Hypernetwork functional image representation

Sylwester Klocek, Łukasz Maziarka, Maciej Wołczyk, Jacek Tabor, Jakub Nowak, and Marek Śmieja

Faculty of Mathematics and Computer Science
Jagiellonian University
Łojasiewicza 6, 30-348, Kraków, Poland
marek.smieja@ii.uj.edu.pl

Abstract. Motivated by the human way of memorizing images we introduce their functional representation, where an image is represented by a neural network. For this purpose, we construct a hypernetwork which takes an image and returns weights to the target network, which maps point from the plane (representing positions of the pixel) into its corresponding color in the image. Since the obtained representation is continuous, one can easily inspect the image at various resolutions and perform on it arbitrary continuous operations. Moreover, by inspecting interpolations we show that such representation has some properties characteristic to generative models. To evaluate the proposed mechanism experimentally, we apply it to image super-resolution problem. Despite using a single model for various scaling factors, we obtained results comparable to existing super-resolution methods.

Keywords: Hypernetwork · Image representation · Deep learning.

1 Introduction

Classical machine learning approaches are based on optimizing a predefined class of functions, such as linear or kernel classifier [32], Gaussian clustering [3], least squares regression [14], etc.. With the emergence of deep learning, we learned how employ general nonlinear functions given by complex neural networks. Replacing shallow methods by deep neural networks opened new opportunities in machine learning, because arbitrarily complex functions can be approximated [4].

However, when one considers a typical representation of the data, we still use a somehow shallow and restrictive vector approach. Although real data, such as sound or image, have analog character, we represent them in an artificial vector form. In consequence, one cannot easily access an arbitrary position of the image, rescale the image or perform (even linear) operations like the rotation [12] [26], [10] without the additional use of interpolation [13]. Concluding, it is impossible to satisfactory utilize advanced deep learning methods without creating natural mechanisms for data representation.

The aim of this paper is a proof of concept that one can effectively construct and train *functional representations* of images. By a *functional (or deep) representation of an image* we understand a function (neural network) $\mathcal{I} : \mathbb{R}^2 \to \mathbb{R}^3$

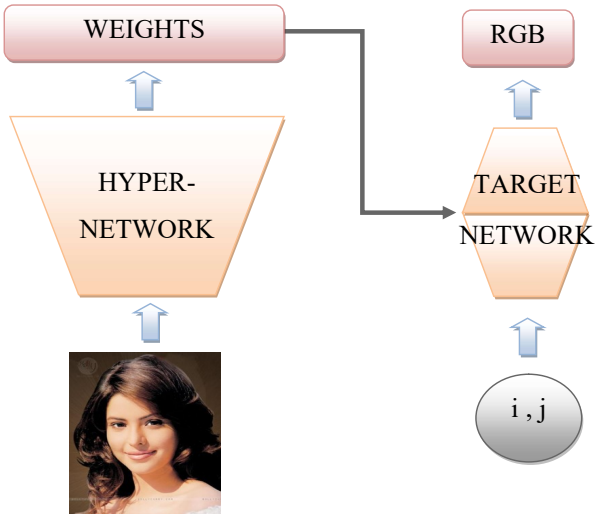


Fig. 1. The scheme of our approach. The hypernetwork takes an image and produces the weights to target network, which is responsible for approximating an image at every real-valued coordinate pair $(i, j) \in [0, 1]^2$.

which given a point (with arbitrary coordinates) (x, y) in the plane returns the point in $[0, 1]^3$ representing the RGB values of the color of the image at (x, y). Observe that given a functional image representation, we can easily obtain a corresponding vector representation by taking $(\mathcal{I}(i, j))_{i=1..k,j=1..n}$. In constrast to the vector form, the functional image representation in its idea can be compared by the way a human represents the image[1].

The main achievement of the paper is the functional image representation constructed with the use of hypernetwork [16]. The hypernetwork takes an image and produces a target neural network, which is an approximation of the input image, see Figure 1 for illustration. Instead of creating the whole architecture of the target network from scratch, the hypernetwork returns only the weights to its fixed, predefined architecture. This allows us to effectively train both hypernetwork and target networks at the same time, using stochastic gradient descent.

We summarize the advantages and applications of our representation:

1. We use a single hypernetwork, which gives a recipe of how to represent images by target networks. In consequence, we return the functional representation of any input image at test time without additional training.
2. It is a well-known fact that true images represent a manifold embedded in high-dimensional space [41]. In consequence, it is difficult to interpolate between two images in the pixel space and not to fall out of the distribution

[1] we can reasonably hypothesise that a human representation of an image in the memory is given by some neural network

Fig. 2. A linear interpolation between weights of two target networks (upper row) compared with a typical pixel-wise interpolation (bottom row). Images produced by target networks with interpolated weights corresponds to natural images coming from true data distributions. There in no superimposition of images, unlike in the case of pixel-wise interpolation.

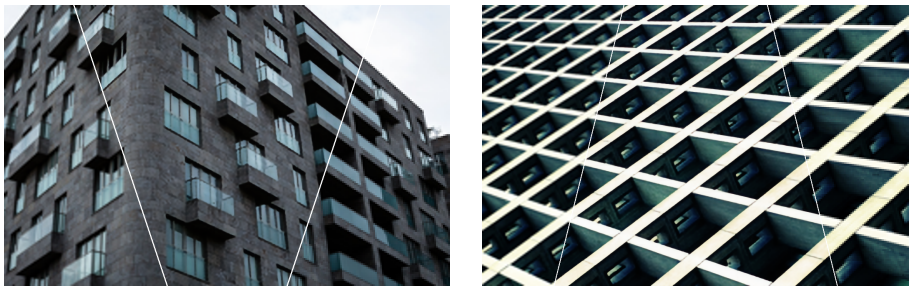


Fig. 3. Higher resolution images (scaling factor $\times 4$) obtained with a use of the bicubic interpolation (left) and our method (middle). A low-resolution input image is on the right side.

of true images. In contrast, the geometry of target networks' weights is less complex (Section 4.1). We have verified that simple linear interpolation between weights of target networks representing images leads to natural images (Figure 2).

3. Due to the continuity of this representation, we are not limited to a fixed resolution, but can operate on a continuous range of coordinates. To confirm this property, we applied our approach to a super-resolution task (Figure 3). In contrast to typical super-resolution methods [44], [12], which are trained for a single scaling factor, our model is able to upscale the image to any size. While its effects are competitive compared to existing approaches (Section 4.2), we can also use non-standard scales at test time (Figure 4).

2 Related work

We briefly outline related approaches. We start with describing the hypernetwork model. Next, we discuss typical methods for image representation. Finally, we move on to super-resolution techniques.

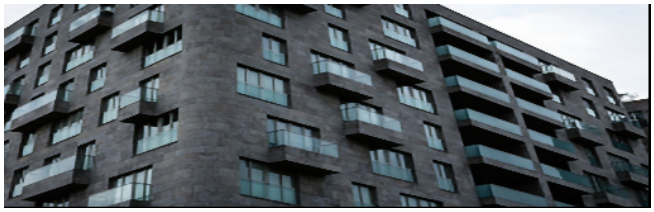


Fig. 4. Resizing the image to non-standard 2.5×1 resolutions.

Hypernetworks were introduced in [16] to refer to a network generating weights for a target network solving a specific task. The authors aimed to reduce the number of trainable parameters by designing a hypernetwork with a smaller number of parameters than the target network being generated. Making an analogy between hypernetworks and generative models, the authors of [33], used this mechanism to generate a diverse set of target networks approximating the same function. This is slightly similar to our technique, but instead of creating multiple networks for the same task, we aim at generating individual network for each task (image). In [33], the hypernetwork was used to directly maximize the conditional likelihood of target variables given a certain input. Moreover, hypernetworks were also applied in Bayesian context [22], [30]. Finally, the authors of [6], [43], [29] used the hypernetwork mechanism to create or improve the search of the whole network architecture.

Images are commonly represented as two-dimensional matrices with a fixed size (resolution). Due to the high redundancy in image features, one can use auto-encoders to create compressed representation in lower-dimensional space [2]. By adding controlled noise to input data at training stage, we can obtain a representation, which is less sensitive to the image perturbations at test stage [38]. Auto-encoders can also be used as a basis for generative models, such as VAE [21] or WAE [36], which can learn an even more compact representations and generate new images using a decoder network [27]. Although auto-encoders can be used to transfer knowledge to less explored domains [39], the representations they learn are in a vector space. To represent an image as a function, one can approximate pixel intensities by a regression function, e.g. polynomial or kernel regression [35], or interpolate between neighbor pixels, e.g. bicubic or B-spline interpolation [18], [13], [37]. More advanced approaches rely on using wavelets [24] or ridgelets [11], which play a key role in JPEG2000 compressor [7]. Nevertheless, the aforementioned methods require manual selection of the regression model, which makes them difficult to use in practice. To overcome these problems we follow deep learning approach and allow the hypernetwork to select the optimal function based on the data set.

Super-resolution area has been dominated by deep learning models. One of the earliest approaches applied a lightweight convolutional neural network to directly map a low-resolution image to its high-resolution counterpart [12]. In contrast, the authors of [20] used a very deep convolutional network inspired

by the VGG Net to predict residuals instead of the output image itself. The paper [23] introduced a discriminator network as in the case of GANs to make upscaled images more realistic. In [25] the authors reused residual networks, but removed unnecessary modules, which resulted in increasing the speed and model stabilization. The authors of [44] also used residual networks, but exploited the hierarchical features from all the convolutional layers instead of the last one. Despite a huge progress in super-resolution area, most of existing models are trained for a single scale factor. In contrast, the proposed hypernetwork technique allows us to upscale the image to multiple sizes using only a single hypernetwork.

3 Functional image representation

In this section, we introduce our functional image representation. First, we describe our learning model and define its cost function. Next, we discuss the architectures of the hypernetwork and the target network.

3.1 Hypernetwork model

Let $f : [0,1]^2 \to [0,255]^3$ be a function describing the image. In practice, we only observe pixel intensities in a fixed grid. To improve this discrete representation, we aim at creating a function:

$$T_\theta(i,j) = T((i,j),\theta) : [0,1]^2 \times \Theta \to [0,255]^3,$$

which approximates RGB values of each coordinate pair $(i,j) \in [0,1]^2$. Our objective is to find an optimal weight vector $\theta \in \Theta$ for every image.

In the simplest case, T_θ can be obtained by linear or quadratic interpolation [13], which however may not be sufficient for approximating complex image structures. To achieve higher flexibility, one could model every image with a specific neural network T_θ. Nevertheless, training separate networks for each image using backpropagation may be computationally inefficient.

We approach this task by introducing a hypernetwork

$$H_\varphi : X \ni x \to \theta \in \Theta,$$

which for an image $x \in X$ returns weights θ to the corresponding target network T_θ. Thus, an image x is represented by a function $T((i,j); H_\varphi(x))$, which for any coordinates $(i,j) \in [0,1]^2$ returns corresponding RGB intensities of the image $x \in X$.

To use the above model, we need to train the weights φ of the hypernetwork. For this purpose, we minimize classical mean squared error (MSE) over training images. More precisely, we take an input image $x \in X$ and pass it to H_φ. The hypernetwork returns weights θ to target network T_θ. Next, the input image x

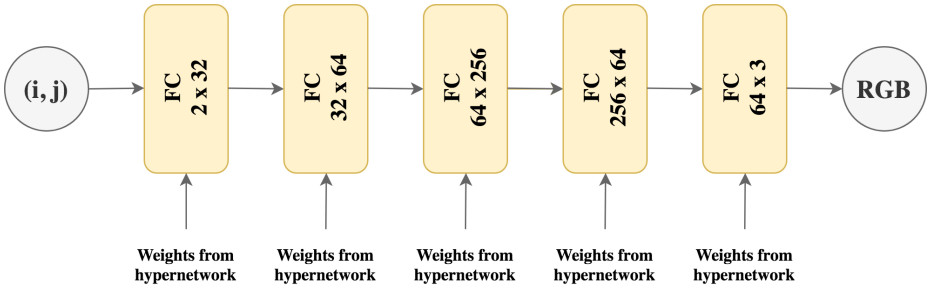


Fig. 5. The architecture of the target network.

is compared with the output from target network T_θ over known pixels. In other words, we minimize the expected MSE over the training set of images:

$$MSE = \sum_{x \in X} \sum_{(i,j)} [x[i,j] - T((i,j); H_\varphi(x))]^2.$$

Observe that we only train a single neural model (hypernetwork), which allows us to produce a great variety of functions at test time. In consequence, we might expect that target networks for similar images will be similar (see experimental section). In contrast, if we created a single network for every image, such identification would be misleading.

3.2 Architecture

In this part, we present the architectures used for creating functional image representation.

Target network. An architecture of the target network is supposed to be simple and small. This allows us to keep the performance of training phase at the highest possible level as the target network is not directly trained. Moreover, small networks can be easily reused for other applications.

Target network maps a pair of two coordinates to corresponding three dimensional RGB intensities. The target network consists of five fully-connected layers, see Figure 5 with biases added in each of them. The layers' dimensions are being gradually increased. This is happening up to the middle layer. Later on, they are being decreased. This is because steep transitions of layers' dimensions negatively affect the learning ability of neural network. The matrices used in the target network have the following dimensions: 2x32, 32x64, 64x256, 256x64, 64x3. Additionally, batch normalization is used after each layer [19]. We have chosen cosine to be the activation function between two consecutive layers. This choice is motivated by a typical approach used in mathematical image transformation, which is based on discrete cosine transform (DCT), see JPEG compression. For

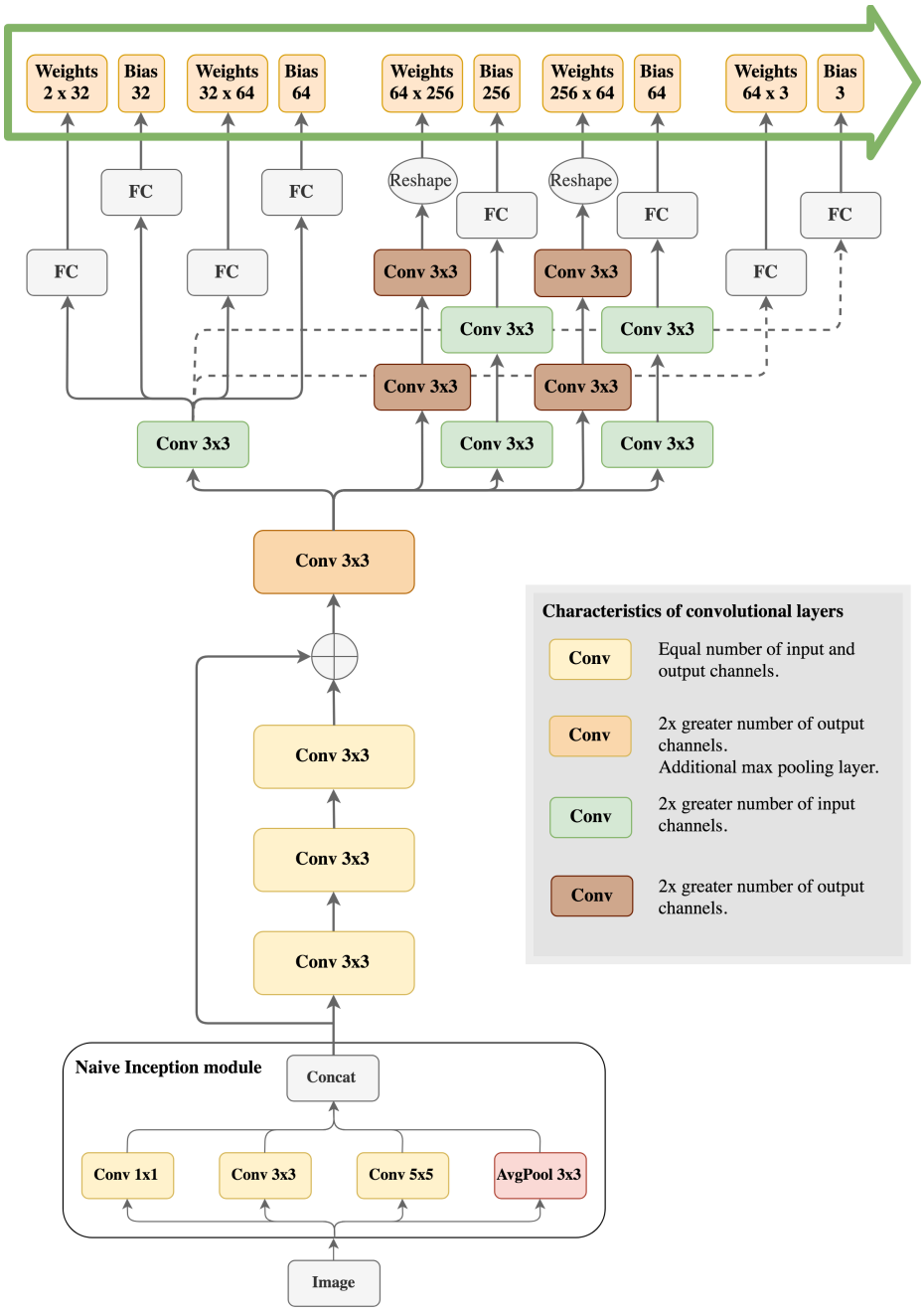


Fig. 6. The architecture of the hypernetwork.

our purposes bounded cosine function worked much better than ReLU, which can give arbitrary high outputs[2]. We use sigmoid as the activation function for the last layer. No convolutions were used, because the input of the target network is too simple.

Hypernetwork. The hypernetwork used in our model is a convolutional neural network with one residual connection, see Figure 6. There are two main parts of the hypernetwork's architecture. The first one is common and takes part in generating weights for all of the target network's layers. Second part, on the other hand, contains several branches. Each branch calculates weights for a different layer of target network. This approach enabled faster training, compared to creating a separate hypernetwork for each layer of the target network.

The task of common layers is to extract meaningful features. This extraction is performed using Naive Inception Module [34] followed by four convolution layers. Inception module leverages three different convolutions and average pooling. It improves network accuracy and does not negatively influence training time. At the very end of common part, we introduced max pooling to decrease the number of features.

Next, there are multiple branches responsible for converting extracted features into actual weights of target network. Dimensions of the target network's layers vary. Therefore, convolutions with different number of input and output channels are used in the branches. Their role is to adjust the sizes of tensors. Shapes of tensors also need to be modified, hence convolutions are followed by either fully-connected layers or simple reshapes. Fully-connected layers are used in branches generating weights for smaller layers of target network. Batch normalization is used after each layer of hypernetwork and ReLU is chosen as the activation function.

Finally, to accelerate training phase even more, we reduced number of trainable parameters by adapting an approach from inception network [8]. More precisely, we replaced each $n \times n$ convolution with $1 \times n$ convolution followed by a $n \times 1$ convolution.

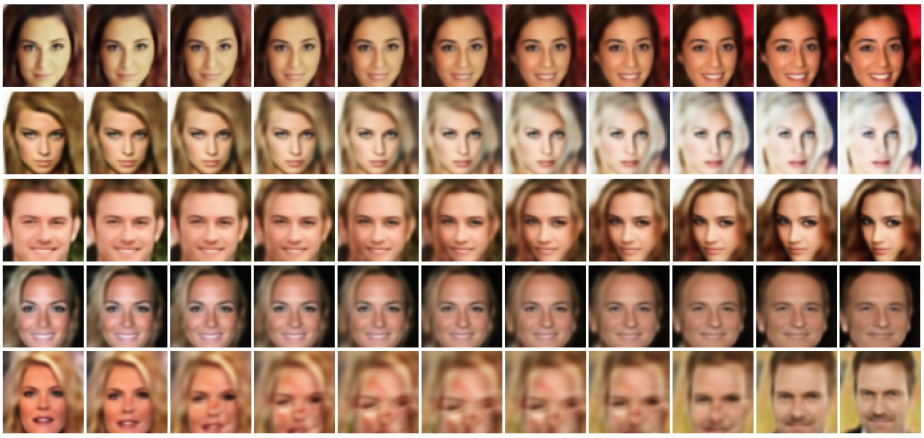
4 Experiments

In this section, we examine the proposed functional image representation. First, we analyze the space of target networks' weights and show some interesting geometrical properties. Next, we use the continuity of our representation and apply it to super-resolution.

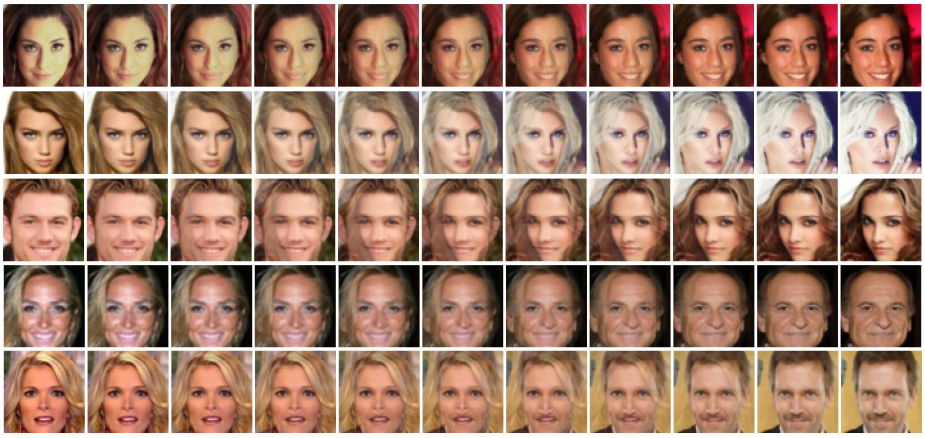
4.1 Target networks geometry

It is believed that high dimensional data, e.g. images, is embedded in low-dimensional manifolds [15]. In consequence, direct linear interpolation between images does not produce pictures from true data distributions.

[2] Other experimental studies report that there are not much difference between using cosine and ReLU as activity function [15].



(a) Interpolation between target networks weights.



(b) Pixel-wise interpolation.

Fig. 7. Comparison of the interpolation between weights of two target networks (a) and linear pixel-wise interpolation (b). We also include an example of unsuccessful interpolation (last row), where interpolation went beyond the manifold of true images.

Fig. 8. Layerwise interpolation on the CelebA. In i-th row we interpolate only over the first i layers of the network.

In this experiment, we would like to inspect the space of target network weights. In particular, we verify whether linear interpolation between weights of two target networks produces true images. We use the CelebA data set [28] with trivial preprocessing of cropping central 128x128 pixels from the image and resizing it to 64x64 pixels.

In test phase, we generate target networks for two images and linearly interpolate between weights of these networks. Figure 7a presents exemplary images returned by interpolated target networks. In most cases, interpolation produces natural images which come from the true data distribution (first four rows). It means that we transformed a manifold of images into a more compact structure, where linear interpolation can be applied. It allows us to suspect that similar images have similar weights in their corresponding target networks. Occasionally, interpolated weights produce images from outside of images manifold (last row). This negative effect sometimes occurred when interpolated images were very different, e.g. in this examples, we have a pair of images of people of different sex, hairstyle, skin color and photographed in different poses. These rare cases are accepted, because we did not use any additional constraints. For a comparison, we generate classical pixel-wise interpolation between analogical examples. As can be seen in Figure 7b, the results are much worse, resulting in a superimposition of images.

Going further, we verify a layer-wise interpolation. Namely, we take weights to one target network and gradually change weights of the first i layers in the direction of corresponding weights in the second target network. As can be seen in Figure 8, each layer may be understood as having different functionality, i.e. third layer is responsible for the general shape, while the last layer corrects the colors in the image.

Table 1. The average PSNR values obtained for a super-resolution task.

	Scale	**Bicubic**	**SRCNN** [12]	**RDN** [44]	**Hypernetwork**
Set5	2x	33.64	36.66	38.30	36.09
	3x	30.41	32.75	34.78	32.85
	4x	28.42	30.49	32.61	30.69
Set14	2x	30.33	32.45	34.10	32.30
	3x	27.63	29.30	30.67	29.37
	4x	26.08	27.50	28.92	27.61
B100	2x	29.48	31.36	32.40	31.11
	3x	27.12	28.41	29.33	28.31
	4x	25.87	26.90	27.80	26.86
Urban100	2x	26.85	29.50	33.09	29.43
	3x	24.43	26.24	29.00	26.26
	4x	23.11	24.52	26.82	24.56

4.2 Super-resolution

The hypernetwork allows us to describe every image as a continuous function (target network) defined on a unit 2D square. As a result, we can evaluate this function on every grid and upscale the image to any size. In this experiment, we compare the effects of this process with super-resolution approaches.

To make this approach successful we feed the hypernetwork with low resolution images and evaluate MSE loss on high resolution ones. More precisely, we take the original image of the size $m \times n$, downscale it to $k \times l$ using bicubic interpolation and input it to the hypernetwork. The hypernetwork produces the weights to target network, which defines the functional representation of the input image. To evaluate its quality, we take a grid of the size $m \times n$ on the image returned by target network and compare the values of this grid with pixels intensities of the original image.

Since input images can have different resolutions, we split them into overlapping parts of fixed sizes. In consequence, the value at each coordinate is described by multiple target networks. To produce a single output for every coordinate at test phase, we take the (weighted) average of values returned by all target networks covering this coordinate. This also allows us to smooth the output function. Moreover, we supplied a target network with an additional parameter α, which indicated the scaling factor.

To test our approach, we train the model on 800 examples from `DIV2K` data set [1]. Its performance is evaluated on `Set5` [5], `Set14` [42], `B100` [31], and `Urban100` [17]. As a quality measure we use PSNR and SSIM [40], which are common score functions applied in super-resolution. Their high values indicate better performance of a model. We consider scale factors $\times 2, \times 3, \times 4$.

As a baseline, we use bicubic interpolation. Moreover, we compare our approach with SRCNN [12], which was a state-of-the-art method in 2016. We also include a recent state-of-the-art method – RDN [44]. Our goal is to train a single hypernetwork model to generate images at various scales. This is a more general

Table 2. The average SSIM values obtained for a super-resolution task.

	Scale	**Bicubic**	**SRCNN** [12]	**RDN** [44]	**Hypernetwork**
Set5	2x	0.930	0.9542	0.9616	0.9505
	3x	0.869	0.909	0.930	0.910
	4x	0.812	0.863	0.900	0.869
Set14	2x	0.869	0.907	0.922	0.900
	3x	0.775	0.822	0.848	0.816
	4x	0.703	0.751	0.789	0.751
B100	2x	0.843	0.888	0.902	0.879
	3x	0.738	0.786	0.811	0.778
	4x	0.666	0.710	0.743	0.706
Urban100	2x	0.839	0.895	0.937	0.891
	3x	0.733	0.799	0.868	0.798
	4x	0.656	0.722	0.807	0.723

solution than typical super-resolution approaches, where every model is responsible for upscaling the image to only one resolution. Therefore, it is expected that both SRCNN and RDN will perform better than our method.

The results presented in Tables 1 and 2 demonstrate that our model gives significantly better performance than baseline bicubic interpolation (see also Figure 3 for sample result). Surprisingly, a single hypernetwork trained on all scales achieves a comparable performance to SRCNN, which created a separate model for each scale factor. It shows high potential of our model. Nevertheless, our model was not able to obtain scores as high as recent state-of-the-art super-resolution methods. It might be caused by insufficient architecture of hypernetwork. In our opinion, designing similar architecture of hypernetwork to RDN should lead to comparable performance. The main advantage of our approach is its generality – we train a single model for various scale factors.

5 Conclusion

In this work, we have presented an extension of hypernetworks mechanism, which makes it possible to create functional representations of images. Due to the continuity of the representation, we were able to upscale images to any resolution, while obtaining results comparable to those achieved by specialized super-resolution models. We also observed that the hypernetwork transforms a manifold of images to a more compact space with a convenient topology. In particular, we were able to traverse linearly from one image to another (using the weights of their target networks) without falling out from the true data distribution.

In future work, we will investigate other applications of the proposed hypernetwork-based functional image representation. One natural direction of further research is testing other image restoration tasks, such as deblurring, denosing and deblocking. We will also investigate the potential of hypernetworks in image inpainting. Furthermore, we are going to ensure better continuity of the learned

representation, by using derivative-based methods such as Sobolev Training [9], and based on the improved representation, we will work on implementing continuous filters for convolutional neural networks.

Another research direction is to further explore the geometry of the target network space. For example, we will try to use the distance in the weights space between two target network-based representations as an alternative to the typical mean squared error loss used in many deep learning models, e.g. autoencoders or generative models.

Acknowledgements

This work was partially supported by the National Science Centre (Poland) grant no. 2018/31/B/ST6/00993.

References

1. Agustsson, E., Timofte, R.: Ntire 2017 challenge on single image super-resolution: Dataset and study. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 126–135 (2017)
2. Baldi, P.: Autoencoders, unsupervised learning, and deep architectures. In: Proceedings of ICML workshop on unsupervised and transfer learning. pp. 37–49 (2012)
3. Banfield, J.D., Raftery, A.E.: Model-based gaussian and non-gaussian clustering. Biometrics pp. 803–821 (1993)
4. Bengio, Y., Goodfellow, I.J., Courville, A.: Deep learning. Nature **521**(7553), 436–444 (2015)
5. Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M.L.: Low-complexity single-image super-resolution based on nonnegative neighbor embedding (2012)
6. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: SMASH: one-shot model architecture search through hypernetworks. CoRR **abs/1708.05344** (2017), http://arxiv.org/abs/1708.05344
7. Christopoulos, C., Skodras, A., Ebrahimi, T.: The jpeg2000 still image coding system: an overview. IEEE transactions on consumer electronics **46**(4), 1103–1127 (2000)
8. Czarnecki, W.M., Osindero, S., Jaderberg, M., Swirszcz, G., Pascanu, R.: Rethinking the inception architecture for computer vision. In: Advances in Neural Information Processing Systems. pp. 4278–4287 (2017)
9. Czarnecki, W.M., Osindero, S., Jaderberg, M., Swirszcz, G., Pascanu, R.: Sobolev training for neural networks. In: Advances in Neural Information Processing Systems. pp. 4278–4287 (2017)
10. Danelljan, M., Robinson, A., Khan, F.S., Felsberg, M.: Beyond correlation filters: Learning continuous convolution operators for visual tracking. In: European Conference on Computer Vision. pp. 472–488. Springer (2016)
11. Do, M.N., Vetterli, M.: The finite ridgelet transform for image representation. IEEE Transactions on image Processing **12**(1), 16–28 (2003)
12. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE transactions on pattern analysis and machine intelligence **38**(2), 295–307 (2016)

13. Gao, S., Gruev, V.: Bilinear and bicubic interpolation methods for division of focal plane polarimeters. Optics express **19**(27), 26161–26173 (2011)
14. Geladi, P., Kowalski, B.R.: Partial least-squares regression: a tutorial. Analytica chimica acta **185**, 1–17 (1986)
15. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
16. Ha, D., Dai, A., Le, Q.V.: Hypernetworks. arXiv preprint arXiv:1609.09106 (2016)
17. Huang, J.B., Singh, A., Ahuja, N.: Single image super-resolution from transformed self-exemplars. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5197–5206 (2015)
18. Hwang, J.W., Lee, H.S.: Adaptive image interpolation based on local gradient features. IEEE signal processing letters **11**(3), 359–362 (2004)
19. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
20. Kim, J., Kwon Lee, J., Mu Lee, K.: Accurate image super-resolution using very deep convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1646–1654 (2016)
21. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
22. Krueger, D., Huang, C.W., Islam, R., Turner, R., Lacoste, A., Courville, A.: Bayesian hypernetworks. arXiv preprint arXiv:1710.04759 (2017)
23. Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4681–4690 (2017)
24. Lee, T.S.: Image representation using 2d gabor wavelets. IEEE Transactions on pattern analysis and machine intelligence **18**(10), 959–971 (1996)
25. Lim, B., Son, S., Kim, H., Nah, S., Mu Lee, K.: Enhanced deep residual networks for single image super-resolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 136–144 (2017)
26. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2117–2125 (2017)
27. Liu, M.Y., Breuel, T., Kautz, J.: Unsupervised image-to-image translation networks. In: Advances in Neural Information Processing Systems. pp. 700–708 (2017)
28. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of International Conference on Computer Vision (ICCV) (2015)
29. Lorraine, J., Duvenaud, D.: Stochastic hyperparameter optimization through hypernetworks. CoRR **abs/1802.09419** (2018), http://arxiv.org/abs/1802.09419
30. Louizos, C., Welling, M.: Multiplicative normalizing flows for variational bayesian neural networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 2218–2227. JMLR. org (2017)
31. 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: null. p. 416. IEEE (2001)
32. Scholkopf, B., Smola, A.J.: Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press (2001)
33. Sheikh, A.S., Rasul, K., Merentitis, A., Bergmann, U.: Stochastic maximum likelihood optimization via hypernetworks. arXiv preprint arXiv:1712.01141 (2017)
34. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)

35. Takeda, H., Farsiu, S., Milanfar, P., et al.: Kernel regression for image processing and reconstruction. Ph.D. thesis, Citeseer (2006)
36. Tolstikhin, I., Bousquet, O., Gelly, S., Schoelkopf, B.: Wasserstein auto-encoders. arXiv preprint arXiv:1711.01558 (2017)
37. Unser, M., Aldroubi, A., Eden, M.: Fast b-spline transforms for continuous image representation and interpolation. IEEE Transactions on Pattern Analysis & Machine Intelligence (3), 277–285 (1991)
38. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning. pp. 1096–1103. ACM (2008)
39. Wang, N., Yeung, D.Y.: Learning a deep compact image representation for visual tracking. In: Advances in neural information processing systems. pp. 809–817 (2013)
40. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P., et al.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)
41. Yeh, R.A., Chen, C., Yian Lim, T., Schwing, A.G., Hasegawa-Johnson, M., Do, M.N.: Semantic image inpainting with deep generative models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5485–5493 (2017)
42. Zeyde, R., Elad, M., Protter, M.: On single image scale-up using sparse-representations. In: International conference on curves and surfaces. pp. 711–730. Springer (2010)
43. Zhang, C., Ren, M., Urtasun, R.: Graph hypernetworks for neural architecture search. CoRR **abs/1810.05749** (2018), http://arxiv.org/abs/1810.05749
44. Zhang, Y., Tian, Y., Kong, Y., Zhong, B., Fu, Y.: Residual dense network for image super-resolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2472–2481 (2018)