CoInGP: Convolutional Inpainting with Genetic Programming

Domagoj Jakobovic[1], Luca Manzoni[2], Luca Mariot[3], Stjepan Picek[3]

[1] Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, Zagreb, Croatia
domagoj.jakobovic@fer.hr

[2] Dipartimento di Matematica e Geoscienze, Università degli Studi di Trieste
Via Valerio 12/1, 34127 Trieste, Italy
lmanzoni@units.it

[3] Cyber Security Research Group, Delft University of Technology,
Mekelweg 2, Delft, The Netherlands
{l.mariot,s.picek}@tudelft.nl

Abstract. We investigate the use of Genetic Programming (GP) as a convolutional predictor for supervised learning tasks in signal processing, focusing on the use case of predicting missing pixels in images. The training is performed by sweeping a small sliding window on the available pixels: all pixels in the window except for the central one are fed in input to a GP tree whose output is taken as the predicted value for the central pixel. The best GP tree in the population scoring the lowest prediction error over all available pixels in the population is then tested on the actual missing pixels of the degraded image. We experimentally assess this approach by training over four target images, removing up to 20% of the pixels for the testing phase. The results indicate that our method can learn to some extent the distribution of missing pixels in an image and that GP with Moore neighborhood works better than the Von Neumann neighborhood, although the latter allows for a larger training set size.

Keywords: Genetic Programming · Convolution · Supervised learning · Prediction · Images · Inpainting

1 Introduction

Today, images represent a standard media to share information, but they also represent a common testbed to evaluate the performance of many algorithms, especially those coming from the deep learning domain, see, e.g., [20,9,4]. The usability of images, as well as the results on their prediction, will be impaired if such images are damaged/incomplete. Indeed, having a set of images with missing pixels can severely impact the information they carry or the results that artificial intelligence techniques can obtain. As such, there is often a need to use the image inpainting techniques. Image inpainting denotes techniques that reconstruct lost or damaged parts of images by algorithms that replace such parts of images.

Commonly, there are several directions one can follow for the image inpainting procedure. For instance, patch-based techniques fill in the missing region patch-by-patch by searching for replacement patches in the parts of images that are not damaged [21]. Diffusion-based techniques fill in the missing regions by spreading image information from the boundary to the center of the missing region [12]. More recently, convolutional neural networks (CNNs) have shown excellent results on the image inpainting tasks due to their ability to use large training sets [15]. The part where CNNs truly have an advantage over other image inpainting techniques is the fact that they can better capture the global structure [25]. Finally, researchers use generative adversarial networks (GANs) for many image-to-image translation tasks, where one of them is image inpainting [9]. For a survey of image inpainting techniques, we refer readers to [6]. When considering evolutionary algorithms, there are not many works examining the image inpainting task. Li et al. used a combination of a total variation method and a genetic algorithm for the image completion task [14]. Li and Yang proposed a patch-based method based on evolutionary algorithms that search for the optimal patch in the area around the damaged region [13]. Interestingly, while convolutional neural networks represent state-of-the-art in the image translation tasks, there are not many attempts to bring the power of convolutions to other artificial intelligence techniques. For instance, to the best of our knowledge, there is only a single work that considers how to combine convolutions and genetic programming [19]. There, the authors use their technique to develop image denoising filters.

In this paper, we propose a novel technique for the image inpainting task that is based on Genetic Programming (GP) [11] and convolutions. We denote our approach as CoInGP – Convolutional Inpainting with Genetic Programming. More precisely, our technique uses convolutions to slide over small parts of the image and reconstructs the central missing pixel based on the surrounding pixels. We conduct our experiments with four different test images, and we consider two topologies: Moore and Von Neumann. In our experimental analysis, we removed around 20% of pixels in images, and we successfully reconstructed them with CoInGP. Interestingly, we notice that the Moore neighborhood works better despite the fact that it has smaller training set sizes. Finally, we observe that the edges of images represent the most difficult part for GP to reconstruct, which is to be expected. We believe this work opens an interesting new research direction for genetic programming. Indeed, while GP has been successfully applied in many domains, like symbolic regression [2], scheduling tasks [16], or even network intrusion detection [17], to the best of our knowledge, this is the first time it is considered for the image inpainting task.

The rest of this paper is organized as follows. Section 2 covers the background related to the problem of recovering missing pixels in degraded images. Section 3 describes the details of our CoInGP method, showing how a GP tree can be used as a convolutional predictor for missing pixels and defining the corresponding supervised learning task for. Section 4 explains the experimental settings that we adopted to test our CoInGP method, while Section 5 reports the results of our experiments. Section 6 gives an interpretation of the main experimental findings that can be drawn from our results, and formulate some hypotheses worth exploring to investigate the observed behavior of CoInGP further. Finally, Section 7 sums up the main contributions of our paper and points out some future directions of research on the subject.

2 The Missing Pixels Recovery Problem

We begin by formalizing the problem of reconstructing missing pixels in an image. In what follows, we consider an input image as a matrix I of size $M \times N$, where each entry $x_{(i,j)}$ is the intensity value of the pixel at coordinates (i,j) for $i \in [M]$ and $j \in [N]$, where $[M] = \{1, \cdots, M\}$ and $[N] = \{1, \cdots, N\}$. For illustration purposes, we deal only with 8-bit greyscale images, so that each entry $x_{(i,j)}$ in the matrix is an integer number between 0 and 255; nevertheless, our approach can be straightforwardly generalized to any color depth.

Suppose now that the input image is *damaged*, in particular that the intensities of a subset of its pixels $S = \{(i_1, j_1), \cdots (i_k, j_k)\} \subseteq [M] \times [N]$ are missing. The goal is to recover the original intensities $x_(i_1, j_1), \cdots x_(i_k, j_k)$ starting from those that are still available in the image, i.e., the pixels in the complementary set $P = [M] \times [N] \setminus S$. This task is also known as *inpainting* in the image processing literature [3,8], and one of the possible approaches to perform it stands on the fundamental observation that the intensities of neighboring pixels are *correlated*. In a probabilistic framework, this property can also be restated as the fact that the probability distribution of a pixel's intensity given the intensities of the pixels in its neighborhood is independent of the rest of the image [5].

This observation suggests that, in order to recover the intensity of a missing pixel in an image, one can use just the values of its neighboring pixels as an input for the prediction. More formally, the two main topologies that can be adopted are the *Moore* neighborhood and the *Von Neumann* neighborhood [22]. Considering only neighborhoods of radius 1 (i.e., only the immediate neighbors of a pixels are taken into account), for the Moore neighborhood the input for the prediction of a pixel in position (i,j) will be a 3×3 matrix defined as:

$$\mathcal{N}_{i,j} = \begin{bmatrix} x_{(i-1,j-1)} & x_{(i-1,j)} & x_{(i-1,j+1)} \\ x_{(i,j-1)} & & x_{(i,j+1)} \\ x_{(i+1,j-1)} & x_{(i+1,j)} & x_{(i+1,j+1)} \end{bmatrix} , \tag{1}$$

where the 8 elements on the border represent the intensities of the pixels in the neighborhoods, and the goal is to predict the value of the central pixel. Analogously, for a Von Neumann neighborhood the input to the prediction will be the following matrix:

$$\mathcal{N}_{i,j} = \begin{bmatrix} & x_{(i-1,j)} & \\ x_{(i,j-1)} & & x_{(i,j+1)} \\ & x_{(i+1,j)} & \end{bmatrix} , \tag{2}$$

where, in this case, we do not consider the elements in the corner and the input for predicting the central pixel are only the four elements which are respectively at its top, bottom, left, and right.

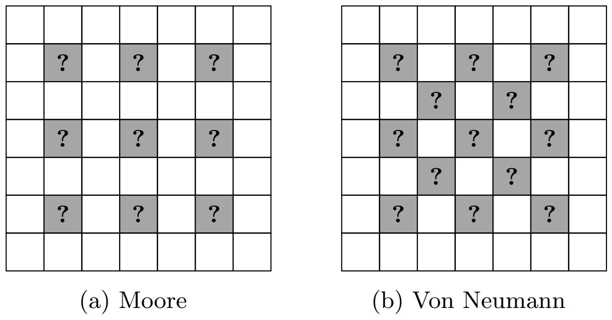


Fig. 1: Densest packings of missing pixels allowed respectively under unitary Moore and Von Neumann neighborhoods.

Intuitively, the quality of the prediction will also depend upon the number of available neighboring pixels: in particular, if also some of the neighboring pixels of $\mathcal{N}_{i,j}$ are missing in the degraded image, then we will have less information at our disposal to predict the central pixel $x_{(i,j)}$. In what follows, we adopt the simplifying assumption that each missing pixel in the degraded image is "sufficiently far" from all other missing pixels, or equivalently that each missing pixel has a complete neighborhood. Formally, in the case of Moore neighborhood this means that the *Chebyshev distance* d_∞ between any pair of missing pixels $(i_{t_1}, j_{t_1}), (i_{t_2}, j_{t_2}) \in S$ must be strictly greater than 1:

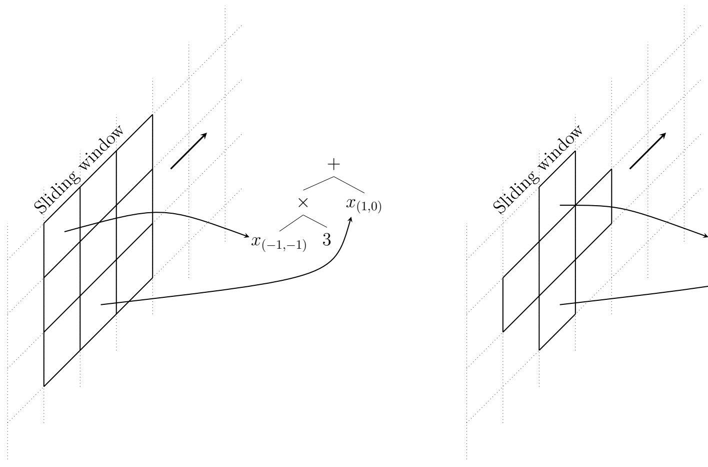
$$d_\infty((i_{t_1}, j_{t_1}), (i_{t_2}, j_{t_2})) = \max\{|i_{t_1} - i_{t_2}|, |j_{t_1} - j_{t_2}|\} > 1 \ .$$

Analogously, for the Von Neumann neighborhood the constraint is that the *Manhattan distance* d_1 between (i_{t_1}, j_{t_1}) and (i_{t_2}, j_{t_2}) has to be greater than 1:

$$d_1((i_{t_1}, j_{t_1}), (i_{t_2}, j_{t_2})) = |i_{t_1} - i_{t_2}| + |j_{t_1} - j_{t_2}| > 1 \ .$$

The consequence of these constraints is that missing pixels can share the frontier of the neighborhood under consideration, but a missing pixel cannot be in the *frontier* of another one. In particular, the frontier of a neighborhood of radius r is defined as the set of pixels that are at distance r from the central one. Since we are only considering the case of radius $r = 1$, the frontier corresponds to the set of all pixels in the neighborhood except the central one. As an example, Figure 1 shows the densest packing of missing pixels that one can have respectively for the Moore and Von Neumann neighborhood. The Von Neumann topology allows for more missing pixels under the same image size, since it includes less neighbors than the Moore topology. Also, observe that for both neighborhoods the missing pixels cannot occur on the border of the image, i.e., $1 < i < M$ and $1 < j < N$ for every missing pixel $(i, j) \in S$.

Although this separation hypothesis does not always hold in realistic scenarios of degraded images, we decided to adopt it to initially validate the feasibility of our method, since, as far as we are aware, no one up to now employed GP to predict missing pixels in images with a convolutional approach.



(a) Moore convolution (b) Von Neumann convolution

Fig. 2: Convolutional prediction based on GP with a Moore and Von Neumann neighborhood of radius 1. The pixels in the frontier of the neighborhood currently looked by the sliding window are fed as input variables to the GP tree, and its output is taken as the predicted value for the central pixel.

3 GP as a Convolutional Predictor

The main idea that we investigate in this paper is to evolve GP trees that act as *convolutional predictors* to solve the missing pixels recovery problem. Similarly to what is done in *Convolutional Neural Networks* (CNN) [7], we assume that the transformation used to predict the values of the missing pixels is *shift-invariant*. This means that we have a *local function* f which is applied over a small *sliding window* of neighboring pixels, and which is shifted one place at a time over the whole image. The output of function f corresponds to the predicted intensity of the pixel at the center of the window in the current position.

In our setting, we consider both the case of a square 3×3 sliding window, which corresponds to the Moore neighborhood of radius 1, and a cross-shaped window of width 3, which amounts to the Von Neumann neighborhood of radius 1. In the former case, the local function has the form $f : [0, 255]^8 \to [0, 255]$, while in the latter it is $f : [0, 255]^4 \to [0, 255]$; either way, the local function is expressed with a GP tree. Thus, the 8 (respectively, 4) intensities of the pixels on the border of the window are taken as terminal nodes of the GP tree, and the value generated at the root node will be the prediction for the central pixel. Figure 2 depicts the idea of using a GP tree as a convolutional predictor by sliding a window over the image for the case of Moore and Von Neumann neighborhoods.

To construct such a convolutional predictor, we need to define an appropriate fitness function that measures how good a particular GP tree is in determining the correct value for the central pixel. The idea is to use the available pixels in the degraded image as a *training set*. Formally, let I be the target image of size $M \times N$, $S = \{(i_1, j_1), \cdots (i_k, j_k)\}$ be the subset of missing pixels that satisfy respectively the Chebyshev distance $d_\infty > 1$ constraint (if the Moore neighborhood is adopted) or the Manhattan distance $d_1 > 1$ constraint (if the Von Neumann neighborhood is used). Further, let $P = [M] \times [N] \setminus S$ be the complementary subset of available pixels. We define the training set as follows:

$$T = \{(\mathcal{N}_{i,j}, x_{(i,j)}) : \ (i,j) \in P, \ 1 < i < M, \ 1 < j < M\} \ , \tag{3}$$

where $\mathcal{N}_{i,j}$ is the punctured neighborhood matrix defined as in Eqs. (1) and (2), respectively when the Moore and Von Neumann neighborhood is used. In other words, for each pixel (i,j) in the available set P (except for those on the border of the image), we construct the corresponding neighborhood matrix $\mathcal{N}_{i,j}$ *without* the value of the pixel in the center, which is used as an input to a GP tree τ, while the actual intensity $x_{i,j}$ of the central pixel (i,j) is kept as the correct label for the training example. Given the output $\hat{x}_{(i,j)} = \tau(\mathcal{N}_{(i,j)})$, we can compute the error that the GP tree τ made in predicting the correct pixel intensity $x_{(i,j)}$. Generalizing to all available training examples, we define the fitness function for the GP tree τ as the *root mean square error* (RMSE) over the training set T:

$$fit(\tau) = \sqrt{\frac{\sum_{(\mathcal{N}_{i,j}, x_{(i,j)}) \in T} (\tau(\mathcal{N}_{i,j}) - x_{(i,j)})^2}{|T|}} \ . \tag{4}$$

Hence, the optimization objective is to minimize fit, since having a GP tree that achieves a small RMSE means that its predictions are close to the actual pixel values. Once the GP evolution process has terminated, the best individual undergoes a *testing phase*, where the tree is used to predict the values of the pixels in the missing set S. The performance of the best tree is then evaluated again with the RMSE measure. Clearly, this approach to testing assumes that the missing set S can be artificially created, so that one can retain the original values of the pixels in it to compute the RMSE.

4 Experimental Setting

To experimentally verify that the proposed approach can improve with respect to simpler methods, we employ four 256×256 grayscale images on which approximately 20% of the pixels were removed. The four images (with the removed pixels) are presented in Figure 3.

We adopt the following procedure to generate the damaged images: each image was first resized to 256×256 pixels, and for every two columns of the image, one was kept unchanged while 100 non-adjacent pixels were randomly removed from the next one. Thus, for each 512 pixels, 100 of them (slightly less than 20%) were removed. The same pattern of removed pixels was employed on all images. This procedure resulted in the removal of 12 700 pixels out of 65 536, corresponding to a percentage of removed pixels of 19.38%. As training samples, all pixels with

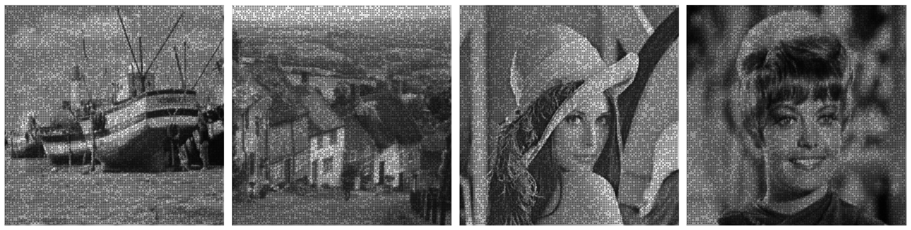


Fig. 3: The damaged images.

a complete neighborhood were used. Due to the different neighborhood shapes considered, the number of fitness cases for the Moore neighborhood was 4 950, and for the Von Neumann neighborhood was 21 036. That is, while the Von Neumann neighborhoods contain fewer pixels, they also allow a larger number of training samples to be employed.

To experimentally assess our method, we performed 100 GP runs for each combination of the four considered images, and the two possible neighborhood topologies. In each GP run, we evolved a population of 500 individuals for 500 generations. The selection phase was performed using tournament selection with a tournament size of 3, where the worst individual is replaced by the offspring generated by applying crossover and mutation on the best two individuals. For the crossover, we adopted simple subtree, uniform, size fair, one-point, and context preserving crossover, randomly selected at each crossover operation. For the mutation, we used a simple subtree mutation with a mutation probability of 0.3 [18]. To avoid bloat, we set the maximum tree depth to 8, which corresponds to the number of input variables available in the Moore neighborhood. The terminal symbols for the GP trees included constants values in the range $[-1, 1]$ and either the 8 (for Moore neighborhood) or 4 (for Von Neumann neighborhood) input variables corresponding to the intensities of the available pixels in the respective neighborhood. The functional symbols for the internal nodes are from the following set: sin, cos, $+$, $-$, $/$ (protected), $*$, min, max, avg, $\sqrt{\cdot}$ and pos. The square root operator is protected so that it returns zero if the argument is negative. Further, the unary operator pos is defined as $\mathrm{pos}(x) = x$ if $x \geq 0$ and 0 otherwise.

Since we require the predicted pixel intensity to be an integer number between 0 and 255, we constrained the output of a GP tree by first clipping it in the interval $[0, 255]$ (i.e., if $|\tau(\mathcal{N}_{i,j})| > 255$ we set $|\tau(\mathcal{N}_{i,j})| = 255$), and then by applying a *linear scaling operator* to obtain the closest integer value, using the method proposed by Keijzer [10]. An alternative solution to force the output of the GP tree to the desired $[0, 255]$ range would be to directly use byte-oriented operators in the functional set, such as bitwise logical operations, modular additions, and rotations. However, we deemed that this approach would have constrained too much the search space explored by GP, hindering its ability to generate good tree predictors with low RMSE fitness values.

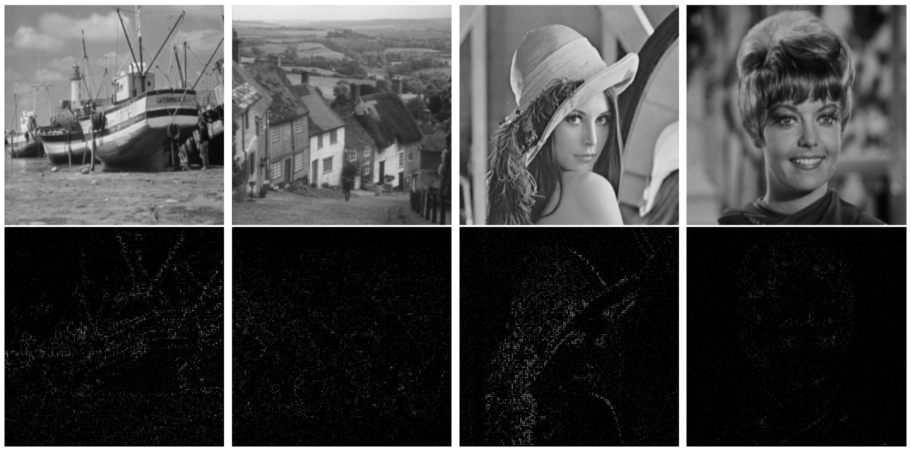


Fig. 4: At the top, the images corrected using the Moore neighborhood. At the bottom, the difference, increased ten times, between the reconstructed and the original image.

5 Results

The results of the reconstruction process are presented in Figure 4 for the Moore neighborhood, and in Figure 5 for the Von Neumann neighborhood. The reconstructed images are all taken from a random GP run. For each image, we also present the pixel-by-pixel difference between the reconstructed image and the original one, where each difference is 10-fold increased in order to make it visible. As it is possible to observe, the errors in both cases are limited (i.e., there are no extremely different pixels), and distributed mainly across the edges of the objects in the image. This is especially visible in the Lena image. There, the distribution of the error almost follows the profile of the face, hairs, and the hat.

Besides qualitative considerations on the reconstructed images, an important aspect is to quantitatively assess whether GP learned to predict missing pixels with respect to a benchmark method. In particular, as a baseline for comparison, we considered two simple predictors that replace each missing pixel with the average intensity of the surrounding pixels, one for the Moore neighborhood and the other for the Von Neumann neighborhood.

To perform this comparison, we plotted the distribution of the best fitness over the 100 experimental runs performed for each image. The results obtained are presented in Figure 6. As a general remark, in most cases, all fitness values obtained are below both baselines, independently of the underlying neighborhood. The only exceptions occur for the Boat and Goldhill images, which, however, are limited to a few outliers. In particular, for those images, a few outliers in the distribution of the Moore neighborhood scored an RMSE value between the two baselines, while a small part of the right tail of the Von Neumann distribution overlaps the corresponding baseline in the Goldhill image. In any case, we notice

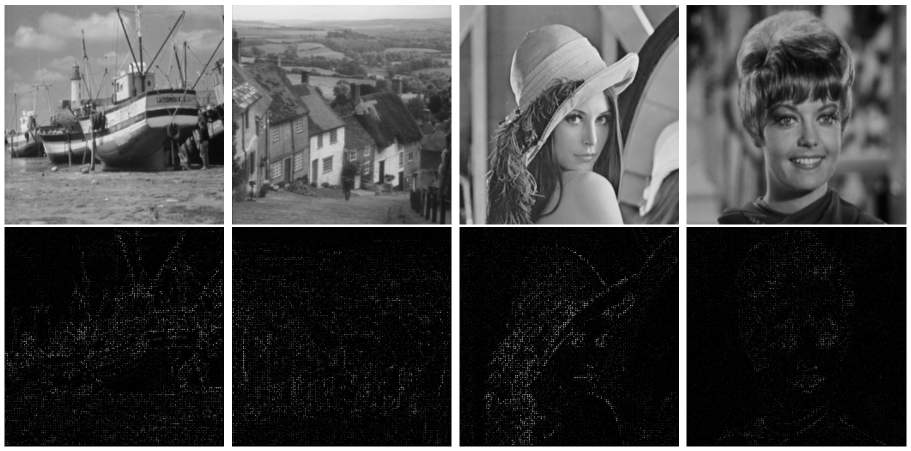


Fig. 5: At the top, the images corrected using the Von Neumann neighborhood. At the bottom, the difference, increased ten times, between the reconstructed and the original image.

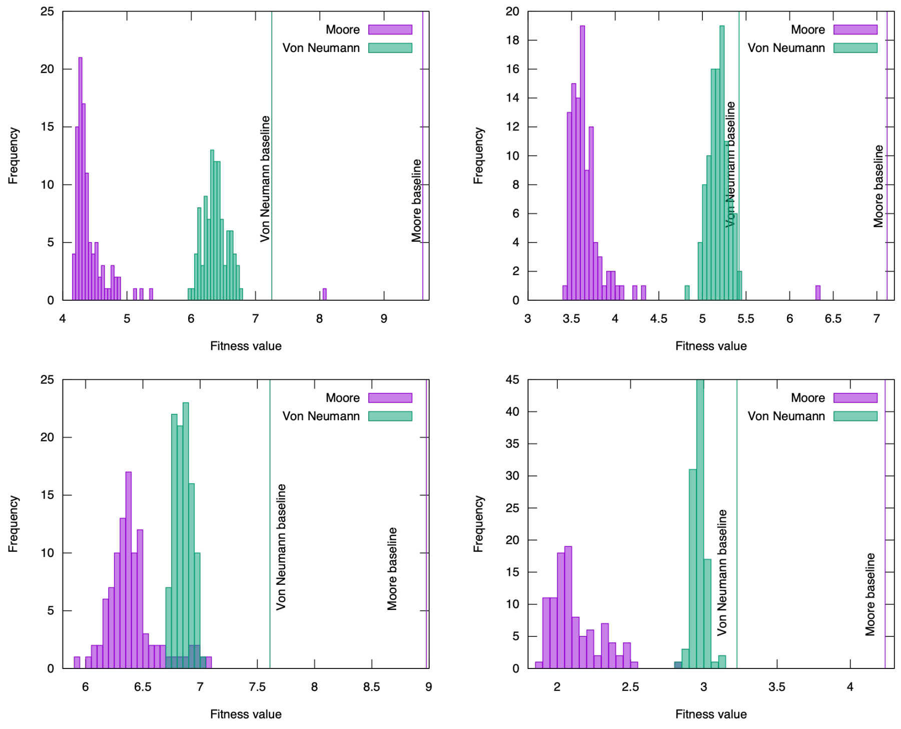


Fig. 6: The distribution of best fitness over 100 runs with both Moore and Von Neumann neighborhoods. From left to right, at the top the results for Boat and Goldhill and at the bottom the results for Lena and Zelda.

that the peaks of all GP distributions are significantly distant from the respective baseline fitness values. Further, in all four test images, the use of the Moore neighborhood produces lower fitness values than the Von Neumann neighborhood, even if it allows fewer training samples to be generated.

6 Discussion

As mentioned in Section 1, to the best of our knowledge, this work is the first one addressing the inpainting problem with GP trees used as convolutional predictors. Hence, the first natural question that may arise is whether the proposed method works. Considering the plots of the best fitness values obtained in our experiments, we can empirically conclude that GP can learn to some extent the distribution of the missing pixels since it achieved lower RMSE values with both neighborhood topologies than the respective baseline predictors. Given our experimental setting, the missing pixels accounted for roughly 20% of the pixels of each test image. The main limitation of our approach is that the training process requires a complete neighborhood, i.e., no missing pixels must occur in the frontier of the central pixel whose value has to be predicted. This limits both the number of missing pixels that one can have in the degraded image and their relative positions. However, the preliminary results that we obtained are promising enough to encourage further improvements in this direction, by extending our method to consider also the case of *adjacent* missing pixels in the degraded image. An interesting idea to accomplish this could be to employ a diffusion-based inpainting approach [12]. In this case, the GP predictor would be first convolved on the border of a missing region, and then gradually shifted towards its interior.

The first remarkable finding when going into the details of our experiments is that the GP predictors based on the Moore neighborhood achieved a better performance (i.e., a lower RMSE value) than those using the Von Neumann neighborhood. This is even though the Von Neumann neighborhood requires fewer input variables to compute the predicted missing pixel, and thus can be optimized on a larger training set. Consequently, this result seems to indicate that GP can learn better using a larger number of input variables and a smaller training set. It would be interesting to investigate if this difference in performances also holds for larger neighborhoods. Still, for radius 2, this would already yield GP trees with 24 and 12 input variables, respectively, for the Moore and Von Neumann neighborhood, thereby increasing both the time needed to perform the training and the size of the resulting GP predictors.

Recall that the baseline predictors simply computed the average of the pixels in the neighborhood to predict the value of the central one. An interesting fact that can be observed from our experiments is that the RMSE achieved by the Von Neumann baseline predictor is lower than that scored by the Moore baseline. Hence, this suggests that the information for predicting the central pixel is not uniformly distributed across the surrounding ones: in particular, it seems that the 4 "diagonal" pixels in the Moore neighborhood contain less information to predict the central one. Nonetheless, this observation is in stark contrast with the fact that GP scored a lower RMSE value with the Moore neighborhood than with the Von Neumann neighborhood. This seems to indicate that GP can learn which pixels in the frontier of the neighborhood are more useful to reach an accurate prediction. It would be interesting to further investigate this issue by analyzing the structure of the trees evolved by GP with the Moore neighborhood.

Finally, from the qualitative point of view, we observed that the prediction errors made by GP individuals mostly focused around the edges in the test images. This is an expected side effect: if one considers images as two-dimensional spatial signals, edges correspond to *high-frequency* regions, where abrupt changes of the intensity value occur among neighboring pixels. Consequently, the pixels' intensities in a neighborhood where an edge occurs have a lower correlation, and the independence hypothesis that the probability distribution of a pixel given the surrounding ones is independent of the rest of the image does not hold. This explains why our GP convolutional predictor obtains a higher error in the proximity of an edge. However, this is not necessarily a negative effect. One could exploit this observation to use our GP-based convolutional inpainting method to perform edge detection as a by-product. Further, an interesting idea to decrease the prediction error on the edges would be to develop a 2-layer architecture: the first layer would be used to detect the edges, while the second one would perform the inpainting task by discriminating between the edge and non-edge pixels. For the latter case, one could, for instance, evolve GP trees over a larger neighborhood, so that more information can be used to predict the central pixel.

7 Conclusions

In this paper, we proposed a method for performing convolutional inpainting with GP – CoInGP. The main idea is to sweep a small sliding window over a degraded image with missing pixels, where the neighborhood pixels captured by the window are fed as input to a GP tree, whose output is then taken as the predicted value for the central pixel. Given a degraded image with missing pixels, we cast the problem of evolving an optimal GP tree predictor to repair it as a supervised learning task, where the training set is composed of all available pixels. For each position of the sliding window, the central pixel is removed and replaced with the value predicted by a GP tree. The fitness function is the RMSE between the original pixel intensities and those predicted by the GP tree, which has to be minimized. The testing phase consists of applying the best tree evolved by GP on the actual missing pixels. We experimented with four test images and two different topologies for the sliding window, namely the Moore neighborhood and the Von Neumann neighborhood. The results showed that GP could evolve trees with better prediction accuracy than the respective baseline predictor. In particular, we observed a clear performance difference between the Moore and the Von Neumann neighborhood, with the former achieving lower RMSE scores than the latter on the test sets. This is in spite of the fact that the Von Neumann baseline predictor has a lower RMSE than the Moore one, pointing to the fact that GP is able to learn how to appropriately weight the information contained in the pixels at the corners of the Moore neighborhood.

We conclude by pointing out directions for future research besides those already discussed in the previous section.

The experiments presented in this paper suggest that using GP as a convolutional predictor represents an interesting building block to be plugged in more complex architectures for supervised learning tasks in the image domain. We sketched the first idea of this approach in Section 6, where we proposed to use a first GP convolutional layer for detecting the edges in an image and then use the second layer to perform inpainting. It would thus be interesting to generalize this concept to multiple GP-based convolutional layers and see how the performance of the overall system compares to other analogous and more established methods such as deep CNNs. Besides the inpainting technique, one could also consider the application of GP to other image processing tasks that can be formulated as supervised learning problems. This includes not only tasks where the training has to be performed on a single target image as in the inpainting case, but also on multiple images, such as image classification. In particular, this would likely benefit from the use of a multi-layered architecture where each GP-based convolutional layer would be used to extract a particular feature of an image.

Further, the convolution strategy is general enough that can be, in principle, applied to any kind of learning task in the signal processing domain. In this paper, we addressed the use case of images, which can be considered as two-dimensional spatial signals, but it could be interesting to explore how convolutional GP also behaves on one-dimensional signals such as time series. In particular, the problem of predicting missing data in general signals (which corresponds to inpainting in the image case) is also known as *imputation*, which is quite useful in the context of symbolic regression over incomplete datasets. As far as we know, there are a few works in the literature addressing the imputation problem using GP [23,24,1], but none of them use a convolutional approach like the one proposed in this paper.

References

1. Al-Helali, B., Chen, Q., Xue, B., Zhang, M.: Hessian complexity measure for genetic programming-based imputation predictor selection in symbolic regression with incomplete data. In: Genetic Programming - 23rd European Conference, EuroGP 2020, Held as Part of EvoStar 2020, Seville, Spain, April 15-17, 2020, Proceedings. pp. 1–17 (2020)
2. Augusto, D.A., Barbosa, H.J.C.: Symbolic regression via genetic programming. In: Proceedings. Vol.1. Sixth Brazilian Symposium on Neural Networks. pp. 173–178 (2000)
3. Bugeau, A., Bertalmío, M., Caselles, V., Sapiro, G.: A comprehensive framework for image inpainting. IEEE Trans. Image Processing **19**(10), 2634–2645 (2010)
4. Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., Su, J.: This looks like that: Deep learning for interpretable image recognition. In: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing

Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada. pp. 8928–8939 (2019)

5. Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. In: Proceedings of the International Conference on Computer Vision, Kerkyra, Corfu, Greece, September 20-25, 1999. pp. 1033–1038 (1999)
6. Elharrouss, O., Almaadeed, N., Al-Maadeed, S., Akbari, Y.: Image inpainting: A review. Neural Processing Letters (Dec 2019). https://doi.org/10.1007/s11063-019-10163-0, http://dx.doi.org/10.1007/s11063-019-10163-0
7. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
8. Guillemot, C., Meur, O.L.: Image inpainting : Overview and recent advances. IEEE Signal Process. Mag. **31**(1), 127–144 (2014)
9. Isola, P., Zhu, J., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 5967–5976 (2017)
10. Keijzer, M.: Improving symbolic regression with interval arithmetic and linear scaling. In: Genetic Programming, 6th European Conference, EuroGP 2003, Essex, UK, April 14-16, 2003. Proceedings. pp. 70–82 (2003)
11. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
12. Li, H., Luo, W., Huang, J.: Localization of diffusion-based inpainting in digital images. IEEE Transactions on Information Forensics and Security **12**(12), 3050–3064 (2017)
13. Li, K., Yang, Z.: Exemplar image completion based on evolutionary algorithms. In: 2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies. pp. 696–701 (2013)
14. Li, K., Wei, Y., Yang, Z., Wei, W.: Image inpainting algorithm based on tv model and evolutionary algorithm. Soft Computing **20**, 885–893 (2016)
15. Liu, G., Reda, F.A., Shih, K.J., Wang, T.C., Tao, A., Catanzaro, B.: Image inpainting for irregular holes using partial convolutions (2018)
16. Nguyen, S., M.Y..Z.M.: Genetic programming for production scheduling: a survey with a unified framework. Complex Intell. Syst. **3**, 41–66 (2017)
17. Picek, S., Hemberg, E., Jakobovic, D., O'Reilly, U.M.: One-class classification of low volume dos attacks with genetic programming. In: Banzhaf, W., Olson, R.S., Tozier, W., Riolo, R. (eds.) Genetic Programming Theory and Practice XV. pp. 149–168. Springer International Publishing, Cham (2018)
18. Poli, R., Langdon, W.B., McPhee, N.F.: A Field Guide to Genetic Programming. lulu.com (2008)
19. Rodriguez-Coayahuitl, L., Morales-Reyes, A., Escalante, H.J.: Convolutional genetic programming. In: Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F., Olvera-López, J.A., Salas, J. (eds.) Pattern Recognition. pp. 47–57. Springer International Publishing, Cham (2019)
20. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
21. Ružić, T., Pižurica, A.: Context-aware patch-based image inpainting using markov random field modeling. IEEE Transactions on Image Processing **24**(1), 444–456 (2015)

22. Schiff, J.L.: Cellular automata: a discrete view of the world, vol. 45. John Wiley & Sons (2011)
23. Tran, C.T., Zhang, M., Andreae, P.: Multiple imputation for missing data using genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015. pp. 583–590 (2015)
24. Tran, C.T., Zhang, M., Andreae, P., Xue, B.: Multiple imputation and genetic programming for classification with incomplete data. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2017, Berlin, Germany, July 15-19, 2017. pp. 521–528 (2017)
25. Yan, Z., Li, X., Li, M., Zuo, W., Shan, S.: Shift-net: Image inpainting via deep feature rearrangement. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. pp. 3–19. Springer International Publishing, Cham (2018)