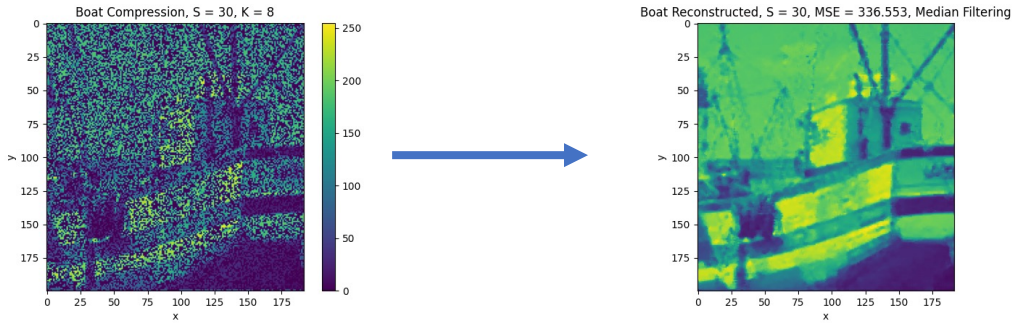
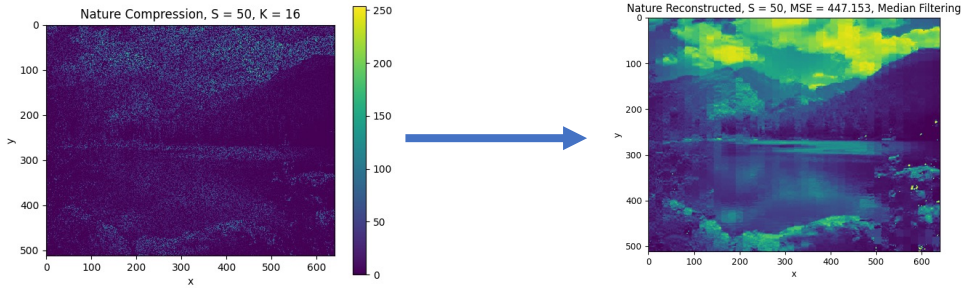
Mini Project 1 – Image Recovery

Ben Matz

March 1, 2023

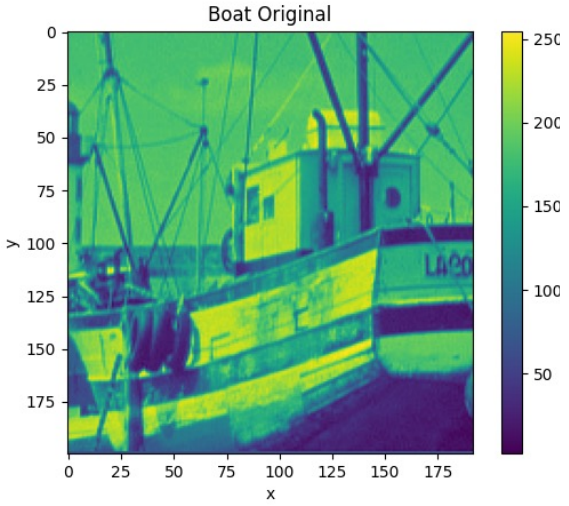
ECE580

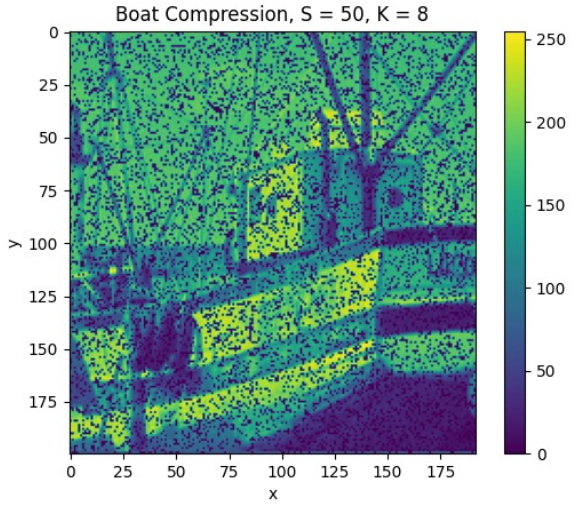


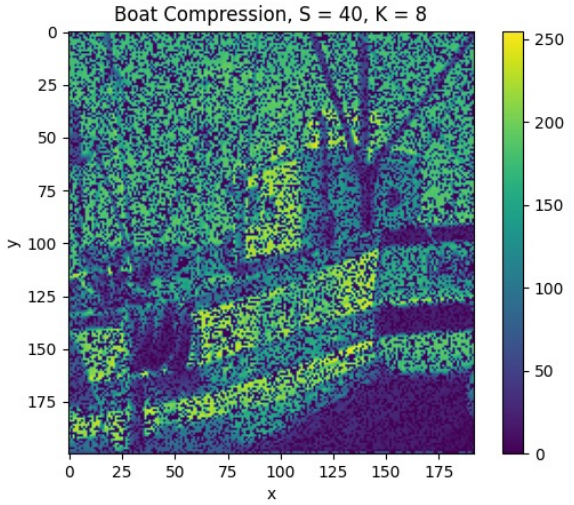


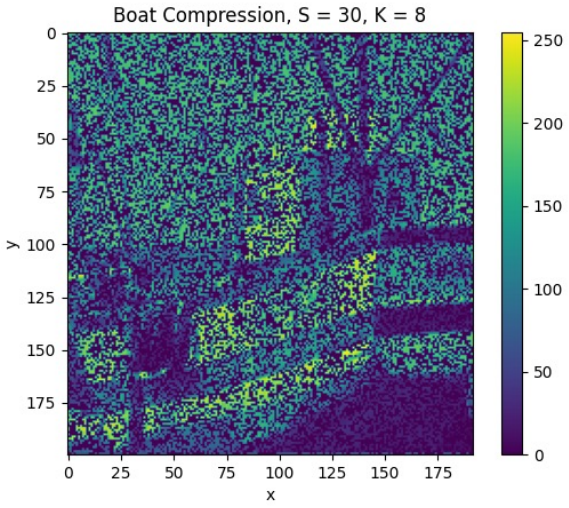
Boat Compressions

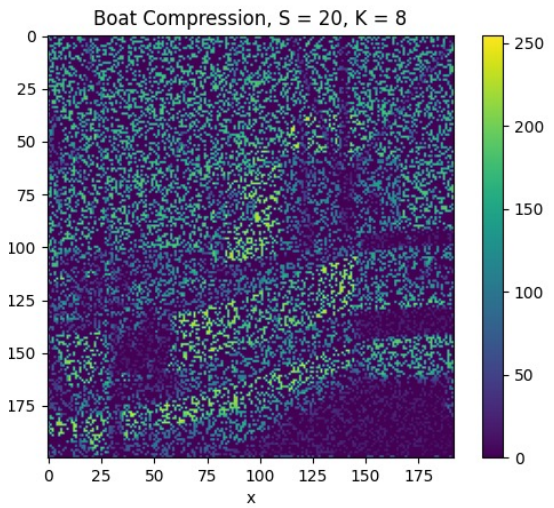
Pictured to the right are the original boat image and its representation at different levels of simulated compression. S is the number of pixels that are maintained for every K x K image block in the overall picture. So, the proportion of sensed pixels in a given compressed picture is given by the equation $p = \frac{S}{K^2}$.

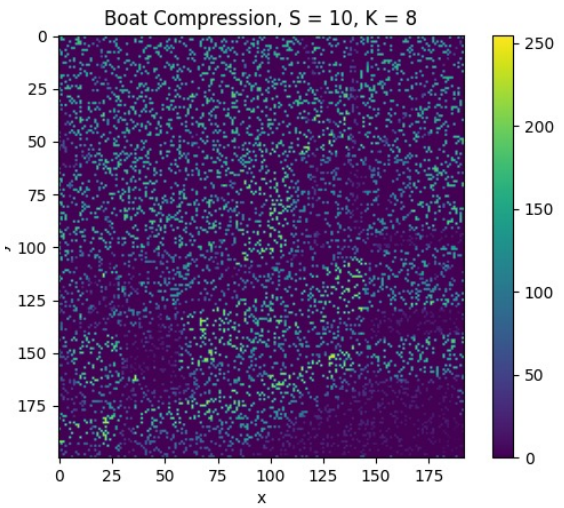






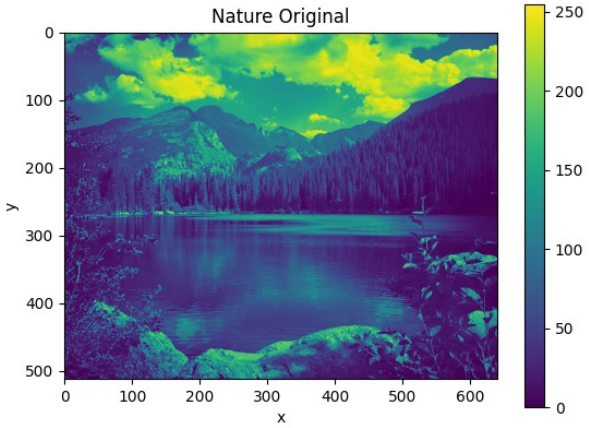


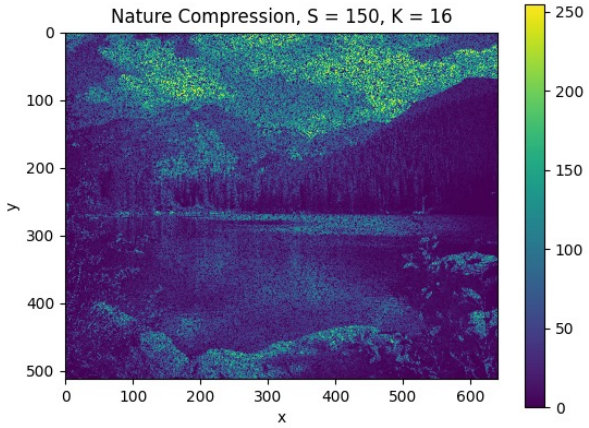


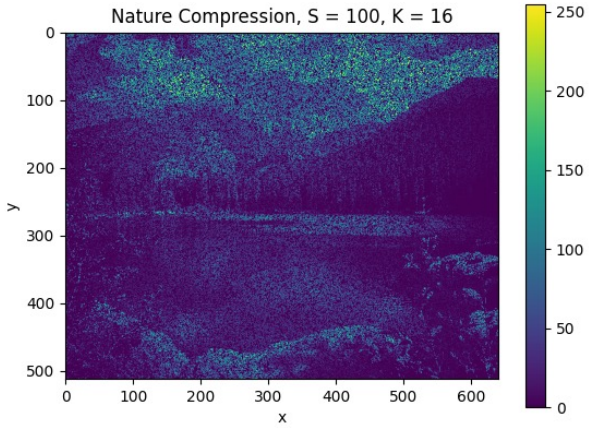


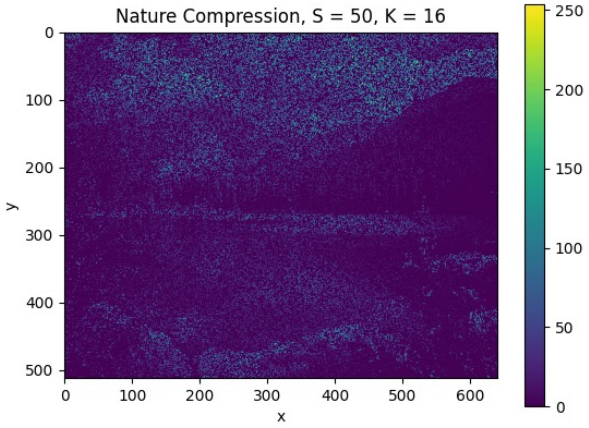
Nature Compressions

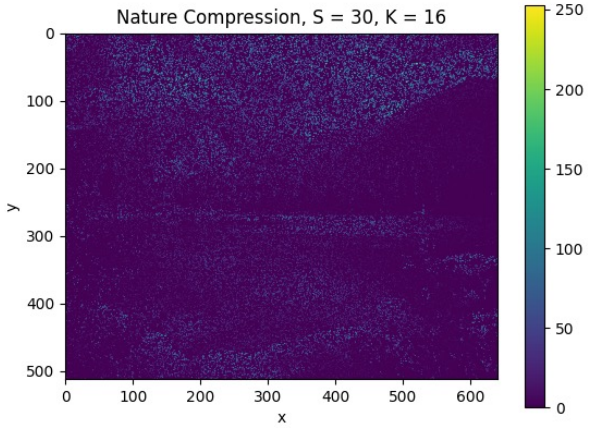
Pictured to the right are the original nature image and its representation at different levels of simulated compression. S is the number of pixels that are maintained for every K x K image block in the overall picture. So, the proportion of sensed pixels in a given compressed picture is given by the equation $p = \frac{S}{K^2}$.

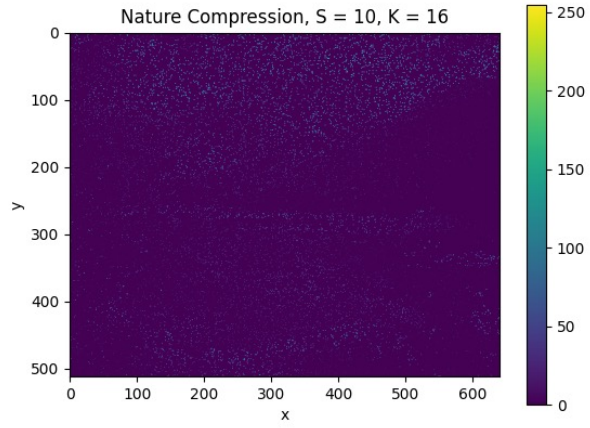












Basis Vector Matrices

To the right are visualizations of the Discrete Cosine Transform (DCT) (Ahmed, 1974) basis vector matrix for K = 8 and K = 16. On the vertical axis are the spatial locations (x, y) and on the horizontal axis are the spatial frequencies (u, v). The matrix is comprised of individual basis vectors which are given by the equation:

$$\vec{basis}_{(u,v)} = \alpha_u \cdot \beta_v \cdot \cos\frac{\pi(2x-1)(u-1)}{2P} \cdot \cos\frac{\pi(2y-1)(v-1)}{2Q}$$

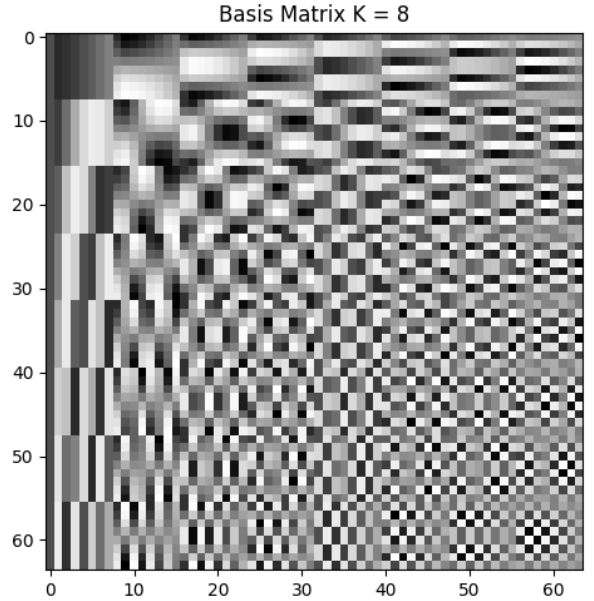
where P is the number of pixels horizontally, Q is the number of pixels vertically,

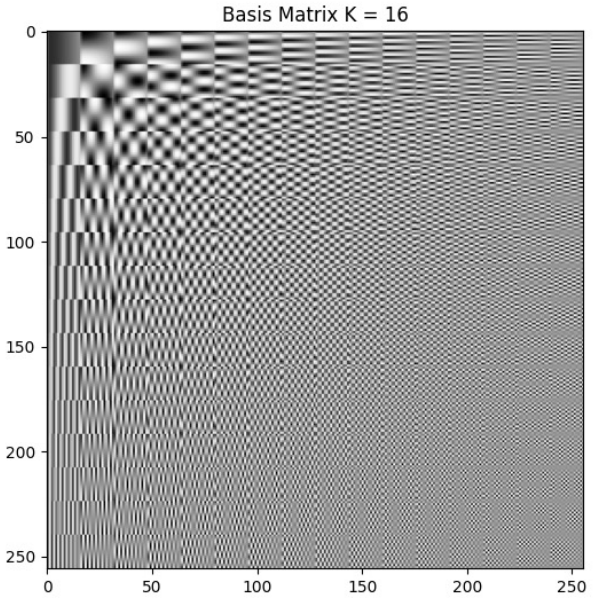
$$\alpha_u = \begin{cases} \sqrt{1/P} & \text{if } u = 1 \\ \sqrt{2/P} & \text{if } 2 \leq u \leq P \end{cases}$$

and

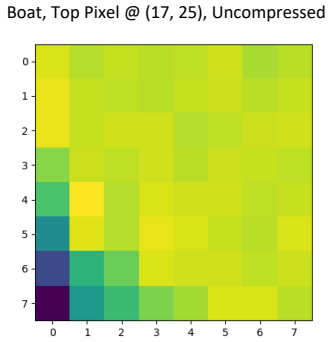
$$\beta_v = \begin{cases} \sqrt{1/Q} & \text{if } u = 1 \\ \sqrt{2/Q} & \text{if } 2 \leq u \leq Q \end{cases}$$

Each vector is the rasterization of the above equations evaluated at all x and y. Together, this set of vectors comprises the basis for the reconstructed pixels. A linear combination of these vectors will be calculated to best represent individual image blocks.





Checkpoint 2 - Fishing Boat 8x8, Top Pixel @ (17, 25)



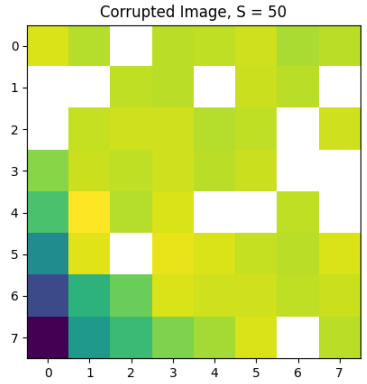
Above is the original image block (random 8 x 8 chunk) from the boat picture. To the right are compressions at two different compression levels. Then, using the regularization parameter selected in cross validation, we estimate the missing pixels to “reconstruct” the compressed images (column two). In the third column, we have MSE as a function of the log(regularization parameter). We will select the parameter that minimizes this MSE.

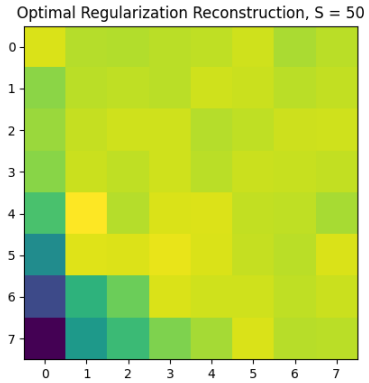
Corruption

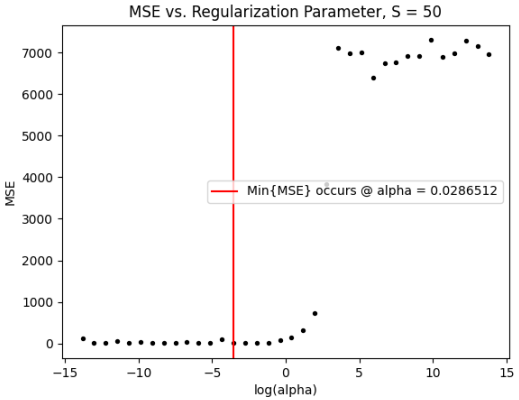
Best Reconstruction

MSE vs. alpha

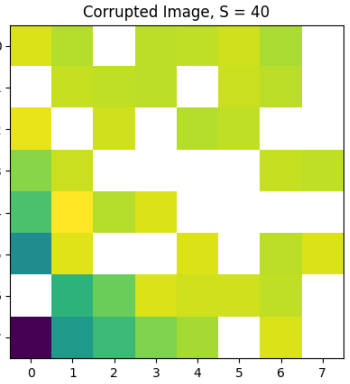
S = 50

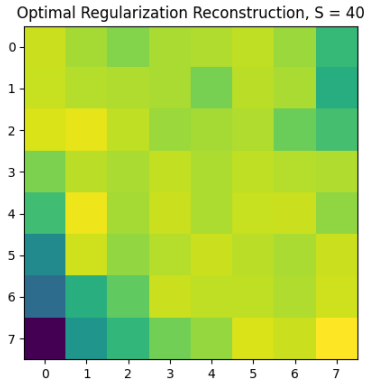


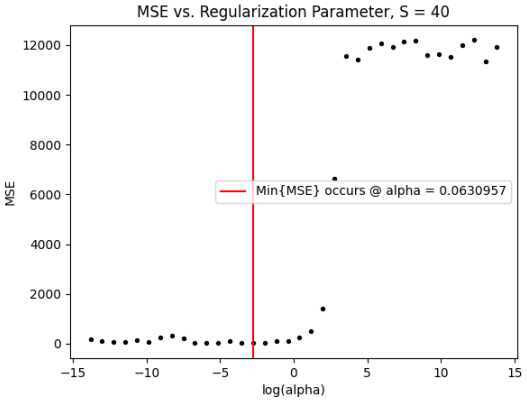




S = 30

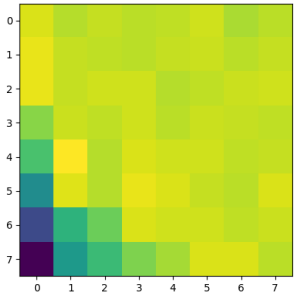






Checkpoint 2 - Fishing Boat 8x8, Top Pixel @ (17, 25) Cont.

Boat, Top Pixel @ (17, 25), Uncompressed



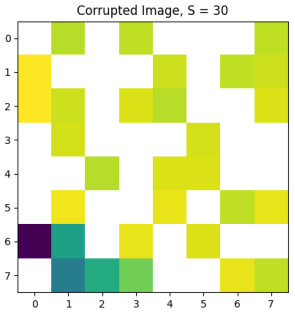
Above is the original image block (random 8 x 8 chunk) from the boat. To the right are compressions at three different compression levels. Then, using the regularization parameter selected in cross validation, we estimate the missing pixels to “reconstruct” the compressed images (column two). In the third column, we have MSE as a function of the log(regularization parameter). We will select the parameter that minimizes this MSE.

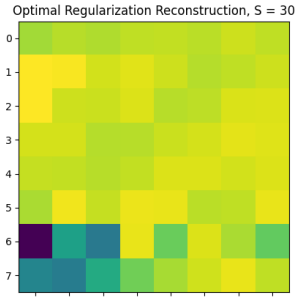
Corruption

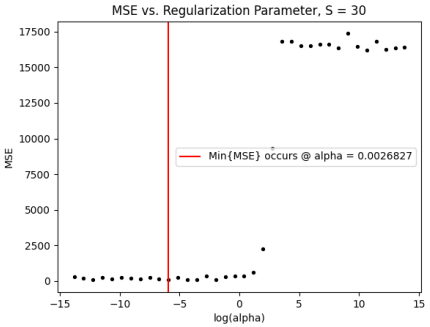
Best Reconstruction

MSE vs. alpha

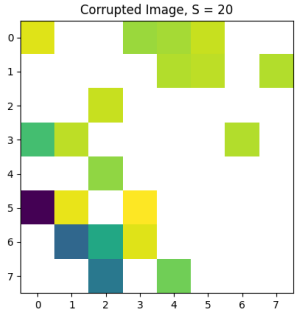
S = 30

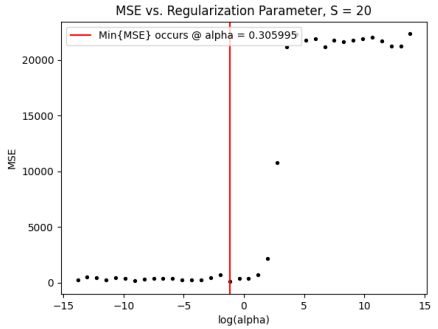




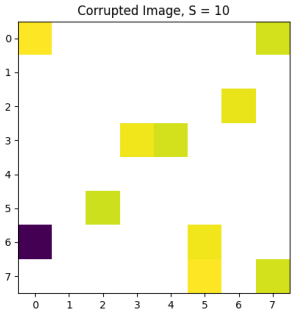


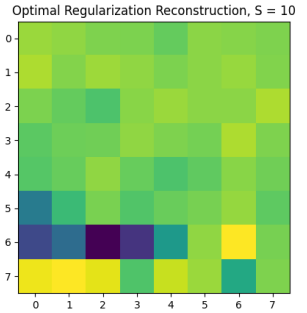
S = 20

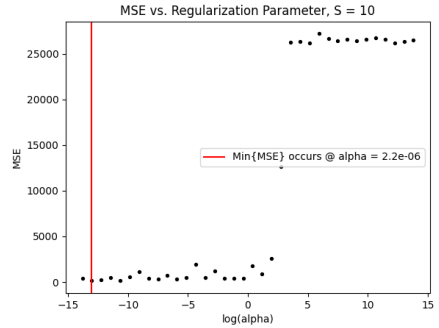




S = 10

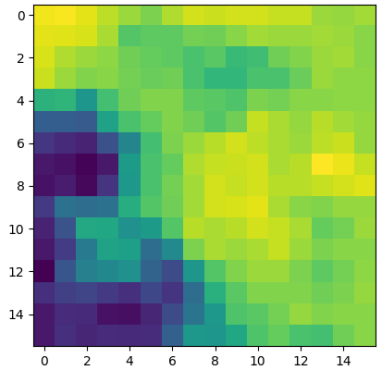






Checkpoint 2 - Nature 16x16, Top Pixel @ (17, 193)

Nature, Top Pixel @ (17, 193), Uncompressed



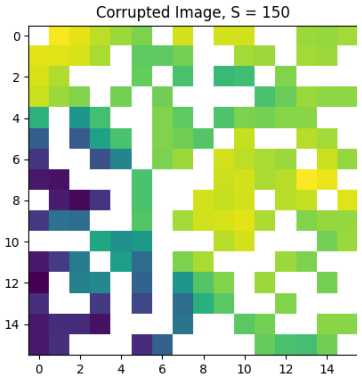
Above is the original image block (random 16 x 16 chunk) from the nature photo. To the right are compressions at two different compression levels. Then, using the regularization parameter selected in cross validation, we estimate the missing pixels to “reconstruct” the compressed images (column two). In the third column, we have MSE as a function of the log(regularization parameter). We will select the parameter that minimizes this MSE.

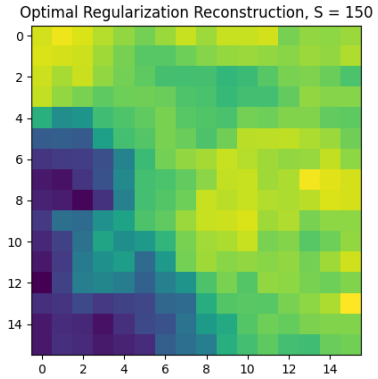
Corruption

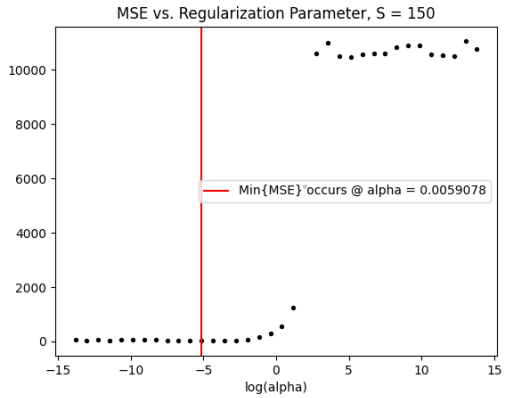
Best Reconstruction

MSE vs. alpha

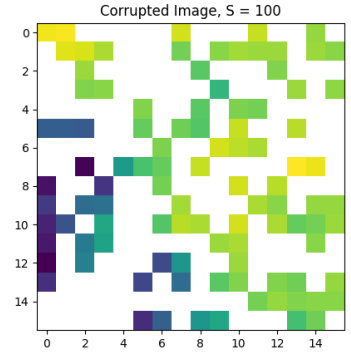
S = 150

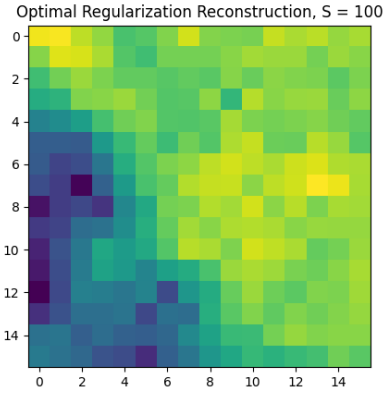


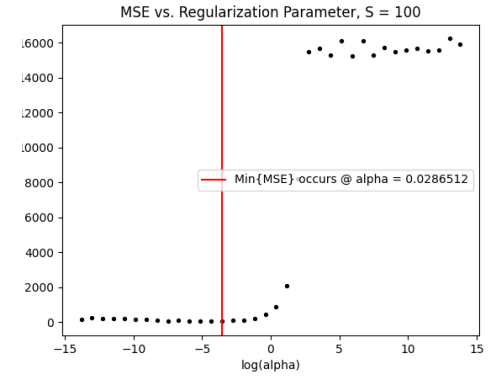




S = 100

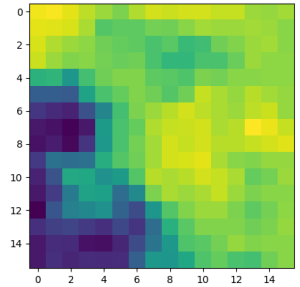




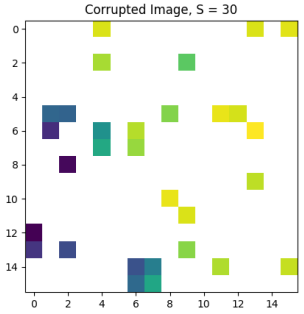
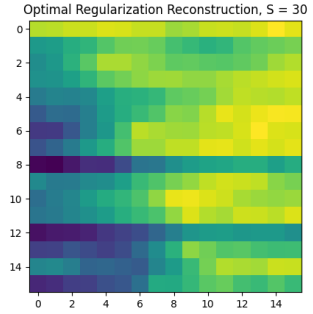
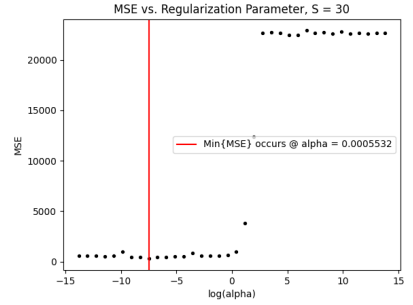
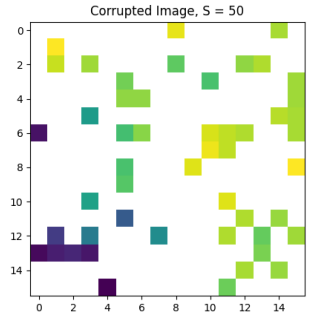
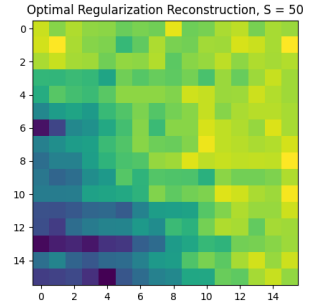
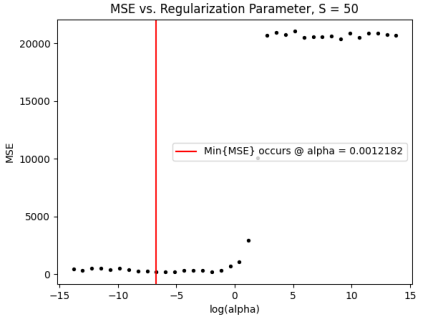
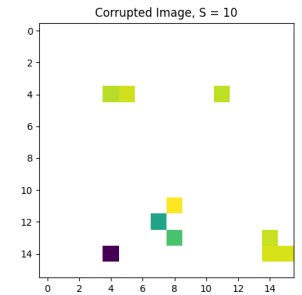
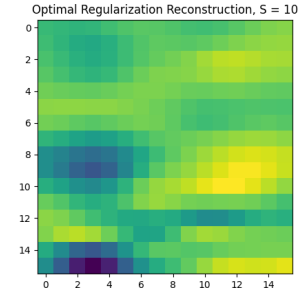
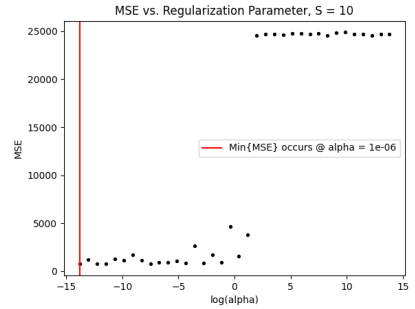


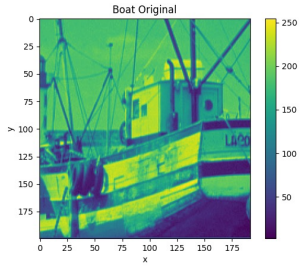
Checkpoint 2 - Nature 16x16, Top Pixel @ (17, 193) Cont.

Nature, Top Pixel @ (17, 193), Uncompressed



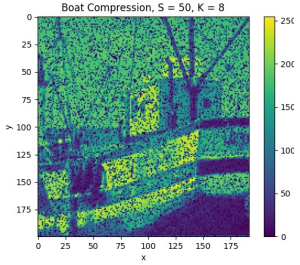
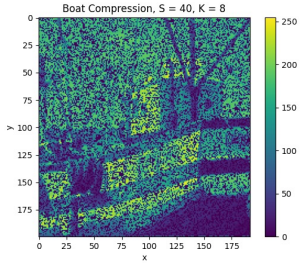
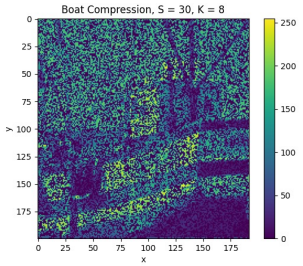
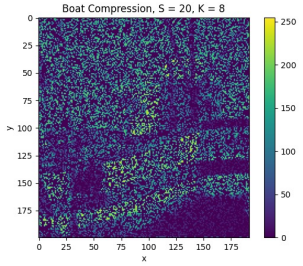
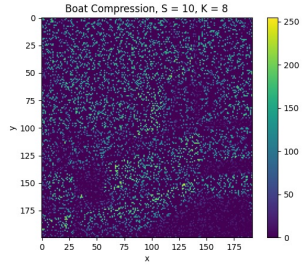
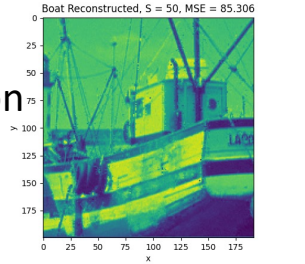
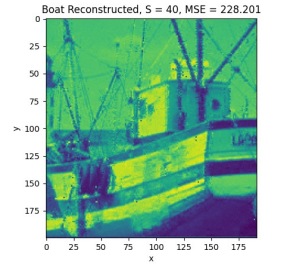
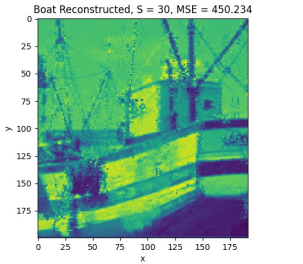
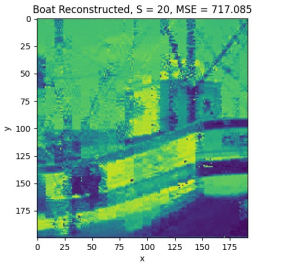
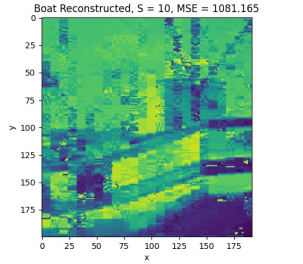
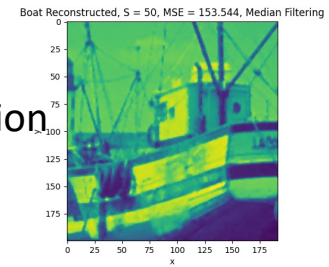
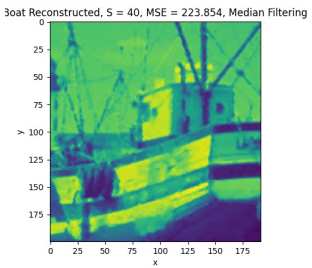
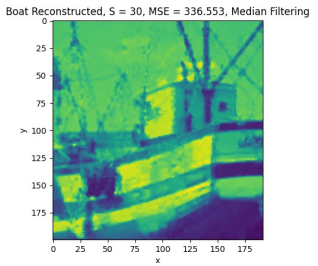
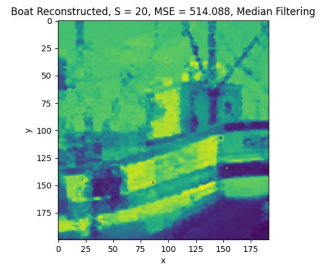
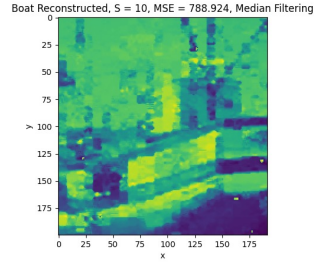
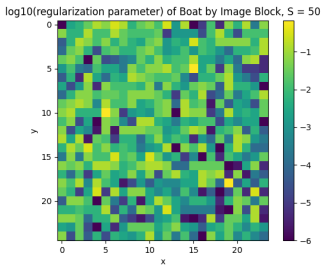
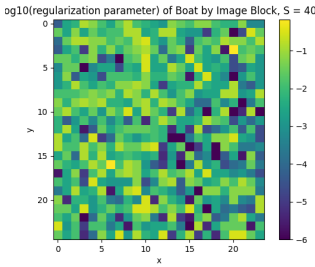
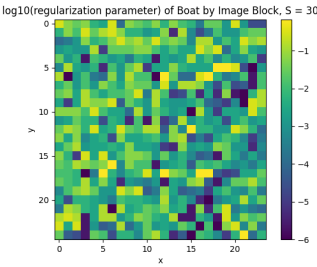
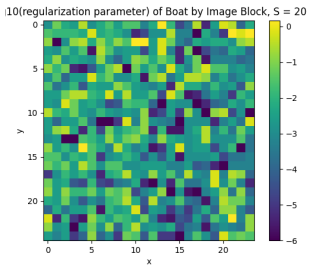
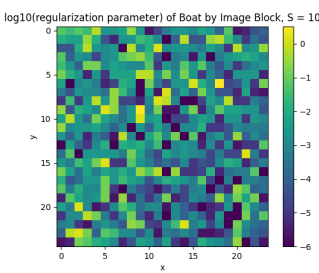
Above is the original image block (random 16 x 16 chunk) from the nature photo. To the right are compressions at three different compression levels. Then, using the regularization parameter selected in cross validation, we estimate the missing pixels to “reconstruct” the compressed images (column two). In the third column, we have MSE as a function of the log(regularization parameter). We will select the parameter that minimizes this MSE.

	Corruption	Best Reconstruction	MSE vs. alpha
S = 50			
S = 30			
S = 10			



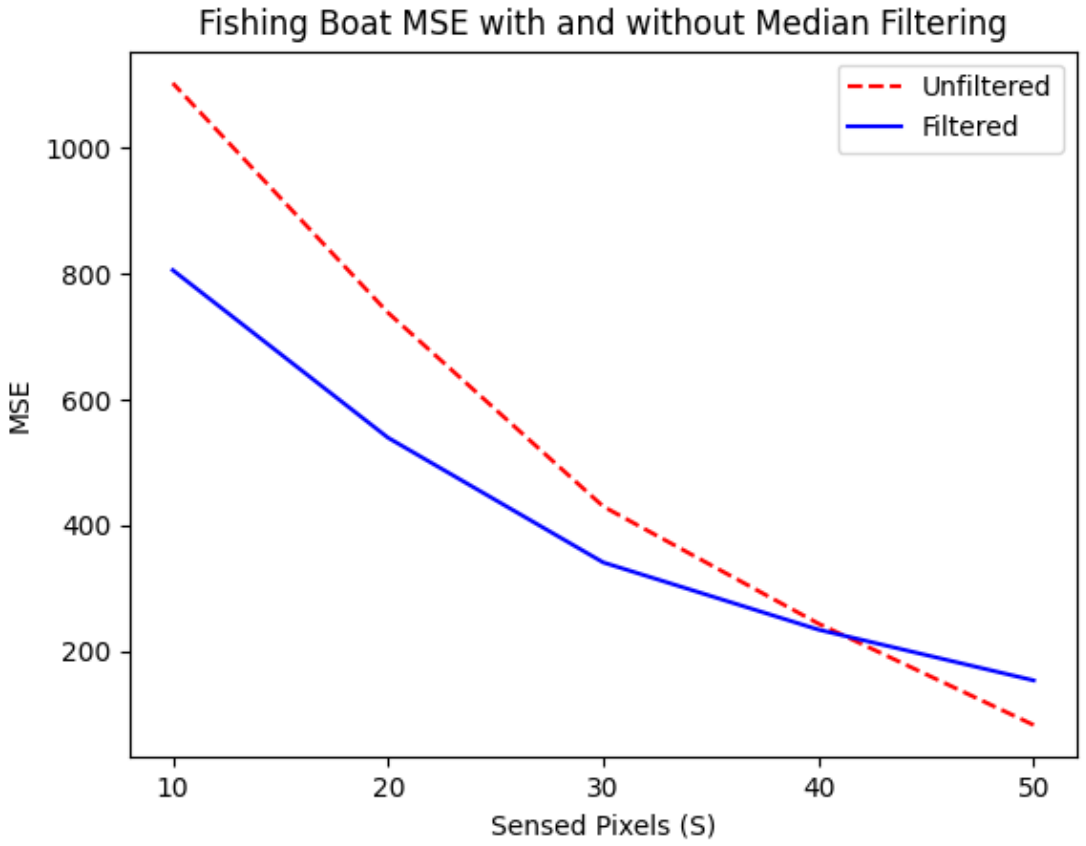
Fishing Boat Full Reconstructions (K = 8)

In the top left corner is the original boat image. To the right is a grid of figures where the columns align to the number of sensed pixels in each 8 x 8 block. The first row shows a sample compression at that sensed pixel level. The second shows a reconstruction without any filter. The third shows the median filter applied to the reconstructed image, smoothing it out. The fourth shows what the chosen regularization parameter was for each image block in log10 space.

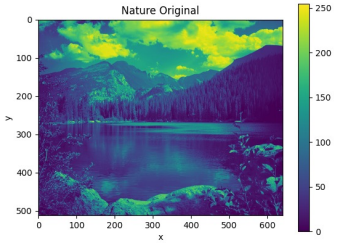
	S = 50	S = 40	S = 30	S = 20	S = 10
Compression					
Reconstruction without Filter					
Reconstruction with Filter					
Alpha Grid					

Boat MSE

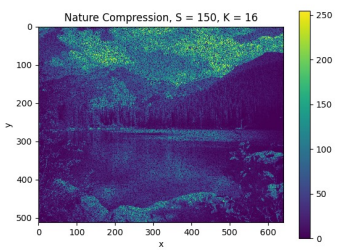
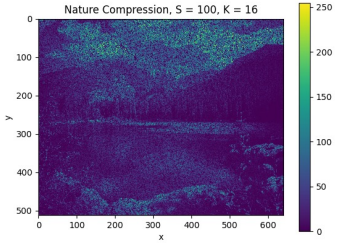
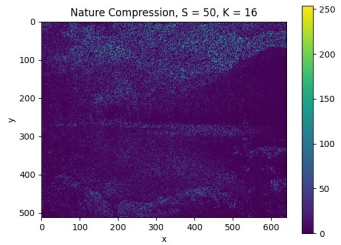
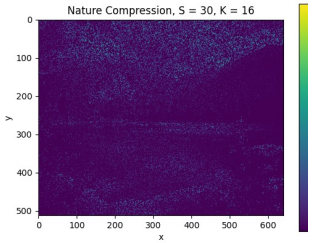
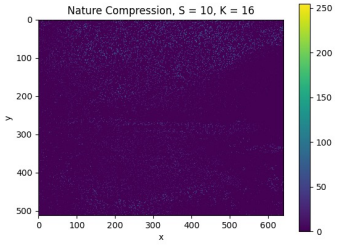
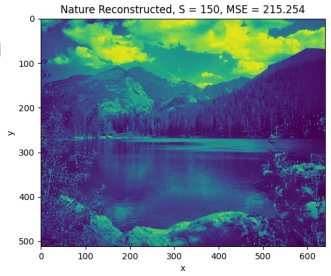
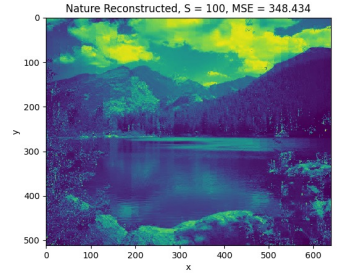
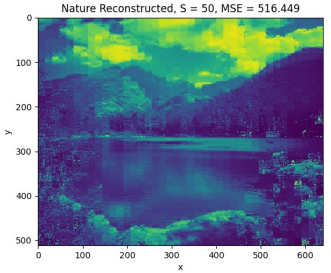
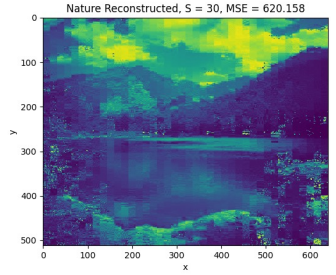
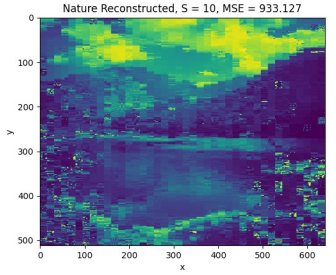
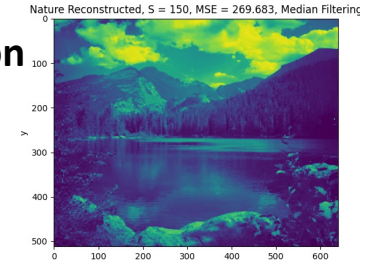
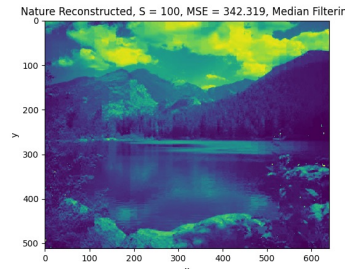
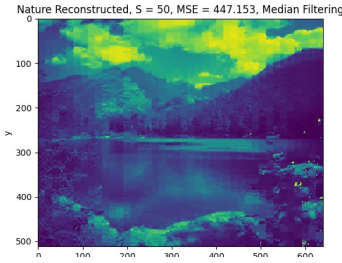
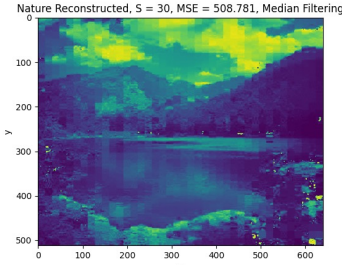
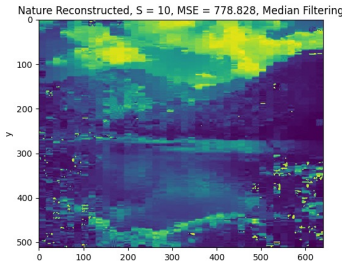
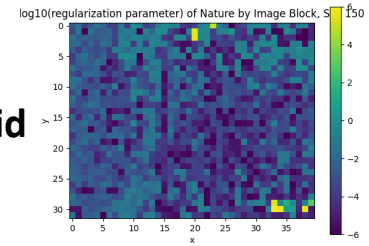
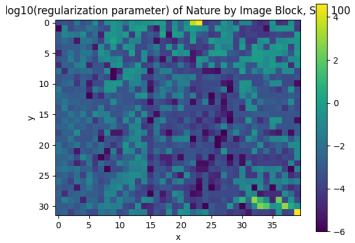
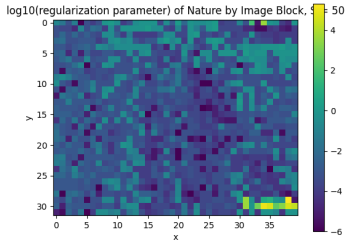
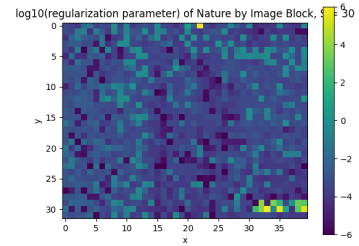
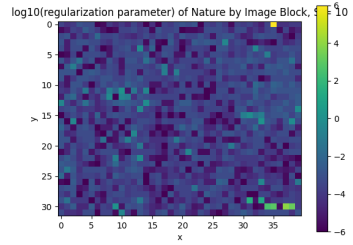
To the right is a graph of MSE as a function of sensed pixels. This is shown for both the filtered and unfiltered reconstructions of the compressed boat image. As one expects, MSE decreases as the corrupted image becomes less compressed and S increases. This makes sense because we have a better idea of unsensed pixels if there's more context around them. Also, for S < 40, the filtered image has lower MSE as filtering smooths out some of the jumps in pixel values. However, at S > 40 we see that the unfiltered image is better in terms of MSE. This is because when we apply the filter, we alter the unsensed AND the sensed pixels. When there are many sensed pixels (like S = 50), we can actually worsen MSE as we replace known values with filtered ones.



Nature Full Reconstructions (K = 16)

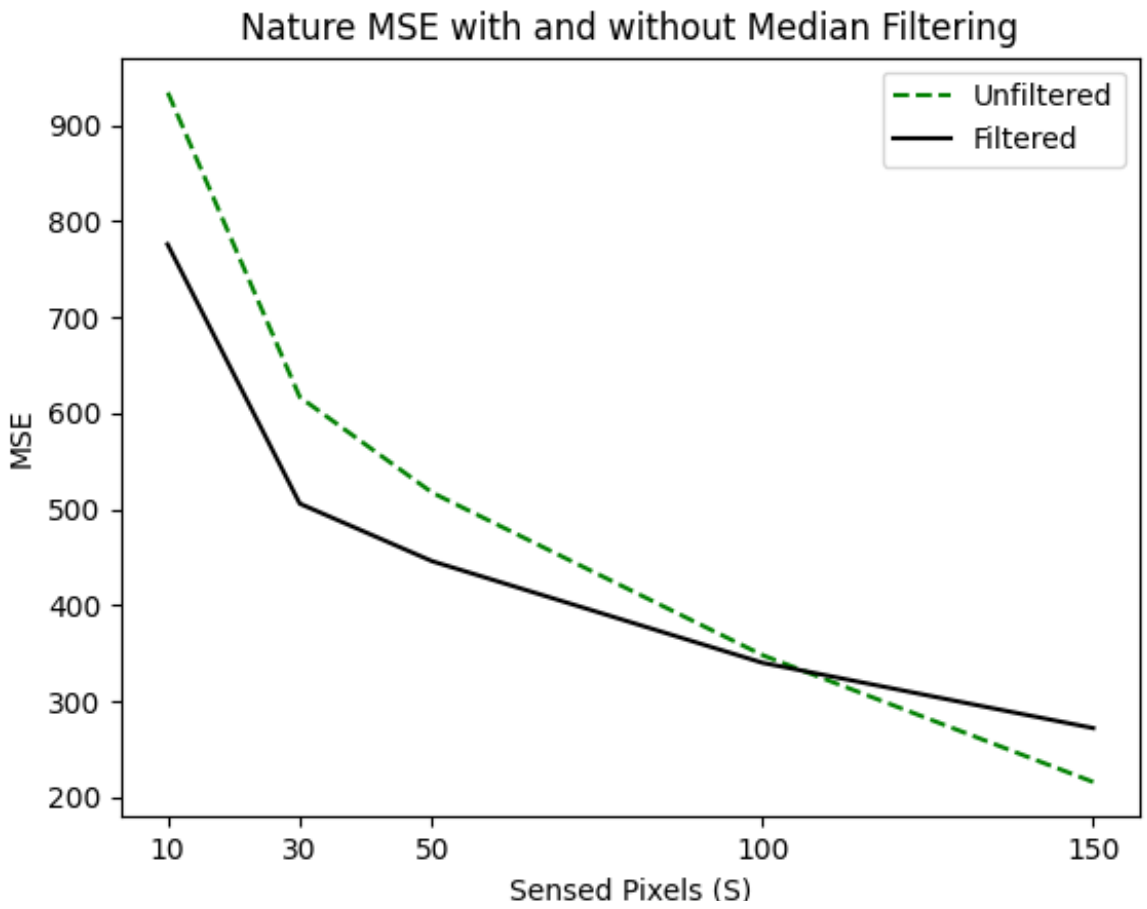


In the top left corner is the original nature image. To the right is a grid of figures where the columns align to the number of sensed pixels in each 16 x 16 block. The first row shows a sample compression at that sensed pixel level. The second shows a reconstruction without any filter. The third shows the median filter applied to the reconstructed image, smoothing it out. The fourth shows what the chosen regularization parameter was for each image block in log10 space.

	S = 150	S = 100	S = 50	S = 30	S = 10
Compression					
Reconstruction without Filter					
Reconstruction with Filter					
Alpha Grid					

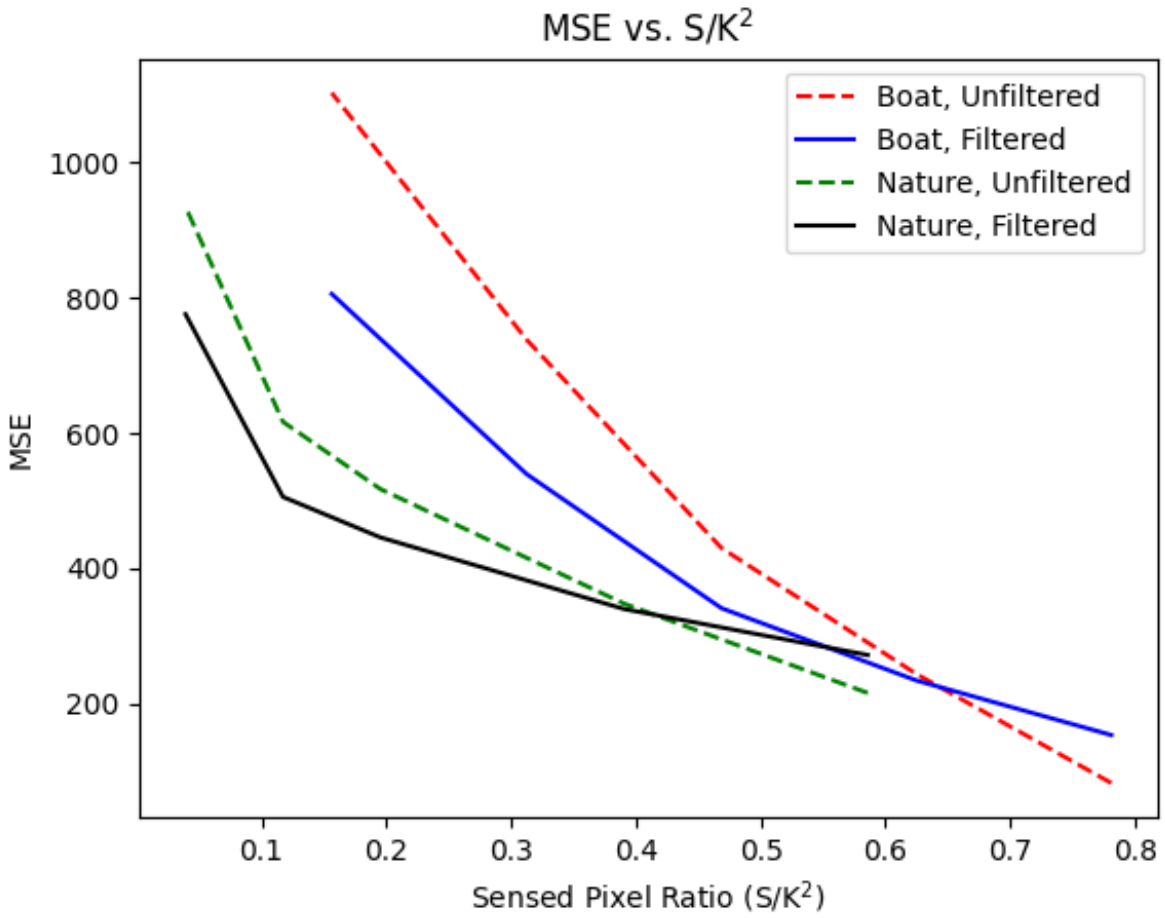
Nature MSE

To the right is a graph of MSE as a function of sensed pixels. This is shown for both the filtered and unfiltered reconstructions of the compressed nature image. As one expects, MSE decreases as the corrupted image becomes less compressed and S increases. This makes sense because we have a better idea of unsensed pixels if there's more context around them. Also, for S < 100, the filtered image has lower MSE as filtering smooths out some of the jumps in pixel values. However, at S > 100 we see that the unfiltered image is better in terms of MSE. This is because when we apply the filter, we alter the unsensed AND the sensed pixels. When there are many sensed pixels (like S = 50), we can actually worsen MSE as we replace known values with filtered ones.



MSE vs. S/K^2 Ratio

To the right is a graph of MSE as a function of sensed pixel ratio (percent of sensed pixels). There are four lines: unfiltered and filtered for both the boat and nature images. As one expects, MSE decreases as the corrupted image becomes less compressed (S increases). This makes sense because we have a better idea of unsensed pixels if there's more information surrounding them. As with the trend we've seen in previous slides, there is a certain threshold at which the filtering worsens the MSE. Generally, the boat predictions were worse than that of the nature predictions at the same ratio which indicates that it was a tougher image to recover.

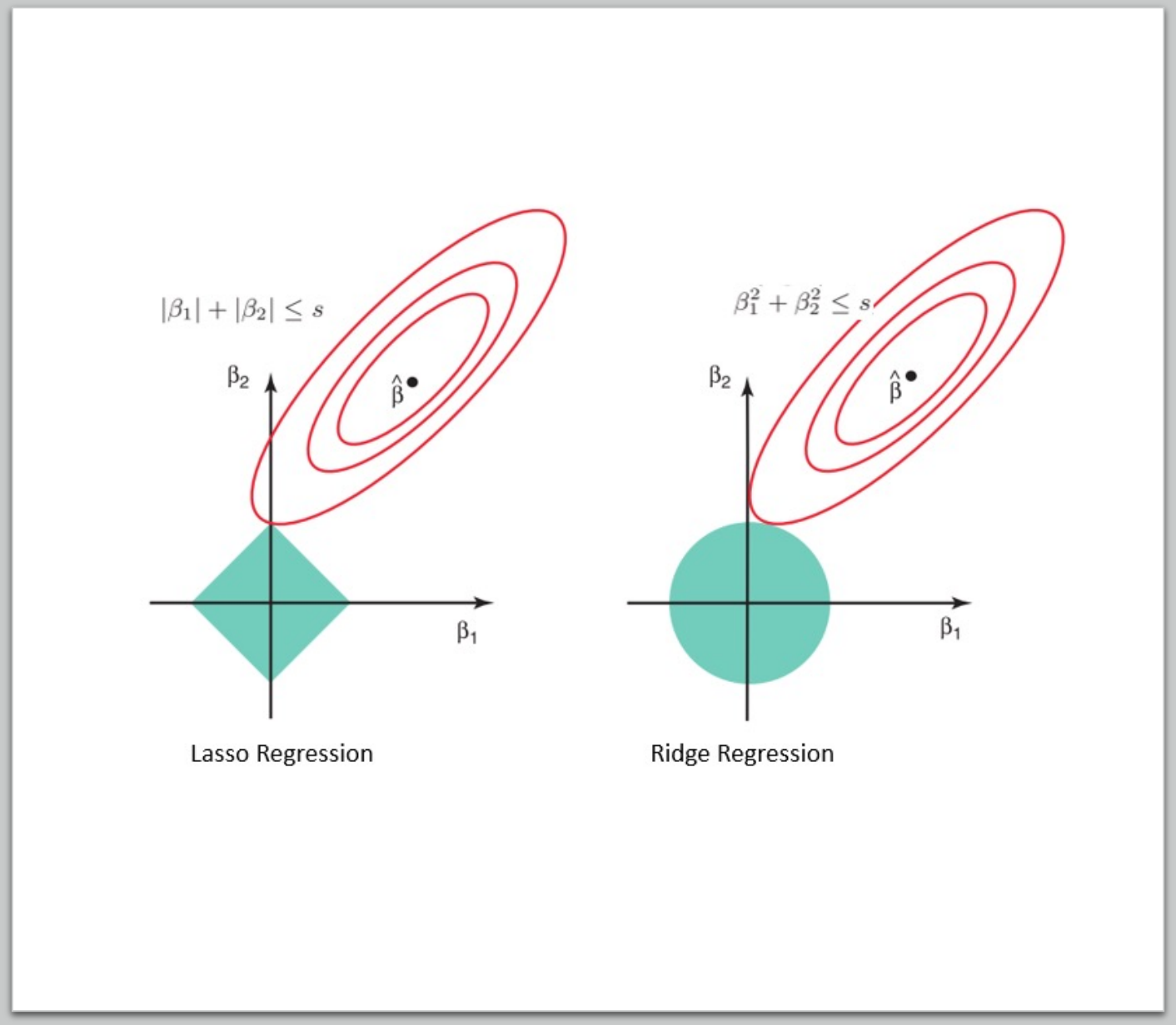


Why Underdetermined Linear Systems?

We can think of our image as represented by an array of pixels P that can be decomposed into a basis vector matrix B and some weights w. Here, $P = Bw$ and $w = B^{-1}P$. However, when the image is corrupted and some of the pixels are unknown, we lose rows of P and B, and B becomes uninvertible. This means we get an underdetermined linear system. So,. in order to solve for w, we need to use the types of techniques that are used to solve underdetermined systems. Once we find w, we will be able to restore the pixels in our image with a simple matrix multiplication $P = Bw$.

Why LASSO?

LASSO (L1-norm regularization) is preferable to Ridge (L2-norm regularization) it can lead to ”feature selection” (Bhattacharyya, 2018) This forces our weights vector to be sparse (ie. sends weights to zero rather than near zero as demonstrated to the right). This is the constraint that enables us to solve our underdetermined linear system.



Source: https://www.datacamp.com/tutorial/tutorial-lasso-ridge-regression

How LASSO (Tibshirani, 1996)?

LASSO aims to minimize the sum of the error and the magnitude of the weights (times a regularization constant) when determining appropriate weights. The equation to the right is the exact equation that is minimized in sklearn's LASSO package that I used for this project. Once the weights are generated, we simply multiply by the original basis vector matrix to obtain the predictions for all pixels. But we only use these predictions for the unsensed pixels because we already know the sensed ones and don't want to lose that information. I decided to leave the "constant" basis vector in the matrix as a test to see if the image reconstruction would work this way. It did work and so there was nothing more for me to do with the constant. However, the main issue one can run into is that regularization constrains the intercept to a small value when often a large one is needed for an accurate model fit.

$$\min_w \frac{1}{2n_{\text{samples}}} ||Xw - y||_2^2 + \alpha ||w||_1$$

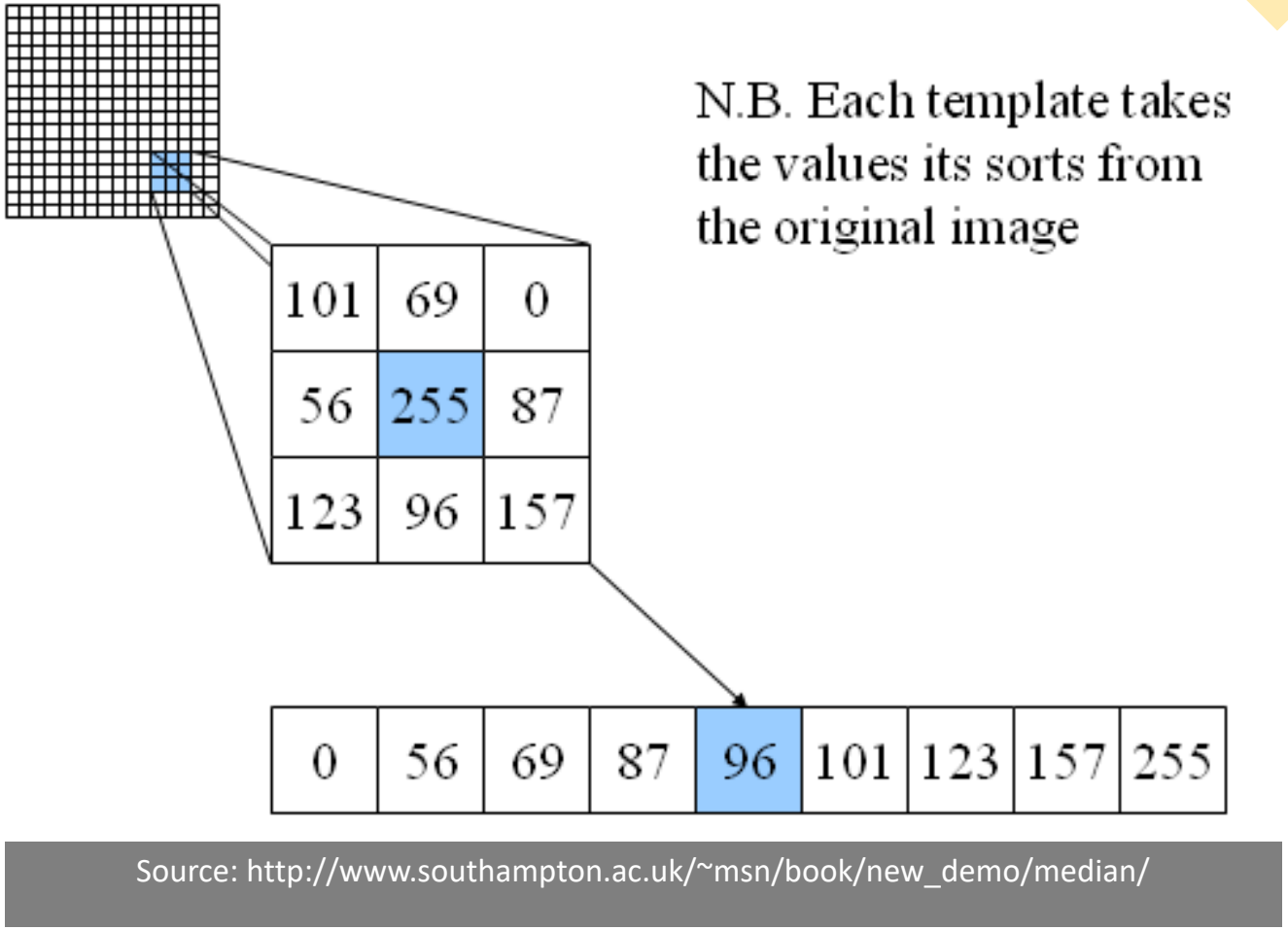
Source: https://scikit-learn.org/stable/modules/linear_model.html#lasso

Why Filtering?

Filtering is necessary to offset the effect of unwanted noise and variance. In effect, filtering “smooths out” an image. Since some of the predicted pixels may look choppy and discontinuous because they are far jumps from their neighbors, filtering ensures that the predictions will match more closely with their surroundings. In general, it brings a whole neighborhood of pixels closer to the same average value.

Why Median Filtering?

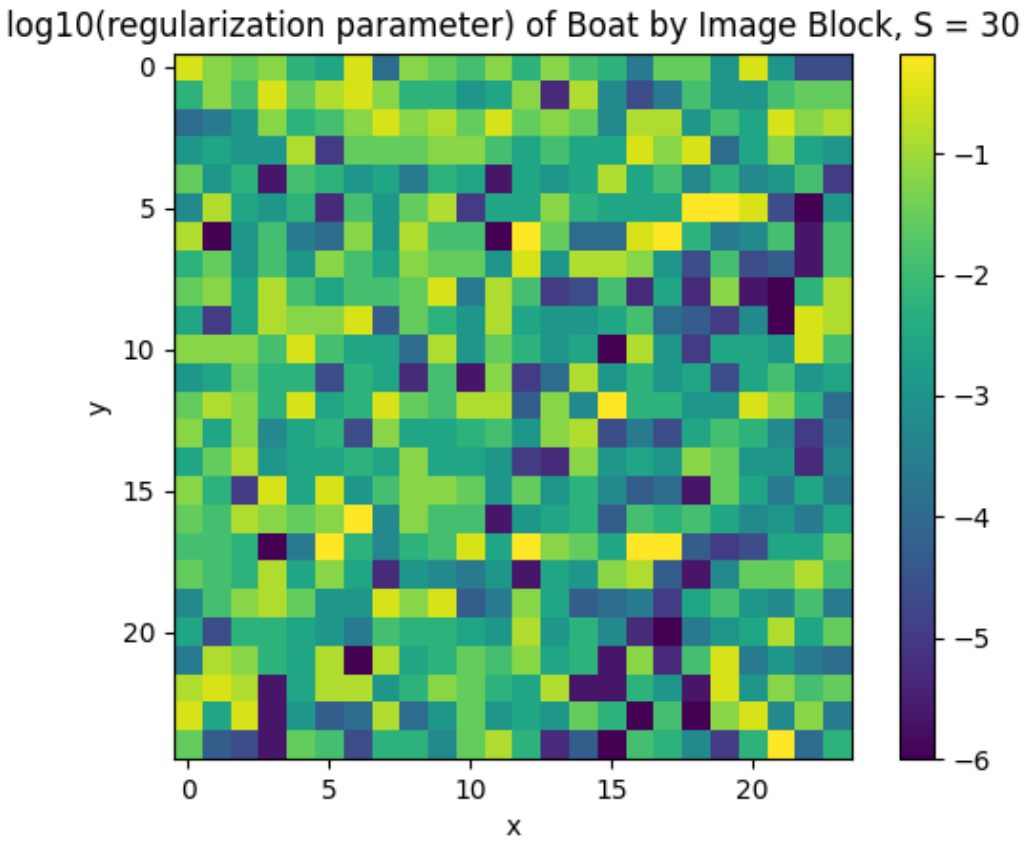
Median filtering replaces each pixel in the image with the median value of its surrounding pixels. For this project, we used the surrounding 3x3 grid to calculate the desired, filtered value. Median filtering is better than frequency filters for this project because in general, it does a better job of smoothing out "spiky noise." In our reconstructions we get many spikes because we are filling in unknown pixels. Using the median ensures that outliers won't have a massive impact on the filtered pixels (Fisher et al., 2003). Also, since the median is likely to be equal to the value of one of its neighboring pixels, we get the sense of a smooth image. Further, this helps maintain sharp edges in the image.



Source: http://www.southampton.ac.uk/~msn/book/new_demo/median/

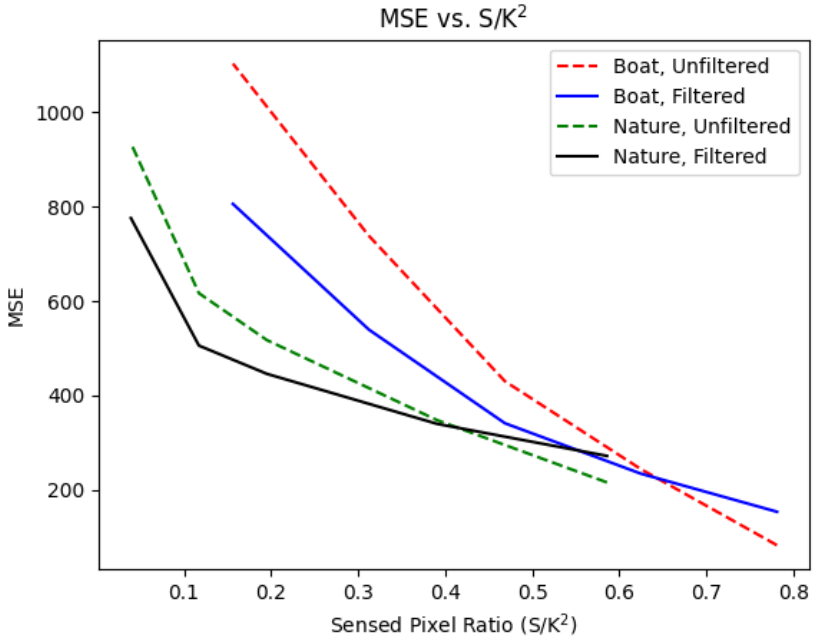
The Regularization Parameter

I had trouble finding a trend in my regularization parameters (alphas). However, one may expect alpha to be greater in more complicated image blocks and/or blocks that cause greater model variance. The purpose of regularization is to reduce overfitting. This is more likely to occur in complicated parts of the photo where a complex model could reduce error by fitting excessively to the noise. This results in a greater regularization parameter because the larger the alpha, the more we try to minimize the weights, driving them towards zero. This trend could be seen in comparing different values of S. As the number of sensed pixels decreased, (less information and complexity), the alphas also decreased.



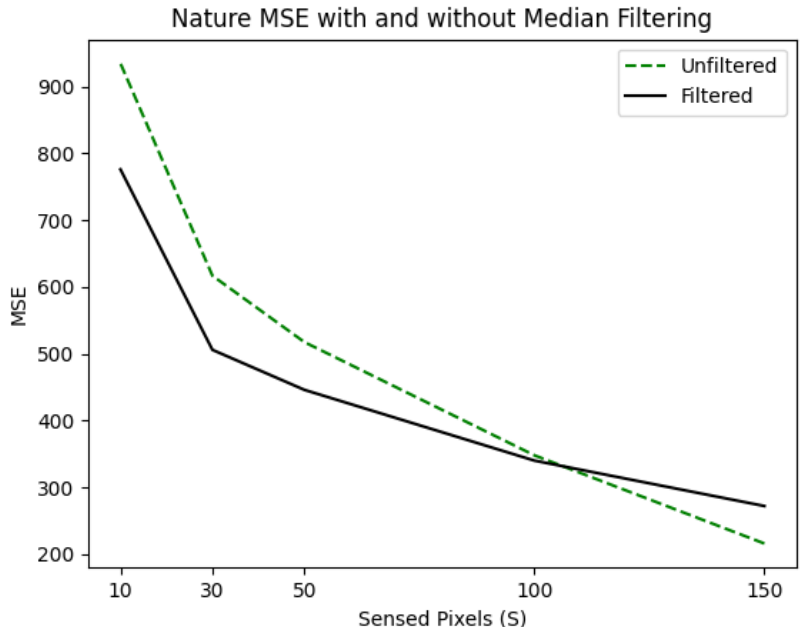
MSE vs. Sensed Pixels

As the number of sensed pixels in a block increase, MSE decreases. This makes sense for two reasons. First, for the S sensed pixels, our MSE will be zero (pre-filter). So, as we get more sensed pixels, we anticipate having less error. Second, when we have more sensed pixels, we can generate better predictions for the unsensed pixels and more accurate weights in the basis vector matrix. With more information, our linear system is “less indeterminate” so to speak.



Median Filtering and the MSE

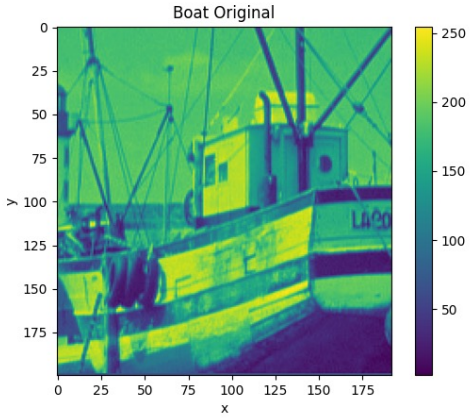
Generally, as seen in the graph to the right, median filtering decreases MSE. This is because it removes “spiky noise.” However, when S gets very large, median filtering can actually increase MSE. This is because median filtering alters all of the sensed pixels which initially had a MSE of 0. This makes their collective MSE nonzero. So, when there are lots of sensed pixels, we can worsen the MSE.

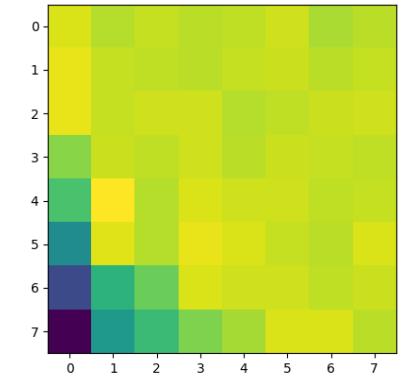


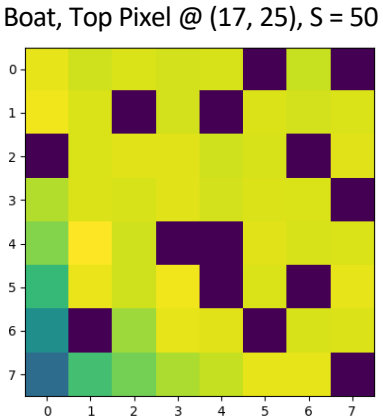
Collaborations

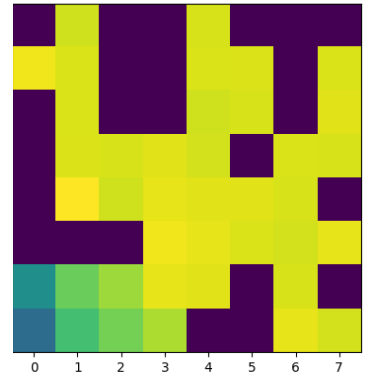
- Zion and I discussed our MSE curves and alpha grids. We shared the general shape and trends in these plots to get a sense of if we were proceeding correctly.
- Mark and I talked about median filtering and how it was possible for this to increase the MSE.
- Danny and I covered various strategies for the intercept in LASSO. I showed him that my recovery was still functioning without any additional thought given to the intercept. Nonetheless, he told me his strategy for removing and reincorporating the constant into the basis matrix to avoid the plights of the intercept.

Checkpoint 1, Fishing Boat

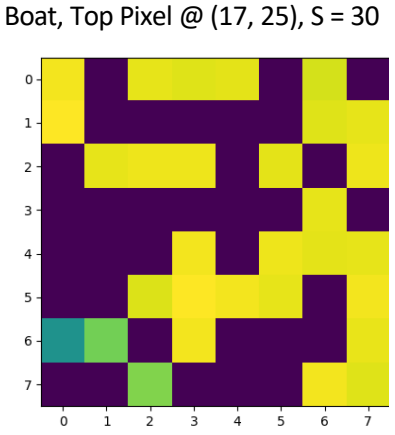


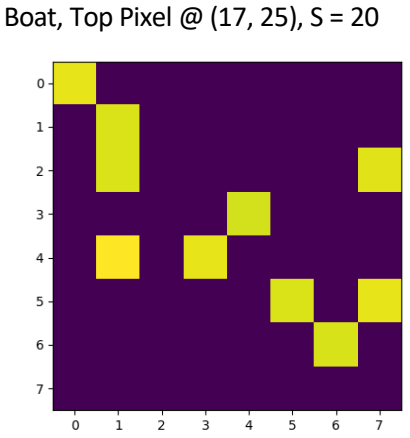


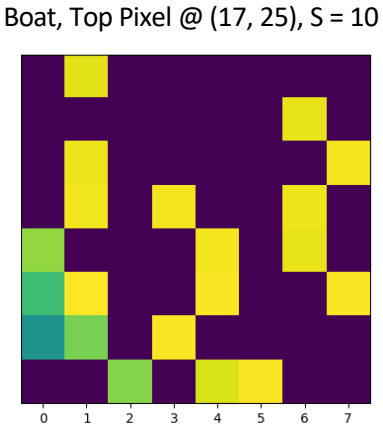




To the right are the original boat image and an 8 x 8 block whose top pixel lives at (17, 25). There are also compressions for various values of S where S = the number of sensed pixels in that 8 x 8 grid.

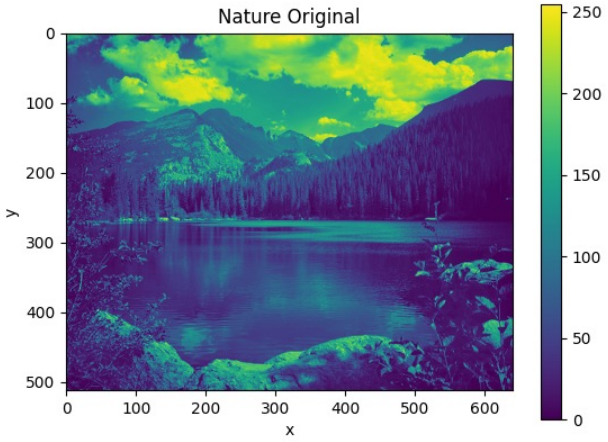




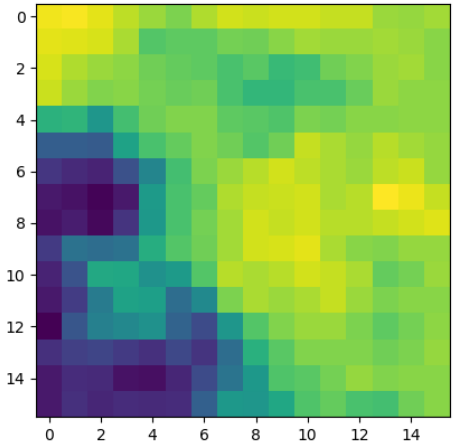


Checkpoint 1, Nature

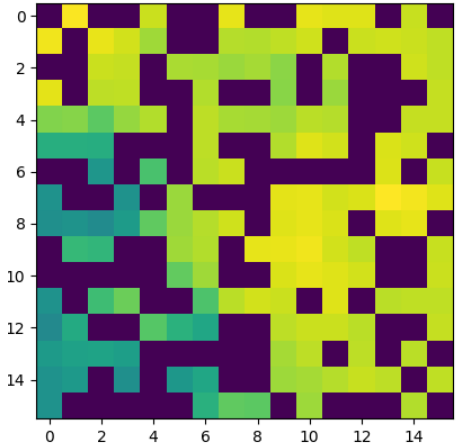
To the right are the original nature image and a 16 x 16 block whose top pixel lives at (17, 193). There are also compressions for various values of S where S = the number of sensed pixels in that 16 x 16 grid.



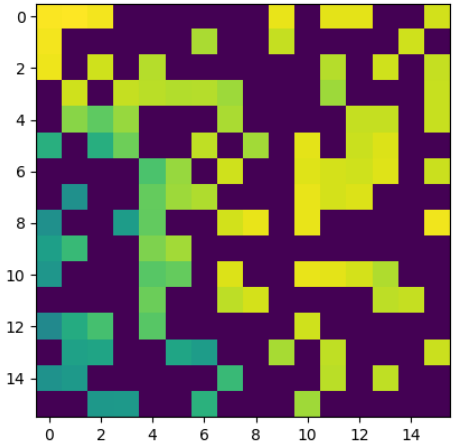
Nature, Top Pixel @ (17, 193), Uncompressed



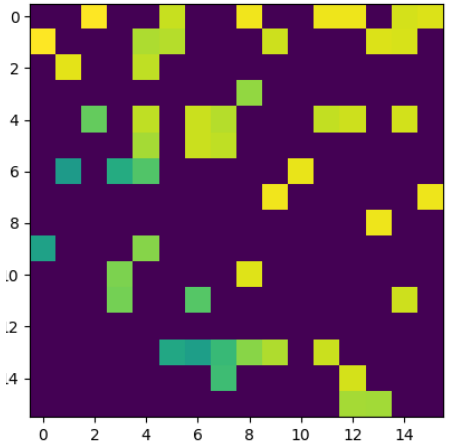
Nature, Top Pixel @ (17, 193), S = 150



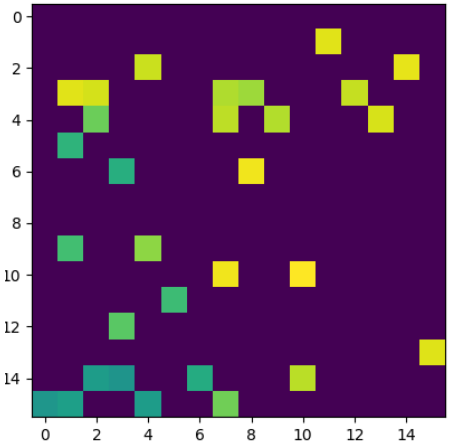
Nature, Top Pixel @ (17, 193), S = 100



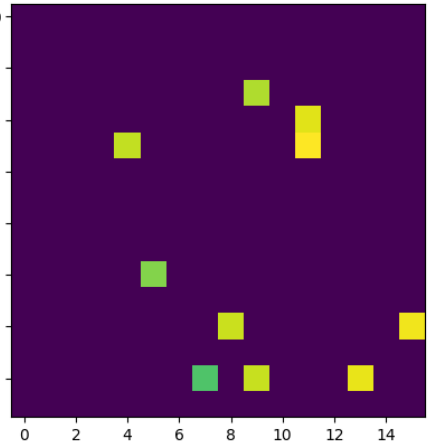
Nature, Top Pixel @ (17, 193), S = 50



Nature, Top Pixel @ (17, 193), S = 30



Nature, Top Pixel @ (17, 193), S = 10



References – Background and Math

Ahmed, N., Natarajan, T., & Rao, K. R. (1974). Discrete cosine transform. *IEEE Transactions on Computers*, *C-23*(1), 90–93. https://doi.org/10.1109/t-c.1974.223784

Bhattacharyya, S. (2018, September 26). *Ridge and lasso regression: L1 and L2 regularization*. Medium. Retrieved March 1, 2023, from https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b

Fisher, R., Perkins, S., Walker, A., & Wolfhart, E. (2003). *Median filter*. Spatial Filters - Median Filter. Retrieved March 1, 2023, from https://homepages.inf.ed.ac.uk/rbf/HIPR2/median.htm

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

References - Visualizations

https://scikit-learn.org/stable/modules/linear_model.html#lasso

https://www.datacamp.com/tutorial/tutorial-lasso-ridge-regression

http://www.southampton.ac.uk/~msn/book/new_demo/median/

References – Packages and Toolboxes

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. *Array programming with NumPy*. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.

J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

McKinney, W., & others. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference* (Vol. 445, pp. 51–56).

Pedregosa *et al.* Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830, 2011.

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. *Array programming with NumPy*. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17(3), 261-272.