INPAINTING CONVOLUTIONAL NEURAL NETS: GLOBAL AND LOCAL ANALYSIS

Alexander Dong, (awd275), Michael Stanley (mhs592)

1013: Math Tools for Data Science

1. INTRODUCTION

Recent accomplishments by convolutional neural nets (CNNs) have spurred an increased interest in designing novel CNN architectures. However, neural nets in general, and convolutional neural nets in particular, still lack interpretability. Tools for analyzing these models are lacking. In this report, we present two approaches for analyzing the inner workings of a CNN model that was trained on the inpainting task. The first approach is a global approach using Principal Component analysis, and the second approach is a local approach using Jacobian Analysis of a bias-free CNN. The global approach allows us to compare the principal components of reconstructed images and the originals and the energy captured in the first few principal components. The local approach visualizes the impact of input pixels on particular reconstructed, output pixels, and quantitatively compares the locality of the inpainting model for different mask sizes.

These analyses shed light on how the “black box” CNN is learning to perform the task of inpainting, and how it compares to traditional methods such as diffusion and exemplar methods.

2. STATE OF THE ART

This paper focuses on the introspection of an inpainting CNN algorithm and does not attempt to improve upon performance. Inpainting models are typically evaluated with the peak signal to noise ratio (PSNR) on a benchmark dataset. We will not report PSNR benchmarks below as it is not pertinent to the work of the paper, but the following provides a contextual history of inpainting and bias-free CNNs.

2.1 Inpainting

Inpainting is a classic problem in image processing. Inpainting is the act of inferring (“filling in”) an unknown region of an image (or other signal). Applications of inpainting include repairing damaged images, removing unwanted objects in images, or filling in parts of an image that were blocked. An example of inpainting is shown in Figure 1.

There are two traditional approaches to inpainting: diffusion methods and exemplar methods. Diffusion methods extend adjacent patterns (with an emphasis on extending isophote lines of equal gray value) and textures into the unknown region by solving a partial differential equation describing color propagation (Bertalmio et al., 2000). These methods iterate from the border to the center of the unknown region. The approach described in (Telea, 2004) implements the fast marching method to improve computational performance and was widely adopted.

While diffusion methods are inherently local, exemplar-based methods (”patching”) identify similar patches elsewhere in the image and copy them into the unknown region. (Criminisi et al., 2003) extends texture synthesis methods to inpainting. (Wong, Orchard, 2006) extracts multiple exemplar samples from the image and combines them by a similarity score weighting approach. (Barnes et al., 2009) introduced an algorithm that facilitates searching entire images for similar exemplars.

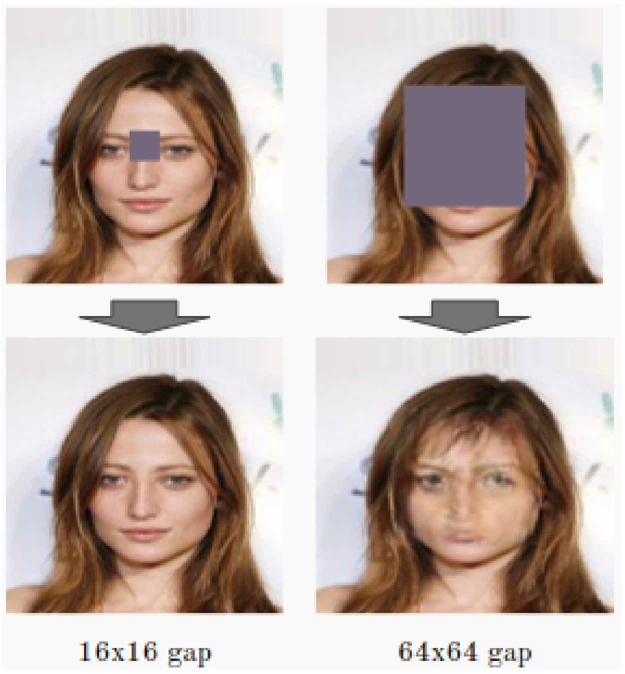

Figure 1. Masked input image before and after masked region reconstruction for 16x16 (left) and 64x64 (right) mask size.

Recently, convolutional neural networks and generative adversarial networks have been put to use in inpainting. (Pathak et al., 2016) uses an encode-decode framework that attempts to encode semantic features (in a single receptive field) from the input image, and use the semantic features to reconstruct the missing pixels. (Wang et al., 2018) introduce a generative multi-column similar to (Pathak et al., 2016) except with different columns representing different receptive fields. PEPSI, described in (Sagong et al., 2019), combines a single encoding network with multiple, parallel decoding networks to improve computational performance and inpainting quality. The model appears to improve upon benchmarks on the CelebA dataset we use in this paper.

2.2 Bias-Free Convolutional Neural Networks

This paper applies linear algebraic analysis techniques to the Jacobian matrix of a bias-free convolutional neural network trained for inpainting. This paper follows two approaches applied to denoising in (Mohan* et al., 2020): i) interpreting the magnitude of elements of the Jacobian as weights on the input pixels for each output pixel and ii) performing a singular value decomposition of the Jacobian.

3. METHODOLOGY

3.1 Context Encoder CNN

We modified and trained a bias-free version of the Context Encoders model described in (Pathak et al., 2016). This model was chosen because it is near the state of the art in inpainting, and the architecture lends itself to the removal of bias. We trained over 100,000 images in the CelebA Faces Dataset (Liu et al., 2015). The original paper did not apply Context Encoders to faces. Each image was 128x128 pixels, with a centered 64x64 pixel mask, or a centered 16x16 pixel mask. Bias terms were removed from each layer of the neural net to attain the local linearity described in Section 3.3.

3.2 Principal Component Analysis

For the "global" portion, we analyzed results over 1000 images in a holdout set. We first computed the pixel-wise eigenvalues and eigenvectors of the original images and reconstructed images for the centered 64x64 pixel masks. Each entry of an eigenvector represents a coefficient for a single pixel coordinate. In total, there were 4096 eigenvectors each with 4096 entries. In total, there are two sets of eigenvalue/eigenvectors - one set for the original set, and one set for the reconstructed set.

We compared the relationship of eigenvectors for the original and reconstructed images. We projected Context Encoder's reconstructed images onto the original image set's eigenvectors, which provide qualitatively better reconstructions than Context Encoder's reconstructions. We also examine the amount of energy that is captured by the projected reconstructed images.

3.3 Bias-Free Jacobian Analyis

(Mohan* et al., 2020) shows that, for a bias-free convolutional neural net using Rectified Linear Unit (ReLU) activation functions is locally linear, and can be represented by a matrix transformation. For the Context Encoder used in this paper:

$$f_{BF}(y) = W_L(R_{leaky}(W_{L-1}...R_{leaky}(W_1 y)) = A_y y \quad (1)$$

where f_{BF} is the reconstructed region, W_i is the weight matrix for layer i, R_{leaky} is the leaky ReLU activation layer, and y is the 128x128 pixel masked input image[1]. The dimensionality is as follows:

$$y \in \mathbb{R}^{128\times128} \quad (2)$$
$$A_y \in \mathbb{R}^{m\times m\times128\times128} \quad (3)$$
$$f_{BF} \in \mathbb{R}^{m\times m} \quad (4)$$

where $m \times m$ is the size of the masked region ($m \in \{16, 64\}$ here).

A_y is the Jacobian of the Context Encoder at input image y. For an output pixel $z = (i, j)|i, j \in \{0, ..., m-1\}$, we can express z as:

$$z = A_{y,[i,j]}y \quad (5)$$
$$A_{y,[i,j]} = A_y[i, j, :, :] \quad (6)$$

where $A_{y,[i,j]} \in \mathbb{R}^{128\times128}$. The elements of $A_{y,[i,j]}$ represent the weights of each input pixel on the output pixel z. The spatial distribution of the large weights over the input image reveals how the Context Encoder is filling pixel z. If the weights are only large near z, the Context Encoder is highly local at point z. If the weights are comparable across the input region, the Context Encoder is using more information from across the input image.

[1] The Context Encoder uses leaky ReLU activation functions, which provide the same local linearity as ReLU activation functions.

To quantify locality, we define ϕ_p as the ratio of the Frobenius norm of an inner window of $A_{y,[i,j]}$ to the Frobenius norm of $A_{y,[i,j]}$:

$$\phi_p = \frac{\|A^{(p)}_{y,[i,j]}\|_F}{\|A_{y,[i,j]}\|_F}\text{, where } p \in [0, 1] \quad (7)$$

$$A^{(p)}_{y,[i,j]} = A_{y,[i,j]}\left[\tfrac{128-pm}{2} : \tfrac{128+pm}{2}, \tfrac{128-pm}{2} : \tfrac{128+pm}{2}\right] \quad (8)$$

Intuitively, ϕ_p is higher if the pixels near the masked region are heavily weighted relative to the rest of the input image. Figure 2 illustrates $A^{(p)}_{y,[i,j]}$.

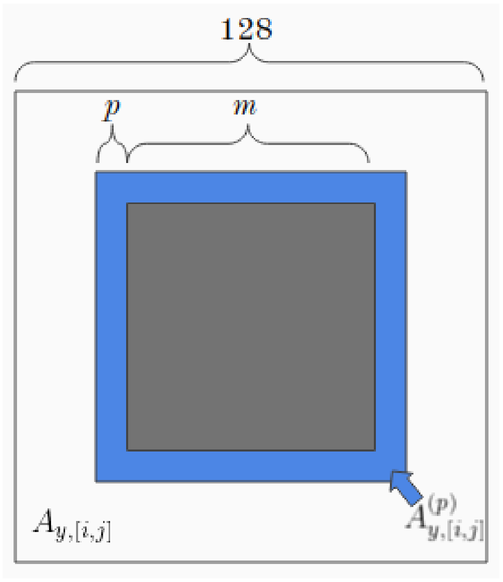

Figure 2. $A^{(p)}_{y,[i,j]}$ is the inner window of pixels from the masked input image.

The focus of the bias-free Jacobian analysis is how $A_{y,[i,j]}$ compares for different output pixels $[i, j]$ (e.g., corner, edge, center) and for different sizes of the masked region. High locality would imply that the Context Encoder is emulating a diffusion model where nearby pixels are weighted most heavily for inpainting. Large weights elsewhere in the image would indicate either an exemplar method or perhaps that the CNN has learned about the broader structure of images during training ("context").

4. RESULTS

4.1 Principal Component Analysis

Figure 3 presents an overview of the Top 10 eigenvectors from both the (cropped) original and (cropped) reconstructed image sets. Firstly, one should notice that the "eigenimages" in the original dataset strongly resemble faces, and this observation is fundamental to the rest of the work in this section. A visual overview of the "eigenimages" in both the original and reconstructed image sets suggest that the "eigenimages" might be

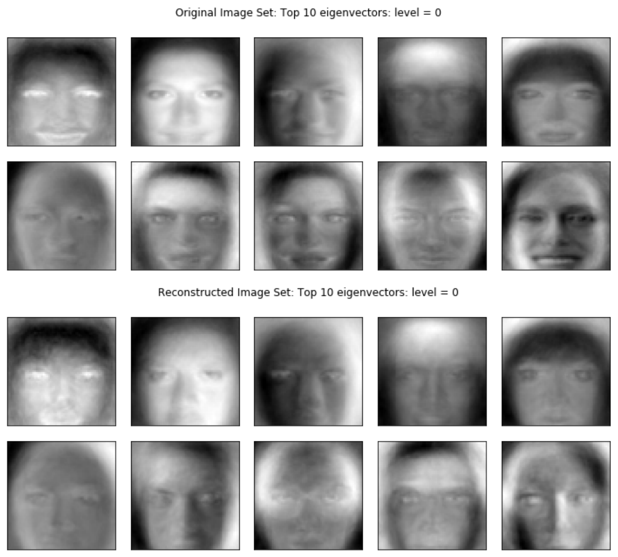
Original Image Set: Top 10 eigenvectors: level = 0
Reconstructed Image Set: Top 10 eigenvectors: level = 0

Figure 3. Top 10 Eigenvectors for over the original images and reconstructed images. Images were reconstructed using the Context Encoder CNN architecture. There were a total of 1000 images in the holdout set. The images were originally 128x128 pixels, and the images shown are the 64x64 center that correspond to the masked pixels.

similar. This suggests that the CNN was able to properly learn the semantic features of faces, well enough that the "eigenimages" in the reconstructed image set resemble those in the original image set.

Figure 4 shows the cosine similarity of the eigenvectors in the original image set versus the eigenvectors in the reconstructed image set. We can see that the first few eigenvectors all have very high cosine similarities. After the 15th eigenvector, the cosine similarities really start to drop off. This provides some evidence that the CNN is able to learn semantic features of the faces in the training set.

Figure 5 shows a spectrum of reconstructed images that have been projected onto the original image set's "eigenimages". Visual inspection indicates that projecting onto 100 principal component vectors seems to be where the projected reconstructions begin to show some resemblance to the reconstructed face. Using 250 or 500 principal components allows for finer detail to

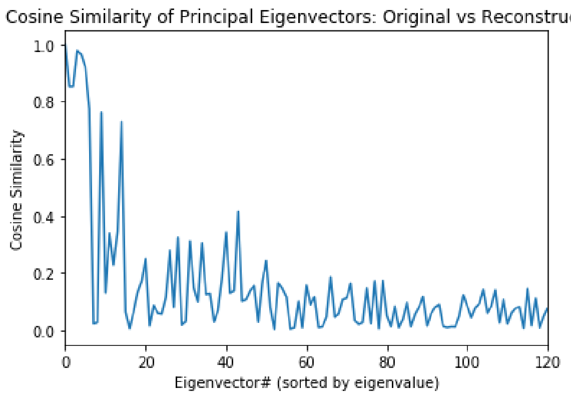

Figure 4. Cosine similarity of eigenvectors of the original image set and reconstructed image set (ranked by eigenvalues). In order to account for sign inconsistencies between eigenvectors, we show the absolute value of the cosine similarity.

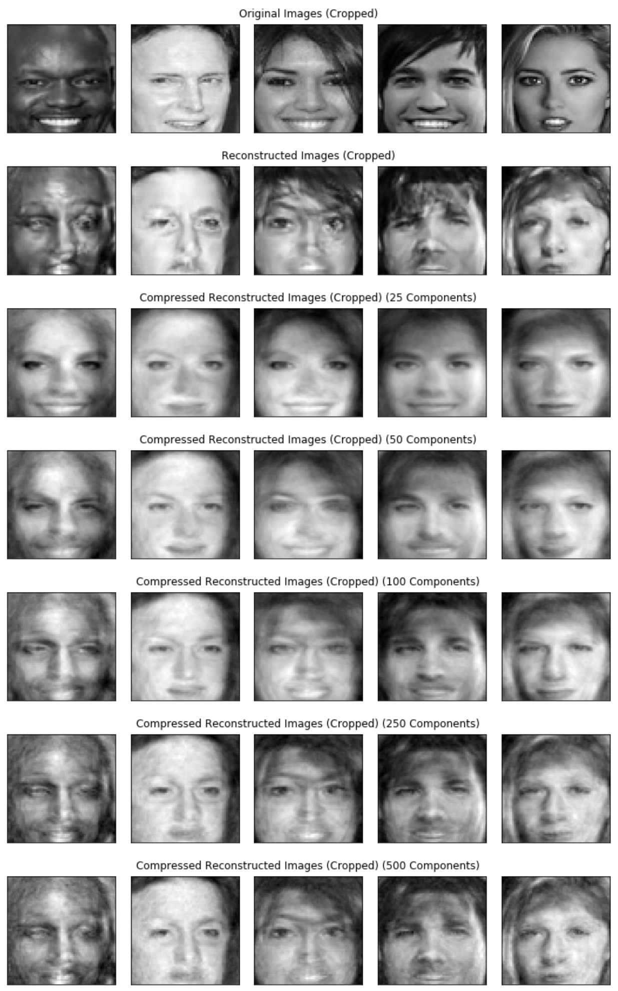
Original Images (Cropped)
Reconstructed Images (Cropped)
Compressed Reconstructed Images (Cropped) (25 Components)
Compressed Reconstructed Images (Cropped) (50 Components)
Compressed Reconstructed Images (Cropped) (100 Components)
Compressed Reconstructed Images (Cropped) (250 Components)
Compressed Reconstructed Images (Cropped) (500 Components)

Figure 5. Five hold-out images projected on differing numbers of principal components.From Top to bottom: Original Image, Reconstructed Image, then projections onto 25,50,100,250 and 500 principal components

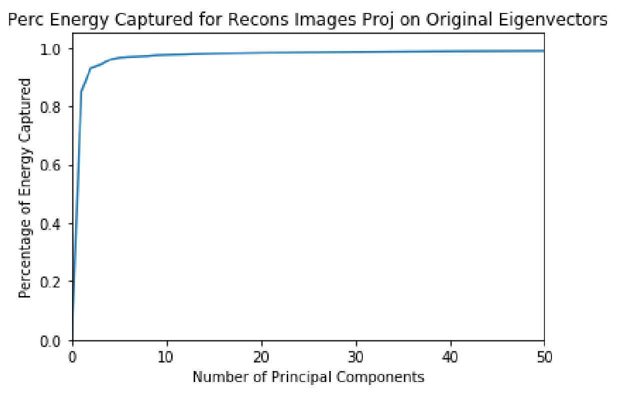

Figure 6. Average percent of energy captured for all 1000 holdout set images projected onto the original image set's eigenvectors.

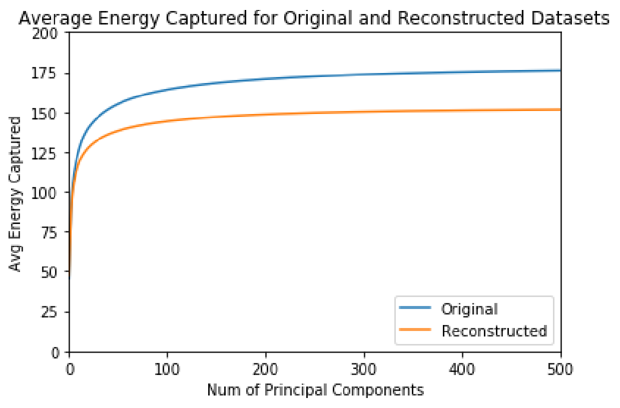

Figure 7. Average Total Energy Captured for Original Images and Reconstructed Images.

be seen (such as the hair, or shadows on the face). Overall, the reconstructed images do bear some semblance to the original images, but they grotesquely mess up the fine details. However, the projected reconstructed images also retain strong semblance to the reconstructed images, and also a strong semblance to the *original* images. We believe that this is a particularly interesting result, and extensions from this observation are proposed in the Discussion section.

Figure 6 displays the average percentage of energy captured when reconstructed images are projected onto the original image set's eigenvectors. Unsurprisingly, the first few principal components capture the vast majority of the energy, defined in equation 9.

$$EC = \frac{||Proj_{Q_r}(Im)||_2}{||Im||_2} \tag{9}$$

Another unrelated result is the figure displayed in Figure 7. This figure shows that the average energy in the reconstructed images is noticeably lower than the average energy in the original images. We're not exactly sure what the reason is, but we believe that it's an artifact of the CNN.

4.2 Local Analysis

The Jacobian matrix of the bias-free Context Encoder was calculated for a few test images with both mask sizes, 64x64 pixels and 16x16 pixels. Figure 10 shows the weights on each input pixel for four output pixels and both mask sizes. For context, Figure 1 shows the masked input and reconstructed images for both mask sizes.

Corner and edge pixels exhibit high locality, indicating that the Context Encoder is essentially combining nearby pixels to fill the mask borders. As we move toward the center of the image, locality diminishes for the large mask and spreads to all mask borders for the small image. Figure 8 quantifies this, comparing ϕ_p for a corner, edge, and center pixel. ϕ_p is higher for the edge and corner pixels than for the center, across all values of p. Surprisingly, ϕ_p is higher for the edge pixel than for the corner pixel.

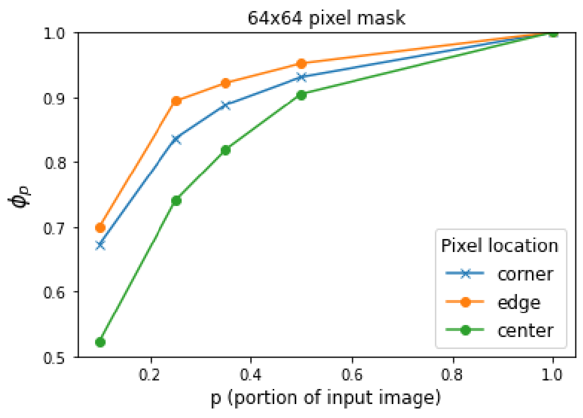

Figure 8. ϕ_p for output pixels: corner ([0,0]), edge ([32,63]), and center ([32,32])

The smaller masked region is more localized than the larger mask, as shown in Figure 9 comparing ϕ_p for the center pixel of both mask sizes. Note that, because the masks are different sizes, p represents a different number of pixels for the two masked regions.

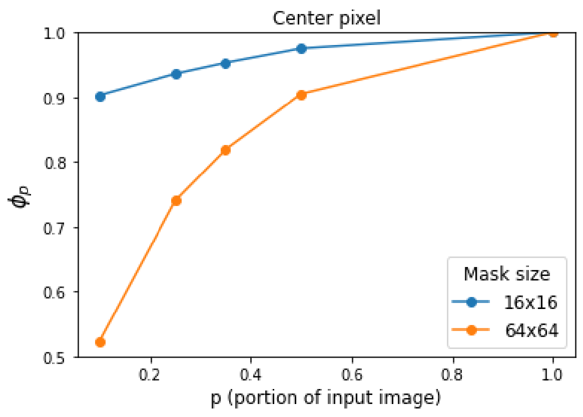

Figure 9. ϕ_p for center output pixels and both mask sizes.

Interestingly, center output pixels have larger weights on input pixels in the lower half of the image. This may indicate that the Context Encoder has learned that the neck is more similar to the face than the background. The emphasis on the neck pixels may also explain how the model matches skin color quite well, even if there are no skin-colored pixels on the upper or side borders of the masked region.

There is a faint grid pattern to the pixel weightings, perhaps most evident in the bottom left of Figure 10. The squares are larger for the 64x64 pixel mask, rougly proportional to the mask size. While we have not investigated further, this may be the Context Encoder learning to use an exemplar inpainting method. However, it could also be an artifact of the Context Encoder's convolution window size.

5. DISCUSSION

The PCA analysis seems to suggest that Context Encoder's reconstructed images are well represented by their projections onto the original image set's eigenvectors. The projected reconstructed images clearly resemble the reconstructed images, and perhaps surprisingly, resemble the original image. Furthermore, the projected reconstructed images contain less of the distortions that are present in the many of reconstructed images.

Analyzing the Jacobian of the bias-free Context Encoder, we have developed a deeper understanding of how the model transforms the masked input image to inpaint the masked region. The impact of input pixels is highest in the local region around the input image, and more localized for pixels near the border of the masked region than center pixels. Comparing large and small masked regions, the inpainting mechanics are different: smaller masked regions are more impacted by nearby pixels, whereas larger masked regions are impacted more by the broader image.

The work in this paper points to a number of extensions. Extending the analysis outside of facial images to images exhibiting more patterns would disentangle the findings here from the nature of facial images (e.g., backgrounds with no information in the top and sides of the image). The linear-algebraic analysis can be extended further, notably via a singular value decomposition of the Jacobian matrix, as reported for denoising in (Mohan* et al., 2020).

Architecturally, a possible extension would be to investigate whether the principal components could be integrated into some form of regularization for the neural net. The reduced distortion in the projected images in Figure 5 suggest that the CNN is able to reconstruct the main components of the image, but the image is ultimately polluted with some sort of noise. Another possible extension would be to investigate whether the CNN architecture could be modified to integrate the principal components directly into the training process. As of now, after the CNN is trained, we hope that the CNN weights somehow encode semantic features; it would be interesting to see if incorporating the eigenvectors during the training process leads to improved results.

REFERENCES

Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D., 2009. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics*.

Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C., 2000. Image in painting. *In Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pages 417–424. ACM Press/Addison-Wesley Publishing Co.*

Criminisi, A., Perez, P., Toyama, K., 2003. Object removal by exemplar-based inpainting. *CVPR 2, 721.*

Liu, Z., Luo, P., Wang, X., Tang, X., 2015. Deep learning face attributes in the wild. *Proceedings of International Conference on Computer Vision (ICCV).*

Mohan*, S., Kadkhodaie*, Z., Simoncelli, E., Fernandez-Granda, C., 2020. Robust and interpretable blind image denoising via bias-free convolutional neural networks. *ICLR.*

Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., Efros, A. A., 2016. Context encoders: Feature learning by inpainting. *CVPR.*

Sagong, M.-C., Shin, Y.-G., Kim, S.-W., Park, S., Ko, S.-J., 2019. Pepsi: Fast image inpainting with parallel decoding network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, to be published.*

Telea, A., 2004. An image inpainting technique based on the fast marching method. *Journal of graphics tools., 9(1):23–34,.*

Wang, Y., Tao, X., Qi, X., Shen, X., Jia, J., 2018. Image inpainting via generative multi-column convolutional neural networks. *Advances in Neural Information Processing Systems*, 331–340.

Wong, A., Orchard, J., 2006. A nonlocal-means approach to exemplar-based inpainting. *Proc. IEEE Int. Conf. Image Processing (ICIP), pp. 2600-2603.*

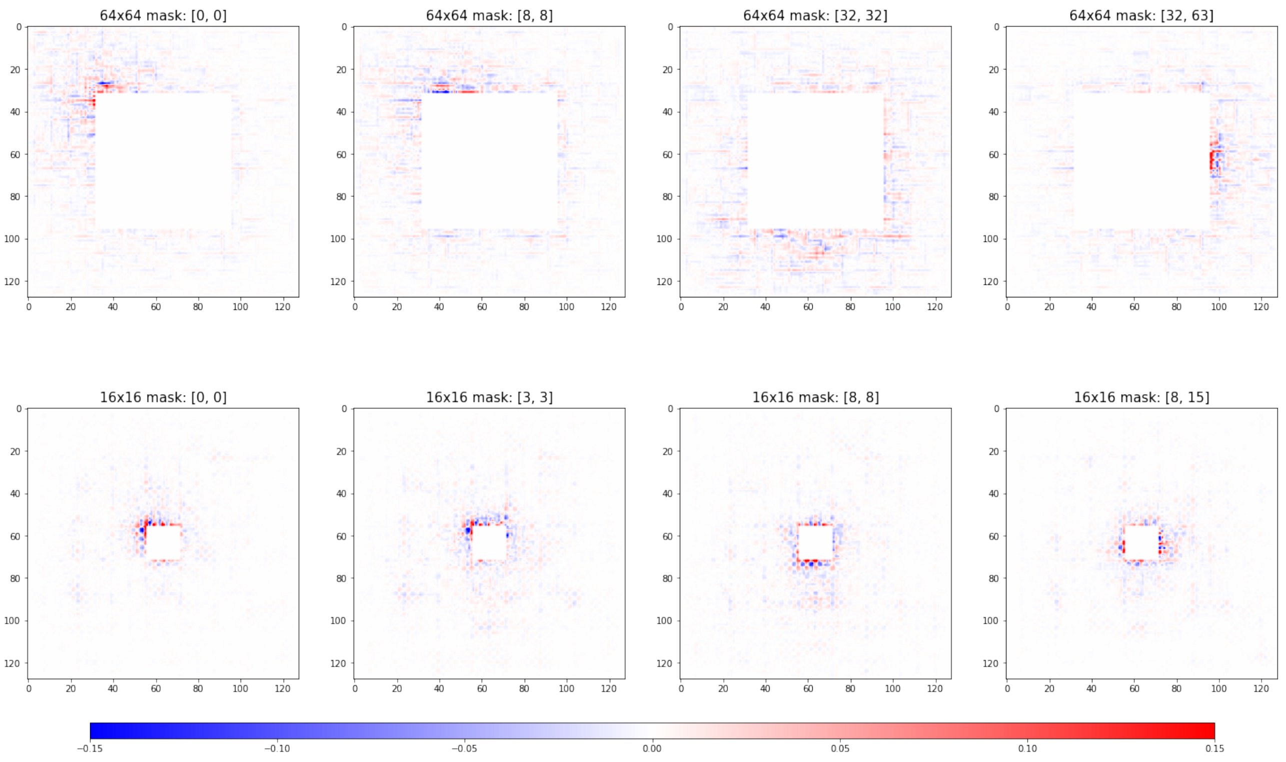

Figure 10. Input pixel weightings for 64x64 mask (top) and 16x16 mask (bottom). Pixel indices indicated on the images, corresponding to i) top left, ii) diagonally in from top left, iii) center, and iv) middle right edge. Input image is the image from Figure 1.