Astroinformatics
Proceedings IAU Symposium No. 325, 2016
M. Brescia, S.G. Djorgovski, E. Feigelson,
G. Longo & S. Cavuoti, eds.

© International Astronomical Union 2017
doi:10.1017/S1743921316012771

Galaxy Classifications with Deep Learning

Vesna Lukic and Marcus Brüggen

Hamburger Sternwarte, University of Hamburg,
Gojenbergsweg 112, 21029, Hamburg, Germany
email: `Vesna.Lukic@hs.uni-hamburg.de, Marcus.Brueggen@hs.uni-hamburg.de`

Abstract. Machine learning techniques have proven to be increasingly useful in astronomical applications over the last few years, for example in object classification, estimating redshifts and data mining. One example of object classification is classifying galaxy morphology. This is a tedious task to do manually, especially as the datasets become larger with surveys that have a broader and deeper search-space. The Kaggle Galaxy Zoo competition presented the challenge of writing an algorithm to find the probability that a galaxy belongs in a particular class, based on SDSS optical spectroscopy data. The use of convolutional neural networks (convnets), proved to be a popular solution to the problem, as they have also produced unprecedented classification accuracies in other image databases such as the database of handwritten digits (MNIST †) and large database of images (CIFAR ‡). We experiment with the convnets that comprised the winning solution, but using broad classifications. The effect of changing the number of layers is explored, as well as using a different activation function, to help in developing an intuition of how the networks function and to see how they can be applied to radio galaxy images.

Keywords. Galaxy morphology, deep learning, classifications, deep neural networks

1. Introduction

The study of galaxy morphology can help us to learn about the evolution of the universe. We now have quite a detailed understanding of the galaxies in the present day universe which includes properties such as age, luminosity and morphology. This knowledge forms a background against which galaxy evolution can be tested and quantified (Giavalisco 2006). Certain types of galaxies with particular morphologies are more likely to exist at certain epochs in time, hence it is worthwhile to classify the galaxy morphologies. This will be an impossible task to do manually in the future, as current and upcoming surveys such as the LOw Frequency ARray (LOFAR ¶) and Square Kilometre Array (SKA ||) will detect increasingly higher numbers of radio galaxies. SKA alone will discover up to a billion galaxies (Whittam 2016). The classification of galaxy morphologies is based on a few simple rules that can be taught to non-experts, which makes it suitable for machine learning tools.

Citizen science emerged from such a task being presented to non-experts. The Kaggle Galaxy Zoo (Willett *et al.* 2013) was a competition where the aim was to predict the probability distribution of how citizen scientists responded to questions about a galaxies morphology. The responses were broken down into 37 different categories based on the optical galaxy image data. In order to increase the reliability of the users prediction, each galaxy was presented to a number of different users, and their responses were converted to a probability distribution. Given the image data as the input, and the probability

† http://yann.lecun.com/exdb/mnist/
‡ https://www.cs.toronto.edu/ kriz/cifar.html
¶ http://www.lofar.org
|| https://www.skatelescope.org

distribution as the output which must be predicted, this is a supervised machine learning task involving regression.

The winning solution to the competition used convnets (Dieleman *et al.* 2015). With regards to machine learning with galaxy images, the convnet approach has mainly been taken on optical galaxy images, and it has only very recently started to take place on radio galaxy images (Alger 2016). This work aims to develop a convnet to classify radio galaxies according to some classification scheme, such as Fanaroff-Riley (Fanaroff & Riley 1974).

2. Overview

Neural networks can be used to perform classifications of data. They are initialised with a set of weights and biases in the hidden layers. The pixel intensities of the images with the correponding labels are input into the network, propagated through the network and the output layer computes a prediction. An error is calculated based on the difference between the true output and the predicted output, known as a cost or loss function. This error is propagated back through the network, and the network adjusts the weights and biases to reduce the error. These steps are iterated a number of times until the cost function is minimised, a process known as training (Nielsen 2015).

In traditional neural networks, the nodes in the hidden layers are fully connected to the nodes in the adjacent layers. Therefore, the deeper the network becomes, the more computationally intensive and time consuming it is to train, and the propagated gradient becomes smaller with every added layer, leading to the vanishing gradient problem. This stalls the training loss, hence prevents the network from learning further. Convolutional neural networks circumvent this problem, by introducing a number of user-defined filters with initialised weights and biases, that are connected to a small spatial region of the input data. This greatly reduces the number of connections, hence the network is easier and faster to train. The amount of data in the convolutional layers is further reduced by using a pooling layer, where only the maximum output in a certain region is stored. The convolutional and pooling layers are stacked with the end result being a hierarchical extraction of features. These layers are usually followed by one or more fully-connected layers, before finishing at the output layer, where a prediction is given (Karpathy 2016).

A big advantage of using convnets on image data is that minimal preprocessing is required. The convnet learns the features of the images when they are close to their original form and performs the feature extraction. The prediction accuracy increases with the number of images that are provided, and image augmentation can be used to artificially produce more image data (Krizhevsky *et al.* 2012). In the case of galaxy images, augmentation can be done through transformations such as rotation and translation, as they are invariant to these. (Dieleman *et al.* 2015).

Creating augmented images also helps to reduce overfitting, which occurs when the architecture and parameters in the neural network fails to generalise to a dataset extracted from the same source. Another way to reduce overfitting is to use dropout, where a certain proportion of connections to nodes in adjacent layers are dropped to stop the network relying on the presence of particular neurons, hence it is made to be more robust. The winning solution of the KGZ (Dieleman *et al.* 2015) was used to explore the behaviour of networks with different numbers of layers, in order to explore what the layers in the network see and to prevent the addition of unnecessary layers. The user solutions were converted into a vector of 8 classifications, and the cost function used was the mean squared error (mse). The mse in this case is the **label** vector corresponding to the type of galaxy, subtracted from the **prediction** vector (composed of a vector of 8 probabilities), squared, then calculating the mean.

One other difference used here that was not available at the time the winning solution was made, is the Parametric Rectified Linear Unit (PReLU) activation function. This is a generalisation of the ReLU. The ReLU function is max(0,x), therefore only positive inputs are sent forward, and the negative ones are set to 0. This makes the network more sparse, therefore more efficient. Since the output is linear only in parts of the network, this ensures that the gradients flow across the active neurons, hence avoiding the vanishing gradient problem (Glorot, Bordes & Bengio 2011). The difference with a PReLU is that the negative part is not 0, but is instead a negative linear function that uses a coefficient to control the slope. When the coefficient is 0, the function reduces to the ReLU. The coefficient is initialised for each convolutional layer, and is adjusted through training. It is successful in avoiding zero gradients. The introduction of the coefficient adds very few extra parameters, and is negligible compared to the total number of weights, hence there is no extra risk of overfitting (He *et al.* 2015).

With the insights acquired from experimenting with the number of layers and different activation functions, we can have a better idea of how to build and train a convnet to classify radio galaxy images.

3. Results

The neural networks were trained with stochastic gradient descent with a minibatch size of 16. 200 training epochs were used. The learning rate was set to 0.04 at the start of training, decreased to 0.004 at epoch 100, and finally 0.0004 at epoch 150. All 61578 images were used in total, with 90% for training and 10% for validation. The chunk size was 500. Both the ReLU and PReLU (He *et al.* 2015) activation functions were explored. These activation functions are used in the convolutional layers only. Dropout is applied to the fully connected layers, with a dropout probability of 0.5.

The input layers in Figure 1 represent cropped downscaled images of size (69,69) at 0 and 45 degrees, that are also horizontally flipped. They are then cut into 4 overlapping quadrants of size (45,45). This results in 16 images obtained from a single image.

One conv2D one pooling with ReLU

The network fails to detect any features when only one convolutional layer is used, indicating that the model is too simple and more layers need to be put in.

Two conv2D two pooling with ReLU

With the addition of an extra convolutional and pooling layer, the network begins to detect features, coming from the first convolutional layer. However, the second one now fails to detect features, again indicating the model is not big enough to capture the complexity in the images.

Full model with PReLU

All three convolutional layers now output features, therefore the network is learning during training. The activation function used in this model is the Parametric ReLU (PReLU). As we go further along the convolutional layers, more complex, subtle features are detected.

Summary

Table 1 summarises the training and validation losses for the different models explored. The results indicate that the more complex the model is, the lower will be the loss, therefore the predictions are more accurate, and the model will detect the complex features better. The training and validation losses within a model should be close; if they are very different it is an indicator of overfitting. The best performing model is the full model with the PReLU activation function, most likely because the coefficient of the negative part is adjusted through training and results in positive gradients being propagated, compared to the ReLU where the negative parts are set to 0 (Gu *et al.* 2016).

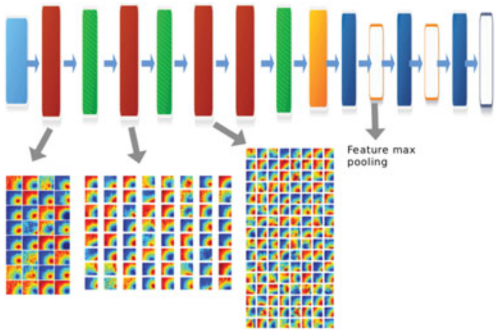

Figure 1. Architecture of full model from the winning solution and output of filters from the first three convolutional layers. The different numbers of tiles correspond to the number of filters specified for the convolutional layers by the user. Each tile is a feature map. The colours in the tiles represent different activation values of each convolutional layer, red and blue indicate high and low activation values respectively.

	Mean training loss	**Mean validation loss**
One conv one pool ReLU	0.279	0.176
Two conv two pool ReLU	0.18	0.175
Full model ReLU	0.178	0.176
Full model PReLU	0.169	0.165

Table 1. Training and validation losses calculated up to epoch 200, across models of increasing complexity. The optimal error achieved across both training and validation sets is by the full model using the PReLU activation function.

4. Implications

We have explored deep neural architectures of increasing complexity and two different activation functions. Adding layers results in lower, more stable training losses, however adding unnecessary layers results in more overfitting. The PReLU activation function performs better than the ReLU.

References

Dieleman, S., Willett, K. W., & Dambre, J. 2015, *Mon. Not. R. Astron. Soc.*, 000, 1-20
Willett, K. W. 2016, *Proceedings of Science*
He, K., Zhang, X., Ren, S., & Sun, J. 2015, *Microsoft Research*
Glorot, X. & Bengio, Y. 2010, *JMLR W&CP*, 9:249-256
Glorot, X. Bordes, A., & Bengio, Y. 2011, *JMLR W&CP*, 15:315-323
Giavalisco, M. 2006, *IOP Publishing Ltd*
Whittam, I. 2016, *Huffington post http://www.huffingtonpost.com/the-conversation-africa/radio-galaxies-the-myster_b_12554818.html*
Willett, K. W., Lintott, C. J., & Bamford, S. P., *et al.* 2013, *Mon. Not. R. Astron. Soc.*, 000, 1-29
Nielsen, M. A. 2015, *Determination Press*
Karpathy, A. 2016, *CS231n Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/convolutional-networks//#overview*
Alger, M. 2016, *PhD Thesis*
Gu, J., Wang, Z., & Kuen, J., *et al.* 2016, *arXiv:1512.07108 [cs.CV]*
Krizhevsky. A, *et al.* 2012, *Advances in Neural Information Processing Systems 25*
Fanaroff, B. & Riley, J. 1974, *Mon. Not. R. Astron. Soc.*