Volume 6, Issue 4, April 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

# Deep Learning Approach for Classifying Galaxies

**Remya G, Anuraj Mohan**
Assistant Professor, Dept of CSE, NSS College of Engineering, Palakkad,
Kerala, India

***Abstract— Structural analysis of galaxy images is useful for studying galaxy formation and evolution - for example, we can observe that stars in an elliptical galaxy tend to be older than the stars in spiral galaxies. Our universe consists of billions of galaxies and studying the properties of galaxies will allow us to obtain a deeper understanding of physics of the universe that we live in. Surveys conducted by Sloan Digital Sky Survey (SDSS) have provided us with millions of images, which has allowed researchers to conduct analysis on galaxy morphology. Morphological analysis was usually carried out by trained experts, which takes a lot of time and is impractical when large numbers of images are involved. Previous efforts to build machine learning based systems were not able to achieve the required levels of precision. Instead, the Galaxy Zoo project applied a crowd sourcing strategy to classification, using online volunteers to classify the images by answering a series of questions. However, even this approach does not scale properly to keep up with the increasing size of datasets. To deal with these problems, an automated approach became necessary. Thanks to the availability of training set from crowd sourcing, it has become feasible to train machine learning models for this task effectively. In this paper, we use a deep neural network model for galaxy structure classification, which exploits translational and rotational symmetry in the images.***

***Keywords— Deep Learning, Neural Network, Classification***

## I. INTRODUCTION

Structural characteristics of galaxies is an important area of interest in the large scale study of the Universe. Galaxy classification is the first of the many steps towards a greater understanding of the origins and the formative processes of galaxies, and by extension the evolution processes of the Universe. Galaxy classification is important for two main reasons - First, to produce extensive catalogues for statistical and astronomical programs, and second foriidiscovering underlying physics.

In the recent years, with numerous digital sky surveys across a wide range of wavelengths, astronomy has become an immensely data-intensive field. For example, the Large Synoptic Survey Telescope (LSST) is a planned wide-field "survey" reflecting telescope that will take pictures of the entire available sky every few nights and will produce more than 200,000 images of galaxies every year which is impossible to be reviewed by humans. This overload creates a need for techniques to automate the difficult problem of classification. Many attempts to build Machine Learning based classification systems for galaxies have had difficulties in reaching precisions needed for astronomical studies [1].

The Galaxy Zoo project was born out of this necessity. The intended goal of the project was to obtain reliable classifications for over 900,000 galaxies by allowing members of the public - also called citizen scientists to contribute classifications using a web platform. The project was very successful, with the entire dataset being labelled within several months. This labelled dataset along with recent advancements in image classification [2], [3] is what allows us to use deep neural networks, since they require large amounts of training data to perform well. In this paper, we seek to use a deep neural network for galaxy structural classification that is specially crafted to the properties of images of galaxies.

The rest of this paper is structured as follows: in Section 2 we discuss the previous works in this area, in Section 3 we discuss the dataset. Section 4 explains the theory behind deep learning. Convolutional neural networks are described in Section 5, an overview of our modelling approach is discussed in Section 6, implementation in Section 7. Finally we discuss the conclusions and future work in Section 8.

## II. RELATED WORK

Machine learning techniques have been used in astronomy research for more than 20 years. Neural networks were first applied for differentiating between star and galaxy and classification of galaxies [4], [5]. Earlier work in this field had to operate on much smaller datasets and the networks had very few trainable parameters (between 100 and 1000). It's only recently that we had access to larger training sets using data from surveys such as the SDSS [6], [7]. Convolutional networks were first applied to solve galaxy classification problem in 2014 after the success of convnets in other image classification problems.

## III. DATASET

The dataset was obtained from Galaxy Zoo, which is an online crowdsourcingproject, where users are asked to describe the galaxy structure based on JPEG color images [8], [9]. Participants were asked to answer questions such as 'Doesthe galaxy have a mostly clumpy appearance?' and 'Is there any sign of a spiral armpattern?'

 

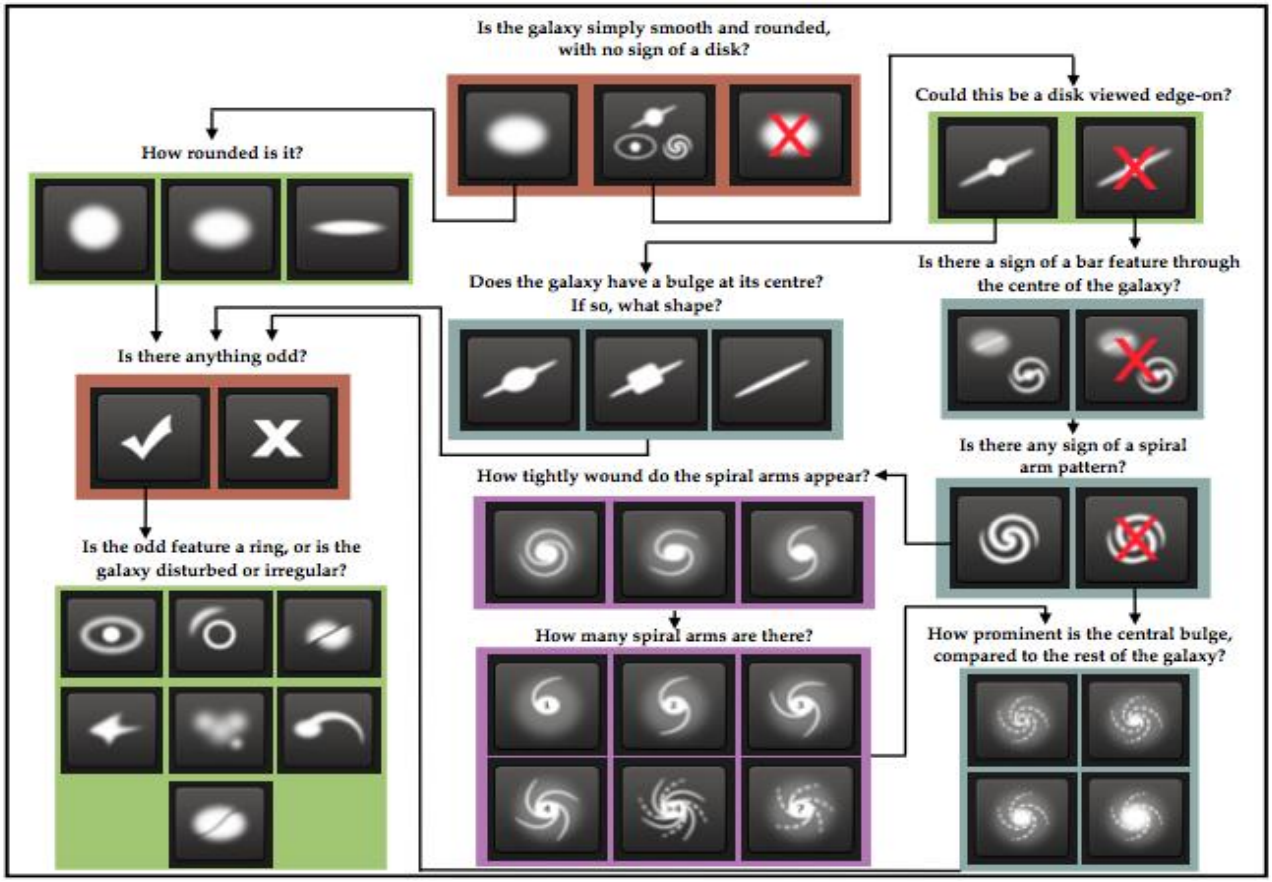


Figure 1. The Galaxy Zoo decision tree *(Courtesy: www.kaggle.com)*

The training set of data consisted of61,578 JPEG colour images of galaxies eachof which had thedimensions 424 by 424 pixels, along with probabilities for each of the 37 answers in the decision tree. The first column in each solution is labeled by a randomly generated IDcalled GalaxyID. This ID allows us to match the probability distributionswith the images. The next 37columns are all floating point numbers between 0 and 1 inclusive. Theserepresent the morphology (or shape) of the galaxy in 37 different categories as identified by crowdsourced volunteer classifications aspart of the Galaxy Zoo 2 project. These morphologies are related to probabilities for each category; a high number (close to 1) indicates that many users identified this morphology category for the galaxy with a high level of confidence. Low numbers for a category (close to 0) indicate the feature is likely not present.

A test set of 79,975 imageswas also provided, but with no labelling.Galaxy Zoo guides itscitizen scientists through a nested decision tree - this is what constitutes the classification process. Each galaxy's classification is the result of a specific path down a decision tree. Multiple individuals (typically 40-50) all classified the same galaxy, resulting in multiple paths along the decision tree. These multiple paths generate probabilities for each node. Volunteers begin with general questions and move on to more specific ones. As a result, at each node or question, the total initial probability of a classification will sum to 1.0.

## IV. DEEP LEARNING

The idea of deep learning or hierarchical learning is to build models that attempt to represent the data at multiple levels of abstraction, and which can discover accurate representations automatically from the data itself [11]. Deep learning models consist of several layers that form a hierarchy: each layer extracts a gradually more abstract representation of the input data and builds upon the image from the previous layer, which is done by computing a non-linear transformation of its input. The parameters of these transformations are optimized by training the model on a dataset.

A feed-forward neural network is an example of such a model, where each layer consists of a number of units which arecalled neurons that compute a weighted linear combination of the layer input, followed by an element wise non-linearity. These weights constitute the model parameters. Let the vector $x_{n-1}$ be the input to layer $n$, $W_n$ be a matrix of weights, and $b_n$ be a vector of biases. Then the output of layer $n$ can be represented as the vector.

$$\mathbf{x}_n = f(\mathbf{W}_n\mathbf{x}_{n-1} + \mathbf{b}_n), \qquad (2)$$

where f is the activation function, an element wise nonlinear function. Common choices for the activation function are linear rectification ($f(x) = \max(x, 0)$), which gives rise to rectified linear units [12], or a sigmoidal function ($f(x) = (1 + e^{-x})^{-1}$ or ($f(x) = (\tanh(x))$.Another possibility is to compute the maximum across several linear combinations of the input, which gives rise to maxout units [13]. We will consider a network with N layers. The network input is represented by $x_0$ , and its output by $x_N$. A schematic representation of a feed-forward neural network is shown in Figure 2.

 

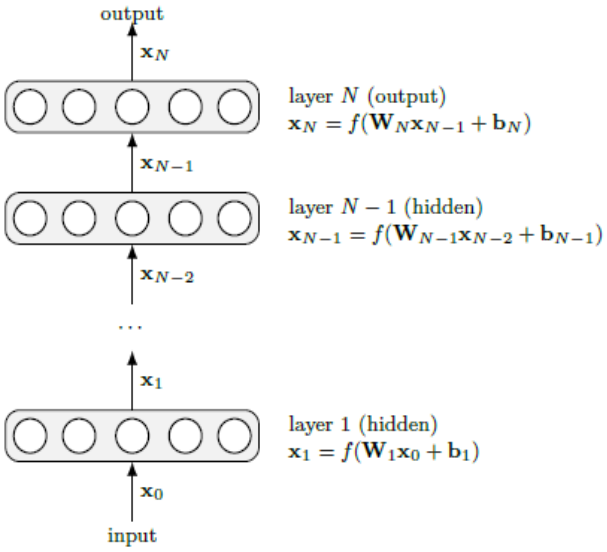


**Figure 2.** Schematic representation of a feed-forward neural network with $N$ layers.

The network computes a function of the input $x_0$. The output $x_N$ of this function is a prediction of one or more quantities of interest. We will use t to represent the desired output (target) corresponding to the network input $x_0$. The topmost layer of the network is referred to as the output layer. All the other layers below it are hidden layers.

During training, the parameters of all layers of the network are jointly optimized to make the output $x_N$ approximate the desired output t as closely as possible. We can quantify the prediction error using an error measure $e(x_N,t)$. As a result, the hidden layers will learn to produce representations of the input data that are useful for the task at hand, and the output layer will learn to predict the desired output from these representations.

$$\mathbf{W}_n \leftarrow \mathbf{W}_n - \eta\frac{\partial e(\mathbf{x}_N, \mathbf{t})}{\partial \mathbf{W}_n}, \qquad (3)$$

$$\mathbf{b}_n \leftarrow \mathbf{b}_n - \eta\frac{\partial e(\mathbf{x}_N, \mathbf{t})}{\partial \mathbf{b}_n}. \qquad (4)$$

To determine how the parameters should be changed to reduce the prediction error across the dataset, we can use gradient descent: the gradient of $e(x_N,t)$ is computed with respect to the model parameters $W_n$ , $b_n$ for n = 1…N. The parameter values of each layer are then modified by repeatedly taking small steps in the direction opposite to the gradient: Here, η is the learning rate, a hyper parameter controlling the step size. Traditionally, models with many non-linear layers of processing were not in common use, because they were difficult to train: gradient information would vanish as it propagated through the layers, making it difficult to learn the parameters of lower layers [15].

Practical applications of neural networks were limited to models with one or two hidden layers. Since 2006, the invention of several new techniques, along with a significant increase in available computing power, have made this task much more feasible. By replacing traditional activation functions with linear rectification, the vanishing gradient problem was significantly reduced. The introduction of Dropout [16] and DropConnect regularization [17] makes it possible to train larger networks with many more parameters.

## V. CONVOLUTIONAL NEURAL NETWORKS

Convolutional neural networks also knowns as CNN's or convnets are a special type of neural networks with constrained connectivity patterns between some of the layers. They are used when the input data exhibits some known grid-like topology, pixels of an image, or time series data. Convolutional neural networks consists of two types of layers: convolutional layers and pooling layers. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication. A convolutional layer takes a stack of feature maps (e.g. the colour channels of an image) as input,and convolves each of these with a set of learnable filters to produce a stack of output feature maps. This can be implemented efficiently by replacing the matrix-vector product $W_n x_{n-1}$ in Equation 2 with a sum of convolutions. We represent the input of layer n as a set of K matrices $X^k_{n-1}$with k = 1 ….K. Each of these matrices represents a different input feature map. The output feature maps $X^{(l)}_n$ with l = 1….L are represented as follows:

$$\mathbf{X}_n^{(l)} = f\left(\sum_{k=1}^{K} \mathbf{W}_n^{(k,l)} * \mathbf{X}_{n-1}^{(k)} + b_n^{(l)}\right). \qquad (5)$$

Here, * represents the two-dimensional convolution operation, the matrices $X^{(k,l)}_n$ represent the filters of layer n, and $b^l_n$ represents the bias for feature map l. Note that a feature map $X^{(l)}_n$ is obtained by computing a sum of K convolutions with the feature maps of the previous layer. The bias $b^{(l)}_n$ can optionally be replaced by a matrix $B^{(l)}_n$ , so that each spatial position in the feature map has its own bias (`untied' biases). This allows the sensitivity of the filters to vary across the

 

input. By replacing the matrix product with a sum of convolutions, the connectivity of the layer is effectively restricted to take advantage of the input structure and to reduce the number of parameters. Each unit is only connected to a local subset of the units in the layer below, and each unit is replicated across the entire input.

As a consequence of this restricted connectivity pattern, convolutional layers typically have far fewer parameters than traditional dense (or fully-connected) layers that compute a transformation of their input according to Equation 2. This reduction in parameters can drastically improve generalization performance (i.e., predictive performance on unseen examples) and make the model scale to larger input dimensionalities.

Because convolutional layers are only able to model local correlations in the input, the dimensionality of the feature maps is often reduced in between convolutional layers by inserting pooling layers. This allows higher layers to model correlations across a larger part of the input, albeit with a lower resolution. A pooling layer reduces the dimensionality of a feature map by computing some aggregation function (typically the maximum or the mean) across small local regions of the input [18]. This also makes the model invariant to small translations of the input, which is a desirable property for modelling images and many other types of data.

Unlike convolutional layers, pooling layers typically do not have any trainable parameters. By alternating convolutional and pooling layers, higher layers in the network see a progressively more coarse representation of the input. As a result they are able to model higher level abstractions more easily, because each unit is able to see a larger part of the input. Convolutional neural networks constitute the state of the art in many computer vision problems. Since their effectiveness for large-scale image classification was demonstrated, they have been ubiquitous in computer vision research [2][3].

## VI. APPROACH

We describe the steps in our processing pipeline and how to handle overfitting. The proposed pipeline consists of three steps: input pre-processing, augmentation and a convolutional neural network.

### *A. Handling overfitting*

Convolutional neural networks require enormous amounts of training data due to very high numbers of parameters, but our training set has only 61,578 images. A natural consequence of this is that there is a high risk of our network overfitting. Overfitting means that our network will try to memorize the training images, and will not generalize properly to new data.

### *B. Pre-processing*

The first stage is to perform dimensionality reduction by cropping the images which can then be followed up with a rescaling operation. We can confidently crop the images from 424 by 424 to 201 by 201 because the galaxy is located at the center of the JPEG image with the remaining portion being noisy background. We then down sample three times to 67 by 67 pixels.

### *C. Data augmentation*

We use data augmentation to artificially increase the size of the data set.

The following steps were used:

i. random rotation with an angle between 0° and 360
ii. translate the image by randomly shifting the samples uniformly between +5 and -5 pixels
iii. random rescaling with a scale factor selected log-uniformly between 1/1.6 and 1.6
iv. Bernoulli flipping the image with a probability of 0.5
v. The colour of the image is adjusted as described in [2]. This amounts to brightness adjustment.

After pre-processing and augmentation, we proceed to extract the viewpoints by rotating, flipping and cropping the input images.

### *D. Network architecture*

All views were presented to the network as 45 by 45 pixels by 3 arrays of RGB values, scaled to the interval [0:1] and processed by the same CNN model. The resulting feature maps were then combined and fed to a stack of two fully connected layers to map them to the 37 answer probabilities.

There are four convolutional layers, all with square filters, with filter sizes 8, 4, 3 and 3 respectively, and with untied biases. The ReLU non-linearity is applied after each layer [12]. A 2 by 2 max-pooling follows the first, second and fourth convolutional layers. The combined feature maps from all viewpoints are processed by a stack of two fully connected layers, consisting of two maxout layers [13] with 2048 units with two linear filters each, and a linear layer that outputs 37 realnumber.

### *E. Training*

To train the models we used mini batch gradient descent with a batch size of 20 and Nesterov momentum [19] with coefficient $\mu$= 0.9. Nesterov momentum is a method for accelerating gradient descent by accumulating gradients over time in directions that consistently decrease the objective function value. This and similar methods have become popular for neural network training in recent years, because they speed up the training process and often lead to improved

 

predictive performance. We refer to Sutskever et al [20] for a thorough treatment and evaluation. Following Krizhevsky et al. [2] we used a discrete learning rate schedule to improve convergence. This was necessary to ensure convergence. Weights in the model were During training, we used DropConnect [17] in all two dense layers.Dropout [16], is perhaps a biggest invention in the field of neural networks in recent years

## VII. IMPLEMENTATION

We implemented the model using Python and Theano on a 64 bit Linux OS [21],[22]. Theano was also used to perform automatic differentiation, which simplifies the implementation of gradient-based optimization techniques. Networks were trained on NVIDIA GeForce GTX 970 cards. Data augmentation was performed on the CPU using thescikit-image package [23] in parallel with model training on the GPU. Training the network took roughly 70 hours in real time.

## VIII. CONCLUSION AND FUTURE WORK

We used a convolutional neural network for galaxy structure classification.The network was trained on data from the Galaxy Zoo project and was able to predict various aspects of galaxy morphology directly from raw pixel data. It can automatically annotate large collections of images, enabling quantitative studies of galaxy morphology on an unprecedented scale.

Performing such large-scale analyses is an important direction for future research. We can also try model averaging by averaging the predictions of different models to increase the accuracy.From previous applications in the domain of computer vision, it has become apparent that the performance of convolutional neural networks scales very well with the size of the dataset. Another possibility is the application of our approach to classifying images from other projects in the zoo universe, including sun spots, craters on lunar surfaces, and images from mars and so on.

## REFERENCES

[1] Clery, D. "Galaxy Zoo Volunteers Share Pain and Glory of Research."*Science* 333.6039 (2011): 173-75.

[2] Alex Krizhevsky , Ilya Sutskever and Geoffrey E. Hinton , “ImageNet Classification with Deep Convolutional Neural Networks” in *Advances in Neural Information Processing Systems 25* ,2012

[3] Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki and Stefan Carlsson, “Visual Instance Retrieval with Deep Convolutional Networks.” In ICLR 2015.

[4] Odewahn, S. C., Stockwell, E. B., Pennington, R. L., Humphreys, R. M., & Zumach, W. A. (1992). Automated star/galaxy discrimination with neural networks. In Digitised Optical Sky Surveys (pp. 215-224). Springer Netherlands.

[5] Bertin, E. (1994). Classification of astronomical images with a neural network. In *Science with Astronomical Near-Infrared Sky Surveys* (pp. 49-51). Springer Netherlands.

[6] Banerji, M., Lahav, O., Lintott, C. J., Abdalla, F. B., Schawinski, K., Bamford, S. P., ... & Vandenberg, J. (2010). Galaxy Zoo: reproducing galaxy morphologies via machine learning. *Monthly Notices of the Royal Astronomical Society*, *406*(1), 342-353.

[7] Aguerri, J. A. L., Bernardi, M., Mei, S., & Almeida, J. S. (2011). Revisiting the Hubble sequence in the SDSS DR7 spectroscopic sample: a publicly available Bayesian automated classification. *Astronomy & Astrophysics*, *525*, A157.

[8] Lintott, Chris J., Kevin Schawinski, Anže Slosar, Kate Land, Steven Bamford, Daniel Thomas, M. Jordan Raddick, Robert C. Nichol, Alex Szalay, Dan Andreescu, Phil Murray, and Jan Vandenberg. "Galaxy Zoo: Morphologies Derived from Visual Inspection of Galaxies from the Sloan Digital Sky Survey ." Monthly Notices of the Royal Astronomical Society 389.3 (2008): 1179-189.

[9] Lintott, Chris, Kevin Schawinski, Steven Bamford, Anže Slosar, Kate Land, Daniel Thomas, Edd Edmondson, Karen Masters, Robert C. Nichol, M. Jordan Raddick, Alex Szalay, Dan Andreescu, Phil Murray, and Jan Vandenberg. "Galaxy Zoo 1: Data Release of Morphological Classifications for Nearly 900 000 Galaxies." Monthly Notices of the Royal Astronomical Society 410.1 (2010): 166-78.

[10] Kyle W. Willett, Chris J. Lintott, Steven P. Bamford, Karen L. Masters, Brooke D. Simmons, Kevin R. V. Casteels, Edward M. Edmondson, Lucy F. Fortson, Sugata Kaviraj, William C. Keel, Thomas Melvin, Robert C. A. Skibba, Arfon M. Smith, Daniel Thomas: “Galaxy Zoo 2: detailed morphological classifications for 304,122 galaxies from the Sloan Digital Sky Survey”, in Monthly Notices of the Royal Astronomical Society (2013) : 1-29

[11] Bengio, Y. "Learning Deep Architectures for AI." Foundations and Trends in Machine Learning FNT in Machine Learning 2.1 (2009):1-127.

[12] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Proc. 27th International Conference on Machine Learning.

[13] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, Yoshua Bengio: “Maxout Networks”, 2013, JMLR WCP 28 (3): 1319-1327, 2013;arXiv:1302.4389.

[14] Dieleman, S., Willett, K. W., & Dambre, J. (2015). Rotation-invariant convolutional neural networks for galaxy morphology prediction. *arXiv preprint arXiv:1503.07077*.

[15] Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kremer, S. C. and Kolen, J. F., editors, A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press.

 

[16] Dahl, G. E., Sainath, T. N., & Hinton, G. E. (2013, May). Improving deep neural networks for LVCSR using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (pp. 8609-8613). IEEE.

[17] Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., & Fergus, R. (2013). Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)* (pp. 1058-1066).

[18] Boureau, Y. L., Ponce, J., & LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (pp. 111-118).

[19] Bengio Y., Boulanger-Lewandowski N., Pascanu R., 2013, in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) Advances in optimizing recurrent networks. pp 8624-8628

[20] Sutskever I., Martens J., Dahl G., Hinton G., 2013, in Proceedings of the 30th International Conference on Machine Learning (ICML-13) On the importance of initialization and momentum in deep learning. pp 1139-1147

[21] Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A. & Bengio, Y. (2012). Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*.

[22] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G & Bengio, Y. (2010, June). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)* (Vol. 4, p. 3).

[23] Van der Walt, S., Schönberger, J. L**, J.**, Boulogne F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., and the scikit-image contributors (2014). scikit-image: image processing in Python. PeerJ 2:e453, PeerJ PrePrints 2:e336v2.

 