Improving Synthetic Image Generation for Better Object Detection

Denglin Jiang
Center for Data Science
New York University
dj1369@nyu.edu

Diwen Lu
Center for Data Science
New York University
dl3209@nyu.edu

Yuwei Wang
Center for Data Science
New York University
yw1854@nyu.edu

Abstract

In deep learning, the abundance of data is the pivotal factor in whether the model will achieve desirable performance. Yet in many scenarios, labeled data is a luxury either due to the scarcity of data itself or the huge cost of labeling. In the field of object detection, today's advanced game physics engine is powerful enough to generate vivid images that closely resemble the real ones. Physically rendered synthetic images has a promising future as a data augmentation method. However, the question of what features should be generated within the synthetic images still remain challenging as currently people usually rely on simple analysis of the data set or just the gut feeling. It is thus very attempting to establish a close feedback loop to improve synthetic data generation, so that synthetic data can be generated in a systematic self-guided way. In this project, we developed 2 visualization methods to understand what drives the wrong detection and potentially improve synthetic image generation in order to get better object detection results. We adopted latent variable models to compare real and synthetic data in latent space as well as advanced convolution deep learning visualization tools in an attempt to understand model detection.

1 Introduction

Object detection as a technology related to computer vision and image processing has made great contribution in various application areas. In this project, we aim to design and conduct a series of experiments to apply existing state of the art machine learning and deep learning frameworks, to explore approaches to use synthetically generated data for object detection. We utilized an existing real/synthetic combined data set of satellite images to detect different types of aircrafts. The goal is to construct a structured approach for generating better synthetic images as complement to real data, and ultimately obtain better accuracy for deep learning based object detection.

The approaches for improving the generation process of synthetic data can be roughly classified into two major categories: supervised approach, which analyzes the prediction output of object detection with real data and explores the lacking features of model learning; and unsupervised approach, which conducts analysis on input images directly and conjecture the patterns needed for synthetic images based on analysis results. Most progress in this project lies into the unsupervised category. We developed 2 new methodologies to understand model detection. First, we use latent variable models to compare real and synthetic data in latent space. Second, we apply advanced convolution deep learning visualization tools in an attempt to understand model performance. We will discuss in details these methodologies in Section 3.2.

Preprint. Under review.

2 Background

2.1 R-CNN Family for Object Detection

The term "object detection" and "object recognition" are often interchangeably used by practitioners, and can be referred to different classes of tasks. One of the most clear definitions given by the paper (1) distinguishes object detection as *Locate the presence of objects with a bounding box and types or classes of the located objects in an image*, and addresses the difference from other computer vision tasks such as image classification and object localization.

The most prevalent models used for object detection and localization are R-CNN family, which stands for "Regions with CNN Features" or "Region-Based Convolutional Neural Network. The family includes R-CNN, Fast R-CNN, and Faster-RCNN, in the order of development series. R-CNN was firstly introduced in (2), R-CNN model is composed of three modules - Rigional Proposal, which generate and extract category independent region proposals (e.g. bounding boxes); Feature Extraction, which extract feature from each candidate region; and Classifier, which assigns features as one of the known class. It works very effectively in most object detection tasks, however, there also exists some obvious limitations. Most drawbacks of R-CNN are due to the multi-stage structure of training, which causes great time complexity and expensive space. Thus, Fast R-CNN and Faster R-CNN are introduced to solve this issue.

Fast R-CNN was built based on prior work proposed to speed up the technique. The method is called spatial pyramid pooling networks, or SPPnets, in (3). It takes an image as input as well as a set of object proposals, then processes the image with convolutional and max-pooling layers to produce a convolutional feature map. A fixed-layer feature vector is then extracted from each feature map by a region of interest pooling layer for each region proposal.

Faster R-CNN developed by Microsoft Research in the 2016 paper (4), adds a module of Region Proposal Network, or RPN, on top of Fast R-CNN. RPN work as an attention mechanism and improves efficiency by telling the second layer framework where to pay attention. Closely related, Mask R-CNN adds functions which support image segmentation, in the 2017 paper (5).

2.2 Latent Variable Model

Latent factors (or latent variables in statistics) are defined as variables that are not directly observed, but can be derived from existing variables. In machine learning or deep learning settings, latent factors are often generated when input dimension space is high and contains low information gain on average. Latent variables can effectively improve model performance, especially in training efficiency, as it provides much cleaner input information with maximized information gains reserved.

Both PCA (Principal Component Analysis) and NMF (Non-negative Matrix Factorization) are widely adopted methods for dimension reduction. Principal Components are created by projecting data points onto the direction vector on which the matrix variance is maximized. Numbers of principal components are determined both from the nature of data sets and the needs of model feeding. NMF takes non-negative matrix and factorize it into two non-negative matrices, one of which would be of same number of rows but fewer columns, and can be taken as dimension reduced input. In Computer Vision, we process the images as pixels and then represent data points in each image as large matrices. Then we can directly conduct PCA and NMF on the matrices to get latent factors directly.

Auto-encoder and especially VAE (Variational Auto Encoder) are particularly useful in computer vision. Auto-encoder takes input images as vectors, and through different layers of deep learning, it produces a "bottleneck" of neural network structure, which consists of significantly smaller number of vectors for further training. In order to get an output image, a decoder system would be added symmetrically after the bottleneck part is done, to reverse the dimension and complexity of core model output and makes it into a new image. Variational Auto Encoder works in very similar way, but instead of producing determined latent vectors, it generates a distribution of latent vectors and then sample from there. Moreoever, it is often useful to adopt a disentangled representation, which forces variation auto encoder to use selected latent factors for training models.

2.3 Interpret Deep Convolutional Networks

In this subsection we introduce various localization algorithms which work as the beginning stages of R-CNN models we adopted. We experimented on Spatial Attention Map (SAP), Class Activation Map (CAM) and Deep Taylor Decomposition (DTD).

The essence of the spatial attention is to learn a weight map which represents the relative importance of activations within the same channel, by utilizing the inner spatial relationship of features. To observe the spatial attention, we start by applying average-pooling and max-pooling operations along the channel axis and concatenate them to generate an efficient feature descriptor. Then on the concatenated feature descriptor, we apply a convolution layer to generate a spatial attention map which encodes for the model to decide where to look close.

Deep Taylor Decomposition (DTD) is a powerful method to explain individual neural network predictions in terms of input variables, first introduced in (8). It is widely used to interpret deep learning models in computer vision, especially in image classification. DTD determines which input variables (pixels) have contributed most relevantly to the prediction outcome. The interpretation can be visualized as a heatmap to emphasize the most contributing pixels, as explanation of the classification decision.

3 Problem Definition and Methodologies

3.1 Problem Definition

Our goal is to improve synthetic data generation in order to improve object detection performance. When generating a synthetic image, this physics engine based ANA synthetic data generation platform gives us a variety of options of features to tweak (number of planes, color of planes, adding cloud ... check the website!) A brute-force method to improve synthetic data generation is to 1) randomly select a bunch of features to tweak; 2) use these newly generated images to train a pre-trained Faster-RCNN; and 3) see if there are any improvements in validation MaP score after adding new synthetic data. However, there are over hundreds of options of features to tweak and a combination of these options exponentially increases the option space, which makes this brute-force trial-and-error method impossible to implement. Therefore, we need to design new methodologies to systematically pick out features worth tweaking.

3.2 Methodologies

3.2.1 Latent Variable Model

A natural direction is to find a way to differentiate synthetic images from real images. As synthetic images generated from ANA platform resemble real data a lot in pixel intensity, image texture, sharpness etc, it is very hard to come out with really concrete comparison metrics in original data space. Therefore, we use latent variable models on synthetic/real data in the hope of finding discrepancies in the latent space.

There are various latent variable models. We start easy with a PCA model, and then we switch to NMF model for its property in dealing with local features, and in future we will experiment with Convolutional Variational Auto-encoderv (VAE) like ResNet VAE.

Steps are as follows. First, we train a latent variable model by real data. Second, we map both synthetic images and real images into the latent space. Then, we cluster all our synthetic and real images in the latent space in order to find important latent variable. Third, we try to visualize the differentiating latent variables hoping this could be used to guide future synthetic data generation. 1

A motivating example
A famous example of NMF is to decompose faces results in a couple of interesting component features like eyes, brows, noses etc 3. In the case of aircraft detection, an ideal latent variable should be able to be easily interpreted as a single component of the aircraft, for example, the wing. Then, by projecting synthetic data onto real hidden space, we could possibly find latent variables in which synthetic data and real data differ a lot. This way, by adding synthetic images with those features (represented by the hugely different hidden variables) into the training sets, we could possibly improve model performance.

The choices of latent variable model and hidden dimension are of vital importance to the success of our project. We start simple with Principal Component Analysis (PCA), whose constraints differ from Non-Matrix

Algorithm 1: Latent variable algorithm for finding the difference between synthetic and real data

Input : A collection of real images and synthetic images, COCO annotation, Parameters (hidden dimensions h)

1 Crop out the bounding boxes in each image using COCO annotation;

2 Center, Standardize, and Reshape all image vectors into the same square size using any upsampling function, to get $D_{real}, D_{syn} \in \mathbb{R}^{m \times n}$ where m is the number of images in the collection, and n is the flattened image size;

3 Project D_{syn} onto $\mathbb{H}$;

4 Cluster real and synthetic image data points in $\mathbb{H}$ to recognize patterns;

Output : Hidden vectors

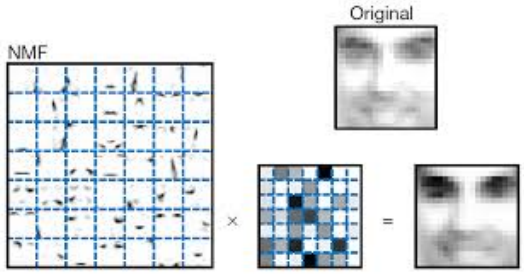


Figure 1: A famous NMF example (11) to extract local features in face detection, where hidden variables are visualized into local features like eyes, noses etc.

Factorization (NMF) in that PCA adopts an orthonormal constraint in matrix factorization where NMF holds a non-negative constraint, which makes the latter easier to be interpreted and extract local features.

3.2.2 Explain Faster-RCNN

Another direction is to explain the results of Faster-RCNN, especially how it makes a false detection and what causes the mistake. Then we could try to improve on those mistakes with the help of synthetic data. Below is the methodology we designed to guide synthetic data generation from a model interpretation perspective.

Algorithm 2: Analying False Detections of Faster-RCNN

Input : A pretrained Faster-RCNN (trained on real data), a false instance

1 Implement localization algorithms from the pretrained Faster-RCNN:
1) Spatial Attention Map
2) Deep Taylor Decomposition
3) Class Activation Map (future work)

Output : A saliency map from the input false instance, overlay it on the original image if possible

Spatial Attention Map

This algorithm takes as input 1 false instance, and outputs 5 successive spatial attention maps, where each map captures the lower-level to high-level features that the Faster-RCNN pays attention to. This attention map is useful in improving synthetic image generation because it reveals what confuses the model, which gives us a direction to tweak our synthetic images.

Deep Taylor Decomposition

Spatial Attention Map reveals more broadly which high-level features attracts Faster-RCNN's attention, while DTD gives a more detailed explanation of specific contributing pixels. DTD, not like the gradient-based visualization technique, back-propagates the activation score layer by layer following the constraints described in (8), where the calculated scores on input pixels can be understood as the sheer contribution of that pixel in the final classification confidence. By comparing the values of different pixels on DTD map we are able to know what pixels (features) that make the model decide its corresponding class. In practice, we loaded the pretrained Faster-RCNN on civil role data set from (7) and visualized DTD map with regards to all classes:

Algorithm 3: Obtaining Spatial Attention Map

Input : a pretrained Faster-RCNN $model$, a false predicted image x, parameter: $p = 2$

for $i \in [2, 6]$ **do**

  Feed x into the i-th Feature Pyramid layer of the Faster-RCNN to get feature map z: $z_i = \text{FPN}_i(x)$

  Take the power of p for each pixel of the feature map z_i: $a_i = z_i^p$

  overlay a_i on the original image x

end

Output : 5 spatial attention maps

Algorithm 4: Obtaining DTD Map

Input : A pretrained Faster-RCNN, all detected instances x, class type c

Load the weights of the backbone of the Faster-RCNN into a ResNet50;

Add a fully connected linear layer and a softmax at the end of the ResNet50

Freeze the backbone layers, and train only the classification layer using labeled predictions from Faster-RCNN

Choose the class with which we want to do DTD and back-propagate the final activation score throughout the ResNet50 to input layer

Output : DTD map

civil_small, civil_medium, and civil_large. We borrowed implementation of DTD on ResNet50 from (10) and overrode their DTD class to enable DTD on a specific class.

A motivating example

On our dataset Faster-RCNN makes multiple types of mistake. The most common mistake is to confuse civil medium planes as civil large planes (Figure 2(f)), then civil small planes as civil medium (Figure 2(e)), and finally sub-parts of airplanes as whole airplanes (Figure 2(d)). And there are even cases, though rarely, that a non-airplane is detected as an airplane. For example, the model mistakes an airplane stair into a small civil aircraft in Figure 2(e). We hope to explain which features, more specifically, which pixels, cause the mistake. If later we observe from our DTD saliency map that the head of the stairs is responsible for the false prediction, we could add more labeled stair heads into the synthetic data.

You may find this idea similar to GAN, where the discriminator supervises the predictions from the generator. In our case, you can think of the physical-based ANA data platform as the generator, and we people as the discriminator. It is true that this method involves human labor, but it is far better than a brute-force trial-and-error approach to select features and contexts in synthetic data generation.

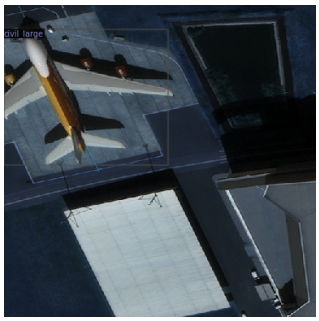
4 Experimental Evaluation

4.1 Data Description

We utilized a dataset named RarePlanes developed by CosmiqWorks. RarePlanes is a unique open-source dataset that incorporates both real and synthetically generated satellite imagery from Maxar and AI.Reverie. Compared with other synthetic/real combination datasets, RarePlanes is the first dataset built to test the value of synthetic data from an overhead perspective.

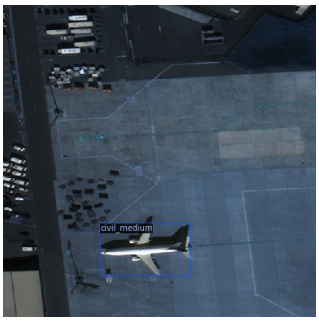
The real portion of the dataset consists of 253 Maxar WorldView-3 satellite images spanning 112 locations with 14,700 hand annotated aircraft. The accompanying synthetic dataset is generated via AI.Reverie's simulation platform and features 50,000 synthetic satellite images with over 600,000 aircraft annotations. Both the real and synthetically generated aircraft feature 10 fine grain attributes including: aircraft length, wingspan, wing-shape, wing-position, FAA wingspan class, propulsion, number of engines, number of vertical-stabilizers, if it has canards, and aircraft role. The below tree structure map gives an overview of all possible categories.

This data adopts COCO format and covers 3 more-and-more refined object detection tasks: 1) detect whether an aircraft exists; 2) detect aircraft role; and 3) detecting aircraft type. We only experimented task 1) and 2). Also, we did not cover all 8 classes of aircraft role, but only 3 classes of civil aircraft roles (small, medium, large) instead (the only public data available). We used only the real test set from RarePlanes in experiments.



(a) 69_104005000CDBE200_tile_1097_test

(b) 124_104001001F2A5700_tile_126_test



(c) 69_104005000CDBE200_tile_1534_test

(d) 69_104005000CDBE200_tile_1097_inf

(e) 124_104001001F2A5700_tile_126_inf

(f) 69_104005000CDBE200_tile_1534_inf

Figure 2: Wrongly classified instances examples

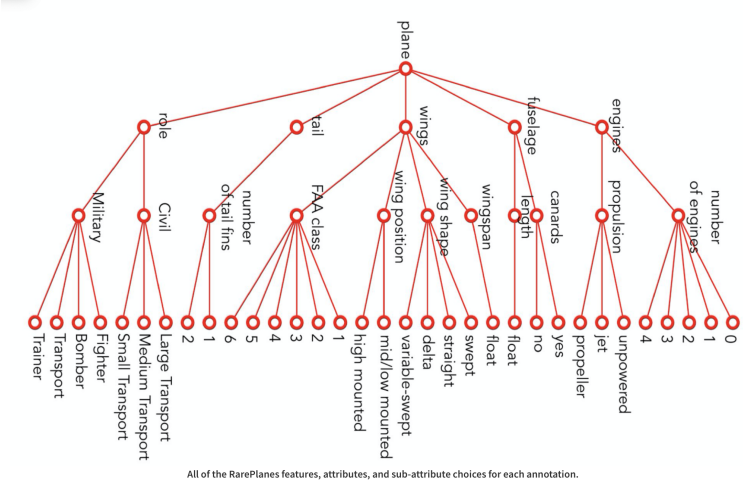


Figure 3: Catogery Hierachy of aircrafts (9)

4.2 Evaluation

The best way to evaluate our methodologies of improving synthetic data generation is to finish a closed-up loop to justify whether newly added features indeed improve object detection MAP score, where MAP(Mean Average Precision) is the evaluation criteria for object detection. However, due to time and resources limits, we did not proceed to the final step. Instead, we adopted human evaluation.

For latent variable models, we visualized every hidden dimension of the real hidden space as well as hidden variables projected from synthetic data, and vice versa. We also drew 2D scatter plots of real and synthetic image data points of any two hidden dimensions: if there's an overlap of data points, AND those hidden dimensions could be easily interpreted without further adjustments, then we might conclude that both real and

synthetic data share similarities in the latent space. In other words, real and synthetic data share features whose representations are those hidden variables. Therefore, we should focus on the latent space which could be well-interpreted AND in which there's a clear boundary between synthetic and real data. In this case, during synthetic data generation, we will particularly examine features which could be represented by these hidden variables. However, if all latent variables are inexplicable, we can hardly carry out any convincing conclusions.

For localization models, our team members act like discriminators to compare the input saliency map with the output false prediction. Two things are required to further improve synthetic data generation: 1) correctly identifying the type of mistake Faster RCNN makes; 2) identifying the contributing features/pixels in the saliency map. However, due to large volumes of data and limited time and resources, we haven't collected all contributing features for all categories of mistakes the FasterRCNN has made. Crowdsourcing might be a better solution to this. Once this step is done, we could start to generate synthetic data, and close the feed-back loop.

4.3 Results and Explanation

4.3.1 Latent Variable Model Results

PCA results
We visualized each latent variable of PCA trained on real data, given number of latent variables $= 9$. However, the results could hardly be interpreted into any aircraft components, which means PCA fails its goal.

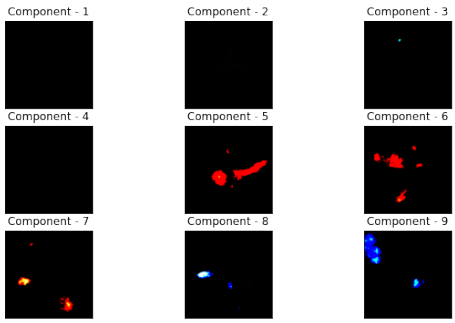


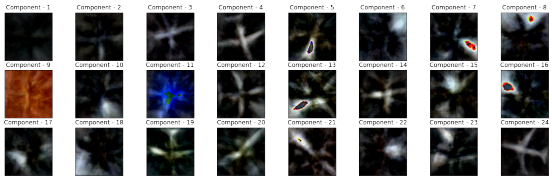
Figure 4: Visualization: PCA Latent Variables

NMF results
We also visualized the latent variables of NMF model trained on real data as well as on synthetic data (see 5a and 5c). The projected results (the synthetic onto real hidden space and the real onto synthetic hidden space) suggest that both real and synthetic data could not be explained using each other's latent variables. Also, the scatter plot, which shows the distribution of real and synthetic data distribution, reveals no overlap at all.

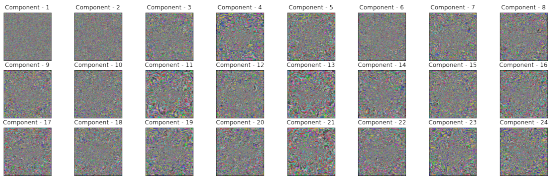
4.3.2 Localization Algorithm Results

Spatial Attention Map
Each row in Figure 6 shows 5 successive output spatial attention maps overlaid on a false prediction. Each of these map is a spatial attention map for a layer i in the Feature Pyramid Network (FPN) of the Faster-RCNN. Given an appropriate FPN layer, say $i = 2$ in Figure 6(b) and 6(g), the spatial attention map reveals the focus of the Faster-RCNN. In Figure 6(b), we see that Faster-RCNN is paying attention to a runway. In Figure 6(g), we see that Faster-RCNN is paying attention to the road as well as vehicles on the road. This suggest context features like runways, roads and vehicles are worth adding into the synthetic images in the ANA physical-based rendering system.

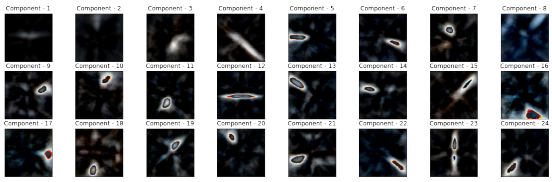
DTD Map
Our results for DTD on 5 types of detection are shown in Figure 7, 8, 9, 10, and 11. We also have one DTD on pretrained ImageNet just for reference.



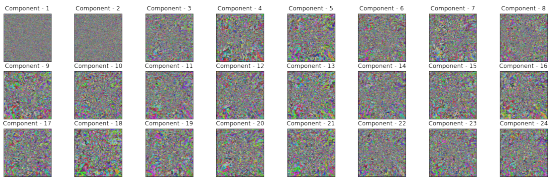
(a) Latent Variables from NMF trained by real data



(b) Synthetic data projected to Real latent space

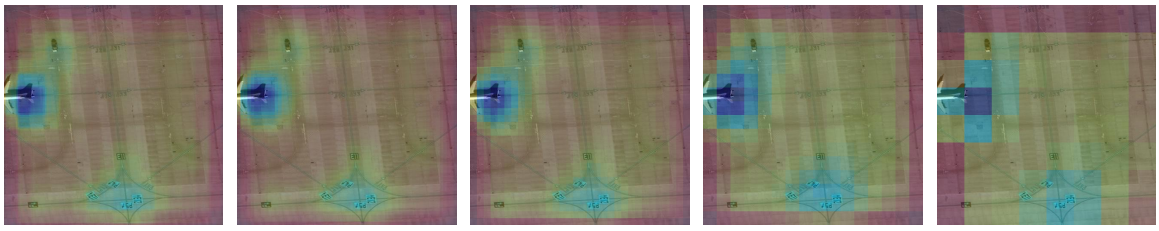


(c) Latent Variables from NMF trained by real data

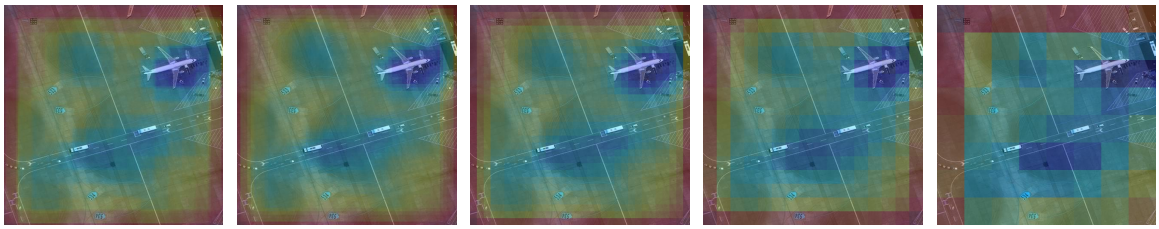


(d) Real data projected to synthetic latent space

Figure 5: Visualization: NMF Latent Variables



(a) FPN layer 2 (b) FPN layer 3 (c) FPN layer 4 (d) FPN layer 5 (e) FPN layer 6



(f) FPN layer 2 (g) FPN layer 3 (h) FPN layer 4 (i) FPN layer 5 (j) FPN layer 6

Figure 6: Examples of successive output Spatial Attention Maps overlaid on original images: transition from Feature Pyramid Network (FPN) layer 2 to 6 where feature level goes from low to high

4.4 Discussion

Latent Variable Models
Both PCA and NMF fail in yielding easily interpretable latent variables, even after tuning hidden_dim. They both fail the goal of comparing synthetic data and real data, possibly because both PCA and NMF are somewhat too naive. If the latent space is better constructed, say more complex, we might be able to interpret the latent variables.

In future, we plan to experiment with a 4-layer Convolutional VAE, hoping it could extract better local features than NMF. The optimal solution would be using the backbone of the pretrained Faster-RCNN as the encoder, which is equivalent to a ResNet50 architecture.

Localization Algorithms
SAP (Spatial Attention Map) succeeds in capturing higher-level contributing factors of Faster-RCNN. Compared with DTD, Spatial Attention Map is easy and fast to implement and requires no further tuning. However, the contributing factors are not as accurate and detailed as DTD. It is only capable of explaining easier task like detecting aircraft existence, but not more refined tasks like aircraft role or type detection. In future, we

shall combine the idea of Spatial Attention Map and Class Actication Map (12) to only focus on the objects that could be contribute to a particular class.

DTD, whose results are illustrated in Figure 7 to 10, is able to give a very clear-cut of silhouette of airplanes and intuitive highlighting of what pixel contribute how much to instance classification confidence. We try to combine the idea of Class Activation Map with DTD: using activation scores of 3 classes to do back-propagation. However, across 3 civil roles classes, the difference among DTD maps are negligible. In other words, DTD is not sensitive enough to differentiate classes in complex network, even though the weights in final layer differ much, all previous layers (159 layers in ResNet 50) act as regularization that offsets the difference manifested in the last layer. Our 3 civil roles class are still very coarse as airplanes differ in size, make, utility, wing-shape, number of engines etc. DTD should have a better differentiating power had applied on finer classes.

Additionally, all of our analysis and visualization are based on first cropping the instance out and treating as an image-classification problem. This approach is rather easy to follow as there are available implementation of deep taylor decomposition on VGG, ResNet, DenseNet, etc, but not on more complicated object detection framework such as Faster-RCNN and Mask-RCNN. The shortage of our DTD method is: 1) object detection is more than image classification in that it needs to propose a region that possibly contains an instance (creating a bounding box) before image classification on that region. But proposing the region requires background information, which we have no way to track in the current approach as we are only looking locally within the bounding box; 2) we must fine-tune the image classification model as not all layers are coming from Faster-RCNN, which could possibly change the actual predicted class from Faster-RCNN. For example, the prediction accuracy using Faster-RCNN predicted instances to train ResNet50 is 75% after our fine-tuning, which in the ideal case should be 100%.

Our exploratory work on explaining Faster-RCNN is a proof-of-concept that DTD can be a promising direction to follow. There are implementations of DTD on individual layers of CNN, and we consider it possible to extend those implementation to regional proposal network in Faster-RCNN so DTD can be performed on the whole image. But such engineering practice is beyond our capstone scope due to its complexity, given the limited time and resources we have.

5 Conclusions

In this project, we developed two new methodologies to improve synthetic image generation in order to get better object detection results. We adopted latent variable models to compare real and synthetic data in latent space as well as advanced convolution deep learning visualization tools in an attempt to understand model performance. Though our visualization is still in primitive stage, we successfully implemented proof-of-concept of local algorithms (Spatial Attention Map/Deep Taylor Decomposition) on satellite images, which is a promising direction to go along for future work. To have a holistic qualitative and quantitative evaluation of DTD, there are still much work in labeling, coming up with metrics, etc, which lie out of our project scope, but would be interesting to explore.

6 Lessons learned

In our initial attempt to explain Faster-RCNN, we thought there should already be some third-party package available there so we spent much time in searching for one but eventually our search was futile. Then we realize why can't we decompose that complex structures into smaller and simpler ones as there are existing DTD packages implementation on them. We trade quality of explanation for implementation simplicity. And our progress benefited much for this idea. This approach should be one of our first ideas in later project development.

References

[1] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, 2014; arXiv:1409.0575.

[2] Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2013; arXiv:1311.2524.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, 2014; arXiv:1406.4729. DOI: 10.1007/978-3-319-10578-9_23.

[4] Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, 2015; arXiv:1506.01497.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár and Ross Girshick. Mask R-CNN, 2017; arXiv:1703.06870.

[6] Grégoire Montavon, Sebastian Bach, Alexander Binder, Wojciech Samek and Klaus-Robert Müller. Explaining NonLinear Classification Decisions with Deep Taylor Decomposition, 2015; arXiv:1512.02479. DOI: 10.1016/j.patcog.2016.11.008.

[7] Jacob Shermeyer, Thomas Hossler, Adam Van Etten, Daniel Hogan, Ryan Lewis and Daeil Kim. RarePlanes: Synthetic Data Takes Flight, 2020; arXiv:2006.02963.

[8] Grégoire Montavon, Sebastian Bach, Alexander Binder, Wojciech Samek and Klaus-Robert Müller. Explaining NonLinear Classification Decisions with Deep Taylor Decomposition, 2015; arXiv:1512.02479. DOI: 10.1016/j.patcog.2016.11.008.

[9] Thomas Hossler, Jake Shermeyer, RarePlanes, (2020), GitHub repository, `https://github.com/aireveries/RarePlanes`

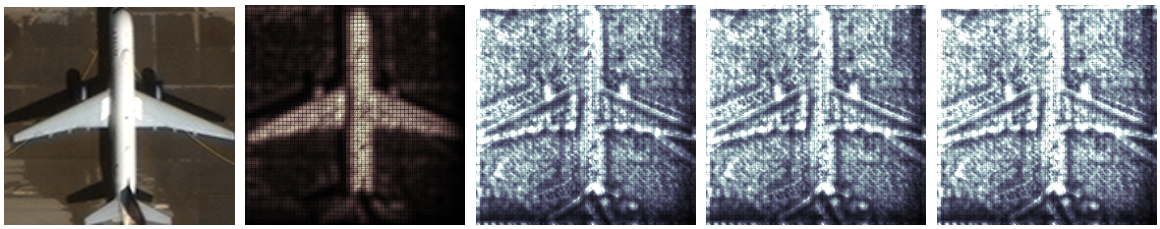
[10] ycmun, Deep-Taylor-Decomposition, (2019), GitHub repository, `https://github.com/myc159/Deep-Taylor-Decomposition`

[11] Lee, Daniel D and Seung, H Sebastian Learning the parts of objects by non-negative matrix factorization, 1999; Nature, vol401

[12] Kwaśniewska A, Rumiński J, Rad P. Deep features class activation map for thermal face detection and tracking[C]//2017 10th International Conference on Human System Interactions (HSI). IEEE, 2017: 41-47.

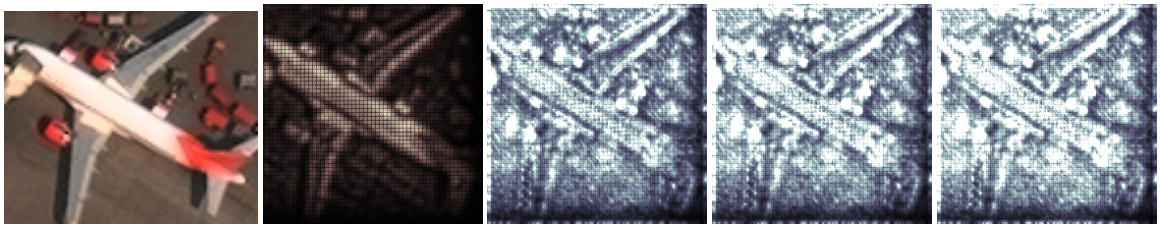
Student Contributions

Denglin Jiang: Latent Variable Models, Spatial Attention Map, Faster-RCNN inference, writing report
Diwen Lu: Deep Taylor Decomposition, Faster-RCNN inference, writing report
Yuwei Wang: Object Detection model training, Faster-RCNN inference, literature review, writing report



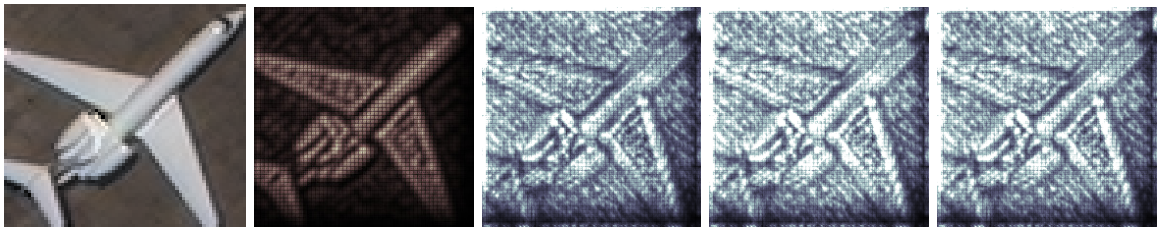
(a) original (b) ImageNet (c) small (40.91) (d) med (43.01) (e) large (46.67)

Figure 7: a large detected as a large (77_104001004299B200_tile_1004)



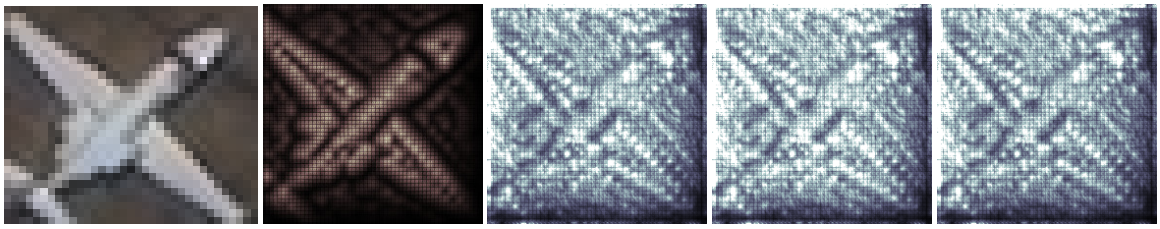
(a) original (b) ImageNet (c) small (41.64) (d) med (46.15) (e) large (50.16)

Figure 8: a medium detected as a large (31_10400100443CFD00_tile_814)



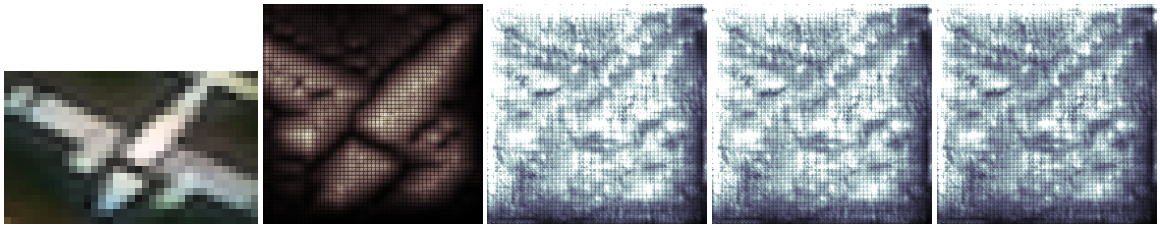
(a) original (b) ImageNet (c) small (38.52) (d) med (44.23) (e) large (42.22)

Figure 9: a medium detected as a medium (106_104001003D8DB300_tile_87)



(a) original (b) ImageNet (c) small (39.16) (d) med (43.00) (e) large (39.62)

Figure 10: a small detected as a medium (106_104001003D8DB300_tile_87)



(a) original (b) ImageNet (c) small (44.86) (d) med (43.25) (e) large (38.32)

Figure 11: a small detected as a small (81_1040010032B86F00_tile_373)