GAVIN S. HARTNETT, LANCE MENTHE, JASMIN LÉVEILLÉ, DAMIEN BAVEYE, LI ANG ZHANG, DARA GOLD, JEFF HAGEN, JIA XU

Operationally Relevant Artificial Training for Machine Learning

Improving the Performance of Automated Target Recognition Systems

For more information on this publication, visit www.rand.org/t/RRA683-1

Library of Congress Cataloging-in-Publication Data is available for this publication.
ISBN: 978-1-9774-0595-1

Published by the RAND Corporation, Santa Monica, Calif.

Preface

Automated target recognition (ATR) is one of the most important potential military applications of the many recent advances in artificial intelligence and machine learning. A key obstacle to creating a successful ATR system with machine learning is the collection of high-quality labeled data sets. In this report, we investigate whether this obstacle can be sidestepped by training object-detection algorithms on data sets of high-resolution, high-quality artificial images. Although we found that artificial images cannot replace real images, artificial images can supplement an existing data set of real images to boost performance.

RAND Project AIR FORCE

RAND Project AIR FORCE (PAF), a division of the RAND Corporation, is the Department of the Air Force's (DAF's) federally funded research and development center for studies and analyses. PAF provides the DAF with independent analyses of policy alternatives affecting the development, employment, combat readiness, and support of current and future air, space, and cyber forces. Research is conducted in four programs: Strategy and Doctrine; Force Modernization and Employment; Manpower, Personnel, and Training; and Resource Management.

Additional information about PAF is available on our website:
www.rand.org/paf/

Funding

Funding for this research was made possible by the independent research and development provisions of the RAND Corporation's contracts for the operation of its U.S. Department of Defense federally funded research and development centers.

Contents

Figures

Tables

Summary

Issue

Automated target recognition (ATR) is one of the most important potential military applications of the many recent advances in artificial intelligence and machine learning. A key obstacle to creating a successful ATR system based on machine learning is the collection of high-quality labeled data sets. We investigated whether this was an obstacle that could be sidestepped by training object-detection algorithms on data sets of high-resolution artificial images.

Approach

We chose the high-mobility multipurpose wheeled vehicle (HMMWV) as our primary test case. We created a software pipeline that enabled us to use the commercial video game platform *Arma 3* to generate and label high-quality artificial images of the HMMWV from different angles and altitudes, in different environments, and under different lighting conditions. These artificial images were used to train a series of object-detection models.

To evaluate the resulting performance of these models, we created a second data set of real images by renting a HMMWV and contracting a photographer who used a drone-mounted high-resolution camera. The conditions of this photo shoot were chosen to imitate the scenes generated by *Arma 3* as closely as possible so that a reasonably fair comparison could be made. These images were labeled by hand. In addition to considering data sets consisting of either purely artificial or purely real images, we also evaluated the performance of models trained on hybrid data sets consisting of both artificial and real images.

For the machine learning program, we used an open-source implementation of the Mask-RCNN algorithm provided by Facebook Artificial Intelligence Research. We used the same standard settings for all relevant hyperparameters in all cases.

Conclusions

Despite the apparent similarity of the artificial and real-image sets, we found that models trained on artificial images performed very poorly on real images. However, we found that hybrid training sets—those consisting of a combination of artificial and real images—produced a better performance than algorithms trained on real images alone. The improvements were most noticeable when the number of real images was severely limited. For example, by boosting a data set of five real images with ten artificial ones, we were able to improve the precision and

recall of the model by 54 percent and 29 percent, respectively.[1] It should be noted that, even with this boost, the baseline performances of these models trained on limited data sets are not competitive with state-of-the-art models and would likely have limited operational utility. However, our results do suggest a technique for improving the training of models when existing data are quite scarce and there is a source of high-quality artificial images. We emphasize also that it is possible that both the positive and negative results might not generalize to image sets with different parameters. More research would be needed to determine under what conditions, if any, models trained successfully on artificial images might perform well on real images. More work is also needed to better understand the ability of models trained on hybrid data sets to perform well on purely real images.

To illustrate our goal, Figure S.1 shows a real image that was analyzed by two artificial intelligence and machine learning algorithms. The first was based solely on real training data and failed to find the HMMWV. The second was trained on hybrid data and identified the HMMWV. Of course, this is just a single example. In Chapter 4, we more thoroughly compare the performances of models trained on purely real images, purely artificial images, and hybrid images.

[1] *Precision* is defined as the fraction of the positive identifications made by the model that are correct, and *recall* is defined as the fraction of positive instances that the model is able to correctly identify.

Figure S.1. Real-to-Hybrid Comparison

Acknowledgments

We would like to thank our RAND colleagues Ted Harshberger, Susan Marquis, and Howard Shatz for their support of and enthusiasm for this project. We are indebted to Dahlia Goldfeld and Christian Johnson for their careful reading of an earlier draft of this report. Their comments and critiques greatly improved the report. We would also like to thank Adrian Salas and Dave Nguyen for their help setting up and maintaining the Amazon Web Services resources used for this project, and Gary Briggs for his generous assistance navigating RAND's analytic computing resources. Lastly, we thank Marcus Hunt, the drone camera operator who traveled with us to the High Desert area in Southern California and expertly captured the videos and images that were essential to this project.

Abbreviations

3D	three dimensional
AI	artificial intelligence
API	application program interface
ATR	automated target recognition
AWS	Amazon Web Services
BISim	Bohemia Interactive Simulations
FAIR	Facebook Artificial Intelligence Research
HMMWV	high-mobility multipurpose wheeled vehicle
IoU	intersection over the union
mAP	mean average precision
mAR	mean average recall
ML	machine learning
PEO STRI	U.S. Army Program Executive Officer, Simulation, Training and Instrumentation
RCNN	region-based convolutional neural network
SAM	surface-to-air missile
SSD	single shot detection
VBS	Virtual Battlespace series

1. Introduction

Overview

We set out to demonstrate that an automated target recognition (ATR) system could be built using nothing but commercial off-the-shelf artificial intelligence/machine learning (AI/ML) algorithms and artificially rendered imagery to train them—but we did not succeed. What the AI/ML algorithms learned about detecting artificial objects in artificial images failed to transfer meaningfully toward the task of detecting real objects in real images. However, to our surprise, we discovered that a hybrid training set consisting of both real and artificial images together produced a more robust ATR system than a training set of real images alone. In other words, we were able to boost the performance of the ATR system consistently by adding artificially generated images of the target to the original training set.

Although we did not create the revolutionary new ATR training method we had hoped for, we did find a novel way to make ATR systems better when training data are scarce—which is often the case in military contexts. We also successfully prototyped a method of generating artificial images, acquired a unique data set of real images, and created a smooth workflow using open-source code provided by Facebook together with the Amazon Web Services (AWS) cloud computing platform. Our main result, specifically that artificial images can be used to boost the performance of an ATR system, suggests a new line of research into this effect. More work is needed to understand when this approach will be most useful. By tuning hyperparameters and varying the characteristics of the artificial images, it may be possible to enhance this effect further.

Background

ATR is one of the most important potential military applications of AI and machine learning today. A high-performance ATR system could improve the timeliness, accuracy, and completeness of threat warnings and sensor data exploitation across all services. With the recent advances of deep learning–based approaches to AI, object detection is becoming a reality, as evidenced in everyday life from Google's image search, Facebook's tagging of people in photographs, and Apple's facial recognition technology on iPhones. A key obstacle to creating a successful ATR system for military purposes is the curation of carefully labeled data sets to train them.

Project Maven, the U.S. Department of Defense's effort to create an ATR system for analysis of drone video footage, is a good example. Project Maven uses commercially available AI/ML algorithms but requires more than 100,000 hand-labeled images to train the system to recognize

each object.[1] Generating such a training set is a massive undertaking. Furthermore, the resulting ATR system may still be unable to recognize objects in new environments and contexts. One of our main motivations is that, in times of conflict, there may be a need to quickly create labeled data sets of adversaries' military assets in previously unencountered environments or contexts. It is possible that, in such a scenario, a realistic labeled data set could be quickly generated through such an approach as the one we explore, provided that enough information about the targets is known so that accurate three-dimensional (3D) models are available.

Many organizations are now interested in using synthetic or simulated data to improve machine learning models for a wide variety of tasks. For example, NVIDIA has developed a sophisticated simulation engine to develop and test its self-driving car technology (Greenstein, 2019). Also, OpenAI et al. (2019) has recently used training on simulated data to successfully teach a robotic hand to perform complex manipulations. Additionally, the idea of using synthetic data generated from computer games to improve the performance of computer vision models has also received a fair amount of attention in recent years. For example, Taylor, Chosak, and Brewer (2007) used the video game *Half-Life 2* to train a video surveillance system, and Shafaei, Little, and Schmidt (2016) used a video game to train image segmentation and depth-estimation algorithms using deep neural networks. The U.S. government has also shown interest in this approach. Intelligence Advanced Research Projects Activity has funded the development of an open-source computer vision plug-in for the popular video game engine *Unreal Engine 4* (Qiu and Yuille, 2016). Lastly, the relevance of this approach for operationally relevant scenarios is underscored by the fact that, as this project was in its final stages, the U.S. Air Force released a Small Business Innovation Research (or SBIR) solicitation for "Transfer Learning and Deep Transfer Learning for Military Applications" to "build Aided Target Recognition (AiTR) and other algorithms for environments and targets where we currently lack data or lack labeled data" (SBIR STTR, undated).

Objectives

In this project, we investigated a novel approach to training an ATR system. Instead of using real, hand-labeled images, we constructed an *artificial training set* using macros and commercial high-quality environment simulators. The software enabled us to combine 3D renderings of the desired object with simulated environments to generate artificial images of an object from different vantage points in different environments under different lighting conditions. We then trained an object-detection model on this simulated data and tested it on real data. Our efforts were organized into two main tasks: data generation (described in Chapter 2) and image classification (described in Chapters 3 and 4).

[1] The algorithm itself is just "75 lines of Python code" on top of Google's TensorFlow (Pellerin, 2017; Rolen, 2017).

We used the high-mobility multipurpose wheeled vehicle (HMMWV) as our test object because it is both an object of military interest and an object that can be obtained for study in the civilian world without too much difficulty. This proved fortunate because, in the course of this project, we realized that we needed a larger set of real images for comparison than was publicly available—so we rented a HMMWV and hired a drone operator to film it from the air in different locations from different angles at different times of the day.

Structure of Report

In Chapter 2, we describe the process that we created to generate simulated images of military-relevant objects and how we obtained the comparison set of real images. In Chapter 3, we describe the AI/ML algorithms we employed and the different evaluation criteria used to measure their performance. In Chapter 4, we discuss the results—what we found and what we did not find. Finally, in Chapter 5, we describe what we learned from this endeavor and how this work could be extended in the future. Additional technical details are contained in two appendixes.

2. Operationally Relevant Data

In this project, we tested whether an object-detection model that was successfully trained on high-quality artificial images could perform well when evaluated on real images. This problem is an example of what is called *domain adaptation* in the machine learning literature, in which the training and testing data are drawn from similar but different distributions (Daumé and Marcu, 2006). We are particularly interested in investigating the domain adaptation of object-detection models in the context of *operationally relevant scenarios*, which we loosely define to be scenarios in which the objects of interest are military assets, such as tanks, HMMWVs, or surface-to-air missiles (SAMs). Moreover, operationally relevant scenarios also include environments in which these military assets are located that are closely related to those that might plausibly occur in future military conflicts. In this chapter, we describe two sources or domains of operationally relevant data: first, the source of artificial images we developed, followed by a description of the data set of real images we obtained for evaluation purposes. We will say that a model *transfers* from one domain to another if it performs well on images from both domains, although only one domain was used to train the model.

Artificial Image Generation

We used the video game platform *Arma 3* to generate artificial images for training object-detection algorithms. Commercially released in September 2013, *Arma 3* is a tactical, first-person military simulation video game using Bohemia Interactive Studio's Real Virtuality 4 engine. *Arma 3* is commercially available from either the Bohemia Interactive Store directly or via third-party vendors for $29.99 (Bohemia Interactive Store, undated), with additional downloadable content packs available ranging from $2.99 to $27.99.[1] The game continues to be actively supported by Bohemia Interactive. In addition to official content releases, the *Arma 3* gaming community is highly active, with more than 500,000 members in the Steam (a popular third-party hosting service) community group and more than 67,000 individual modifications (including models, maps, and scenarios) available free to game users (Steam, undated-c). Minimum requirements for *Arma 3* are consistent with commercially available midrange laptops, although best performance can be achieved with enhanced processors, graphics hardware, and memory. The total disk space required to run the program is approximately 37 GB, depending on downloaded content (Steam, undated-a).

[1] Two copies were purchased for our project. They were run on personally owned computers because of the limitations of available RAND laptops in terms of graphics and memory.

Attractive aspects of *Arma 3* are the in-game map editor and a built-in "splendid camera" mode, which allows the user to place 3D object models (including user-generated models) into a variety of fully customizable environments. The mode also allows for these scenes to be captured as still images or videos by means of a comprehensive user interface. Advanced camera controls, model movement, and interactions between models or between models and the environment can be scripted using Status Quo Format (a unique pseudocode based in the C syntax) through an application program interface (API) based in C#, which is fully documented online (Bohemia Interactive Community Wiki, undated). This scripting capability allows for a maximum level of user control.

We used this scripting capability to generate artificial images of operationally relevant scenes. We developed software that systematically situates a target model, such as an HMMWV, in a variety of preset locations within a map. We also varied the relative position and angle of the camera. To ensure that images of the object were generated from a wide variety of angles, the camera positions were chosen to form a half-dome centered on the object, and the camera angles were chosen so that the generated images were centered around the object. The background map used for the image generation was a terrain map corresponding to the Diyala province in eastern Iraq, which was provided by the user community in the package CUP Terrains Core (Steam, undated-b). Installing packages for the targets and the world map required installing add-ons to satisfy certain dependency requirements.[2] Examples of the generated *Arma 3* images are shown in Figure 2.1. Each image contains a single HMMWV located in the center surrounded by a black bounding box.

To train an object-detection algorithm, the location and labels of the objects within each image are also needed. In our case, we stored the location of each object using the bounding box format. As discussed in more detail next, bounding boxes are rectangles that bound the object of interest within the image. We used *Arma 3*'s scripting capabilities to automatically generate the bounding boxes for each image we produced. The scripting API contains a function that provides the bounding cube of a given object in the scene, which can then be used to produce a bounding box by projecting down to two-dimensional range. This requires a proper understanding of the geometry of the scene, especially the relative location of the camera and the object. Unfortunately, the automatic image generation took a nontrivial amount of time. For example, it took four hours to generate and save 2,500 high-resolution images using a gaming PC. Another difficulty we encountered was that the scripting API had a function to determine whether a target was occluded or not from the viewpoint of the camera, but this function appeared to be unreliable.

[2] These dependencies are, respectively, POOK Camonets (see hcpookie, undated) and PLP Containers (see poolpunk, undated).

Figure 2.1. Examples of *Arma 3* Images

NOTE: These are examples of the *Arma 3*–generated images. Each image depicts a single HMMWV in an environment similar to the one in Diyala Province in Iraq. Also shown in black are bounding boxes containing each HMMWV.

Real-World Evaluation Images

To evaluate the effectiveness of the artificially trained models, we required a reference data set of real-world images. Ideally, the images in this set would be as similar as possible to the *Arma 3* artificial images to isolate the effect of training on artificial images.

To this end, on June 5, 2019, two members of our team traveled to Yermo, California, and supervised a photo shoot of a privately owned military surplus HMMWV. Yermo is located approximately 125 miles northeast of Los Angeles and is home to the Yermo Annex of Marine Corps Logistics Base Barstow. This facility functions as a vehicle-storage depot for the U.S. Marine Corps and maintains a large volume of HMMWV variants currently in dry storage (Defense Logistics Agency, undated). The HMMWV used in this event was an M1038A1 variant that was decommissioned from the Yermo facility and sold at auction to a private buyer. The photo shoot was carried out by a professional drone pilot and camera operator with a long history of drone cinematography, including work on multiple television studio productions based in the Los Angeles area (Internet Movie Database, undated). His participation provided both the remote camera system and technical expertise necessary to ensure high-quality video and still-image shots of the vehicle.

To ensure that the images captured in the photo shoot were similar to those of the artificial *Arma 3* data set, various scenes were considered, including open areas, scenes in which the vehicle was partly occluded (e.g., by trees), and scenes with similarly shaped decoy objects, such as nonmilitary vehicles. The angle and altitude of the drone were also varied to provide supplementary perspectives. In addition, vehicle positions were repeated throughout the day to provide differing lighting and shadowing effects. The photo shoot lasted approximately five hours, during which time the project team generated approximately 22 GB worth of data. The collected data had more than 1,000 still images and also a variety of tracking (vehicle in motion) and orbiting (stationary vehicle) videos from which additional still images could be extracted. An example of four such images is shown in Figure 2.2. As in Figure 2.1, each image contains a single HMMWV surrounded by a black bounding box. To give the reader a better sense of how the real and artificial images compare, in Figure 2.3, we show a close-up side-by-side comparison of HMMWVs from the data sets. Lastly, these images were hand-labeled so that object-detection algorithms could be trained on them. We estimate that the hand-labeling took about 30 seconds per image.

Figure 2.2. Example of Photo Shoot Images

NOTE: Examples of the real photo shoot images taken in Yermo, California. Each image depicts a single HMMWV contained within a black bounding box.

Figure 2.3. Side-by-Side Comparison of Real and Artificial Images

NOTE: Close-up images of a real HMMWV (left) and an artificial HMMWV generated using *Arma 3* (right).

Object Classes

We decided to restrict our attention to a single object class—a two-door HMMWV. More realistically, we might expect an operationally relevant object-detection system to be able to identify a more-diverse range of objects, including both civilian vehicles (such as cars and trucks) and a wider range of military assets (such as mobile SAM launchers and armored transports). One of the advantages of *Arma 3* is the availability of many such assets. Ultimately, we only used the images for a single HMMWV class (the HMMWV M1152 ECV class in *Arma 3*) so that we could compare against the real photo shoot images. Both the real HMMWV we used and the *Arma 3* model were two-door models. Our focus in this work was to explore whether artificial image training sets could be used for operationally relevant object detection. To carry out such an analysis, it is crucial to be able to compare the performance of artificially trained models against benchmark models trained on real data. We were able to rent a HMMWV for a day and collect real images that enabled such a comparison. Renting a mobile SAM launcher of any variety would be quite another matter.

Data Set Summary

Our data collection efforts resulted in three distinct image data sets—a set consisting of 640 artificial *Arma 3* images, a set consisting of 418 real photo shoot images, and another set of 214 real photo shoot images that we obtained from still frames of high-definition video taken during the Yermo photo shoot. The still frames were used as a testing set, whereas the other two sets were used for training, as summarized in Table 2.1.

Table 2.1. Data Sets

Name	Number of Images
Arma–train	640
Photo shoot–train	418
Photo shoot–test	214

Unfortunately, these data sets are quite small by modern computer vision standards, which often vary from hundreds of thousands to millions of images. The size of the photo shoot data sets was limited by the need to hand-label them. This was expected, as the need for time-consuming and costly hand-labeling was one of the motivations for this project. The size of the photo shoot test data set that was created from still frames of videos we took in Yermo was also limited by the need to avoid including overly similar images. To address this, we took just one frame for every five-second interval, corresponding to one frame out of every 120 for the 24 frames per second video we recorded.

The small number of the *Arma 3* images is more disappointing given our expectation of being able to quickly and automatically obtain large numbers of these by scripting. However, the collection of *Arma 3* images faced a similar difficulty that was faced in extracting still frames from video. Although the data set collection and labeling steps were automated, some work was needed to set up each environment. Within each environment, multiple images were taken from a variety of camera angles and heights (as described earlier). The general requirement that the images not be too similar to one another severely limited our ability to collect large numbers of images. However, we believe that this issue could be surmounted by better automated scripting and that our basic approach could be scaled up to generate larger data set sizes.

3. Methodology

Our main goal in this work was to investigate the domain adaptation of high-quality artificial images for operationally relevant applications. Although some theoretical results do exist for the general problem of domain adaptation in machine learning (e.g., Daumé and Marcu, 2006; and Mansour, Mohri, and Rostamizadeh, 2009), we are not aware of any such results that address the particular object-detection problem in which we are interested. Indeed, many recent studies on domain adaptation in computer vision rely on empirical investigations rather than theoretical analysis. Therefore, we followed a similar approach and performed an empirical investigation that relied on training multiple neural network models. In this chapter, we discuss the methodological details of our approach; and, in Chapter 4, we present the results of our experiments.

Mask-RCNN Object-Detection Algorithm

The implementation of a modern, state-of-the-art computer vision algorithm represents a significant software-engineering challenge. We decided to use an existing open-source implementation so that we could focus on the domain adaptation question that motivated this project. We chose to use an open-source implementation of the region-based convolutional neural network (RCNN), known as Mask-RCNN (He et al., 2020), provided by Facebook Artificial Intelligence Research (FAIR) (Github, undated-b). We also explored a second algorithm, single shot detection (SSD), but ultimately chose to narrow our focus to a single algorithm because Mask-RCNN consistently outperformed SSD on our image set (Wei et al., 2016).

The Mask-RCNN algorithm was developed by FAIR researchers in 2017 as an improvement over a previous algorithm, Faster-RCNN (Ren et al., 2015), which was itself an improvement over Fast-RCNN (Girshick, 2015). Briefly, when given an image (of variable size) as input, the Mask-RCNN algorithm returns predictions in the form of bounding boxes and segmentation masks around candidate objects within the image. Segmentation masks are more fine-grained predictions of per-pixel class assignments. We chose not to work with these, however, because we found it much more convenient to work with bounding boxes for the labeled data. An example of a bounding box is shown in Figure 3.1. The black rectangle represents the "ground truth," which we determined manually by using labeling software to draw a bounding box around the object. The yellow-green box represents the Mask-RCNN prediction, and, in this case, the two boxes almost perfectly coincide.

Figure 3.1. Bounding Box Object Detection

NOTE: An example of a correct detection. The black bounding box was drawn by a hand labeler. The yellow-green box represents the prediction of the Mask-RCNN algorithm. Note that, in addition to predicting the location of the image, the algorithm predicts the class (two-door HMMWV) and reports a confidence estimate of this prediction (in this case, 100 percent).

Evaluation Criteria

To train the algorithm, we needed a way to compare the predicted bounding box against the ground truth bounding box. Ideally, the two boxes would completely coincide, although requiring this to be the case for the detection to be considered successful is an overly high standard. The extent to which the two boxes agree is measured in terms of the *intersection over the union* (IoU). As the name implies, this is simply the ratio of the area of the intersection of the two bounding boxes to the area of the union:

$$IoU = \frac{area\ of\ the\ intersection}{area\ of\ the\ union}.$$

This is illustrated in Figure 3.2. As noted earlier, the ground truth is represented by the black rectangle, and the prediction made by a notional algorithm is shown in the yellow-green box. The intersection is represented by the red square, and the IoU value is computed by comparing the area of this to the total area contained within the two boxes. In practice, a threshold IoU value is set so that a prediction will only be considered correct (a true positive) if the predicted bounding box has sufficient overlap with the ground truth box. Note that the IoU value is separate from the output of the Mask-RCNN algorithm, and it is only used to determine whether

a prediction is correct or not. Also, if the algorithm predicts an object in an empty region of the image, then the IoU value is zero.

Figure 3.2. IoU Example

NOTE: It is rare for the algorithm's prediction (yellow-green box) to align exactly with the ground truth (black box). The IoU metric is used to quantify such imperfect predictions as this one. The IoU is defined as the area of the intersection (shared area outlined in red) divided by the total area of box black and green boxes. In this case, the IoU is 0.3.

The performance of an object-detection algorithm such as Mask-RCNN is typically measured in terms of both precision and recall, which are defined as:

$$Precision = \frac{TP}{TP+FP}, \qquad Recall = \frac{TP}{TP+FN},$$

where TP, FP, and FN stand for the number of true positive, false positive, and false negative detections, respectively. Both precision and recall range from 0 to 1, with 0 being obtained if no correct identifications are made. A perfect precision value of 1 is obtained when no false positive errors are made, whereas a perfect recall value of 1 is obtained when no false negatives are obtained. Thus, each score alone does not provide sufficient insight into the model performance, and both scores must be used to gain a complete picture.

The precision and recall values depend on the confidence threshold needed to make a bounding box prediction, and they also depend on the IoU threshold just discussed, which determines whether a prediction is correct or not. It is standard practice in computer vision research to report results for averaging over a range of these values, leading to the "mean average precision" (mAP) and "mean average recall" (mAR) scores. Throughout this work, we

use the evaluation criteria developed as part of the COCO Detection Challenges.[1] In particular, the mAP and mAR scores were computed by averaging over the following ten IoU threshold values:

$$IoU \in (0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95) .$$

Implementation Details

As mentioned earlier, we used the Mask-RCNN object-detection algorithm. We chose to use a publicly available implementation created by FAIR. In 2018, FAIR released the code base *Detectron*, which implements many state-of-the-art object-detection algorithms, including Mask-RCNN.[2] *Detectron* is written in Python and uses the Caffe2 deep learning framework.

We used the same training details for each model we trained to isolate, as best as possible, the effect of the artificial images in the training data. All training and inference were performed using a virtual machine available from AWS, specifically a g3.4xlarge EC2 instance, which we rented for $1.14 per hour and employed a single Nvidia Tesla M60 GPU. We decided to initialize the Mask-RCNN network using pretrained weights from an R-50 Feature Pyramid Network trained on ImageNet (Lin et al., 2017). The weights were then additionally trained (fine-tuned) on the various data sets we considered. The additional training consisted of 300,000 iterations, with each iteration made using a batch size of two images. Model checkpoints were saved every 50,000 iterations, and the checkpoint with the best performance on the test set was used for the results reported in the next section.[3] The training time varied with the size of the data set, with each model taking roughly three days to reach 300,000 training iterations. We mostly used the default settings and hyperparameter values for the Mask-RCNN by *Detectron.* More details on these values may be found in Appendix B and in the original *Detectron* documentation (Github, undated-a).

[1] For the technical details of how mAP and mAR are defined for the COCO challenges, see Common Objects in Detection, undated.

[2] In late 2019, a new version of *Detectron* was released, *Detectron2*. The first version is now deprecated.

[3] Because of the limited number of images from a real photo shoot, we did not use a separate validation set here.

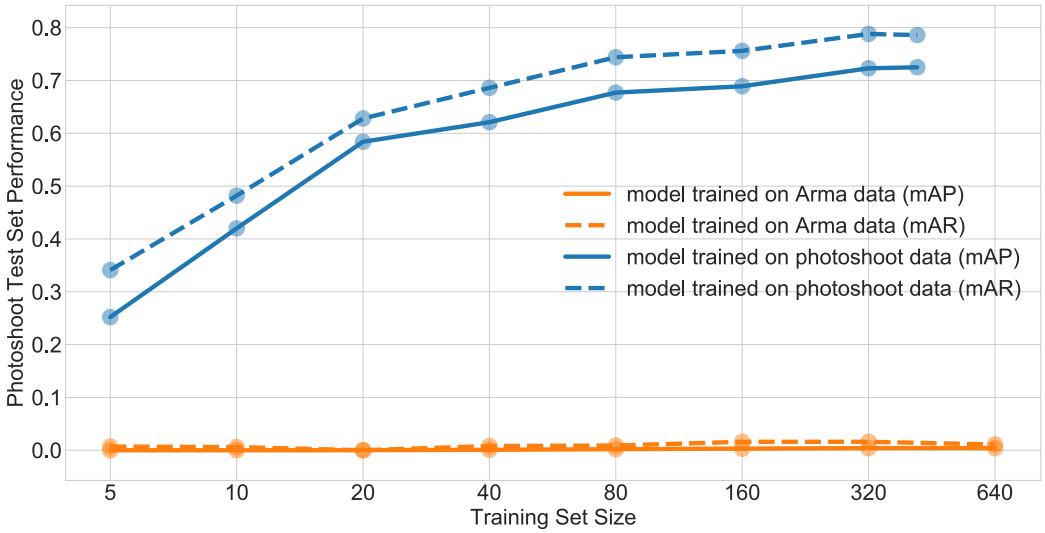
4. Results

In this chapter, we present the results of our numerical experiments. We collected data on the performance of the Mask-RCNN model when trained on several distinct data sets, including data sets consisting of purely real images, purely *Arma3* images, and hybrid data sets made up of a mix of both types of images.

Training on *Arma 3* Data

In Figure 4.1, we present our first major result. This plot compares the performance of models trained on real HMMWV data against the performance of models trained on artificial data generated from *Arma 3*. Both models are evaluated on the same test data set consisting of real HMMWV images. Unfortunately, the plot clearly indicates that domain adaptation has failed to occur; the models trained on artificial data do not transfer to the real test set. This failure is rather dramatic—despite the apparent similarity of the images to the eye, both the mAP and mAR scores of the *Arma 3*–trained models are essentially zero.

Figure 4.1. Performance of Mask-RCNN Algorithm on Real Photo Shoot Images



NOTE: The performance of the Mask-RCNN algorithm evaluated on real photo shoot images. The performance of two models are shown. The blue curves correspond to the model trained on real photo shoot images, and the orange curves correspond to the model trained on artificial *Arma 3* images. The first model performs reasonably well on the testing data set, whereas the second one fails completely. The performance is measured in terms of both mean average precision (mAP, solid lines) and mean average recall (mAR, dotted lines), both of which were computed by averaging over a range of IoU values as described earlier in this report.

These results establish quite clearly that models trained on *Arma 3* images only *do not* continue to perform well when the domain is modified to be real-world HMMWV images. Thus one of the main motivations for this work is not borne out in practice. Of course, we cannot prove that this approach will never work—indeed, there is always the possibility that, with a different problem setup, the domain adaptation might work. In fact, we cannot even conclusively rule out the current setup from ever working. It could be the case that the results would qualitatively change with more training time, the use of larger data sets, or a different underlying object-detection algorithm, although we see no reason to expect this to be the case. Another potential cause for the failure of *Arma 3* images to transfer is the fact that our object-detection algorithm uses a pretrained ImageNet model, which we then fine-tune. We discuss potential remedies for this failure in Chapter 5.

Training on Hybrid Data

Although the results noted earlier show that models trained on *Arma 3* images do not transfer to real HMMWV images, it could still be the case that the artificial images are useful when used in combination with real images. Here, we study whether a hybrid training set that consists of a mix of real and *Arma 3* images can lead to improved performance over just real images alone. Concretely, we took both the *Arma 3* and real-world training sets and split them into nine subsets of images. Letting N_A, N_P denote the number of *Arma 3* and photo shoot images, respectively, we assembled hybrid data sets with N_A, N_P images of each type. To study the effect of the ratio of real to artificial images, we considered the following ranges of these parameters:

$$N_P = 0, 5, 10, 20, 40, 80, 160, 320, 418,$$
$$N_A = 0, 5, 10, 20, 40, 80, 160, 320, 640.$$

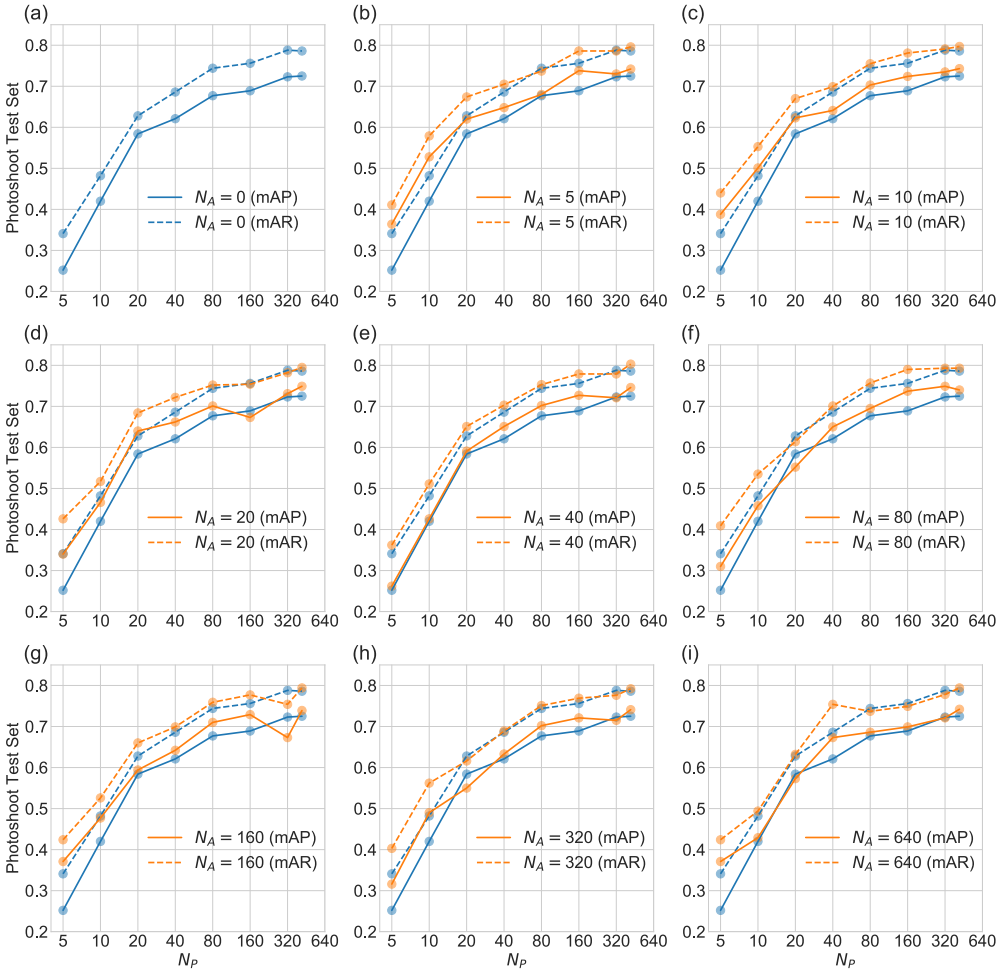
We then used these smaller subsets to form derivative data sets with (N_A, N_P) images of each kind. There are nine subsets for each data set, so there are 81 such hybrid data sets. However, the (0,0) case is just the empty set, leading to 80 nontrivial hybrid data sets. The data sets with $N_A = 0$ or $N_P = 0$ correspond to the pure photo shoot or pure *Arma 3* data sets considered earlier.

The results of training 80 models on each of these data sets are depicted in the multiple plots of Figure 4.2. The top-left plot shows the performance of the model trained on purely real images for reference. All other plots additionally show the performance of a model trained on hybrid data sets, with the number of artificial *Arma 3* images N_A held fixed. The hybrid data in all cases are performing similarly to the real data. In many cases, the hybrid data lead to a noticeable improvement, while adding artificial images actually hurts the performance in a few other cases.

In more detail, of the 80 models we trained, 64 of them correspond to hybrid data sets with *Arma 3* and photo shoot images. For each of these, the performance may be compared against the corresponding experiment with no *Arma 3* images. We found that the addition of *Arma 3* images

improves the mAP score in 56 out of 64 experiments, corresponding to 87.5 percent of cases. The mAR score is improved in 52 out of 64 experiments, corresponding to 81.3 percent of cases. The most significant improvements seem to occur when the number of real images is very low. For example, in Figure 4.2c, adding ten *Arma 3* images to five photo shoot images led to a 54-percent mAP improvement and a 29-percent mAR improvement. We have thus found that, for the most part, adding *Arma 3* images to an existing data set of real photo shoot images helps with model performance. Therefore, these results suggest that our approach could be used to improve the performance of an operationally relevant computer vision system, especially in extreme data-limited settings where only a few high-quality real images are available.

Figure 4.2. Hybrid Results: mAP, mAR Curves



NOTE: The performance of the models trained on the hybrid data sets containing both real and artificial *Arma 3* images. The top-left plot depicts the case in which all the training images are real, and this set of curves are also included in all other plots to allow for a comparison against the performance of hybrid models. Holding fixed the number of photo shoot images, the addition of *Arma 3* images improves performance in the majority of cases. Specifically, 87.5 percent and 81.3 percent of the measurements of mAP, mAR, respectively, are larger for the hybrid model than for the pure photo shoot model.

Of course, it is necessary to assess the statistical significance of these results to confidently conclude that training on hybrid data sets improves performance. Assuming that the mAR and mAP score values are noisy, our null hypothesis is that the addition of the *Arma 3* images would have no real impact on the performance (measured in terms of mAR or mAP) and that the observed results would be purely random. Under this hypothesis, one would expect that adding *Arma 3* images would lead to an improvement over the pure photo shoot baseline 50 percent of the time on average. Introducing the subtracted scores

$$\tilde{y}(N_P, N_A) = y(N_P, N_A) - y(N_P, N_A = 0),$$

where y can be either the mAP or mAR value, under the null hypothesis $\tilde{y}$, will be equally likely to be positive or negative. Therefore, the outcomes of the 64 experiments with $N_A \neq 0$ can be regarded as draws of a Bernoulli random variable with probability $p = 0.5$. The p-values may then be computed to be 5.6×10^{-10}, 4.6×10^{-7} for mAP, mAR, respectively. Thus, we can reject the null hypothesis with high probability. Importantly, this result does not imply that adding *Arma 3* images always helps, only that the observed data are very unlikely under the null hypothesis.

Table 4.1. *p*-Values for Significance Testing

	mAP	mAR
p-value	5.6×10^{-10}	4.6×10^{-7}

Having established that the improvement obtained by adding *Arma 3* images is statistically significant, the natural next step is to measure the expected improvement as a function of N_A, N_P. We expect that adding *Arma 3* images should be more beneficial when the data set size is limited. In fact, we can reason that the addition of *Arma 3* images should actually hurt performance if the ratio of artificial to real images became too large (i.e., as $N_A/N_P \rightarrow \infty$) because, in this case, the problem reduces to the pure domain adaptation case considered earlier; also, the results depicted in Figure 4.1 clearly show that, in this case, the models trained on *Arma 3* images only did not transfer to real images. In future work, we would like to more accurately measure these relationships. Unfortunately, our data are too limited and too noisy to provide any useful insight into the functional form of the (N_A, N_P) dependence.

In summary, our main findings are that models trained on *Arma 3* do *not* transfer to data sets of real images, whereas models trained on hybrid data consisting of a mix of both real and *Arma 3* images do transfer to data sets of real images. The addition of *Arma 3* images to a training data set generally leads to an improvement in performance. The amount of improvement varies, but for small data sets, we observed performance boosts as high as 54 percent and 29 percent for mAP and mAR, respectively.

5. Conclusions

The initial motivation for this project was to investigate whether models trained on artificial images could perform well when evaluated on data sets of real-world images. Focusing on the operationally relevant task of identifying HMMWVs from overhead aerial images, we chose to use the video game platform *Arma 3* as our source of high-quality artificial images. To address this, we developed a software pipeline that enables the automatic generation *and* labeling of *Arma 3* images. This pipeline was necessary for our domain adaptation investigation, and we believe it also should be useful for other machine learning tasks.

Using this pipeline, we generated a data set of artificial HMMWV images, which we then used to train series of object-detection models. To evaluate the performance of these models, we assembled a second data set of real-world aerial images by renting a HMMWV and contracting a photographer who used a drone-mounted high-resolution camera. The conditions of this photo shoot were chosen to closely imitate the scenes generated by *Arma 3* to make a fair comparison between models trained on the two different data sets. We split this set of real images into a training and testing set and used the training set to train a series of models, just as had been done for the artificial images. We then used the testing set of real images to compare the two series of models. We very clearly found—for the number of training images that we used and the pretrained FAIR model—that the artificial training *does not* transfer to real-world images. The dramatic failure of the artificially trained models is displayed in Figure 4.1.

We then investigated the transferability of what we called hybrid data sets—that is, whether a model trained on real-world images could be improved by adding artificial images to the data set. The results of our experiments suggest that, in many cases, the addition of *Arma 3* does indeed improve performance, especially when the number of real-world images is severely limited. We found that the *Arma 3* images improved the precision 87.5 percent of the time and the recall 81.3 percent of the time. The size of the improvement can be quite significant. For example, by adding ten *Arma 3* images to a data set consisting of five real images, we were able to improve mAP and mAR by about 54 percent and 29 percent, respectively. Even with this boost, the performance of the models trained on limited data sets is probably not high enough for them to be used in real operational scenarios. However, our result does suggest a technique for improving the training of models when existing data are quite scarce and there is a source of high-quality artificial images. More work is needed to fully understand this effect. In particular, more experiments are needed to estimate the variance of the performance boost and to conduct more-rigorous statistical significance testing.

Limitations

This work represents an exploratory investigation into the transferability of artificially generated images for defense applications of neural network–based object-detection systems. Therefore, there are several limitations and shortcomings, and we caution against overgeneralization of our results. Concretely, we only considered one source of artificial images (*Arma 3*), and we also mostly restricted our attention to a single object class (HMMWVs). It is entirely possible that different results would have been obtained for a different source of artificial images and for a different set of object classes.

Regarding our implementation, one major limitation is that the data set sizes are rather small by computer vision standards. One of the initial motivations for this study was that artificial data sources could be used to generate truly massive data sets, which would be prohibitively costly to obtain in the real world. Because of the time constraints, we were unable to realize this initial goal, and the final training set of usable *Arma 3* images numbered in the hundreds for the single HMMWV class we used for our main results. The key point is that the resources (measured in time, money, and man-hours) needed to generate a much-better labeled data set scale for artificial images than for real images. Thus the benefit obtained by using our approach should be significant for large-scale projects.

Another major limitation is that our results are noisy and depend on several variables, such as the precise set of images used for training and evaluation, the initial parameter values of the neural network, and the many hyperparameters used to train the neural network, such as the learning rate schedule or the batch size. Ideally, for each datapoint in the plots noted earlier, we would have run many experiments with different values for these variables and averaged over the results. This would also allow us to obtain error bars, although we were able to carry out a basic significance test to verify that our results were unlikely to occur by random chance alone. With more experiments, we could go beyond this and measure the dependence of the precision and recall on the number of real and artificial images.

A final limitation is that we did not attempt to optimize over the hyperparameters used in the Mask-RCNN algorithm, using instead the default values used in the open-source implementation provided by FAIR. In particular, we initialized our networks with pretrained weights. It is possible that our results could be improved by starting with randomly initialized networks. Although acknowledging this out-of-the-box approach as a limitation, we purposefully sought to use publicly available, open-source implementations of modern computer vision algorithms both for convenience and because we wished to investigate whether artificial training could succeed with only a moderate investment of resources. As a consequence of this approach, the broader question of whether or not it is possible *in principle* for domain adaptation to succeed in our setup remains unanswered. Of course, given that this was the initial motivation for this work, we still believe that this is possible in principle, and our results underscore the wide divide that often separates *in principle* from *in practice*, especially in machine learning. In the next and final

section, we discuss some recent results in the computer vision literature that might help to bridge this gap.

Open Questions and Future Directions

This work was motivated by the fact that, in times of conflict, it might be challenging to quickly acquire the large number of labeled images required to train an AI/ML-based ATR system tailored to the conflict in question. We investigated whether this obstacle could be avoided by training neural network–based object-detection models using either artificial images or a mixture of both artificial and real images. We found that, for a fixed budget of real images, the performance of the model can be improved by adding some additional artificial ones—moreover, the performance boost is greatest when there are just a few real images (i.e., five to ten). Therefore, a key issue is whether this performance boost is enough to declare this method useful in real operational scenarios. We are not aware of any performance guidelines that an ATR system must satisfy in order for it to be deployed. However, in the case where there are only five to ten real images, our experiments yield values of the precision and recall in the range [0.4, 0.6]. These values hardly inspire the level of confidence needed to trust the ATR system in real combat settings. As a result, the artificial training explored in this work would likely not be immediately useful in a real operational scenario with severely limited data. However, such a system might still be of value as a "second set of eyes" to help the analyst, given that analysts may be similarly challenged in their performance by the lack of real images for training.

We believe this study produced interesting technical results that raise several questions and possible directions for future work. First, as discussed earlier, despite the significant amount of computing resources we used, we have not fully explored this problem. With more time and resources, it would be good to extend our results to larger data sets and longer training runs and to also train multiple models for a given number of real and artificial images. This would allow us to avoid drawing conclusions from unusually high- or low-performing instances, and it would also allow for error bars to be obtained.

The major question concerning our results is whether the failure we observed for the transferability of pure *Arma 3* images was a fundamental failure because the images were not sufficiently similar to real-world images or whether the failure was the result of the many implementation choices we made throughout this study. In attempting to answer this question, we became aware of other research that found a similar result, specifically a failure of domain adaptation for object detection (see, for example, Bousmalis et al., 2017). One possible explanation for the failure in our experiment is suggested by the recent work of Geirhos et al. (2019), who found that ImageNet-trained classifiers are strongly biased toward recognizing textures rather than shapes. Therefore, it could well be the case that the failure of the model trained on pure *Arma 3* images to transfer to real images is primarily due to the difference in the textures of the real and artificial images, as shown in Figure 2.3.

Regardless of the cause for this failure, there has been recent progress on the domain adaptation problem, which might be able to address this issue. One large class of approaches involves attempting to learn a map, which transforms one of the domains into the other. This approach could be applied to the raw data itself or to intermediate representations (e.g., see Kulis and Darrel, 2011; Duan, Xu, and Tsang, 2014; Hoffman et al., 2013; Hoffman et al., 2014; Long et al., 2015; and Bousmalis et al., 2017). Closely related to these works is Shrivastava et al. (2017), who used generative adversarial networks to improve the realism of synthetic training data and found that the improved synthetic images then led to better-performing models. Another recent approach is to randomize the conditions of the simulator, such as lighting, pose, textures, and other elements (Tremblay et al., 2018). It would be interesting to see whether these approaches might allow the models trained on purely *Arma 3* images to perform well on real images.

A final question concerns the reason for the success of the hybrid-trained models to transfer to real images. It is perhaps unsurprising that these models transfer better than those trained on purely artificial images, because in the hybrid case, the two domains are closer to one another (i.e. the training domain is a mixture of real and artificial images, and the testing domain is real images). However, in light of the failure for purely *Arma 3*–trained models to transfer, it is interesting that, for fixed numbers of real images N_P, the addition of artificial images tends to improve the performance. It would be beneficial to have an understanding for this apparent discrepancy and also to better quantify the magnitude of the improvement and how it depends on the numbers of real and artificial images in the training data set.

Appendix A. The U.S. Military's Use of Bohemia Interactive Products

Bohemia Interactive products have integrated with U.S. and foreign militaries and agencies since the early 2000s. In parallel to the 2003 commercial success of *Operation Flashpoint,* an *Arma 3* predecessor, the Defense Advanced Research Projects Agency began work with Bohemia Interactive on military training simulations based on the Real Virtuality engine. This effort resulted in the creation of Bohemia Interactive Simulations (BISim), first as a division of the parent studio and then as an independent company after its purchase by the Riverside Company investment firm in 2013. Throughout its history, BISim has released products that are customized to military training tasks in close cooperation and partnership with Bohemia Interactive development in addition to releasing new commercial gaming products. The Virtual Battlespace series (VBS) is BISim's flagship product and is currently in use by more than 50 militaries around the world. Although the *Arma* series is set in fictional scenarios and uses notional weapon systems (real-world models are available as user-generated modifications), VBS includes models of over 14,000 current and historical systems aligned to known military forces and formations. Importantly, the underlying engine (Bohemia Interactive's Real Virtuality engine), game mechanics, and general functionality are nearly identical between *Arma* and VBS; models built in Status Quo Function (the variant of C used for *Arma 3*) can be cross-loaded between the two platforms. Costs for individual licenses of VBS3, the latest in the series, begin at approximately $3,000.

The U.S. Marine Corps first adopted VBS1 for operational and tactical training in 2006, followed quickly by VBS2 (2008), and VBS3 (2015). Ongoing efforts to integrate all Marine Corps simulation training (including such legacy systems as *Pointman*) with VBS3 virtual reality and augmented reality capabilities are sponsored by the Director, Expeditionary Warfare of the Office of the Chief of Naval Operations, and are being conducted as a joint effort with BISim, the Office of Naval Research, Program Executive Officer–Expeditionary Warfare, and the Marine Corps Training and Education Command. Currently, VBS3 comprises the backbone of the Infantry Tool Kit portion of the Deployable Virtual Training Environment.

VBS3 deployment for the U.S. Army is currently managed by the Program Manager, Integrated Training Environment, and overseen by the U.S. Army Program Executive Officer, Simulation, Training and Instrumentation (PEO STRI). BISim was awarded an Other Transaction Authority contract extension to develop cloud-based systems to augment the capabilities to be delivered as part of BISim's 2017 five-year contract with PEO STRI (BISim, 2017). Although the U.S. Army is in the midst of a redesign of its Synthetic Training Environment, every available indication is that this environment is intended to be largely built around the VBS system (PEO STRI, undated).

In addition to integration with the military services outlined earlier, BISim maintains contracts with the U.S. Navy and the U.S. Air Force to provide capabilities based on the VBS series (BISim, undated). Because the complementary tools that these services rely on are fundamentally based on the Real Virtuality engine and are therefore compatible with the commercial Bohemia Interactive products, *Arma 3* provides a cost-effective proxy for VBS and the simulated training environments.

Appendix B. Additional Model and Hyperparameter Details

In this appendix, we provide additional technical details on the Mask-RCNN model used and the various hyperparameter values we employed. For the sake of reproducibility, we include the content of the .yaml configuration file that we used to train our models.

```
MODEL:
  META_ARCHITECTURE: "GeneralizedRCNN"
  WEIGHT:
"catalog://ImageNetPretrained/MSRA/R-50"
  BACKBONE:
    CONV_BODY: "R-50-FPN"
    OUT_CHANNELS: 256
  RPN:
    USE_FPN: True
    ANCHOR_STRIDE: (4, 8, 16, 32, 64)
    PRE_NMS_TOP_N_TRAIN: 2000
    PRE_NMS_TOP_N_TEST: 1000
    POST_NMS_TOP_N_TEST: 1000
    FPN_POST_NMS_TOP_N_TEST: 1000
  ROI_HEADS:
    USE_FPN: True
  ROI_BOX_HEAD:
    POOLER_RESOLUTION: 7
    POOLER_SCALES: (0.25, 0.125, 0.0625, 0.03125)
    POOLER_SAMPLING_RATIO: 2
    FEATURE_EXTRACTOR:
"FPN2MLPFeatureExtractor"
    PREDICTOR: "FPNPredictor"
  ROI_MASK_HEAD:
    POOLER_SCALES: (0.25, 0.125, 0.0625, 0.03125)
    FEATURE_EXTRACTOR:
"MaskRCNNFPNFeatureExtractor"
    PREDICTOR: "MaskRCNNC4Predictor"
    POOLER_RESOLUTION: 14
    POOLER_SAMPLING_RATIO: 2
    RESOLUTION: 28
    SHARE_BOX_FEATURE_EXTRACTOR: False
  MASK_ON: True
DATASETS:
  TRAIN: ("train",)
  TEST: ("test",)
DATALOADER:
  SIZE_DIVISIBILITY: 32
SOLVER:
  BASE_LR: 0.0025
  WEIGHT_DECAY: 0.0001
```

STEPS: (480000, 640000)
MAX_ITER: 720000
IMS_PER_BATC

References

BISim—*See* Bohemia Interactive Simulations.

Bohemia Interactive Community Wiki, "Introduction to Arma Scripting," webpage, undated. As of November 18, 2019: https://community.bistudio.com/wiki/Introduction_to_Arma_Scripting

Bohemia Interactive Simulations, "VBS3: Virtual Desktop Training & Simulation Host," webpage, undated. As of November 18, 2019: https://bisimulations.com/products/virtual-battlespace

———, "U.S. Army Selects BISim for a 5-Year Prime Contract to Support Games for Training Program," press release, Orlando, Fla., August 28, 2017. As of November 18, 2019: https://bisimulations.com/company/news/press-releases/mon-08282017-0238/us-army-selects-bisim-5-year-prime-contract-support-games-training-program

Bohemia Interactive Store, "Arma 3 Editions," webpage, undated. As of November 18, 2019: https://store.bistudio.com/category/arma3-editions

Bousmalis, K., N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks," in the *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, Hawaii, July 2017, pp. 3722–3731.

COCO—*See* Common Objects in Detection.

Common Objects in Detection, "Detection Evaluation," webpage, undated. As of August 6, 2020: https://cocodataset.org/#detection-eval

Daumé, H., III, and D. Marcu, "Domain Adaptation for Statistical Classifiers," *Journal of Artificial Intelligence Research*, Vol. 26, 2006, pp. 101–126.

Defense Logistics Agency, "DLA Distribution Barstow, California," webpage, undated. As of November 18, 2019: https://www.dla.mil/Distribution/Locations/Barstow/

Duan, L., D. Xu, and I. Tsang, "Learning with Augmented Features for Heterogeneous Domain Adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 36, No. 6, June 2014. As of March 25, 2020: https://arxiv.org/ftp/arxiv/papers/1206/1206.4660.pdf

Geirhos, R., P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, *ImageNet-Trained CNNs Are Biased Towards Texture; Increasing Shape Bias Improves Accuracy and Robustness*, paper presented at International Conference on Learning Representations, New Orleans, La., May 6–9, 2019. As of March 25, 2020: https://arxiv.org/pdf/1811.12231.pdf

Girshick, Ross, "Fast R-CNN," *Conference Proceedings of IEEE International Conference on Computer Vision (ICCV)*, New York: Institute of Electrical and Electronics Engineers, December 2015, pp. 1440–1448.

Github, "facebookresearch / Detectron," webpage, undated-a. As of March 26, 2020: https://github.com/facebookresearch/Detectron

———, "facebookresearch / maskrcnn-benchmark," webpage, undated-b. As of November 18, 2019: https://github.com/facebookresearch/maskrcnn-benchmark

Greenstein, Zvi, "The Test Fleet of the Future Is Virtual: DRIVE Constellation Now Available," *NVIDIA Blog*, March 18, 2019. As of March 26, 2020: https://blogs.nvidia.com/blog/2019/03/18/drive-constellation-now-available/

hcpookie, "POOK Camonets," webpage, undated. As of November 18, 2019: http://www.armaholic.com/page.php?id=29908

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, "Mask-RCNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 42, No. 2, February 1, 2020. As of August 6, 2020: https://ieeexplore.ieee.org/document/8372616/keywords#keywords

Hoffman, Judy, Erik Rodner, Jeff Donahue, Trevor Darrell, and Kate Saenko, *Efficient Learning of Domain-Invariant Image Representations*, paper presented at International Conference on Learning Representations, Scottsdale, Ariz., May 2–4, 2013. As of March 25, 2020: https://arxiv.org/abs/1301.3224

Hoffman, Judy, Sergio Guadarrama, Eric S. Tzeng, Ronghang Hu, Jeff Donahue, Ross Girshick, Trevor Darrell, and Kate Saenko, "LSDA: Large Scale Detection Through Adaptation," *Advances in Neural Information Processing Systems 27*, 2014, pp. 3536–3544.

Internet Movie Database, "Marcus Hunt," webpage, undated. As of November 18, 2019: https://www.imdb.com/name/nm2910198/

Kulis, B., K. Saenko, and T. Darrell, "What You Saw Is Not What You Get: Domain Adaptation Using Asymmetric Kernel Transforms," in the *2011 IEEE Conference Proceedings of the CVPR 2011,* Providence, R. I., June 2011, pp. 1785–1792.

Lin, Tsung-Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature Pyramid Networks for Object Detection," *Conference Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, New York: Institute of Electrical and Electronics Engineers, 2017, pp. 936–944.

Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg, "SSD: Single Shot Multibox Detector," *Proceedings of the European Conference on Computer Vision (ECCV) 2016*, Cham, Switzerland: Springer, 2016.

Long, M., Y. Cao, J. Wang, and M. I. Jordan, "Learning Transferable Features with Deep Adaptation Networks," in the *Proceedings of the 37th Annual International Conference on Machine Learning*, Lille, France, July 7–9, 2015. As of March 25, 2020: http://proceedings.mlr.press/v37/long15.pdf

Mansour, Y., M. Mohri, and A. Rostamizadeh, "Domain Adaptation: Learning Bounds and Algorithms," in the *Proceedings of the 22nd Annual Conference on Learning Theory*, Montreal, Canada, 2009. As of March 25, 2020: https://www.cs.mcgill.ca/~colt2009/papers/003.pdf

OpenAI, Akkaya, Ilge, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang, "Solving Rubik's Cube with a Robot Hand," *OpenAI Blog*, October 15, 2019. As of March 25, 2020: https://openai.com/blog/solving-rubiks-cube/

Pellerin, Cheryl, "Project Maven to Deploy Computer Algorithms to War Zone by Year's End," press release, *Defense News*, U.S. Department of Defense webpage, July 21, 2017. As of July 3, 2018: https://www.defense.gov/News/Article/Article/1254719/project-maven-to-deploy-computer-algorithms-to-war-zone-by-years-end/

PEO STRI—*See* U.S. Army Program Executive Officer for Simulation, Training and Instrumentation.

poolpunk, "PLP Containers," webpage, undated. As of November 18, 2019: http://www.armaholic.com/page.php?id=29294

Qiu, W., and A. Yuille, "UnrealCV: Connecting Computer Vision to Unreal Engine," in Hua G., Jégou H., eds., *Computer Vision – ECCV 2016 Workshops, Proceedings, Part I*, Amsterdam, The Netherlands: Springer, October 2016, pp. 909–916. As of August 6, 2020: https://link.springer.com/chapter/10.1007/978-3-319-49409-8_75

Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Suin, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds., *Advances in Neural Information Processing Systems,* Vol. 28, New York: Curran Associates, Inc., 2015, pp. 91–99.

Rolen, Lynette M., "New Artificial Intelligence Technology Assists Air Commandos with Decision-Making," Air Force Special Operations Command Public Affairs, September 13, 2017. As of July 3, 2018: http://www.afsoc.af.mil/News/Article-Display/Article/1309844/new-artificial-intelligence-technology-assists-air-commandos-with-decision-maki/

SBIR STTR—*See* Small Business Innovation Research Program/Small Business Technology Transfer Program.

Small Business Innovation Research Program/Small Business Technology Transfer Program, "Transfer Learning and Deep Transfer Learning for Military Applications," webpage, undated. As of March 26, 2020: https://www.sbir.gov/node/1621003

Shafaei, A., J. J. Little, and M. Schmidt, *Play and Learn: Using Video Games to Train Computer Vision Models*, Vancouver, Canada: Department of Computer Science, University of British Columbia, 2016. As of March 25, 2020: http://www.bmva.org/bmvc/2016/papers/paper026/paper026.pdf

Shrivastava, A., T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from Simulated and Unsupervised Images Through Adversarial Training," in the *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, Hawaii, July 2017, pp. 2242–2251.

Steam, "Arma 3," webpage, undated-a. As of November 19, 2019: https://store.steampowered.com/app/107410/Arma_3/

———, "CUP Terrains - Core," webpage, undated-b. As of November 18, 2019: https://steamcommunity.com/workshop/filedetails/?id=583496184

———, "Steam Community: Arma 3," webpage, undated-c. As of November 18, 2019: https://steamcommunity.com/app/107410/workshop/

Taylor, Geoffrey R., Andrew J. Chosak, and Paul C. Brewer, "Ovvv: Using Virtual Worlds to Design and Evaluate Surveillance Systems," in the *Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, Minn., June 2007, pp. 1–8.

Tremblay, Jonathan, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield, "Training Deep Networks

with Synthetic Data: Bridging the Reality Gap by Domain Randomization," in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, Salt Lake City, Utah, June 2018, pp. 969–977.

U.S. Army Program Executive Officer for Simulation, Training and Instrumentation, "Synthetic Training Environment (STE)," webpage, undated. As of November 18, 2019: https://www.peostri.army.mil/synthetic-training-environment-ste/

Automated target recognition (ATR) is one of the most important potential military applications of the many recent advances in artificial intelligence and machine learning. A key obstacle to creating a successful ATR system with machine learning is the collection of high-quality labeled data sets. The authors investigated whether this obstacle could be sidestepped by training object-detection algorithms on data sets made up of high-resolution, realistic artificial images. The authors generated large quantities of artificial images of a high-mobility multipurpose wheeled vehicle (HMMWV) and investigated whether models trained on these images could then be used to successfully identify real images of HMMWVs. The authors obtained a clear negative result: Models trained on the artificial images performed very poorly on real images. However, they found that using the artificial images to supplement an existing data set of real images consistently results in a performance boost. Interestingly, the improvement was greatest when only a small number of real images was available. The authors suggest a novel method for boosting the performance of ATR systems in contexts where training data are scarce. Many organizations, including the U.S. government and military, are now interested in using synthetic or simulated data to improve machine learning models for a wide variety of tasks. One of the main motivations is that, in times of conflict, there may be a need to quickly create labeled data sets of adversaries' military assets in previously unencountered environments or contexts.

$19.00

ISBN-10 1-9774-0595-9
ISBN-13 978-1-9774-0595-1

www.rand.org

RR-A683-1