Using Convolutional Neural Networks for the Helicity Classification of Magnetic Fields

Nicolò Oreste Pinciroli Vago,[a,b,*] **Ibrahim A. Hameed**[a] **and M. Kachelrieß**[c]

[a]*Institutt for IKT og realfag, NTNU, Ålesund, Norway*

[b]*Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy*

[c]*Institutt for fysikk, NTNU, Trondheim, Norway*

The presence of non-zero helicity in intergalactic magnetic fields is a smoking gun for their primordial origin since they have to be generated by processes that break CP invariance. As an experimental signature for the presence of helical magnetic fields, an estimator Q based on the triple scalar product of the wave-vectors of photons generated in electromagnetic cascades from, e.g., TeV blazars, has been suggested previously. We propose to apply deep learning to helicity classification employing Convolutional Neural Networks and show that this method outperforms the Q estimator.

---

*Presenter

1. Introduction

Magnetic fields are known to play a prominent role in the dynamics and the energy budget of astrophysical systems on galactic and smaller scales, but their role on larger scales is still elusive [1]. In galaxies and galaxy clusters, the observed magnetic fields are assumed to result from the amplification of much weaker seed fields. Such seeds could be created in the early universe, e.g. during phase transitions or inflation, and then amplified by plasma processes. If the generation mechanism of such primordial fields (e.g. by sphaleron processes) breaks CP, then the field will have a non-zero helicity. Since helical fields decay slower than non-helical ones, a small non-zero initial helicity is increasing with time, making the intergalactic magnetic field (IGMF) either completely left- or right-helical today. A clean signature for a primordial origin of the IGMF is therefore its non-zero helicity. In a series of works, Vachaspati and collaborators worked out possible observational consequences of a helical IGMF, introducing the "Q statistics" as a statistical estimator for the presence of helicity in the IGMF [2].

In this work, we re-analyze the detection prospects of non-zero helicity in the IGMF using individual sources as, e.g., TeV blazars. We propose the use of Deep Learning, a subfield of Machine Learning based on deep networks (i.e., networks with more than one layer), which can perform complex transformations on the original raw data in order to classify helicity data. A family of deep learning networks, in particular, is the one of Convolutional Neural Networks (CNN). They can be used for the analysis of tensor-encoded data as, e.g., images. Such networks have proven to be useful in many fields which are related to visual properties of images, ranging from artworks classification [3] to writer identification [4], medicine [5] and particle physics [6]. Here, we evaluate the efficiency of different CNN models applied to synthetical images of TeV blazars and compare the results to a previous work using the Q statistics [7]. We describe the data transformation necessary to use CNNs, present a quantitative comparison of the adopted CNN models, and describe MobileNet v2 [8], the most promising model among those we investigated.

2. Methodology

This section presents the classification workflow, from the generation of mock data sets using `ELMAG` [9, 10] to the application of a CNN to detect prominent visual patterns in the data. Our approach consists of six main stages:

1. Generation of datasets using `ELMAG`
2. Data pre-processing
3. Image generation
4. Creation of the training, validation and test set
5. Training and validation using a Convolutional Neural Network
6. Testing of the obtained model on new data

It is possible to consider the proposed classifier as a black box that, given an input image, outputs the corresponding helicity, as shown in Fig. 1. This workflow can be applied for any possible initial simulation setting. In particular, it is possible to modify the data pre-processing parameters to give more importance to some parts of the plot (e.g., the central part), and to compare the performances and the metrics of different CNNs.
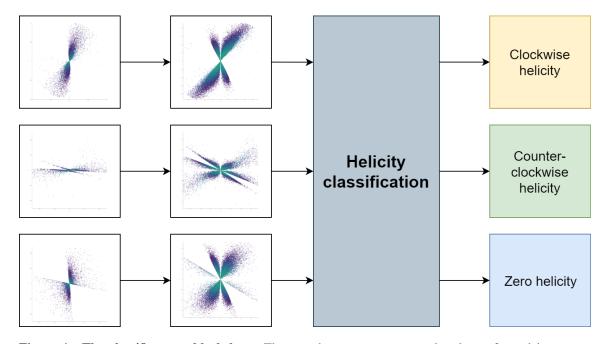
Figure 1: The classifier as a black box. The raw data are pre-processed and transformed in a more convenient format. The data processed in this way serve as input to the helicity classifier which in turn outputs the predicted helicity.

2.1 Convolutional Neural Networks

A CNN is a neural network that given an input datum represented as a tensor and a set of classes can be exploited to predict the class (or, more in general, the classes) to which the datum belongs. In our case, the input datum is an image of photons whose energy is colour encoded and the output is the corresponding helicity label, $h = \{0, \pm 1\}$, as presented in Fig. 1. More precisely, CNNs are a class of artificial neural networks (ANN), which, differently from traditional ANNs, can perform convolutions in one or more dimensions using convolutional layers. In particular, convolutions are defined by introducing one or more filters, which are tensors of numbers. A filter is originally placed in correspondence of the top-left corner of the tensor representing an input datum (e.g. an image), and an element-wise multiplication between the filter and the underlying datum elements is performed. The multiplied elements are then summed and placed in an output tensor, in the same position as the filter top-left corner position. The filter, then, is moved by a quantity called *stride* along all the tensor elements, until the entire tensor has been covered. It is also possible to introduce *padding*, which consists of adding zeros to the tensor borders to obtain an output tensor with the same dimensions as the input tensor. Figure 2 presents the result of a convolution operation on a bidimensional tensor with a single filter. The highlighted elements represent the first operation performed by the convolution, which in this example is given by $\left(\begin{smallmatrix} 1 & 2 \\ 0 & 4 \end{smallmatrix}\right) * \left(\begin{smallmatrix} 1 & 2 \\ 0 & -1 \end{smallmatrix}\right) = 1 \cdot 1 + 2 \cdot 2 + 0 \cdot 0 + 4 \cdot (-1) = 1$.

Additional to the convolution, many CNNs include pooling operations which output, for each position of the filter, the maximum element below the filter (*max pooling*) or the average of the elements below the filter (*average pooling*). In the final part of a CNN, it is necessary to insert fully connected layers, which consider all the inputs from the previous layer, perform a linear
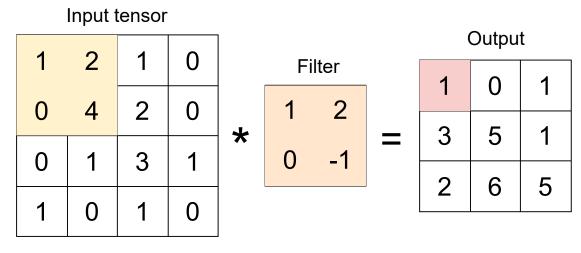
Input tensor

1	2	1	0
0	4	2	0
0	1	3	1
1	0	1	0

*

Filter

1	2
0	-1

=

Output

1	0	1
3	5	1
2	6	5

Figure 2: A convolution with a single filter.

combination of such inputs, and output a vector whose length corresponds to the number of classes of the problem (i.e., the helicity cases). The content of this vector consists of the probabilities associated with the different classes. A CNN, then, is formed by a sequence of convolutional layers, pooling layers, more complex layers based on them, and fully connected layers. This means that it encodes a complex transformation of the initial data into the labels associated with them. When images are considered, a convolution is defined over three dimensions (i.e., the width, the height and the depth). The depth initially allows encoding the colour channels in the RGB format, arranging them in subsequent layers. During the learning process, each layer of the transformed tensor represents a filter. Applying a filter over a tensor (e.g., the input image or its transformations) further transforms it. Consequently, the input tensor has the shape $n \times m \times 3$ for a coloured image with $n \times m$ pixels. After the first convolution, the output tensor will have the shape $n' \times m' \times f$ where f is the number of filters, and n' and m' depend on the filter size and the presence of padding.
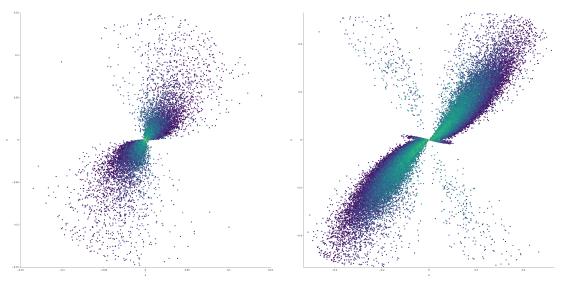
2.2 Dataset generation

The first stage of the workflow consists of the generation of an initially labelled dataset using `ELMAG`. In particular, in this first stage, the dataset contains the sky positions and energies of the simulated photons in a tabular form and the corresponding label (i.e., for each simulation, the helicity value of the turbulent field, as set in `ELMAG`). For each of the three helicity cases, 1160 photons reaching the observer were simulated. It is in principle possible to generate a larger dataset, but as the dataset size increases, the time required for training and the storage requirements increase.

2.3 Data pre-processing and image generation

Data pre-processing is essential to transform the initial raw tabular data in a more convenient format, which can be used for the generation of plots formatted as images and used as the input of CNN models. Energy is not encoded automatically in a bi-dimensional plot. Since energy is not automatically encoded in a bidimensional plot, we use the colour channel to represent different energy levels, associating colour to the logarithm of the photon energy. This continuous colour encoding of energy has to be compared to the use of energy bins in the calculation of the Q estimator.

We also encode the positions differently than in the original data, to visually emphasize the helicity direction. In the first step, we remove the 1% of photons with the largest distance to the

(a) The original plot.   **(b)** The plot after the transformation.

Figure 3: Image transformation. Left: Original image for a counter-clockwise helix. Right: After the cubic root operation. Energy is encoded as colour, varying from yellow (highest energies) to blue (lowest).

origin. Then we introduce dimensionless x, y coordinates and, if necessary, rescale them such that the coordinates of all remaining photons are contained in the range $[-1, +1]$. Next, we want to transform the image to emphasize its overall helical curvature. A set of functions leaving the sign of the x, y coordinates unchanged is defined, for a single coordinate, by $F = \{x^n \mid n = \frac{1}{2k+1} \wedge k \in \mathbb{N} \wedge x \in \mathbb{R}\}$. In particular, the cubic root is part of this set, and we chose it to transform the initial data since it is the lowest order transformation in the set. Figure 3 shows an example of this transformation. While the original image looks almost symmetrical with respect to the bisector of the first and third quadrant, the image after the transformation is less symmetrical and emphasizes the counter-clockwise helicity. Finally, the image dimensions used as the network input are set to 500×500 pixels.

2.4 Data splitting and training

Once the data are generated, they are sorted into a training (70% of the total data), validation (15%) and test set (15%). The training data are used for the training of the CNN (i.e., during the network learning phase). The validation data are used to validate the results from the training to check whether there is overfitting (i.e., whether the network memorizes the data, but is not able to generalize on new data). The test set is used, after the training and the validation, on data previously unknown to the network. In this work, we generate also the test data using `ELMAG`. However, the same procedure can be applied to real data, if they are pre-processed in the same way as the simulated data.

2.5 Architecture

The proposed classifier exploits the MobileNet v2 architecture [8], pre-trained on the ImageNet [11] data set. We decided to use a pre-trained model, rather than performing the entire

training from scratch, to exploit the initial layers' content, which is expected to contain filters for the recognition of simple patterns and geometrical shapes. Moreover, this choice allowed us to use fewer images in the training process, which corresponds to a shorter overall computational time spent on the initial simulation. Since the images' complexity and variability are rather low across our data set, we did not perform data augmentation.

MobileNet v2's main building block, differently from traditional CNNs, is a residual block that contains three convolutional layers: the expansion layer, the depth-wise convolutional layer and the projection layer. The expansion layer has the purpose of expanding the number of channels in the data. Given a $n \times m \times f$ tensor, its purpose is to generate a $n \times m \times cf$ tensor, where $c \in \mathbb{N}$ is defined as the expansion factor. To perform this operation, the expansion layer applies a 1×1 convolution considering cf filters. The depth-wise convolutional layer, differently from the traditional convolutional layer, considers each feature map separately, i.e. the input tensor feature maps are not combined. The final layer of this block is the projection layer, which reduces the number of feature maps by projecting high-dimensional data into a lower-dimensional tensor. Finally, the CNN contains residual blocks, which aim at propagating promising results in complex networks. The residual blocks introduce, in addition to the sequential path starting from the input tensor and ending in the prediction result, connections that can skip one or more blocks. The outputs of such connections are summed to the outputs obtained in the main path.

3. Results and Discussion

We choose physical parameters as in Ref. [7], to compare to their results. In particular, we use as strength of the turbulent magnetic field $B = 10^{-16}$ G, set $10 \, \text{GeV}$ as low energy cut-off, and neglect the finite angular resolution of realistic experiments. We consider how well CNNs can distinguish positive and negative helicity. Table 1 shows a comparison among the accuracy of several CNN models, which achieve an accuracy between 87% and 96%. MobileNet v2 accomplishes the highest accuracy among all networks, followed by ShuffleNet. The 96% accuracy of MobileNet v2 corresponds to an overlap of 8% in the language of Ref. [7], surpassing boldly the 53% overlap obtained with the Q estimator. Figure 4a represents the confusion matrix for MobileNet v2, while Fig. 5 presents the validation loss during the training. The greatest confusion is observed for the zero helicity case since it is an intermediate case and is more easily confused with the clockwise (or positive) or counter-clockwise (or negative) helicity cases. Moreover, one can see that the loss decreases both for the training and for the validation set, which suggests that the network has good generalization abilities.

4. Conclusions

We have analyzed the application of Convolutional Neural Networks to the problem of predicting the helicity of the IGMF using synthetical images of TeV blazars. Among the networks examined, MobileNet v2 achieved the highest accuracy and the results obtained outperform those derived using the Q estimator. While the obtained increase in accuracy is encouraging, additional research is needed to address more realistic cases. In particular, the angular resolution and energy-dependent sensitivity of specific experiments will deteriorate the results presented here for the case

Table 1: Comparison of models for the classification of clockwise and counter-clockwise helicity. This table shows, for each model and each hyperparameter setting, the accuracy on the train set, the validation set and the test set. Here η refers to the learning rate, while the batch size refers to the number of images considered simultaneously during the training phase. All the experiments used Adagrad as the optimizer and were based on networks pre-trained on ImageNet.

Model	η	η decay	Batch size	# epochs	Train accuracy	Validation accuracy	Test accuracy
ResNet50	0,010	0,0008	2	15	93,80%	94,74%	91,00%
ResNet18	0,010	0,0008	2	12	95,43%	95,32%	93,00%
GoogleNet	0,010	0,0008	2	8	93,73%	92,11%	90,00%
MobileNet v2	0,010	0,0008	2	10	97,62%	94,44%	95,00%
	0,001	0,0008	2	16	95,43%	93,86%	90,00%
	0,010	0,0008	4	16	99,81%	97,37%	96,00%
ShuffleNet	0,010	0,0008	2	8	99,12%	94,15%	95,00%
	0,005	0,0008	2	6	97,50%	95,32%	95,00%
	0,005	0,0008	8	6	99,38%	95,03%	95,00%
	0,005	0,0008	4	11	99,69%	95,91%	92,00%
DenseNet121	0,010	0,0008	2	10	90,29%	91,52%	87,00%
HardNet68	0,010	0,0008	2	17	90,16%	89,47%	87,00%
GhostNet	0,010	0,0008	2	14	99,69%	94,44%	92,00%

of an "ideal" experimental set-up. Another question to address is if the data pre-processing can be improved further, by changes in the colour-coding of energy or transformations alternative to the cubic root.

References

[1] R. Durrer, A. Neronov, Astron. Astrophys. Rev. 21 (2013) 62. doi:`10.1007/s00159-013-0062-7`. `arXiv:1303.7121`.

[2] H. Tashiro, T. Vachaspati, Mon. Not. Roy. Astron. Soc. 448 (2015) 299–306. doi:`10.1093/mnras/stu2736`. `arXiv:1409.3627`.

[3] F. Milani, P. Fraternali, A data set and a convolutional model for iconography classification in paintings, 2020. `arXiv:2010.11697`.

[4] A. Rehman, S. Naz, M. I. Razzak, I. A. Hameed, IEEE Access 7 (2019) 17149–17157. doi:`10.1109/ACCESS.2018.2890810`.

[5] S. Deepak, P. M. Ameer, Comput. Biol. Medicine 111 (2019). URL: `https://doi.org/10.1016/j.compbiomed.2019.103345`. doi:`10.1016/j.compbiomed.2019.103345`.

[6] A. Aurisano, A. Radovic, D. Rocco, A. Himmel, M. Messier, E. Niner, G. Pawloski, F. Psihas, A. Sousa, P. Vahle, Journal of Instrumentation 11 (2016) P09001.

[7] M. Kachelrieß, B. C. Martinez, Phys. Rev. D 102 (2020) 083001. URL: `https://link.aps.org/doi/10.1103/PhysRevD.102.083001`. doi:`10.1103/PhysRevD.102.083001`.
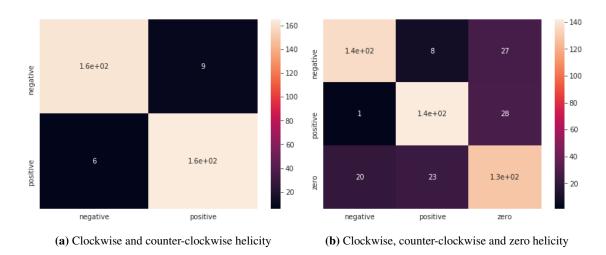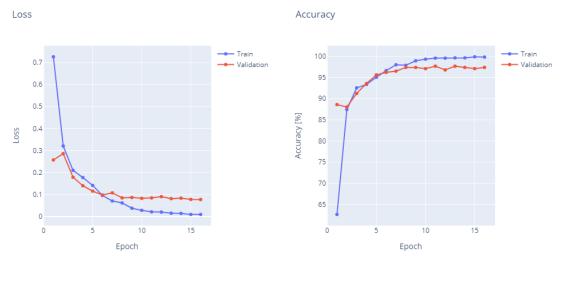
(a) Clockwise and counter-clockwise helicity

(b) Clockwise, counter-clockwise and zero helicity

Figure 4: **Confusion matrices for clockwise (positive), counter-clockwise (negative) and zero helicity.**

(a) Train and validation loss

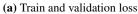**(b)** Train and validation accuracy

Figure 5: **Train and validation loss and accuracy for two classes.** Loss and accuracy for MobileNet v2, with a learning rate $\eta = 0.01$, a batch size of 4 and 16 epochs.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L. Chen, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4510–4520. doi:10.1109/CVPR.2018.00474.

[9] M. Kachelrieß, S. Ostapchenko, R. Tomas, Computer Physics Communications 183 (2012) 1036–1043.

[10] M. Blytt, M. Kachelrieß, S. Ostapchenko, Computer Physics Communications (2020) 107163.

[11] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, Li Fei-Fei, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. doi:10.1109/CVPR.2009.5206848.