Training Rare Object Detection in Satellite Imagery with Synthetic GAN Images

Eric Martinson, Bridget Furlong, Andy Gillies
Soar Technology
3600 Green St, Ste 600, Ann Arbor, MI, USA
{eric.martinson, bridget.furlong, andy.gillies}@soartech.com

Abstract

When creating a new labeled dataset, human analysts or data reductionists must review and annotate large numbers of images. This process is time consuming and a barrier to the deployment of new computer vision solutions, particularly for rarely occurring objects. To reduce the number of images requiring human attention, we evaluate the utility of images created from 3D models refined with a generative adversarial network to select confidence thresholds that significantly reduce false alarms rates. The resulting approach has been demonstrated to cut the number of images needing to be reviewed by 50% while preserving a 95% recall rate, with only 6 labeled examples of the target.

1. Introduction

Satellites capture the majority of the planet's surface every day, generating millions of images. Even knowing regional boundaries or terrain classification, searching such volumes of data for specific objects is impossible without computer image recognition. For objects that already have large supporting labeled training data, there are many existing solutions. But what if a human analyst needs to find a new class of object without an existing labeled dataset? Often, the simplest answer is to capture new imagery and label them by hand. But if the object is rare or new imagery cannot be captured, then it is very difficult to build a reliable model. Active learning [1] is one solution whereby people are still involved to label reduced numbers of images during training; algorithms cut the load by selecting images for human labeling with high potential gain. While much improved, these solutions still often require significant initial training data (e.g. 50 images) that may be impossible to acquire for some objects. In this paper, we detail the effectiveness of data reduction techniques that augment small numbers of labeled images (6 or 10) with 3D models modified with generative adversarial networks (GAN) to both classify images and detect objects within. In the longer term, this initial bootstrapping can support active learning to build up more capable models over time.

Ideal training data for deep neural network based classification models includes examples of an object in all environments of interest, with many variations in pose, lighting conditions, and levels of occlusion. But for many rare objects, that option simply does not exist. And unlike ground data, it is difficult to capture your own training data from satellites. One solution is to create synthetic images blending 3D models of the object of interest with a variety of backgrounds [2], using random selection to decide model location and background choice. We will demonstrate, however, that with only 6 labeled real examples the classification model quickly learns to recognize synthetic imagery without improving performance on real imagery.

One approach to improve the utility of 3D models is to employ generative adversarial networks. Generative adversarial networks like Cycle-GAN [3] can be used to transform existing images into something else. The classic examples are to convert an image of a horse into a zebra, or convert a summer driving image into winter [4]. The training process for a GAN is to provide examples of both media, training a set of networks to convert in both directions. By training a GAN on a multitude of related objects with more labeled training data, we aim to introduce common atmospheric distortion to our synthetic imagery. Although visually improved, we will demonstrate on the xView dataset [5] that the impact on training a convolutional neural network-based object detection network such as RetinaNet [6] with such synthetic images is not significant. However, such GAN-modified imagery will be demonstrated to be valuable as verification data for selecting the best training epoch and setting detection thresholds prior to real deployment. Furthermore, an additional value of this approach is that it contributes both to image level classification and bounding box detection based methods.

2. Related Work

Training image recognizers usually involves using large numbers of labeled samples. The major thrust of this paper deals with cases where we have a very limited number of samples; the so-called *few-shot learning* problem. In the image domain this topic breaks down into two related scenarios: *few-shot classification* and *few-shot object detection*.

In the few-shot classification scenario each image is an example of one of N classes, and the job is to recognize

which class each image belongs to. This area has been studied extensively, although the authors of [7] argue that implementation details have prevented a fair comparison between competing methods. Some of the approaches in this area use samples from abundant classes to first train a base recognizer, and then extend that recognizer using samples from the rare classes [8]. Another theme in this area involves using a cosine-similarity-based distance metric [8].

The tasks addressed in this paper require using few-shot object detection. In this scenario, each image may contain objects of many classes (or none) and the recognizer must localize each object in addition to identifying its class. This is a much more difficult problem, both because the system must produce bounding boxes as well as classes, and because the recognizer must reject background areas of the image which contain none of the classes under study.

In the few-shot object detection scenario the strategy of first training on abundant classes is also common [9] [10] [11] [12]. The cosine similarity distance metric [11] [7] is also used, as is a more general concept of distance metric learning [13] [12]. The notion is that placing objects and classes in an embedding space where members of a class are close to one another and far from other classes will lead to better generalization which can be exploited when extending the recognizer to the rare classes. Some of these methods use triplet loss [14] and related loss functions [13] [12] in the training process. In these systems, points in the embedding space (or metric space) become prototypes [15] which are either single class examples or the centroid of a class or subclass.

3. Creating Synthetic Imagery

Generating synthetic imagery requires the use of 3D models. We obtained a range of 3D models for 3 different categories of objects (at least 3 models each):

- Construction Equipment: Cement Mixer, Dump Truck, Excavator, Front Loader, Haul Truck, Scrapper Tractor
- Marine Vessels: Ferry, Fishing Vessel, Maritime Vessell, Tugboat
- Aircraft: Cargo Plane, Fixed-wing Aircraft, Helicopter, Small Aircraft

3D models were obtained from the TurboSquid.com commercial repository. Although free 3D models existed for both objects, they were generally of poorer visual quality than the commercial options.

After models were acquired, Blender3D was used to generate images from 3D models that contain a specified (1) viewing angle, (2) lighting condition, and (3) shadow. Figure 1 demonstrates examples with different object rotations and shadow positions. The black and white images display the masks that will actually be inserted into satellite imagery for training object recognition.

The next step is to merge one of these model images with a designated satellite chip (300x300 pixels). Currently, image synthesis takes as parameters: model name, background chip id, sun position (theta and phi), and model

Figure 1. Images created from 3D models with specified rotation and sun positioning

pose (x, y, theta), and scale. Note that we do not currently "recognize" proper lighting conditions for each satellite image, instead manually identifying the proper sun/shadow positioning for each image. This is not an unreasonable requirement, as sun position should be determinable from other, publicly available data sources, given a known satellite position and time, but is information that was not released with the xView dataset. Scale is also a known factor, as the distance to the ground is known, and the size of the object is also known. However, we do allow scale to fluctuate +/- 10% around the target. All remaining parameters are randomized.

The actual insertion process is illustrated in Figure 2. The first step is to use the mod-Poisson [16] function to blend the masked model together in the target position. We use an open-source implementation [17] for this purpose. Figure 3 shows examples of the result.

Next, the region surrounding the target can be extracted and passed to a trained CycleGAN for improvement. Only a 64x64 pixel region of the image is modified by the CycleGAN (Tugboats, Fishing Vessels, and Helicopters

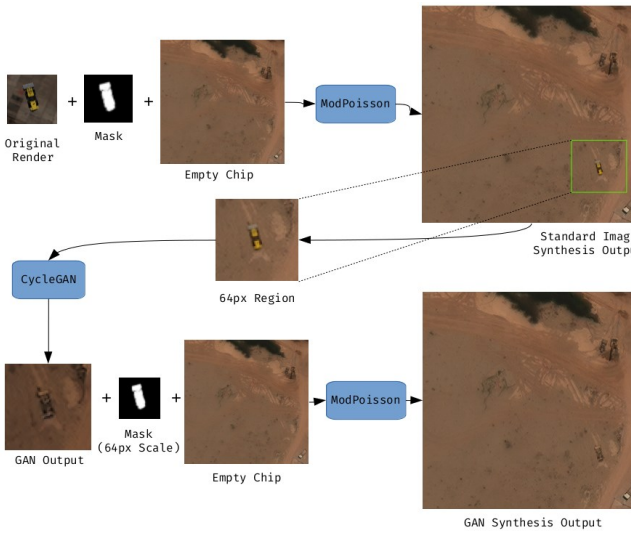

Figure 2. Inserting the 3D model into the selected background chip is a multi-step process.

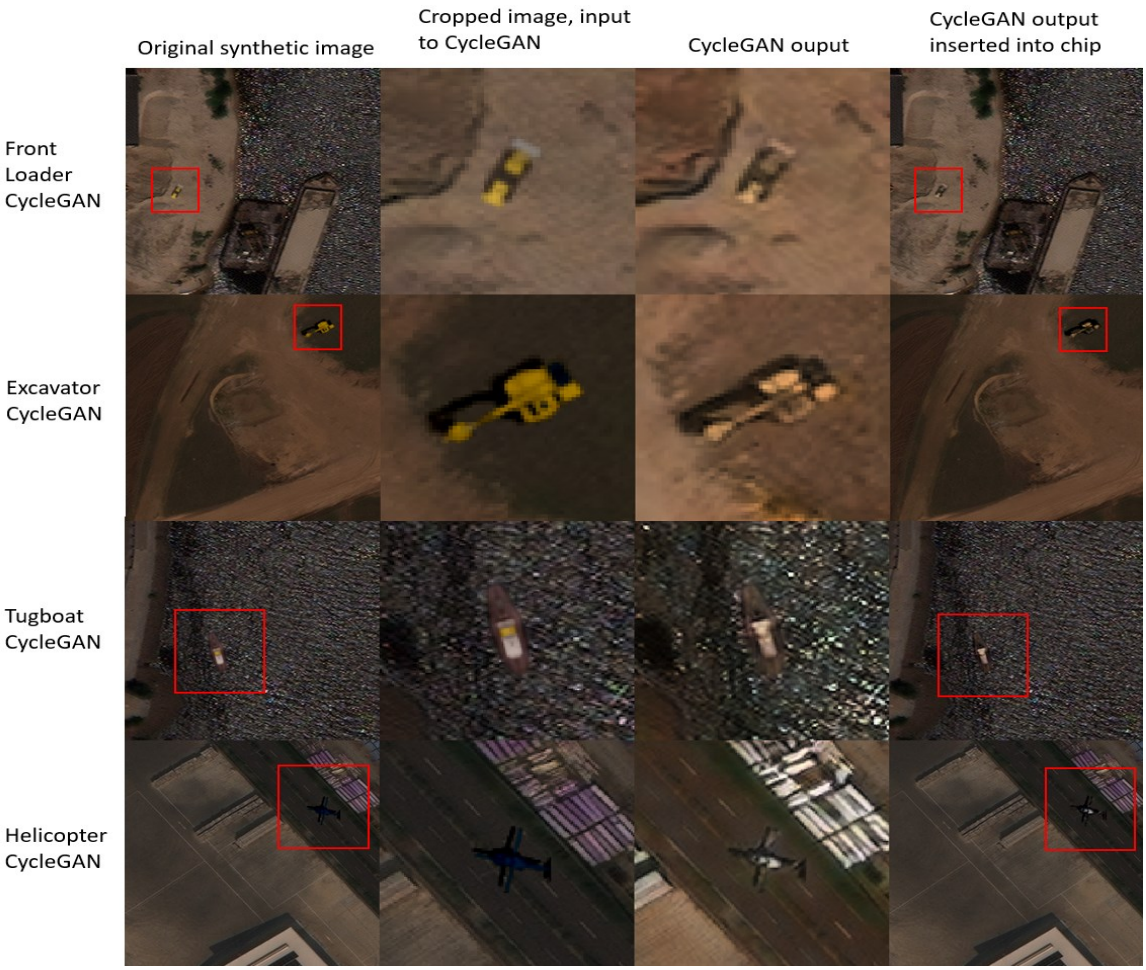

Figure 3. Stages of CycleGAN output for 4 different objects

use a 128x128 region due to object size). Note that only the RGB image is modified, with the original mask image preserved to maintain object contours when it is reinserted in the final step using mod-Poisson back into the image.

4. Training and Validation Process

In our satellite image processing application, we fully expect human data analysts to remain involved in the process of finding rare objects. The automation goal is not to remove people entirely, but make them more effective with support from computer vision in two ways: (1) reduce the volume of images requiring human attention (e.g. image filtering); and (2) provide visual cues on remaining images to speed up analysis (e.g. target cueing). To build a real system, this means we need to both train models for rare object detection and determine thresholds for classification of images and detection of objects within them. This process is described in greater detail in Figure 4. The remainder of this section describes the methods for training the GAN and RetinaNet layers as part of this process.

4.1. GAN Layer

We use a publicly available Cycle-GAN implementation [3] to support rare object detection. Cycle-GAN is an image-to-image translation method that is trained in an unsupervised fashion without paired examples. An example application of the Cycle-GAN method was to translate photographs of horses to zebras and vice versa by giving it sets of horse images and sets of zebra images and having it learn the modifications to make them more similar to each other. A similar process was applied to turn summer roads into winter [17].

In our application, the GAN will be taught to convert images between two domains: synthetic models, and real satellite images. Because we have very limited real data for the target class, we need to use related classes of vehicles to train the GAN. For each object of interest, we used 3-5 related "contrast classes" from the xView dataset.

- **Front Loader**: Cement Mixer, Dump Truck, Excavator, Haul Truck, Scraper Tractor
- **Excavator**: Cement Mixer, Dump Truck, Front Loader, Haul Truck, Scraper Tractor

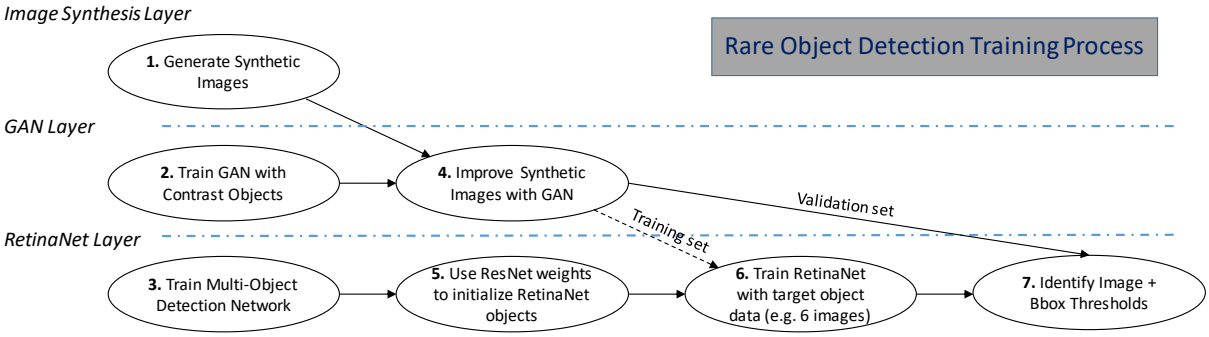

Figure 4. Training process illustrating creation and use of GAN imagery

- **Cement Mixer**: Dump Truck, Excavator, Front Loader, Haul Truck, Scraper Tractor
- **Haul Truck**: Cement Mixer, Dump Truck, Excavator, Front Loader, Scraper Tractor
- **Tugboat**: Ferry, Fishing Vessel, Maritime Vessel
- **Fishing Vessel**: Ferry, Maritime Vessel, Tugboat
- **Helicopter**: Cargo Plane, Fixed-wing Aircraft, Small Aircraft

Therefore, with Front-loaders, we used real imagery from other construction equipment and 3D models from the same contrast class set plus available Front-loader models. Figure 3 illustrates example results for four different target classes. All GAN models were trained with zero real examples of the target.

In our example, the GAN can be used to generate either training or verification data. When generating verification data, additional background images with neither synthetic nor real examples of the target class are added to the verification dataset to set detection thresholds.

4.2. RetinaNet Layer

In this work, we used RetinaNet [6] to both filter images unlikely to contain a rare object and detect rare objects in the remaining images. Note that because the purpose of this work was to supplement human analysis rather than replace it, we made the detection problem simpler than standard by increasing all bounding box sizes in the training/testing data to a minimum of 60-pixels. This adjustment allows an automated process to highlight areas most in need of human attention before moving on to the next image.

To train RetinaNet to detect the target object, we first trained a multi-class recognition network on 19 alternative classes from the xView dataset. Training was run for 100 epochs with Adam optimization. A held-out verification set identified the network with the greatest mean average precision, which was then used to initialize weights of the ResNet layers within a new, single class RetinaNet instance (RetinaNet extends the original ResNet classification network). This new network is then trained with all available rare object instances for a fixed number of epochs (20), again using an Adam optimizer. During this training step, randomly generated GAN imagery can be included, but as will be demonstrated in Section 5.1, this is not a significant performance booster in this domain.

Once the network is trained, we evaluate performance characteristics such as mean average precision and false alarm rates on the test set, but for a real deployment, we also need to specify a threshold at which to reject images and/or bounding boxes. There are actually two rejection thresholds of interest:

- **Image Filtering Threshold** – if all bounding boxes evaluated as part of this image have confidence scores below the specified image threshold, then the image is rejected and not shown to the data analyst.
- **Target Cueing Threshold** – an image threshold may be set low to achieve high recall, but a separate threshold can be used to reduce the number of cues shown to the data analyst.

The standard approach for estimating these thresholds is to use a holdout set of real images never included in training to construct precision/recall statistics for different threshold values. Armed with these statistics, the user can select the values most appropriate for the application. Unfortunately, with rare objects, we may have 10 images or less. Splitting these data into training and validation sets is impractical. As we will demonstrate in Section V, GAN images can be used in place of a validation set, leaving all real data available for training the RetinaNet model

5. Results

The xView dataset consists of high-resolution imagery (>HD) of variable sizes. The objects of interest (cement mixer, excavator, fishing vessel, front loader, haul truck, helicopter, tugboat) generally ranged in size from 20-100 pixels. With such small objects, high-res images were broken into smaller 300x300 chips for analysis. Labeled instances of each object were split into train and test using a 2:1 split, with additional rules: (1) in the event that there were more than 600 labeled instances, only 400 were set aside as dedicated training data, (2) all chips from the same high resolution image had to go into either test or train. The goal of both of these rules was to separate enough training data from which to sample rare objects while preserving a distinct, but large test set. Additional "empty" background images were added to each test set to create a 1:10 ratio of images with valid objects to images without valid objects.

As this was a rare object detection challenge, training data were re-sampled after the split to create 10 subsets each of 6 and 10 labels for use in training detection. Note that a training subset of 6 labels could have fewer than 6 images if there were more than 1 instance of the target class in an image (same for 10 images). We then used two different RetinaNet initialization procedures with these images:

- **Baseline** networks used the default training process in our implementation of RetinaNet [19]. ResNet models were initialized from pretrained weights available on torch model zoo, with all other layers randomly initialized. The entire network was then retrained for 20 epochs on either 6 or 10 images of the target class.
- **LargeNet** models were trained similarly to Baseline, except that a ResNet model trained on 19 other classes of vehicles in the xView dataset was used for initialization before retraining with the 6 or 10 real images.

For each of these network initialization methods, we also explored adding either an additional 30 GAN images or an additional 30 synthetic images without using the GAN, making a total of 36 and 40 training images respectively. We chose 30 images after evaluating different numbers. These results align with [18] [2] which show that training sets with around 75-90% synthetic imagery get optimal results.

To estimate image filtering and target cueing thresholds, we used separate validation sets comprised of 150 synthetic images (with or without GAN) and 1500 background images. This ratio of 1:10 images was the same ratio used to construct a test set. To calculate confidence thresholds, a minimum score threshold of 0.01 was investigated, and a successful detection was classified as an Intersection Over Union (IOU) score > 0.3.

5.1. Using GAN Images to Train Object Detection

We tested all 6 methods of training using the target objects' test set. Mean average precision (MAP) results averaged across the seven objects are found in Table 1. Per object results for 10 training images are in Figure 5.

Table 1. Mean average precision for each training method across all 7 objects.

Initialization	**Baseline**			**LargeNet**		
# Labels	Only Real	30 GAN	30 Synth	Only Real	30 GAN	30 Synth
6	0.15	0.19	0.21	0.21	0.22	**0.24**
10	0.19	0.21	0.25	0.28	0.25	**0.28**

As expected, the Baseline training method demonstrates significantly worse MAP with either 6 or 10 training images than the LargeNet initialized with a model trained on xView data. Furthermore, in general, although the GAN step generates more visually appealing training data, it actually decreases MAP compared to synthetic images created from models without a GAN.

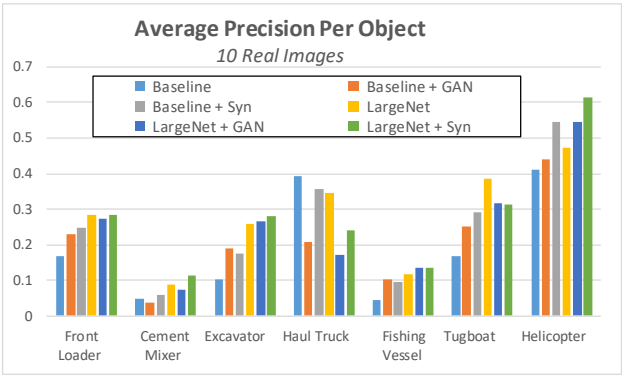

Figure 5. Average precision per object with 10 training images.

Specific objects, however, show some interesting trends. The Haul Truck, in particular, has an inverted trend line. Baseline with no additional training images outperforms all other methods, with added training data through either synthesis or initialization actually decreasing MAP. Other objects like Front Loader and Tugboat show LargeNet approach to be best, outperforming all added training images.

More generally, the impact of synthetic images with or without the GAN is greatest with only 6 real labeled examples. With 10 labels, Baseline initialization improves with added images, but LargeNet initialization either decreases or stays about the same.

5.2. Using Synthetic Images to Set Image Filtering Threshold

Improving MAP scores is only one potential use of synthetic imagery. Once the model has been created, we also need to establish the *confidence_score* threshold for detection. More specifically, we are building a system to help human analysts reduce the volume of data needing to be examined. Because we are searching for rare objects with models that demonstrate low average precision, we will focus first on reducing the set of images (not bounding boxes) to be displayed while still hitting a target recall (e.g. 95%). A synthetic validation set was used to identify the *confidence_score* that achieves this target recall. Synthetic imagery with and without the GAN step were evaluated.

The question when using an artificial validation set is how closely it tracks performance with real data. Starting with the LargeNet initialization and only 6 training labels, Figure 6 demonstrates that synthetic validation data (with and without GAN) track surprisingly close to real imagery across all objects. In general, the synthetic image data underestimate recall on the test set. But even though these data have little or no impact on training with LargeNet, they are still highly predictive of system recall.

Table 2 illustrates this relationship, detailing the image filtering recall and false alarm rate achieved on the test set when using a confidence threshold derived from a synthetic validation set and target recalls of either 80% or 95%.

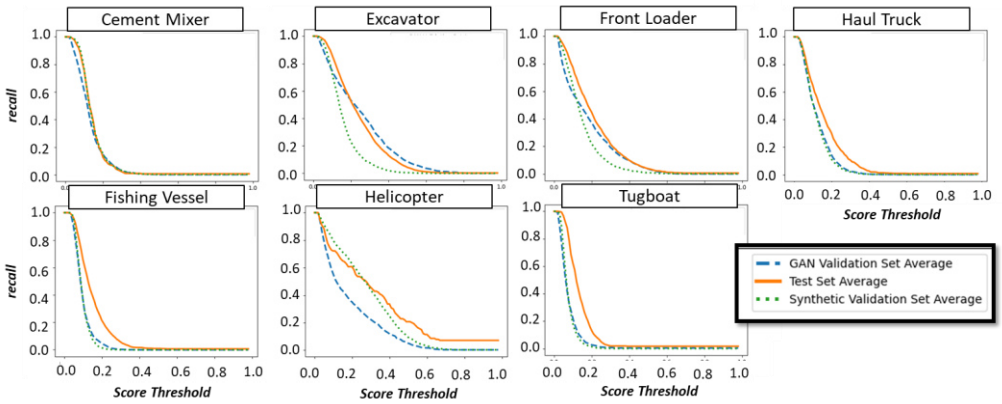

Figure 6. Recall vs score threshold for the LargeNet models with 6 real training labels. Both synthetic image validation sets generally underestimate recall on the test, but otherwise follow the same recall curve.

Table 2. Image filtering recall and (false alarm rate) achieved on a test set of real images using either synthetic or GAN validation sets against each of the 6 model types. Highlighted entries are within 5% of the target recall.

	Target Recall	# Real Labels	Baseline Initialization			LargeNet		
			None Added	30 GAN	30 Synth	None Added	30 GAN	30 Synth
Synthetic Images	0.8	6	**0.721** (0.255)	**0.407** (0.07)	**0.038** (0.001)	**0.838** (0.229)	**0.45** (0.091)	**0.07** (0.002)
	0.8	10	**0.654** (0.291)	**0.409** (0.067)	**0.049** (0.002)	**0.848** (0.21)	**0.484** (0.103)	**0.082** (0.002)
	0.95	6	**0.901** (0.477)	**0.683** (0.185)	**0.232** (0.024)	**0.944** (0.395)	**0.686** (0.222)	**0.347** (0.035)
	0.95	10	**0.795** (0.506)	**0.693** (0.191)	**0.294** (0.029)	**0.951** (0.368)	**0.723** (0.221)	**0.382** (0.032)
GAN Images	0.8	6	**0.836** (0.371)	**0.44** (0.078)	**0.752** (0.183)	**0.931** (0.326)	**0.531** (0.068)	**0.794** (0.171)
	0.8	10	**0.856** (0.351)	**0.484** (0.083)	**0.778** (0.182)	**0.93** (0.314)	**0.565** (0.067)	**0.814** (0.167)
	0.95	6	**0.954** (0.609)	**0.83** (0.293)	**0.94** (0.461)	**0.98** (0.601)	**0.91** (0.327)	**0.968** (0.512)
	0.95	10	**0.964** (0.591)	**0.864** (0.305)	**0.949** (0.459)	**0.985** (0.59)	**0.913** (0.326)	**0.976** (0.517)

Averaging across all 7 objects, there are two standout statistics. First, regardless of which synthetic image set used, most LargeNet models trained without any synthetic data averaged within 5% of the target recall. However, these models also demonstrated higher false alarm rates than other methods because they were trained with less data. The second standout is training with 30 synthetic images, but testing with a GAN validation set. This approach achieves a target recall within 5% of target on average for both initialization methods and both recalls evaluated. It also achieves the lowest false alarm rate, eliminating >80% of images with an 80% recall target, and ~50% of images with a 95% recall target. By contrast, using GAN images for both training and validation only reached the target recall only 25% of the time. And validating with synthetic after training with either GAN or synthetic demonstrated very large reductions in recall, reaching only 4% recall on the test set when trained and validated with synthetic (no GAN) images.

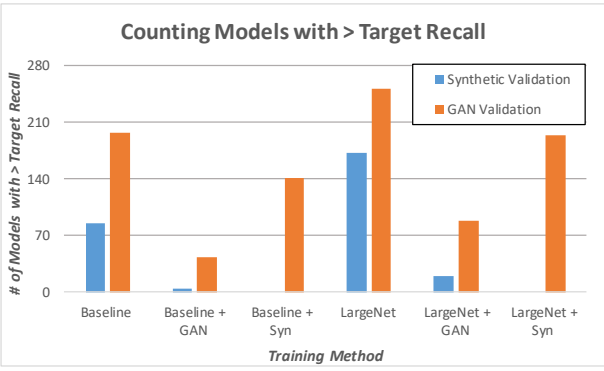

Figure 7. Across all objects, the number of objects achieving the target recall is significantly higher when using the GAN validation set across a number of model training methods.

This relationship between validation set type and training method is further illustrated in Figure 7. There were 280 models created per training method to generate the statistic in Table 2. If we count the number of models per method that crossed the target recall threshold, the GAN validation set clearly improves model reliability more than the Synthetic validation set across all 6 training methods

5.3. Target Cueing

Up to this point, we have demonstrated training and validation methods for reducing the number of images requiring human attention by 50% with only 6 real training examples. Our networks, however, are localizing objects in addition to classifying images. We want to use that information speed up analysis through target cueing – indicating regions of greatest interest first. But if we use the

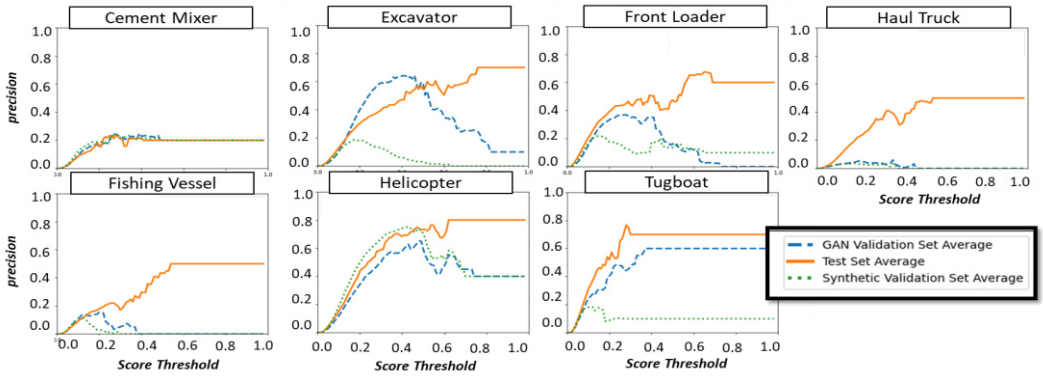

Figure 8. Precision vs score threshold data with the LargeNet training method and only 6 labeled examples.

same threshold as was used for image filtering, we demonstrate an average precision of <5%. With such a low precision, we are concerned that analysts will simply turn off target cueing, and, worse yet, also associate that performance with image filtering. Therefore, in this section, we evaluate our ability to set additional confidence thresholds to support target cueing, by achieving precision targets.

Figure 8 demonstrates the variability of the challenge with only 6 training examples. Precision was only predictable out to ~30%. In general, the GAN validation set was more predictive of precision (~33%) than without it (27%), where predictability was measured as being within 5%. Some objects were significantly less (Haul Truck-3%, Fishing Vessel-15%), while others were higher (Helicopter-60%, Tugboat-60%). We hypothesize that this is reflective of category variability. Haul Truck and Fishing Vessel seem to have many more variations of type, scale, and other distinctive features within the same category as opposed to helicopter or Tugboat.

Table 3 details per object target cueing test precision and recall for a targeted precision of 30% with 10 training labels. The variability is much greater than with image filtering, but the two training methods that are closest to achieving the target are LargeNet and LargeNet + 30 Synthetic.

Table 3. Target cueing precision and (recall) achieved on the test set with a targeted precision of 0.3 and 10 training labels. Yellow highlights indicate within 5% of target. Red highlights indicate higher precision.

	Model Training Method			
Target Object	**LargeNet**	**LargeNet + 30 GAN**	**Baseline + 30 Syn**	**LargeNet + 30 Syn**
Front Loader	**0.37** (0.34)	**0.27** (0.42)	**0.25** (0.42)	**0.26** (0.44)
Cement Mixer	**0.07** (0.38)	**0.11** (0.19)	**0.08** (0.14)	**0.13** (0.23)
Excavator	**0.26** (0.48)	**0.23** (0.54)	**0.18** (0.40)	**0.55** (0.22)
Haul Truck	**0.10** (0.81)	**0.19** (0.38)	**0.29** (0.53)	**0.23** (0.41)
Fishing Vessel	**0.07** (0.75)	**0.15** (0.42)	**0.20** (0.30)	**0.15** (0.43)
Tugboat	**0.62** (0.28)	**0.21** (0.51)	**0.20** (0.58)	**0.21** (0.61)
Helicopter	**0.42** (0.46)	**0.23** (0.65)	**0.21** (0.72)	**0.22** (0.74)
Average	0.27 (0.50)	0.2 (0.44)	0.19 (0.44)	0.25 (0.44)

6. Conclusion

In summary, we have evaluated the utility of synthetic imagery for supporting the recognition of rare objects in satellite images. We have initialized models with two different training sets (ImageNet and xView) and tested synthetic imagery with and without the use of a generative adversarial network (GAN). For training purposes, the proposed synthetic images created from 3D models without the use of a GAN demonstrated the greatest improvement on mean average precision. However, this improvement was limited when the model was initialized with highly related imagery (e.g. xView).

With rare objects, where labeled examples are at a premium, synthetic imagery can also be used as a validation set to identify a good confidence threshold, leaving all real data available for training the RetinaNet model. In this work, we have demonstrated that synthetic images generated with the use of a GAN are best for this purpose. With only 6 training images, a GAN validation set was used to select a threshold that averaged 95% recall, while still eliminating %50 of the background images. Precision targets are more difficult with these rare object models, but when aiming for 30% precision, we could still achieve an average of 25% on a test set with a 1:10 real vs background image ratio.

We find it very interesting that synthetic imagery without a GAN are better for training, but adding the GAN step create better validation imagery. This may suggest that our GAN step is introducing too much noise into the training process – we are not training our GAN on the target rare images because we have too few of them and instead using objects with similar properties (e.g. other construction equipment). However, while the GAN is not creating ideal training imagery, it is still modeling noise that the synthetic validation step otherwise lacks.

In the future, we intend to integrate these training and threshold estimation steps into an active learning cycle (Figure 9). With both a model and a threshold, we can create a deployable system for use by a human analyst. After they have identified some small number of new examples of the

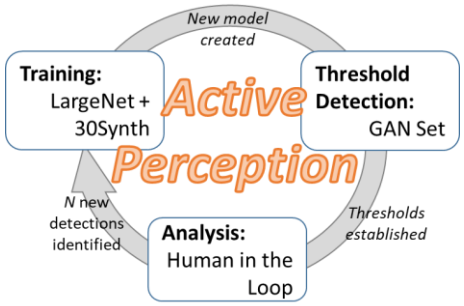

Figure 9. Proposed active perception training cycle uses the LargeNet + 30 Synth initial training, then applies the GAN validation set to determine thresholds before showing images to the analyst.

target using this system (as few as 1), models can re-trained, thresholds can be re-evaluated, and a new, improved system is ready for the analyst to use. In this way, we will create a virtuous update cycle for detecting rare objects

References

[1] K. Wang, D. Zhang, Y. Li, R. Zhang and L. Lin, "Cost-Effective Active Learning for Deep Image," *IEEE Trans on Circuits and Systems for Video Technology,* vol. 27, no. 12, pp. 2591 - 2600, 2017.

[2] G. Georgakis, A. Mousavian, A. Berg and J. Kosecka, "Synthesizing Training Data for Object Detection in Indoor Scenes," in *Robotics: Science and Systems Conference (RSS)*, Cambridge, MA, 2017.

[3] J.-Y. Zhu, T. Park, P. Isola and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," in *Int Conf on Computer Vision*, Venice, Italy, 2017.

[4] P. Isola, J.-Y. Zhu, T. Zhou and A. Efros, "Image-to-Image Translation with Conditional Adversarial Nets," in *Computer Vision & Pattern Recognition*, Berkeley, CA, 2017.

[5] D. Lam, R. Kuzma, K. McGee, S. Dooley, M. Laielli, M. Klaric, Y. Bulatov and B. McCord, "xView: Objects in Context in Overhead," February 2018. [Online]. Available: https://arxiv.org/pdf/1802.07856. [Accessed 12 August 2019].

[6] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection," in *Int Conf on Computer Vision (ICCV)*, Venice, Italy, 2017.

[7] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. Wang and J.-B. Huang, "A Closer Look at Few-Shot Classification," in *International Conference on Learning Representations*, New Orleans, LA, 2019.

[8] S. Gidaris and N. Komodakis, "Dynamic Few-Shot Visual Learning Without Forgetting," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, 2018.

[9] Z. Xu, L. Zhu and Y. Yang, "Few-Shot Object Recognition from Machine-Labeled Web Images," in *Computer Vision and Pattern Recognition*, Honolulu, Hawaii, 2017.

[10] F. E. Nowruzi, P. Kapoor, D. Kolhatkar, F. A. Hassanat and R. Laganiere, "How much real data do we actually need: Analyzing object detection performance using synthetic and real data," in *International Conference on Machine Learning (ICML 2019) Workshop on AI for Autonomous Driving*, Long Beach, CA, 2019.

[11] X. Wang, T. E. Huang, T. Darrell, J. E. Gonzalez and F. Yu, "Frustratingly Simple Few-Shot Object Detection," in *Int Conf on Machine Learning*, Vienna, Austria, 2020.

[12] L. Karlinsky, J. Shtok, S. Harary, E. Schwartz, A. Aides, R. Feris, R. Giryes and A. Bronstein, "RepMet: Representative-Based Metric Learning for Classification and Few-Shot Object Detection," in *Computer Vision and Pattern Recognition*, Long Beach, CA, 2019.

[13] Y. Yang, F. Wei, M. Shi and G. Li, "Restoring Negative Information in Few-Shot Object Detection," in *Conf on Neural Information Processing System*, Vancouver, Canada, 2020.

[14] V. Balntas, E. Riba, D. Ponsa and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks," in *British Machine Vision Conference*, York, UK, 2016.

[15] J. Snell, K. Swersky and R. Zemel, "Prototypical Networks for Few-shot Learning," in *Conference on Neural Information Processing Systems*, Long Beach, CA, 2017.

[16] P. P´erez, M. Gangnet and A. Blake, "Poisson Image Editing," in *Int Conf on Computer Graphics and Interactive Techniques*, 2003.

[17] W. Emmanual, "GitHub - Poisson-image-editing," [Online]. Available: https://github.com/willemmanuel/poisson-image-editing. [Accessed 2 February 2021].

[18] M.-Y. Liu, T. Breuel and J. Kautz, "Unsupervised Image-to-Image Translation Networks," in *Neural Information Processing Systems*, Long Beach, CA, 2017.

[19] Y. Henon, "GitHub - Pytorch Implementation of Retinanet Object Detection," [Online]. Available: https://github.com/yhenon/pytorch-retinanet. [Accessed 18 February 2021].

[20] E. Martinson, "Interactive Training of Object Detection without ImageNet," in *Int Conf on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 2018.