Magnetic CNNs: Convolutional Neural Nets for magnetism problems

author: Francesco Casola[1,2]

1-Department of Physics, Harvard University, 17 Oxford St., Cambridge, MA 02138, USA.
2-Harvard-Smithsonian Center for Astrophysics, 60 Garden St., Cambridge, MA 02138, USA.

CONTENTS

I. GOAL OF THE PROJECT

Computing the magnetic field produced by a certain magnetization pattern is a task performed by many available numerical routines, with applications ranging from science to engineering. What about the opposite calculation instead? Is it possible, in general, to reconstruct a certain magnetization pattern given the field $\mathbf{B}$ it produces?

We have recently been the first to answer this question and studied the related math in one of our works here at Harvard[1]. We realized that obtaining the manifold of solutions compatible with the measured field is, in the general case, an extremely non-trivial task involving complex numerical variational minimizations (see section "formalism" and the supplement of Ref. 1 for details). By *solution*, in our case, we mean a magnetization (i.e. a 2D vector field) which produces a stray field compatible with the measurements.

At the moment, obtaining these solutions presents different problems, for instance:

1. The solution is usually obtained via numerical variational steepest descent minimization.
   The minimization therefore requires an initial "guessed" solution, of which we may have *a priori* no idea about.
2. The manifold of compatible solutions can be sorted by a number called *helicity*[1]. Fixing the helicity corresponds, ultimately, to fixing the value of the local divergence of the 2D vector field. Only once the helicity is fixed, the solution obtained variationally can be considered unique[1]. The physical meaning of imposing such constraint is visually shown in Fig. 1f,g of Ref. 1. At the moment we can directly obtain solutions imposing only two helicities, not arbitrary values.
3. The shape of the magnetic region producing the stray field can vary in space. If that's the case (and it usually is), uniqueness is achieved only when the local normalization of the 2D vector field is known. The normalization is usually obtained by direct inversion in Fourier space of the magnetic field, in a regime in which the magnetization is collinear. Unfortunately, the orientation of the magnetization is usually not known. Furthermore, experimental noise hampers the inversion procedure.

Here we test the possibility of solving the problems above using convolutional neural networks (CNNs). In the first version presented here, we address in particular problem 1 and 3 above. We will finally present an idea to tackle 1+2+3 (see section V). The CNN algorithm is inspired by recent publications addressing the possibility of convolutional nets to learn other, extremely important for applications, laws of physics, e.g. Eulerian fluid simulations:

J. Tompson, K. Schlachter, P. Sprechmann and K. Perlin, "Accelerating Eulerian Fluid Simulation With Convolutional Networks", arXiv:1607.03597v6 (2017).

We took the scheme for the CNN from their paper, as the equations to be solved are similar in nature.
The present version of the code (not optimal and not making use of GPUs) is available at:

`https://github.com/fcasola/MagneticCNN`

II. INTRODUCTION TO THE FORMALISM

The components of $\mathbf{B}$ are linearly dependent in Fourier space[1]. We select the out-of-plane component B_z. The latter is related with the in-plane component of the magnetization $\mathbf{m}_{x,y}$ and the out-of-plane one m_z as:

$$B_z(\boldsymbol{r}, d) = -\frac{\mu_0 M_s}{2}\left(\alpha_z(d,t) * \nabla^2 m_z(\boldsymbol{r}) + \alpha_{x,y}(d,t) * \nabla \cdot \mathbf{m}_{x,y}(\boldsymbol{r})\right), \tag{1}$$

d is the distance from the magnetization to the magnetic field plane. t is the thickness of the magnetic layer. We define $\boldsymbol{r} = (x, y)$. μ_0, M_s are constants. In this notation, $0 \leq ||\boldsymbol{m}|| \leq 1$. The two resolution functions $\alpha_{x,y}$ and α_z for the in-plane and out-of-plane component of the magnetization are not equal. In the limit $t \ll d$ it can be shown[1] that:

$$\alpha_z(d, t \ll d) \approx \frac{1}{2\pi}\frac{t}{(d^2 + r^2)^{1/2}}. \tag{2}$$

$$\alpha_{x,y}(d, t \ll d) \approx \frac{1}{2\pi}\frac{dt}{(d^2 + r^2)^{3/2}}. \tag{3}$$

III. STRATEGY FOR TRAINING

In order to simplify the training process, invariance to scaling has been used to feed the net. In particular, upon rescaling spatial coordinates as $\tilde{\boldsymbol{r}} = (x/d, y/d)$, one can rewrite (1) as:

$$B_z\left(-\frac{4\pi d^3}{\mu_0 M_s t}\right) = \left[\tilde{\alpha}_z(\tilde{\boldsymbol{r}}) * \tilde{\nabla}^2 m_z(\tilde{\boldsymbol{r}}) + \tilde{\alpha}_{x,y}(\tilde{\boldsymbol{r}}) * \tilde{\nabla} \cdot \mathbf{m}_{x,y}(\tilde{\boldsymbol{r}})\right], \tag{4}$$

which is manifestly scale-invariant. In summary, besides a mere rescaling by a constant prefactor of the net input, we also rescaled spatial coordinates in the images such that the (x, y) units are adimensional and expressed in multiples of the distance d. The CNN labels are represented by the magnetization $m(\tilde{\boldsymbol{\rho}})$, with the definition $\boldsymbol{m} = m(\tilde{\boldsymbol{\rho}})\boldsymbol{f}(\theta, \phi)$. The vector $\boldsymbol{f}(\theta, \phi)$ defines the magnetization orientation in space; in this first version of the code, both θ and ϕ have no spatial dependence. The extension to case in which θ and ϕ are allowed to vary in space is discussed in section V.

In order to generate the training dataset, we start producing a set of randomly generated elliptical-like shapes for $m(\tilde{\boldsymbol{\rho}})$ (see top of Fig. 1). Orientations for $\boldsymbol{f}(\theta, \phi)$ were uniformly sampled over the unit sphere. It should be noted that even in this *collinear* case, there exists more than one magnetization pattern compatible with a given stray field. To see this, it is enough to express the right-member of (4) in Fourier space. We have:

$$\begin{aligned}\mathscr{F}\left(\tilde{\alpha}_z(\tilde{\boldsymbol{r}}) * \tilde{\nabla}^2 m_z(\tilde{\boldsymbol{r}}) + \tilde{\alpha}_{x,y}(\tilde{\boldsymbol{r}}) * \tilde{\nabla} \cdot \mathbf{m}_{x,y}(\tilde{\boldsymbol{r}})\right) &= -\tilde{\alpha}_z(\tilde{\boldsymbol{k}})\tilde{\boldsymbol{k}}^2 m_z(\tilde{\boldsymbol{k}}) + i\tilde{\alpha}_{x,y}(\tilde{\boldsymbol{k}})\tilde{\boldsymbol{k}} \cdot \mathbf{m}_{x,y}(\tilde{\boldsymbol{k}}) \\ &= m(\tilde{\boldsymbol{k}})\left(-\tilde{\alpha}_z(\tilde{\boldsymbol{k}})\tilde{\boldsymbol{k}}^2, i\tilde{\alpha}_{x,y}(\tilde{\boldsymbol{k}})\tilde{\boldsymbol{k}}\right) \cdot \boldsymbol{f}(\theta, \phi).\end{aligned} \tag{5}$$

Eq. (5) implies that given the Fourier transform of the magnetic field and a certain direction θ, ϕ, one can in principle always find a solution for $m(\tilde{\boldsymbol{k}})$. The idea is that by training the CNN with many $m(\tilde{\boldsymbol{\rho}})$ satisfying $0 \leq ||m(\tilde{\boldsymbol{\rho}})|| \leq 1$, the CNN will predict the best approximant to this particular type of solution.

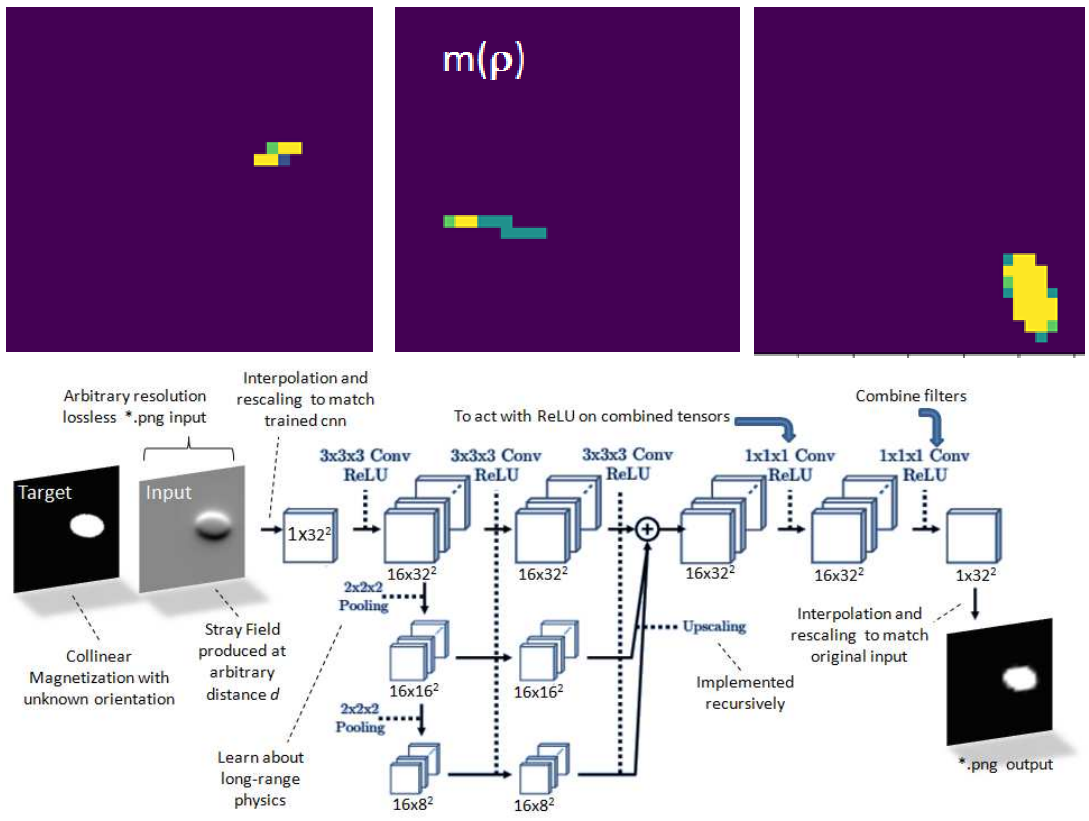
m(ρ)
Arbitrary resolution lossless *.png input
Interpolation and rescaling to match trained cnn
To act with ReLU on combined tensors
Combine filters
3x3x3 Conv ReLU
3x3x3 Conv ReLU
3x3x3 Conv ReLU
1x1x1 Conv ReLU
1x1x1 Conv ReLU
Target
Input
1x32²
16x32²
16x32²
16x32²
16x32²
1x32²
2x2x2 Pooling
16x16²
16x16²
2x2x2 Pooling
16x8²
16x8²
Upscaling
Implemented recursively
Interpolation and rescaling to match original input
Collinear Magnetization with unknown orientation
Stray Field produced at arbitrary distance d
Learn about long-range physics
*.png output

FIG. 1. **Model used in training the CNNs. Top:** Exemplary elliptical-like shapes for the magnetization, randomly generated on a 32x32 pixel grid. The colors represent the magnetization amplitude $m(\boldsymbol{\rho})$. **Bottom:** Model for the CNN, adapted from Ref. 2. The figure also pictorially shows an exemplary target of the CNN, producing the stray field shown in "input". The CNN output after training is also shown (see sec. IV for more details).

IV. RESULTS

In the present version of the code, the network has been trained after generating a set of 5000 randomly oriented and shaped ellipses, $N_s = 3000$ of which were used for training and 2000 for validation. The magnetization is collinear and has an elliptical-like shape (see exemplary image above, showing 3 shapes defined on a N_pxN_p = 32x32 pixels grid). Next, for each magnetization we computed the out-of-plane component B_z of the magnetic field using the `Compute_Bz` function in the `Create_TrainingCNN.py` module. Data are then saved in a hierarchical data format (.h5) file as $[\theta, \phi, m(\tilde{\boldsymbol{\rho}}), B_z]$ with dimensions $[N_s \text{ x } (2 + 2N_p^2)]$. The $(m(\tilde{\boldsymbol{\rho}}), B_z)$ pair is then used as the (y, x) label/feature of the CNN. The CNN dimensions and design are those described in Fig. 1. The cost function for the CNN was defined using the `reduce_mean()` Tensorflow function as the square of the deviation of the target from the CNN output. The CNN parameters are contained in a configuration file called `config_training.cfg`. Empirically, it was observed that online training performs better than batch one. An example of the network output after training is shown in Fig. 3. The result is very exciting! The loss function evaluated on the validation set returned an error of $9.55 * 10^{-4}$, similar to the training set.

It should be pointed out that the training was carried out on a single GPU Tesla K20Xm by NVIDIA, available on the https://www.rc.fas.harvard.edu/odyssey/ and took approximately 45 minutes with the current settings.

V. OUTLOOK AND OPEN QUESTIONS

We envisage the following next step for the project:

- **Extension of the code to include the helicity:** the main goal would be to obtain CNN predictions given as inputs the stray field B_z, the magnetic structure helicity and the shape (i.e. the boundaries) of the magnetic

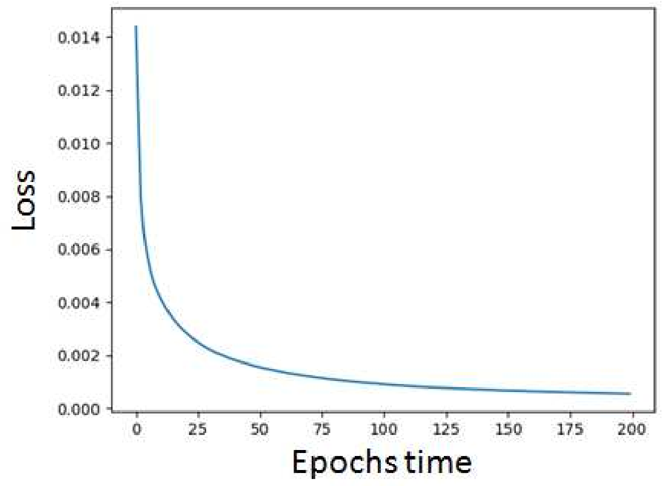

FIG. 2. **Loss as a function of the epochs.** Typical loss function obtained using a small batch size (2 elements) on a 32x32 grid.

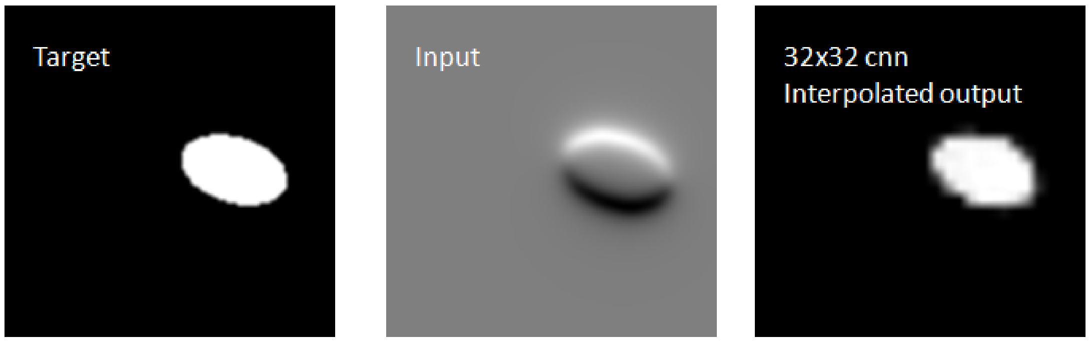

FIG. 3. **Exemplary CNN output.** Target default magnetization $m(\tilde{\rho})$ defined in the configuration file `test_shape.cfg` (left). Stray field input to the CNN (center) computed with the module `Create_TrainingCNN.py`. The interpolated output of the CNN using the current training is shown to the right.

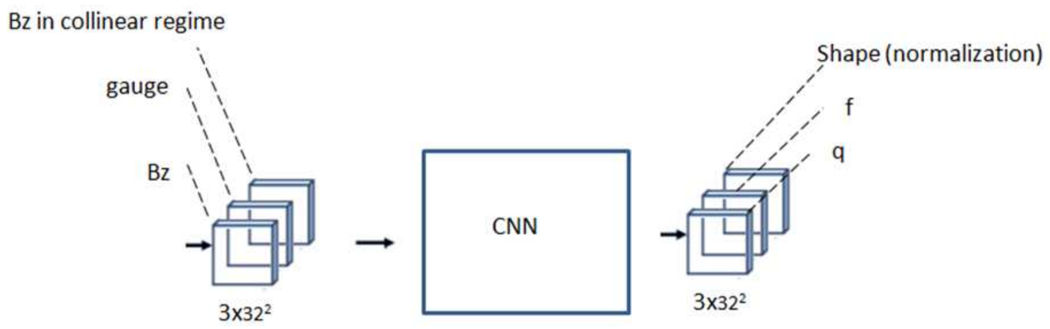

FIG. 4. **Outlook for a future implementation.** q, f stand for the spatially dependent θ, ϕ variables.

region. This will allow us to get pedictions for the magnetization, e.g., in situations where the magnetization is not only collinear but actually **varies within** a defined magnetic region such as the one in Fig. 3. For this, a possible implementation is shown in Fig. 4. We plan on feeding the CNN with the B_z value obtained from a magnetic state with a given helicity γ, and at the same time the field B_z obtained in the collinear regime, such as to define the shape. As we said, the helicity is related with the value of the local divergence $\nabla \cdot \mathbf{m}_{x,y}$ of the 2D vector field. In particular, when thinking in Fourier space $\nabla \cdot \mathbf{m}_{x,y} \rightarrow \mathbf{k} \cdot \mathbf{m}_{x,y}$, γ is the angle between $\mathbf{k}$ and $\mathbf{m}_{x,y}(\mathbf{k})$ (see Ref. 1).

A way to feed the network with γ is shown in Fig. 4, where "gauge" indicates an N_pxN_p matrix of replicated γ. When γ has a certain value, the network will be trained exclusively with stray fields B_z created by magnetic structures satisfying that given helicity.

The following are still open questions:

[1] Y. Dovzhenko, F. Casola, S. Schlotter, T. Zhou, F. Büttner, R. L. Walsworth, G. S. D. Beach, and A. Yacoby, arXiv:1611.00673 (2017).

[2] J. Tompson, K. Schlachter, P. Sprechmann and K. Perlin, "Accelerating Eulerian Fluid Simulation With Convolutional Networks", arXiv:1607.03597v6 (2017).