] as the baseline model and compare against our proposed model using the PU objective, making no other changes. We split the dataset into 70%-15%-15% parts for training, validation, and test. Following [YWLS17], we report results in terms of free receiver operating characteristic (FROC) and sensitivity of lesion detection versus intersection-over-union (IoU) threshold for a range of allowed false positives (FP) per image (Figure 4.8). In both cases, we show that switching from a PN objective to a PU one results in gains in performance.

4.6 Conclusions and Future Work

Having observed that object detection data more closely resembles a positive-unlabeled (PU) problem, we propose training object detection models with a PU objective. Such an objective requires estimation of the class probability of the positive class, but we demonstrate how this can be estimated dynamically with little modification to the existing architecture. Making these changes allows us to achieve improved detection performance across a diverse set of datasets, some of which are real datasets with significant labeling difficulties. While we primarily focused our attention on object detection, a number of other popular tasks share similar characteristics and could also benefit from being recast as PU learning problems (e.g., segmentation [RFB15, LSD15, HGDG17], action detection [SZS12, IZJ+17, GSR+18]).

In our current implementation, we primarily focus on applying the PU objective to the binary object-or-not classifier in Faster R-CNN's Region Proposal Network. A natural extension of this work would be to apply the same objective to the second stage classifier, which must also separate objects from background. However, as the second stage classifier outputs one of several classes (or background), the classification is no longer binary, and requires estimating multiple class priors $\{\pi_c\}_{c=1}^{k}$ [XXXT17], which we leave to future work. Such a multi-class PU loss would also allow extension to single-stage detectors like SSD [LAE+16] and YOLO [RDGF16, RF17]. Given the performance gains already observed, we believe this to be an effective and natural improvement to the object detection classification loss.

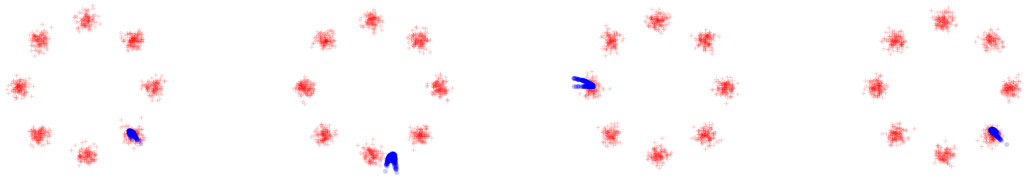
Chapter 5

Generative Adversarial Network Training is a Continual Learning Problem

5.1 Introduction

Generative Adversarial Networks [GPAM+14] (GANs) are a popular framework for modeling draws from complex distributions, demonstrating success in a wide variety of settings, for example image synthesis [RMC16, KALL18] and language modeling [LMS+17]. In the GAN setup, two agents, the *discriminator* and the *generator* (each usually a neural network), are pitted against each other. The generator learns a mapping from an easy-to-sample latent space to a distribution in the data space, which ideally matches the real data's distribution. At the same time, the discriminator aims to distinguish the generator's synthesized samples from the real data samples. When trained successfully, GANs yield impressive results; in the image domain for example, synthesized images from GAN models are significantly sharper and more realistic than those of other classes of models [LSLW16]. On the other hand, GAN training can be notoriously finicky. One particularly well-known and common failure mode is *mode collapse* [CLJ+17, SVR+17]: instead of producing samples sufficiently representing the true data distribution, the generator maps the entire latent space to a limited subset of the real data space.

When mode collapse occurs, the generator does not "converge," in the conventional sense, to a stationary distribution. Rather, because the discriminator can easily learn to recognize a mode-collapsed set of samples and the generator is optimized to avoid the discriminator's detection, the two end up playing a never-ending



(a) Iteration 11960 (b) Iteration 12000 (c) Iteration 12160 (d) Iteration 12380

Figure 5.1: Real samples from a mixture of eight Gaussians in red; generated samples in blue. (a) The generator is mode-collapsed in the bottom right. (b) The discriminator learns to recognize the generator oversampling this region and pushes the generator away, so the generator gravitates toward a new mode. (c) The discriminator continues to chase the generator, causing the generator to move in a clockwise direction. (d) The generator eventually returns to the same mode as (a). Such oscillations are common while training a vanilla GAN. Best seen as a video: https://youtu.be/91a2gPWngo8.

game of cat and mouse: the generator meanders towards regions in the data space the discriminator thinks are real (likely near where the real data lie) while the discriminator chases after it. Interestingly though, if generated samples are plotted through time (as in Figure 5.1), it appears that the generator can revisit previously collapsed modes. At first, this may seem odd. The discriminator was ostensibly trained to recognize that mode in a previous iteration and did so well enough to push the generator away from generating those samples. Why has the discriminator seemingly lost this ability?

We conjecture that this oscillation phenomenon is enabled by *catastrophic forgetting* [MC89, Rat90]: neural networks have a well-known tendency to forget how to complete old tasks while learning new ones. In most GAN models, the discriminator is a binary classifier, with the two classes being the real data and the generator's outputs. Implicit to the training of a standard classifier is the assumption that the data are drawn independently and identically distributed (i.i.d.). Importantly, this assumption does *not* hold true in GANs: the distribution of the generator class (and thus the discriminator's training data) evolves over time. Moreover, these changes in

the generator's distribution are adversarial, designed specifically to deteriorate discriminator performance on the fake class as much as possible. Thus, the alternating training procedure of GANs in actuality corresponds to the discriminator learning tasks sequentially, where each task corresponds to recognizing samples from the generator at that particular point in time. Without any measures to prevent catastrophic forgetting, the discriminator's ability to recognize fake samples from previous iterations will be clobbered by subsequent gradient updates, allowing a mode-collapsed generator to revisit old modes if training runs long enough. Given this tendency, a collapsed generator can wander indefinitely without ever learning the true distribution.

With this perspective in mind, we cast training the GAN discriminator as a continual learning problem, leading to two main contributions. (*i*) While developing systems that learn tasks in a sequential manner without suffering from catastrophic forgetting has become a popular direction of research, current benchmarks have recently come under scrutiny as being unrepresentative to the fundamental challenges of continual learning [FG18]. We argue that GAN training is a more realistic setting, and one that current methods tend to fail on. (*ii*) Such a reframing of the GAN problem allows us to leverage relevant methods to better match the dynamics of training the min-max objective. In particular, we build upon the recently proposed *elastic weight consolidation* [KPR$^+$17] and *intelligent synapses* [ZPG17]. By preserving the discriminator's ability to identify previous generator samples, this memory prevents the generator from simply revisiting past distributions. Adapting the GAN training procedure to account for catastrophic forgetting provides an improvement in GAN performance for little computational cost and without the need to train additional networks. Experiments on CelebA and CIFAR10 image generation and COCO Captions text generation show discriminator continual learning leads to

better generations.

5.2 Catastrophic Forgetting in GANs

Consider distribution $p_{\text{real}}(\boldsymbol{x})$, from which we have data samples $\mathcal{D}^{\text{real}}$. Seeking a mechanism to draw samples from this distribution, we learn a mapping from an easy-to-sample latent distribution $p(\boldsymbol{z})$ to a data distribution $p_{\text{gen}}(\boldsymbol{x})$, which we want to match $p_{\text{real}}(\boldsymbol{x})$. This mapping is parameterized as a neural network $G_{\boldsymbol{\phi}}(\boldsymbol{z})$ with parameters $\boldsymbol{\phi}$, termed the generator. The synthesized data are drawn $\boldsymbol{x} = G_{\boldsymbol{\phi}}(\boldsymbol{z})$, with $\boldsymbol{z} \sim p(\boldsymbol{z})$. The form of $p_{\text{gen}}(\boldsymbol{x})$ is not explicitly assumed or learned; rather, we learn to draw samples from $p_{\text{gen}}(\boldsymbol{x})$.

To provide feedback to $G_{\boldsymbol{\phi}}(\boldsymbol{z})$, we simultaneously learn a binary classifier that aims to distinguish synthesized samples $\mathcal{D}^{\text{gen}}$ drawn from $p_{\text{gen}}(\boldsymbol{x})$ from the true samples $\mathcal{D}^{\text{real}}$. We also parameterize this classifier as a neural network $D_{\boldsymbol{\theta}}(\boldsymbol{x}) \in [0, 1]$ with parameters $\boldsymbol{\theta}$, with $D_{\boldsymbol{\theta}}(\boldsymbol{x})$ termed the discriminator. By incentivizing the generator to fool the discriminator into thinking its generations are actually from the true data, we hope to learn $G_{\boldsymbol{\phi}}(\boldsymbol{z})$ such that $p_{\text{gen}}(\boldsymbol{x})$ approaches $p_{\text{real}}(\boldsymbol{x})$.

These two opposing goals for the generator and discriminator are usually formulated as the following min-max objective:

$$\min_{\boldsymbol{\phi}} \max_{\boldsymbol{\theta}} \mathcal{L}^{\text{GAN}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{real}}(\boldsymbol{x})}[\log D_{\boldsymbol{\theta}}(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\log(1 - D_{\boldsymbol{\theta}}(G_{\boldsymbol{\phi}}(\boldsymbol{z})))] \quad (5.1)$$

At each iteration t, we sample from $p_{\text{gen}}(\boldsymbol{x})$, yielding generated data $\mathcal{D}_t^{\text{gen}}$. These generated samples, along with samples from $\mathcal{D}^{\text{real}}$, are then passed to the discriminator. A gradient descent optimizer nudges $\boldsymbol{\theta}$ so that the discriminator takes a step towards maximizing $\mathcal{L}^{\text{GAN}}(\boldsymbol{\theta}, \boldsymbol{\phi})$. Parameters $\boldsymbol{\phi}$ are updated similarly, but to minimize $\mathcal{L}^{\text{GAN}}(\boldsymbol{\theta}, \boldsymbol{\phi})$. These updates to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ take place in an alternating fashion.

The expectations are approximated using samples from the respective distributions, and therefore learning only requires observed samples $\mathcal{D}^{\text{real}}$ and samples from $p_{\text{gen}}(\boldsymbol{x})$.

The updates to $G_{\boldsymbol{\phi}}(\boldsymbol{z})$ mean that $p_{\text{gen}}(\boldsymbol{x})$ changes as a function of t, perhaps substantially. Consequently, samples $\{\mathcal{D}_1^{\text{gen}}, ..., \mathcal{D}_t^{\text{gen}}\}$ come from a sequence of different distributions. At iteration t, only samples from $\mathcal{D}_t^{\text{gen}}$ are available, as $G_{\boldsymbol{\phi}}(\boldsymbol{z})$ has changed, and saving previous instantiations of the generator or samples $\{\mathcal{D}_1^{\text{gen}}, ..., \mathcal{D}_{t-1}^{\text{gen}}\}$ can be prohibitive. Thus, $D_{\boldsymbol{\theta}}(\boldsymbol{x})$ is typically only provided $\mathcal{D}_t^{\text{gen}}$, so it only learns the most recent distribution, with complete disregard for previous $p_{\text{gen}}(\boldsymbol{x})$. Because of the catastrophic forgetting effect of neural networks, the ability of $D_{\boldsymbol{\theta}}(\boldsymbol{x})$ to recognize these previous distributions is eventually lost in the pursuit of maximizing $\mathcal{L}^{\text{GAN}}(\boldsymbol{\theta}, \boldsymbol{\phi})$ with respect to *only* $\mathcal{D}_t^{\text{gen}}$. This opens the possibility that the generator goes back to generating samples the discriminator had previously learned (and then forgot) to recognize, leading to unstable mode-collapsed oscillations that hamper GAN training (as in Figure 5.1). Recognizing this problem, we propose that the discriminator should be trained with the temporal component of $p_{\text{gen}}(\boldsymbol{x})$ in mind.

5.3 Method

5.3.1 Classic Continual Learning

Catastrophic forgetting has long been known to be a problem with neural networks trained on a series of tasks [MC89, Rat90]. While there are many approaches to addressing catastrophic forgetting, here we primarily focus on elastic weight consolidation (EWC) and intelligent synapses (IS). These are meant to illustrate the potential of catastrophic forgetting mitigation to improve GAN learning, with the expectation that this opens up the possibility of other such methods to significantly improve GAN training, at low additional computational cost.

Elastic Weight Consolidation (EWC)

To derive the EWC loss, [KPR+17] frames training a model as finding the most probable values of the parameters $\boldsymbol{\theta}$ given the data $\mathcal{D}$. For two tasks, the data are assumed partitioned into independent sets according to the task, and the posterior for Task 1 is approximated as a Gaussian with mean centered on the optimal parameters for Task 1 $\boldsymbol{\theta}_1^*$ and diagonal precision given by the diagonal of the Fisher information matrix F_1 at $\boldsymbol{\theta}_1^*$. This gives the EWC loss the following form:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_2(\boldsymbol{\theta}) + \mathcal{L}^{\text{EWC}}(\boldsymbol{\theta}), \quad \text{with} \quad \mathcal{L}^{\text{EWC}}(\boldsymbol{\theta}) \triangleq \frac{\lambda}{2} \sum_i F_{1,i}(\theta_i - \theta_{1,i}^*)^2 \ , \tag{5.2}$$

where $\mathcal{L}_2(\boldsymbol{\theta}) = \log p(\mathcal{D}_2|\boldsymbol{\theta})$ is the loss for Task 2 individually, λ is a hyperparameter representing the importance of Task 1 relative to Task 2, $F_{1,i} = \big(\frac{\partial \mathcal{L}_1(\boldsymbol{\theta})}{\partial \theta_i}\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_1^*}\big)^2$, i is the parameter index, and $\mathcal{L}(\boldsymbol{\theta})$ is the new loss to optimize while learning Task 2. Intuitively, the EWC loss prevents the model from straying too far away from the parameters important for Task 1 while leaving less crucial parameters free to model Task 2. Subsequent tasks result in additional $\mathcal{L}^{\text{EWC}}(\boldsymbol{\theta})$ terms added to the loss for each previous task. By protecting the parameters deemed important for prior tasks, EWC as a regularization term allows a single neural network (assuming sufficient parameters and capacity) to learn new tasks in a sequential fashion, without forgetting how to perform previous tasks.

Intelligent Synapses (IS)

While EWC makes a point estimate of how essential each parameter is at the conclusion of a task, IS [ZPG17] protects the parameters according to their importance along the task's entire training trajectory. Termed synapses, each parameter θ_i of the neural network is awarded an *importance measure* $\omega_{1,i}$ based on how much it reduced the loss while learning Task 1. Given a loss gradient $\boldsymbol{g}(t) = \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$ at time t,

the total change in loss during the training of Task 1 then is the sum of differential changes in loss over the training trajectory. With the assumption that parameters $\boldsymbol{\theta}$ are independent, we have:

$$\int_{t^0}^{t^1} \boldsymbol{g}(t) d\boldsymbol{\theta} = \int_{t^0}^{t^1} \boldsymbol{g}(t)\boldsymbol{\theta}' dt = \sum_i \int_{t^0}^{t^1} g_i(t)\theta_i' dt \triangleq -\sum_i \omega_{1,i} \ , \tag{5.3}$$

where $\boldsymbol{\theta}' = \frac{d\boldsymbol{\theta}}{dt}$ and (t^0, t^1) are the start and finish of Task 1, respectively. Note the added negative sign, as importance is associated with parameters that decrease the loss.

The importance measure $\omega_{1,i}$ can now be used to introduce a regularization term that protects parameters important for Task 1 from large parameter updates, just as the Fisher information matrix diagonal terms $F_{1,i}$ were used in EWC. This results in an IS loss very reminiscent in form[1]:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_2(\boldsymbol{\theta}) + \mathcal{L}^{\text{IS}}(\boldsymbol{\theta}), \quad \text{with} \quad \mathcal{L}^{\text{IS}}(\boldsymbol{\theta}) \triangleq \frac{\lambda}{2} \sum_i \omega_{1,i} (\theta_i - \theta_{1,i}^*)^2 \ . \tag{5.4}$$

5.3.2 GAN Continual Learning

The traditional continual learning methods are designed for certain canonical benchmarks, commonly consisting of a small number of clearly defined tasks (e.g., classification datasets in sequence). In GANs, the discriminator is trained on dataset $\mathcal{D}_t = \{\mathcal{D}^{\text{real}}, \mathcal{D}_t^{\text{gen}}\}$ at each iteration t. However, because of the evolution of the generator, the distribution $p_{\text{gen}}(x)$ from which $\mathcal{D}_t^{\text{gen}}$ comes changes over time. This violates the i.i.d. assumption of the order in which we present the discriminator data. As such, we argue that different instances in time of the generator should be viewed

[1] [ZPG17] instead consider $\Omega_{1,i} = \frac{\omega_{1,i}}{(\Delta_{1,i})^2 + \xi}$, where $\Delta_{1,i} = \theta_{1,i} - \theta_{0,i}$ and ξ is a small number for numerical stability. We however found that the inclusion of $(\Delta_{1,i})^2$ can lead to the loss exploding and then collapsing as the number of tasks increases and so omit it. We also change the hyperparameter c into $\frac{\lambda}{2}$.

as separate tasks. Specifically, in the parlance of continual learning, the training data are to be regarded as $\mathcal{D} = \{(\mathcal{D}^{\text{real}}, \mathcal{D}_1^{\text{gen}}), (\mathcal{D}^{\text{real}}, \mathcal{D}_2^{\text{gen}}), ...\}$. Thus motivated, we would like to apply continual learning methods to the discriminator, but doing so is not straightforward for the following reasons:

- **Definition of a task**: EWC and IS were originally proposed for discrete, well-defined tasks. For example, [KPR+17] applied EWC to a DQN [MKS+15] learning to play ten Atari games sequentially, with each game being a clear, independent task. For GAN, there is no such precise definition as to what constitutes a "task," and as discriminators are not typically trained to convergence at every iteration, it is also unclear how long a task should be.

- **Computational memory**: While Equations 5.2 and 5.4 are for two tasks, they can be extended to K tasks by adding a term $\mathcal{L}^{\text{EWC}}$ or $\mathcal{L}^{\text{IS}}$ for each of the $K-1$ prior tasks. As each term $\mathcal{L}^{\text{EWC}}$ or $\mathcal{L}^{\text{IS}}$ requires saving both a historical reference term $\boldsymbol{\theta}_k^*$ and either F_k or $\boldsymbol{\omega}_k$ (all of which are the same size as the model parameters $\boldsymbol{\theta}$) for each task k, employing these techniques naively quickly becomes impractical for bigger models when K gets large, especially if K is set to the number of training iterations T.

- **Continual *not* learning**: Early iterations of the discriminator are likely to be non-optimal, and without a forgetting mechanism, EWC and IS may forever lock the discriminator to a poor initialization. Additionally, the unconstrained addition of a large number of terms $\mathcal{L}^{\text{EWC}}$ or $\mathcal{L}^{\text{IS}}$ will cause the continual learning regularization term to grow unbounded, which can disincentivize any further changes in $\boldsymbol{\theta}$.

To address these issues, we build upon the aforementioned continual learning techniques, and propose several changes.

Number of tasks as a rate: We choose the total number of tasks K as a function of a constant rate α, which denotes the number of iterations before the conclusion of a task, as opposed to arbitrarily dividing the GAN training iterations into some set number of segments. Given T training iterations, this means a rate α yields $K = \frac{T}{\alpha}$ tasks.

Online Memory: Seeking a way to avoid storing extra $\boldsymbol{\theta}_k^*$, F_k, or $\boldsymbol{\omega}_k$, we observe that the sum of two or more quadratic forms is another quadratic, which gives the classifier loss with continual learning the following form for the $(k+1)^{\text{th}}$ task:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{k+1}(\boldsymbol{\theta}) + \mathcal{L}^{\text{CL}}(\boldsymbol{\theta}), \quad \text{with} \quad \mathcal{L}^{\text{CL}}(\boldsymbol{\theta}) \triangleq \frac{\lambda}{2} \sum_i S_{k,i}(\theta_i - \bar{\theta}_{k,i}^*)^2 \ , \tag{5.5}$$

where $\bar{\theta}_{k,i}^* = \frac{P_{k,i}}{S_{k,i}}$, $S_{k,i} = \sum_{\kappa=1}^{k} Q_{\kappa,i}$, $P_{k,i} = \sum_{\kappa=1}^{k} Q_{\kappa,i}\theta_{\kappa,i}^*$, and $Q_{\kappa,i}$ is either $F_{\kappa,i}$ or $\omega_{\kappa,i}$, depending on the method. We name models with EWC and IS augmentations EWC-GAN and IS-GAN, respectively.

Controlled forgetting: To provide a mechanism for forgetting earlier non-optimal versions of the discriminator and to keep $\mathcal{L}^{\text{CL}}$ bounded, we add a discount factor γ: $S_{k,i} = \sum_{\kappa=1}^{k} \gamma^{k-\kappa} Q_{\kappa,i}$ and $P_{k,i} = \sum_{\kappa=1}^{k} \gamma^{k-\kappa} Q_{\kappa,i}\theta_{\kappa,i}^*$. Together, α and γ determine how far into the past the discriminator remembers previous generator distributions, and λ controls how important memory is relative to the discriminator loss. Note, the terms S_k and P_k can be updated every α steps in an online fashion:

$$S_{k,i} = \gamma S_{k-1,i} + Q_{k,i}, \qquad P_{k,i} = \gamma P_{k-1,i} + Q_{k,i}\theta_{k,i}^* \tag{5.6}$$

This allows the EWC or IS loss to be applied without necessitating storing either Q_k or $\boldsymbol{\theta}_k^*$ for every task k, which would quickly become too costly to be practical. Only a single variable to store a running average is required for each of S_k and P_k, making this method space efficient.

Augmenting the discriminator with the continual learning loss, the GAN objective

becomes:

$$\min_{\boldsymbol{\phi}} \max_{\boldsymbol{\theta}} \mathcal{L}^{\mathrm{CL}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{L}^{\mathrm{GAN}}(\boldsymbol{\theta}, \boldsymbol{\phi}) - \mathcal{L}^{\mathrm{CL}}(\boldsymbol{\theta}) \tag{5.7}$$

Note that the training of the generator remains the same; full algorithms are in Appendix A. Here we have shown two methods to mitigate catastrophic forgetting for the original GAN; however, the proposed framework is applicable to almost all of the wide range of GAN setups.

5.4 Related Work

Continual Learning in GANs There has been previous work investigating continual learning within the context of GANs. Improved GAN [SGZ+16] introduced historical averaging, which regularizes the model with a running average of parameters of the most recent iterations. Simulated+Unsupervised training [SPT+17] proposed replacing half of each minibatch with previous generator samples during training of the discriminator, as a generated sample at any point in time should always be considered fake. However, such an approach necessitates a historical buffer of samples and halves the number of current samples that can be considered. Continual Learning GAN [SBSL18] applied EWC to GAN, as we have, but used it in the context of the class-conditioned generator that learns classes sequentially, as opposed to all at once, as we propose. [TTTV18] independently reached a similar conclusion on catastrophic forgetting in GANs, but focused on gradient penalties and momentum on toy problems.

Multiple network GANs The heart of continual learning is distilling a network's knowledge through time into a single network, a temporal version of the ensemble described in [HVD15]. There have been several proposed models utilizing multiple generators [HNLP18, GKN+18] or multiple discriminators [DGM17, NBC17],

while Bayesian GAN [SW17] considered distributions on the parameters of both networks, but these all do not consider time as the source of the ensemble. Unrolled GAN [MPPSD17] considered multiple discriminators "unrolled" through time, which is similar to our method, as the continual learning losses also utilize historical instances of discriminators. However, both EWC-GAN and IS-GAN preserve the important parameters for prior discriminator performance, as opposed to requiring backpropagation of generator samples through multiple networks, making them easier to implement and train.

GAN convergence While GAN convergence is not the focus of this paper, convergence does similarly avoid mode collapse, and there are a number of works on the topic [HRU+17, UNS+18, NK17, MNG17]. From the perspective of [HRU+17], EWC or IS regularization in GAN can be viewed as achieving convergence by slowing the discriminator, but per parameter, as opposed to a slower global learning rate.

5.5 Experiments

5.5.1 Discriminator Catastrophic Forgetting

While Figure 5.1 implies catastrophic forgetting in a GAN discriminator, we can show this concretely. To do so, we first train a DCGAN [RMC16] on the MNIST dataset. Since the generator is capable of generating an arbitrary number of samples at any point, we can randomly draw 70000 samples to comprise a new, "fake MNIST" dataset at any time. By doing this at regular intervals, we create datasets $\{\mathcal{D}_1^{\text{gen}}, ..., \mathcal{D}_T^{\text{gen}}\}$ from $p_{\text{gen}}(x)$ at times $1, ..., T$. Samples are shown in Appendix B.

Having previously generated a series of datasets during the training of a DCGAN, we now reinitialize the discriminator and train to convergence on each $\mathcal{D}_t^{\text{gen}}$ in sequence. Importantly, we do *not* include samples from $\mathcal{D}_{<t}^{\text{gen}}$ while fine-tuning on

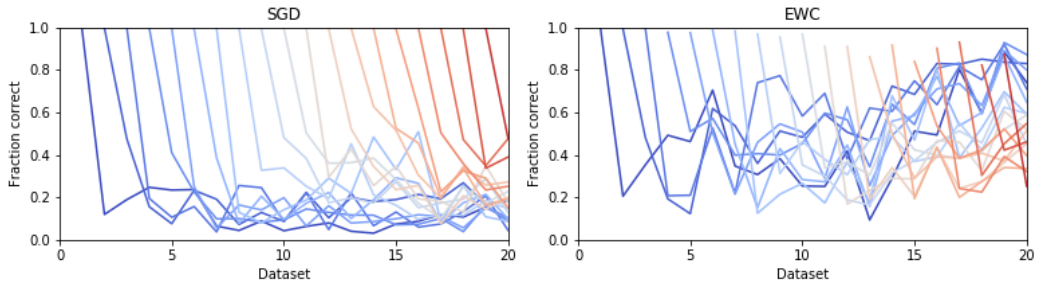


Figure 5.2: Each line represents the discriminator's test accuracy on the fake GAN datasets. Note the sharp decrease in the discriminator's ability to recognize previous fake samples upon fine-tuning on the next dataset using SGD (left). Forgetting still occurs with EWC (right), but is less severe.

$\mathcal{D}_t^{\text{gen}}$. After fine-tuning on the train split of dataset $\mathcal{D}_t^{\text{gen}}$, the percentage of generated examples correctly identified as fake by the discriminator is evaluated on the test splits of $\mathcal{D}_{\leq t}^{\text{gen}}$, with and without EWC (Figure 5.2). The catastrophic forgetting effect of the discriminator trained with SGD is clear, with a steep drop-off in discriminating ability on $\mathcal{D}_{t-1}^{\text{gen}}$ after fine-tuning on $\mathcal{D}_t^{\text{gen}}$; this is unsurprising, as $p_{\text{gen}}(x)$ has evolved specifically to deteriorate discriminator performance. While there is still a dropoff with EWC, forgetting is less severe.

While the training outlined above is not what is typical for GAN, we choose this set-up as it closely mirrors the continual learning literature. With recent criticisms of some common continual learning benchmarks as either being too easy or missing the point of continual learning [FG18], we propose GAN as a new benchmark providing a more realistic setting. From Figure 5.2, it is clear that while EWC certainly helps, there is still much room to improve with new continual learning methods. However, the merits of GAN as a continual learning benchmark go beyond difficulty. While it is unclear why one would ever use a single model to classify successive random permutations of MNIST [GMX+13], many real-world settings exist where the data distribution is slowly evolving. For such models, we would like to be able to update the deployed model without forgetting previously learned performance, es-

pecially when data collection is expensive and thus done in bulk sometime before deployment. For example, autonomous vehicles [HWT+15] will eventually encounter unseen car models or obstacles, and automated screening systems at airport checkpoints [LHG+18] will have to deal with evolving bags, passenger belongings, and threats. In both cases, sustained effectiveness requires a way to appropriately and efficiently update the models for new data, or risk obsolescence leading to dangerous blindspots.

Many machine learning datasets represent singe-time snapshots of the data distribution, and current continual learning benchmarks fail to capture the slow drift of the real-world data. The evolution of GAN synthesized samples represents an opportunity to generate an unlimited number of smoothly evolving datasets for such experiments. We note that while the setup used here is for binary real/fake classification, one could also conceivably use a conditional GAN [MO14] to generate an evolving multi-class classification dataset. We leave this exploration for future work.

5.5.2 Mixture of Eight Gaussians

We show results on a toy dataset consisting of a mixture of eight Gaussians, as in the example in Figure 5.1. Following the setup of [MPPSD17], the real data are evenly distributed among eight 2-dimensional Gaussian distributions arranged in a circle of radius 2, each with covariance $0.02I$ (see Figure 5.4). We evaluate our model with Inception Score (ICP) [SGZ+16], which gives a rough measure of diversity and quality of samples; higher scores imply better performance, with the true data resulting in a score of around 7.870. For this simple dataset, since we know the true data distribution, we also calculate the symmetric Kullback–Leibler divergence (Sym-KL); lower scores mean the generated samples are closer to the true data. We show computation time, measured in numbers of training iterations per

Table 5.1: Iterations per second, inception score, and symmetric KL divergence comparison on a mixture of eight Gaussians.

Model						
Method	α	λ	γ	Iter/s ↑	ICP ↑	Sym-KL ↓
GAN	-	-	-	87.59 ± 1.45	2.835 ± 2.325	19.55 ± 3.07
GAN + ℓ_2 weight	1	0.01	0		5.968 ± 1.673	15.19 ± 2.67
GAN + historical avg.	1	0.01	0.995		7.305 ± 0.158	13.32 ± 0.88
GAN + SN	-	-	-	49.70 ± 0.13	6.762 ± 2.024	13.37 ± 3.86
GAN + IS	1000	100	0.8	42.26 ± 0.35	7.039 ± 0.294	15.10 ± 1.51
GAN + IS	100	10	0.98	42.29 ± 0.10	7.500 ± 0.147	11.85 ± 0.92
GAN + IS	10	100	0.99	41.07 ± 0.07	7.583 ± 0.242	11.88 ± 0.84
GAN + SN + IS	10	100	0.99	25.69 ± 0.09	7.699 ± 0.048	11.10 ± 1.18
GAN + EWC	1000	100	0.8	82.78 ± 1.55	7.480 ± 0.209	13.00 ± 1.55
GAN + EWC	100	10	0.98	80.63 ± 0.39	7.488 ± 0.222	12.16 ± 1.64
GAN + EWC	10	10	0.99	73.86 ± 0.16	7.670 ± 0.112	11.90 ± 0.76
GAN + SN + EWC	10	10	0.99	44.68 ± 0.11	7.708 ± 0.057	11.48 ± 1.12

second (Iter/s), averaged over the full training of a model on a single Nvidia Titan X (Pascal) GPU. Each model was run 10 times, with the mean and standard deviation of each performance metric at the end of 25K iterations reported in Table 5.1.

The performance of EWC-GAN and IS-GAN were evaluated for a number of hyperparameter settings. We compare our results against a vanilla GAN [GPAM+14], as well as a state-of-the-art GAN with spectral normalization (SN) [MKKY18] applied to the discriminator. As spectral normalization augments the discriminator loss in a way different from continual learning, we can combine the two methods; this variant is also shown.

Note that a discounted version of discriminator historical averaging [SGZ+16] can be recovered from the EWC and IS losses if the task rate $\alpha = 1$ and $Q_{k,i} = 1$ for all i and k, a poor approximation to both the Fisher information matrix diagonal and importance measure. If we also set the historical reference term $\bar{\theta}^*_k$ and the discount factor γ to zero, then the EWC and IS losses become ℓ_2 weight regularization. These two special cases are also included for comparison.

We observe that augmenting GAN models with EWC and IS consistently results in generators that better match the true distribution, both qualitatively and quantitatively, for a wide range of hyperparameter settings. EWC-GAN and IS-GAN result in a better ICP and FID than ℓ_2 weight regularization and discounted historical averaging, showing the value of prioritizing protecting important parameters, rather than all parameters equally. EWC-GAN and IS-GAN also outperform a state-of-the-art method in SN-GAN. In terms of training time, updating the EWC loss requires forward propagating a new minibatch through the discriminator and updating S and P, but even if this is done at every step ($\alpha = 1$), the resulting algorithm is only slightly slower than SN-GAN. Moreover, doing so is unnecessary, as higher values of α also provide strong performance for a much smaller time penalty. Combining EWC with SN-GAN leads to even better results, showing that the two methods can complement each other. IS-GAN can also be successfully combined with SN-GAN, but it is slower than EWC-GAN as it requires tracking the trajectory of parameters at each step. Sample generation evolution over time is shown in Figure 5.4 of Appendix C.

5.5.3 Image Generation of CelebA and CIFAR-10

Since EWC-GAN achieves similar performance to IS-GAN but at less computational expense, we focus on the former for experiments on two image datasets, CelebA and CIFAR-10. Our EWC-GAN implementation is straightforward to add to any GAN model, so we augment various popular implementations. Comparisons are made with the TTUR [HRU+17] variants[2] of DCGAN [RMC16] and WGAN-GP [GAA+17], as well as an implementation[3] of a spectral normalized [MKKY18] DCGAN (SN-DCGAN). Without modifying the learning rate or model architecture, we show results with and without the EWC loss term added to the discriminator for each.

[2] `https://github.com/bioinf-jku/TTUR`

[3] `https://github.com/minhnhat93/tf-SNDCGAN`

Table 5.2: Fréchet Inception Distance and Inception Score on CelebA and CIFAR-10

	CelebA	CIFAR-10	
Method	FID ↓	FID ↓	ICP ↑
DCGAN	12.52	41.44	6.97 ± 0.05
DCGAN + EWC	10.92	34.84	7.10 ± 0.05
WGAN-GP	-	30.23	7.09 ± 0.06
WGAN-GP + EWC	-	29.67	7.44 ± 0.08
SN-DCGAN	-	27.21	7.43 ± 0.10
SN-DCGAN + EWC	-	25.51	7.58 ± 0.07

Performance is quantified with the Fréchet Inception Distance (FID) [HRU+17] for both datasets. Since labels are available for CIFAR-10, we also report ICP for that dataset. Best values are reported in Table 5.2, with samples in Appendix C. In each model, we see improvement in both FID and ICP from the addition of EWC to the discriminator.

5.5.4 Text Generation of COCO Captions

We also consider the text generation on the MS COCO Captions dataset [CFL+15], with the pre-processing in [GLC+18]. Quality of generated sentences is evaluated by BLEU score [PRWZ02]. Since BLEU-b measures the overlap of b consecutive words between the generated sentences and ground-truth references, higher BLEU scores indicate better fluency. Self BLEU uses the generated sentences themselves as references; lower values indicate higher diversity.

We apply EWC and IS to textGAN [ZGF+17], a recently proposed model for text generation in which the discriminator uses feature matching to stabilize training. This model's results (labeled "EWC" and "IS") are compared to a Maximum Likelihood Estimation (MLE) baseline, as well as several state-of-the-art methods: SeqGAN [YZWY17], RankGAN [LLH+17], GSGAN [JGP16] and LeakGAN [GLC+18]. Our variants of textGAN outperforms the vanilla textGAN for all BLEU scores (see

Table 5.3: Test BLEU ↑ results on MS COCO

Method	MLE	SeqGAN	RankGAN	GSGAN	LeakGAN	textGAN	EWC	IS
BLEU-2	0.820	0.820	0.852	0.810	0.922	0.926	0.934	0.933
BLEU-3	0.607	0.604	0.637	0.566	0.797	0.781	0.802	0.791
BLEU-4	0.389	0.361	0.389	0.335	0.602	0.567	0.594	0.578
BLEU-5	0.248	0.211	0.248	0.197	0.416	0.379	0.400	0.388

Table 5.4: Self BLEU ↓ results on MS COCO

Method	MLE	SeqGAN	RankGAN	GSGAN	LeakGAN	textGAN	EWC	IS
BLEU-2	0.754	0.807	0.822	0.785	0.912	0.843	0.854	0.853
BLEU-3	0.511	0.577	0.592	0.522	0.825	0.631	0.671	0.655
BLEU-4	0.232	0.278	0.288	0.230	0.689	0.317	0.388	0.364

Table 5.3), indicating the effectiveness of addressing the forgetting issue for GAN training in text generation. EWC/IS + textGAN also demonstrate a significant improvement compared with other methods, especially on BLEU-2 and 3. Though our variants lag slightly behind LeakGAN on BLEU-4 and 5, their self BLEU scores (Table 5.4) indicate it generates more diverse sentences. Sample sentence generations can be found in Appendix C.

5.6 Conclusions

We observe that the alternating training procedure of GAN models results in a continual learning problem for the discriminator, and training on only the most recent generations leads to consequences unaccounted for by most models. As such, we propose augmenting the GAN training objective with a continual learning regularization term for the discriminator to prevent its parameters from moving too far away from values that were important for recognizing synthesized samples from previous training iterations. Since the original EWC and IS losses were proposed for discrete tasks, we adapt them to the GAN setting. Our implementation is simple to add to almost

any variation of GAN learning, and we do so for a number of popular models, showing a gain in ICP and FID for CelebA and CIFAR-10, as well as BLEU scores for COCO Captions. More importantly, we demonstrate that GAN and continual learning, two popular fields studied independently of each other, have the potential to benefit each other, as new continual learning methods stand to benefit GAN training, and GAN generated datasets provide new testing grounds for continual learning.

Appendix

A. Algorithm

We summarize the continual learning GAN implementations in Algorithm 1 and 2.

Algorithm 1 Continual learning GAN with EWC

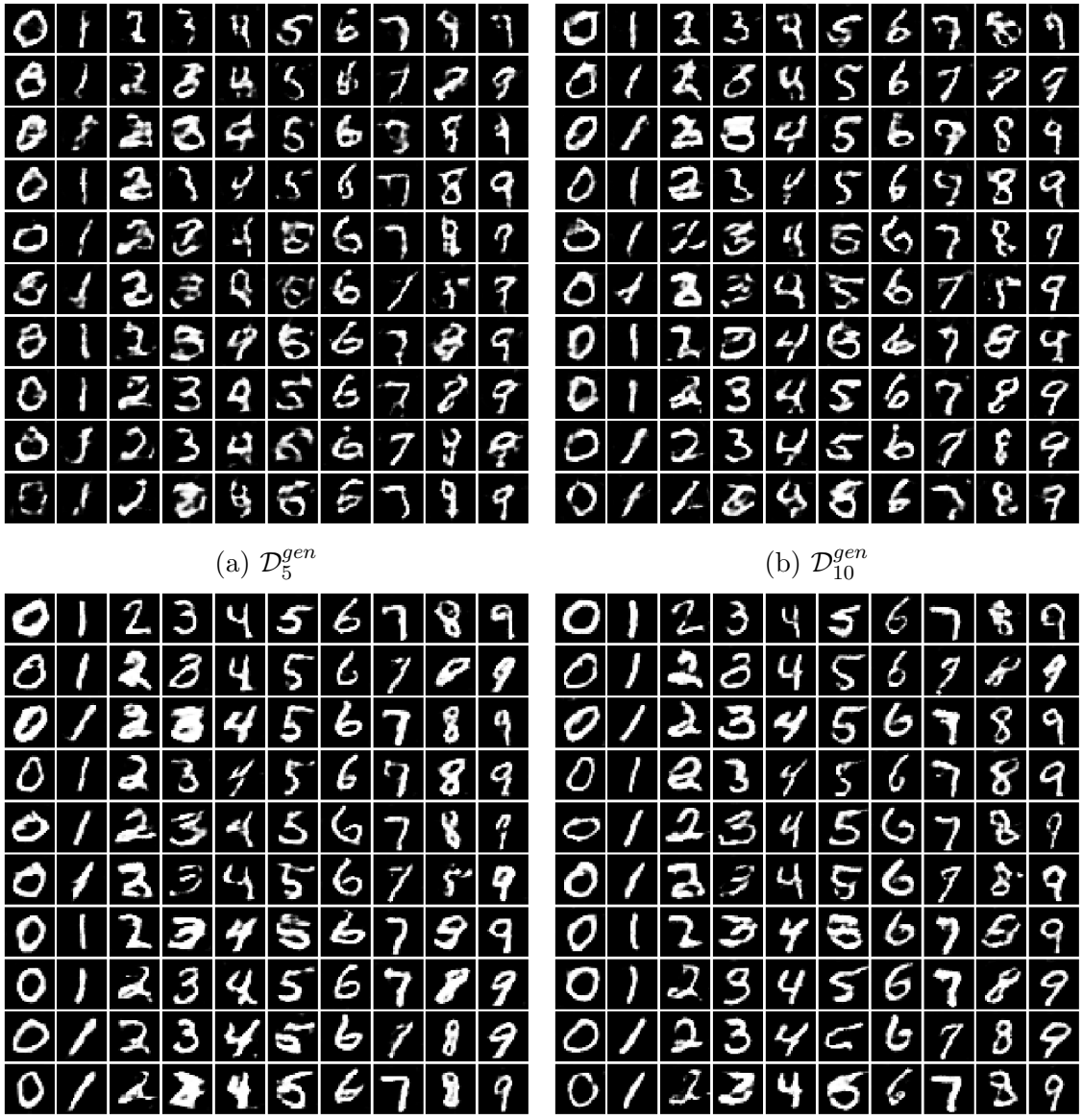
Input: Training data $\mathcal{D}^{\text{real}}$, latent distribution $p(\boldsymbol{z})$, hyperparameters of continual learning α, γ, λ, step size ϵ
Output: $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, and generated samples $\mathcal{D}^{\text{gen}} = \{\boldsymbol{x}_j\}_{j=1}^{N}$
for $t = 1, ..., T$ **do**
  Noise sample: $\{\boldsymbol{z}_j\}_{j=1}^{m} \sim p(\boldsymbol{z})$
  Data sample: $\{\boldsymbol{x}_j\}_{j=1}^{m} \sim \mathcal{D}^{\text{real}}$
  % `Calculate current discriminator loss`
  $\mathcal{L}_{\boldsymbol{\theta}} = \frac{1}{m}\sum_{j=1}^{m}[\log D_{\boldsymbol{\theta}}(\boldsymbol{x}_j)] + \frac{1}{m}\sum_{j=1}^{m}[\log(1 - D_{\boldsymbol{\theta}}(G_{\boldsymbol{\phi}}(\boldsymbol{z}_j)))]$
  % `Update history buffer for previous tasks every` α `steps`
  if $\mod(t, \alpha) = 0$ **then**
    for parameters θ_i in $\boldsymbol{\theta}$:
      $Q_i = \left(\frac{\partial \mathcal{L}_{\boldsymbol{\theta}}}{\partial \theta_i}\right)^2$ % Q is the Fisher for EWC
      $S_i = \gamma S_i + Q_i$
      $P_i = \gamma P_i + Q_i \theta_i^*$
      $\bar{\theta}_i^* = \frac{P_i}{S_i}$
  end if
  % `Update discriminator parameter` $\boldsymbol{\theta}$, `adding EWC`
  $\bar{\mathcal{L}}_{\boldsymbol{\theta}} = \mathcal{L}_{\boldsymbol{\theta}} - \frac{\lambda}{2}\sum_i S_{k,i}(\theta_i - \bar{\theta}_{k,i}^*)$
  $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \epsilon_t \frac{\partial \bar{\mathcal{L}}_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}_t}$
  % `Update generator parameter` $\boldsymbol{\phi}$
  $\mathcal{L}_{\boldsymbol{\phi}} = \frac{1}{m}\sum_{i=1}^{m}[\log(1 - D_{\boldsymbol{\theta}}(G_{\boldsymbol{\phi}}(\boldsymbol{z}_i)))]$
  $\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \epsilon_t \frac{\partial \mathcal{L}_{\boldsymbol{\phi}}}{\partial \boldsymbol{\phi}_t}$
end for

Algorithm 2 Continual learning GAN with IS

Input: Training data $\mathcal{D}^{\text{real}}$, latent distribution $p(\boldsymbol{z})$, hyperparameters of continual learning α, γ, λ, step size ϵ
Output: $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, and generated samples $\mathcal{D}^{\text{gen}} = \{\boldsymbol{x}_j\}_{j=1}^{N}$
for $t = 1, ..., T$ **do**
  Noise sample: $\{\boldsymbol{z}_j\}_{j=1}^{m} \sim p(\boldsymbol{z})$
  Data sample: $\{\boldsymbol{x}_j\}_{j=1}^{m} \sim \mathcal{D}^{\text{real}}$
  % `Calculate current discriminator loss`
  $\mathcal{L}_{\boldsymbol{\theta}} = \frac{1}{m}\sum_{j=1}^{m}[\log D_{\boldsymbol{\theta}}(\boldsymbol{x}_j)] + \frac{1}{m}\sum_{j=1}^{m}[\log(1 - D_{\boldsymbol{\theta}}(G_{\boldsymbol{\phi}}(\boldsymbol{z}_j)))]$
  % `Update discriminator parameter` $\boldsymbol{\theta}$, `adding IS`
  $\bar{\mathcal{L}}_{\boldsymbol{\theta}} = \mathcal{L}_{\boldsymbol{\theta}} - \frac{\lambda}{2}\sum_i S_{k,i}(\theta_i - \bar{\theta}^*_{k,i})$
  $\boldsymbol{g} = \frac{\partial \bar{\mathcal{L}}_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}_t}$
  $\boldsymbol{\delta} = \epsilon \frac{\partial \bar{\mathcal{L}}_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}_t}$
  for θ_i in $\boldsymbol{\theta}$ **do**
    $\omega_i = \omega_i + g_i \delta_i$
  end for
  $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \epsilon_t \frac{\partial \bar{\mathcal{L}}_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}_t}$
  % `Update history buffer for previous tasks every` α `steps`
  if $\mod(t, \alpha) = 0$ **then**
    for parameters θ_i in $\boldsymbol{\theta}$:
      $Q_i = \omega_i$ % Q is the Importance measure for IS
      $S_i = \gamma S_i + Q_i$
      $P_i = \gamma P_i + Q_i \theta_i^*$
      $\bar{\theta}_i^* = \frac{P_i}{S_i}$
      $\boldsymbol{\omega} = \mathbf{0}$
  end if
  % `Update generator parameter` $\boldsymbol{\phi}$
  $\mathcal{L}_{\boldsymbol{\phi}} = \frac{1}{m}\sum_{i=1}^{m}[\log(1 - D_{\boldsymbol{\theta}}(G_{\boldsymbol{\phi}}(\boldsymbol{z}_i)))]$
  $\boldsymbol{\phi}_{t+1} \leftarrow \boldsymbol{\phi}_t - \epsilon_t \frac{\partial \mathcal{L}_{\boldsymbol{\phi}}}{\partial \boldsymbol{\phi}_t}$
end for

B. Generated MNIST Datasets for Continual Learning Benchmarking

To produce a smoothly evolving series of datasets for continual learning, we train a DCGAN on MNIST and generate an entire "fake" dataset of 70K samples every 50 training iterations of the DCGAN generator. We propose learning each of these generated datasets as individual tasks for continual learning. Selected samples are shown in Figure 5.3 from the datasets $\mathcal{D}_t^{gen}$ for $t \in \{5, 10, 15, 20\}$, each generated from the same 100 samples of z for all t. Note that we actually trained a conditional DCGAN, meaning we also have the labels for each generated image. For experiments in Figure 5.2, we focused on the real versus fake task to demonstrate catastrophic forgetting in a GAN discriminator and thus ignored the labels, but future experiments can incorporate such information.



(a) $\mathcal{D}_5^{gen}$ (b) $\mathcal{D}_{10}^{gen}$

(c) $\mathcal{D}_{15}^{gen}$ (d) $\mathcal{D}_{20}^{gen}$

Figure 5.3: Image samples from a few generated "fake MNIST" datasets

C. Examples of Generated Samples

Sample generations are plotted during training at 5000 step intervals in Figure 5.4. While vanilla GAN occasionally recovers the true distribution, more often than not, the generator collapses and then bounces around. Spectral Normalized GAN converges to the true distribution quickly in most runs, but it mode collapses and exhibits

the same behavior as GAN in others. EWC-GAN consistently diffuses to all modes, tending to find the true distribution sooner with lower α. We omit IS-GAN, as it performs similarly to EWC-GAN.

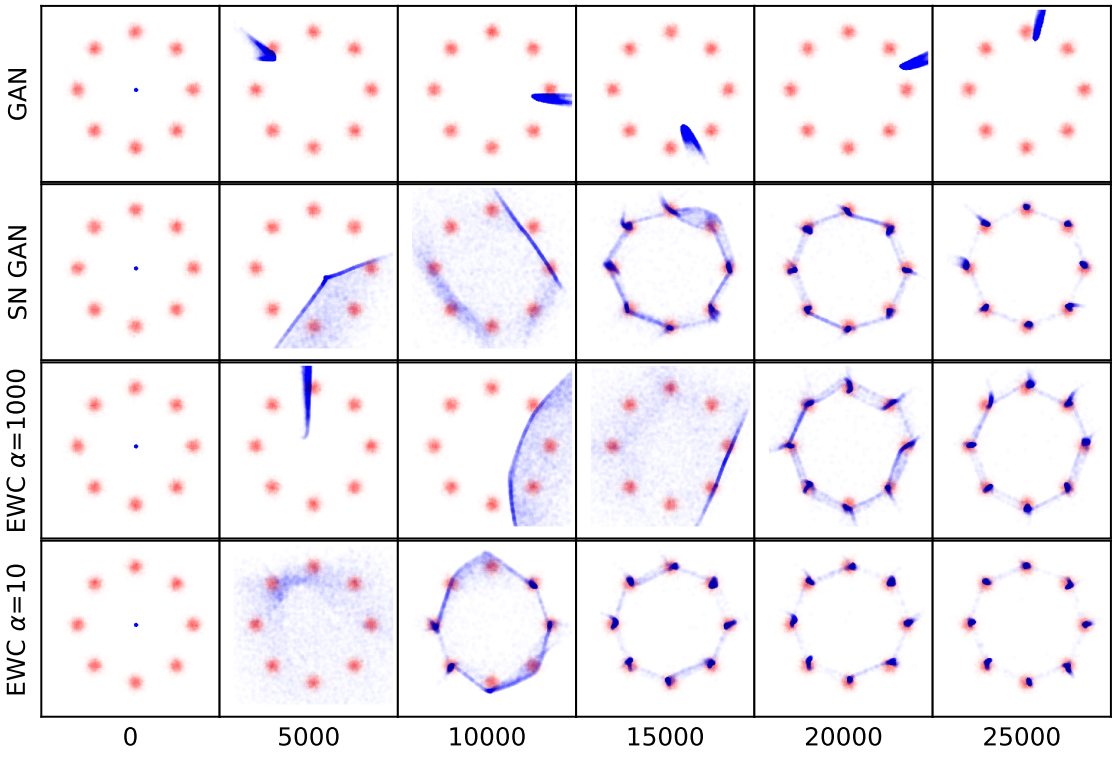


Figure 5.4: Each row shows the evolution of generator samples at 5000 training step intervals for GAN, SN-GAN, and EWC-GAN for two α values. The proposed EWC-GAN models have hyperparameters matching the corresponding α in Table 5.1. Each frame shows 10000 samples drawn from the true eight Gaussians mixture (red) and 10000 generator samples (blue).

We also show the generated image samples for CIFAR 10 and CelebA in Figure 5.5, and generated text samples for MS COCO Captions in Table 5.5.

Table 5.5: Sample sentence generations from EWC + textGAN

a couple of people are standing by some zebras in the background
the view of some benches near a gas station
a brown motorcycle standing next to a red fence
a bath room with a broken tank on the floor
red passenger train parked under a bridge near a river
some snow on the beach that is surrounded by a truck
a cake that has been perform in the background for takeoff
a view of a city street surrounded by trees
two giraffes walking around a field during the day
crowd of people lined up on motorcycles
two yellow sheep with a baby dog in front of other sheep
an intersection sits in front of a crowd of people
a red double decker bus driving down the street corner
an automobile driver stands in the middle of a snowy park
five people at a kitchen setting with a woman
there are some planes at the takeoff station
a passenger airplane flying in the sky over a cloudy sky
three aircraft loaded into an airport with a stop light
there is an animal walking in the water
an older boy with wine glasses in an office
two old jets are in the middle of london
three motorcycles parked in the shade of a crowd
group of yellow school buses parked on an intersection
a person laying on a sidewalk next to a sidewalk talking on a cell phone
a chef is preparing food with a sink and stainless steel appliances

(a) CIFAR 10

(b) CelebA

Figure 5.5: Generated image samples, drawn randomly from GANs with EWC regularization.

Chapter 6

Kernel-Based Approaches for Sequence Modeling: Connections to Neural Methods

6.1 Introduction

There has been significant recent effort directed at connecting deep learning to kernel machines [ARTP15, BM17, Mai16, WHSX16]. Specifically, it has been recognized that a deep neural network may be viewed as constituting a feature mapping $x \rightarrow \varphi_\theta(x)$, for input data $x \in \mathbb{R}^m$. The nonlinear function $\varphi_\theta(x)$, with model parameters θ, has an output that corresponds to a d-dimensional feature vector; $\varphi_\theta(x)$ may be viewed as a mapping of x to a Hilbert space $\mathcal{H}$, where $\mathcal{H} \subset \mathbb{R}^d$. The final layer of deep neural networks typically corresponds to an inner product $\omega^\intercal \varphi_\theta(x)$, with weight vector $\omega \in \mathcal{H}$; for a vector output, there are multiple ω, with $\omega_i^\intercal \varphi_\theta(x)$ defining the i-th component of the output. For example, in a deep convolutional neural network (CNN) [LBD$^+$89], $\varphi_\theta(x)$ is a function defined by the multiple convolutional layers, the output of which is a d-dimensional feature map; ω represents the fully-connected layer that imposes inner products on the feature map. Learning ω and θ, *i.e.*, the cumulative neural network parameters, may be interpreted as learning within a reproducing kernel Hilbert space (RKHS) [BTA04], with ω the function in $\mathcal{H}$; $\varphi_\theta(x)$ represents the mapping from the space of the input x to $\mathcal{H}$, with associated kernel $k_\theta(x, x') = \varphi_\theta(x)^\intercal \varphi_\theta(x')$, where x' is another input.

Insights garnered about neural networks from the perspective of kernel machines provide valuable theoretical underpinnings, helping to explain why such models work

well in practice. As an example, the RKHS perspective helps explain invariance and stability of deep models, as a consequence of the smoothness properties of an appropriate RKHS to variations in the input x [BM17, Mai16]. Further, such insights provide the opportunity for the development of new models.

Most prior research on connecting neural networks to kernel machines has assumed a single input x, *e.g.*, image analysis in the context of a CNN [ARTP15, BM17, Mai16]. However, the recurrent neural network (RNN) has also received renewed interest for analysis of sequential data. For example, long short-term memory (LSTM) [HS97, GSK+17] and the gated recurrent unit (GRU) [CvMG+14] have become fundamental elements in many natural language processing (NLP) pipelines [JZS15, CvMG+14, GH16]. In this context, a *sequence* of data vectors $(\dots, x_{t-1}, x_t, x_{t+1}, \dots)$ is analyzed, and the aforementioned single-input models are inappropriate.

In this paper, we extend to *recurrent* neural networks (RNNs) the concept of analyzing neural networks from the perspective of kernel machines. Leveraging recent work on recurrent kernel machines (RKMs) for sequential data [HS12], we make new connections between RKMs and RNNs, showing how RNNs may be constructed in terms of recurrent kernel machines, using simple filters. We demonstrate that these recurrent kernel machines are composed of a memory cell that is updated sequentially as new data come in, as well as in terms of a (distinct) hidden unit. A recurrent model that employs a memory cell and a hidden unit evokes ideas from the LSTM. However, within the recurrent kernel machine representation of a basic RNN, the rate at which memory fades with time is fixed. To impose adaptivity within the recurrent kernel machine, we introduce adaptive gating elements on the updated and prior components of the memory cell, and we also impose a gating network on the output of the model. We demonstrate that the result of this refinement of the recurrent kernel machine is

a model closely related to the LSTM, providing new insights on the LSTM and its connection to kernel machines.

Continuing with this framework, we also introduce new concepts to models of the LSTM type. The refined LSTM framework may be viewed as convolving learned filters across the input sequence and using the convolutional output to constitute the time-dependent memory cell. Multiple filters, possibly of different temporal lengths, can be utilized, like in the CNN. One recovers the CNN [LB95, ZZL15, Kim14] and Gated CNN [DFAG17] models of sequential data as special cases, by turning off elements of the new LSTM setup. From another perspective, we demonstrate that the new LSTM-like model may be viewed as introducing gated memory cells and feedback to a CNN model of sequential data.

In addition to developing the aforementioned models for sequential data, we demonstrate them in an extensive set of experiments, focusing on applications in natural language processing (NLP) and in analysis of multi-channel, time-dependent local field potential (LFP) recordings from mouse brains. Concerning the latter, we demonstrate marked improvements in performance of the proposed methods relative to recently-developed alternative approaches [LMM+17].

6.2 Recurrent Kernel Network

Consider a *sequence* of vectors $(\dots, x_{t-1}, x_t, x_{t+1}, \dots)$, with $x_t \in \mathbb{R}^m$. For a language model, x_t is the embedding vector for the t-th word w_t in a sequence of words. To model this sequence, we introduce $y_t = Uh_t$, with the recurrent hidden variable satisfying

$$h_t = f(W^{(x)}x_t + W^{(h)}h_{t-1} + b) \tag{6.1}$$

where $h_t \in \mathbb{R}^d$, $U \in \mathbb{R}^{V\times d}$, $W^{(x)} \in \mathbb{R}^{d\times m}$, $W^{(h)} \in \mathbb{R}^{d\times d}$, and $b \in \mathbb{R}^d$. In the context of a language model, the vector $y_t \in \mathbb{R}^V$ may be fed into a nonlinear function to

predict the next word w_{t+1} in the sequence. Specifically, the probability that w_{t+1} corresponds to $i \in \{1, \ldots, V\}$ in a vocabulary of V words is defined by element i of vector $\text{Softmax}(y_t + \beta)$, with bias $\beta \in \mathbb{R}^V$. In classification, such as the LFP-analysis example in Section 6.6, V is the number of classes under consideration.

We constitute the factorization $U = AE$, where $A \in \mathbb{R}^{V \times j}$ and $E \in \mathbb{R}^{j \times d}$, often with $j \ll V$. Hence, we may write $y_t = Ah'_t$, with $h'_t = Eh_t$; the columns of A may be viewed as time-invariant factor loadings, and h'_t represents a vector of dynamic factor scores. Let $z_t = [x_t, h_{t-1}]$ represent a column vector corresponding to the concatenation of x_t and h_{t-1}; then $h_t = f(W^{(z)} z_t + b)$ where $W^{(z)} = [W^{(x)}, W^{(h)}] \in \mathbb{R}^{d \times (d+m)}$. Computation of Eh_t corresponds to inner products of the rows of E with the vector h_t. Let $e_i \in \mathbb{R}^d$ be a column vector, with elements corresponding to row $i \in \{1, \ldots, j\}$ of E. Then component i of h'_t is

$$h'_{i,t} = e_i^\intercal h_t = e_i^\intercal f(W^{(z)} z_t + b) \tag{6.2}$$

We view $f(W^{(z)} z_t + b)$ as mapping z_t into a RKHS $\mathcal{H}$, and vector e_i is also assumed to reside within $\mathcal{H}$. We consequently assume

$$e_i = f(W^{(z)} \tilde{z}_i + b) \tag{6.3}$$

where $\tilde{z}_i = [\tilde{x}_i, \tilde{h}_0]$. Note that here $\tilde{h}_0$ also depends on index i, which we omit for simplicity; as discussed below, $\tilde{x}_i$ will play the primary role when performing computations.

$$e_i^\intercal h_t = e_i^\intercal f(W^{(z)} z_t + b) = f(W^{(z)} \tilde{z}_i + b)^\intercal f(W^{(z)} z_t + b) = k_\theta(\tilde{z}_i, z_t) \tag{6.4}$$

where $k_\theta(\tilde{z}_i, z_t) = h(\tilde{z}_i)^\intercal h(z_t)$ is a Mercer kernel [SS02]. Particular kernel choices correspond to different functions $f(W^{(z)} z_t + b)$, and θ is meant to represent kernel parameters that may be adjusted.

We initially focus on kernels of the form $k_\theta(\tilde{z}, z_t) = q_\theta(\tilde{z}^\intercal z_t) = \tilde{h}_1^\intercal h_t$,[1] where $q_\theta(\cdot)$ is a function of parameters θ, $h_t = h(z_t)$, and $\tilde{h}_1$ is the implicit latent vector associated with the inner product, *i.e.*, $\tilde{h}_1 = f(W^{(x)}\tilde{x} + W^{(h)}\tilde{h}_0 + b)$. As discussed below, we will not need to explicitly evaluate h_t or $\tilde{h}_1$ to evaluate the kernel, taking advantage of the recursive relationship in (6.1). In fact, depending on the choice of $q_\theta(\cdot)$, the hidden vectors may even be infinite-dimensional. However, because of the relationship $q_\theta(\tilde{z}^\intercal z_t) = \tilde{h}_1^\intercal h_t$, for rigorous analysis $q_\theta(\cdot)$ should satisfy Mercer's condition [Gen01, SS02].

The vectors $(\tilde{h}_1, \tilde{h}_0, \tilde{h}_{-1}, \dots)$ are assumed to satisfy the same recurrence setup as (6.1), with each vector in the associated sequence $(\tilde{x}_t, \tilde{x}_{t-1}, \dots)$ assumed to be the same $\tilde{x}_i$ at each time, *i.e.*, associated with e_i, $(\tilde{x}_t, \tilde{x}_{t-1}, \dots) \to (\tilde{x}_i, \tilde{x}_i, \dots)$. Stepping backwards in time three steps, for example, one may show

$$k_\theta(\tilde{z}_i, z_t) = q_\theta[\tilde{x}_i^\intercal x_t + q_\theta[\tilde{x}_i^\intercal x_{t-1} + q_\theta[\tilde{x}_i^\intercal x_{t-2} + q_\theta[\tilde{x}_i^\intercal x_{t-3} + \tilde{h}_{-4}^\intercal h_{t-4}]]]] \tag{6.5}$$

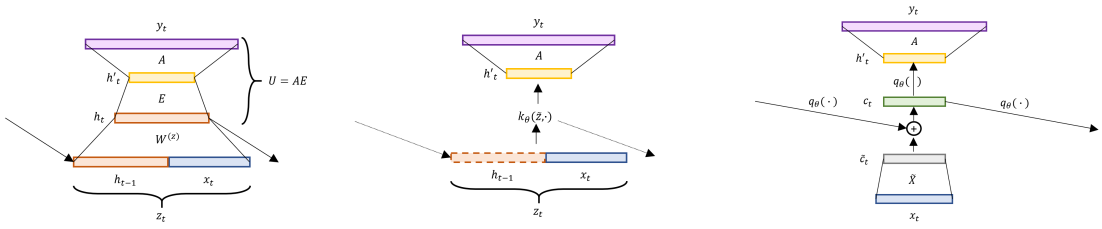
The inner product $\tilde{h}_{-4}^\intercal h_{t-4}$ encapsulates contributions for all times further backwards, and for a sequence of length N, $\tilde{h}_{-N}^\intercal h_{t-N}$ plays a role analogous to a bias. As discussed below, for stability the repeated application of $q_\theta(\cdot)$ yields diminishing (fading) contributions from terms earlier in time, and therefore for large N the impact of $\tilde{h}_{-N}^\intercal h_{t-N}$ on $k_\theta(\tilde{z}_i, z_t)$ is small.

The overall model may be expressed as

$$h'_t = q_\theta(c_t)\ , \quad c_t = \tilde{c}_t + q_\theta(c_{t-1})\ , \quad \tilde{c}_t = \tilde{X}x_t \tag{6.6}$$

where $c_t \in \mathbb{R}^j$ is a *memory cell* at time t, row i of $\tilde{X}$ corresponds to $\tilde{x}_i^\intercal$, and $q_\theta(c_t)$ operates pointwise on the components of c_t (see Figure 6.1). At the start of the

[1] One may also design recurrent kernels of the form $k_\theta(\tilde{z}, z_t) = q_\theta(\|\tilde{z} - z_t\|_2^2)$ [HS12], as for a Gaussian kernel, but if vectors x_t and filters $\tilde{x}_i$ are normalized (*e.g.*, $x_t^\intercal x_t = \tilde{x}_i^\intercal \tilde{x}_i = 1$), then $q_\theta(\|\tilde{z} - z_t\|_2^2)$ reduces to $q_\theta(\tilde{z}^\intercal z_t)$.



(a) (b) (c)

Figure 6.1: a) A traditional recurrent neural network (RNN), with the factorization $U = AE$. b) A recurrent kernel machine (RKM), with an implicit hidden state and recurrence through recursion. c) The recurrent kernel machine expressed in terms of a memory cell.

sequence of length N, $q_\theta(c_{t-N})$ may be seen as a vector of biases, effectively corresponding to $\tilde{h}_N^\intercal h_{t-N}$; we henceforth omit discussion of this initial bias for notational simplicity, and because for sufficiently large N its impact on h'_t is small.

Note that via the recursive process by which c_t is evaluated in (6.6), the kernel evaluations reflected by $q_\theta(c_t)$ are defined entirely by the elements of the sequence $(\tilde{c}_t, \tilde{c}_{t-1}, \tilde{c}_{t-2}, \dots)$. Let $\tilde{c}_{i,t}$ represent the i-th component in vector $\tilde{c}_t$, and define $x_{\leq t} = (x_t, x_{t-1}, x_{t-2}, \dots)$. Then the sequence $(\tilde{c}_{i,t}, \tilde{c}_{i,t-1}, \tilde{c}_{i,t-2}, \dots)$ is specified by convolving in time $\tilde{x}_i$ with $x_{\leq t}$, denoted $\tilde{x}_i * x_{\leq t}$. Hence, the j components of the sequence $(\tilde{c}_t, \tilde{c}_{t-1}, \tilde{c}_{t-2}, \dots)$ are completely specified by convolving $x_{\leq t}$ with each of the j filters, $\tilde{x}_i$, $i \in \{1, \dots, j\}$, *i.e.*, taking an inner product of $\tilde{x}_i$ with the vector in $x_{\leq t}$ at each time point.

In (6.4) we represented $h'_{i,t} = q_\theta(c_{i,t})$ as $h'_{i,t} = k_\theta(\tilde{z}_i, z_t)$; now, because of the recursive form of the model in (6.1), and because of the assumption $k_\theta(\tilde{z}_i, z_t) = q_\theta(\tilde{z}_i^\intercal z_t)$, we have demonstrated that we may express the kernel equivalently as $k_\theta(\tilde{x}_i * x_{\leq t})$, to underscore that it is defined entirely by the elements at the output of the convolution $\tilde{x}_i * x_{\leq t}$. Hence, we may express component i of h'_t as $h'_{i,t} = k_\theta(\tilde{x}_i * x_{\leq t})$.

Component $l \in \{1, \dots, V\}$ of $y_t = Ah'_t$ may be expressed

$$y_{l,t} = \sum_{i=1}^{j} A_{l,i} k_\theta(\tilde{x}_i * x_{\leq t}) \tag{6.7}$$

where $A_{l,i}$ represents component (l, i) of matrix A. Considering (6.7), the connection of an RNN to an RKHS is clear, as made explicit by the kernel $k_\theta(\tilde{x}_i * x_{\leq t})$. The RKHS is manifested for the final output y_t, with the hidden h_t now absorbed within the kernel, via the inner product (6.4). The feedback imposed via latent vector h_t is constituted via update of the memory cell $c_t = \tilde{c}_t + q_\theta(c_{t-1})$ used to evaluate the kernel.

Rather than evaluating y_t as in (6.7), it will prove convenient to return to (6.6). Specifically, we may consider modifying (6.6) by injecting further feedback via h'_t, augmenting (6.6) as

$$h'_t = q_\theta(c_t) \ , \quad c_t = \tilde{c}_t + q_\theta(c_{t-1}) \ , \quad \tilde{c}_t = \tilde{X} x_t + \tilde{H} h'_{t-1} \tag{6.8}$$

where $\tilde{H} \in \mathbb{R}^{j \times j}$, and recalling $y_t = Ah'_t$ (see Figure 6.2a for illustration). In (6.8) the input to the kernel is dependent on the input elements $(x_t, x_{t-1}, \dots)$ and is now also a function of the kernel *outputs* at the previous time, via h'_{t-1}. However, note that h'_t is still specified entirely by the elements of $\tilde{x}_i * x_{\leq t}$, for $i \in \{1, \dots, j\}$.

6.3 Choice of Recurrent Kernels & Introduction of Gating Networks

6.3.1 Fixed Kernel Parameters & Time-invariant Memory-cell Gating

The function $q_\theta(\cdot)$ discussed above may take several forms, the simplest of which is a linear kernel, with which (6.8) takes the form

$$h'_t = c_t \ , \quad c_t = \sigma_i^2 \tilde{c}_t + \sigma_f^2 c_{t-1} \ , \quad \tilde{c}_t = \tilde{X} x_t + \tilde{H} h'_{t-1} \tag{6.9}$$

where σ_i^2 and σ_f^2 (using analogous notation from [HS12]) are scalars, with $\sigma_f^2 < 1$ for stability. The scalars σ_i^2 and σ_f^2 may be viewed as *static* (*i.e.*, time-invariant) gating elements, with σ_i^2 controlling weighting on the new input element to the memory cell, and σ_f^2 controlling how much of the prior memory unit is retained; given $\sigma_f^2 < 1$, this means information from previous time steps tends to fade away and over time is largely forgotten. However, such a kernel leads to time-invariant decay of memory: the contribution $\tilde{c}_{t-N}$ from N steps before to the current memory c_t is $(\sigma_i \sigma_f^N)^2 \tilde{c}_{t-N}$, meaning that it decays at a constant exponential rate. Because the information contained at each time step can vary, this can be problematic. This suggests augmenting the model, with *time-varying* gating weights, with memory-component dependence on the weights, which we consider below.

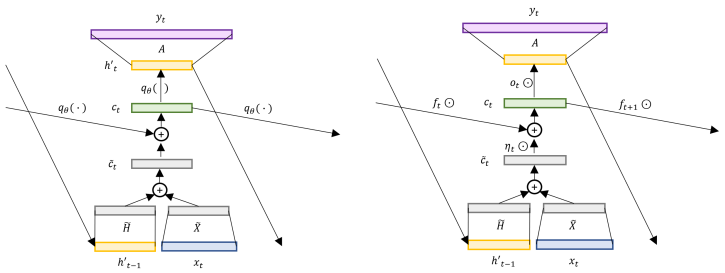
6.3.2 Dynamic Gating Networks & LSTM-like Model

Recent work has shown that dynamic gating can be seen as making a recurrent network quasi-invariant to temporal warpings [TO18]. Motivated by the form of the model in (6.9) then, it is natural to impose dynamic versions of σ_i^2 and σ_f^2; we also introduce dynamic gating at the output of the hidden vector. This yields the model:

$$h'_t = o_t \odot c_t \ , \qquad c_t = \eta_t \odot \tilde{c}_t + f_t \odot c_{t-1} \ , \qquad \tilde{c}_t = W_c z'_t \tag{6.10}$$

$$o_t = \sigma(W_o z'_t + b_o) \ , \qquad \eta_t = \sigma(W_\eta z'_t + b_\eta) \ , \qquad f_t = \sigma(W_f z'_t + b_f) \tag{6.11}$$

where $z'_t = [x_t, h'_{t-1}]$, and W_c encapsulates $\tilde{X}$ and $\tilde{H}$. In (6.10)-(6.11) the symbol $\odot$ represents a pointwise vector product (Hadamard); W_c, W_o, W_η and W_f are weight matrices; b_o, b_η and b_f are bias vectors; and $\sigma(\alpha) = 1/(1 + \exp(-\alpha))$. In (6.10), η_t and f_t play dynamic counterparts to σ_i^2 and σ_f^2, respectively. Further, o_t, η_t and f_t are *vectors*, constituting vector-component-dependent gating. Note that starting from a recurrent kernel machine, we have thus derived a model closely resembling the LSTM. We call this model RKM-LSTM (see Figure 6.2).



(a) (b)

Figure 6.2: a) Recurrent kernel machine, with feedback, as defined in (6.8). b) Making a linear kernel assumption and adding input, forget, and output gating, this model becomes the RKM-LSTM.

Concerning the update of the hidden state, $h'_t = o_t \odot c_t$ in (6.10), one may also consider appending a hyperbolic-tangent tanh nonlinearity: $h'_t = o_t \odot \tanh(c_t)$. However, recent research has suggested *not* using such a nonlinearity [LLZ17, DFAG17, CFB+18], and this is a natural consequence of our recurrent kernel analysis. Using $h'_t = o_t \odot \tanh(c_t)$, the model in (6.10) and (6.11) is in the form of the LSTM, except without the nonlinearity imposed on the memory cell $\tilde{c}_t$, while in the LSTM a tanh nonlinearity (and biases) is employed when updating the memory cell [HS97, GSK+17], *i.e.*, for the LSTM $\tilde{c}_t = \tanh(W_c z'_t + b_c)$. If $o_t = 1$ for all time t (no output gating network), and if $\tilde{c}_t = W_c x_t$ (no dependence on h'_{t-1} for update of the memory cell), this model reduces to the recurrent additive network (RAN) [LLZ17].

While separate gates η_t and f_t were constituted in (6.10) and (6.11) to operate on the new and prior composition of the memory cell, one may also also consider a simpler model with memory cell updated $c_t = (1 - f_t) \odot \tilde{c}_t + f_t \odot c_{t-1}$; this was referred to as having a Coupled Input and Forget Gate (CIFG) in [GSK+17]. In such a model, the decisions of what to add to the memory cell and what to forget are made jointly, obviating the need for a separate input gate η_t. We call this variant RKM-CIFG.

6.4 Extending the Filter Length

6.4.1 Generalized Form of Recurrent Model

Consider a generalization of (6.1):

$$h_t = f(W^{(x_0)}x_t + W^{(x_{-1})}x_{t-1} + \cdots + W^{(x_{-n+1})}x_{t-n+1} + W^{(h)}h_{t-1} + b) \tag{6.12}$$

where $W^{(x_\cdot)} \in \mathbb{R}^{d\times m}$, $W^{(h)} \in \mathbb{R}^{d\times d}$, and therefore the update of the hidden state h_t[2] depends on data observed $n \geq 1$ time steps prior, and also on the previous hidden state h_{t-1}. Analogous to (6.3), we may express

$$e_i = f(W^{(x_0)}\tilde{x}_{i,0} + W^{(x_{-1})}\tilde{x}_{i,-1} + \cdots + W^{(x_{-n+1})}\tilde{x}_{i,-n+1} + W^{(h)}\tilde{h}_i + b) \tag{6.13}$$

The inner product $f(W^{(x_0)}x_t + W^{(x_{-1})}x_{t-1} + \cdots + W^{(x_{-n+1})}x_{t-n+1} + W^{(h)}h_{t-1} + b)^\intercal f(W^{(x_0)}\tilde{x}_{i,0} + W^{(x_{-1})}\tilde{x}_{i,-1} + \cdots + W^{(x_{-n+1})}\tilde{x}_{i,-n+1} + W^{(h)}\tilde{h}_i + b)$ is assumed represented by a Mercer kernel, and $h'_{i,t} = e_i^\intercal h_t$.

Let $X_t = (x_t, x_{t-1}, \ldots, x_{t-n+1}) \in \mathbb{R}^{m\times n}$ be an n-gram input with zero padding if $t < (n-1)$, and $\tilde{\boldsymbol{X}} = (\tilde{X}_0, \tilde{X}_{-1}, \ldots, \tilde{X}_{-n+1})$ be n sets of filters, with the i-th rows of $\tilde{X}_0, \tilde{X}_{-1}, \ldots, \tilde{X}_{-n+1}$ collectively represent the i-th n-gram filter, with $i \in \{1, \ldots, j\}$. Extending Section 6.2, the kernel is defined

$$h'_t = q_\theta(c_t) \quad , \quad c_t = \tilde{c}_t + q_\theta(c_{t-1}) \quad , \quad \tilde{c}_t = \tilde{\boldsymbol{X}} \cdot X_t \tag{6.14}$$

where $\tilde{\boldsymbol{X}} \cdot X_t \equiv \tilde{X}_0 x_t + \tilde{X}_{-1}x_{t-1} + \cdots + \tilde{X}_{-n+1}x_{t-n+1} \in \mathbb{R}^j$. Note that $\tilde{\boldsymbol{X}} \cdot X_t$ corresponds to the t-th component output from the n-gram convolution of the filters $\tilde{\boldsymbol{X}}$ and the input sequence; therefore, similar to Section 6.2, we represent $h'_t = q_\theta(c_t)$ as $h'_t = k_\theta(\tilde{\boldsymbol{X}} * x_{\leq t})$, emphasizing that the kernel evaluation is a function of outputs of the convolution $\tilde{\boldsymbol{X}} * x_{\leq t}$, here with n-gram filters. Like in the CNN [LB95, ZZL15, Kim14], different filter lengths (and kernels) may be considered to constitute different components of the memory cell.

[2]Note that while the same symbol is used as in (6.12), h_t clearly takes on a different meaning when $n > 1$.

6.4.2 Linear Kernel, CNN and Gated CNN

For the linear kernel discussed in connection to (6.9), equation (6.14) becomes

$$h'_t = c_t = \sigma_i^2(\tilde{\boldsymbol{X}} \cdot X_t) + \sigma_f^2 h'_{t-1} \tag{6.15}$$

For the special case of $\sigma_f^2 = 0$ and σ_i^2 equal to a constant (*e.g.*, $\sigma_i^2 = 1$), (6.15) reduces to a convolutional neural network (CNN), with a nonlinear operation typically applied subsequently to h'_t.

Rather than setting σ_i^2 to a constant, one may impose *dynamic* gating, yielding the model (with $\sigma_f^2 = 0$)

$$h'_t = \eta_t \odot (\tilde{\boldsymbol{X}} \cdot X_t) \quad , \qquad \eta_t = \sigma(\tilde{\boldsymbol{X}}_\eta \cdot X_t + b_\eta) \tag{6.16}$$

where $\tilde{\boldsymbol{X}}_\eta$ are distinct convolutional filters for calculating η_t, and b_η is a vector of biases. The form of the model in (6.16) corresponds to the Gated CNN [DFAG17], which we see as a a special case of the recurrent model with linear kernel, and dynamic kernel weights (and without feedback, *i.e.*, $\sigma_f^2 = 0$). Note that in (6.16) a nonlinear function is *not* imposed on the output of the convolution $\tilde{\boldsymbol{X}} \cdot X_t$, there is only dynamic gating via multiplication with η_t; the advantages of which are discussed in [DFAG17]. Further, the n-gram input considered in (6.12) need not be consecutive. If spacings between inputs of more than 1 are considered, then the dilated convolution (*e.g.*, as used in [vdODZ$^+$16]) is recovered.

6.4.3 Feedback and the Generalized LSTM

Now introducing feedback into the memory cell, the model in (6.8) is extended to

$$h'_t = q_\theta(c_t) \ , \quad c_t = \tilde{c}_t + q_\theta(c_{t-1}) \ , \quad \tilde{c}_t = \tilde{\boldsymbol{X}} \cdot X_t + \tilde{H} h'_{t-1} \tag{6.17}$$

Again motivated by the linear kernel, generalization of (6.17) to include gating networks is

$$h'_t = o_t \odot c_t \ , \quad c_t = \eta_t \odot \tilde{c}_t + f_t \odot c_{t-1} \ , \quad \tilde{c}_t = \tilde{\boldsymbol{X}} \cdot X_t + \tilde{H} h'_{t-1} \tag{6.18}$$

$$o_t = \sigma(\tilde{\boldsymbol{X}}_o \cdot X_t + \tilde{W}_o h'_{t-1} + b_o), \ \eta_t = \sigma(\tilde{\boldsymbol{X}}_\eta \cdot X_t + \tilde{W}_\eta h'_{t-1} + b_\eta), \ f_t = \sigma(\tilde{\boldsymbol{X}}_f \cdot X_t + \tilde{W}_f h'_{t-1} + b_f) \tag{6.19}$$

where $y_t = A h'_t$ and $\tilde{\boldsymbol{X}}_o$, $\tilde{\boldsymbol{X}}_\eta$, and $\tilde{\boldsymbol{X}}_f$ are separate sets of n-gram convolutional filters akin to $\tilde{\boldsymbol{X}}$. As an n-gram generalization of (6.10)-(6.11), we refer to (6.18)-(6.19) as an n-gram RKM-LSTM.

The model in (6.18) and (6.19) is similar to the LSTM, with important differences: (i) there is not a nonlinearity imposed on the update to the memory cell, $\tilde{c}_t$, and therefore there are also no biases imposed on this cell update; (ii) there is no nonlinearity on the output; and (iii) via the convolutions with $\tilde{\boldsymbol{X}}$, $\tilde{\boldsymbol{X}}_o$, $\tilde{\boldsymbol{X}}_\eta$, and $\tilde{\boldsymbol{X}}_f$, the memory cell can take into account n-grams, and the length of such sequences n_i may vary as a function of the element of the memory cell.

6.5 Related Work

In our development of the kernel perspective of the RNN, we have emphasized that the form of the kernel $k_\theta(\tilde{z}_i, z_t) = q_\theta(\tilde{z}_i^\intercal z_t)$ yields a recursive means of kernel evaluation that is only a function of the elements at the output of the convolutions $\tilde{X} * x_{\leq t}$ or $\tilde{\boldsymbol{X}} * x_{\leq t}$, for 1-gram and $(n > 1)$-gram filters, respectively. This underscores that at the heart of such models, one performs convolutions between the sequence of data $(\dots, x_{t+1}, x_t, x_{t-1}, \dots)$ and filters $\tilde{X}$ or $\tilde{\boldsymbol{X}}$. Consideration of filters of length greater than one (in time) yields a generalization of the traditional LSTM. The dependence of such models entirely on convolutions of the data sequence and filters is evocative of CNN and Gated CNN models for text [LB95, ZZL15, Kim14, DFAG17], with this

made explicit in Section 6.4.2 as a special case.

The Gated CNN in (6.16) and the generalized LSTM in (6.18)-(6.19) both employ dynamic gating. However, the generalized LSTM explicitly employs a memory cell (and feedback), and hence offers the potential to leverage long-term memory. While memory affords advantages, a noted limitation of the LSTM is that computation of h'_t is sequential, undermining parallel computation, particularly while training [DFAG17, VSP+17]. In the Gated CNN, h'_t comes directly from the output of the gated convolution, allowing parallel fitting of the model to time-dependent data. While the Gated CNN does not employ recurrence, the filters of length $n > 1$ do leverage extended temporal dependence. Further, via deep Gated CNNs [DFAG17], the *effective* support of the filters at deeper layers can be expansive.

Recurrent kernels of the form $k_\theta(\tilde{z}, z_t) = q_\theta(\tilde{z}^\intercal z_t)$ were also developed in [HS12], but with the goal of extending recurrent kernel machines to sequential inputs, rather than making connections with RNNs. The formulation in Section 6.2 has two important differences with that prior work. First, we employ the *same* vector $\tilde{x}_i$ for all shift positions t of the inner product $\tilde{x}_i^\intercal x_t$. By contrast, in [HS12] effectively infinite-dimensional filters are used, because the filter $\tilde{x}_{t,i}$ changes with t. This makes implementation computationally impractical, necessitating truncation of the long temporal filter. Additionally, the feedback of h'_t in (6.8) was not considered, and as discussed in Section 6.3.2, our proposed setup yields natural connections to long short-term memory (LSTM) [HS97, GSK+17].

Prior work analyzing neural networks from an RKHS perspective has largely been based on the feature mapping $\varphi_\theta(x)$ and the weight ω [ARTP15, BM17, Mai16, WHSX16]. For the recurrent model of interest here, function $h_t = f(W^{(x)}x_t + W^{(h)}h_{t-1} + b)$ plays a role like $\varphi_\theta(x)$ as a mapping of an input x_t to what may be viewed as a feature vector h_t. However, because of the recurrence, h_t is a function

of $(x_t, x_{t-1}, \dots)$ for an arbitrarily long time period prior to time t:

$$h_t(x_t, x_{t-1}, \dots) = f(W^{(x)}x_t + b + W^{(h)}f(W^{(x)}x_{t-1} + b + W^{(h)}f(W^{(x)}x_{t-2} + b + \dots))) \tag{6.20}$$

However, rather than explicitly working with $h_t(x_t, x_{t-1}, \dots)$, we focus on the kernel $k_\theta(\tilde{z}_i, z_t) = q_\theta(\tilde{z}_i^\intercal z_t) = k_\theta(\tilde{x}_i * x_{\leq t})$.

The authors of [LJBJ17] derive recurrent neural networks from a string kernel by replacing the exact matching function with an inner product and assume the decay factor to be a nonlinear function. Convolutional neural networks are recovered by replacing a pointwise multiplication with addition. However, the formulation cannot recover the standard LSTM formulation, nor is there a consistent formulation for all the gates. The authors of [RKF19] introduce a kernel-based update rule to approximate backpropagation through time (BPTT) for RNN training, but still follow the standard RNN structure.

Previous works have considered recurrent models with n-gram inputs as in (6.12). For example, strongly-typed RNNs [BG16] consider bigram inputs, but the previous input x_{t-1} is used as a replacement for h_{t-1} rather than in conjunction, as in our formulation. Quasi-RNNs [BMXS17] are similar to [BG16], but generalize them with a convolutional filter for the input and use different nonlinearities. Inputs corresponding to n-grams have also been implicitly considered by models that use convolutional layers to extract features from n-grams that are then fed into a recurrent network (*e.g.*, [CL16, WYLZ16, ZSLL15]). Relative to (6.18), these models contain an extra nonlinearity $f(\cdot)$ from the convolution and projection matrix $W^{(x)}$ from the recurrent cell, and no longer recover the CNN [LB95, ZZL15, Kim14] or Gated CNN [DFAG17] as special cases.

6.6 Experiments

In the following experiments, we consider several model variants, with nomenclature as follows. The ***n*-gram LSTM** developed in Sec. 6.4.3 is a generalization of the standard LSTM [HS97] (for which $n = 1$). We denote **RKM-LSTM** (recurrent kernel machine LSTM) as corresponding to (6.10)-(6.11), which resembles the n-gram LSTM, but without a tanh nonlinearity on the cell update $\tilde{c}_t$ or emission c_t. We term **RKM-CIFG** as a RKM-LSTM with $\eta_t = 1 - f_t$, as discussed in Section 6.3.2. **Linear Kernel w/ o_t** corresponds to (6.10)-(6.11) with $\eta_t = \sigma_i^2$ and $f_t = \sigma_f^2$, with σ_i^2 and σ_f^2 time-invariant constants; this corresponds to a linear kernel for the update of the memory cell, and dynamic gating on the output, via o_t. We also consider the same model without dynamic gating on the output, *i.e.*, $o_t = 1$ for all t (with a tanh nonlinearity on the output), which we call **Linear Kernel**. The **Gated CNN** corresponds to the model in [DFAG17], which is the same as Linear Kernel w/ o_t, but with $\sigma_f^2 = 0$ (*i.e.*, no memory). Finally, we consider a **CNN** model [LB95], that is the same as the Linear Kernel model, but without feedback or memory, *i.e.*, $z'_t = x_t$ and $\sigma_f^2 = 0$. For all of these, we may also consider an n-gram generalization as introduced in Section 6.4. For example, a 3-gram RKM-LSTM corresponds to (6.18)-(6.19), with length-3 convolutional filters in the time dimension. The models are summarized in Table 6.1. All experiments are run on a single NVIDIA Titan X GPU.

Document Classification We show results for several popular document classification datasets [ZZL15] in Table 6.2. The AGNews and Yahoo! datasets are topic classification tasks, while Yelp Full is sentiment analysis and DBpedia is ontology classification. The same basic network architecture is used for all models, with the only difference being the choice of recurrent cell, which we make single-layer and unidirectional. Hidden representations h'_t are aggregated with mean pooling across

Table 6.1: Model variants under consideration, assuming 1-gram inputs. Concatenating additional inputs $x_{t-1}, \ldots, x_{t-n+1}$ to z'_t in the **Input** column yields the corresponding n-gram model. Number of model parameters are shown for input $x_t \in \mathbb{R}^m$ and output $h'_t \in \mathbb{R}^d$.

Model	Parameters	Input	Cell	Output
LSTM [HS97]	$(nm+d)(4d)$	$z'_t = [x_t, h'_{t-1}]$	$c_t = \eta_t \odot \tanh(\tilde{c}_t) + f_t \odot c_{t-1}$	$h'_t = o_t \odot \tanh(c_t)$
RKM-LSTM	$(nm+d)(4d)$	$z'_t = [x_t, h'_{t-1}]$	$c_t = \eta_t \odot \tilde{c}_t + f_t \odot c_{t-1}$	$h'_t = o_t \odot c_t$
RKM-CIFG	$(nm+d)(3d)$	$z'_t = [x_t, h'_{t-1}]$	$c_t = (1-f_t) \odot \tilde{c}_t + f_t \odot c_{t-1}$	$h'_t = o_t \odot c_t$
Linear Kernel w/ o_t	$(nm+d)(2d)$	$z'_t = [x_t, h'_{t-1}]$	$c_t = \sigma_i^2 \tilde{c}_t + \sigma_f^2 c_{t-1}$	$h'_t = o_t \odot c_t$
Linear Kernel	$(nm+d)(d)$	$z'_t = [x_t, h'_{t-1}]$	$c_t = \sigma_i^2 \tilde{c}_t + \sigma_f^2 c_{t-1}$	$h'_t = \tanh(c_t)$
Gated CNN [DFAG17]	$(nm)(2d)$	$z'_t = x_t$	$c_t = \sigma_i^2 \tilde{c}_t$	$h'_t = o_t \odot c_t$
CNN [LB95]	$(nm)(d)$	$z'_t = x_t$	$c_t = \sigma_i^2 \tilde{c}_t$	$h'_t = \tanh(c_t)$

time, followed by two fully connected layers, with the second having output size corresponding to the number of classes of the dataset. We use 300-dimensional GloVe [PSM14] as our word embedding initialization and set the dimensions of all hidden units to 300. We follow the same preprocessing procedure as in [WLW+18]. Layer normalization [BKH16] is performed after the computation of the cell state c_t. For the Linear Kernel w/ o_t and the Linear Kernel, we set[3] $\sigma_i^2 = \sigma_f^2 = 0.5$.

Notably, the derived RKM-LSTM model performs comparably to the standard LSTM model across all considered datasets. We also find the CIFG version of the RKM-LSTM model to have similar accuracy. As the recurrent model becomes less sophisticated with regard to gating and memory, we see a corresponding decrease in classification accuracy. This decrease is especially significant for Yelp Full, which requires a more intricate comprehension of the entire text to make a correct prediction. This is in contrast to AGNews and DBpedia, where the success of the 1-gram CNN indicates that simple keyword matching is sufficient to do well. We also observe that generalizing the model to consider n-gram inputs typically improves performance; the highest accuracies for each dataset were achieved by an n-gram model.

Language Modeling We also perform experiments on popular word-level

[3] σ_i^2 and σ_f^2 can also be learned, but we found this not to have much effect on the final performance.

Table 6.2: Document classification accuracy for 1-gram and 3-gram versions of various models. Total parameters of each model are shown, excluding word embeddings and the classifier.

Model	Parameters		AGNews		DBpedia		Yahoo!		Yelp Full	
	1-gram	3-gram	1-gram	3-gram	1-gram	3-gram	1-gram	3-gram	1-gram	3-gram
LSTM	720K	1.44M	91.82	**92.46**	98.98	98.97	**77.74**	77.72	**66.27**	66.37
RKM-LSTM	720K	1.44M	91.76	92.28	98.97	99.00	77.70	77.72	65.92	**66.43**
RKM-CIFG	540K	1.08M	**92.29**	92.39	**98.99**	**99.05**	77.71	**77.91**	65.93	65.92
Linear Kernel w/ o_t	360K	720K	92.07	91.49	98.96	98.94	77.41	77.53	65.35	65.94
Linear Kernel	180K	360K	91.62	91.50	98.65	98.77	76.93	76.53	61.18	62.11
Gated CNN [DFAG17]	180K	540K	91.54	91.78	98.37	98.77	72.92	76.66	60.25	64.30
CNN [LB95]	90K	270K	91.20	91.53	98.17	98.52	72.51	75.97	59.77	62.08

Table 6.3: Language model perplexity (PPL) on validation and test sets of the Penn Treebank and Wikitext-2 language modeling tasks.

Model	PTB		Wikitext-2	
	PPL valid	PPL test	PPL valid	PPL test
LSTM [HS97, MKS18]	61.2	58.9	68.74	65.68
RKM-LSTM	**60.3**	**58.2**	**67.85**	**65.22**
RKM-CIFG	61.9	59.5	69.12	66.03
Linear Kernel w/ o_t	72.3	69.7	84.23	80.21

language generation datasets Penn Tree Bank (PTB) [MSM93] and Wikitext-2 [MXBS17], reporting validation and test perplexities (PPL) in Table 6.3. We adopt AWD-LSTM [MKS18] as our base model[4], replacing the standard LSTM with RKM-LSTM, RKM-CIFG, and Linear Kernel w/ o_t to do our comparison. We keep all other hyperparameters the same as the default. Here we consider 1-gram filters, as they performed best for this task; given that the datasets considered here are smaller than those for the classification experiments, 1-grams are less likely to overfit. Note that the static gating on the update of the memory cell (Linear Kernel w/ o_t) does considerably worse than the models with dynamic input and forget gates on the memory cell. The RKM-LSTM model consistently outperforms the traditional LSTM, again showing that the models derived from recurrent kernel machines work well in practice for the data considered.

[4]We use the official codebase `https://github.com/salesforce/awd-lstm-lm` and report experiment results before two-step fine-tuning.

Table 6.4: Mean leave-one-out classification accuracies for mouse LFP data. For each model, ($n = 40$)-gram filters are considered, and the number of filters in each model is 30.

Model	n-gram LSTM	RKM-LSTM	RKM-CIFG	Linear Kernel w/ o_t	Linear Kernel	Gated CNN [DFAG17]	CNN [LMM+17]
Accuracy	**80.24**	79.02	77.58	76.11	73.13	76.02	73.40

LFP Classification We perform experiments on a Local Field Potential (LFP) dataset. The LFP signal is multi-channel time series recorded inside the brain to measure neural activity. The LFP dataset used in this work contains recordings from 29 mice (wild-type or CLOCKΔ19 [vEMY13]), while the mice were (i) in their home cages, (ii) in an open field, and (iii) suspended by their tails. There are a total of $m = 11$ channels and the sampling rate is 1000Hz. The goal of this task is to predict the state of a mouse from a 1 second segment of its LFP recording as a 3-way classification problem. In order to test the model generalizability, we perform leave-one-out cross-validation testing: data from each mouse is left out as testing iteratively while the remaining mice are used as training.

SyncNet [LMM+17] is a CNN model with specifically designed wavelet filters for neural data. We incorporate the SyncNet form of n-gram convolutional filters into our recurrent framework (we have *parameteric* n-gram convolutional filters, with parameters learned). As was demonstrated in Section 6.4.2, the CNN is a memory-less special case of our derived generalized LSTM. An illustration of the modified model (Figure 6.3) can be found in Appendix 6.7, along with other further details on SyncNet.

While the filters of SyncNet are interpretable and can prevent overfitting (because they have a small number of parameters), the same kind of generalization to an n-gram LSTM can be made without increasing the number of learned parameters. We do so for all of the recurrent cell types in Table 6.1, with the CNN correspond-

ing to the original SyncNet model. Compared to the original SyncNet model, our newly proposed models can jointly consider the time dependency within the whole signal. The mean classification accuracies across all mice are compared in Table 6.4, where we observe substantial improvements in prediction accuracy through the addition of memory cells to the model. Thus, considering the time dependency in the neural signal appears to be beneficial for identifying hidden patterns. Classification performances per subject (Figure 6.4) can be found in Appendix A.

6.7 Conclusions

The principal contribution of this paper is a new perspective on gated RNNs, leveraging concepts from recurrent kernel machines. From that standpoint, we have derived a model closely connected to the LSTM [HS97, GSK+17] (for convolutional filters of length one), and have extended such models to convolutional filters of length greater than one, yielding a generalization of the LSTM. The CNN [LB95, ZZL15, Kim14], Gated CNN [DFAG17] and RAN [LLZ17] models are recovered as special cases of the developed framework. We have demonstrated the efficacy of the derived models on NLP and neuroscience tasks, for which our RKM variants show comparable or better performance than the LSTM. In particular, we observe that extending LSTM variants with convolutional filters of length greater than one can significantly improve the performance in LFP classification relative to recent prior work.

Appendix

A. More Details of the LFP Experiment

In this section, we provide more details on the Sync-RKM model. In order to incorporate the SyncNet model [LMM+17] into our framework, the weight $W^{(x)} = \left[W^{(x_0)}, W^{(x_{-1})}, \cdots, W^{(x_{-n+1})}\right]$ defined in Eq. (6.12) is parameterized as wavelet filters. If there is a total of K filters, then $\boldsymbol{W}^{(x)}$ is of size $K \times C \times n$.

Specifically, suppose the n-gram input data at time t is given as $\boldsymbol{X}_t = [\boldsymbol{x}_{t-n+1}, \cdots, \boldsymbol{x}_t] \in \mathbb{R}^{C \times n}$ with channel number C and window size n. The k-th filter for channel c can be written as

$$\boldsymbol{W}_{kc}^{(x)} = \alpha_{kc} \cos\left(\omega_k \boldsymbol{t} + \phi_{kc}\right) \exp(-\beta_k \boldsymbol{t}^2) \tag{6.21}$$

$\boldsymbol{W}_{kc}^{(x)}$ has the form of the Morlet wavelet base function. Parameters to be learned are α_{kc}, ω_k, ϕ_{kc} and β_k for $c = 1, \cdots C$ and $k = 1, \cdots, K$. $\boldsymbol{t}$ is a time grid of length n, which is a constant vector. In the recurrent cell, each $\boldsymbol{W}_{kc}^{(x)}$ is convolved with the c-th channel of $\boldsymbol{X}_t$ using 1-d convolution. Figure 6.3 gives the framework of this Sync-RKM model. For more details of how the filter works, please refer to the original work [LMM+17].

When applying the Sync-RKM model on LFP data, we choose the window size as $n = 40$ to consider the time dependencies in the signal. Since the experiment is performed by treating each mouse as test iteratively, we show the subject-wise classification accuracy in Figure 6.4. The proposed model does consistently better across nearly all subjects.

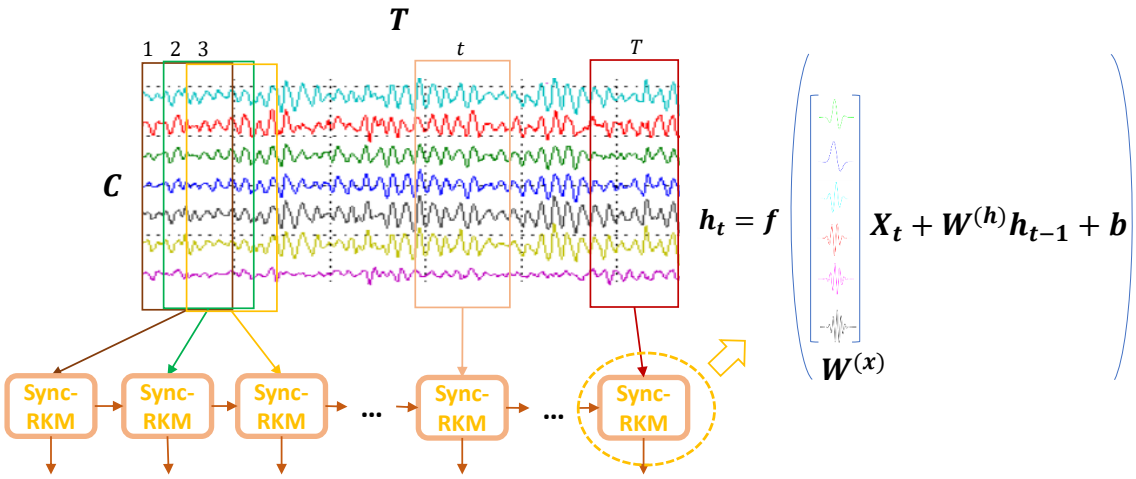


Figure 6.3: Illustration of the proposed model with SyncNet filters. The input LFP signal is given by the $C \times T$ matrix. The SyncNet filters (right) are applied on signal chunks at each time step.

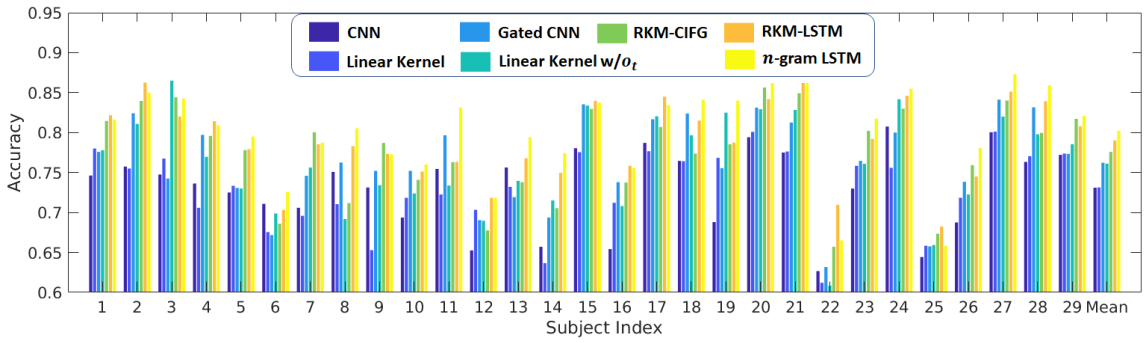


Figure 6.4: Subject-wise classification accuracy comparison for the various models on the LFP dataset.

Chapter 7

Conclusions

With these works, we explore the possibility of applying deep learning to automatic threat recognition for X-ray baggage screening at airport checkpoints, successfully creating functional prototype machines with full integration of convolutional object detection algorithms with X-ray scanning hardware [AB20]. These prototype systems are capable of simulating real-world deployment, scanning physical bags and projecting predictions directly on the constructed projections on display screens, all in real-time. We initially restricted our study to 4 threat classes: firearms, sharps, blunts, and LAGs, collecting a large dataset of scans on both Smiths and Rapiscan machines, and we find that deep learning models can indeed do an impressive job finding these types of objects, both on held-out evaluation sets and during live tests. Training a variety of models, we found that given a time budget of about 1 second per bag, Faster R-CNN [RHGS15] coupled with a ResNet [HZRS16] feature extractor performed best, and a simple OR-gate scheme to combine predictions across multiple views successfully boosted overall performance.

While these results represent a promising start, demonstrating the potential of deep learning algorithms for automatic threat recognition, some additional studies need to be conducted to see these models officially deployed at airports. For example, our data collection effort was limited to select categories of prohibited items, and specifically only subsets of those classes. Assembling a broader dataset providing more coverage of all potential threats may be necessary to some degree for automated threat detectors to be able to fully generalize to all the items they will encounter in the real world. A study of human factors [SM98] may also be necessary. Not only

is it important to determine the best ways to alert a human operator, but it is also critical to determine to what degree over-reliance on an automated-assist may lead to reduced human attention [LS04].

Several ideas from the deep learning literature are also worthy avenues of future extensions. We have explored positive-unlabeled (PU) learning [YLC20], continual learning [LLWC18a, LLWC18b, MLC20], and understanding neural networks [LWL+19]. While we did not necessarily show experimental results for these ideas, applying these concepts to automated threat recognition at airport checkpoints may be worthy directions of future research. For example, domain adaptation methods [GUA+16] have been shown to improve performance by leveraging Stream-of-Commerce data to show models more examples of benign backgrounds [SSLC20]. A study of distributed or federated learning methods [DCM+12, MMR+17] may also help scale the training of these systems across airports. Knowledge distillation or model compression [HVD15, CGL+20] methods can reduce the size and cost of the model, both speeding up inference time and requiring less computational resources for each machine. Finally, for sensitive applications, a study of adversarial attacks [SZS+14, PMG+17, ILCC20, ILW+20] would also be useful for understanding how these models may fail. While not impossible, such attacks on X-ray scan threat detection are fortunately still challenging, as inserted confounders must be robust to 3-dimensional rotation in X-ray transmission material properties to successfully attack an X-ray scanner. Many of these fields of machine learning represent opportunities to further improve engineered deep automatic threat recognition systems.

Deep learning [LBH15, GBC16] has made exciting progress over the past decade, and the potential for applying some of these methods to real-world applications like X-ray baggage screening shows promise to improve security at airports. Although an initial exploration, our efforts have demonstrated that it is indeed possible to build

these systems, producing working prototypes that successfully find certain classes of threat objects, even when hidden. With more data collection, system engineering, and incorporation of other ideas from the machine learning system, we hope such systems may eventually see real-world deployment.

Bibliography

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *Software available from tensorflow.org*, 2015.

[AB17] Samet Akcay and Toby P Breckon. An Evaluation of Region Based Object Detection Strategies within X-ray Baggage Security Imagery. *IEEE International Conference on Image Processing*, 2017.

[AB20] Samet Akcay and Toby Breckon. Towards automatic threat detection: A survey of advances of deep learning within x-ray security imaging. *arXiv:2001.01293*, 2020.

[Adm19] Federal Aviation Administration. Air Traffic By The Numbers. *https://www.faa.gov/air_traffic/by_the_numbers/*, 2019.

[AHB+18] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. *Computer Vision and Pattern Recognition*, 2018.

[AKDB16] Samet Akcay, Mikolaj E Kundegorski, Michael Devereux, and Toby P Breckon. Transfer Learning Using Convolutional Neural Networks for Object Classification within X-ray Baggage Security Imagery. *IEEE International Conference on Image Processing*, 2016.

[AKWB18] Samet Akcay, Mikolaj E Kundegorski, Chris G Willcocks, and Toby P Breckon. Using Deep Convolutional Neural Network Architectures for Object Classification and Detection within X-ray Baggage Security Imagery. *IEEE Transactions on Information Forensics and Security*, 2018.

[ALA+15] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C Lawrence Zitnick, Dhruv Batra, and Devi Parikh. VQA: Visual Question Answering. *International Conference on Computer Vision*, 2015.

[ARTP15] Fabio Anselmi, Lorenzo Rosasco, Cheston Tan, and Tomaso Poggio. Deep Convolutional Networks are Hierarchical Kernel Machines. *arXiv:1508.01084*, 2015.

[BBB13] Muhammet Bastan, Wonmin Byeon, and Thomas Breuel. Object Recognition in Multi-View Dual Energy X-ray Images. *British Machine Vision Conference*, 2013.

[BG16] David Balduzzi and Muhammad Ghifary. Strongly-Typed Recurrent Neural Networks. *International Conference on Machine Learning*, 2016.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer Normalization. *arXiv:1607.06450*, 2016.

[BM17] Alberto Bietti and Julien Mairal. Invariance and Stability of Deep Convolutional Representations. *Neural Information Processing Systems*, 2017.

[BMXS17] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. *International Conference on Learning Representations*, 2017.

[BTA04] Alain Berlinet and Christine Thomas-Agnan. Reproducing Kernel Hilbert spaces in Probability and Statistics. *Kluwer Publishers*, 2004.

[BV16] Hakan Bilen and Andrea Vedaldi. Weakly Supervised Deep Detection Networks. *Computer Vision and Pattern Recognition*, 2016.

[BYB11] Muhammet Bastan, Mohammad Reza Yousefi, and Thomas M Breuel. Visual Words on Baggage X-Ray Images. *Computer Analysis of Images and Patterns*, 2011.

[CFB+18] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. *arXiv:1804.09849*, 2018.

[CFL+15] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dolí, and C Lawrence Zitnick. Microsoft COCO Captions: Data Collection and Evaluation Server. *arXiv:1504.00325*, 2015.

[CGL+20] Liqun Chen, Zhe Gan, Kevin J Liang, Dong Wang, Ke Bai, Yitong Li, Chenyang Tao, Jingjing Liu, and Lawrence Carin. Wasserstein Contrastive Representation Distillation. *arXiv preprint*, 2020.

[CL16] Jianpeng Cheng and Mirella Lapata. Neural Summarization by Extracting Sentences and Words. *Association for Computational Linguistics*, 2016.

[CLJ+17] Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode Regularized Generative Adversarial Networks. *International Conference on Learning Representations*, 2017.

[COR+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. *Computer Vision and Pattern Recognition*, 2016.

[CvMG+14] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Empirical Methods in Natural Language Processing*, 2014.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *North American Chapter of the Association for Computational Linguistics*, 2019.

[DCM+12] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large Scale Distributed Deep Networks. *Neural Information Processing Systems*, 2012.

[DDGL99] Francesco De Comité, François Denis, Rémi Gilleron, and Fabien Letouzey. Positive and Unlabeled Examples Help Learning. *Algorithmic Learning Theory*, 1999.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, 2009.

[Den98] François Denis. PAC Learning from Positive Statistical Queries. *Algorithmic Learning Theory*, 1998.

[DFAG17] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language Modeling with Gated Convolutional Networks. *International Conference on Machine Learning*, 2017.

[DGM17] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative Multi-Adversarial Networks. *International Conference on Learning Representations*, 2017.

[DLHS16] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In *Neural Information Processing Systems*, 2016.

[DNS14] Marthinus C Du Plessis, Gang Niu, and Masashi Sugiyama. Analysis of learning from positive and unlabeled data. In *Neural Information Processing Systems*, 2014.

[DNS15] Marthinus Du Plessis, Gang Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. *International Conference on Machine Learning*, 2015.

[DZM+18] Xuanyi Dong, Liang Zheng, Fan Ma, Yi Yang, and Deyu Meng. Few-shot Object Detection. *Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[EN08] Charles Elkan and Keith Noto. Learning Classifiers from Only Positive and Unlabeled Data. *International Conference on Knowledge Discovery and Data Mining*, 2008.

[EVW+10] Mark Everingham, Luc Van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 2010.

[FG18] Sebastian Farquhar and Yarin Gal. Towards Robust Evaluations of Continual Learning. *arXiv:1805.09733*, 2018.

[FLTZ10] Zhouyu Fu, Guojun Lu, Kai Ming Ting, and Dengsheng Zhang. A Survey of Audio-based Music Classification and Annotation. *IEEE Transactions on Multimedia*, 2010.

[GAA+17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. *Neural Information Processing Systems*, 2017.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, Cambridge, MA, USA, 1st ed edition, 2016.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[Gen01] Marc G Genton. Classes of Kernels for Machine Learning: A Statistics Perspective. *Journal of Machine Learning Research*, 2001.

[GH16] David Golub and Xiaodong He. Character-Level Question Answering with Attention. *Empirical Methods in Natural Language Processing*, 2016.

[Gir15] Ross Girshick. Fast R-CNN. In *International Conference on Computer Vision*, 2015.

[GKN+18] Arnab Ghosh, Viveka Kulharia, Vinay Namboodiri, Iit Kanpur, Philip H S Torr, and Puneet K Dokania. Multi-Agent Diverse Generative Adversarial Networks. *Computer Vision and Pattern Recognition*, 2018.

[GLC+18] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long Text Generation via Adversarial Training with Leaked Information. *AAAI Conference on Artificial Intelligence*, 2018.

[GMX+13] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *arXiv:1312.6211*, 2013.

[GPAM+14] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. *Neural Information Processing Systems*, 2014.

[GSK+17] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. LSTM: A Search Space Odyssey. *Transactions on Neural Networks and Learning Systems*, 2017.

[GSR+18] Chunhui Gu, Chen Sun, David A Ross, Carl Vondrick, Caroline Pantofaru, Yeqing Li, Sudheendra Vijayanarasimhan, George Toderici, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. AVA: A Video Dataset of Spatio-Temporally Localized Atomic Visual Actions. *Computer Vision and Pattern Recognition*, 2018.

[GUA+16] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 2016.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B Girshick. Mask R-CNN. *International Conference on Computer Vision*, 2017.

[HGT+14] Judy Hoffman, Sergio Guadarrama, Eric Tzeng, Jeff Donahue, Ross B Girshick, Trevor Darrell, and Kate Saenko. LSDA: Large Scale Detection Through Adaptation. *Neural Information Processing Systems*, 2014.

[HNLP18] Quan Hoang, Tu Dinh Nguyen, Trung Le, and Dinh Phung. MGAN: Training Generative Adversarial Nets with Multiple Generators. *International Conference on Learning Representations*, 2018.

[HRS+17] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Et al. Speed/Accuracy Trade-offs for Modern Convolutional Object Detectors. In *Computer Vision and Pattern Recognition*, 2017.

[HRU+17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *Neural Information Processing Systems*, 2017.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 1997.

[HS12] Michiel Hermans and Benjamin Schrauwen. Recurrent Kernel Machines: Computing with Infinite Echo State Networks. *Neural Computation*, 2012.

[HSS12] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural Networks for Machine Learning – Lecture 6a – Overview of Mini-Batch Gradient Descent, 2012.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv:1503.02531*, 2015.

[HWT+15] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, Fernando Mujica, Adam Coates, and Andrew Y. Ng. An Empirical Evaluation of Deep Learning on Highway Driving. *arXiv:1504.01716*, 2015.

[HZC+17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, 2017.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Computer Vision and Pattern Recognition*, 2016.

[ILCC20] Nathan Inkawhich, Kevin J Liang, Lawrence Carin, and Yiran Chen. Transferable Perturbations of Deep Feature Distributions. *International Conference on Learning Representations*, 2020.

[ILW+20] Nathan Inkawhich, Kevin J Liang, Binghui Wang, Matthew Inkawhich, Lawrence Carin, and Yiran Chen. Perturbing Across the Feature Hierarchy to Improve Standard and Strict Blackbox Attack Transferability. *arXiv:2004.14861*, 2020.

[IS15] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *International Conference on Machine Learning*, 2015.

[IZJ+17] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The THUMOS challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 2017.

[JGP16] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv:1611.01144*, 2016.

[JZL+18] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. MentorNet: Learning Data-Driven Curriculum for Very Deep Neural Networks on Corrupted Labels. *International Conference on Machine Learning*, 2018.

[JZS15] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An Empirical Exploration of Recurrent Network Architectures. *International Conference on Machine Learning*, 2015.

[KAD+16] Mikolaj E Kundegorski, Samet Akçay, Michael Devereux, Andre Mouton, and Toby P Breckon. On Using Feature Descriptors as Visual Words for Object Detection within X-ray Baggage Security Screening. In *International Conference on Imaging for Crime Detection and Prevention*, 2016.

[KALL18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *International Conference on Learning Representations*, 2018.

[Kim14] Yoon Kim. Convolutional Neural Networks for Sentence Classification. *Empirical Methods in Natural Language Processing*, 2014.

[KNDS17] Ryuichi Kiryo, Gang Niu, Marthinus C Du Plessis, and Masashi Sugiyama. Positive-Unlabeled Learning with Non-Negative Risk Estimator. *Neural Information Processing Systems*, 2017.

[KPR+17] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan,

Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, 2017.

[KRA+18] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv:1811.00982*, 2018.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Neural Information Processing Systems*, 2012.

[KW05] Jens Krüger and Rüdiger Westermann. Linear Algebra Operators for GPU Implementation of Numerical Algorithms. In *ACM SIGGRAPH 2005 Courses*, 2005.

[KZG+17] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael S Bernstein, and Li Fei-Fei. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *International Journal Computer Vision*, 2017.

[LAE+16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single Shot MultiBox Detector. *European Conference on Computer Vision*, 2016.

[LB95] Yann LeCun and Yoshua Bengio. Convolutional Networks for Images, Speech, and Time Series. *The Handbook of Brain Theory and Neural Networks*, 1995.

[LBD+89] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1989.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[LDG00] Fabien Letouzey, François Denis, and Rémi Gilleron. Learning from Positive and Unlabeled Examples. *Algorithmic Learning Theory*, 2000.

[LGH+17] Rebecca Sawyer Lee, Francisco Gimenez, Assaf Hoogi, Kanae Kawai Miyake, Mia Gorovoy, and Daniel L Rubin. A curated mammography

data set for use in computer-aided detection and diagnosis research. *Nature Scientific Data*, 2017.

[LHG+18] Kevin J Liang, Geert Heilmann, Christopher Gregory, Souleymane O Diallo, David Carlson, Gregory P Spell, John B Sigman, Kris Roe, and Lawrence Carin. Automatic Threat Recognition of Prohibited Items at Aviation Checkpoint with X-ray Imaging: A Deep Learning Approach. *SPIE Anomaly Detection and Imaging with X-Rays (ADIX) III*, 2018.

[LHP+15] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous Control with Deep Reinforcement Learning. *arXiv:1509.02971*, 2015.

[LJBJ17] Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Deriving Neural Architectures from Sequence and Graph Kernels. *International Conference on Machine Learning*, 2017.

[LLH+17] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Adversarial ranking for language generation. *Neural Information Processing Systems*, 2017.

[LLWC18a] Kevin J Liang, Chunyuan Li, Guoyin Wang, and Lawrence Carin. Generative Adversarial Network Training is a Continual Learning Problem. *arXiv:1811.11083*, 2018.

[LLWC18b] Kevin J Liang, Chunyuan Li, Guoyin Wang, and Lawrence Carin. Generative Adversarial Networks and Continual Learning. *Neural Information Processing Systems: Continual Learning Workshop*, 2018.

[LLZ17] Kenton Lee, Omer Levy, and Luke Zettlemoyer. Recurrent Additive Networks. *arXiv:1705.07393*, 2017.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. *European Conference on Computer Vision*, 2014.

[LMM+17] Yitong Li, Michael Murias, Samantha Major, Geraldine Dawson, Kafui Dzirasa, Lawrence Carin, and David E Carlson. Targeting EEG/LFP Synchrony with Neural Nets. *Neural Information Processing Systems*, 2017.

[LMS+17] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial Learning for Neural Dialogue Generation. *Empirical Methods in Natural Language Processing*, 2017.

[Low99] David G Lowe. Object Recognition from Local Scale-Invariant Features. *International Conference on Computer Vision*, 1999.

[LS04] John D Lee and Katrina A See. Trust in Automation: Designing for Appropriate Reliance. *Human Factors*, 2004.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *Computer Vision and Pattern Recognition*, 2015.

[LSLW16] Anders Larsen, Søren Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *International Conference on Machine Learning*, 2016.

[LSS+19] Kevin J Liang, John B Sigman, Gregory P Spell, Dan Strellis, William Chang, Felix Liu, Tejas Mehta, and Lawrence Carin. Toward Automatic Threat Recognition for Airport X-ray Baggage Screening with Deep Convolutional Object Detection. *arXiv:1912.06329*, 2019.

[LWL+19] Kevin J Liang, Guoyin Wang, Yitong Li, Ricardo Henao, and Lawrence Carin. Kernel-Based Approaches for Sequence Modeling: Connections to Neural Methods. *Neural Information Processing Systems*, 2019.

[MAD+12] Inês C Moreira, Igor Amaral, Inês Domingues, António Cardoso, Maria João Cardoso, and Jaime S Cardoso. INbreast: Toward a Full-field Digital Mammographic Database. *Academic Radiology*, 2012.

[Mai16] Julien Mairal. End-to-End Kernel Learning with Supervised Convolutional Kernel Networks. *Neural Information Processing Systems*, 2016.

[MC89] Michael McCloskey and Neal J Cohen. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *The Psychology of Learning and Motivation*, 1989.

[MKKY18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. *International Conference on Learning Representations*, 2018.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

[MKS18] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and Optimizing LSTM Language Models. *International Conference on Learning Representations*, 2018.

[MLC20] Nikhil Mehta, Kevin J Liang, and Lawrence Carin. Bayesian Nonparametric Weight Factorization for Continual Learning. *arXiv:2004.10098*, 2020.

[MMR+17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. *Artificial Intelligence and Statistics*, 2017.

[MNG17] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The Numerics of GANs. *Neural Information Processing Systems*, 2017.

[MO14] Mehdi Mirza and Simon Osindero. Conditional Generative Adversarial Nets. *arXiv:1411.1784*, 2014.

[MPPSD17] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled Generative Adversarial Networks. *International Conference on Learning Representations*, 2017.

[MSA16] Domingo Mery, Erick Svec, and Marco Arias. Object Recognition in Baggage Inspection Using Adaptive Sparse Representations of X-ray Images. *Image and Video Technology*, 2016.

[MSM93] Mitchell P Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Association for Computational Linguistics*, 1993.

[MXBS17] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. *International Conference on Learning Representations*, 2017.

[NBC17] Behnam Neyshabur, Srinadh Bhojanapalli, and Ayan Chakrabarti. Stabilizing GAN Training with Multiple Random Projections. *arXiv:1705.07831*, 2017.

[NK17] Vaishnavh Nagarajan and J Zico Kolter. Gradient descent GAN optimization is locally stable. *Neural Information Processing Systems*, 2017.

[NV06] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *International Conference on Pattern Recognition*, 2006.

[OBLS15] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is Object Localization for Free? Weakly-supervised Learning with Convolutional Neural Networks. *Computer Vision and Pattern Recognition*, 2015.

[ODZ+16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A Generative Model for Raw Audio. *arXiv:1609.03499*, 2016.

[PGH+16] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational Autoencoder for Deep Learning of Images, Labels and Captions. *Neural Information Processing Systems*, 2016.

[PKP+19] German Ignacio Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual Lifelong Learning with Neural Networks: A Review. *Neural Networks*, 2019.

[PMG+17] Nicolas Papernot, Patrick D McDaniel, Ian J Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Machine Learning. *AsiaCCS*, 2017.

[PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. *Association for Computational Linguistics*, 2002.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. *Empirical Methods in Natural Language Processing*, 2014.

[Qia99] Ning Qian. On the Momentum Term in Gradient Descent Learning Algorithms. *Neural Computation*, 1999.

[RASC14] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN Features off-the-shelf: an Astounding Baseline for Recognition. In *Computer Vision and Pattern Recognition*, 2014.

[Rat90] Roger Ratcliff. Connectionist Models of Recognition Memory: Constraints Imposed by Learning and Forgetting Functions. *Psychology Review*, 1990.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *Computer Vision and Pattern Recognition*, 2016.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015.

[RF17] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. In *Computer Vision and Pattern Recognition*, jul 2017.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer Assisted Intervention*, 2015.

[RGBYO18] Alexandre Rame, Emilien Garreau, Hedi Ben-Younes, and Charles Ollion. OMNIA Faster R-CNN: Detection in the wild through dataset merging and soft distillation. *arXiv:1812.02611*, 2018.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Neural Information Processing Systems*, 2015.

[RJMG17] Thomas W Rogers, Nicolas Jaccard, Edward J Morton, and Lewis D Griffin. Automated X-ray Image Analysis for Cargo Security: Critical Review and Future Promise. *Journal of X-ray Science and Technology*, 2017.

[RKF19] Christopher Roth, Ingmar Kanitscheider, and Ila Fiete. Kernel RNN Learning (KeRNL). *International Conference on Learning Representations*, 2019.

[RMC16] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *International Conference on Learning Representations*, 2016.

[RW15] Mrigank Rochan and Yang Wang. Weakly supervised localization of novel objects using appearance transfer. *Computer Vision and Pattern Recognition*, 2015.

[SBSL18] Ari Seff, Alex Beatson, Daniel Suo, and Han Liu. Continual Learning in Generative Adversarial Nets. *arXiv:1705.08395*, 2018.

[SGZ+16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. *Neural Information Processing Systems*, 2016.

[SHM+16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

[SIVA16] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *International Conference on Learning Representations Workshop*, 2016.

[SM98] Mark S Sanders and Ernest James McCormick. Human Factors in Engineering and Design. *Industrial Robot: An International Journal*, 1998.

[SPT+17] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from Simulated and Unsupervised Images through Adversarial Training. *Computer Vision and Pattern Recognition*, 2017.

[SS02] Bernhard Scholkopf and Alexander J Smola. Learning with Kernels. *MIT Press*, 2002.

[SSLC20] John B Sigman, Gregory P Spell, Kevin J Liang, and Lawrence Carin. Background Adaptive Faster R-CNN for Semi-supervised Convolutional Object Detection of Threats in X-ray Images. *SPIE Anomaly Detection and Imaging with X-Rays (ADIX) V*, 2020.

[SSSG17] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *International Conference on Computer Vision*, 2017.

[SVR+17] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning. *Neural Information Processing Systems*, 2017.

[SW17] Yunus Saatchi and Andrew Gordan Wilson. Bayesian GAN. *Neural Information Processing Systems*, 2017.

[SZ15] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations*, 2015.

[SZS12] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. T. *Technical Report CRCV-TR-12-01, University of Central Florida*, 2012.

[SZS+14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[TMB13] Diana Turcsany, Andre Mouton, and Toby P Breckon. Improving Feature-Based Object Recognition for X-Ray Baggage Security Screening Using Primed Visual Words. *International Conference on Industrial Technology*, 2013.

[TO18] Corentin Tallec and Yann Ollivier. Can Recurrent Neural Networks Warp Time? *International Conference on Learning Representations*, 2018.

[TSA17] Passenger Screening Algorithm Challenge. *https://www.kaggle.com/c/passenger-screening-algorithm-challenge/overview/description*, 2017.

[TSA20] TSA Year in Review: 2019. *https://www.tsa.gov/blog/2020/01/15/tsa-year-review-2019*, 2020.

[TSdC+19] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An Empirical Study of Example Forgetting during Deep Neural Network Learning. *International Conference on Learning Representations*, 2019.

[TTTV18] Hoang Thanh-Tung, Truyen Tran, and Svetha Venkatesh. On Catastrophic Forgetting and Mode Collapse in Generative Adversarial Networks. *arXiv:1807.04015*, 2018.

[TWG+16] Yuxing Tang, Josiah Wang, Boyang Gao, Emmanuel Dellandrea, Robert Gaizauskas, and Liming Chen. Large Scale Semi-Supervised Object Detection Using Visual and Semantic Knowledge Transfer. *Computer Vision and Pattern Recognition*, 2016.

[UNS+18] Thomas Unterthiner, Bernhard Nessler, Calvin Seward, Günter Klambauer, Martin Heusel, Hubert Ramsauer, and Sepp Hochreiter. Coulomb GANs: Provably Optimal Nash Equilibria Via Potential Fields. *International Conference on Learning Representations*, 2018.

[VAC+17] Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. Learning From Noisy Large-Scale Datasets With Minimal Supervision. *Computer Vision and Pattern Recognition*, 2017.

[vdODZ+16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and

Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *arXiv:1609.03499*, 2016.

[vEMY13] Jordy van Enkhuizen, Arpi Minassian, and Jared W Young. Further evidence for ClockΔ19 mice as a model for bipolar disorder mania using cross-species tests of exploration and sensorimotor gating. *Behavioural Brain Research*, 2013.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Neural Information Processing Systems*, 2017.

[Wag20] Jay Wagner. TSA Year in Review: A Record Setting 2018. *https://www.tsa.gov/blog/2019/02/07/tsa-year-review-record-setting-2018*, 2020.

[WBS+19] Zhe Wu, Navaneeth Bodla, Bharat Singh, Mahyar Najibi, Rama Chellappa, and Larry S Davis. Soft Sampling for Robust Object Detection. *British Machine Vision Conference*, 2019.

[WHSX16] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep Kernel Learning. *International Conference on Artificial Intelligence and Statistics*, 2016.

[WLW+18] Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. Joint Embedding of Words and Labels for Text Classification. *Association for Computational Linguistics*, 2018.

[WPL+17] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. ChestX-ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. *Computer Vision and Pattern Recognition*, 2017.

[WSC+16] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's Neural Machine Translation System: Bridging the Gap Between Human and Machine Translation. *arXiv:1609.08144*, 2016.

[WYLZ16] Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model. *Association for Computational Linguistics*, 2016.

[XXXT17] Yixing Xu, Chang Xu, Chao Xu, and Dacheng Tao. Multi-Positive and Unlabeled Learning. In *International Joint Conferences on Artificial Intelligence*, 2017.

[YCBL14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Neural Information Processing Systems*, 2014.

[YLC20] Yuewei Yang, Kevin J Liang, and Lawrence Carin. Object Detection as a Positive-Unlabeled Problem. *arXiv:2002.04672*, 2020.

[YWLS17] Ke Yan, Xiaosong Wang, Le Lu, and Ronald M Summers. DeepLesion: Automated Deep Mining, Categorization and Detection of Significant Radiology Image Findings using Large-Scale Clinical Lesion Annotations. *arXiv:1710.01766*, 2017.

[YZWY17] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *AAAI Conference on Artificial Intelligence*, 2017.

[ZBB+17] Chiyuan Zhang, Samy Bengio, Google Brain, Moritz Hardt, Benjamin Recht, Oriol Vinyals, and Google Deepmind. Understanding Deep Learning Requires Re-thinking Generalization. *International Conference on Learning Representations*, 2017.

[ZF14] Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *European Conference on Computer Vision*, sep 2014.

[ZGF+17] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. *International Conference on Machine Learning*, 2017.

[ZPG17] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. *International Conference on Machine Learning*, 2017.

[ZSLL15] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C M Lau. A C-LSTM Neural Network for Text Classification. *arXiv:1511.08630*, 2015.

[ZVSL18] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning Transferable Architectures for Scalable Image Recognition. In *Computer Vision and Pattern Recognition*, 2018.

[ZZL15] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification. *Neural Information Processing Systems*, 2015.

Biography

Kevin J Liang earned his Bachelor of Science in Engineering (BSE) degrees in Electrical and Computer Engineering (ECE) and Biomedical Engineering (BME) from Duke University in 2015, graduating *summa cum laude* and with ECE departmental distinction. As one of the top 2 students in both ECE and BME, he simultaneously also received the George Sherrerd III Memorial Award and the Da Vinci Award, respectively. He obtained a Master of Science (MS) in ECE from Duke University in 2019 and a Doctor of Philosophy (Ph.D.) in the spring of 2020. Part of his graduate studies was graciously funded by the E Bayard Halsted Scholarship.

Kevin has been advised throughout his Ph.D. by James L. Meriam Distinguished Professor of Engineering Lawrence Carin. His primary research interests have been in deep learning, including computer vision, natural language processing, continual learning, and representation learning. Throughout his Ph.D., he has also been interested in teaching, acting as an instructor for four Duke Machine Learning Schools, Duke University's +DataScience program, and the Duke Introduction to Machine Learning Coursera.