A Machine Learning Approach on Restoration of Images with Missing Pixels of Fixed Patterns

Ignasius Ian Savio Gunawan (6613230003-9)
Graduate School of Information Science and Engineering
Ritsumeikan University
Kusatsu, Japan
is0532ps@ed.ritsumei.ac.jp
https://github.com/iansavio/image-restoration

***Abstract*—In the field of computer vision, a single image can contain hefty information. As images are comprised of pixels, some information may be lost if some of those pixels are missing. For instance, a video taken in a camera with dead pixels may produce image frames that have missing pixels in a fixed pattern. Additionally, if the image contains a high number of pixels, losing a small portion will not have a glaring effect. An interpolation technique to fill in the missing pixels can adequately address the problem most of the time. However, should the loss of pixels be large enough that it obscures key regions in the image, it can potentially hinder the performance of Machine Learning models that rely on those pixels. For example, in Simultaneous Localization and Mapping projects, the loss of pixels in key areas may result in a highly inaccurate output. Similarly, projects that are based on object recognition may not be able to work properly as certain features of said object become indistinguishable. Thus, a method that enables such images with missing pixels to be restored is necessary. This project proposes three Machine Learning methods to undertake the problem, namely Convolutional Neural Networks, Long Short-Term Memory, and Bidirectional Long Short-Term Memory. The result suggests that the Convoluted Neural Networks works best in this case of images with missing pixels of fixed patterns.**

***Index Terms*—image restoration, missing pixels, deep learning, Long Short-Term Memory, Convolution Neural Networks**

I. Introduction

An image can contain a hefty amount of information. For example, an image taken from a satellite can show the movement of clouds critical for weather forecasts, provide topographic maps of specific regions to help in urban planning and disaster risk management, and even identify the effects of climate change on observed ecosystems [1]. Thus, in the field of computer vision, a single image can yield vital insights. Nevertheless, in the process of image acquisition, issues related to the visual sensor may lead to missing pixels in the collected images. Dust on the lens or dead pixels in cameras, for example, will produce images with consistent patterns of missing pixels. However, depending on how these images are processed, the loss of some pixels may not significantly impede their usability. In cases where compression is applied, the loss of a small fraction of a high-resolution image might go unnoticed. Additionally, if the number of missing pixels is limited, interpolation techniques can be employed to fill in those gaps. In most instances, such methods can effectively address the issue.

In situations where the loss of pixels substantially obscure crucial regions within an image, it can potentially hinder the performance of Machine Learning models that heavily rely on the consistency of features found in consecutive images. One such example is the Simultaneous Localization and Mapping (SLAM) algorithm [2]. In order for SLAM to work properly, features between consecutive images must be identified and matched. Having feature points abruptly appearing and disappearing will only be a detriment on the final result. Similarly, projects based on object recognition also suffer from the loss of pixels as losing certain distinctive features of the objects may make them indistinguishable and therefore result in a highly inaccurate performance of the model. Thus, it is of utmost importance that effective methods to restore images with missing pixels are developed.

In the context of image restoration, Convolutional Neural Networks (CNN) seem to be one of the most longstanding and widely used techniques. With the way CNNs process data that are fed into them, key features and patterns can be extracted accurately and effectively, making CNNs pivotal in image-related tasks. In 2017, Zhang et al. developed a CNN model that can essentially restore blurry images by removing the noises [5]. In principle, the problem being addressed in this report shares some similarities with the aforementioned research, albeit being fundamentally less complicated. The are two additional methods used in this report. They both assume that missing pixels can be filled by developing models fit for regression problems. Thus, the Long Short-Term Memory (LSTM) model is chosen due to its effectiveness and accuracy in predicting a value based on sequential data. In this case, the pixels are treated as consecutive values over time where every pixel's value influences its next neighboring pixel. Although one could argue that the relationship between pixels should be spatial rather than temporal, the usage of LSTMs and Recurrent Neural Networks (RNN) have been proven to be capable of completing images with more than half of their pixels missing [6]. In this report, a variant of the LSTM called Bidirectional LSTM (BiLSTM) is also used as it can use information that flows forward and backward. By incorporating these architectures, this report aims to explore potential Machine Learning-based solutions for restoring images with missing pixels of fixed patterns. The results are assessed and compared to see which method best fits the task.

TABLE I
CNN ARCHITECTURE SUMMARY

Layer	Number of Filters	Filter Size	Stride	Padding	Activation	Output Size	Parameters
Convolution 1	128	4×4	1	Same	ReLU	8 x 8 x 128	2,176
Average Pooling 1	-	2×2	2	-	-	4 x 4 x 128	0
Convolution 2	256	4×4	1	Same	ReLU	4 x 4 x 256	524,544
Average Pooling 2	-	2×2	2	-	-	2 x 2 x 256	0
Convolution 3	512	4×4	1	Same	ReLU	2 x 2 x 512	2,097,664
Average Pooling 3	-	2×2	2	-	-	1 x 1 x 512	0
Flatten	-	-	-	-	-	512	0
Dropout (0.2)	-	-	-	-	-	512	0
Dense 1	64	-	-	-	ReLU	64	32,832
Dense 2	4	-	-	-	-	4	260
Total Parameters							**2,658,972**

II. ALGORITHMS

In this report, three Machine Learning models are proposed, namely Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM), and Bidirectional Long Short-Term Memory (BiLSTM). Thesee models share a common set of training images containing a total of 1,500 samples. The limited dataset size is due to constraints in time and resources. It is encouraged, however, to expand the dataset whenever feasible. Out of the 1,500 images included, 29 grayscale images are retrieved from the Missing Pixels Database [3], while the rest are sourced from the ImageNet dataset used in the 2012 ImageNet Large Scale Visual Recognition Challenge [4]. Despite variations in sizes and formats, all images undergo the same preprocessing method to make the standardize the final dataset. The preprocessing first transforms each image into grayscale and resizes them to a uniform size of 256 by 256 pixels. In addition, further preprocessing is employed to ensure that each image shares the same pattern for the missing pixels. As depicted in Figure 1, in this experiment, each of the image is separated into blocks of size 8 by 8 pixels. The center 4 pixels are then assigned a certain value (510) above the range of the grayscale pixels (0-255) to indicate their status as missing. Thus, every image will then retain the same fixed pattern of missing pixels. Furthermore, this section details the processes undergone in the model as well as the architecture.

A. Convolutional Neural Networks

For the CNN model, a typical architecture is adopted, which consists of a convolution layer immediately followed by a pooling layer, repeated three times. Afterward, a flatten layer is used to convert the 2D arrays into a single vector, which is then fed to the two dense layers. Before the first dense layer receives the vector, however, a dropout layer is applied to minimize the risk of overfitting. The detailed parameters can be seen in Table I. Regarding the pooling technique, average pooling is chosen over max pooling as the task at hand can be classified as a regression problem in which the goal is to estimate the values of missing pixels. Average pooling is preferred in this case as it preserves the overall information of the pooled region, which should be more suitable in this task where spatial information plays a critical role. Max pooling, on the other hand, is commonly used to accentuate features and may not be as effective in this specific objective. Average pooling, however, should be capable of dealing with this image restoration project as the values of each pixel are likely influenced by their neighboring pixels, indicating that spatial information could be decisive in determining accurate values. The input of this model is the 8 by 8 blocks where the 4 central missing pixels are designated as the targets.

B. Long Short-Term Memory

For this image restoration task, LSTM is proposed due to its ability to intake sequential data to predict certain values. Thus, LSTMs are widely used in tasks that require temporal information to still be retained such as stock price prediction or sentiment analysis in sentences. While both examples typically involve one-way information flow, in this case of image

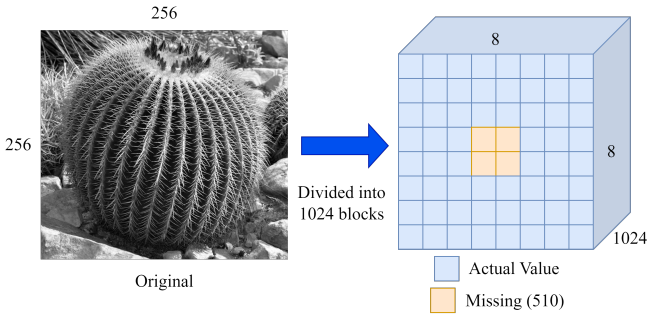

Fig. 1. The pattern of missing pixels.

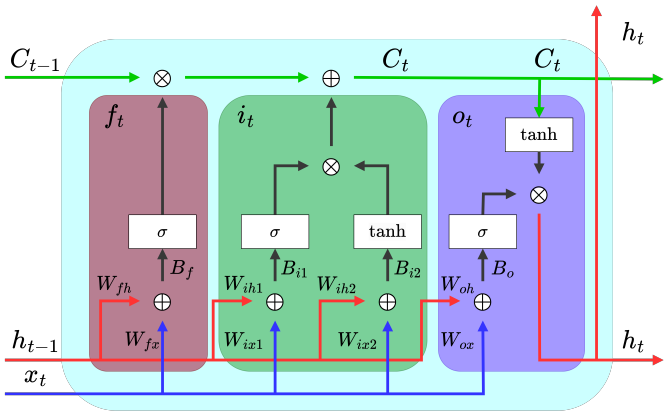

Fig. 2. The structure of an LSTM cell.

restoration, it is assumed that the value of a pixel is affected by the pixels to its left, deliberately converting spatial information into temporal context.

Regarding how the LSTM model works, the Figure 2 provides a reference. An LSTM cell has three main gates that help control the flow of information. These gates help mitigate the problems that traditional Recurrent Neural Networks (RNN) suffer from, vanishing and exploding gradients. By effectively processing information through these gates, the LSTM model can handle sequential data and generate the desired output.

The illustration in Figure 2 depicts how information is processed in each time frame t whereas x, h, and C, each stand for input data, hidden state, and cell state respectively. The sigmoid σ and $tanh$ are activation functions used in this model. Additionally, the following calculations are performed in each gate f_t, i_t, and o_t:

- Forget Gate (f_t)
  This gate determines what information of a specific time t should be forgotten, hence the name. This gate uses σ as the activation function which yields a result from 0 to 1, meaning that this gate decides how much information is retained. The output of the f_t can be derived from:

$$f_t = \sigma((h_{t-1} \cdot W_{fh}) \oplus (x_t \cdot W_{fx}) + B_f), \quad (1)$$

  where the matrix of the previous short-term hidden state h_{t-1} is multiplied by the weight matrix W_{fh} and then concatenated with the matrix of input vector x_t that is multiplied by the weight matrix W_{fx}. After adding it with the bias for the forget gate B_f, it is plugged into the sigmoid activation function σ.
- Input Gate (i_t)
  This gate decides how much information will be relayed to the state at time t. There are two main calculations here in the gate. The one calculated through the σ activation function is to see how much of the update should be kept from the potential long-term memory. The potential long-term memory itself is calculated using the $\tanh$ activation function which yields a result ranging from -1 to 1. The formula is shown as follows:

$$\begin{aligned} i_t = \sigma((h_{t-1} \cdot W_{ih1}) \oplus (x_t \cdot W_{ix1}) + B_{i1}) \\ \otimes \tanh((h_{t-1} \cdot W_{ih2}) \oplus (x_t \cdot W_{ix2}) + B_{i2}), \end{aligned} \quad (2)$$

  where the matrix of the previous short-term hidden state h_{t-1} is multiplied by the weight matrices W_{ih1} and W_{ih2}, so is the input vector x_t by its own weight matrices W_{ix1} and W_{ix2}. Afterward, they are added by each of their biases B_{i1} and B_{i2}. Following the flow from Figure 2, each of those matrices' operation results is then plugged into their respective activation functions. Element-wise multiplication is then applied to both of them. Its output is then added to the calculation for the new cell state C_t.
- Cell State (C_t) Update
  This part cannot really be considered as a gate itself, but the calculation of C_t is necessary to determine the new updated long-term information that has been processed through the two preceding gates. The formula for the calculation of this part can be seen as follows:

$$C_t = f_t \otimes C_{t-1} \oplus i_t \quad (3)$$

- Output Gate (o_t)
  The output gate calculation ends after the result of the sigmoid activation function that determines how much of the short-term information should be kept. It will then be multiplied with the long-term information C_t that has been plugged into the $\tanh$ activation function. The result of the element-wise multiplication is then considered the output h_t of the memory cell. The calculation of this gate is as follows:

$$o_t = \sigma((h_{t-1} \cdot W_{oh}) \oplus (x_t \cdot W_{ox}) + B_o), \quad (4)$$

$$h_t = o_t \otimes \tanh(C_t). \quad (5)$$

  Through this gate, the calculation of the output h_t will then be used in the short-term memory of the next time frame $t+1$ or the final output depending on whether there is any input vector left in the data sequence. If not, then h_t will be the output of the cell.

In this particular experiment, the LSTM's model objective is to predict the values of two consecutive missing pixels in a row. To achieve this, the model uses three consecutive pixels to the left of the missing ones, as illustrated in Figure 3. This approach aims to convert spatial context into temporal information, compensating the LSTM's inherent temporal nature. Organizing the input data in this manner leads to an image of size 256 by 256 pixels yielding 64,152 sequences. As the training process involves 1,500 images, this means that the model is fed with 96,228,000 sequences of pixels. Although there could have been alternative methods to organize the data, the pattern of the missing pixels restricts the model to using only three consecutive pixels to predict the values of the next two pixels in a row. Additionally, the architecture of the LSMT model used in this experiment is detailed in Table II.

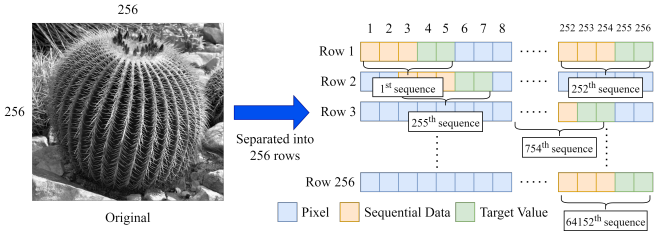

Fig. 3. Sequential data organization of an image for LSTM.

TABLE II
LSTM ARCHITECTURE SUMMARY

Layer	Units	Activation	Output Size	Parameters
LSTM 1	128	Tanh	3×128	66,560
Dropout (0.2)	-	-	3×128	0
LSTM 2	128	Tanh	128	131,584
Dropout (0.2)	-	-	128	0
Dense	2	-	2	258
Total parameters				198,402

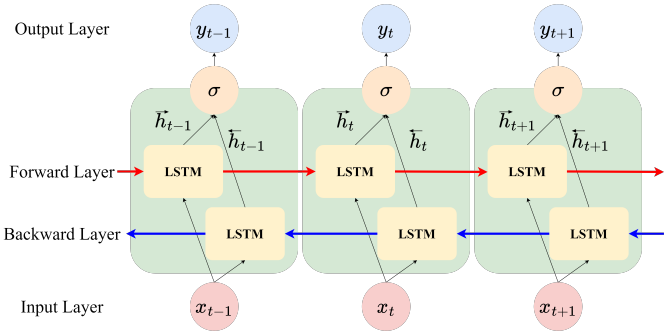

Fig. 4. The layers of BiLSTM.

It follows a stacked LSTM architecture to discover intricate patterns within the sequences. Dropout layers are incorporated to prevent overfitting, enhancing the model's generalization.

C. Bidirectional Long Short-Term Memory

Although traditional LSTMs have proven to be excellent in analyzing sequential data, they still have limitations. The limitation is mainly because LSTMs can only use past information while processing the data. To address this problem, Bidirectional LSTM (BiLSTM), a variant of the LSTM, can be employed. BiLSTMs function similarly to their unidirectional counterpart, utilizing the same three main gates to control information flow. The difference lies in the direction of the data flow. Traditional LSTMs are unidirectional whereas BiLSTMs, as the name suggests, are bidirectional. BiLSTMs consist of two separate LSTM layers, one for the data moving from the past to future, and the other in the opposite direction. This unique architecture allows the network to gather context from both the future and the past, providing it with more insights than a traditional LSTM could possibly achieve.

From Figure 4, three main processes can be observed. The notations used in BiLSTM are also used in LSTM, hence the similarity. Firstly, there is a layer of forward LSTM, which functions similarly to a traditional LSTM. At the same time, the reverse of the forward input is fed to the backward LSTM layer, also working in a manner akin to a traditional LSTM. After calculating the output of each layer, the information is combined, integrating both past and future contexts. Thus, the data should be organized while taking these aspects into account.

In this experiment, similar to LSTM, each row of the image is disassembled, and the three pixels to the right and left of the two missing pixels are used as input whereas the missing pixels are designated as the output, as depicted in Figure 5.

TABLE III
BILSTM ARCHITECTURE SUMMARY

Layer	Units	Activation	Output Size	Parameters
LSTM 1	128	Tanh	3×128	33,792
Dropout (0.2)	-	-	3×128	0
LSTM 2	128	Tanh	128	98,816
Dropout (0.2)	-	-	128	0
Dense	2	-	2	258
Total parameters				132,866

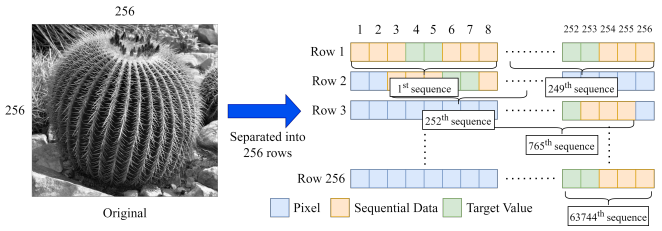

Fig. 5. Sequential data organization of an image for BiLSTM.

This arrangement enables the BiLSTM network to capture the information surrounding the missing pixels. Each image generates 63,744 sequences of pixels, and with 1,500 training samples, a total of 95,616,000 sequences that are fed to the model. Furthermore, to advocate fair comparison between the LSTM and BiLSTM models, the same number of units is used as seen in Table III. The only difference lies in the input size and the resulting number of parameters. The model here, akin to its unidirectional LSTM counterpart, follows a stacked architecture to discover intricate patterns with the addition of dropout layers to prevent overfitting.

III. RESULTS AND DISCUSSION

In this project, a total of four metrics are used to evaluate the models, alongside three test images to observe their performance. The evaluation metrics used in this experiment are Mean Square Error (MSE), Root Mean Square Error (RMSE), Peak Signal to Noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM), each serving specific purpose. MSE quantifies the average squared difference in pixel values between the reconstructed and original images. Higher MSE values indicate poorer model performance, as the metric penalizes larger errors due to its quadratic nature. RMSE is added to facilitate easier and more straightforward interpretations, as its values are in the same units as the ground truth image values. PSNR gauges the ratio between the maximum pixel value and the MSE between the two compared images. The higher the value is, the more the images bear resemblance to one another. As for SSIM, it assesses the structural similarity between two images, in this case, the

TABLE IV
PERFORMANCE ASSESSMENT OF EACH MODEL

Model (Image 1)	**MSE**	**RMSE**	**PSNR**	**SSIM**
CNN	8.2001	2.8636	38.9926	0.9995
LSTM	41.0799	6.4094	31.9945	0.9974
BiLSTM	17.8972	4.2305	35.6030	0.9989
Model (Image 2)	**MSE**	**RMSE**	**PSNR**	**SSIM**
CNN	0.9972	0.9986	48.1430	0.9998
LSTM	5.1621	2.2720	41.0025	0.9988
BiLSTM	1.5658	1.2513	46.1833	0.9997
Model (Image 3)	**MSE**	**RMSE**	**PSNR**	**SSIM**
CNN	4.1670	2.0413	41.9326	0.9994
LSTM	30.8605	5.5552	33.2368	0.9958
BiLSTM	12.5231	3.5388	37.1537	0.9983
Model (Overall Average)	**MSE**	**RMSE**	**PSNR**	**SSIM**
CNN	4.4548	1.9678	43.6894	0.9996
LSTM	25.7010	4.0789	35.4113	0.9973
BiLSTM	10.6620	3.0069	39.9800	0.9989

reconstructed and original images. This index has a range of -1 to 1. The closer the value is to 1 the higher the similarity, while scores near -1 imply dissimilarity.

Three test images from the Missing Pixels Dataset [3] are used to evaluate the models The images are balloon.bmp, bricks.bmp, and WALL.BMP, hereafter referred to as images 1, 2, and 3 respectively. Each test image is reconstructed with its own version of missing pixels, while still sharing the same pattern. Their pixel values are then fed into the models for pixel estimation. The image restoration results can be seen in Figure 6, whereas in Table IV, a comprehensive comparison of the three models' performances is presented. At first glance, based on the images in Figure 6, there does not seem to be any glaring differences between the reconstructed and ground truth images. This is further supported by the SSIM values for each model in all test cases, which are very close to 1. These values suggest that the predicted images by each model bear striking similarities to their respective ground truth image in terms of structural similarity and pixel values. Likewise, the similar PSNR values also imply that the performances of the three models are not substantially different. The overall results imply that using either of these three models would prove to be sufficient for tasks requiring image restoration that is deemed adequate by human visual perception.

Table IV reveals that among the three models, CNN demonstrates the best performance, followed closely by BiLSTM in the second place, and LSTM in the last position. In terms of MSE and RMSE, it shows that CNN has excellent performance across all three test cases, with the BiLSTM performing decently and the LSTM rather poorly.

Upon further inspection of the LSTM predicted pixels, it can be seen that many of them seem to not align with the actual pixel values from the original images in test case 1 and 3. This discrepancy can be attributed to the limited data organization for LSTM as using only three pixels to the left of the two missing pixels restricts the estimation accuracy. The LSTM models seems to have difficulties in predicting pixel values with sudden changes in the sequences, motifs on the balloon and the shadows in the wall being some of the examples. While the first pixel predictions are mostly acceptable, the second one often deviate significantly from the ground truth. However, in regions with gradients, like the sky, and areas with few contrasting colors in motifs, the LSTM seems to perform relatively well in comparison to the other two models, as shown in the second test case.

As for the BiLSTM model, it seems capable of performing better than its unidirectional variant in all scenarios presented in the test cases. This improvement is most likely due to the added context as the sequences are organized such that the two missing pixels are interposed between the three pixels to the left and right. Similar to the LSTM, the BiLSTM model works better in regions with few drastic changes. Although this model can estimate pixel values better in scenarios with contrasting color in motifs, the CNN still outperforms both LSTM-based models. CNN's ability to garner spatial information and context in any direction around the missing pixels allows it to reign supreme in this experiment. This statement is further accentuated by the fact that BiLSTM has better performance than LSTM as the former has more spatial contexts than the latter. A possible solution to improve the LSTM-based methods is to adopt the data organization suggested in [6]. The Diagonal LSTM and Row LSTM can capture information in any direction near the target pixel, albeit the latter's restricted flexibility. Employing those methods may in turn allow the LSTM-based models to retrieve more contextual information, achieving performance on par with or even better than CNN.

Taking practicality into consideration, it can be argued that the CNN might be impractical due to its dependency on fixed artificial patterns shown in Figure 1. This pattern is unlikely to be naturally replicated. The LSTM-based methods, however, can be adjusted according to the specific needs and the pattern of missing pixels, making it better suited to handle various real-world scenarios. While in this experiment the CNN outperformed the other methods, it would be wise to consider the practicality of the models based on the requirements of the image restoration task.

IV. Conclusions and Future Work

This paper proposes three Machine Learning-based approaches as solutions to restore images with missing pixels of fixed patterns. Despite limited training samples of 1,500 images, all three models, namely CNN, LSTM, and BiLSTM, seem to be capable of addressing the problem. The experimental results indicate that CNN works best among the three models, followed by BiLSTM in the second place and LSTM in the last place. Additionally, based on the experiment, it seems that CNN is capable of estimating the values of missing pixels in a fixed pattern. However, uniformity is not always the case in real-world scenarios. In practical situations such as recording videos with lens damage, the image frames yielded from the video should have random patterns of missing pixels. Since the pattern may not be constant, it is quite unlikely that CNN can solve the issue. The LSTM-based methods, however, can be adjusted to handle the random patterns better than CNN. In the future, expanding the dataset should be considered to improve performance of all three models. For LSTM-based methods, other data organization techniques should be explored to provide them more spatial information around the missing pixels.

References

[1] T. W. Gillespie, J. Chu, E. Frankenberg, and D. Thomas, "Assessment and prediction of natural hazards from satellite imagery," *Progress in Physical Geography: Earth and Environment*, vol. 31, no. 5, pp. 459–470, 2007.

[2] X. Gao and T. Zhang, *Introduction to Visual SLAM: From Theory to Practice.* Singapore: Springer, 2022.

[3] D. M. Chandler, *Missing Pixels Database*, July 13, 2022, [Online]. Available: https://github.com/dmc27/missing_pixels.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211-252, 2015, doi:10.1007/s11263-015-0816-y.

[5] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3929-3938.

[6] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *Proceedings of the International Conference on Machine Learning*, 2016, pp. 1747-1756.

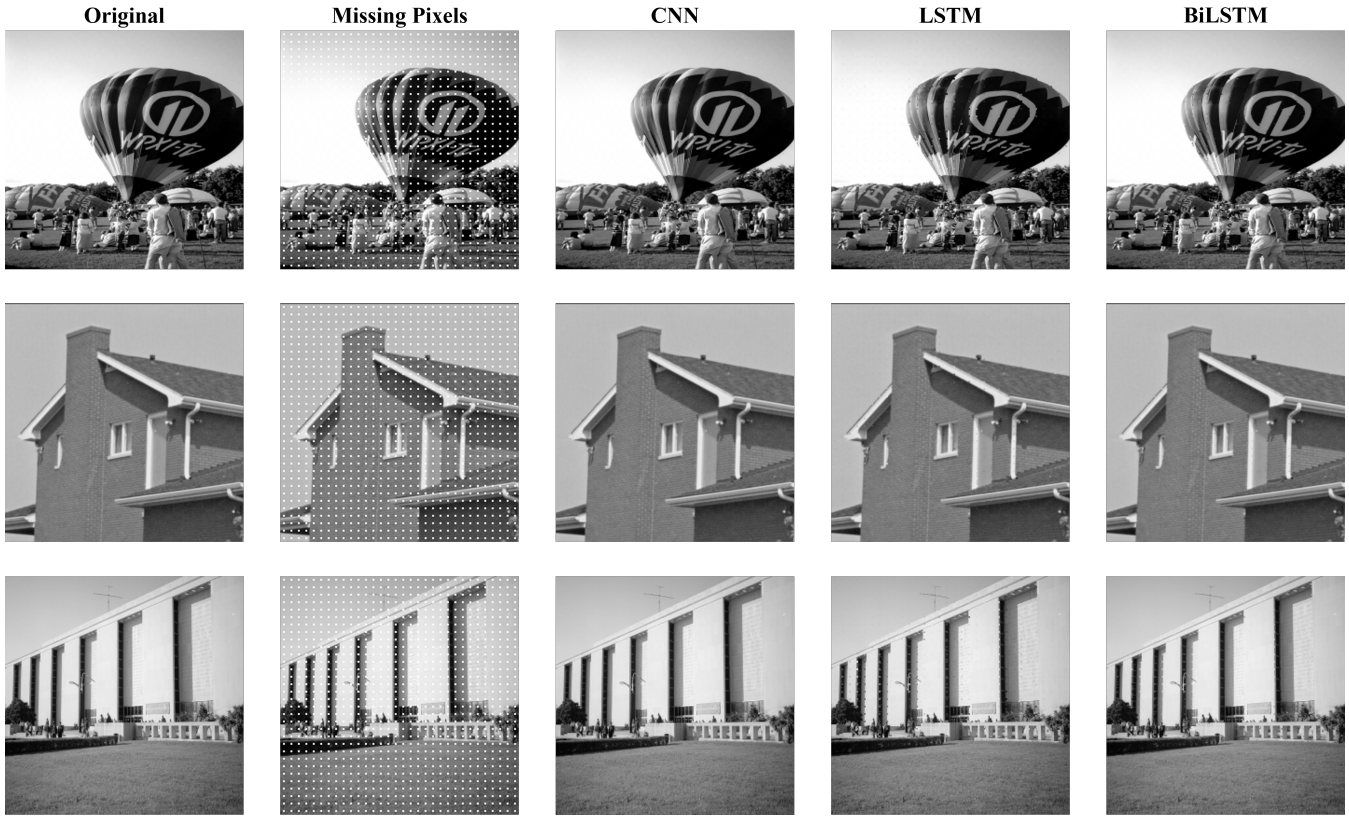

Fig. 6. Comparison of pixel estimation. In the order from top to bottom, the images are referred to as image 1 (balloon.bmp), image 2 (bricks.bmp), and image 3 (WALL.BMP). Images are sourced from the Missing Pixels Database [3].