Inverse Design of Magnetic Fields using Deep Learning

Pollok, Stefan; Bjørk, Rasmus; Jørgensen, Peter Stanley

Published in:
I E E E Transactions on Magnetics

Link to article, DOI:
10.1109/TMAG.2021.3082431

Publication date:
2021

Document Version
Peer reviewed version

Inverse Design of Magnetic Fields using Deep Learning

Stefan Pollok[1], Rasmus Bjørk[1], and Peter Stanley Jørgensen[1]

[1]Department of Energy Conversion and Storage, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark

We here present a novel deep learning (DL) approach for designing structures of permanent magnets. The challenge for the DL method in this kind of problem is to learn the mapping from a desired magnetic field to a simple magnetic structure, i.e. an inverse design approach. We demonstrate this approach by training six different standard convolutional neural network (CNN) structures previously used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) to inversely predict the properties of a single hard magnet (magnetization, size and location) from a given two-dimensional magnetic field. We show that the best network, ResNeXt-50, can perform this prediction with an error of 0.22% in the properties of the magnet.

***Index Terms*—Artificial neural networks, deep learning (DL), design optimization, magnetostatics, permanent magnets.**

I. Introduction

PERMANENT magnets play an important role in different areas such as magnetic resonance imaging to obtain anatomical pictures of the body, frictionless bearings in fly-wheels to store energy, and electric motors and generators. Almost all these structures are formed out of several small permanent magnets with simple geometries, which have to be designed to together provide the desired magnetic field strength and homogeneity.

Such optimized magnet structures can in the simplest cases be analytically found, such as the Halbach cylinder [1], or through numerical methods [2], [3]. For example a genetic algorithm has been used to identify the optimal placement of magnetic segments, which suffer from magnetization variations due to production limitations [4]. Most of the published methods follow the principle of iterating over the design of a structure, until this produces a field that is close to the desired field. Here, we are interested in exploring the opposite path, namely directly inferring the shape and properties of the single permanent magnets directly from the requirements of the desired magnetic field. This is also known as inverse design of materials. We do this by considering an approach using deep learning (DL) to explore such a mapping from requirement to magnet design.

Within inverse design of materials in general, which the inverse design of magnetic fields is a subsection of, several DL techniques for a desired functionality have been presented [5]. For example generative models using neural networks have been used to explore the chemical space, where it was found that the crucial step is the formulation of the molecular representation [5]. Inverse design has also been applied to e.g. integrated photonic power splitters with the use of a fully-connected neural network with residual learning [6]. Here, the spectral transmission response is used as input to populate elements in a fixed topology target space. On a silicon-on-insulator platform, which is represented as a 20×20 hole vector, a binary classification determines for each pixel if it is etched or not etched.

Corresponding author: S. Pollok (email: spol@dtu.dk).

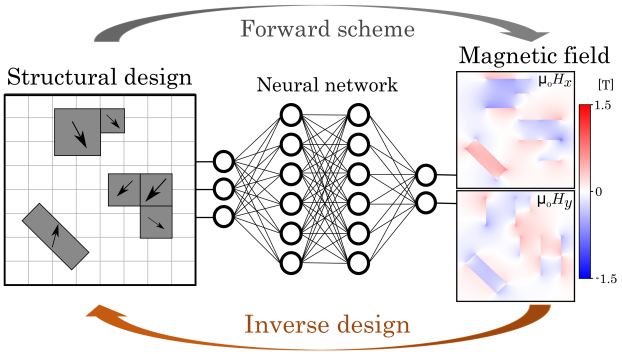

Fig. 1. Overview of two different design approaches using DL. In the forward scheme, a neural network is trained to approximate the magnetic field from a given magnetic structure. In the inverse design pipeline, which is the system studied here, a desired magnetic field is specified and mapped to an optimal setup.

Specifically regarding magnetic fields, DL techniques have been demonstrated to approximate the solution of Maxwell's equations for electromagnetic structures [7], i.e. computing the magnetic field generated by a collection of permanent magnets. The main benefit of this is having a computationally cheaper option to traditional finite element frameworks for calculating the magnetic field. By using a SegNet [8] architecture, an input image of the magnetic structure is encoded to an intermediate latent space, which is fed to a decoding block to produce the approximated magnetic field. Finally, DL can accelerate topology optimization [9].

In the last few years, the DL community experienced rapid progress in neural networks architectures of stacked convolutional layers, which, combined with increased computational power, achieved outstanding performance in feature extraction and surpassed human ability to classify images in the popular ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [10]. A major breakthrough dates back to the introduction of residual learning [11], which facilitates the training of very deep neural networks. In addition to that, representational power is increased through channel-wise scaling with Squeeze-and-Excitation blocks [12] and depth-wise convolutions integrated has been shown to be more effective

while maintaining accuracy [13]. The latest improvement in image classification is achieved via transfer learning and the application of Transformer architectures [14].

In this work we establish representations for the magnetic field and the design parameters of magnetic structures, which can be fed into modern convolutional neural network (CNN) architectures. Subsequently, we show that it is feasible to extract the properties of a single hard magnet from a magnetic field, i.e. inverse design for a single magnet. We train several state-of-the-art CNNs and compare these. In addition to their performance, we analyze the influence of different magnetic field resolutions on the prediction ability. The main focus of our work is put on exploring DL frameworks suitable for magnetics. These findings serve as a baseline for extended setups, i.e. superposition of multiple magnets of different shapes.

II. Formulation of Representations and Experimental Setup

For the inverse design of magnetic fields the unknown mapping between a given field and a valid structural design has to be modeled. DL is capable of approximating such underlying functions of demanding complexity. Our approach for designing systems of permanent magnets is shown in Fig. 1. As input to our inverse model a point-wise desired magnetic field $\mathbf{H}$ is given. The expert system shall output the design parameters, i.e. location, shape and magnetization $\mathbf{M}$ of the hard magnets, required to produce such a field. We investigate here how to formulate suitable representations for $\mathbf{H}$ and the design parameters to facilitate feature extraction with winning CNNs of ILSVRC image classification.

A. Magnetic field as input

In ILSVRC the expert systems have to extract features of given RGB images and categorize it to a given set of classes. The CNNs take as input a three-dimensional data array, which is denominated as tensor in DL. This tensor contains the three color channels of each pixel of a given image, which is cropped to have a resolution of 224×224 in most cases. To resemble such data, we evaluate the magnetic field in a two-dimensional area of interest (AOI), which is depicted as a blue square in Fig. 2. It is described parametrically by

$$AOI = \begin{pmatrix} 0 \\ 0 \\ 0.5 \end{pmatrix} + s * \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + t * \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \tag{1}$$

where s and t range from 0 to 10 cm. We divide AOI into pixels with a resolution-dependent side length, 10 cm / 224, and specify the magnetic field $\mathbf{H}$ point-wise at the center of each pixel scaled by the vacuum permeability μ_0:

$$input = \begin{bmatrix} \mu_0\mathbf{H}_{(x_1,y_1,0.5)} & \cdots & \mu_0\mathbf{H}_{(x_1,y_R,0.5)} \\ \vdots & \ddots & \vdots \\ \mu_0\mathbf{H}_{(x_R,y_1,0.5)} & \cdots & \mu_0\mathbf{H}_{(x_R,y_R,0.5)} \end{bmatrix}, \tag{2}$$

which takes a general tensor shape of [3, 224, 224]. However, for the Inception-ResNet-v2 CNN, described subsequently, a slightly higher resolution of 299×299 is used, to comply with the standard network architecture. For this we perform another simulation, where the magnetic field values are requested at pixel centers with a smaller distance in between. Additionally, we create further datasets of varying resolution to be able to investigate how performance depends on the magnetic field resolution. For simplicity, our work focuses merely on a two-dimensional $\mathbf{H}$, i.e. H_z equals 0 A/m. Hence, we end up with an input tensor with two channels.

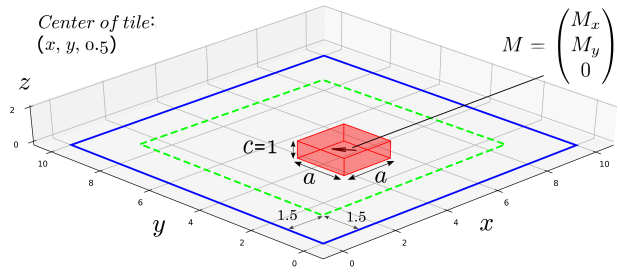

Fig. 2. An illustration of the experimental setup. A single hard magnet, which is shaped as a quadratic prism with fixed height of 1 cm, is placed in an area of interest, which is marked by a blue line and extends from 0 to 10 cm in the x- and y-direction, respectively. The center position of the magnet is set to 0.5 cm in the z-direction, whereas the other two dimensions are contained by the green dotted line. The magnetization vector is set to 0 T in the z-direction.

B. Design parameters for magnetic structures

We restrict our setup to consist of a single hard magnet shaped as a quadratic prism with a fixed height of 1 cm. The magnet is assumed to be oriented with its sides parallel to the chosen coordinate system and we set the relative permeability μ_r to 1 in all axes. The side length of the prism, a, varies from 0.5 to 2 cm. The center of the magnet is set to (x, y, 0.5), where x and y can vary from 1.5 to 8.5 cm, such that the magnet is always fully enclosed and its center is cut through by the AOI. The magnetization can be represented with its norm $|M|$ and the direction of the easy axis of the magnet. This direction is fully described by an azimuthal angle ϕ and a polar angle θ in the spherical coordinate system. The magnitude of the magnetization is varied from 0.5 to 1.5 T, while ϕ is varied between 0 and 2π and θ is fixed to $\pi/2$ as we want to set the z-component of its magnetization to 0 T. However, the angle ϕ has a discontinuity between 0 and 2π, which means that a target angle lying close to the discontinuity has a huge training loss when lying on the other side of that discontinuity. This results in the weights within the neural network being updated in such a manner that the prediction is moved away from its ground truth. Therefore, we instead represent the magnetization as its non-zero components M_x and M_y, which resulted in lower test errors compared to the angle representation. We train the CNNs with the following five-dimensional target vector:

$$target = \left(x, y, a, M_x, M_y\right)^T, \tag{3}$$

where x, y, and a are normalized to range from 0 to 1, whereas M_x and M_y lie in between -1 and 1. We calculate the magnetic components from the original data as $(|M| - 0.5) * \cos\phi$ and $(|M| - 0.5) * \sin\phi$, respectively.

C. Data sampling

The approach of learning the desired mapping with artificial neural networks is data-driven. We use the open-source software MagTense [15], which is a magnetostatic and micromagnetism calculation framework with interfaces to Matlab and Python, to create a large amount of exact training data fast and efficiently. We create two datasets with 30,000 and 60,000 input-target pairs, respectively, using a design space population derived from Latin Hypercube Sampling [16]. We divide these sets into 60% training samples, 30% validation samples, and a test dataset with 10% samples.

D. Neural networks and Parameterization

To realize a neural network capable of inverse design, we consider six modern CNNs used for image classification: ResNet-50 [11], which uses residual learning, ResNeXt-50 ($32\times4d$) [17], which applies group convolutions in the residual blocks, SEResNeXt-50 ($32 \times 4d$) [12], which adds modular Squeeze-and-Excitation blocks to the residual units, DenseNet-169 [18], which makes features of identity mappings accessible throughout several convolutional layers, EfficientNet-B1 [19], which achieved similar accuracy as ResNet-50 on several benchmarks with a 0.3 times larger network, and Inception-ResNet-v2 [20], which combines convolutions of different scales with the ResNet layout. The specific architectures are chosen to have approximately similar accuracy on ILSVRC. It has to be remarked that Inception-ResNet-v2 is approximately twice as large as the ResNet-50. We use PyTorch [21] re-implementations of these models [22].

As loss function we use the root-mean-square error (RMSE) between prediction and ground-truth target vector. It is back-propagated through the neural network using the Stochastic Gradient Descent (SGD) optimizer with a momentum of 0.9, a weight decay of 1e-4 and an initial learning rate of 0.1 that is multiplied by 1/10 every 30 epochs. These hyperparameters, which have been optimized for the ResNet architectures, are similar to the ones used in [12]. After 100 epochs we stop the training manually as convergence is observed in most runs shortly after the second learning rate decay to 0.001 at epoch 60. All training runs are performed on a NVIDIA Quadro RTX 8000. With a batch size of 60 and a resolution of 224×224 an epoch of ResNet-50 takes around 80 s, 115 s for ResNeXt-50, 120 s for SEResNeXt-50, 120 s for DenseNet-169, 80 s for EfficientNet-B1, and 270 s for Inception-ResNet-v2 with a resolution of 299×299.

III. Results

Table I shows the test error of the selected CNNs on our setup as percentage of the full target parameter span. During training of a network we store the learned weights after each epoch and use the state with the lowest validation error for subsequent evaluation with a batch size of 1. In addition to the RMSE between the predicted and the ground-truth target vector of the test samples, we calculate the error of each single target parameter. Hereby, the RMSE leads to the same results as the mean absolute test error (MAE), which is listed for the best performing network of each architecture. All tested CNNs perform well on our multi-output regression task, with all having test error less than 1% for a dataset size of 30k. However, SEResNeXt-50 and EfficientNet-B1 have an error more than twice as high as the other tested architectures. When looking at error distributions of the single test samples, we observe that the lower performance is related to around 20 bad predictions out of the 3k test samples with errors of more than 10%. Interestingly, SEResNeXt-50 has similar training errors compared to the other ResNets, which is not depicted here, but lacks the ability to generalize the model to the validation samples similarly well as shown in Fig. 3. Here, SEResNeXt-50 fluctuates around 6%, which indicates that the validation

TABLE I
Test error of selected neural network architectures [%]

Neural network architecture	RMSE 30k	RMSE 60k	MAE x	MAE y	MAE a	MAE M_x	MAE M_y
ResNet-50	0.42	0.24	0.11	0.11	0.20	0.27	0.26
ResNeXt-50 ($32 \times 4d$)	0.43	**0.22**	0.09	0.09	0.19	**0.25**	**0.24**
SEResNeXt-50 ($32 \times 4d$)	0.96	0.23	**0.08**	**0.08**	**0.18**	0.27	0.27
DenseNet-169	0.38	0.24	0.10	0.11	0.20	0.27	0.26
EfficientNet-B1	0.86	1.38	0.54	0.65	0.61	0.78	0.94
Inception-ResNet-v2	**0.33**	0.25	0.11	0.10	0.22	0.29	0.27

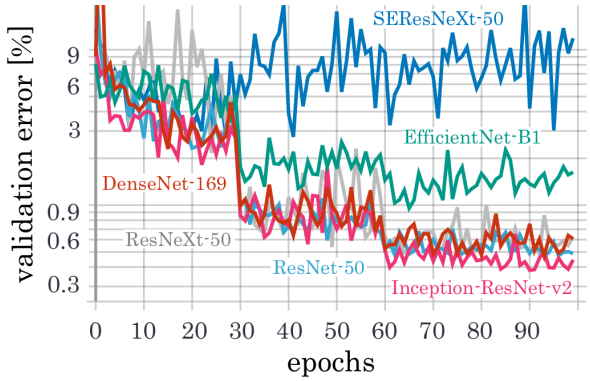

Fig. 3. Validation errors during training with 30k samples. Inception-ResNet-v2 outperforms the other architectures with a small margin, whereas SEResNeXt-50 and EfficientNet-B1 are not able to reduce the validation loss after the first learning rate update at epoch 30.

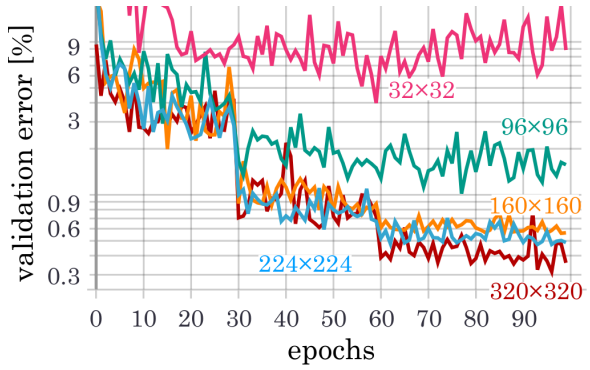

Fig. 4. Influence of different resolutions on the validation error during training of ResNet-50 with 30k samples. The resolution of the input magnetic field is increased step-wise from 32×32 to 320×320. From resolution a of 160×160 on, the trained network can generalize the problem significantly better compared to lower resolutions.

set contains even more of these huge outliers. Nevertheless, this architecture achieves one of the best performances and is able to generalize better with the larger dataset of 60k samples. On the other hand, EfficientNet-B1 performs even worse provided with more data, which is most probably due to non-optimal training, as the chosen hyperparameters are the ones optimized for ResNet architectures. Besides that, we want to point out the stochastic nature of neural network training. After each epoch the network is updated by SGD with randomly shuffled training samples. On top of that, the initialization of the network parameters influences the results. To summarize, Inception-ResNet-v2 has the lowest test error of 0.33% for 30k samples, which could be due to the higher resolution of 299×299 of the input magnetic field. Provided with 60k samples during training, ResNeXt-50 achieves the overall lowest error of 0.22%. The other architectures besides EfficientNet-B1 perform similarly well.

IV. Discussion

To investigate the influence of the input pixel size, we trained ResNet-50 with 30k samples on a broader range of resolutions. The validation losses during training are shown in Fig. 4. Trained with a magnetic field resolution of 320×320 the network has a RMSE of 0.27%, which is lower than the Inception-ResNet-v2 RMSE of 0.33% for a resolution of 299×299. As can also been seen from the figure, reducing the resolution below 160×160 leads to a quickly growing RMSE, i.e. the ability to sense fine differences in the target vector diminishes rapidly.

Regarding the inverse calculated properties in general, it is seen that the location parameters x and y are learned best by all the neural networks. The magnetization components M_x and M_y have a MAE around two to three times higher compared to the center location in all architectures, which can be partially explained by the target range being twice as large. We note that unequal MAEs in the location and magnetization parameters indicate huge outliers in the test error and hence show a problem to generalize the learned mapping to unseen magnetic fields. The capability of the models to predict side length a is worse compared to the location parameters. A reason for that could be its smaller physical range from 0.5 to 2 cm. When relating the errors to the physical parameters, the MAE of our overall best performing network, i.e. ResNeXt-50 trained with 60k samples and a resolution of 224×224, corresponds to an offset of 63.0 μm for the location parameters and 28.5 μm for the side length. For the magnetization we calculate the errors back to the norm $|M|$ and the azimuthal angle ϕ of the easy axis and obtain a MAE corresponding to 3.1 mT and 0.43°, respectively. The highest RMSE of a single test sample evaluated by that network is 1.67%, and even for this error the predicted single permanent magnet is visually hardly distinguishable from its ground-truth. It should be mentioned here that the magnetization of a permanent magnet varies from its specification in a range of several percentages due to manufacturing variations. Therefore, further improvements of the network performance on this inverse design problem by e.g. adjusting the neural network architecture or fine-tuning the hyperparameters of the optimizer are mostly redundant.

V. Conclusion

In this work we showed that it is feasible to infer the properties of a single permanent magnet from a two-dimensional magnetic field with modern CNNs. Representing the magnetic field component as a two-channel tensor of varying resolution, we showed that the ResNeXt-50 trained with 60k samples can predict the properties of a magnet with an average error of 0.22%, which is significantly below manufacturing tolerances. Other tested standard CNNs performed only slightly worse. Increasing the complexity of the problem to multiple magnets of different shapes and an input magnetic field in three dimensions is subject to further research.

References

[1] K. Halbach, "Design of permanent multipole magnets with oriented rare earth cobalt material", *Nucl. Instrum. Methods,* vol. 169, no. 1, pp. 1-10, 1980.

[2] R. Bjørk, C.R.H. Bahl, A. Smith, and N. Pryds, "Improving magnet designs with high and low field regions", *IEEE Trans. Magn.,* vol. 47, no. 6, pp. 1687-1692, 2011.

[3] A.R. Insinga, A. Smith, C.R.H. Bahl, K.K. Nielsen, and R. Bjørk, "Optimal Segmentation of Three-Dimensional Permanent-Magnet Assemblies", *Phys. Rev. Appl.,* vol. 12, no. 6, 2019, Art. no. 064034.

[4] K.K. Nielsen, A.R. Insinga, C.R.H. Bahl, and R. Bjørk, "Optimizing a Halbach cylinder for field homogeneity by remanence variation", *J. Magn. Magn. Mater.,* vol. 514, 2020, Art. no. 167175.

[5] B. Sanchez-Lengeling and A. Aspuru-Guzik, "Inverse molecular design using machine learning: Generative models for matter engineerings", *Science,* vol. 361, no. 6400, pp. 360-365, 2018.

[6] M.H. Tahersima et al., "Deep Neural Network Inverse Design of Integrated Photonic Power Splitters", *Sci. Rep.,* vol. 9, 2019, Art. no. 1368.

[7] A. Khan, V. Ghorbanian , and D. Lowther, "Deep Learning for Magnetic Field Estimation", *IEEE Trans. Magn.,* vol. 55, no. 6, 2019, Art. no. 7202304.

[8] V. Badrinarayanan, A. Handa, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling", *arXiv:1505.07293,* 2015.

[9] H. Sasaki and H. Igarashi, "Topology Optimization Accelerated by Deep Learning", *IEEE Trans. Magn.,* vol. 55, no. 6, 2019, Art. no. 7401305.

[10] O. Russakovsky et al., "ImageNet Large Scale Visual Recognition Challenge", *Int. J. Comput. Vis.,* vol. 115, pp. 211-252, 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.,* 2016, pp. 770-778.

[12] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks", *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.,* 2018, pp. 7132-7141.

[13] M. Tan et al., "MnasNet: Platform-Aware Neural Architecture Search for Mobile", *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.,* 2019, pp. 2820-2828.

[14] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", *arXiv:2010.11929,* 2020.

[15] R. Bjørk and K.K. Nielsen, MagTense, www.magtense.org, 2019, doi: 10.11581/DTU:00000071.

[16] M.D. McKay, R.J. Beckman, and W.J. Conover, "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code", *Technometrics,* vol. 42, no. 1, pp. 55-61, 2000.

[17] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks", *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.,* 2017, pp. 1492-1500.

[18] G. Huang, Z. Liu, L. van der Maaten, and K.Q. Weinberger, "Densely Connected Convolutional Networks", *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.,* 2017, pp. 4700-4708.

[19] M. Tan and Q.V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks", *Proc. 36th Int. Conf. Mach. Learn.,* 2019, pp. 6105-6114.

[20] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning", *Proc. 31st AAAI Conf. Artif. Intell.,* 2017, pp. 4278-4284.

[21] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library", *Adv. Neural Inf. Process. Syst.,* vol. 32, 2019.

[22] O. Sémery, "Sandbox for training convolutional networks for computer vision", https://github.com/osmr/imgclsmob. Accessed: Nov. 30, 2020.