TRAINING SET EFFECT ON SUPER RESOLUTION FOR AUTOMATED TARGET RECOGNITION

Matthew Ciolino, David Noever, Josh Kalin

PeopleTec, Inc. – Huntsville, AL 35805 – 256-319-3800

February 11, 2020

ABSTRACT

Single Image Super Resolution (SISR) is the process of mapping a low-resolution image to a high resolution image. This inherently has applications in remote sensing as a way to increase the spatial resolution in satellite imagery. This suggests a possible improvement to automated target recognition in image classification and object detection. We explore the effect that different training sets have on SISR with the network, Super Resolution Generative Adversarial Network (SRGAN). We train 5 SRGANs on different land-use classes (e.g. agriculture, cities, ports) and test them on the same unseen dataset. We attempt to find the qualitative and quantitative differences in SISR, binary classification, and object detection performance. We find that curated training sets that contain objects in the test ontology perform better on both computer vision tasks while having a complex distribution of images allows object detection models to perform better. However, Super Resolution (SR) might not be beneficial to certain problems and will see a diminishing amount of returns for datasets that are closer to being solved.

Keywords super-resolution · deep learning · satellite imagery · image classification · object detection

1 Introduction

Single Image Super Resolution is the process of taking a low resolution (LR) image and running it through a model to increase the resolution with higher fidelity information than any scaling algorithm (Fig 1). This process currently does and has the potential to remove the need for increasingly large and expensive satellite cameras as running SISR could effectively increase the spatial resolution of your images. Since there are a multitude of ways to increase the resolution of an image, this is an ill-posed problem with many possible solutions. While significant work has been done on non-satellite images for SISR, not a lot has been done for satellite specific SR networks. In addition, most papers have tried to show the improvement in model scores while the purpose of this paper is to show the difference in how the networks are trained and its effect on computer vision tasks.

1.1 Past Works

A comprehensive review of SISR can be found here [1], but we attempt to provide a brief summary. SISR at its core is trying to map a low-resolution image to a higher resolution image based upon the pixels in the image (Fig 2). This process has had a growing interest over past years due to the rise of deep learning
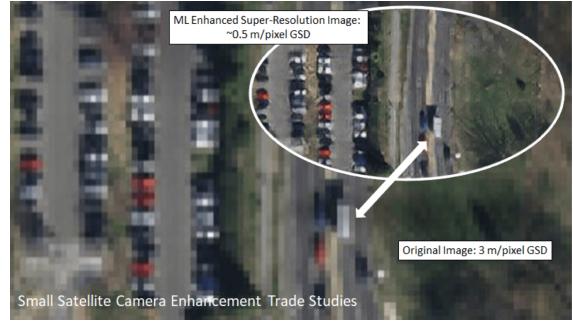

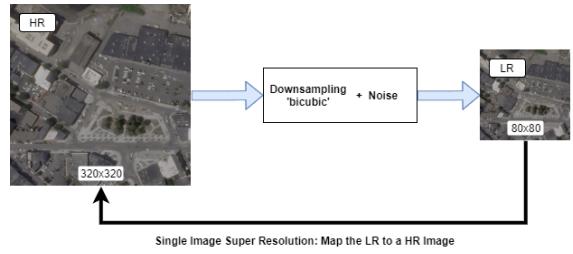Figure 1: SR's effect of increasing spatial resolution

Figure 2: Overview of SISR

and the growth of computing power. Yang et al. states that there are three categories of SISR: interpolation-based, reconstruction-based, and learning-based methods. 'Bicubic' [2] is the most popular algorithm for interpolation-based methods. An example of reconstruction-based method is [3] where Dai et al. details a soft edge smoothness algorithm. The final category is learning based methods which is the realm for this paper.

1.2 Related Works

A few papers have sparked our interest in performing this experiment. In [4], Kawulok et al. tested the down sampling method for training SR networks. They tested SRResNet and FSRCNN using DIV2K images [5] and Sentinal-2 images [6] (10m). They showed different image quality scores from different down sampling methods: Nearest Neighbors (NN), Bilinear, Bicubic, Lanczos, Lanczos-B, Lanczos-N, Lanczos-BN, and Mixed. NN showed the highest quality with 31 PSNR with other methods mostly scoring around 28 PSNR.

Furthermore, in [7], Shermeyer et al. performed SR as preprocessing step for object detection. The SR method used was Very-Deep-Super- Resolution [8] (VDSR) which was introduced in 2016 and used residual-learning and extremely high learning rates to optimize a very deep network fast. They scored the two object detection methods, YOLT [9] and SSD [10] and compared mean average precision (mAP) before and after super resolution. Shermeyer et al. found that for YOLT and SSD the largest gain in performance is achieved at the highest resolutions, as super-resolving native 30 cm imagery to 15 cm yields a 13-36% improvement in mAP.

In our research, we found another paper looking at the difference in performance from SRGAN training sets on non-satellite imagery [11]. They used the same network as this paper and applied it to datasets: CelebA [12], Dining Room [13], and Tower [13]. Intuitively, Takano et al. found that test images that contained objects from the training set performed much better than test images that contained no objects from the training set.

These papers and the changes they made to the single image super resolution pipeline was the inspiration for studying the effects that different training sets have on super resolution. We will first look at the networks used in the experiment before diving into the datasets used and the experiment itself.

2 Networks

This experiment used 3 networks; SRGAN for SR, a CNN for image classification and Mask R-CNN for object detection.

2.1 SRGAN

We use the SRGAN as described in [14] by Ledig et al in 2017. This was implemented in Keras in [15] with minor changes. The GAN in general flows from a generator to a discriminator (Fig 3). We down sample a high-resolution (HR) image into a LR image. Then we take the LR images and pass them through the generator to get SR images. We then compare the HR image to the SR image to get the content loss. We then pass the HR and SR images to the discriminator to try to predict if it is a fake image or not. We then get the GAN loss and pass that back to the generator. The generator and the discriminator networks can be seen in detail here (Fig 4) as shown in the original paper by Ledig. The interesting additions to the GAN network are the loss functions used [14]. The loss function for the discriminator is a Keras default binary cross entropy while the loss function for the generator (Eq 3) consists of two parts, content loss (Eq 2) and adversarial loss (Eq 1). As Ledig et al. describes, at large upscaling factors pixel-wise loss, such as mean squared error, fails to capture high-frequency content and leads to smoothed textures. Therefore, the content loss used is a perceptual loss function which compares the weights of the 19th layer of the VGG19 network for the HR vs SR images. As stated by Ledig et al, this was used in [16] for style transfer.
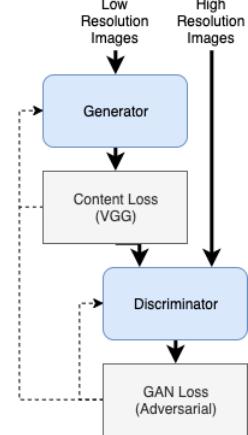

Figure 3: GAN flow outline

$$l_{GEN}^{SR} = \sum_{n=1}^{n} -log(D_{\theta_D}(G_{\theta_G}(I^{SR}))) \tag{1}$$

$$l_{VGG}^{SR} = \frac{1}{W_{(i,j)}H_{(i,j)}} \sum_{x=1}^{W_{(i,j)}} \sum_{y=1}^{H_{(i,j)}} \phi_{(i,j)}(I^{HR})_{(x,y)} - \phi_{(i,j)}(G_{\theta_g}(I^{LR}))_{(x,y)}^2 \tag{2}$$

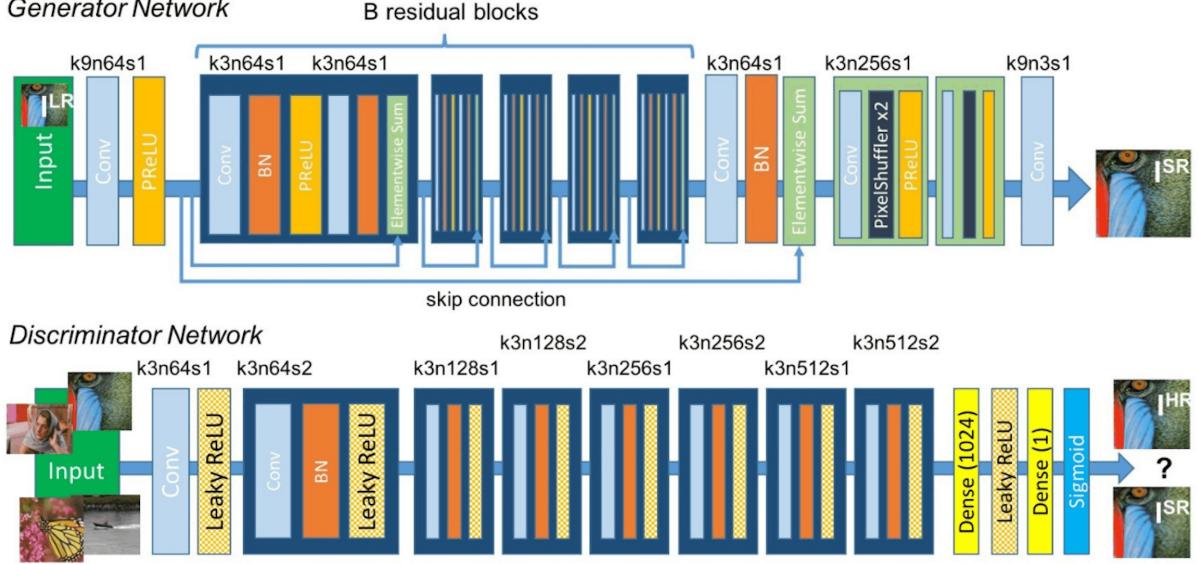$$l^{SR} = l_{VGG}^{SRR} + 10^{-3}(l_{GEN}^{SR}) \tag{3}$$

Figure 4: Architecture of Generator and Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer. Copy of original network image in [14].

2.2 Image Classification

The image classification network (Fig 5) attempts to reduce the image data to the same size latent space. This allows consistency between both networks. For both the 80px and 320px input, 2D convolutional layers are used with depth of 64, kernel size of 3 by 3, and rectified linear unit (ReLU) activation followed by 2D max pooling layers of size 2 by 2 until a feature space of 80 by 80 is achieved. We then instead use a Conv2D with depth 32 followed by a Maxpool2D with size 2 by 2 to achieve a latent space of 38 by 38. We then apply a dropout layer with rate of .25, Flatten the features, add a dense layer of size 128 with ReLU activation, a dropout layer with rate .25, and a SoftMax output to 2 classes. Loss is categorical cross-entropy and the optimizer is the default Keras Adadelta.
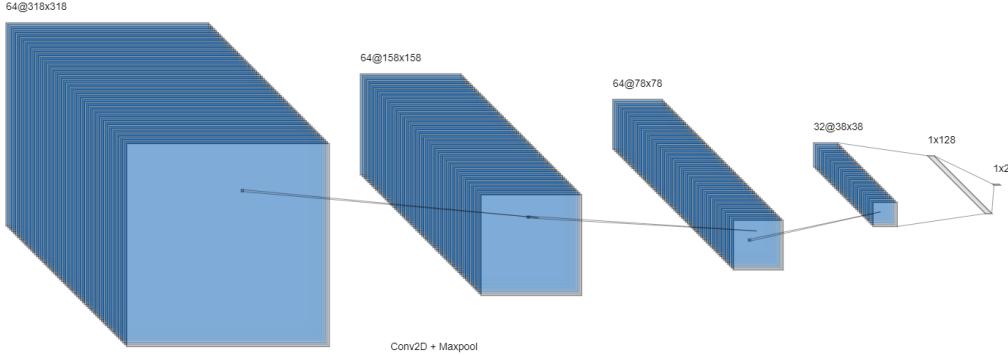

Figure 5: Conv2D + Maxpool + ReLu until 38x38 latent space for both 320x320 and 80x80 input

2.3 Object Detection

The object detection framework is Mask R-CNN released by He et al. in March of 2017 [17]. We used a fork of Matterport's code [18] which converted He's architecture to Python 3 and Tensorflow. We apply transfer learning from the MS COCO [19] pretrained Mask R-CNN model to our testing data. "Mask R-CNN is a simple, flexible, and general framework for object instance segmentation that efficiently detects objects and generates a high-quality segmentation mask. The method extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition." [17]

3

3 Datasets

The training set was from Skysat's 0.8m samples and the testing set was Planetscope's 3.0m Shipsnet images.

3.1 Training Set

To avoid having too many camera artifact variations in the dataset we want to use a single telescope's pictures for all the training images. To this end we went with Planet's 0.8m Visual Skysat samples [20]. This gave us the same telescope imaging system on vastly different landscapes. A visualization and RGB histogram of a single image and an average of the of each of the 5 training sets: Agriculture, Cities, Dry Bulk, Oil, and Ports, is show here (Fig 6).
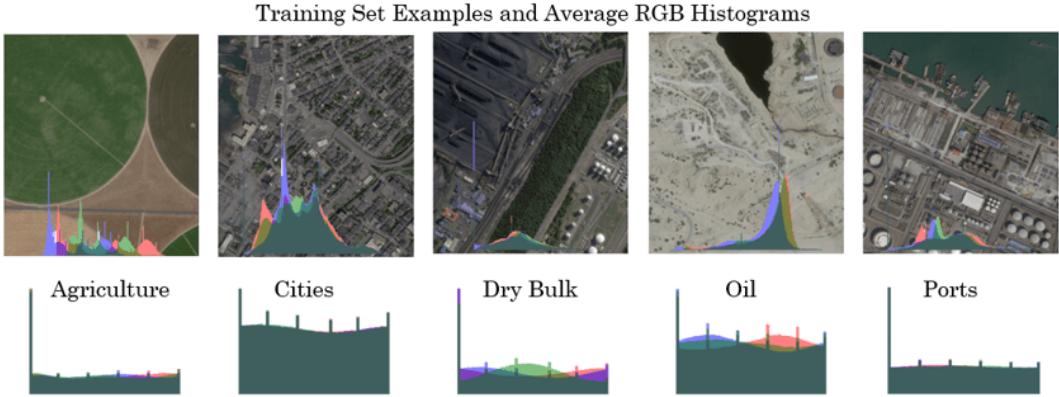

Figure 6: Plant's Skysat Visual Imagery. Example image and RGB histograms of land use classes

Images are pansharpened [21], orthorectified [22], color corrected RGB [23] and of average size 2560px by 1080px. These preprocessing steps alter the image to make it more visually appealing and useful to the user. Comparing images that have had different preprocessing would have added another variable to the experiment. Each training set contains 500 image chips with some sets reduced or slightly altered (rotation) to get to 500 images.

3.2 Testing Set

The testing set contains images from the Kaggle Shipsnet competition. Example image chips are shown here (Fig 7). These images are from Planet's Planetscope satellite (3.0m) They contain images of San Francisco Bay and San Pedro Bay areas of California. Like the training sets' images, these images are also scenes in Planet's visual classification meaning that they are also pansharpened, orthorectified, and color corrected RGB. However, unlike the training set, the scenes in the testing set have already been tiled into 80px by 80px squares. In terms of the classes in the dataset, only full frame ships are being classified as ship. All other images are classified as no-ship. The testing set contains 4000 images with 1000 images classified as ship.
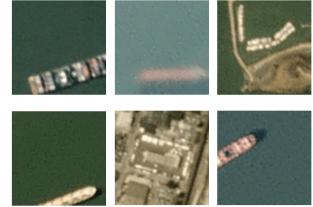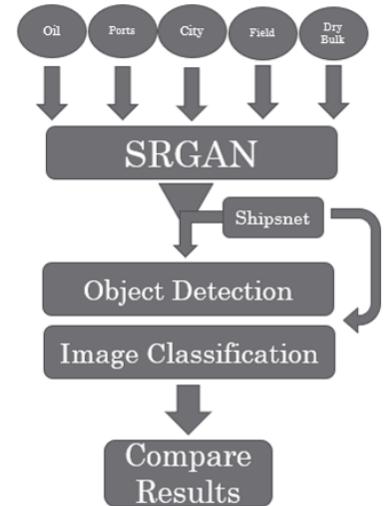

Figure 7: Shipsnet Images

4 Experiment

An overview of the experiment can be found in this flow chart (Fig 8). We took the HR imagery from the training sets and down sampled them to 80 px by 80 px with a bicubic filter and then trained the SR-GAN to up-sample 4x to 320x320 resolution. After the 5 SRGANs were trained we took the testing set and ran it through each of the SR-GANs. We also scaled the testing set to the same 320x320 resolution to compare to the SR images directly. We then ran the SR ships, the raw ships, and the scaled ships through the image classification and object detection models to compare the performance of each computer vision task.

Figure 8: Experimental flow chart

4.1 Preprocessing

Preprocessing the images in the training set required a few steps. Since the size of the test images are 80px by 80px we needed to get 320px by 320px images from our training set in order to satisfy the 4x up-sample. With the average size of the training set scenes being 2560px by 1080px we needed to tile the images into 320px by 320px PNG squares (Fig 9). To compare the super resolution images with those of a scaled-up test set, we used Image Magick's Mitchel-Netravali bicubic filter [24] to scale the 80px test images to 320px.
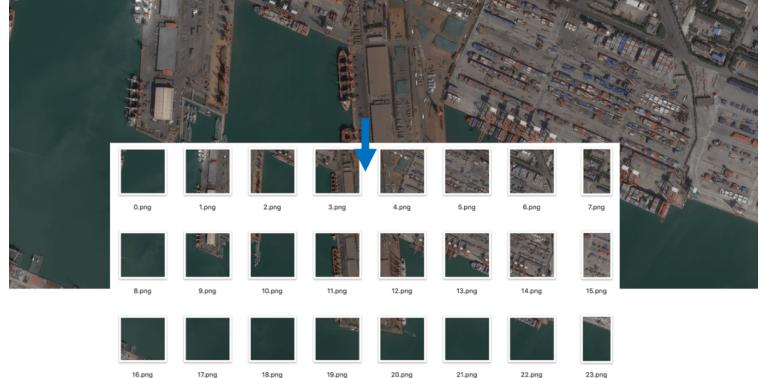

Figure 9: Process of tiling image scenes into desired resolution squares

4.2 SRGAN

With all our images in the required resolution we can now setup our SRGAN network. After loading our data, we need to first down sample to get our LR image. We again use a bicubic filter to down sample the images to reduce the effect of the sampling method on the results as talked about in [4] by Kawulok et al. We then mean normalize all pixel values between -1 and 1 before running the data through the GAN. We train 5 SRGAN's, one for each training set, before moving on to the computer vision tasks. We trained for 5000 epochs with a batch size of 16 for 500 images. After training the SRGANs, we ran inference on some down-samples of popular satellite image samples: Xview(.3m), Pleides(.5m), Quickbird(.65m), Triplesat(.8m), and Ikonos(1m). In addition, PSNR/SSIM metrics were calculated for each SR model on each of the training datasets to compare how a SR model that was trained on its own test data performed against the other SR models.

4.3 Image Classification

The image classification step is straight forward. We create two networks to accommodate the 320px and 80px inputs. As described in the network details, we run the data through convolutional layers and max pools until we reach the same 38x38 latent space size. After creating the network, we trained the image classifier on our 7 data sources: the 5 SR images, the scaled images, and the raw images. We use a batch size of 32, 100 epochs, 20% validation split and normalize data between 0 and 1. In addition we apply some data augmentation: 10-degree random rotation, random width shift and height shift of 0.1, and random horizontal flip.

4.4 Object Detection

To run the image through the object detection framework we need to first grab the images from SRGAN and annotate them. To label for object detection we needed to create PASCAL Visual Object Classes (VOC) object detection xml files. We used LabelImg [25] to draw bounding boxes on our 1000 ships for the SR images and the raw images (all the SR images have the same annotations). Mask R-CNN is built on FPN [26] and ResNet101 [27]. We used the default parameters as provided by the config file of Matterport's repo [18] while running 5 epochs. We take the 1000 labeled images and split it into 700 training images, 150 testing images and 150 validation images.

5 Results

The experiment took approximately a week from start to finish for just training. Training for each SRGAN took around 48 GPU hours each on a Nvidia V100 32GB. Each model was trained on a single GPU. Inference for SRGAN was around 37 fps for the 17MB model on a Xeon E5-2698 v4 (50M Cache, 2.20 GHz). Image classification models took around 2 hours to train while object detection models took 20 minutes on a Nvidia V100.

5.1 SRGAN

We can see the progression over the 5000 epochs in (Fig 10). We ran a couple SR training image in addition to running inference on the test set. We can see that over time the model learned a better color mapping between the super resolved images and the original images. In addition, over time a more defined building structure can be seen in the desert picture highlighting the strength of the perceptual loss function.

For images of the ships (Fig 11), the Agriculture trained SRGAN produced a much darker image and the Cities trained SRGAN produced a much greyer image. This could possibly be due to the different color distributions present in the training sets. We also notice the artifacting present in some of the SR images, particular the "black spots" located in the Oil trained SRGAN.
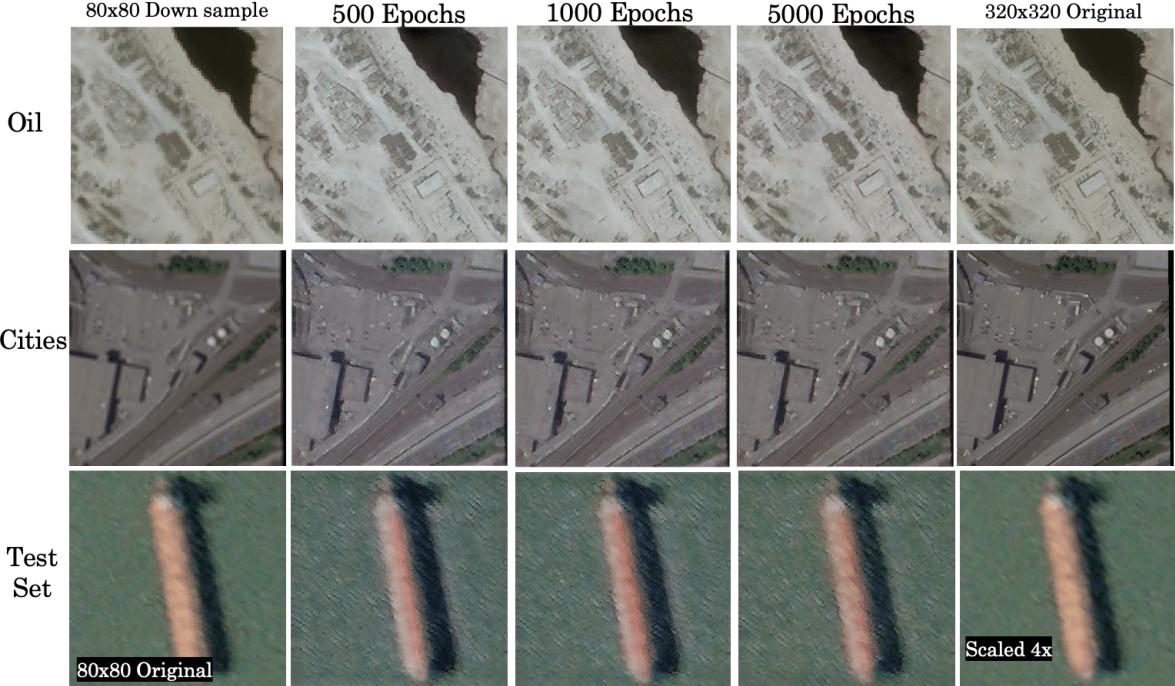

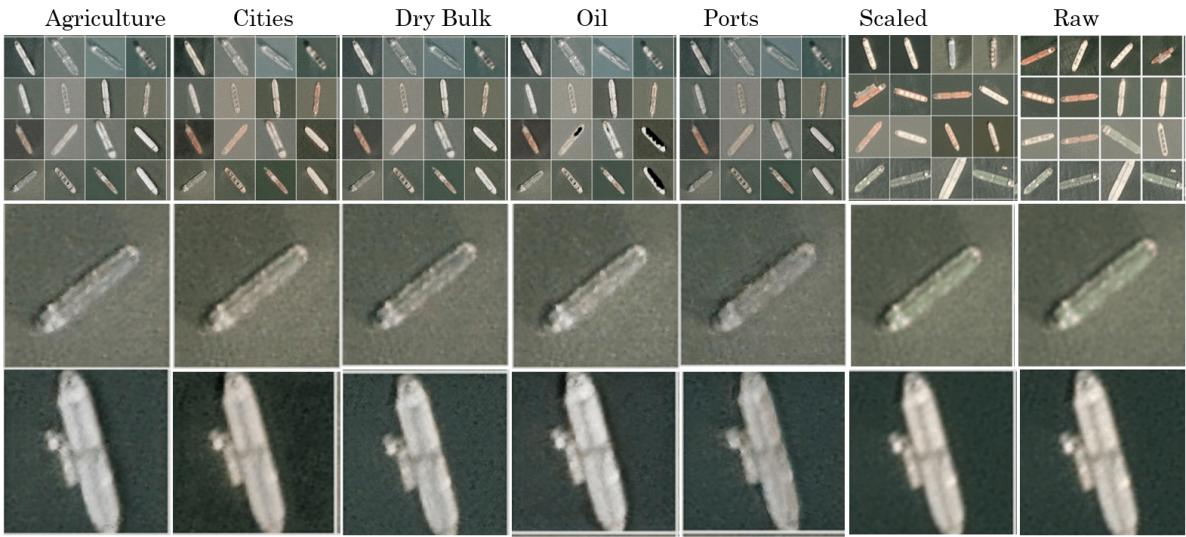Figure 10: Improvement of SRGANs over epochs

Figure 11: Montage image compare between the 7 testing sets

The comparison (Fig 12) between SR models and all the training sets show consistent relative performance for each model across all the datasets. We see that the models did the best on the homogeneous agriculture images while they did the worst on the images of heterogeneous cities.

In terms of metrics, image quality scores for various satellite image samples are shown here (Fig 13). To show SRGANs sensitivity to spatial resolution of the training set we tested various 4x down-samples. For example Pleiades' .5m spatial resolution was down-sampled 4x to 2m while Triplesat's .8m spatial resolution was down-sampled to 3.2m before being super resolved and compared to the original image. This is meant to highlight the need for a tailored training set on not only the ontology but also the spatial resolution.
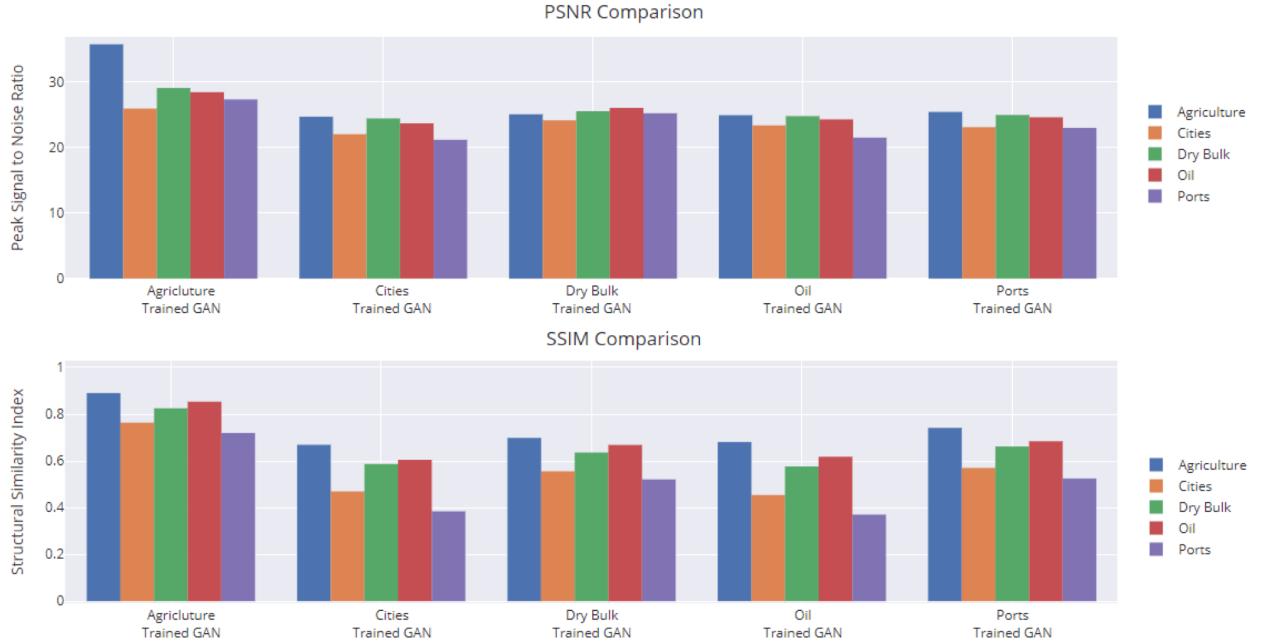

Figure 12: PSNR/SSIM metrics for SR models with all training sets

Dataset	Scoring	Agriculture	Cities	Dry bulk	Oil	Ports	Average	Example Images
Xview (.3m)	PSNR	17.011623	16.981220	17.307997	16.801039	17.162341	17.05284	
	SSIM	0.491469	0.477119	0.451492	0.483934	**0.477334**	0.47627	
Pleiades (.5m)	PSNR	**22.770306**	19.560364	**21.841543**	21.544979	18.182035	**20.77985**	
	SSIM	**0.625712**	0.413790	**0.541247**	**0.562504**	0.293370	**0.487325**	
Quickbird (.65m)	PSNR	22.087340	18.617457	18.647200	17.486228	17.244123	18.81647	
	SSIM	0.583851	0.371042	0.451841	0.463415	0.270718	0.428173	
Triplesat (.8m)	PSNR	20.925616	18.843711	20.599660	20.514242	17.674832	19.71161	
	SSIM	0.471851	0.423649	0.504717	0.509565	0.342879	0.450532	
Ikonos (1m)	PSNR	20.413195	**19.709312**	21.795880	**21.917841**	**19.604518**	20.68815	
	SSIM	0.483809	**0.449529**	0.518172	0.531595	0.394726	0.475566	

Figure 13: Scores for SRGAN applied to sample images from different satellites. Averages for 50-200 image chips.

5.2 Image Classification

Accuracy over epochs shown here (Fig 14). Overall, image classification went very smoothly due to the already high validation accuracy on the raw dataset (98%). However, we can look at the few misclassifications for our networks (Fig 15). Results show that the SRGANs that made the most artifacts (i.e. oil trained SRGAN), had lower image classification scores because of it. We see that the ports SR images outperformed the raw validation accuracy by 20 images.
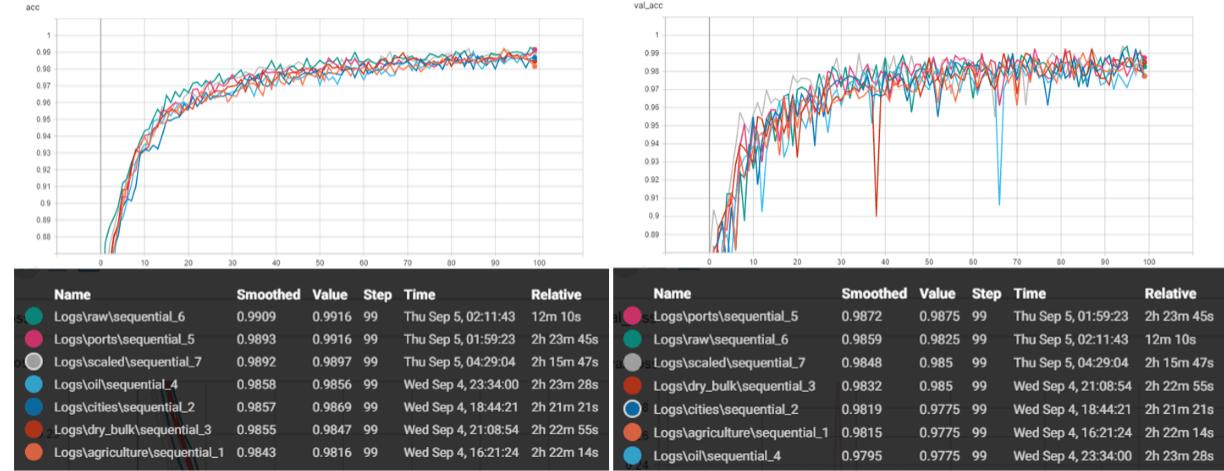

Figure 14: Training and Validation accuracy for the image classification network using the Super Resolution images, the raw images, and the scaled images
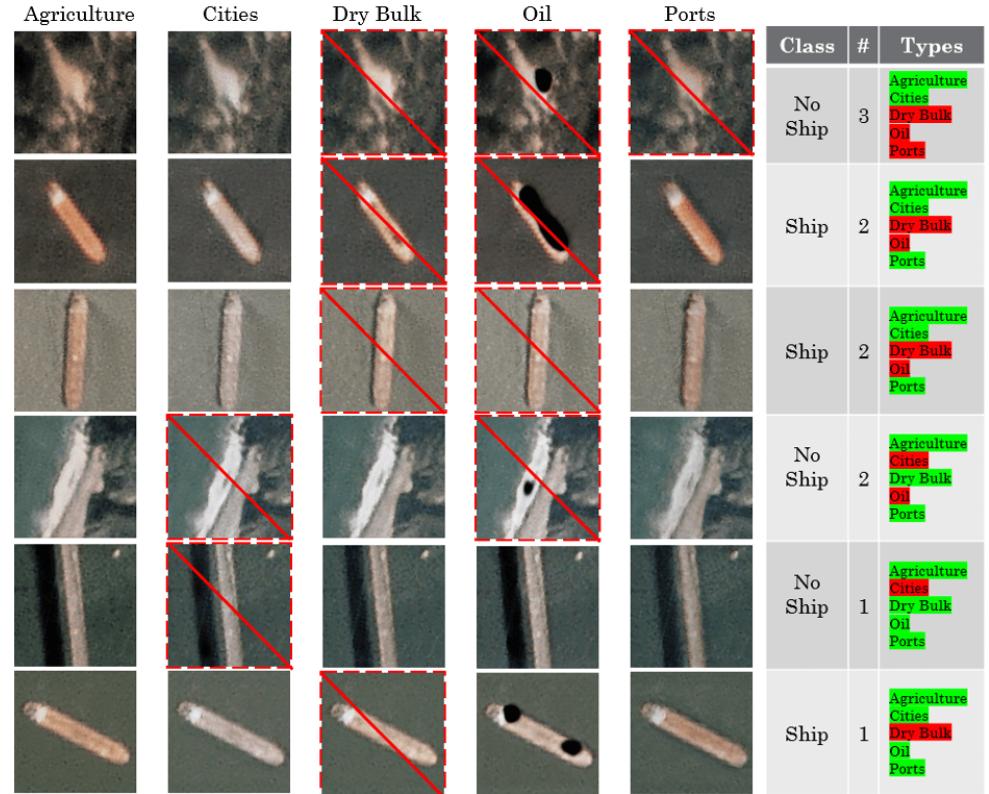

Figure 15: Image classifications failure highlights

5.3 Object Detection

The results for object detection show a clear improvement in average precision (AP) when comparing the SR images to the raw images trained Mask R-CNN (Fig 16). AP for both the training set and the test set are provided. We notice that the cities SR images have a near perfect precision with missing only 4 ships in our training set and 1 image in the testing set. On average, the SR models had a 18.4% higher AP than the raw image trained Mask R-CNN.

We can see the actual predictions made by the network in (Fig 17).We can see the multiple detections made on the same ship possible due to a lower than needed confidence threshold but overall the models performed very well when compared to the raw image trained network.
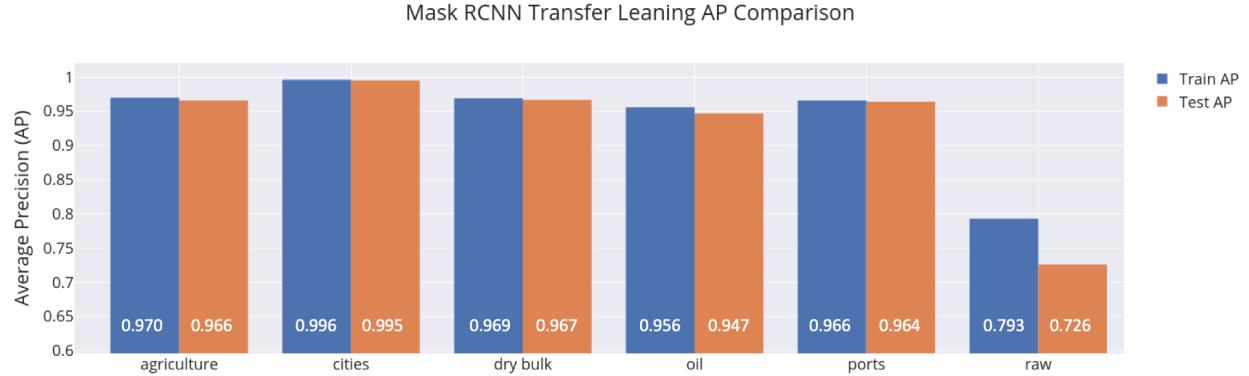
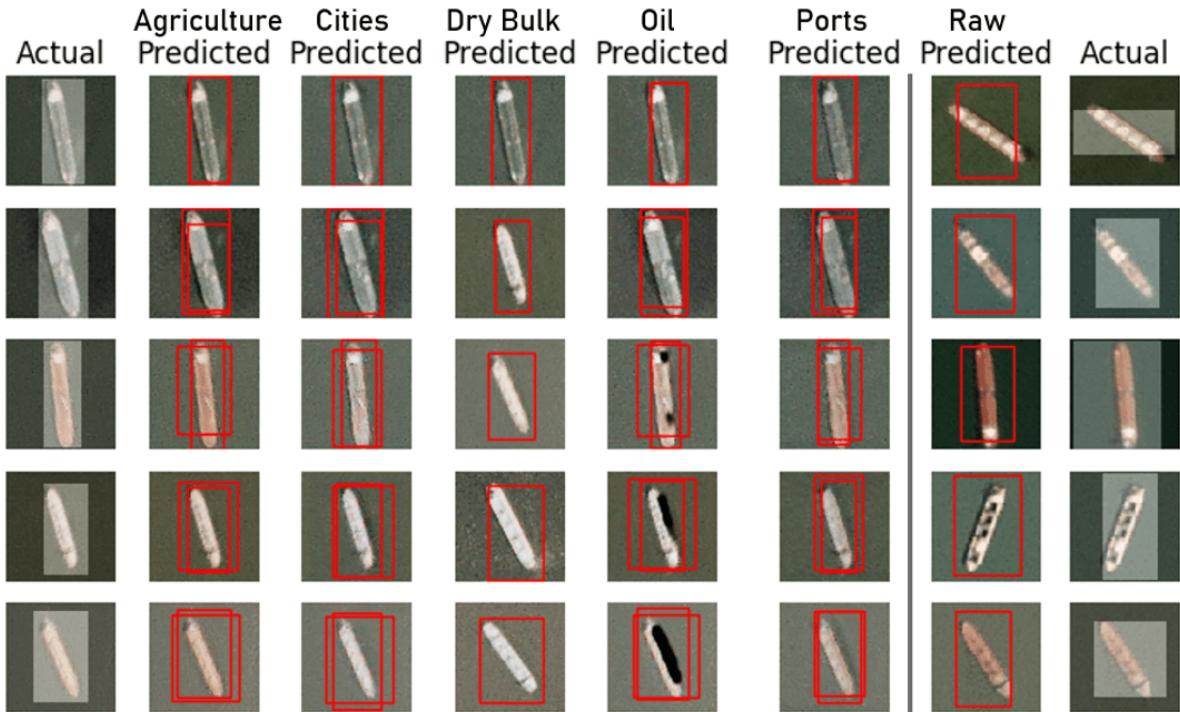

Figure 16: Train/Test Average Precision

Figure 17: Mask RCNN Object Detection Comparisons

9

6 Conclusion

Clear differences are seen between the SRGAN networks when trained on different sets of imagery. Performance of our super resolution (4x) is similar to the state of the art for non-satellite imagery (Agriculture vs. Urban 100) standing at PSNR / SSIM scores around 27.7 / .79 vs. 27.1 / .82. For the SR models with all the training sets comparison we notice consistent relative strength. This indicates that the quality of the data that a SR model is trained on is paramount.

Image classification showed preference for being trained on images that were in the testing set with marginally higher accuracy for SRGANs trained on images of ships. Only 1 (ports) of the 5 trained SRGANs had higher validation accuracy (98.72%) than the raw test set (98.59%). Depending on the difficulty of the test set, will see much smaller, if any, increase in image classification performance if raw accuracy is already high.

For object detection, we see an average of 18.4% improvement in precision when comparing the SR trained model versus the raw image trained model. The cities trained Mask R-CNN followed by the ports trained model preformed the best, out recognizing around 100 images when compared to the raw image trained model. This suggest that having objects in the test regime (ports) and having a complex and diverse set of images (cities) gives good results for super resolution for object detection.

Overall, we notice a clear trend in that having a diverse dataset with objects in the test ontology allow downstream tasks to perform better after super resolution.

6.1 Next Steps

Future work with testing on various datasets [11] will validate the increase in performance from SR networks. Using neural networks that allow varying image size inputs would allow one to use SR on images of arbitrary size unlike the fixed size of SRGAN. Many frameworks are currently out there that do this [28]. These networks avoid the user's need to preprocess the images into a required format since they handle various sizes. Refining the method used for object detection could yield clearer results. Future testing, with different neural network architectures [29], could yield better results. An entire view into super resolution can be found here [30].

6.2 Extras

An interesting paper we came across in our research was Chao Ma's et al. [31] with their comparison of human subject scores as compared with image metrics (Fig 18). This shows that the commonly used metrics are not always as telling of the quality of an image.
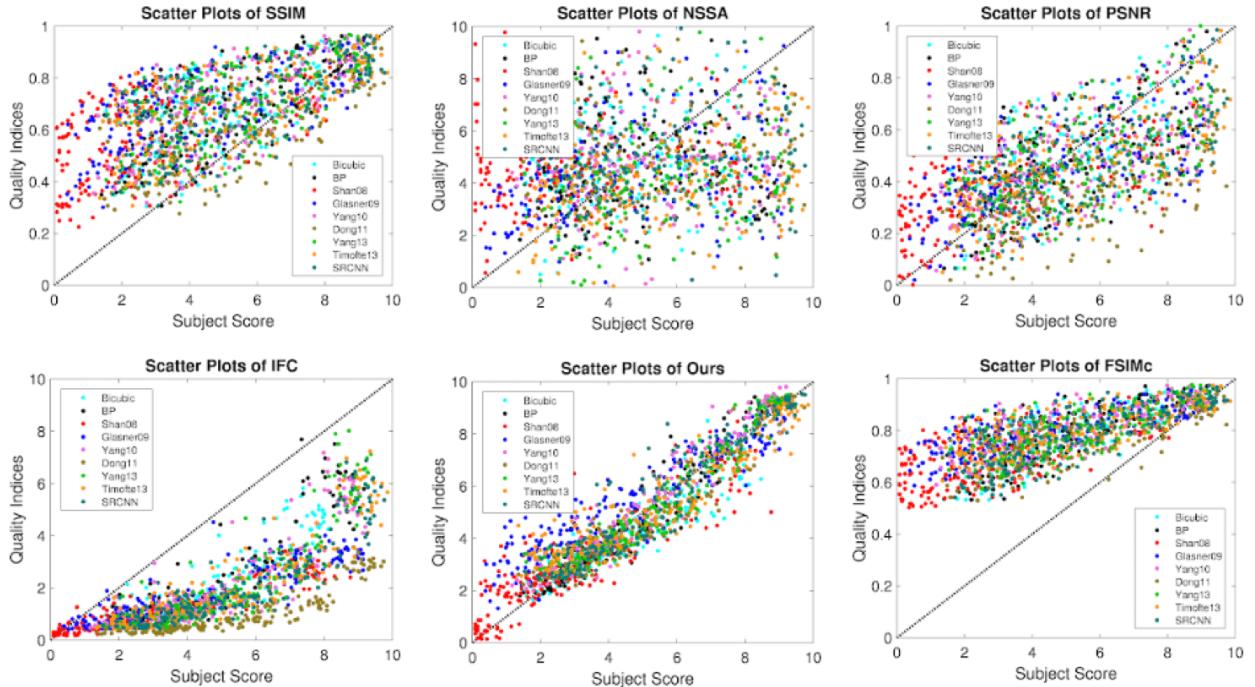

Figure 18: Ma's [31] Human Subject Scores vs Image Metrics

To show the difference between SR models we ran an absolute difference matrix with two sample images (Fig 19).This comparison really emphasized the artifacting that each SRGAN model learned during training. For example, clear artifacting "black spots" are seen in the dry bulk SR image of the city.
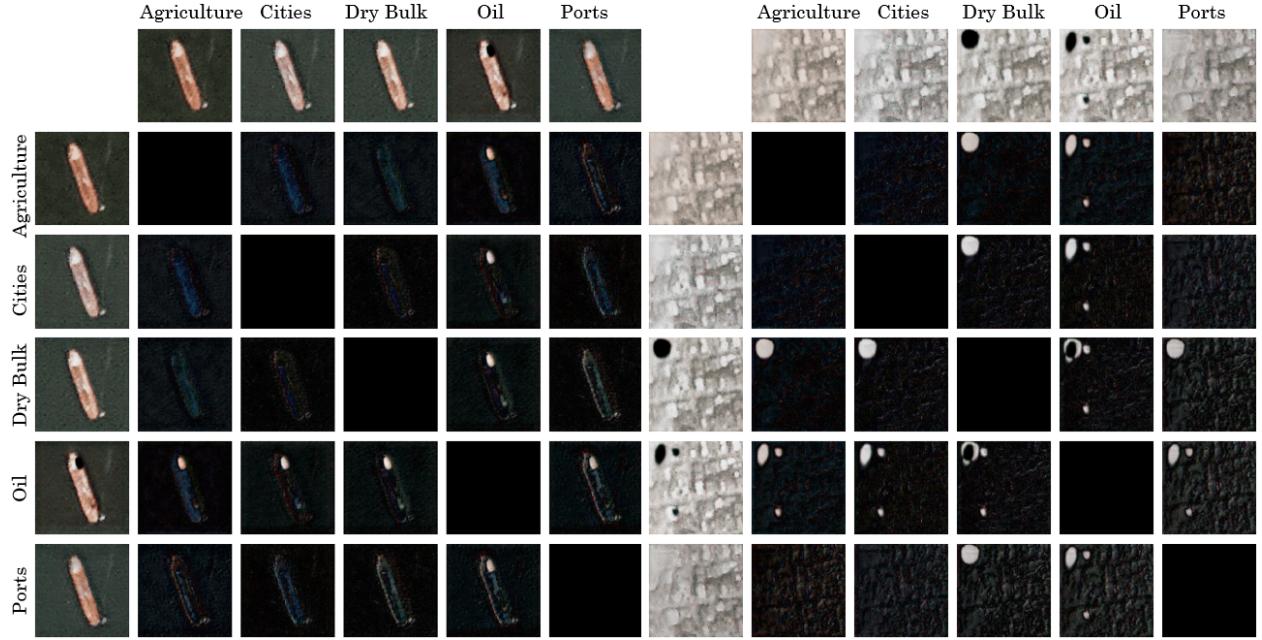

Figure 19: Image comparison between SRGANs

References

[1] Wenming Yang, Xuechen Zhang, Yapeng Tian, Wei Wang, Jing-Hao Xue, and Qingmin Liao. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 2019.

[2] Robert Keys. Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, 29(6):1153–1160, 1981.

[3] Shengyang Dai, Mei Han, Wei Xu, Ying Wu, Yihong Gong, and Aggelos K Katsaggelos. Softcuts: a soft edge smoothness prior for color image super-resolution. *IEEE Transactions on Image Processing*, 18(5):969–981, 2009.

[4] Michal Kawulok, Szymon Piechaczek, Krzysztof Hrynczenko, Pawel Benecki, Daniel Kostrzewa, and Jakub Nalepa. On training deep networks for satellite image super-resolution. *arXiv preprint arXiv:1906.06697*, 2019.

[5] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 126–135, 2017.

[6] Sentinel-2a (10m) satellite sensor.

[7] Jacob Shermeyer and Adam Van Etten. The effects of super-resolution on object detection performance in satellite imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[8] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016.

[9] Adam Van Etten. You only look twice: Rapid multi-scale object detection in satellite imagery. *arXiv preprint arXiv:1805.09512*, 2018.

[10] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[11] Nao Takano and Gita Alaghband. Srgan: Training dataset matters. *arXiv preprint arXiv:1903.09922*, 2019.

[12] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 15:2018, 2018.

[13] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[14] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[15] deepak112. deepak112/keras-srgan, Apr 2019.

[16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[18] Waleed Abdulla. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. `https://github.com/matterport/Mask_RCNN`, 2017.

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[20] Skysat sample imagery, 2019.

[21] Chris Padwick, Michael Deskevich, Fabio Pacifici, and Scott Smallwood. Worldview-2 pan-sharpening. In *Proceedings of the ASPRS 2010 Annual Conference, San Diego, CA, USA*, volume 2630, 2010.

[22] Satellite Imaging Corporation. Orthorectification, 2019.

[23] Planet. Skysat imagery products specification, Nov 2018.

[24] Imagemagick v6 examples – resampling filters.

[25] tzutalin. Labelimg is a graphical image annotation tool and label object bounding boxes in images. `https://github.com/tzutalin/labelImg`, 2017.

[26] T Lin, P Dollár, RB Girshick, K He, B Hariharan, and SJ Belongie. Feature pyramid networks for object detection. corr, vol. *arXiv preprint arXiv:1612.03144*, 2016.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. corr abs/1512.03385 (2015), 2015.

[28] Idealo. idealo/image-super-resolution, Sep 2019.

[29] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11065–11074, 2019.

[30] Saeed Anwar, Salman Khan, and Nick Barnes. A deep journey into super-resolution: A survey. *arXiv preprint arXiv:1904.07523*, 2019.

[31] Chao Ma, Chih-Yuan Yang, Xiaokang Yang, and Ming-Hsuan Yang. Learning a no-reference quality metric for single-image super-resolution. *Computer Vision and Image Understanding*, 158:1–16, 2017.