Mask-specific inpainting with deep neural networks

Rolf Köhler, Christian Schuler, Bernhard Schölkopf and Stefan Harmeling

Max Planck Institute for Intelligent Systems, Tübingen, Germany

Abstract. Most *inpainting* approaches require a good image model to infer the unknown pixels. In this work, we directly learn a mapping from image patches, corrupted by missing pixels, onto complete image patches. This mapping is represented as a deep neural network that is automatically trained on a large image data set. In particular, we are interested in the question whether it is helpful to exploit the shape information of the missing regions, i.e. the masks, which is something commonly ignored by other approaches. In comprehensive experiments on various images, we demonstrate that our learning-based approach is able to use this extra information and can achieve state-of-the-art inpainting results. Furthermore, we show that training with such extra information is useful for *blind* inpainting, where the exact shape of the missing region might be uncertain, for instance due to aliasing effects.

Keywords: inpainting, deep learning, neural-nets, multi-layer-perceptrons

1 Introduction and related work

Image inpainting tries to fill-in missing parts of an image. Commonly, one can distinguish two settings, where pixels in an image are missing:

(i) In the first setting, the goal is to manipulate an existing image. Usually, some image details or larger regions of a given image should be removed. The resulting hole must be filled in to create a *plausible* complete image. For instance, consider an image with two persons, where one person should disappear. Then the task of image inpainting is to fill-in the resulting (possibly large) hole with some background textures or patterns. The goal is not to recover a *true* image but one that looks realistic. Many successful methods (some based on texture synthesis [8]) have been proposed in the past, for instance [7, 17, 5, 26, 13] and references therein. These ideas can be also generalized to video inpainting [23].

(ii) The second setting considers an image that is locally corrupted, for example by super-imposed text or scratches. In this case, the missing regions are small but possibly all over the image. The goal is to recover an image which is as close as possible to the *true* image. Also for this setting many good approaches exist which will be discussed in the following.

Both settings are relevant for image processing, however, in this paper we solely consider the second setting. The existing methods for the second setting can be categorized into two groups:

(i) Many classical approaches are *diffusion-based* methods that propagate the local information, such as edges and gradients, from the boundary to the missing pixels, see e.g. [16, 2, 4].
(ii) The second class is based on *sparse representations* using dictionaries [9, 14, 11]. General purpose image priors based on Markov random fields (MRFs) can be learned on image databases [18, 20]. These have been also successfully applied to inpainting.

While the existing methods lead to impressive image reconstruction results, none of them is exploiting the shape of the mask for inpainting. The (binary) mask has the same size as the image and is 1 for pixels which are missing in the corrupted image and 0 for all other pixels. In this paper we show how this additional information can be utilized to obtain better image reconstruction results. For this we choose a task-specific learning approach employing deep neural networks since they have recently been successfully applied to several other image restoration problems, e.g. to image denoising [3] or to image deblurring [21]. Closest to our work is [25] who apply a deep learning approach to both denoising and inpainting. They are also able to do blind inpainting (as we do in Sec. 3.4), but do not use the mask information.

In a nutshell, the contributions of the present paper are as follows:

- We show that a mask-specific inpainting method can be learned with neural networks, which leads to better results.
- We show that it is relevant to train the inpainting method with the correct masks.
- We show that by training an inpainting method for masks generated with certain fonts, it is possible to *blindly* inpaint an image, i.e. without knowing the locations of the missing pixels.

2 Learning mask-specific inpainting methods

The overall idea is to train a neural network to map corrupted image patches (=rectangular parts from the image) to their uncorrupted counterparts. This mapping is applied to all patches of a corrupted image. The recovered patches are averaged at their overlapping parts to obtain the reconstructed image. In case the true mask for the whole picture is given, the known pixels from the input image are directly copied to the reconstructed image. Note that the input patches are usually larger than the output patches, which matches the intuition that the missing pixels can be recovered by considering some large enough area around them. In the following we briefly recall multi-layer perceptrons (MLPs) and explain the training procedure.

2.1 Multi-layer perceptrons

A multi-layer perceptron is a nonlinear function f that maps input vectors x onto output vectors y. We follow the notation of [3]. An MLP with two hidden layers can be written as

$$f(x) = b_3 + W_3 \tanh(b_2 + W_2 \tanh(b_1 + W_1 x)) \tag{1}$$

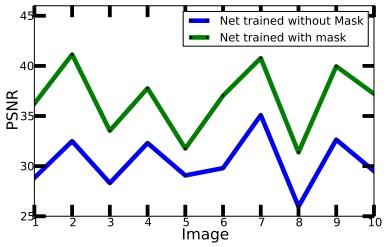


Fig. 1: PSNR for 10 test images for an MLP that takes only the corrupted patch as input (blue line) and an MLP that additionally has the masking patch as input (green line).

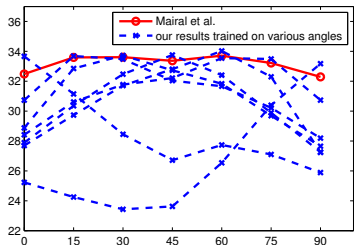


Fig. 2: Performance of the MLP trained on different angles compared against the method by [14]. The comparison against [20] looks similar. The performance of the neural net depends on the orientation of the bars and is best for the orientation on which the neural net was trained on.

(from Eq. (1) in [3]) where vectors b_1, b_2, b_3 and the matrices W_1, W_2, W_3 are the parameters of the MLP. More generally we denote the architecture of an MLP by some integer tuple: e.g. a $(256, 1024, 1024, 64)$-MLP has a 256-dimensional input layer, two 1024-dimensional hidden layers and a 64-dimensional output layer.

2.2 Mask-specific training

To adjust the parameters of the MLP, we need training data that consists of pairs of input patches x and output patches y. Using these we can automatically *learn* the mapping using the backpropagation algorithm [19, 12]. To speed up convergence we use the ADADELTA method [27] that automatically adapts parameter-specific learning rates (fixing the decay rate $\rho = 0.95$, the conditioning constant $\epsilon = 10^{-6}$, and batch size to 128).

To generate large amounts of training pairs we randomly select image patches from an image database (we used imagenet [6] for training). These extracted patches are the uncorrupted output patches.

To corrupt those patches we utilize the knowledge about the masks. For instance for super-imposed text we corrupt the output patches by adding random text of the same font and size, if that information is available. This allows us to create input patches with corruptions that are similar to the corruptions in the test image.

2.3 Training with and w/o mask as input patch

Many inpainting algorithms require the exact locations of the missing pixels. For this reason we consider two versions of our approach: the first considers only the corrupted patch as the input. Note that the corrupted patch does not always contain the information which pixels are missing (e.g. for blind inpainting, see Sec. 3.4). Version two of our approach requires the corrupted patch and additionally the masking patch as the input.

To show that feeding the masking patch additionally helps, we performed a comparison of the two approaches on 10 test images (randomly selected from the Berkeley segmentation dataset [15]) with super-imposed text. Fig. 1 shows the results of two neural nets, both with architecture $(16^2, 512, 512, 512, 8^2)$ which were trained for the same amount of epochs. The first MLP was trained with the mask as an additional input, the second MLP was trained without the mask, but with correctly corrupted patches, i.e. using the correct font and text size. In all cases the MLP with the additional input was better. See also Fig. 3 for example images.

(a) input image (b) trained w/o mask (c) trained with mask

Fig. 3: The performance of the neural net improves if a mask is included in the training process. It can be seen that in image (b), artifacts, especially in the sky, are visible.

2.4 Training with the correct and wrong masks

In this section we demonstrate the advantage that is gained through incorporating the correct mask into the learning process. For that purpose we generate several masks which are similar. We start with a mask, showing vertical bars of size 14×3 pixels. We rotate those bars repeatedly by 15 degrees. We do not allow aliasing, but only binary masks, so each pixel in the mask which is greater than 0 after rotation is set to 1.

We trained several MLPs with the architecture $(16^2, 512, 512, 512, 8^2)$ using training images generated with these different masks. For each of those angles, also 10 test images are generated (10 randomly picked images from the Berkeley segmentation dataset [15]).

Fig. 2 shows that the neural net performs best on the angle it was trained on, and that the performance deteriorates quickly for other angles. The methods by [20] and [14] perform about the same for all angles, no matter in what way the bars are orientated. Our approach is better if the correct angle has been considered during training time.

3 Experiments

We always trained the MLPs on gray-scale images. To apply them to color we applied the same MLP separately to the three color channels. To generate training patches we used 1.8 million color images from imagenet [6], which we converted to gray-scale. Note that in general the larger the architecture of the neural network, the better the results, but the longer the training time until convergence. In average the training time until convergence is 3 to 5 days.

Note that we did not have access to implementations for all competing methods. For Figs. 4, 5 and 6 we applied all algorithms that were available to us. Furthermore, we included for Fig. 5 those methods which published results on the "New Orleans" image in their papers. Fig. 6 shows a comparison on images from [26].

3.1 Comparison with horizontal/vertical lines

We perform an experiment with horizontal and vertical lines painted on an image. We used a $(16^2, 1024, 1024, 1024, 8^2)$ architecture. As can be seen in Fig. 4, we achieve a better PSNR than the methods by [20] and by [14]. This is especially due to the fact that the MLP is able to inpaint the stripes on the shirt in a much better way than the other algorithms.

input, PSNR 17.36 [1], PSNR 32.91 [20], PSNR 35.31 [14], PSNR 35.32 ours, PSNR **37.34**

Fig. 4: Comparison on an image with line artifacts. Note that the chest-part of the shirt shows fewer artifacts with our approach than it does with the other methods.

3.2 Comparison on the New Orleans image

We compare our method against other methods on the New Orleans image, used by [2]. For this comparison a $(39^2, 2047, 2047, 2047, 2047, 13^2)$ architecture was trained. Visually our approach performs better than the approaches by [5], [2], [20], [18], [22] and by [1]. Our result looks similar to the result of [14], but we are still able to recover more detailed information, e.g. the pole (marked with

a green box in the corrupted image). One reason why we are better than the second best performing method [14] might be that we are able to use larger patch sizes. We used input patches of size 39×39 pixels, whereas [14] used a dictionary with patch size 9×9 pixels.

Note that for this example the original image was corrupted with some text of which the true mask is available. We used exactly this mask to corrupt images for the training set.

3.3 Comparison against images from Xu and Sun [26]

Fig. 6 shows the results for one out of five images from [26] (Fig. 8 in their paper). In Sec. 3.5 we show the limitations of our proposed method on one of the other images. The neural net (with architecture $(39^2, 2047, 2047, 2047, 13^2)$) was trained with the given masks from [26]. On the Lena image we achieve better results (in terms of PSNR) than all other algorithms.

3.4 Comparison for blind inpainting

With an MLP we are also able to inpaint images without the exact knowledge of the mask while other inpainting methods do require a binary mask as an input. Fig. 7 shows results for two images, on which random text was written with the same font and font-size as used for the training procedure of the MLP.

Note that the text written on the image is aliased, meaning it is not binary. Extracting a mask from such an image is rather tedious, as the optimal mask cannot be found by just thresholding the image. However, identifying the font and font-size is often possible and provides important information for the training process.

Optimally we would compare against [25] who also consider a deep learning approach to inpainting (but ignoring the mask based training). Unfortunately, we were not able to obtain their images nor their code, so we have to postpone this comparison for a later publication. For this reason we compare only against the method of [20] which seems to be our closest competitor in terms of PSNR. Since the latter method requires the mask we additionally provided it to their method. Although our blind approach did not have access to the mask, it was able to reconstruct a better result.

We see that our method can be used to automatically inpaint images with super-imposed text or for automatic removing of watermarks or logos. It can also be used to batch inpaint several images which were damaged by the same type of corruption, without creating a mask for each corrupted image. Though this method does not perform as well as with the mask as an input (see Sec. 2.3), it is a good way to inpaint several images at once without the necessity of the user to determine the pixels of an image which are corrupted.

3.5 Limitations

The proposed method performs well if the holes to be filled-in are small. If the holes are too large the inpainting result gets blurry, see the two left images of

corrupted image, PSNR 14.83 | ground truth, PSNR ∞

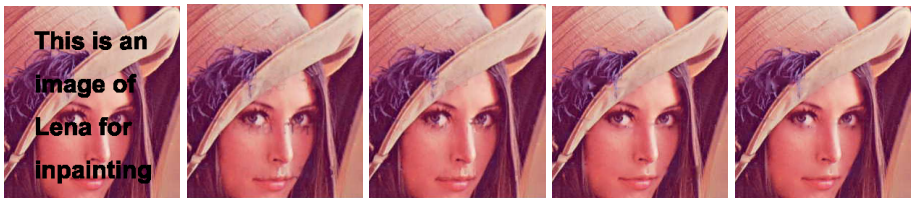
[5] PSNR 26.60 | [2] PSNR 27.56

[20] PSNR 28.87 | [18] PSNR 29.20

[22] PSNR 29.25 | [1], PSNR 30.21

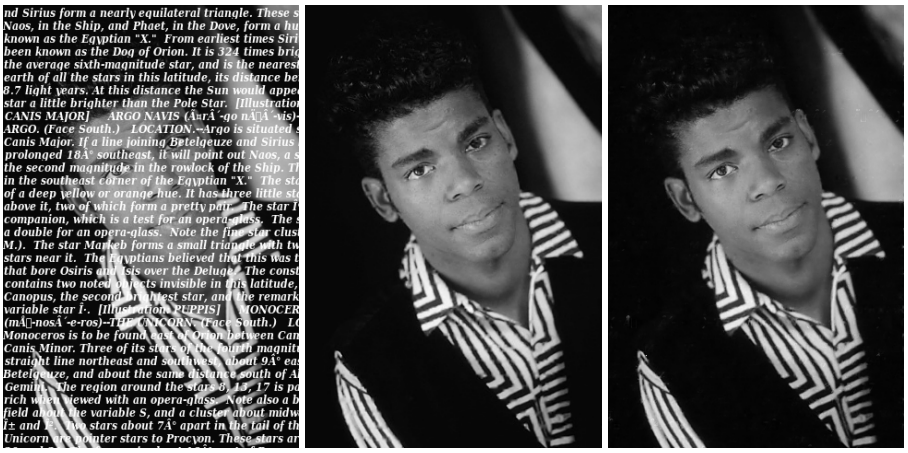
[14] PSNR 32.45 | our approach, PSNR **32.68**

Fig. 5: Comparison on the image "New Orleans" from [2]. While [14] and our approach are close in terms of PSNR, the enlarged image detail (see green box in the corrupted image) shows that our approach is better able to recover the pole.



input, PSNR 13.65 [24], PSNR 34.26 [26], PSNR 35.03 [20], PSNR 35.86 ours, PSNR **36.37**

Fig. 6: Comparison on an image from [26].



corrupted image [20] PSNR 32.13 ours, PSNR **34.22**

Fig. 7: Inpainting an image without knowledge of the exact mask of each image. Only the font and font-size of the masks is known for training the neural net.

Fig. 8. While the sky is inpainted in an appropriate way (since it is smooth), the bridge pylons and the grass on the left part of the image are inpainted in a blurry way. Similarly, in the two right images of Fig. 8 we see that across the grass the inpainted regions appear blurry. This is similar to the phenomenon also present with diffusion-based inpainting methods. This is probably due to the fact that the nonlinear filter learned by the MLP can only propagate a few pixels into the mask, but fails to fill-in larger regions. It seems that the neural net used in this approach is only able to learn to continue image information along isophotes.

4 Towards understanding the trained neural network

A common criticism of methods based on neural networks is that they work like a black box, i.e. even though they are able to reach state-of-the-art performance it remains unclear how the task is solved. To gain insight into how the trained neural network achieves its performance, we study the feature generators for MLPs trained with different distortion types and look how the input feature depend on the shape of the masks.

Fig. 8: Limitations of our approach: large holes (10×10 squares) result in blurry output on textured regions, e.g. at the poles of the bridge. Similar limitations can be seen on the horse image from [26] (line-width: 10 pixels).

4.1 Recognize and play back

Activation maximization [10] finds, for a given neuron in the last hidden layer, the respective input that maximizes that neuron's activation, while constraining the input's L_2-norm. The neurons in the last hidden layer are of interest since they are the *feature generators*, meaning the weights from these neurons to the output layer comprise the features that are linearly superimposed to reconstruct the image. We here analyze the neural net trained for Sec. 3.2. For Fig. 9, we additionally fixed the inpainting mask to be the letter 'e'. The bottom row depicts typical 13x13 outputs of the feature generators, the top row the maximizing 39x39 input. For the Gabor-like feature in the left column, the MLP tries to detect a continuation of the feature outside of the output size, which also motivates the importance of larger input than output patches. In the second column, the feature is not obstructed by the mask and is therefore just copied to the output. The feature in the third column is impossible to reconstruct with the given mask, the result of the activation maximization is only L_2-constrained noise. In the fourth column, the feature is partly visible, however, the input is similar to the previous column, indicating that either the MLP is not yet optimal or that the non-convex activation maximization converged to a local minimum.

Intuitively speaking, it seems that the neural network is able to detect certain basic features which are then generated without missing pixels in the last layers. So image features are detected even though pixels are missing, and played back including the missing information.

4.2 Input features depend on the masks

In Sec. 2.4 we showed that the inpainting performance depends on the masks used for training. To gain additional insight, we can also look at the weights for the first layer, which are shown in Fig. 10. The first four images show weights of a neural network trained on vertical bars, while the second four images on the right were trained on diagonal bars. In general we see that the MLP learns mask specific feature detectors. It would be interesting to better understand the other features that appear; those currently have no obvious interpretation.

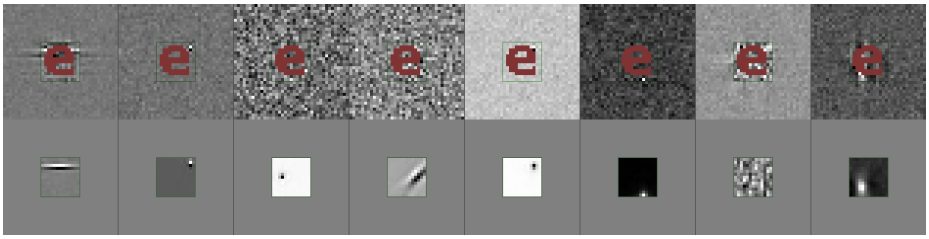


Fig. 9: Input patterns (top row) maximizing the activation of eight of the MLP's feature generators (bottom row) for a given inpainting mask (shown in red), the location of the output patch is marked with a green hairline. The MLP reconstructs features in the output image by trying to find a corresponding input pattern in regions not obstructed by the mask.

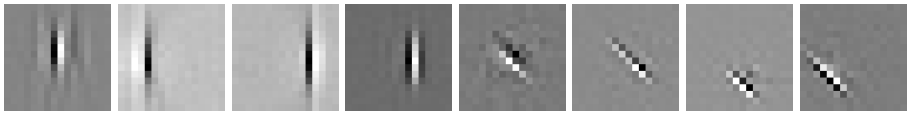


Fig. 10: Selection of weights for the first layer of two MLPs trained on images corrupted with vertical bars (left 4 images) or with 45 diagonal bars (right 4 images). It can be seen that the filters learned by the MLPs are dependent on the distortion type (shape of the mask) given to the MLP.

5 Conclusion

A purely learning-based inpainting approach has the advantage that it is easy to incorporate further information, like the shape of the mask: simply include that information into the input layer and let the training procedure figure out how to make best use of it. Implementing this idea, we are able to show that our inpainting approach based on deep neural networks is able to compete with the currently best inpainting methods that are based on other principles.

Clearly, the mask specific training makes the solution more specific, with the limitation that a trained network will not perform optimally if trained on the wrong mask.

Surprisingly, it is also possible to train the neural network to *blindly* inpaint an image. This is possible if we know the corruption process, e.g. font and font-size, because we can then generate characteristic data to train the neural network. This possibility goes beyond the capabilities of the usual approaches to inpainting which commonly require the exact location of the missing pixels.

References

1. Bertalmio, M., Bertozzi, A.L., Sapiro, G.: Navier-stokes, fluid dynamics, and image and video inpainting. In: Computer Vision and Pattern Recognition (CVPR), 2001 IEEE Conference on. pp. I–355. IEEE (2001)
2. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques. pp. 417–424. ACM Press/Addison-Wesley Publishing Co. (2000)
3. Burger, H.C., Schuler, C.J., Harmeling, S.: Image denoising: Can plain neural networks compete with bm3d? In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. pp. 2392–2399. IEEE (2012)
4. Chan, T.F., Shen, J.: Nontexture inpainting by curvature-driven diffusions. Journal of Visual Communication and Image Representation 12(4), 436–449 (2001)
5. Criminisi, A., Pérez, P., Toyama, K.: Region filling and object removal by exemplar-based image inpainting. Image Processing, IEEE Transactions on 13(9), 1200–1212 (2004)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition (CVPR), 2009 IEEE Conference on. pp. 248–255. IEEE (2009)
7. Drori, I., Cohen-Or, D., Yeshurun, H.: Fragment-based image completion. ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2003 22(3), 303–312 (2003)
8. Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. In: Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on. vol. 2, pp. 1033–1038. IEEE (1999)
9. Elad, M., Starck, J.L., Querre, P., Donoho, D.L.: Simultaneous cartoon and texture image inpainting using morphological component analysis (mca). Applied and Computational Harmonic Analysis 19(3), 340–358 (2005)
10. Erhan, D., Courville, A., Bengio, Y.: Understanding representations learned in deep architectures. Tech. rep., Technical Report 1355, Université de Montréal/DIRO (2010)
11. Fadili, M.J., Starck, J.L., Murtagh, F.: Inpainting and zooming using sparse representations. The Computer Journal 52(1), 64–79 (2009)
12. LeCun, Y., Bottou, L., Orr, G.B., Müller, K.R.: Efficient backprop. In: Neural networks: Tricks of the trade, pp. 9–50. Springer (1998)
13. Liu, Y., Caselles, V.: Exemplar-based image inpainting using multiscale graph cuts. Image Processing, IEEE Transactions on 22(5), 1699–1711 (2013)
14. Mairal, J., Elad, M., Sapiro, G.: Sparse representation for color image restoration. Image Processing, IEEE Transactions on 17(1), 53–69 (2008)
15. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. 8th Int'l Conf. Computer Vision. vol. 2, pp. 416–423 (July 2001)
16. Masnou, S., Morel, J.M.: Level lines based disocclusion. In: Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on. pp. 259–263. IEEE (1998)
17. Pérez, P., Gangnet, M., Blake, A.: Patchworks: Example-based region tiling for image editing. Microsoft Research, Redmond, WA, Tech. Rep. MSR-TR-2004-04 pp. 1–8 (2004)
18. Roth, S., Black, M.J.: Fields of experts: A framework for learning image priors. In: Computer Vision and Pattern Recognition (CVPR), 2005. IEEE Conference on. pp. 860–867. IEEE (2005)

19. Rumelhart, D.E., Hintont, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323(6088), 533–536 (1986)
20. Schmidt, U., Gao, Q., Roth, S.: A generative perspective on mrfs in low-level vision. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. pp. 1751–1758. IEEE (2010)
21. Schuler, C.J., Burger, H.C., Harmeling, S., Scholkopf, B.: A machine learning approach for non-blind image deconvolution. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on. pp. 1067–1074. IEEE (2013)
22. Telea, A.: An image inpainting technique based on the fast marching method. Journal of graphics tools 9(1), 23–34 (2004)
23. Wexler, Y., Shechtman, E., Irani, M.: Space-time completion of video. Pattern Analysis and Machine Intelligence, IEEE Transactions on 29(3), 463–476 (2007)
24. Wong, A., Orchard, J.: A nonlocal-means approach to exemplar-based inpainting. In: Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on. pp. 2600–2603. IEEE (2008)
25. Xie, J., Xu, L., Chen, E.: Image denoising and inpainting with deep neural networks. In: NIPS. pp. 350–358 (2012)
26. Xu, Z., Sun, J.: Image inpainting by patch propagation using patch sparsity. Image Processing, IEEE Transactions on 19(5), 1153–1165 (2010)
27. Zeiler, M.D.: Adadelta: An adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)