Conditional generative adversarial network and identification of planetary nebulae

Chow Yin Hei
Supervisor: Prof. Quentin A. Parker

Summer Research Fellowship 2022
Poster No.: D5
Name: Chow Yin Hei
University No.: u3035922934
Student's Major: Physics and mathematics

Abstract

This is an attempt to, by the method of conditional generative adversarial network (cGAN), obtain a generative planetary nebula (PN) dataset out of limited real data. The validity of such a dataset would be determined by the performance of a PN identifier trained on it versus when trained on a real, established database, along the dimension of interest, which in this case, will be the identification problem of PN from miscellaneous celestial objects. Result shows that with 2,502,018 parameters in the network, no high-level features can be extracted from the real dataset.

Introduction

Few machine learning techniques like transfer learning or cluster analysis have been used in enhancing the study of PN. Yet a fundamental constraint in optimizing these models, and in the broader study of PN, remains to be the need for a large amount of data. It is due to this reason that this project set out to test whether the features characterising PN can be extracted with the framework of generative adversarial network (GAN) (Goodfellow, 2014), since success of the method means a generative dataset is obtained.

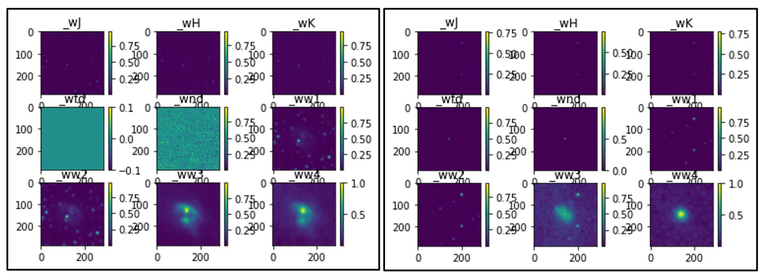


Figure 1. Samples from the training dataset. From left to right, samples are from the class Fpn and Tpn. (Note that not all structures present in the images can be seen due to sensitivity of the human eye)

Methodology

Our model is defined recursively for $l > 0$ using the formulae

$$\begin{cases} x^l_{pqr} = \sum_{k=0}^{c-1}\sum_{j=0}^{n-1}\sum_{i=0}^{m-1} \omega^l_{rijk}\tilde{o}^{l-1}_{i+sp,j+sq,k} + b^l_{pqr} & \text{(linear)} \\ o^l_{pqr} = a(x^l_{pqr}) & \text{(nonlinear)}, \end{cases}$$

where values of the hyperparameters s, m, n, c depends on l, a is the activation function, $\tilde{o}$ is obtained through batch normalization of o over the training batch for layers between hidden layers, with $\tilde{o}=o$ otherwise, $\tilde{o}^0$ is the inputted array, and the paramater arrays w, b are adjusted in optimization.

For generator, $\widetilde{o^0}$ is the Hadamard product of the sparse vector embedding of the conditional information with a gaussian noise. For discriminator, $\widetilde{o^0}$ is the concatenation of its sparse vector embedding of the conditional information with the image input.

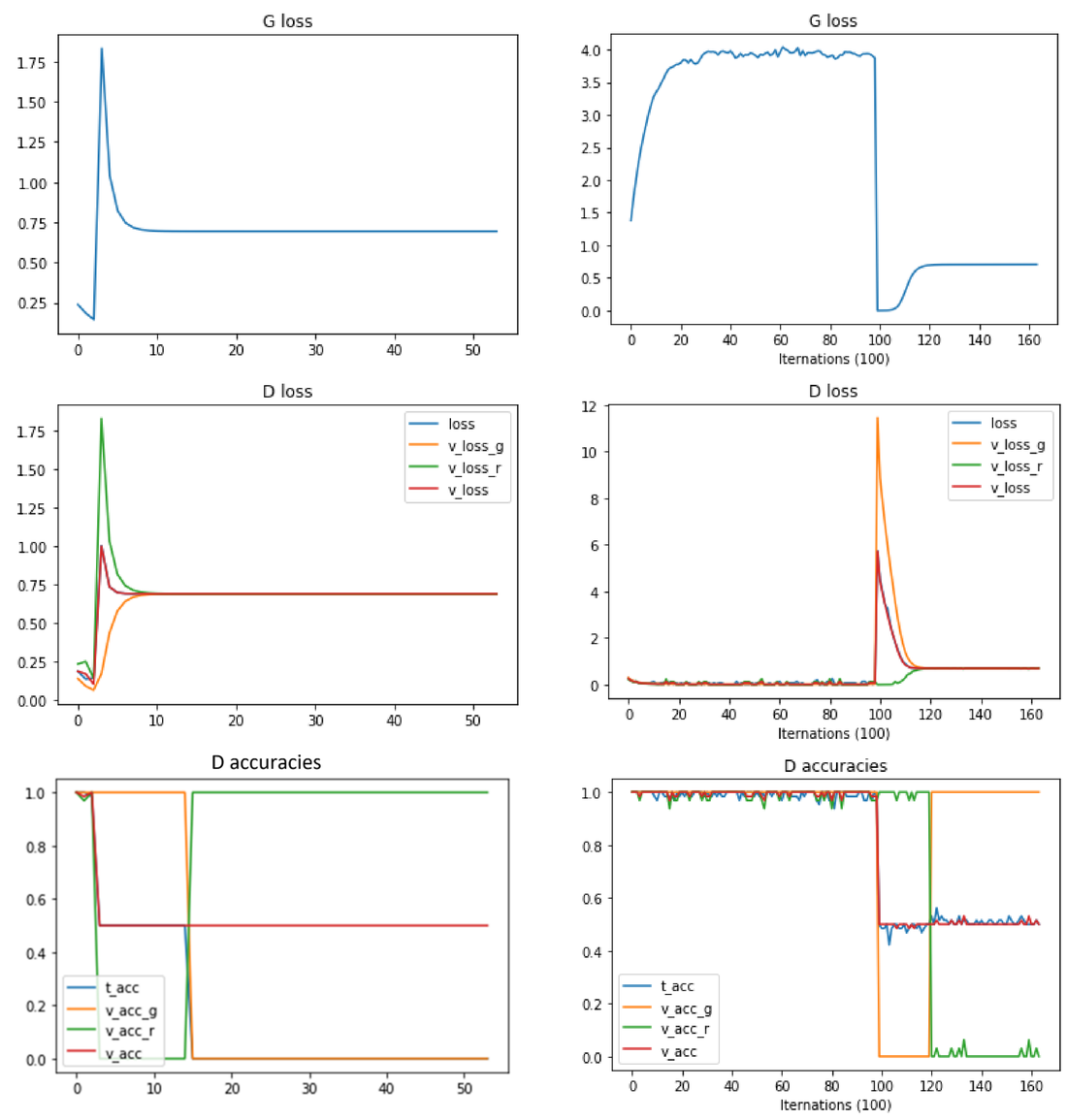
- Convolutional layers before the image input space have stride one-half, and those coming after the input space have stride two. In the same spirit with the zero-padding in data pre-processing step, the arrays $\tilde{o}$ are also zero-padded in the sense that index values at which $\tilde{o}$ is not defined are set to zero. This way during the correlation operation, fractional or out-of-range indexes would not cause a problem. An alternative would be to use interpolation.
- Before entering the dense layers, an array o is flattened to obtain $\tilde{o}$, such that $\tilde{o}$ is run by a single index (k in the above formula).
- The embedding layer takes similar form of a dense layers except the bias vector b is set to zero. The labels r and g, represented as standard vectors $\widehat{e_0}$ and $\widehat{e_1}$, simply multiplies the embedding matrix to extract a column as their sparse vector. The embedding matrix is a trainable parameter.

Performance Metric and Optimization

- The performance of the generator in capturing the features of PN is measured by how well the PN images it generates (g) resembles real PN images (r). Since this difference is taken between high-level abstract features, a neural network (D) is used to do this measurement, which responses a value from 0 to 1.
- The performance of D in doing its task is in turn measured by the binary cross-entropy loss. By definition, it should return a regression value of 1 if given a real image, so it is trained on both real and generated data.
- The gradient of the cost function with respect to the trainable parameters are computed using automatic differentiation, and ADAM optimizer is used to adjust the gradient with an approximation of the Hessian (Kingma & Lei, 2014). The information is then used to update the parameters of the model.

The assumption made here is that there exists a set of latent variables that on their own fully describe the morphology of PN under the nine filters under investigation. If the number of data point available is much larger than the number of these latent variables, the cGAN could capture these latent variables. Memory of these latent variables are said to be stored in the generator neural network if the generator can reconstruct a PN image from a random seed it's given.

Results and Discussion



Plot of the losses and accuracies. For D, loss, validation loss (v_loss), validation loss from real data (v_loss_r), validation loss from generated data (v_loss_g) are plotted separately. The accuracies are calculated from their respective loss. Note that accuracy for G is not well-defined, nor is there a difference in validation and training for G. The sampling interval is 100, and above shows the interaction between D and G over 5200 iterations (first column), or over 16000 iterations for model without batch normalization (second column), with batch size 32.

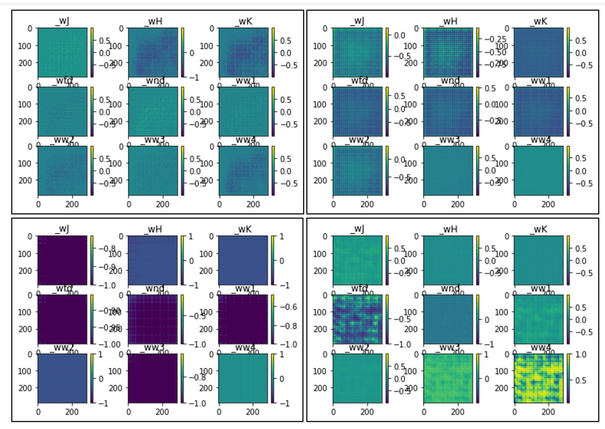
Should the result be successful, we should see D able to push G loss back up, and the cycle repeats, with each cycle ending with G replicating the next feature in the class r, and eventually, the use of the conditional labels T and F by D would lead G into features of the subclass T that distinguish them from the rest of r, ending with a replication of underlying features of PN. The information can extracted by training classification networks on it, or be extracted from D. Yet we see D staying in the confuse state of 0.5.

To eliminate the possibility of the failure in D coming from the batch normalization layers being too restrictive to the output that D can give, a test was done with no batch normalization layers in the cGAN. (shown in second column)

The generated images may look bad, but they are enough to fool D into thinking that they are from the real dataset, removing the incentive for G to delve into higher-level features in the real samples, not to mention into the subclass of PN.

Possible limitations

1. The sample number of 600 may not be enough for the cGAN to identify high-level features
2. The number of latent variables specifying the features of T or r may be too large
3. The design of the network may not be suitable for uncovering the inner structure of the class r, due to the heterogeneity of the class F



Sample outputs from generators with different hyperparameters.

Suggestions for improvement

- Conditional batch normalization could be used to have different scaling and shift for the two classes T and F. Yet it is unlikely a concern before D is able to identify the high-level features distinguishing the classes r and g.
- Self-attention layers could be added to D to help it pull out features that are not close together in intermediate feature maps.
- Separating the class F into known subclasses may help distinguishing them from T.

Reference list

Frew, D.J. & Parker, Q.A. (2010). Planetary Nebulae: Observational Properties, Mimics and Diagnostics. *Publications of the Astronomical Society of Australia*, 27, 129-148.

Goodfellow, I.J..; et al. (2014). Generative Adversarial Nets. *Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014).* pp. 2672–2680.

Jacoby, G.H.; et al. (2010). Searching for Faint Planetary Nebula Using the Digital Sky Survey. *Publications of the Astronomical Society of Australia*, 27, 156-165.

Kingma, D.P & Ba, J.L. (2014). ADAM: A Method for Stochastic Optimization. *arXiv*:1412.6980.

Radford, A.; Metz, L.; Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *ICLR.* S2CID 11758569.

Parker, Q.A.; , Boji˘ci´c, I.S.; Frew, D.J. (2016). HASH: the Hong Kong/AAO/Strasbourg Hα planetary nebula database. *Journal of Physics: Conference Series*, 728, 032008.