Evoke: Interactive Visualization for Voluminous Satellite Data Through Image Super-Resolution

Saptashwa Mitra, Daniel Rammer, Shrideep Pallickara, Sangmi Lee Pallickara
Department of Computer Science, Colorado State University, Fort Collins, USA
{sapmitra,rammerd,shrideep,sangmi}@colostate.edu

***Abstract*—Enabling interactive visualizations over voluminous satellite data collections results in both processing and I/O (network and disk) on the server side. Hotspots may also arise when multiple users are concurrently interested in a particular geographical extent. In this study, we propose a novel methodology to support interactive visualizations through our system, Evoke, which generates models that once installed on the client side, substantially alleviates resource requirements on the server side. The model, based on Generative Adversarial Networks, dynamically generates imagery during zoom-in operations and is space-efficient to facilitate memory-residency at the clients.**

***Index Terms*—super-resolution, generative adversarial networks, in-memory storage, visual analytics**

I. Introduction

Voluminous datasets generated by remote sensing technologies play a key role in making critical decisions in domains such as geosciences, business, public health, national security, and disaster management. However, making data-centric inferences in a timely manner is challenging given the volumes, high dimensionality, and heterogeneity of data sources. Interactive visual analytics allows users to identify target information and frame hypotheses to guide and inform subsequent analytics.

Challenges in enabling interactive visual analytics over geospatial data stem from the nature of data accesses and their I/O footprints and are exacerbated by data volume and frequent data retrievals by users. In the case of geospatial analytics, multiple users can concurrently perform a series of actions such as drill-down, roll-up, panning, and adding overlays which involve frequent and intensive network communication (spatiotemporal queries) and disk I/O at the backend servers.

Users' access patterns for certain activities such as drill-down or roll-up render spatiotemporal information over overlapping regions between successive retrievals. To render visualizations at different resolutions (while coping with limited resources), visual analytics applications often rely on data aggregation [1] [2] and sampling techniques. Several research studies have explored both static and dynamic approaches to caching data in-memory on a distributed server [3]–[6]. Maintenance and load-balancing of caches in a distributed system to alleviate such hotspots have been explored in [4], [5]. However, pre-loading, load-balancing, and maintenance of these caches introduce computation overheads and server-side memory constraints and network bandwidth.

The primary goal of this study is to support interactive visual analytics at scale over multi-resolutions to help reduce data transfers even when the number of retrieval requests is high. Evoke, a generative model, adapts the concepts of progressive Generative Adversarial Networks(GAN) [7] to cope with unique characteristics of satellite imagery.

(a) Input (Bicubic) (b) Basic SR (c) HR Target

Fig. (1) Super-resolution results for 8x upscaling scenario.

A. Research Questions

Research questions that we explore in this study include:
RQ1: How can we reduce data transfers and disk I/O while supporting interactive visual analytics over voluminous satellite imagery?
RQ2: How can we generate an effective representation of data to enable multi-resolution exploration activities without the high-velocity server-side accesses, which the analytics would entail otherwise?

B. Overview of Methodology

Our methodology reconstructs images at diverse resolutions while significantly reducing data communications between the user and the backend storage system. We construct a variation of a Progressive Generative Adversarial Network (GAN) that captures non-linear relationship between low-resolution images with different zoom-levels to their super-resolution (SR) counterparts. The primary benefit of using a progressive GAN is generating a consolidated model that captures interactions that exist between arbitrary pairs of levels across the entire spectrum of resolutions.

Empirical evaluation of Evoke's model showed improvements in interactivity, through reductions in latency of up to ∼23x without GPU acceleration and ∼297x-6627x with GPU acceleration, compared to server-side data retrieval. Our generated images also had good perceptual quality with PSNR values ranging from ∼32.2-36.9, depending on the scenario and upscale factor and showed improvement in PSNR ranging from ∼9.5-13 compared to that of an SRGAN [8].

C. Paper Contributions

Our contributions include: (1) A Super-Resolution model that upscales satellite image tiles with 6 different levels of resolution. At each level, our model quadruples the resolution of the image. (2) Our methodology precludes excessive data

retrievals and transfers from frequent view changes triggered by a user. (3) Evoke is memory-resident at the client side. A single super-resolution model is applicable for any current resolution level with different upscaling rates.

II. Background and Related Work

Tile layers [9] are a widely used data-structure in visual analytics. They consist of a set of partitioned, rasterized, and pre-rendered image tiles, stored on a server, that are fetched in groups through user queries [10]. In visualization systems with interactive, sequential OLAP [11] operations such as panning, drill-down, roll-up, and zooming, a pyramid model of tile layers (raster pyramid) is frequently used to organize the tile layers in increasing order of their resolution (generally in multiples of 2) [12], [13].

QuadTiles [14] is one of the most common tile indexing strategies used to partition the dataset into tile layers for visualization. At each successive level, a quad tile's (parent) spatial extent is split into 4 equal sub-tiles(children). In this study, the problem of *in-situ* image super-resolution boils down to using a given tile image at layer/resolution l to generate the image formed by its children at layer $(l+1)$,$(l+2)$,$(l+3)$,... for magnifications of 2x,4x,8x,... respectively.

Single Image Super Resolution (SISR) [8], [15], [16] takes a low-resolution (LR) image and reconstructs a high-resolution (HR) image as output. The use of residual connections to implement deeper convolutional neural networks has been adopted [16] to avoid the network suffering degradation of accuracy with depth. Recently, adversarial learning using Generative Adversarial Networks**(GAN)** [17] has produced realistic-looking images. However, achieving HR image with higher upscaling factor is challenging because significantly down-sampled image cannot preserve the crucial high-frequency information even if it still shows high PSNR values [18] resulting in blurry images. There are several approaches to improve the perceptual quality of the super-resolved satellite images [18]–[20].

Most existing SISR implementations build models targeted at a single upscaling factor. This increases the complexity of model training; each model with different upscaling factors should be trained separately from scratch and results in prolonged training times. Wang et al [21] have proposed incremental training of GANs (by leveraging progressive GANs [7]) that achieve upscaling factors of up to x8. We harness this approach to train a model to achieve a magnification level, m, and upon convergence append an extra set of dense layers to the trained model to train for 2m and reconstructing the image in intermediate steps by progressively performing a 2x upscaling of the input and combining it with the residual output of the model. EvokeNet also incorporates geospatial metadata into the models to provide conditional information to adapt for satellite imagery.

III. Methodology

Evoke targets space efficiency and residency within an in-memory hierarchical tile cache at the client machine [22] to improve the interactivity of visual analysis. This involves reusing coarser/partial tile information to locally generate a high-resolution version of a tile, which is not currently available in-memory. EvokeNet model is a variation of Generative Adversarial Networks (GANs) that estimates super-resolution images by taking into consideration the amount of partial high-resolution information available in the local cache from previous queries along with additional image metadata.

A. Model Overview: *[RQ1,RQ2]*

Our model learns an upsampling function that transforms a low-resolution image tile ($\mathbf{I_{LR}}$) to generate a high-resolution synthetic rendition ($\mathbf{I_{SR}}$) that is contrasted against the ground truth ($\mathbf{I_{HR}}$).

The main challenge is facilitating a multi-level *super-resolution/upscaling* operation for an image tile for diverse *zoom levels*. The diverse topographical characteristics in satellite imagery at different zoom levels adds complexity to the problem of model-based upscaling of images. We propose a conditional adversarial network that utilizes the concept of *progressive growing of GANs* [7]. Finally the sequential training of all the sub- models (x2,x4 and so on) captures the relations between all the levels of cascading resolutions without going through disjoint training phases.

Model Input: In our work we extract two tile metadata/ properties to **condition the inputs to both our generator (G_1) and discriminator (D_1)** – (1) the zoom level of I_{LR}, and (2) the dominant land cover type(based on their National Land Cover Database (NLCD) codes [23], e.g. urban, forest, barren etc.) of I_{LR}. This extracted conditional information ,g, is embedded and then passed through a fully connected linear activation layer to resize the embedding to an extra 4^{th} metadata channel that gets appended to the 3-channel RGB image as input to G_1 (as shown in Fig.2).

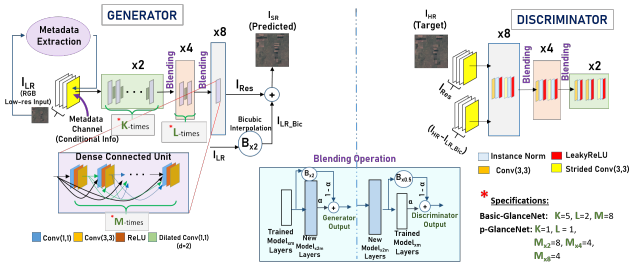


Fig. (2) Evoke Super-Resolution Model

Generator Network(G_1): The generator output for our model for an upscale factor of u can be represented as:

$$\mathbf{I_{Res}} = \mathbf{G_1^u}(\mathbf{I_{LR}}|\mathbf{g}) \quad (1)$$

Instead of the complex mapping between I_{LR} and I_{HR} for any given upscaling factor, the super-resolution problem is simplified to learning the residual, I_{Res} that needs to be applied to a bicubic interpolation (upscaled) of I_{LR} (eq.2).

$$\mathbf{I_{SR}} = \mathbf{I_{Res}} + \mathbf{bic}(\mathbf{I_{LR}}) \quad (2)$$

The progressive training of the models is a form of *curriculum learning*, where more complex upscaling models are built on top of trained models that handle relatively simpler tasks of a lower upscaling factor and improves stability [7].

In order to maintain efficiency in a deep network and to facilitate the flow of weights between its layers, we adapt the concept of a densely connected network as introduced in DenseNet [24], along with a variation of DenseNet's Dense Blocks- the Dense Connected Units [21] (see Fig.2). We also adopt an *asymmetric pyramid* (Fig.2) structure for our generator layers - the number of dense layers [24] used in modeling is higher for lower level upscaling problems and are progressively less complex as we upscale further. We have limited the maximum achievable super-resolution to x8 – beyond that, the generated satellite images cease to maintain their desired perceptual quality.

To achieve high perceptual quality in our super-resolution output, we target improving the *receptive field* of our model, which captures relationships between distant regions in the image. Although increased sub-sampling (downsampling) between layers (eg. through max-pooling) is known to improve the receptive field, it also introduces blurriness in the image output, especially for higher upsampling factors. To help improve super-resolution operations for x4 and x8, we introduce **dilated convolutions** [25] between the Dense Blocks, instead of subsampling. This reduces the complexity of the model while improving the receptive field through the newly added layers, as shown in Fig.2. The switch from one model to the next upscaling model is performed through the process of gradual **blending** [7] (Fig.2).

Discriminator Network(D_1): The discriminator layers (D_1) follow a similar incremental structure as the generator layers, but are significantly simpler. The discriminator(s) are adapted from the PatchGAN discriminator [26] to evaluate local topographical features for the geospatial areas. The discriminator produces an array of output, where each element evaluates a patch of the input image, instead of the entire image, which help model high-frequency details in the generated image.

Objective Functions: We have used two loss functions, optimized using an Adam optimizer [27]. Our *reconstruction loss* is the average pixel-wise difference between the generated and the real high-resolution image. Given a sample (of size N) of the i^{th} input for our model - (x_i, g_i, y_i), where the individual elements represent the low-resolution input (of dimension CxWxH), its geospatial metadata and the corresponding high-resolution target, respectively, its reconstruction loss is:

$$\mathcal{L}_{rec} = \frac{1}{N \times C \times W \times H}[y_i - (bic(x_i) + G_1^u(x_i, g_i))]^2 \quad (3)$$

The *adversarial loss* is computed as

$$\begin{aligned}\mathcal{L}_{adv} &= E[log(D_1^u(y_i, g_i))] \\ &+ E[log(1 - D_1^u((bic(x_i) + G_1^u(x_i, g_i))))]\end{aligned} \quad (4)$$

The **joint loss** is computed as $\mathcal{L} = \lambda_1(\mathcal{L}_{rec}) + \lambda_2(\mathcal{L}_{adv})$. The generator is trained on this joint loss, where λ_1 and λ_2 are pre-determined weights, while the discriminator is trained to minimize the adversarial loss.

IV. System Architecture

Fig.3 outlines the architecture and the interaction between the various components of our system with the Evoke framework.

A. Distributed Data Store

We partition our voluminous satellite imagery based on quadtiles, producing easily-indexed spatiotemporally bounded extents. Our system load-balances cluster storage using a multi-token, single-hop distributed hash table (DHT) [28]. The DHT key is the quadtile for each partitioned image. We distribute on quadtile substrings, effectively collocating data from neighboring regions on a single node.

B. In-memory Tile Cache

Evoke is designed to act in tandem with an in-memory storage **on a client machine** that stores the most relevant/recent image tiles accessed in previous queries. The main contribution of the tile cache framework to our case is its ability to identify and retain the most recent and relevant tiles in-memory, based on the client's historical access patterns. Upon receiving a spatiotemporal query, the client-side Evoke framework first consults its in-memory tile cache to find any requested memory-resident tiles (Fig. 3). For the missing target tiles, we seek coarser tiles in-memory (searching for parents of the target tile in the *tile layer*) and use those to create their corresponding high-resolution version on the fly, thus bypassing communication with the backend servers.

C. Distributed Training

EvokeNet is trained over a distributed spatiotemporal filesystem and upon convergence, made available to the clients for download. Our training leverages the spatiotemporal partitioning scheme of the storage system by collocating the training modules with the partitioned data so as to avoid data movements. The distributed training utilizes a parameter server to aggregate the weights asynchronously at certain intervals through the use of Pytorch Lightning [29] in the Distributed Data Parallel (DDP) mode. To circumvent a large-scale retraining, in light of new data, we re-purpose the previously trained EvokeNet models and update the stale weights through transfer learning [30]. To facilitate efficient transfer learning for our models, we oversample the new data, which is easy to identify in our geospatially partitioned storage system.

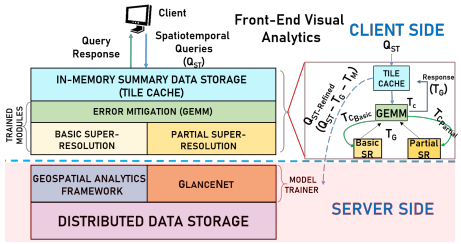


Fig. (3) Evoke System Architecture

V. Evaluation

A. Experimental Setup

EvokeNet models were trained and evaluated over a cluster of 50 nodes (Xeon E5-2620, 64 GB Memory), each with a Quadro P2200 GPU (5GB of memory) with 1280 cores. We performed our experiments with visualization frames of 640 x 640 pixels and 11-16 spatial resolutions (zoom-levels). With every increase in zoom-level, an tile's resolution doubles. At any zoom level, we support upscale factors of x2, x4, and x8.

Model Type	Upscale Factor	x2	x4	x8
Basic SR	Using Supplemental Metadata Info	36.95	33.85	32.17
	Using only RGB Image	35.65	32.77	31.56

TABLE (I) Utility of Supplemental Metadata Information

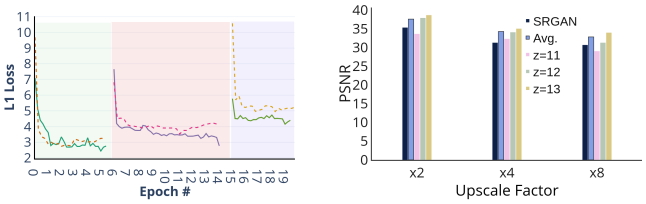
Our satellite imagery dataset constructed from Sentinel-2 images. We evaluate over 3 months of data from June to August 2018 over the States of Colorado, Nevada, and California. Our dataset contains 3.15TB of Sentinel-2 images with over 6000 unique captures. When split into quadtiles of length 11, we produce ∼308,000 image partitions.

B. Utility of Conditional Metadata Information

We evaluate the utility of using geospatial metadata as geospatial information for our super-resolution GAN's in Table I. As expected, we see that incorporation of metadata channel during training benefits the output image quality, as demonstrated by the higher PSNR.

C. Model Convergence Speed

We evaluate the speed of convergence for our super-resolution models (Fig.4a). We demonstrate the three phases of training i.e. for upscale factors 2, 4, and 8 (shaded using the colors green, red, and purple respectively) and show the progression of both training and validation error (bold and dotted curves respectively). In both basic and partial super-resolution, we see the error start high and then quickly reduce before settling into a stable region.



(a) Progression of Training and Validation Errors (b) Model Performance vs Upscale Factor and Zoom Level

Fig. (4)

D. Model Accuracy Evaluation and Comparison

We have used a variety of metrics, such as RMSE, MAE, and SSIM to evaluate the quality of our SR images. In this experiment, we use PSNR as a measure of image quality to compare the accuracy for EvokeNet models. As shown in Fig.4b, the quality of image decreases with increasing upscale factor. This is due to the increased complexity of the problem.

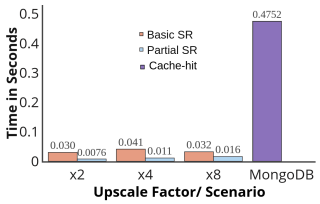


Fig. (5) Client-side Average Image Reconstruction Time vs Cache Fetch Time

We also evaluate image quality on the basis of the zoom level of the input image and find that the image quality improves slightly when the input image is more zoomed-in. Since images with finer resolutions tend to reveal more details and topographical features, the overall accuracy improves when upscaling, especially at the higher end of the zoom spectrum.

Fig.4b contrasts the super-resolution images quality generated by EvokeNet modules against that of **SRGAN** [8] trained over the same satellite dataset. The results clearly demonstrate an improved quality in the output SR image for EvokeNet, especially for higher upscale factors. Our experiments showed improvements in average PSNR's ranging from ∼9.5-13%.

Upscale Factor	#Epochs	Time
x2	3	152.22
x4	2	118.45
x8	2	193.47

TABLE (II) Model Retraining Times in Light of New Data

Method Used	Case	Refresh Time (Seconds)					
		25%		50%		100%	
		CPU	GPU	CPU	GPU	CPU	GPU
Basic SR	x2	1.35	*0.037*	3.02	*0.039*	6.05	*0.042*
	x4	1.27	*0.048*	2.39	*0.050*	4.61	*0.057*
	x8	0.82	*0.054*	1.85	*0.055*	3.38	*0.06*
Fetch Request	z=11	46.08		81.26		139.18	
	z=12	12.23		25.42		40.93	
	z=13	6.29		12.97		17.84	

TABLE (III) Comparison of Client-side Performance of EvokeNet Against Traditional Spatiotemporal Server Query

E. Model Re-Training on New Incoming Data

Since our underlying data store is dynamic, we have evaluated the speed of convergence of our EvokeNet models in light of fresh incoming data. To simulate such a scenario, we have used a pre-trained model only with tiles from California and Nevada. We have then re-trained the model on the new dataset that includes all three states (including Colorado). We can clearly see from Table II, and comparing the results to Fig.4a, that updating the stale model takes comparatively lesser number of epochs; this demonstrates the stability of the model update procedure.

F. Image Reconstruction vs Cached Data Retrieval

In Fig.5, demonstrates the average client-side latency to generate a high resolution image in-house. We have evaluated the average time taken by the EvokeNet models to reconstruct(with GPU acceleration) a set of 100 images with a batch-size of 16 for varying upscale factors. We can see latency in the order of milliseconds for all upscale factors for both EvokeNet super-resolution modules.

Fig.5 also compares the average latency to fetch a similar set of images, cached in a distributed MongoDB [31] database (based on previous queries) on our cluster. We can see that there is a significant improvement in latency using GAN-based *in-situ* image reconstruction, compared to a fetch from a server-side cache, which can be attributed to data movement overheads.

In Table III we contrast the time taken to refresh the screen/visualization grid in scenarios involving varying fractions of in-situ super-resolution (both with and without GPU

acceleration) and server-side fetch through spatiotemporal queries. An OLAP zoom operation of scale x2, x4, x8 would require us to perform super-resolution on a set of 64, 16, and 4 tiles respectively, since higher upscaled tiles have higher pixel coverage. Hence, although the upscaling operations get incrementally more complex, the number of tiles involved follow a descending order. This explains the results of Table III, which shows that in case of no GPU assistance, total time taken to reconstruct the same fraction of the screen x2 is higher than that for x4 and x8, but can be explained once we consider the decreasing number of tiles are involved.

Table III illustrates that the times taken to physically fetch the corresponding amount of tiles from the distributed file system are significantly higher than the case of super-resolution. A full refresh of the window at 2x, 4x, and 8x through physical fetch takes ∼23x, 8, 8x, and 5.3x longer than reconstruction with the basic SR module. With the partial SR, the speed-up was 69.5x, 125x, and 5.1x respectively. In the case of utilizing GPU acceleration (see sectionV-A) on the client machine, the same improvements in time range from ∼3313x to 297x and 6627x to 540x respectively (Table III).

VI. Conclusion

We described our framework, Evoke, for interactive client-side visualizations by re-using historical in-memory low-resolution data. Our system is agnostic to the back-end data-store and the in-memory data cache. **RQ1:** Evoke reduces data transfers with the server by identifying and repurposing memory-resident, low-resolution image objects in the cache and using a pre-trained GAN-based super-resolution model. **RQ2:** Evoke is space-efficient and can be loaded both in the client's RAM or GPU (when available) and can perform super-resolution computations in the order of milliseconds (Sec. V-F).

VII. Acknowledgement

This research was supported by the National Science Foundation [OAC-1931363, ACI-1553685] , the National Institute of Food and Agriculture [COL0-FACT-2019] , and a Cochran Family Professorship.

References

[1] *Tableau Desktop*, 2019, https://www.tableau.com/products/desktop.

[2] V. Sharma, "Getting started with kibana," in *Beginning Elastic Stack*. Springer, 2016, pp. 29–44.

[3] R. Li, J. Fan, X. Wang, Z. Zhou, and H. Wu, "Distributed cache replacement method for geospatial data using spatiotemporal locality-based sequence," *Geo-spatial Information Science*, vol. 18, no. 4, pp. 171–182, 2015.

[4] R. Li, W. Feng, H. Wu, and Q. Huang, "A replication strategy for a distributed high-speed caching system based on spatiotemporal access patterns of geospatial data," *Computers, Environment and Urban Systems*, vol. 61, pp. 163–171, 2017.

[5] R. Li, Y. Zhang, Z. Xu, and H. Wu, "A load-balancing method for network giss in a heterogeneous cluster-based system using access density," *Future Generation Computer Systems*, vol. 29, no. 2, pp. 528–535, 2013.

[6] S. Paul and Z. Fei, "Distributed caching with centralized control," *Computer Communications*, vol. 24, no. 2, pp. 256–268, 2001.

[7] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[8] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. 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*, 2017, pp. 4681–4690.

[9] (2014, April) Visualize large complex data with local tile layers in bing maps windows store apps (c). [Online]. Available: https://blogs.bing.com/Developers-Blog/2014-04/Visualize-Large-Complex-Data-with-Local-Tile-Layers-in-Bing-Maps-Windows-Store-Apps-(C-)

[10] (2019, December) Tiled web map. [Online]. Available: https://en.wikipedia.org/wiki/Tiled_web_map

[11] A. Berson, "Data warehousing, data mining and ol ap, mc grawill./a," *Berson, S. Snith*, 1997.

[12] R. Li, Y. Zhang, Z. Xu, and H. Wu, "A load-balancing method for network giss in a heterogeneous cluster-based system using access density," *Future Generation Computer Systems*, vol. 29, no. 2, pp. 528–535, 2013.

[13] Z. Liu, B. Jiang, and J. Heer, "immens: Real-time visual querying of big data," in *Computer Graphics Forum*, vol. 32, no. 3pt4. Wiley Online Library, 2013, pp. 421–430.

[14] (2018, November) Quadtiles. [Online]. Available: https://wiki.openstreetmap.org/wiki/QuadTiles

[15] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[18] M. S. Sajjadi, B. Scholkopf, and M. Hirsch, "Enhancenet: Single image super-resolution through automated texture synthesis," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4491–4500.

[19] K. Jiang, Z. Wang, P. Yi, and J. Jiang, "A progressively enhanced network for video satellite imagery superresolution," *IEEE Signal Processing Letters*, vol. 25, no. 11, pp. 1630–1634, 2018.

[20] J. Pan, S. Liu, D. Sun, J. Zhang, Y. Liu, J. Ren, Z. Li, J. Tang, H. Lu, Y.-W. Tai *et al.*, "Learning dual convolutional neural networks for low-level vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3070–3079.

[21] Y. Wang, F. Perazzi, B. McWilliams, A. Sorkine-Hornung, O. Sorkine-Hornung, and C. Schroers, "A fully progressive approach to single-image super-resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 864–873.

[22] S. Mitra, P. Khandelwal, S. Pallickara, and S. L. Pallickara, "Stash: Fast hierarchical aggregation queries for effective visual spatiotemporal explorations," in *2019 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2019, pp. 1–11.

[23] (2016) National land cover database 2016 (nlcd2016) legend. [Online]. Available: https://www.mrlc.gov/data/legends/national-land-cover-database-2016-nlcd2016-legend

[24] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[25] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[26] U. Demir and G. Unal, "Patch-based image inpainting with generative adversarial networks," *arXiv preprint arXiv:1803.07422*, 2018.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[28] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on networking*, vol. 11, no. 1, pp. 17–32, 2003.

[29] W. Falcon, "Pytorch lightning," *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, vol. 3, 2019.

[30] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[31] K. Banker, *MongoDB in action*. Manning Publications Co., 2011.