

# Supervised neural networks for helioseismic ring-diagram inversions

Rasha Alshehhi<sup>1</sup>, Chris S. Hanson<sup>2,1</sup>, Laurent Gizon<sup>2,3,1</sup>, and Shравan Hanasoge<sup>4,1</sup>

<sup>1</sup> Center for Space Science, NYUAD Institute, New York University Abu Dhabi, Abu Dhabi, UAE  
e-mail: ra13@nyu.edu

<sup>2</sup> Max-Planck-Institut für Sonnensystemforschung, Justus-von-Liebig-Weg 3, 37077 Göttingen, Germany  
e-mail: hanson@mps.mpg.de

<sup>3</sup> Institut für Astrophysik, Georg-August-Universität Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany

<sup>4</sup> Tata Institute of Fundamental Research, Homi Bhabha Road, Mumbai 400005, India

Received 13 September 2018 / Accepted 24 December 2018

## ABSTRACT

**Context.** The inversion of ring fit parameters to obtain subsurface flow maps in ring-diagram analysis for eight years of SDO observations is computationally expensive, requiring ~3200 CPU hours.

**Aims.** In this paper we apply machine-learning techniques to the inversion step of the ring diagram pipeline in order to speed up the calculations. Specifically, we train a predictor for subsurface flows using the mode fit parameters and the previous inversion results to replace future inversion requirements.

**Methods.** We utilize artificial neural networks (ANNs) as a supervised learning method for predicting the flows in 15° ring tiles. We discuss each step of the proposed method to determine the optimal approach. In order to demonstrate that the machine-learning results still contain the subtle signatures key to local helioseismic studies, we use the machine-learning results to study the recently discovered solar equatorial Rossby waves.

**Results.** The ANN is computationally efficient, able to make future flow predictions of an entire Carrington rotation in a matter of seconds, which is much faster than the current ~31 CPU hours. Initial training of the networks requires ~3 CPU hours. The trained ANN can achieve a rms error equal to approximately half that reported for the velocity inversions, demonstrating the accuracy of the machine learning (and perhaps the overestimation of the original errors from the ring-diagram pipeline). We find the signature of equatorial Rossby waves in the machine-learning flows covering six years of data, demonstrating that small-amplitude signals are maintained. The recovery of Rossby waves in the machine-learning flow maps can be achieved with only one Carrington rotation (27.275 days) of training data.

**Conclusions.** We show that machine learning can be applied to and perform more efficiently than the current ring-diagram inversion. The computation burden of the machine learning includes 3 CPU hours for initial training, then around 10<sup>-4</sup> CPU hours for future predictions.

**Key words.** Sun: helioseismology – Sun: oscillations – Sun: interior – methods: numerical

## 1. Motivation: Speeding up ring-diagram inversions

Local helioseismology seeks to image the subsurface flows utilizing the complex wave field observed at the surface (see review by Gizon & Birch 2005). The procedure of imaging the subsurface flow fields from the Dopplershift images of the surface is summarized as follows: projection and tracking of large Dopplergram times series, transformation into Fourier space, analysis of perturbations in the acoustic power spectrum (local frequency shifts), and inversions. The refinement of large data sets into the significantly smaller flow maps is computationally expensive, and has to be repeated for all future observations. The field of machine learning seeks to develop data-driven models that, given sufficient training samples, will predict future observations, without the need for the aforementioned procedure. With over 20 years of space-based observations, the field of local helioseismology now possesses large amounts of data that can be utilized by machine-learning algorithms to improve existing techniques, or find relationships previously unknown to the field.

The field of machine learning grew out of the work to build artificial intelligence. The application of machine learning is broad in both scientific research and industries that analyze “big

data”, with some impressive results (e.g., Pearson et al. 2018). However, the field of local helioseismology has thus far not used this technique, despite the advantages it could provide given the large amounts of data available. However, some work has been done on the use of deep learning for multi-height local correlation tracking near intergranular lanes (Asensio Ramos et al. 2017).

A widely used technique in local helioseismology is ring-diagram analysis (see Antia & Basu 2007, for detailed review). First presented by Hill (1988), the ring diagram technique analyzes slices (at constant frequency  $\omega$ ) of the 3D power spectrum  $P(\omega, k_x, k_y)$  of an observed (tracked and projected) patch of the solar surface  $D(t, x, y)$ . The cross-section of the power spectrum reveals rings corresponding to the acoustic normal modes of the Sun ( $f$ - and  $p$ -modes). In the absence of flows these rings are symmetric in  $k_x$  and  $k_y$ . However, the presence of flows in the zonal ( $x$ ) or meridional ( $y$ ) directions breaks symmetry of these rings, manifesting as ellipsoids. Acoustic modes traveling with or against the direction of flow experience increases or decreases in travel time, resulting in changes to the phase speed. The frequency shift of a ring is then considered as a horizontal average of the flows within the observed patch. Each mode (ring) is then

fit, and the mode-fit parameters  $U_x$  and  $U_y$  determined, revealing the “flow” required to produce the shifts in each mode (ring). The true subsurface flow field is then considered as a linear combination of the mode-fits. In order to determine the flow field, an inversion is performed using the mode-fit parameters  $U_x$  and  $U_y$  and the sensitivity kernels  $\mathcal{K}^{n\ell}(z)$  that relate frequency shifts of a mode of radial order  $n$  and harmonic degree  $\ell$  to the horizontal velocity components  $u_x$  and  $u_y$  at height  $z$  in the interior (by convention,  $z$  is negative in the interior and zero at the photosphere):

$$U_x^{n\ell} = \int u_x(z) \mathcal{K}^{n\ell}(z) dz, \quad U_y^{n\ell} = \int u_y(z) \mathcal{K}^{n\ell}(z) dz, \quad (1)$$

where  $\mathcal{K}^{n\ell}$  is derived from solar models. The solution to the linear inversion is then a combination of the mode fits and coefficients  $c^{n\ell}$  that give maximum sensitivity at a target height  $z_t$ ,

$$u_x(z_t) = \sum_{n,\ell} c^{n\ell}(z_t) U_x^{n\ell}, \quad u_y(z_t) = \sum_{n,\ell} c^{n\ell}(z_t) U_y^{n\ell}. \quad (2)$$

The ring-diagram pipelines that derive these coefficients (Bogart et al. 2011a,b) are computationally expensive, requiring 16 s per tile or  $\sim 3200$  CPU hours for the entire SDO data set.

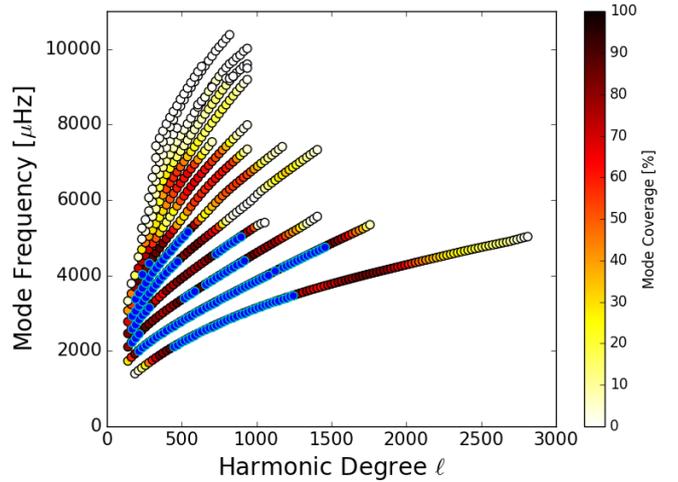
Our endeavor is to use the large data sets available in the pipeline to improve the ring diagram procedure by using deep networks. Specifically, we seek (through machine learning) to determine the complex mapping from the raw Doppler time series to the flows. In this study we present initial results in performing the inversion without the need for any inversions or kernels, using machine-learning techniques.

In Sect. 2 we present the data to be used and the proposed machine-learning technique. Section 3 examines the performance of the proposed method with a number of machine-learning techniques in both accuracy and computational burden. Section 4 examines a case study of the effect of machine learning on equatorial Rossby waves. Discussions and conclusions are given in Sect. 5.

## 2. Proposed method

Machine-learning studies can be divided into two branches, unsupervised and supervised learning. This study will be of the latter kind, in which we know the flow corresponding to the mode fits (from the current pipeline) and thus train an estimator to predict flow values given a new set of mode fits. While no study has directly examined the accuracy of ring-diagram analysis, the results of a number of studies have remained consistent with other measurement techniques (e.g., Giles et al. 1997; Schou & Bogart 1998). It is possible that the existing pipeline has systematic errors that affect the inversion results. Any supervised learning method will inherit these problems, as this is the weakness of data-driven models. If problems were found and resolved in the pipeline, any trained machine-learning models would have to be retrained for the correct flows, although this will not invalidate the results of this study. The proposed supervised method of this study comprises two main components: preprocessing and applying artificial neural networks (ANN) for regression, both of which are described in detail in the following sections.

For clarity in the terminology used here, we define the following. Input data consist of a large number  $N_{\text{obs}}$  of tiles, each with a number of mode-fits/features  $N_{\text{features}}$  identified in the ring pipeline. The output data consist of  $N_{\text{obs}}$  flow values, corresponding to each input tile, for 31 depths.



**Fig. 1.** Mode coverage of each uniquely identified  $[n, \ell]$  mode in the mode-fits pipeline from CR2097 to CR2201. The modes highlighted with blue appear in more than 90% of the tiles and therefore are used in this study. All other modes are neglected.

### 2.1. Extraction of features from the pipeline

The ring-diagram pipeline (Bogart et al. 2011a,b) developed for use on the high-resolution, high-cadence data of the Helioseismic and Magnetic Imager (HMI, Schou et al. 2012) has provided unprecedented analysis of the subsurface flows of the Sun. The data for each step in the pipeline are available on the NetDRMS<sup>1</sup> database, and for this study we use the mode fits  $U_x$  and  $U_y$  and the inverted flows  $u_x$  and  $u_y$ . Due to the statistical nature of the machine learning, we ignore the derived error values of the fits in the training. However, in Sect. 3.2 we show the effect realization noise has on the machine-learning predictions.

The SDO program has been running since 2010, and has observed over 100 Carrington rotations (CRs). For this study we make use of the data from CR2097 to CR2201, which covers eight years. Each CR consists of a maximum of 6816 tiles, but some rotations have less coverage. In total, over the 104 rotations, there are 709 734 inversion results available in the pipeline. Additionally, we focus this study on the  $15^\circ$  maps, which upon inversion probe depths down to 20.88 Mm below the photosphere.

Each tile has a number of mode fits that have been detected by the pipeline. From tile to tile the presence of these modes can vary; sometimes detected, other times overlooked. For this study each unique mode, with radial order  $n$  and harmonic degree  $\ell$ , is considered an independent feature. The presence of a single mode in all tiles is called the mode coverage. In order to avoid bias from missing modes, we reduce the number of features by applying strict mode coverage requirements. Specifically, for a single mode, if it is detected in less than 90% of the tiles then it is neglected. This significantly reduces the number of features in the machine learning, and minimizes any bias (to zero) we have from missing data. Upon application of this mode-coverage requirement, 152 separate modes remain. Figure 1 shows the mode coverage for all modes detected in the pipeline, as well as the modes selected for this machine-learning study.

In summary, the dataset (for each horizontal component  $x$  and  $y$  of the flow) consists of 709 734 tiles (samples) with 152 features (modes) each specifying the frequency shifts of acoustic waves due to flows. The corresponding target consists of 709 734

<sup>1</sup> <http://jsoc.stanford.edu/netdrms/>

flow vectors for each of the 30 depths from a depth of 0.696 Mm to 20.88 Mm.

## 2.2. Preprocessing of input features

The goal of preprocessing the input features is to produce more effective features which show high contrasts. Typically, this involves three steps: interpolation to fill missing data, normalization of the input features, and reduction of the number of input features.

Previously, we neglected modes that appeared in less than 90% of the tiles. However, many of the tiles still do not have complete mode coverage. The problem of how to handle missing data is well known in machine learning. In statistics and machine-learning literature, the replacement of missing data is known as data imputation. A number of solutions exist for completing the data set (e.g., [Little & Rubin 1987](#)). However, for this study we use the simple solution of replacing the missing values by the mean of the mode.

The next stage is the normalization of each feature, which brings all features into a similar dynamic range. The need for this is driven by the fact that many machine-learning techniques will be dominated by features with a large range, while narrow ranged features can be ignored. To avoid this, normalization of each feature is performed in order to bring all features into the same range, and thus remove any preference for a specific feature. There are a number of different approaches that can be taken to achieve this (e.g., minimum-maximum feature scaling, standard score, student's t-statistic and studentized residual), and we recommend the reader to read [Juszczak et al. \(2002\)](#) for an in-depth explanation of each approach. For this study we use standard score. The transformation of a vector of frequency-shift measurements  $X$  to the new normalized feature  $\tilde{X}_i$  of observation  $i$  is as follows:

$$\tilde{X}_i = \frac{X_i - \mu}{\sigma}, \quad (3)$$

where  $\mu$  is the mean of the elements of  $X$  and  $\sigma$  the standard deviation. The new features  $\tilde{X}$  will have a zero mean with unit variance.

The final step in the preprocessing of this study is reduction of the 152 features (modes) to a smaller number, in order to ease computational burden.

Typically, the processing of features is done through either feature selection, that is, choosing only a subsample of modes, or feature reduction, where new feature space is generated from original modes (see [Alpaydin 2010](#), Chap. 6). By limiting our study to those 152 features with sufficient mode coverage, we have already performed feature selection. However, the remaining number of modes is still quite high and we seek to further reduce this through feature reduction. Here, feature reduction is achieved through Canonical Correlation Analysis (CCA; [Hotelling 1936](#); [Hardoon et al. 2004](#)).

The CCA seeks linear combinations of the input data  $X$  and output data  $Y$  (flow velocities), which have a maximum correlation with each other. Specifically, we seek Canonical Correlation Vectors  $a_i$  and  $b_i$  that maximize the correlation

$$\rho(a_i, b_i) = \text{Corr}(a_i^T X, b_i^T Y). \quad (4)$$

Following the method outlined by [Härdle & Simar \(2007\)](#), it can be shown that  $a_i$  and  $b_i$  are related to the covariance matrices  $\Sigma_{XX} = \text{Cov}(X, X)$  and  $\Sigma_{YY} = \text{Cov}(Y, Y)$  through

$$\begin{aligned} a_i &= \Sigma_{XX}^{-1/2} \mathcal{U}_i, \\ b_i &= \Sigma_{YY}^{-1/2} \mathcal{V}_i, \end{aligned} \quad (5)$$

where  $\mathcal{U}_i$  and  $\mathcal{V}_i$  are the  $i$ th left and right singular vectors from the following singular value decomposition (SVD):

$$\text{SVD}(\Sigma_{XX}^{-1/2} \Sigma_{XY} \Sigma_{YY}^{-1/2}) = [\dots \mathcal{U}_i \dots] \Lambda [\dots \mathcal{V}_j \dots]^T, \quad (6)$$

with  $\Lambda$  the diagonal matrix of singular values and  $\Sigma_{XY} = \text{Cov}(X, Y)$ . It remains to be determined how many Canonical correlation vectors are required to capture the relationship between  $X$  and  $Y$ . In Sect. 3.2 we show that upon application of CCA the number of features in the input data reduces from 152 to 1, for each depth and flow component.

Figure 2a shows the coefficients of the modes for feature reduction, computed through CCA for two target depths. Over plotting the phase speed corresponding to modes with a lower turning point at the target depth shows that the CCA gives more weight to modes that are sensitive to horizontal flows at the target depth. Thus, while we have reduced the 152 features to 1 (for each depth and flow component) the sensitivity to horizontal flows at the target depth is maintained.

Figure 2b compares the averaging kernel computed for tile `hmi.rdVtrack_fd15[2196][240][240][0][0]` with one built with the coefficients derived through the CCA. While the CCA finds the coefficients that maximize the correlations between mode-fits and flows for all tiles, the results show that the kernels compare reasonably well (despite no prior requirement on depth localization). We note that the CCA kernels are more sensitive to the surface for deep  $z_t$ , indicating that the CCA may find some depth correlations. The exact correlation of flows with depth is beyond the scope of this study, but is worth future investigation.

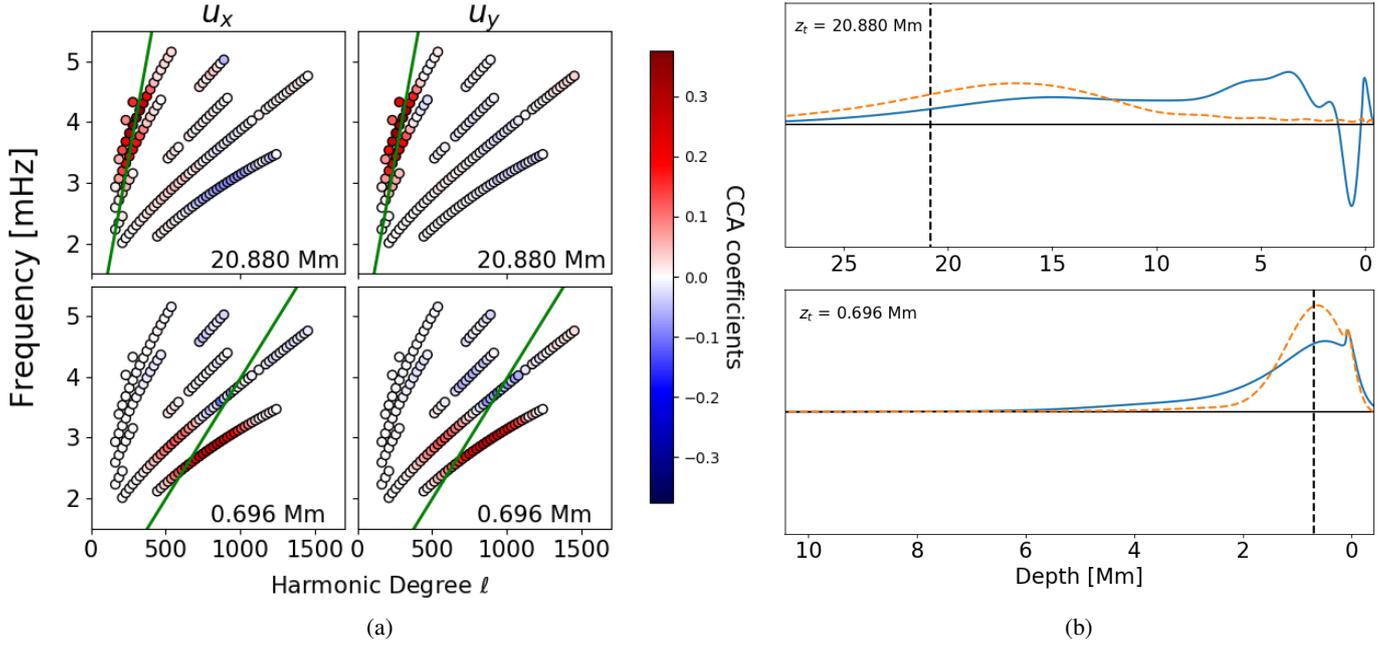
A schematic of the entire preprocessing pipeline is shown in Fig. 3.

## 2.3. Neural networks

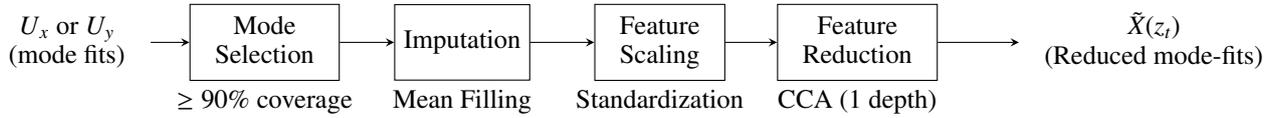
The ANN is one of the most common supervised machine learning methods and has been described in a wide range of literature (e.g., [Hand et al. 2001](#); [Haykin 2009](#); [Bishop 2006](#)). While many other methods exist in machine learning, we have found that the ANN performs best for this study (see Sect. 3.2), and therefore we explain here the ANNs that we use in detail. For an overview of other methods, see [Alpaydin \(2010\)](#). One advantage of the ANN is that it can solve nonlinear problems, which arise in this study due to different mode sets used in the inversion of each tile (dependent on noise and disk position etc.), which directly feed into the inversion results.

In this work a Multi-Layer Perceptron (MLP) neural network (e.g., [Haykin 1998](#)) is used (see Fig. 4 for example). This class of ANN is known as a feed-forward ANN, where each layer consists of multiple neurons (or activation functions) acting as fundamental computation units. Connectivity is unidirectional from neurons in one layer to neurons in the next layer such that the outputs of a neuron in a layer serve as input to neurons in the following layer. The degree to which each neuron is activated is specified by the weight of the neuron.

The MLP uses supervised learning in order to determine the correct weights for each neuron. This algorithm proceeds in two phases: forward and backward propagation. The network is initialized with random values for the weights. The forward propagation then runs the input through the network, generating outputs based on the initial layer weights. In the backward propagation (BP), the errors between the ANN outputs and actual values (flows) are computed. Using this error, the weights of each activation function are then updated (through stochastic gradient descent) in order to minimize the output errors. The BP algorithm is performed in mini-batches (200 samples) of the total



**Fig. 2.** Panel *a*: coefficients computed using cross-correlation analysis for  $u_x$  (left panel) and  $u_y$  (right panel) to reduce the 152 unique modes to a single feature for each depth and flow component. Here we show the coefficients for two depths, 20.88 Mm (top panel) and 0.696 Mm (bottom panel). The CCA computes the ideal weights for the original data, given correlations with the output flows at a single depth. The phase speed corresponding to lower turning points at the target depth is also shown (green line). Modes close to this phase speed are given greater weight by the CCA. Panel *b*: averaging kernel built using the inversion pipeline (orange dashed) for a single tile, and an equivalent kernel built using the CCA coefficients derived using all tiles (blue line).



**Fig. 3.** Preprocessing pipeline proposed by this study for preparing the mode fit parameters for the machine learning. The pipeline must be performed for both  $U_x$  and  $U_y$ , while the CCA must be performed for each depth. The output  $\tilde{X}(z_t)$  is then the input for the ANN (Fig. 4).

dataset, with several passes over the entire set until convergence is achieved. Unlike classic stochastic gradient descent which updates the weights after every sample pass through the ANN, mini-batches settle on the minimum better because they are less subjected to noise. On average, each iteration will improve the weights, minimizing the difference between the predicted output and pipeline output.

The specific algorithm for the forward and backward propagation is as follows. For ring-diagram tile (sample)  $k$  in mini-batch  $n$ ,  $y_{j,k}^l(n)$  is the output of neuron  $j$  in the layer  $l$  (which consists of  $m_l$  neurons) and  $w_{ji}^l(n) \in \mathbb{R}$  is the weight applied to the output of neuron  $i$  in the previous layer  $l - 1$  fed into neuron  $j$  in layer  $l$ . The previous layer ( $l - 1$ ) consists of  $m_{l-1}$  neurons. In the forward propagation, the weights are fixed and the outputs are computed on a neuron-by-neuron basis:

$$y_{j,k}^l(n) = \varphi(v_{j,k}^l(n)), \quad (7)$$

where

$$v_{j,k}^l(n) := b_j^l(n) + \sum_{i=1}^{m_{l-1}} w_{ji}^l(n) y_{i,k}^{l-1}(n), \quad (8)$$

and the function  $\varphi$  refers to the activation function and  $b_j^l(n)$  is the bias. In this work, the activation function in the hidden layers

is the Rectifier Linear Unit (ReLU) function while the identity function is used for the output layer

$$\varphi(v_{j,k}^l(n)) = \begin{cases} \max(0, v_{j,k}^l(n)) & l = \text{hidden}, \\ v_{j,k}^l(n) & l = \text{output}. \end{cases} \quad (9)$$

Our choice in the ReLU function for the hidden layers is due to its improved convergence over other activation functions and the fact that it suffers less from the vanishing gradient problem (Glorot et al. 2011).

For tile  $k$ , the output of the network is denoted by

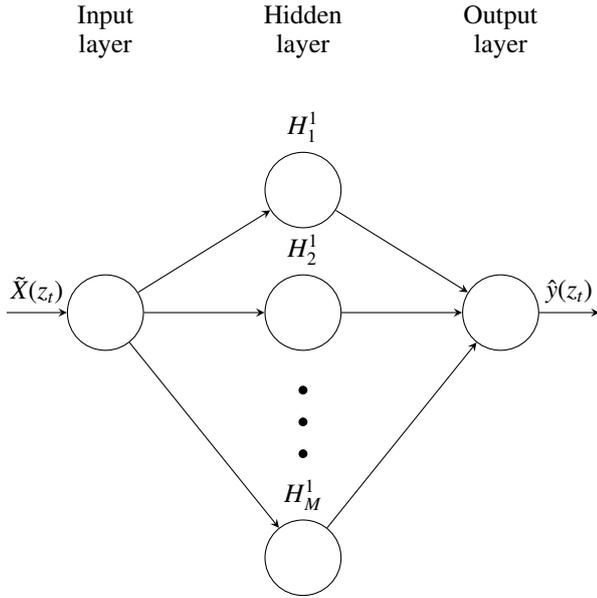
$$\hat{y}_k(n) = y_{1,k}^{\text{output}}(n), \quad (10)$$

where for this study we have a different network for each depth (due to CCA, Sect. 2), and hence only one neuron in the output layer.

In order for the ANN to compute precise solutions, the weights need to be updated iteratively such that the following cost function is minimized.

$$J(w(n), b(n)) = \frac{1}{2} \sum_{k=1}^{N_{\text{tile}}} (y_k - \hat{y}_k(n))^2, \quad (11)$$

where  $y_k$  is the data from the pipeline for tile  $k$  (and for a given depth) and  $\hat{y}_k(n)$  is the output of the ANN for mini-batch  $n$ . For



**Fig. 4.** General structure of the Multi-layer Perceptron. The input layer consists of 1 passive node, which is the output of the preprocessing pipeline, which relays the values directly to the hidden layer. The hidden layers  $H^l$  consist of  $M$  nonlinear active neurons that modify their input values and produce a signal to be fed into the next layer. The output layer then transforms its input values (from the hidden layer) into the flow values at target height  $z_t$ .

the first iteration (mini-batch), the weights are chosen randomly. The weights are then updated through back-propagation to minimize  $J$ .

In back-propagation, the layer weight  $w_{ji}^l(n)$  is adjusted on a layer-by-layer basis, from the output layer to the first hidden layer. These updates occur for each iteration (mini-batch)  $n$ , for several passes through all the training data. Each mini-batch consists a number  $K(n)$  of tiles. In this work the update of the weights and biases is achieved through stochastic gradient descent:

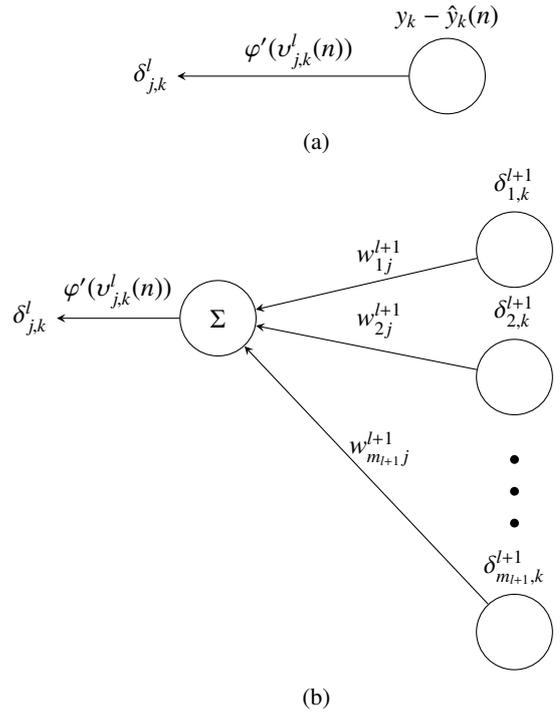
$$\begin{aligned} w_{ji}^l(n+1) &= w_{ji}^l(n) - \eta \frac{\partial J}{\partial w_{ji}^l(n)}, \\ b_j^l(n+1) &= b_j^l(n) - \eta \frac{\partial J}{\partial b_j^l(n)}, \end{aligned} \quad (12)$$

where  $\eta$  is the learning rate which governs how much the weights are changed at each iteration with respect to the cost function. Here the partial derivatives, or intuitively the response of the cost function to changes in a specific weight or bias, are computed through

$$\begin{aligned} \frac{\partial J}{\partial w_{ji}^l(n)} &= \frac{1}{K(n)} \sum_{k=1}^{K(n)} \delta_{j,k}^l(n) y_{i,k}^{l-1}(n), \\ \frac{\partial J}{\partial b_j^l(n)} &= \frac{1}{K(n)} \sum_{k=1}^{K(n)} \delta_{j,k}^l(n), \end{aligned} \quad (13)$$

where  $\delta_{j,k}^l(n)$  is the error of neuron  $j$  in layer  $l$  for tile  $k$  in mini-batch  $n$ . In order to determine  $\delta_{j,k}^l(n)$ , one has to know the error of the proceeding neurons. Hence in order to determine all  $\delta$ , the error of the output neuron is computed first:

$$\delta_{j,k}^{\text{output}}(n) = y_k - \hat{y}_k(n). \quad (14)$$



**Fig. 5.** Schematics of the Back-Propagation process, where the error of each neuron  $\delta$  is computed first from the error in the output layer (*panel a*), through all the hidden layers (*panel b*). The weights and bias are updated accordingly. *Panel a*: schematic of how the error is computed for the output neuron (Eq. (14)). We note that in this study we use the identity function for the activation function of the output,  $\varphi'(v_j^l(n)) = 1$ . *Panel b*: schematic of how the error of neuron  $j$  in hidden layer  $l$  is computed from the error of the  $m_{l+1}$  neurons in layer  $l+1$  (Eq. (15)).

The errors of the  $m_l$  neurons in layer  $l$  are then computed from those  $m_{l+1}$  neurons in layer  $l+1$ , through

$$\delta_{j,k}^l(n) = \varphi'(v_{j,k}^l(n)) \sum_{q=1}^{m_{l+1}} \delta_{q,k}^{l+1}(n) w_{jq}^{l+1}(n), \quad (15)$$

where  $\varphi'$  is the derivative of the activation function.

Figure 5 shows a diagram of the Back-Propagation process in the both the hidden and output layers.

After updating the layer weights in the backward propagation, the next mini-batch is used in forward- and back propagation to further minimize the cost function. This is repeated (numerous times through the whole data set) until the maximum allowed number of iterations is reached, or an early stopping criterion is met. The convergence rate of the ANN weights is governed by the learning rate  $\eta$ , which must be chosen such that the weights converge in a reasonable number of iterations while still finding the global minimum of the cost function. Typically, the determination of  $\eta$  is done experimentally, by slowly increasing  $\eta$  until the loss starts to increase. For this study we find (through a grid search)  $\eta = 0.001$  to be a reasonable learning rate.

In practice a regularization term  $\alpha \sum w^2$  is included in Eq. (11) to penalize complex models that may result in overfitting. Here we have set  $\alpha$  to 0.001.

### 3. Performance

In the previous section, the ANN predicts flows from the mode shifts given by the ring-diagram pipeline. Here, the performance

of the proposed method is shown and compared to alternative approaches. In this study we use the software packages Python 3.5.4 and *scikit-learn* 0.19.1 (Pedregosa et al. 2011), which are freely available.

### 3.1. Experimental metrics and setup

The evaluation metric for this problem will be the root mean square error (RMSE),

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{tile}} - 1} \sum_{k=1}^{N_{\text{tile}}} (y_k - \hat{y}_k)^2}, \quad (16)$$

where  $y_k$  is the flow for tile  $k$  from the ring-diagram pipeline and  $\hat{y}_k$  the predicted flow from the machine learning. For this study the RMSE of each depth will be computed and compared to the mean of the pipeline inversion error. The success of an estimator is then measured by comparing the size of the error of the prediction to that of the inverted flow.

In order to verify that the proposed method (preprocessing and ANN) has the ideal performance when compared to other existing methods, an additional evaluation metric is introduced; the computational time (CPU time). As stated previously, the goal of this study is to improve upon the current pipeline, primarily in reducing the computational burden. As such, each step in preprocessing and the chosen ANN architecture will be compared to other methods/architectures in order to demonstrate both the computational burden and accuracy of the proposed method. We note that a large selection of methods exist and a comparison of them all is beyond the scope of this paper. Here we compare to the most common methods.

To avoid overfitting when training supervised methods, the dataset is split randomly into training and testing subsets, in a manner known as  $k$ -fold cross-validation (e.g., Mosteller et al. 1968).  $k$ -fold cross validation splits the data into  $k$  sets, each roughly equal in size. The training is then used on  $k - 1$  subsets and testing (using an evaluation metric) is performed on the remaining subset. This process is applied  $k$  times, shifting the testing segment through all of the segments. In doing so the entire data set is used for training and, in turn, prediction. With each sample in the entire set being used for validation at one time through this process, we can then measure the performance metric of the prediction by averaging the entire set. For this study we use ten-fold cross-validation on the flow data to obtain a complete set of predicted values. These predicted values are subsequently compared with the actual values to obtain the evaluation metric (RMSE) mentioned above.

### 3.2. Experimental analysis

In this section we present the results of each step in the proposed method, using the aforementioned evaluation metrics (computational time and RMSE). For clarity and brevity, we show the comparison results only for a depth of 10.44 Mm. However, the results are consistent with other depths. In order to assess each step in preprocessing, a basic ANN architecture is chosen. The network consists of one layer with ten neurons. In order to assess each step in the preprocessing, the proposed method of Sect. 2 is used with only the chosen step being varied. The CPU time and rms error of the machine learning is determined from the application of ten-fold cross-validation upon the entire data set (709 734 tiles).

**Table 1.** Comparison between methods for the completion of missing data.

Method	CPU Time (s)		Flow RMSE ( $\text{m s}^{-1}$ )	
	$u_x$	$u_y$	$u_x$	$u_y$
<b>Mean</b>	<b>2</b>	<b>2</b>	<b>8.6</b>	<b>7.3</b>
Median	28	63	8.6	7.3
Most-Frequent	31 860	87 963	8.9	7.4

**Notes.** The bold row is the proposed method.

**Table 2.** Comparison of performance for different methods of normalization.

Method	CPU Time (s)		Flow RMSE ( $\text{m s}^{-1}$ )	
	$u_x$	$u_y$	$u_x$	$u_y$
MM-01	3	3	8.6	7.3
MM-11	3	4	8.6	7.3
MA	5	3	8.6	7.3
<b>SS</b>	<b>4</b>	<b>4</b>	<b>8.6</b>	<b>7.3</b>

**Notes.** The bold row is the proposed method.

**Table 3.** Comparison of the performance of different feature selection/reduction methods.

Method	CPU Time (s)		Flow RMSE ( $\text{m s}^{-1}$ )	
	$u_x$	$u_y$	$u_x$	$u_y$
Kbest	6	3	13.41	11.05
ET	44	44	13.85	8.58
DT	234	242	13.94	8.26
<b>CCA-1</b>	<b>14</b>	<b>12</b>	<b>8.61</b>	<b>7.29</b>
CCA-2	14	14	8.61	7.29
PLS	9	11	13.06	10.55

**Notes.** The bold row is the proposed method.

#### 3.2.1. Data imputation

Table 1 compares the different methods used to replace the missing data in the mode-fit parameters. In terms of computational gain, it is clear that the mean completion is ideal (2 CPU seconds).

#### 3.2.2. Normalization

Table 2 shows the performance of four different normalization methods, namely, the feature scaling e.g., Minimum-Maximum scaled from 0 to 1 (MM-01) or  $-1$  to 1 (MM-11), Maximum Absolute (MA), and the standardization scaling (SS). The computational burden for each step is rather small, with little difference between them. The same is also true for the RMSE, showing that the choice in normalization is arbitrary for the proposed method.

#### 3.2.3. Feature selection/reduction

Table 3 shows the results of applying different feature-reduction methods. The proposed CCA reduction is compared with different feature selection/reduction methods: selecting  $K$  best features using  $f$ -score (Hand et al. 2001), applying tree-based methods such as Decision Trees (Hand et al. 2001;

**Table 4.** Comparison between supervised machine-learning methods with the same preprocessing.

ML	Training and prediction time		RMSE ( $\text{m s}^{-1}$ )	
	$u_x$	$u_y$	$u_x$	$u_y$
Lin	<1	<1	8.8	7.3
Bay	<1	<1	8.8	7.3
DT	19	20	8.8	7.4
ET	2	2	8.7	7.3
RF	39	42	10.5	9.0
GTB	274	154	8.6	7.3
KNN	50	35	9.0	7.7
<b>NN</b>	<b>170</b>	<b>169</b>	<b>8.6</b>	<b>7.3</b>
SVR	61 441	47 555	8.5	7.3

**Notes.** The bold row is the proposed method.

Rokach & Maimon 2014) or Ensemble Trees (Geurts et al. 2006), and Partial Least Squares (Hastie et al. 2001). The computational times show that our chosen method (CCA) is not the fastest, but when comparing the RMSE it outperforms the other feature methods. Interestingly, using only a one-component vector achieves good accuracy, and increasing to two component vectors does not improve the result.

### 3.2.4. Neural networks

The results thus far have focused on the preprocessing steps of the proposed method. We now focus on the performance of a number of machine-learning techniques and network architectures and compare them to the neural network proposed in this study.

Table 4 compares different supervised machine-learning methods after applying data completion, normalization, and feature reduction. The methods examined are Linear Regression, Bayesian Regression (Hastie et al. 2001; Bishop 2006), Decision Tree (Hand et al. 2001; Rokach & Maimon 2014), Ensemble Tree (Geurts et al. 2006), Random Forest (Breiman 2001), Gradient Tree Boosting (Friedman 2000), K-Nearest Neighbor (Hastie et al. 2001), and Support Vector Regression (Bishop 2006; Haykin 2009). The computation time for each method scales with the complexity of the model from the Linear model (<1 s) to Support Vector Regression (~50 000 s). While the proposed ANN takes 200–400 s for training and predicting, this is still significantly small compared to the current burden of the pipeline. A comparison of the accuracy shows that the ANN presented in this paper is among the methods showing the best performance, with an RMSE of  $8.6 \text{ m s}^{-1}$  for  $u_x$  and  $7.3 \text{ m s}^{-1}$  for  $u_y$ .

The above results show that the ANN performs best for the ring-diagram inversions. However, ideal results depend upon the architecture of the ANN, specifically, what number of layers and neurons gives the best results for the ANN. Table 5 shows the performance of the ANN with different numbers of hidden layers. The results show that best performance is obtained with just one hidden layer. The addition of extra layers increases computational burden due to the increase in the complexity of the model. In terms of how many neurons are required per layer, we have found through experimentation that the RMSE does not improve with more than 10 units.

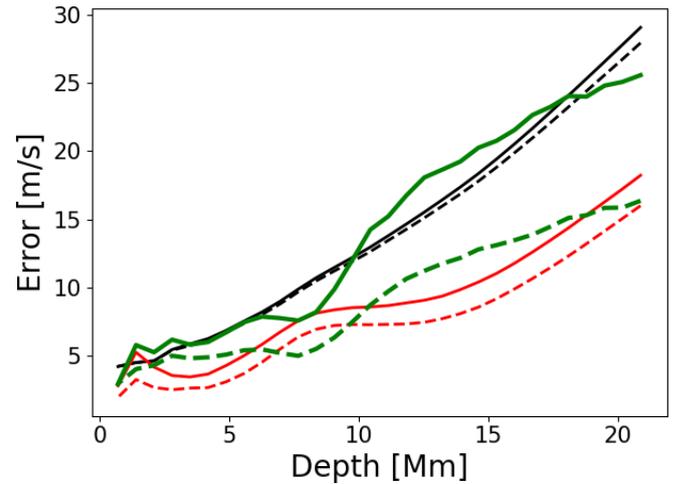
### 3.2.5. RMSE of the model versus depth

The results thus far have been shown for one depth (10.44 Mm); Fig. 6 shows the differences between mean inversion error and

**Table 5.** Comparison of ANN performance for different numbers of hidden layers, with 10 neurons in each layer.

No. of layers	CPU Time (s)		Flow RMSE ( $\text{m s}^{-1}$ )	
	$u_x$	$u_y$	$u_x$	$u_y$
<b>1</b>	<b>170</b>	<b>169</b>	<b>8.59</b>	<b>7.29</b>
2	360	194	8.59	7.29
3	342	330	8.59	7.30
4	427	439	8.61	7.30
5	3099	1695	8.62	7.30

**Notes.** The bold row is the proposed method.

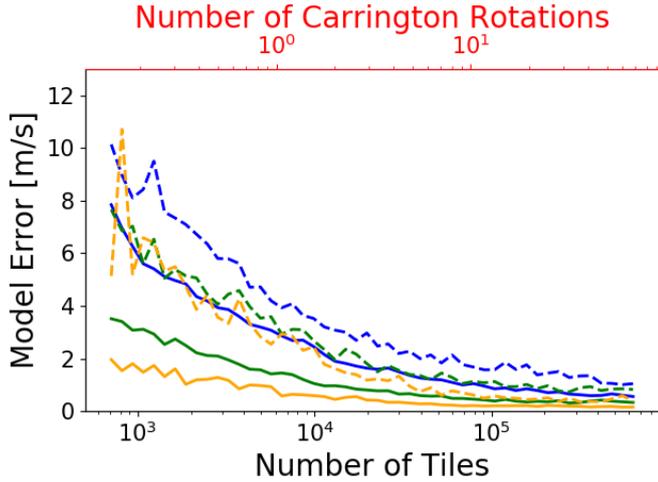


**Fig. 6.** Comparison of the pipeline inversion error (black lines) and machine-learning error (RMSE, red lines), for  $u_x$  (solid) and  $u_y$  (dashed) for all depths below the surface. The green lines are the standard deviation of the prediction values after noise realizations are added to the mode fits.

the RMSE of the ANN for all depths below the photosphere. We have ignored the results at  $z_r = 0 \text{ Mm}$  (photosphere) due to inconsistency in the inverted flows in the HMI pipeline. The results of Fig. 6 show that the proposed ANN of this study achieves a RMSE that is generally below the inversion errors reported in the pipeline. While the errors are not directly comparable, the results provide confidence in the results of the machine learning. Additionally, while the errors reported in the pipeline are similar for  $u_x$  and  $u_y$ , there is a difference in the machine-learning errors. This is due to errors in the machine learning resulting from different variances in  $u_x$  and  $u_y$  with the former having larger variance than the latter. This variance leads to a more difficult fit for the model, and thus higher error. Additionally, the machine learning may have difficulties fitting  $u_x$  due to systematics in the patch tracking in the  $x$  direction.

### 3.2.6. Effect of realization noise

So far, we have used only the mode fits in predicting the flow values. However, the determination of these mode fits is not exact and hence the effect of mode-fit error on the machine-learning model must be considered. In order to examine the propagation of errors we take CR 2107-2201 and build the ANN described previously. The mode fits of the remaining ten rotations are then perturbed by a realization of the errors. Predictions are then made with the new noisy data. This process is repeated 1000



**Fig. 7.** Model error (RMSE) between an ANN trained with the complete 105 CRs and one trained with a subset (using  $k$ -fold cross-validation). The errors converge for both  $u_x$  (solid) and  $u_y$  (dashed) as the subsample size approaches the full sample size. Here we only show the results for depths of 20.88 Mm (blue), 10.44 Mm (green), and 0.696 Mm (orange).

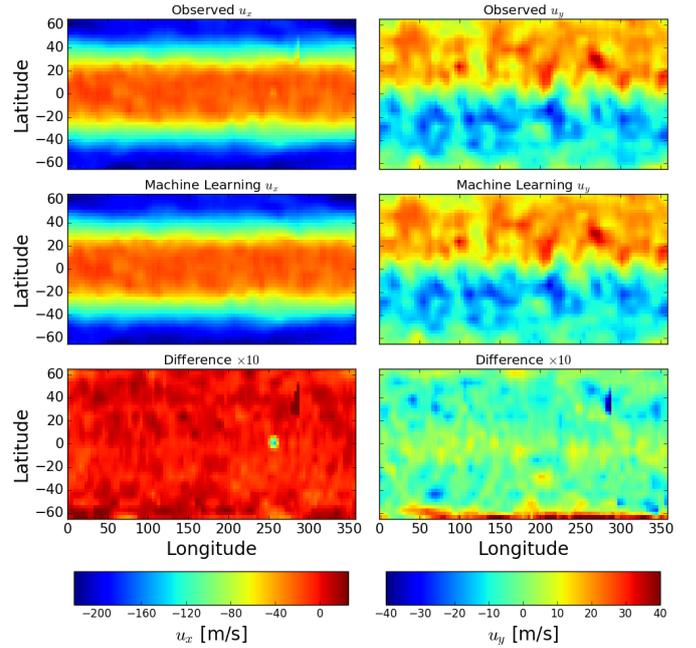
times and the deviation is computed. Figure 6 shows the averaged deviation as a function of depth given the noisy data. For  $u_x$  the errors are of the order of the inversion pipeline, while the  $u_y$  are less. This result is not unexpected. The ANN was trained to find a particular relationship between the mode fits and the flows across the disk. By adding errors to the data, these errors propagate through the model, producing a deviation in the predictions higher than the RMSE of the model. The fact that the deviations are not significantly greater than those reported in the pipeline is a good indicator of the robustness of the model to errors.

### 3.2.7. RMSE of the model versus number of samples

We conclude this section by addressing the question of how many samples are needed for an accurate ANN. In the field of machine learning, the answer to the common question of how to get a better model is: more data. Thus, we compute the model error (RMSE) between a model that is trained with the complete 105 CRs, and those trained with only a subset. Again we use ten-fold cross-validation in prediction. Figure 7 shows the convergence of models trained with increasing subset size to that trained with the full data set. The results show that with just a small subset of around 1–10 CRs ( $\sim 6000$ – $60\,000$  Tiles) the difference in model error between the predictions of an ANN trained with the 105 CRs and those of an ANN trained with a subset is below  $4\text{ m s}^{-1}$  for the three depths examined. The results also show that  $u_y$  converge slower than  $u_x$ . Increasing the sample size used in training slowly improves the model with each additional tile.

## 4. Case study: equatorial Rossby waves

In fitting any model to a vast amount of data, there is a possibility that the subtle helioseismic features in each tile are removed or altered. Figure 8 shows the pipeline and machine-learning flows for both components of the flow, and the difference between them. Close examination shows that while the general structure is nearly identical, some small features present in the pipeline

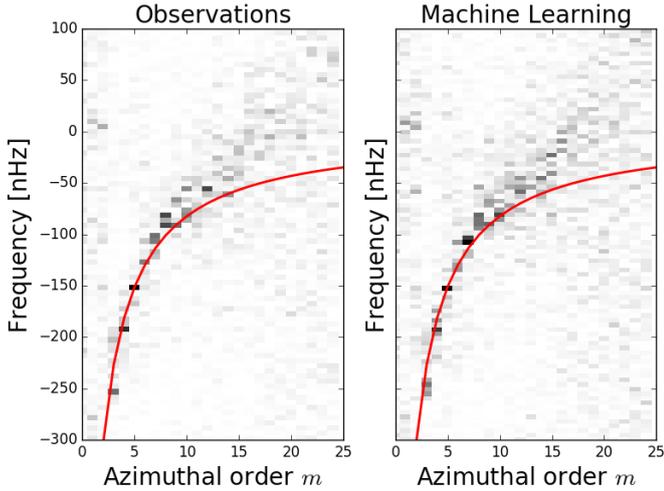


**Fig. 8.** Comparison of the flow maps for  $u_x$  (left panel) and  $u_y$  (right panel) between the pipeline (top panel) and machine learning (middle panel), and the difference between them (bottom panel, scaled by factor 10) at a target depth of 0.696 Mm. The maps are generated from a time average over the CR 2100. The predicted values were obtained from an ANN trained using CR 2107–2201.

flows are not present in the machine learning (e.g., at longitude and latitude  $280^\circ$  and  $40^\circ$ , respectively). While these features appear as artifacts to the keen observer, they raise the question of whether the machine-learning model (in an effort to fit a model) overlooks helioseismic signatures seen after averaging over long timescales. To explore this possibility, we examine equatorial Rossby waves in the same manner as Löptien et al. (2018), but with the predicted flows.

Löptien et al. (2018) recently reported the unambiguous detection of retrograde-propagating vorticity waves in the near-surface layers of the Sun. These waves exhibit the dispersion relation of sectoral Rossby waves. Solar Rossby waves have long periods of several months with amplitudes of a few meters per second, making them difficult to detect without averaging large amounts of data. To detect these Rossby waves in both the raw data and the machine-learning data, we follow the technique of Löptien et al. (2018), specifically:

1. Flow tiles ( $u_x, u_y$ ) are sorted into cubes of Latitude, Stonyhurst Longitude and time.
  2. The one-year frequency signal (B-angle) is removed.
  3. Missing data on the disk are interpolated in both time and latitude.
  4. Data exceeding a distance of  $65^\circ$  from disk center are neglected.
  5. The data are remapped into a reference frame that rotates at the equatorial rotation rate ( $453.1\text{ nHz}$ ) and are then transformed back to a Carrington longitude grid.
  6. The longitude mean is subtracted.
  7. The vorticity is computed.
  8. Spherical harmonic transforms (with  $m = \ell$ ) and temporal Fourier transforms are applied to produce a power spectrum.
- We apply this procedure to the flow maps at depths of 0.696 Mm and 20.88 Mm.



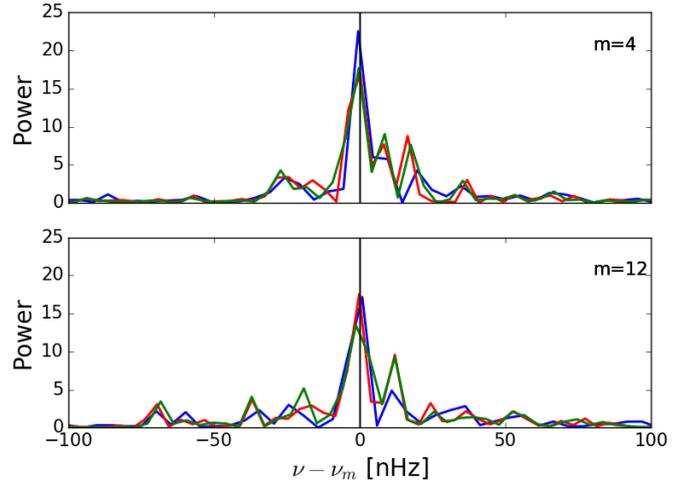
**Fig. 9.** Comparison of the Rossby wave power spectrum computed from  $\sim 6$  years (CR 2089-2201) of pipeline data (*left panel*) and from the machine learning (*right panel*) for a depth of 0.696 Mm. The machine learning method was trained using 10 Carrington rotations (CR 2079-2088). The red line is the dispersion relation  $\omega = -2\Omega_{\text{ref}}/(m + 1)$  of sectoral Rossby waves, measured in a frame rotating at the equatorial rotation rate  $\Omega_{\text{ref}} = 453.1$  nHz.

In order to examine if Rossby waves are still present in the machine learning we take the results from the ANN model outlined in the previous sections for CR 2097-2181. The Rossby wave procedure outlined above was then followed using these new maps. Figure 9 compares the Rossby wave power spectrum from the pipeline flows and the machine learning, computed at a depth of 0.696 Mm. Upon visual inspection they appear to be very similar, validating the ability of the machine-learning method to recover the presence of Rossby waves. Additionally, Fig. 10 shows slices of the power spectrum for different azimuthal order  $m$ , for the proposed method trained with 1, 10, and 20 CRs (from the unused CR 2182-2201). The results show that using as small a sample as 1 CR ( $\sim 6800$  tiles) to train the machine-learning model can produce a model that captures the Rossby wave power spectrum reasonably well.

## 5. Conclusion

Local helioseismology has provided vast amounts of raw observed data for the Sun, but despite 50 years of observations and analysis, we still have no consistent and complete picture of its internal structure. The computational field of machine learning and artificial intelligence has grown in both usage and capability in the last few decades, and has shown promise in other fields in ways that could be extended to local helioseismology.

In this study we have shown that machine learning provides an alternative to computationally expensive methodologies. We have also shown that in using the data of 8 years of HMI observations, we can use an ANN model to predict future flow data with an RMSE that is well below that of the observations, while maintaining the flow components of interest to local helioseismology. Additionally, we find that the propagation of noise realization results in a deviation of the flow values of the order of the pipeline errors. The computational burden was previously 31 CPU hours for 1 CR of data. With a trained ANN the computational cost is now  $10^{-3}$  CPU hours. While we have focused our efforts on obtaining an accurate ANN model, the results of Sect. 3 show that any number of common architectures



**Fig. 10.** Comparison of sectoral power spectra at a depth of 20.88 Mm for the Rossby waves with azimuthal orders  $m = 4$  and  $m = 12$ . The results include the HMI observations (blue) and machine learning trained using 1 (CR 2079) and 10 (CR 2079-2088) CRs (red and green, respectively). For each  $m$ , the power spectrum frequency ( $\nu$ ) is centered on the Rossby wave frequencies ( $\nu_m$ ) reported by Lötptien et al. (2018).

or preprocessing can obtain a reasonable model for future predictions. Nevertheless, nonlinear models (such as the proposed ANN here) can capture some of the nonlinearity (e.g., noise or missing modes) that occurs between all tiles across the disk.

Here we have only shown the computational efficiency gain achieved by the application of machine learning, but future improvements can be made. The method presented here is purely data driven, without introducing a priori constraints. Recent studies (e.g., Raissi et al. 2017a,b) have shown that physics informed neural networks can be built that are capable of enforcing physical laws (e.g., mass conservation) when determining the machine-learning model. While the constraint of physical laws is beyond the scope of the work here, these studies demonstrate the potential of machine learning in determining subsurface solar structures for which we have prior knowledge of constraints. Additionally, the use of synthetic ring diagrams computed using computational methods with machine learning could improve current capabilities of the pipeline in probing solar subsurface flows. Therefore, the application of machine learning and deep learning techniques present a step forward for local helioseismic studies.

*Acknowledgements.* This work was supported by NYUAD Institute Grant G1502 “NYUAD Center for Space Science”. RA acknowledges support from the NYUAD Kawadar Research Program. The HMI data is courtesy of NASA/SDO and the HMI Science Team. Computational resources were provided by the German Data Center for SDO funded by the German Aerospace Center. SMH and LG thank the Max Planck partner group program for supporting this work. Author contributions: CSH and LG designed the research goals. RA proposed the machine-learning methodology for predicting flows and performed experimental and computational analysis. CSH prepared data and performed helioseismic analysis. All authors contributed to the final manuscript. We thank Aaron Birch and Bastian Proxauf for helpful insight into the ring-diagram pipeline.

## References

- Alpaydin, E. 2010, *Introduction to Machine Learning, 2nd edn.* (Cambridge: The MIT Press)
- Antia, H. M., & Basu, S. 2007, *Astron. Nachr.*, **328**, 257
- Asensio Ramos, A., Requerey, I. S., & Vitas, N. 2017, *A&A*, **604**, A11
- Bishop, C. M. 2006, *Pattern Recognition and Machine Learning (Information Science and Statistics)* (Berlin, Heidelberg: Springer-Verlag)

- Bogart, R. S., Baldner, C., Basu, S., Haber, D. A., & Rabello-Soares, M. C. 2011a, *J. Phys. Conf. Ser.*, **271**, 012008
- Bogart, R. S., Baldner, C., Basu, S., Haber, D. A., & Rabello-Soares, M. C. 2011b, *J. Phys. Conf. Ser.*, **271**, 012009
- Breiman, L. 2001, *Mach. Learn.*, **45**, 5
- Friedman, J. H. 2000, *Ann. Stat.*, **29**, 1189
- Geurts, P., Ernst, D., & Wehenkel, L. 2006, *Mach. Learn.*, **63**, 3
- Giles, P. M., Duvall, T. L., Scherrer, P. H., & Bogart, R. S. 1997, *Nature*, **390**, 52
- Gizon, L., & Birch, A. C. 2005, *Liv. Rev. Sol. Phys.*, **2**, 6
- Glorot, X., Bordes, A., & Bengio, Y. 2011, in Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, eds. G. Gordon, D. Dunson, & M. Dudík (Fort Lauderdale, FL, USA: PMLR), *Proc. Mach. Learn. Res.*, **15**, 315
- Hand, D. J., Smyth, P., & Mannila, H. 2001, *Principles of Data Mining* (Cambridge: MIT Press)
- Härdle, W., & Simar, L. 2007, *Applied Multivariate Statistical Analysis*, 2nd edn. (Berlin, Heidelberg: Springer)
- Hardoon, D. R., Szedmak, S., & Shawe-Taylor, J. 2004, *Neur. Comput.*, **16**, 2639
- Hastie, T., Tibshirani, R., & Friedman, J. 2001, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer Series in Statistics (New York: Springer, New York Inc.)
- Haykin, S. 1998, *Neural Networks: A Comprehensive Foundation*, 2nd edn. (Upper Saddle River, NJ, USA: Prentice Hall PTR)
- Haykin, S. S. 2009, *Neural Networks and Learning Machines* (London: Pearson Education)
- Hill, F. 1988, *ApJ*, **333**, 996
- Hotelling, H. 1936, *Biometrika*, **28**, 321
- Juszczak, P., Tax, D. M. J., & Duin, R. P. W. 2002, *Proceedings of the 8th Annu. Conf. Adv. School Comput. Imaging*, 25
- Little, R., & Rubin, D. 1987, *Statistical Analysis With Missing Data*, Wiley Series in Probability and Statistics – Applied Probability and Statistics Section Series (New York: Wiley)
- Löptien, B., Gizon, L., Birch, A., et al. 2018, *Nat. Astron.*, **2**, 568
- Mosteller, F., & Tukey, J. W. 1968, in *Handbook of Social Psychology*, eds. G. Lindzey, & E. Aronson (Boston: Addison-Wesley), 2
- Pearson, K. A., Palafox, L., & Griffith, C. A. 2018, *MNRAS*, **474**, 478
- Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, *J. Mach. Learn. Res.*, **12**, 2825
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. 2017a, ArXiv e-prints [arXiv:1711.10561]
- Raissi, M., Perdikaris, P., & Karniadakis, G. E. 2017b, ArXiv e-prints [arXiv:1711.10566]
- Rokach, L., & Maimon, O. 2014, *Data Mining With Decision Trees: Theory and Applications*, 2nd edn. (Singapore: World Scientific Publishing Co., Inc.)
- Schou, J., & Bogart, R. S. 1998, *ApJ*, **504**, L131
- Schou, J., Scherrer, P. H., Bush, R. I., et al. 2012, *Sol. Phys.*, **275**, 229