Noname manuscript No.
(will be inserted by the editor)

Machine learning in Solar Physics

A. Asensio Ramos · M. C. M. Cheung

Received: date / Accepted: date

Abstract Insert your abstract here. Include keywords, PACS and mathematical subject classification numbers as needed.

Keywords First keyword · Second keyword · More

1 Introduction

Astrophysics is entering in the last decades in the
The field of Solar Physics has been historically

2 Machine learning

2.1 Linear pattern recognition

2.1.1 Principal component analysis

Principal Components Analysis (PCA; see Loeve 1955), also known as Karhunen-Loève transformation, is an algorithm used in multivariate statistics. Briefly, it is used to obtain a self-consistent basis on which the data can be efficiently developed. This basis has the property that the largest amount of variance is explained with the least number of basis vectors. It is useful to reduce the dimensionality of data sets that depend on a very large number of parameters. This property has

A. Asensio Ramos
Instituto de Astrofísica de Canarias, 38205, La Laguna, Tenerife, Spain
Departamento de Astrofísica, Universidad de La Laguna, 38205 La Laguna, Tenerife, Spain
E-mail: aasensio@iac.es

M. C. M. Cheung
Lockheed Martin Solar & Astrophysics Laboratory, 3251 Hanover St, Palo Alto, CA 94304, USA
Hansen Experimental Physics Laboratory, Stanford University, 452 Lomita Mall, Stanford, CA 94305, USA
E-mail: cheung@lmsal.com

been used for denoising purposes. For simplicity, we focus from on the problem of polarised spectra. When a spectrograph is used to observe a spectral line formed in a stellar atmosphere, it is sampled at a finite number of wavelengths, a number that depends on the spectral resolu- tion of the instrumental setup. However, this number is usually much larger than the number of physical variables involved in the spectral line formation mechanism (Asensio Ramos et al. 2007). Moreover, if we observe the full Stokes vector, the num- ber of wavelengths increases by a factor of 4, while the num- ber of physical parameters typically increases more slowly. It is easy to understand that correlations between the observables ex- ist. This is related to the fact that the presence of physical laws constrain the possible values that any observable can take. For instance, all the wavelength points tracing the continuum away from spectral lines provide roughly the same information about the physical conditions. Since the stellar continuum is typically formed in local thermodynamic equilibrium conditions, it can be characterized by a Planck function at a given temperature. Therefore, all the wavelength points are linked by the functional form of the Planck function. Due to these intrinsic correlations that exist in the observ- ables, when a spectral line is observed many times, or, in our case, several spectral lines are observed simultaneously, the cloud of points that represents all spectral lines in the multi- dimensional space of the observables will be elongated in some directions. These directions are the so-called principal compo- nents and the data can be efficiently reproduced as a linear com- bination of vectors along them.

Let us assume that the wavelength variation of each Stokes profile (I, Q, U, or V) of a particular spectral line is described by the quantity Sij. The index i represents the wavelength position while the index j = I, Q, U, V indicates the Stokes parame- ter. Each Stokes parameter is a vector of length N, correspond- ing to the number of wavelength points. In the ideal situation, it would be advantageous to have Nobs N observations, so that the number of observed lines is much larger than the number of wavelength points used to sample each line. Thanks to the cross- dispersed capabilities of instruments like SEMPOL, ESPaDOnS or NARVAL, a very large number of spectral lines is obtained in one exposure when recording spectro-polarimetric data. This allows us to apply statistical techniques to capture the intrinsic behavior of the points in N-dimensional space and to use PCA to reduce its dimensionality. We define O as the Nobs N matrix containing the wave- length variation of all the observed spectral lines. The principal components can be found as the eigenvectors of this matrix of observations. This means that the PCA procedure reduces to the diagonalisation of the matrix O . Since we require that Nobs N holds, this matrix is not square by definition. Moreover, even if one uses the Singular Value Decomposition (SVD; see, e.g., Press et al. 1986) to diagonalise O, the dimension of the matrix can be so large that computational problems can arise. It can be demonstrated that the right singular vectors of of the matrix O are equal to the singular vectors of the cross-product matrix: X = O t O . (1) The matrix X is the N N cross-product matrix and the su- perindex t represents the transposition operator. The same ap- plies to the left singular vectors, which are also eigenvectors of the cross-product matrix X = O O t . The matrix X has dimen- sions Nobs Nobs and is typically much larger than the matrix X . However, one description is the dual of the other and they are completely equivalent. The ith principal component, Bi, fulfills: XBi=kiBi, (2) with ki its associated eigenvalue. All the eigenvectors can be put together in the matrix

B. This matrix has dimensions N N and contains the eigenvectors as column vectors. Note that the cumulative distribution of eigenvalues m k gm=i=1i (3) N ki i=1 gives the relative amount of variance explained by the first m eigenvectors. Since these vectors constitute a basis, the observa- tions can be written as a linear combination of them as follows: O = C B t , (4) C being the Nobs N matrix of coefficients. The element Ci j of this matrix represents the projection of the observation i on the eigenvector j. This matrix can be easily calculated as: C = O B . (5) Note that the transposition operator of the matrix of the eigen- vectors in Eq. (4) replaces the inverse operator because the ma- trix of singular vectors is orthogonal, so that it fulfills B1 = Bt. This greatly simplifies the calculations because no numerical matrix inversion is needed.

2.2 Nonlinear pattern recognition

2.2.1 Kernel principal component analysis

2.3 Artificial neural networks

Artificial neural networks (ANN) are well-known computing systems based on connectionism that can be considered to be very powerful approximants to arbitrary functions (Bishop, 1996). They are constructed by putting together many basic fundamental structures (called neurons) and connecting them massively. Each neuron i is only able to carry out a very basic operation on the input vector: it multiplies all the input values x_j by some weights w_j, adds some bias b_i and finally returns the value of a certain user-defined nonlinear activation function $f(x)$. In mathematical notation, a neuron computes:

$$o_i = f(\Sigma_j\, x_j \cdot w_j + b_i). \tag{1}$$

The output o_i is then input in another neuron that does a similar work.

An ANN can be understood as a pipeline where the information goes from the input to the output, where each neuron makes a transformation like the one described above (see left panel of Fig. 1). Given that neurons are usually grouped in layers, the term deep neural network comes from the large number of layers that are used to build the neural network. Some of the most successful and recent neural networks contain several millions of neurons organized in several tens or hundreds of layers Simonyan and Zisserman (2014). As a consequence, deep neural networks can be considered to be a very complex composition of very simple nonlinear functions, which gives the capacity to do very complex transformations.

The most used type of neural network from the 1980s to the 2000s is the fully connected network (FCN; see Schmidhuber, 2014, for an overview), in which every input is connected to every neuron of the following layer. Likewise, the output transformation becomes the input of the following layer (see left panel of Fig. 1). This kind of architecture succeeded to solve problems that were considered to be not easily solvable as the recognition of handwritten characters Bishop (1996). A selection of applications in Solar Physics include the inversion of Stokes profiles (e.g., Socas-Navarro, 2005a; Carroll and Kopf, 2008a), the acceleration of the solution of chemical equilibrium Asensio Ramos and Socas-Navarro (2005) and the automatic classification of sunspot groups Colak and Qahwaji (2008).

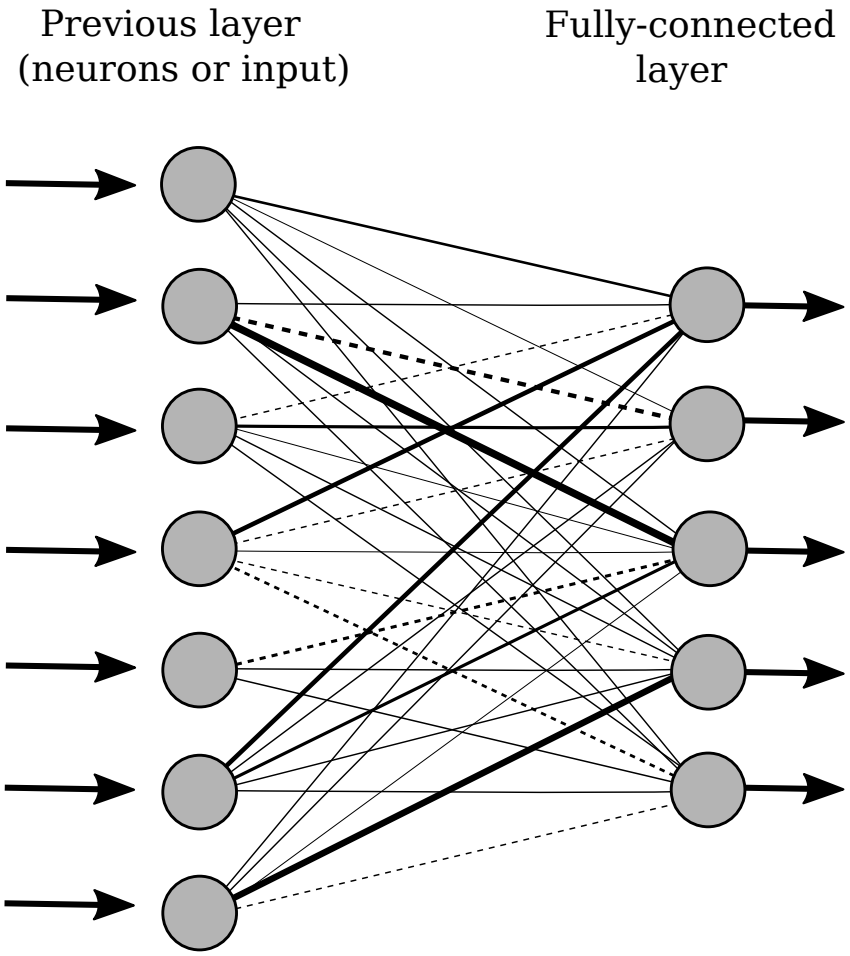

Fig. 1 Left panel: building block of a fully-connected neural network. Each input of the previous layer is connected to each neuron of the output. Each connection is represent by different lines where the width is associated to higher weights and the dashed lines to negative weights. Right panel: three-dimensional convolution carried out by a convolutional layer. The 3D-kernel traverses the whole input, producing a single scalar at each position. At the end, a 2D feature map will be created for each 3D kernel. When all feature maps are stacked, a feature map tensor will be created.

Neural networks are optimized iteratively by updating the weights and biases so that a loss function that measures the ability of the network to predict the output from the input is minimized[1]. This optimization is widely known as learning or training process. In this process a training dataset is required.

[1] This is the case of supervised training. Unsupervised neural networks are also widespread but are of no concern in this paper.

The Artificial Neural Network (ANN) with one hidden layer is a universal approximant to any non-linear continuous function (e.g., Jones 1990; Blum & Li 1991). The schematic structure of such a network is shown in Fig. 1. We have constructed an ANN with an input layer of three neurons where the temperature, hydrogen density and electron density are in- troduced. The neurons of the input layer have a linear activa- tion function. The hidden layer consists on Nh neurons with a non-linear activation function (x). Each hidden neuron is con- nected to all the neurons of the input layer by a certain weight (indicated by arrows in the figure). The value obtained at each hidden neuron is a linear combination of the values at the neu- rons of the input layer multiplied by the weights. These values are then applied the non-linear function (x), multiplied by an- other set of weights and summed to give the final output of the neural network. The output of the network can be written as: Nh N(T,n(H),n(e),m)= vjwTjT+wHjn(H) j +wejn(e)+uj , (13) where the activation function is usually given by the sigmoid or the hyperbolic-tangent function. In this case, we have selected (x) = tanh(x). In the previous expression, m formally repre- sents the whole set of 5 Nh weights which define the neural network.

2.3.1 Multi-layer perceptron

2.3.2 Convolutional neural networks

Motivated by the recent enormous success of deep neural networks (Goodfellow et al., 2016), we propose in this work to leverage DNN to carry out three-dimensional inversions of the solar atmosphere under the assumption of LTE for Hinode-like observations. Our approach is termed (standing for). We defer for the future the study of a neural approach for the inversion of spectropolarimetric data at the diffraction limit of a 4m class telescope like DKIST or EST and/or including non-LTE effects. Data-driven approaches using DNNs have already been applied to solar physics, demonstrating striking capabilities to solve problems that could not be solved otherwise or with a much improved precision and/or speed. For instance, we managed to infer transverse velocities from pairs of consecutive images (DeelVel; Asensio Ramos et al., 2017), which allowed us to identify small-scale vortices in the solar atmosphere that last for a few minutes and with sizes of the order of a few hundred kilometers, something impossible with methods based on local correlation tracking November and Simon (1988). We have also applied them to compensate for the blurring effect of the telescope and the Earth atmosphere Díaz Baso and Asensio Ramos (2018); Asensio Ramos et al. (2018). Illarionov and Tlatov (2018) applied DNNs for the automatic segmentation of coronal holes with great success. The advances in the field of deep learning suggests it is timely to investigate what are the prospects of neural networks in the field of spectropolarimetric inversions. In this paper we leverage convolutional neural networks (CNN; LeCun and Bengio, 1998) that can easily exploit spatial information and the ability to train really deep neural networks that can approximate very nonlinear mappings.

Finally, we point out that the application of neural networks (multi-layer perceptrons; MLP) for the inversion of Stokes profiles is not new. Carroll and Staude (2001) already proposed their use for simple Milne-Eddington inversions and concluded that they were able to obtain physical parameters without any optimization

once the neural networks were trained. As additional advantages, they showed their speed, noise tolerance and stability. This was later verified by other works Socas-Navarro (2003, 2005b). Carroll and Kopf (2008b) later expanded their original work to use MLPs to infer the depth stratification in a geometrical height scale of the temperature, velocity and magnetic field vector, as their network was trained with a quiet Sun simulation Vögler et al. (2005). The application of the neural network pixel by pixel allowed them to recover a tomographic view of the FOV by recombining all individual line-of-sight stratifications. More recently, Osborne et al. (2019) has shown how invertible neural networks (INNs; Ardizzone et al., 2018) can be successfully applied to capture degeneracies and ambiguities in the inference of thermodynamic parameters of plane-parallel chromospheres of flaring regions.

2.4 Convolutional neural networks

In spite of the relative success of neural networks, their application to high-dimensional objects like images or videos turned out to be an obstacle. The fundamental reason was that the number of weights in a fully connected network increases extremely fast with the complexity of the network (number of neurons) and the computation quickly becomes unfeasible. As each neuron has to be connected with the whole input, if we add a new neuron we will add the size of the input in number of weights. Then, a larger number of neurons implies a huge number of connections. This constituted an apparently unsurmountable handicap that was only solved with the appearance of convolution neural networks (CNN or ConvNets; LeCun and Bengio, 1998).

The most important ingredient in the CNN is the convolutional layer which is composed of several convolutional neurons. Each CNN-neuron carries out the convolution of the input with a certain (typically small) kernel, providing as output what is known as feature map. Similar to a FCN, the output of convolutional neurons is often passed through a nonlinear activation function. The fundamental advantage of CNNs is that the same weights are shared across the whole input, drastically reducing the number of unknowns. This also makes CNN shift invariant (features can be detected in an image irrespectively of where they are located).

In mathematical notation, for a two-dimensional input X of size $N \times N$ with C channels[2] (really a cube or tensor of size $N \times N \times C$), each output feature map O_i (with size $N \times N \times 1$) of a convolutional layer is computed as:

$$O_i = K_i * X + b_i, \tag{2}$$

where K_i is the $K \times K \times C$ kernel tensor associated with the output feature map i, b_i is a bias value ($1 \times 1 \times 1$) and the convolution is displayed with the symbol $*$. Once the convolution with M different kernels is carried out and stacked together, the output O will have size $N \times N \times M$. All convolutions are here indeed intrinsically three dimensional, but one could see them as the total of $M \times C$ two dimensional convolutions plus the bias (see right panel of Fig. 1).

[2] The term channels is inherited from the those of a color image (e.g., RGB channels). However, the term has a much more general scope and can be used for arbitrary quantities (see Asensio Ramos et al., 2017, for an application).

CNNs are typically composed of several layers. This layerwise architecture exploits the property that many natural signals are a generated by a hierarchical composition of patterns. For instance, faces are composed of eyes, while eyes contain a similar internal structure. This way, one can devise specific kernels that extract this information from the input. As an example, Fig. **??** shows the effect of a vertical border detection kernel on a real solar image. The result at the right of the figure is the feature map. CNNs work on the idea that each convolution layer extracts information about certain patterns, which is done during the training by iteratively adapting the set of convolutional kernels to the specific features to locate. This obviously leads to a much more optimal solution as compared with hand-crafted kernels. Despite the exponentially smaller number of free parameters as compared with a fully-connected ANN, CNNs produce much better results. It is interesting to note that, since a convolutional layer just computes sums and multiplications of the inputs, a multi-layer FCN (also known as perceptron) is perfectly capable of reproducing it, but it would require more training time (and data) to learn to approximate that mode of operation Peyrard et al. (2015).

Although a convolutional layer significantly decreases the number of free parameters as compared with a fully-connected layer, it introduces some hyperparameters (global characteristics of the network) to be set in advance: the number of kernels to be used (number of feature maps to extract from the input), size of each kernel with its corresponding padding (to deal with the borders of the image) and stride (step to be used during the convolution operation) and the number of convolutional layers and specific architecture to use in the network. As a general rule, the deeper the CNN, the better the result, at the expense of a more difficult and computationally intensive training. CNNs have been used recently in astrophysics for denoising images of galaxies Schawinski et al. (2017), for cosmic string detection in CMB temperature maps Ciuca et al. (2017), or for the estimation of horizontal velocities in the solar surface Asensio Ramos et al. (2017) .

2.5 Activation layers

As said, the output of a convolutional layer is often passed through a non-linear function that is termed the activation function. Since the convolution operation is linear, this activation is the one that introduces the non-linear character of the CNNs. Although hyperbolic tangent, $f(x) = \tanh(x)$, or sigmoidal, $f(x) = [1 + \exp(-x)]^{-1}$, activation units were originally used in ANNs, nowadays a panoply of more convenient nonlinearities are used. The main problem with any sigmoid-type activation function is that its gradient vanishes for very large values, difficulting the training of the network. Probably the most common activation function is the rectified linear unit (ReLU; Nair and Hinton, 2010) or slight variations of it. The ReLU replaces all negative values in the input by zero and keeps the rest untouched. This activation has the desirable property of producing non-vanishing gradients for positive arguments, which greatly accelerates the training.

CNNs are trained by iteratively modifying the weights and biases of the convolutional layers (and any other possibly learnable parameter in the activation layer). The aim is to optimize a user-defined loss function from the output of the network and the desired output of the training data. The optimization is routinely solved using simple first-order gradient descent algorithms (GD; see Rumelhart

et al., 1988), which modifies the weights along the negative gradient of the loss function with respect to the model parameters to carry out the update. The gradient of the loss function with respect to the free parameters of the neural network is obtained through the backpropagation algorithm LeCun et al. (1998). Given that neural networks are defined as a stack of modules (or layers), the gradient of the loss function can be calculated using the chain rule as the product of the gradient of each module and, ultimately, of the last layer and the specific loss function.

In practice, procedures based on the so-called stochastic gradient descent (SGD) are used, in which only a few examples (termed batch) from the training set are used during each iteration to compute a noisy estimation of the gradient and adjust the weights accordingly. Although the calculated gradient is a noisy estimation of the one calculated with the whole training set, the training is faster as we have less to compute and more reliable. If the general loss function Q is the average of each loss Q_j computed on a batch of inputs and it can be written as $Q = \Sigma_j^n Q_j/n$, the weights w_i are updated following the same recipe as the gradient descend algorithm but calculating the gradient within a single batch:

$$w_{i+1} = w_i - \eta \nabla Q(w_i) = w_i - \eta \nabla \Sigma_j^n Q_j(w_i)/n \simeq w_i - \eta \nabla Q_j(w_i), \tag{3}$$

where η is the so-called learning rate. It can be kept fixed or it can be changed according to our requirements. This parameter has to be tuned to find a compromise between the accuracy of the network and the speed of convergence. If η is too large, the steps will be too large and the solution could potentially overshoot the minimum. On the contrary, if it is too small it will take so many iterations to reach the minimum. Adaptive methods like Adam Kingma and Ba (2014) have been developed to automatically tune the learning rate.

Because of the large number of free parameters in a deep CNNs, overfitting can be a problem. One would like the network to generalize well and avoid any type of "memorization" of the training set. To check for that, a part of the training set is not used during the update of the weights but used after each iteration as validation. Desirably, the loss should decrease both in the training and validation sets simultaneously. If overfitting occurs, the loss in the validation set will increase.

Moreover, several techniques have been described in the literature to accelerate the training of CNNs and also to improve generalization. Batch normalization Ioffe and Szegedy (2015) is a very convenient and easy-to-use technique that consistently produces large accelerations in the training. It works by normalizing every batch to have zero mean and unit variance. Mathematically, the input is normalized so that:

$$y_i = \gamma \hat{x_i} + \beta \hat{x_i} = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}, \tag{4}$$

where μ and σ are the mean and standard deviation of the inputs on the batch and $\epsilon = 10^{-3}$ is a small number to avoid underflow. The parameters γ and β are learnable parameters that are modified during the training.

Following the typical scheme of a residual block, there is also a shortcut connection between the input and the output of the block (see more information in **?**Asensio Ramos et al., 2017), so that the input is added to the output. Very deep networks usually saturate during training producing higher errors than shallow networks because of difficulties during training (also known as the degradation problem). The fundamental reason is that the gradient of the loss function with respect to parameters in early layers becomes exponentially small (also known as the

vanishing gradient problem). Residual networks help avoid this problem obtaining state-of-the-art results without adding any extra parameter and with practically the same computational complexity. It is based on the idea that if $y = F(x)$ represents the desired effect of the block on the input x, it is much simpler for a network to learn the deviations from the input (or residual mapping) that it can called $R(x) = y - x$ than the full map $F(x)$, so that $y = F(x) = R(x) + x$.

2.5.1 Loss function

2.5.2 Backpropagation

3 Heliosphere

4 Flare prediction

5 Inversion of Stokes profiles

5.1 Database search

5.2 Artificial neural networks

6 Image deconvolution

Acknowledgements Financial support by the Spanish Ministry of Economy and Competitiveness through project AYA2014-60476-P. This research has made use of NASA's Astrophysics Data System Bibliographic Services.

References

Ardizzone L, Kruse J, Wirkert S, Rahner D, Pellegrini EW, Klessen RS, Maier-Hein L, Rother C, Köthe U (2018) Analyzing Inverse Problems with Invertible Neural Networks. arXiv e-prints arXiv:1808.04730, 1808.04730

Asensio Ramos A, Socas-Navarro H (2005) An artificial neural network approach to the solution of molecular chemical equilibrium. A&A438:1021–1028, DOI 10.1051/0004-6361:20052865, astro-ph/0505322

Asensio Ramos A, Requerey IS, Vitas N (2017) DeepVel: Deep learning for the estimation of horizontal velocities at the solar surface. A&A604:A11, DOI 10.1051/0004-6361/201730783, 1703.05128

Asensio Ramos A, de la Cruz Rodríguez J, Pastor Yabar A (2018) Real-time, multiframe, blind deconvolution of solar images. A&A620:A73, DOI 10.1051/0004-6361/201833648, 1806.07150

Bishop CM (1996) Neural networks for pattern recognition. Oxford University Press

Carroll TA, Kopf M (2008a) Zeeman-tomography of the solar photosphere. Three-dimensional surface structures retrieved from Hinode observations. A&A481:L37–L40, DOI 10.1051/0004-6361:20079197, 0803.1048

Carroll TA, Kopf M (2008b) Zeeman-tomography of the solar photosphere. Three-dimensional surface structures retrieved from Hinode observations. A&A481:L37–L40, DOI 10.1051/0004-6361:20079197, 0803.1048

Carroll TA, Staude J (2001) The inversion of Stokes profiles with artificial neural networks. A&A378:316–326, DOI 10.1051/0004-6361:20011167

Ciuca R, Hernández OF, Wolman M (2017) A Convolutional Neural Network For Cosmic String Detection in CMB Temperature Maps. ArXiv e-prints 1708.08878

Colak T, Qahwaji R (2008) Automated McIntosh-Based Classification of Sunspot Groups Using MDI Images. Sol. Phys.248:277–296, DOI 10.1007/s11207-007-9094-3

Díaz Baso CJ, Asensio Ramos A (2018) Enhancing SDO/HMI images using deep learning. A&A614:A5, DOI 10.1051/0004-6361/201731344, 1706.02933

Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. MIT Press, http://www.deeplearningbook.org

Illarionov EA, Tlatov AG (2018) Segmentation of coronal holes in solar disc images with a convolutional neural network. MNRAS481:5014–5021, DOI 10.1093/mnras/sty2628, 1809.05748

Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Blei D, Bach F (eds) Proceedings of the 32nd International Conference on Machine Learning (ICML-15), JMLR Workshop and Conference Proceedings, pp 448–456, URL http://jmlr.org/proceedings/papers/v37/ioffe15.pdf

Kingma DP, Ba J (2014) Adam: A Method for Stochastic Optimization. ArXiv e-prints 1412.6980

LeCun Y, Bengio Y (1998) In: Arbib MA (ed) The Handbook of Brain Theory and Neural Networks, MIT Press, Cambridge, MA, USA, chap Convolutional Networks for Images, Speech, and Time Series, pp 255–258, URL http://dl.acm.org/citation.cfm?id=303568.303704

LeCun Y, Bottou L, Orr GB, Müller KR (1998) Efficient backprop. In: Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop, Springer-Verlag, London, UK, UK, pp 9–50, URL http://dl.acm.org/citation.cfm?id=645754.668382

Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel, pp 807–814, URL http://www.icml2010.org/papers/432.pdf

November LJ, Simon GW (1988) Precise Proper-Motion Measurement of Solar Granulation. ApJ333:427, DOI 10.1086/166758

Osborne CMJ, Armstrong JA, Fletcher L (2019) RADYNVERSION: Learning to Invert a Solar Flare Atmosphere with Invertible Neural Networks. arXiv e-prints arXiv:1901.08626, 1901.08626

Peyrard C, Mamalet F, Garcia C (2015) A comparison between multi-layer perceptrons and convolutional neural networks for text image super-resolution. In: Braz J, Battiato S, Imai FH (eds) VISAPP (1), SciTePress, pp 84–91

Rumelhart DE, Hinton GE, Williams RJ (1988) Neurocomputing: Foundations of research. MIT Press, Cambridge, MA, USA, chap Learning Representations by Back-propagating Errors, pp 696–699, URL http://dl.acm.org/citation.cfm?id=65669.104451

Schawinski K, Zhang C, Zhang H, Fowler L, Santhanam GK (2017) Generative adversarial networks recover features in astrophysical images of galaxies beyond the deconvolution limit. MNRAS467:L110–L114, DOI 10.1093/mnrasl/slx008, 1702.00403

Schmidhuber J (2014) Deep Learning in Neural Networks: An Overview. ArXiv e-prints 1404.7828

Simonyan K, Zisserman A (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. ArXiv e-prints 1409.1556

Socas-Navarro H (2003) Measuring Solar Magnetic Fields with Artificial Neural Networks. Neural Networks 16:355

Socas-Navarro H (2005a) Strategies for Spectral Profile Inversion Using Artificial Neural Networks. ApJ621:545–553, DOI 10.1086/427431, astro-ph/0410567

Socas-Navarro H (2005b) Strategies for Spectral Profile Inversion Using Artificial Neural Networks. ApJ621:545–553, DOI 10.1086/427431, astro-ph/0410567

Vögler A, Shelyag S, Schüssler M, Cattaneo F, Emonet T, Linde T (2005) Simulations of magneto-convection in the solar photosphere. Equations, methods, and results of the MURaM code. A&A429:335–351, DOI 10.1051/0004-6361:20041507