Rapid Prototyping of Species Classifiers using Deep Learning: A Guide for Non-Experts

Herman Njoroge Chege[1]

[1]Department of Biology, College of Arts and Sciences, University of Vermont, 326A Marsh Life Sciences, 109 Carrigan Drive, Burlington, VT 05405

March 14, 2020

Abstract

Deep learning algorithms are revolutionizing how hypothesis generation, pattern recognition, and prediction occurs in the sciences. In the life sciences, particularly biology and its subfields, the use of deep learning is slowly but steadily increasing. However, prototyping or development of tools for practical applications remains in the domain of experienced coders. Furthermore, many tools can be quite costly and difficult to put together without expertise in Artificial intelligence (AI) computing. We built a biological species classifier that leverages existing open-source tools and libraries. We designed the corresponding tutorial for users with basic skills in python and a small, but well-curated image dataset. We included annotated code in form of a Jupyter Notebook that can be adapted to any image dataset, ranging from satellite images, animals to bacteria, or even data such as song or echolocation recordings transformed into images. The prototype developer is publicly available and can be adapted for citizen science as well as other applications not envisioned in this paper.

We illustrate our approach with a case study of 219 images of 3 three seastar species. We show that with minimal parameter tuning of the AI pipeline we can create a classifier with 87% accuracy. We include additional approaches to understand the misclassified images and to curate the dataset to increase accuracy. The power of AI approaches is becoming increasingly accessible. We can now readily build and prototype species classifiers that can have a great impact on research that requires species identification and other types of image analysis. Such tools have implications for citizen science, biodiversity monitoring, and a wide range of ecological applications.

1 Introduction

Deep learning, a branch of machine learning, is an artificial intelligence approach which has been used for pattern recognition across multiple domains (Shen, Wu, & Suk, 2017; Golden, 2017; Min, Lee, & Yoon, 2016; Heaton, Polson, & Witte, 2016; Esteva et al., 2019; Esteva et al., 2017). Whereas other machine learning approaches have been used for acoustic classification (Aide et al., 2013), ecological modelling and studying animal behaviour (Olden, Lawler, & Poff, 2008; Valletta, Torney, Kings, Thornton, & Madden, 2017; Christin, Hervet, & Lecomte, 2019), deep learning approaches have demonstrated the ability to overcome several machine learning limitations. One of the challenges of machine learning approaches is the need for superior domain knowledge and high-level programming skills (LeCun, Bengio, & Hinton, 2015; Christin, Hervet, & Lecomte, 2019)(Deng et al., 2009; Yosinski, Clune, Bengio, & Lipson, 2014). Further, the data feature engineering step in machine learning is a complex and often tedious task that discourages many from using these techniques. Deep learning overcomes this feature engineering step by ensuring that the algorithm finds features by itself automatically (Jiang et al., 2018).

In ecology, however, the use of deep learning is still in its infancy. A literature review puts both peers and non-peer reviewed papers at 46 as of April 2018, mostly using Convolutional Neural Networks (CNNs) and

Recurrent Neural Networks(RNNs) (Christin, Hervet, & Lecomte, 2019). This is despite its potential to revolutionalize applied ecology in identification and classification of species, behavioural studies, population monitoring and citizen science, ecosystem management and conservation (Christin, Hervet, & Lecomte, 2019; Lamba, Cassey, Segaran, & Koh, 2019; Miao et al., 2019; Ditria et al., 2019). Several research articles continue to implement new, novel and interesting applications: such as using contextual data with image data and generating optimum evolution of camouflage using Generative Adversarial Networks (Terry, Roy, & August, 2020; Talas et al., 2019). However, the techniques used still remain cryptic and inaccessible to most ecologists who are experts in their domains but who have no experience with these techniques.

Naturalists have been identifying species for the past two centuries, laying the foundations of the ecological science. However, even today, most of the taxonomic work and species identification work is still manual and reliant on a few domain experts. Ecology in particular is ripe for the applications of deep learning owing to the increase in complex ecological datasets over the past few years ranging from genomic to ecosystem-scale data, also known as Big Data (White & Bahlai, 2019; Hampton et al., 2013; Farley, Dawson, Goring, & Williams, 2018). The Big data derived from the increasingly sophisticated automatic monitoring by sensors can no longer be manually processed as it is redundant and time consuming (Weinstein, 2017; Norouzzadeh et al., 2018). Deep learning is specifically better than other methods in dealing with non-linear complex data commonly encountered in ecology (Christin, Hervet, & Lecomte, 2019). In fact, all winning methods for the most recent LifeCLEF contests which is a series of multimedia species identification challenges have been deep learning-based (Joly et al., 2017). Reviews and proposals for these have been put forward and the field feels right for disruption (Christin, Hervet, & Lecomte, 2019; Lamba, Cassey, Segaran, & Koh, 2019). Deep learning has been touted as a contender in solving problems with immediate application ranging from illegal trafficking of wildlife products to large scale automated ecosystem management tools - areas that are expensive and logistically expensive to manage (Cantrell, Martin, & Ellis, 2017; Christin, Hervet, & Lecomte, 2019).

A lot of the challenges that prevented deep learning from having practical applications have been eliminated with advancements in research on transfer learning and data augmentation (Shorten & Khoshgoftaar, 2019). This has led to a reduction in the data required to make accurate world-class models. Furthermore, the recent wave in computer hardware innovation for GPU's and CPU's has also accelerated by reducing the cost of accessing the processing power required for accurate model development.

This paper is intended for non-AI experts who want to build species classifiers. We have attached an annotated open source Jupyter Notebook. The tutorial is designed in a way that it can be implemented in the lowest resourced environment and unlock great application in taxa image identification in ecology the world over that we can hardly imagine at the moment. The code can be accessed from the Jupyter Notebook here: `https://bit.ly/39woeLt`

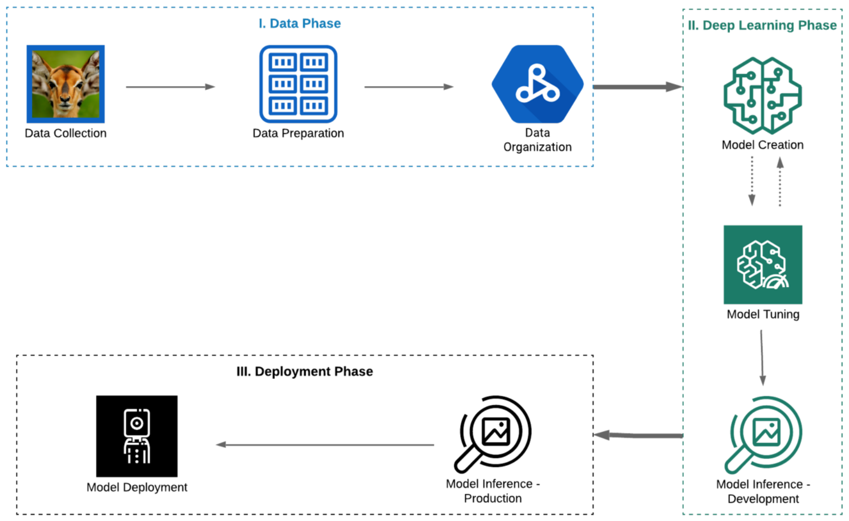

Figure 1: Overview of the various phases for prototyping of image data species classifiers using deep learning

1.1 LIBRARIES AND TOOLS

The following open-source tools were used to develop the functional minimum viable product for the least resourced environment.

Python is a user-friendly general-purpose programming language that has been used to develop many data science libraries and recently deep learning frameworks due to its ease of use and similarity to plain English. Fast.ai is a high-level AI library built on top of the open-source deep-learning library Pytorch released by Facebook in 2017 (Paszke et al., 2017; Howard & others, 2018). The library specializes in rapidly implementing state-of-the-art techniques from newly published research papers. Jupyter notebooks was used as the platform to implement the python code due to its ease of use and reproducibility - in fact, they have been the go-to tool for data scientists in the recent past as notebooks can be easily shared and run compared to scripts that were sometimes cryptic to non-experts (Kluyver et al., 2016; Randles, Pasquetto, Golshan, & Borgman, 2017).

Google Colab is our training platform, because of its free GPU's offered by Google. To train the deep learning model, GPUs have been identified to be better and faster for matrix multiplications compared to CPUs (Shi, Wang, Xu, & Chu, 2016; Kayid, Khaled, & Elmahdy, 2018). These processing units that started out with applications for video games have gained popularity as the go-to for training deep learning models.

In this paper, we are going to use a Jupyter notebook running on top of Google Colab to help guide the readers to implement it on their own as they read the paper. We will default to Google Colab since most of the configuration has been set up for us to use in this platform. This will ensure we worry only about our problem of building a species classifier and not waste time in the configuration.

1.2 DATA COLLECTION

Data collection is an important phase and the first phase in the AI model development pipeline as data collected will determine the accuracy of models or lack thereof. Many approaches can be employed to achieve this: from data discovery, data augmentation to data creation (Roh, Heo, & Whang, 2018). In our ecological context, depending on your species of interest - data can either be manually gathered or acquired from other sources. For image classification tasks such as in our case study, online repositories such as the iNaturalist or GBIF websites have tons of image data that can be accessed using APIs or other advanced data mining approaches (“GBIF”, n.d.; “iNaturalist”, n.d.). Here we go with the assumption that you already have a relatively clean curated dataset and that has balanced classes for each species and which has no noise in the ground truth labels and that the problem is a supervised learning challenge i.e where we already know the labels of our datasets.

Species Image Data used for Classification

In this case study we will use three sea star species. Sea stars are important species in our understanding of marine invertebrate communities. Intertidal relationships between the sea star *Pisaster ochraceus* and the mussel *Mytilus californicus* was actually used to coin the term keystone species (Paine, 1966). Following that classical study, it would, therefore, be interesting to use sea stars as case study species to prototype the classifier AI system. Further, seastars have complicated morphology that might be a challenge even for expert humans - Following that classical study, it would, therefore, be interesting to use sea stars as model species to prototype the classifier AI system. Further, seastars have complicated morphology that might be a challenge even for expert humans - for these reasons we use them to prototype our AI system. Figure 1 illustrates our the general workflow of a deep learning pipeline meant to achieve a minimum viable product:

2 Implementation

2.1 PREPARATION

2.1.1 *Gpu Session Initialization*

Go to the code repository here: `https://bit.ly/39woeLt`. Click on the open with Colab button at the top of the page. It will open a Google Colab page with the Jupyter notebook that runs on an underlying CPU. Before starting out it is important to change your session to a GPU session to take advantage of the aforementioned Google free GPUs that will greatly accelerate your model creation by following the following steps:

- Click on ***Runtime*** on the Menu
- on the panel that appears click on ***Change runtime type***
- on the new pop-up change the ***Runtime type*** to *Python 3* (if it is not already the default) and most importantly the ***hardware acceleration*** to *GPU*

2.1.2

2.1.3 *Importing Libraries*

We mentioned several libraries that we are going to use above. Here is our chance to import them for our use. On the Jupyter Notebook, It is good practice to begin by importing all your libraries. In our case the easy to use fast.ai AI algorithm, and on the second line accessing and importing its vision functionalities to our session:

```
Refer to #CODE BLOCK 1# on the Jupyter Notebook
```

2.2 DATA PHASE

2.2.1 *Importing Data*

In this case study we use three sea star species: *Pisaster ochraceus*, *Pycnopodia helianthiodes* and *Solaster dawsonii*. The data was scraped from publicly available image sources from google using a python script. We will simplify the exercise by assuming that we have data already organized in folders as follows in your local computer:

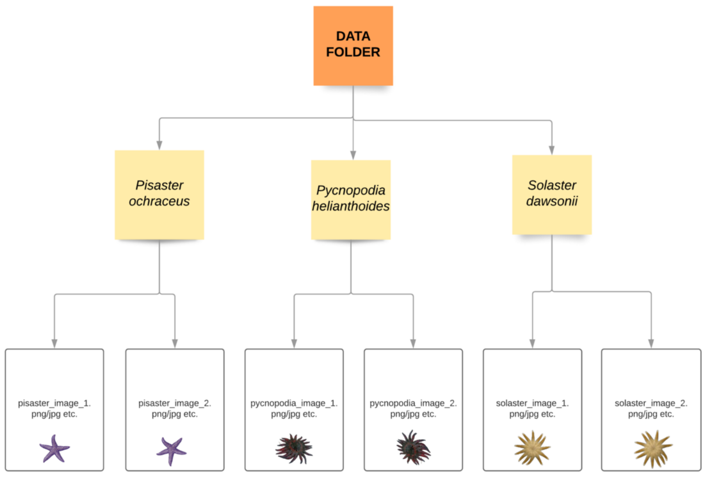
DATA FOLDER
Pisaster ochraceus
Pycnopodia helianthoides
Solaster dawsonii
pisaster_image_1. png/jpg etc.
pisaster_image_2. png/jpg etc.
pycnopodia_image_1. png/jpg etc.
pycnopodia_image_2. png/jpg etc.
solaster_image_1. png/jpg etc.
solaster_image_2. png/jpg etc.

Figure 2: Data organization for the classification of three sea star species: *Pisaster ochraceus*, *Pycnopodia helianthiodes* and *Solaster dawsonii*

The species image data is arranged into subfolders for each class of species that contain different images of the same species. The data is then uploaded as a folder to the platform that we are going to use. In our case study, we simply uploaded our data to google drive and were able to access this data from Google Colab by running the first line of code in the Jupyter notebook.

`Refer to #CODE BLOCK 2# on Jupyter Notebook`

This connects your data on google drive to your Jupyter notebook session running on Google Colab. You might need to provide a secret code on the output to give permissions for this to happen.

2.2.2 *Organizing Data*

The next thing you have to do is save that data path as a path variable and use that path to create training, validation datasets.

`Refer to #CODE BLOCK 3# on Jupyter Notebook`

The training dataset is what the deep learning algorithm will use to learn the features of how one seastar class differentiates itself from another. The validation set avoids overfitting of the data to ensure that our model can generalize broadly to other sea stars it has not seen before. This prevents the danger of overfitting - which is simply the model "cramming into memory" the sea stars that are in the training set, this could lead to misleading interpretations where there is high model accuracy, but which cannot generalize to other sea stars, not found in the original training data set (Kohavi, 2001; Ripley, 1996).

The training dataset depends on the dataset, but would commonly be around 60-80% of the dataset, validation around 20% of the original dataset . You can automate this using the built-in function in fast.ai and creates a data subset to be fed into the neural network for the creation of the species classification model as illustrated by the following code blocks:

`Refer to #CODE BLOCK 4# on Jupyter Notebook`

We can ensure what we have in our data block is accurate by viewing the images and exploring the basic statistics of our various data organization folders created by the build-in fast.ai function by running the following code blocks:

`Refer to #CODE BLOCKS 5,6,7 # on Jupyter Notebook`

We can see that our folders have 3 classes, 219 images in the training dataset and 54 on the validation dataset. We can also see the visuals of the images and their respective labels.

2.3 DEEP LEARNING PHASE

2.3.1 *Deep Learning Model Creation*

In our case study, we have a computer vision problem, so we will use Convolutional Neural Networks (CNN) (Krizhevsky, Sutskever, & Hinton, 2017). In our case, we will use the RESNET 34 architecture that is not too complicated but delivers superior results compared to other architectures (He, Zhang, Ren, & Sun, 2015; Canziani, Paszke, & Culurciello, 2016). We use classification accuracy - which is the summation of the true positives and true negatives divided by the total number of examples as a test metric for our model. We then call in the build-in CNN-learner from the fast.ai library pass in our data, the RESNET 34 architecture and the metrics we are measuring for as illustrated in the following code blocks:

`Refer to #CODE BLOCK 8 # on Jupyter Notebook`

We can then run it for one cycle, passing in the number of epochs or backpropagation steps that we want the model to run for. In our case, we run for 10 epochs and achieved an accuracy of about 85%. Backpropagation is at the central of neural networks and involves minimizing the errors of the model we are creating using gradient descent (Alber et al., 2018).

`Refer to #CODE BLOCK 9 # on Jupyter Notebook`

Too few or too many epochs can be a problem and it is best to aim for stopping when there is no reduction in error rate or increase in accuracy if using accuracy metrics. Too few and the model does not learn efficiently and too many and the model starts to overfit to our data.

We then save the model - we can already have good enough accuracy for the next stage of inference and deployment as illustrated by the following code blocks:

`Refer to #CODE BLOCK 10 # on Jupyter Notebook.`

2.3.2

2.3.3 *Model Tuning*

If there is a need to create a more accurate model particularly while up against benchmark problems there are techniques such as retraining an already trained model, automatically searching for a suitable learning rate using the build-in learning rate finder which is novel to fast.ai. The technique to further tune the model can be found in the below code blocks:

```
Refer to #CODE BLOCKS 11,12,13 # on Jupyter Notebook
```

Then using that learning rate you can fit other cycles until the model gets to a suitable accuracy. Using this approach for our species classifier, we are able to improve the accuracy to 87%.

2.3.4 *Model Tuning by Data Exploration*

With a trained model, we now investigate particular images that might be causing the model to be less accurate. First, we draw a confusion matrix of the species to see the map of the true positives and true negatives and understand which species are making your model less accurate (Ting, 2017). In our case, and according to figure 3, we have misclassified 4% of Pisaster ochraceus, 1% of *Pyncopodia helianthoides*, and 2% of *S. dawsoni.* We also examined each misclassifed image to check the original data curation accuracy (Fig. 4).

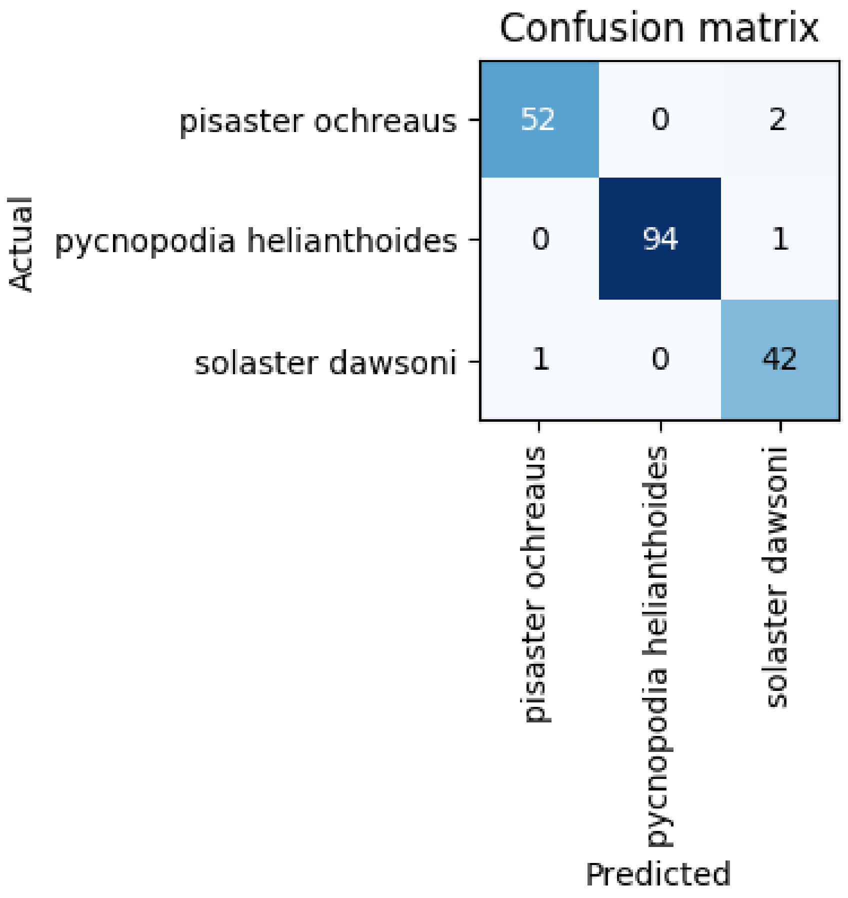

Figure 3: Confusion Matrix of actual vs predicted species from our model, deeper color represents data distribution amongst the species.

`Refer to #CODE BLOCKS 14,15,16,17 # on Jupyter Notebook`

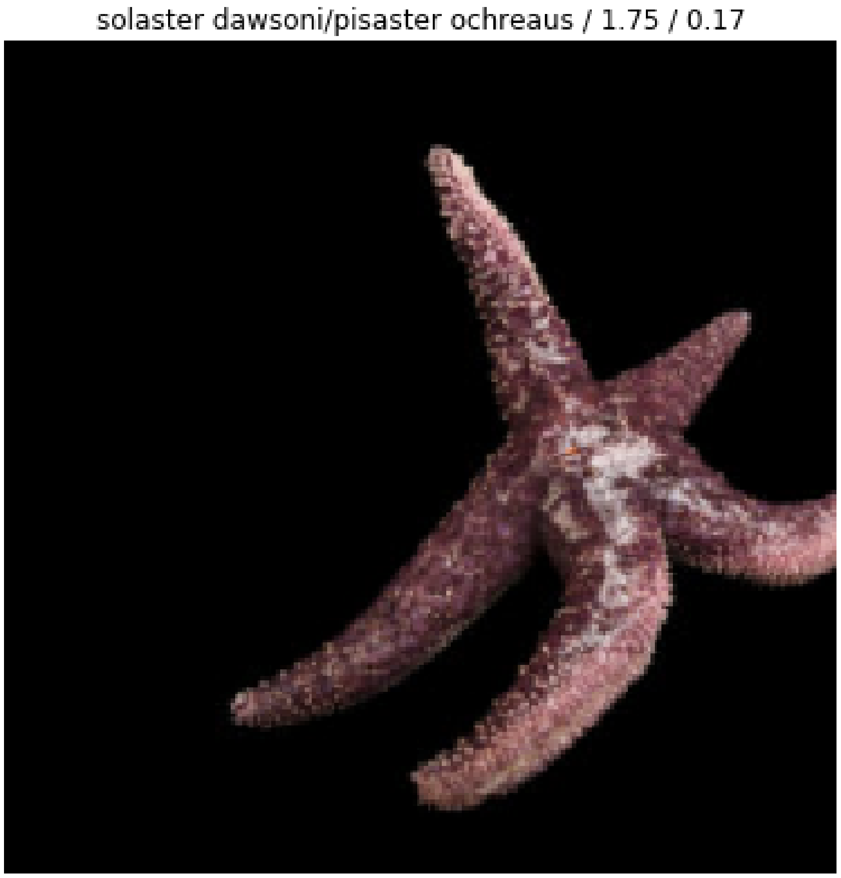

Figure 4: Example image where the model misidentified *Pisaster ochreaus* as *Solaster dawsoni.* Here the training loss is 1.75 and the probability that it is Solaster dawsoni is 0.17.

2.4

2.5 INFERENCE & MODEL DEPLOYMENT PHASE

2.5.1 *Model Inference*

As a test of model, we use images not used previously seen by the model. We use the built-in prediction tools in fast.ai as illustrated in the model inference section code of the development phase on the Jupyter Notebook. We were able to corretly classify an additional single image of *Pycnopodia helianthiodes* as an example.

```
Refer to #CODE BLOCKS 18,19, 20 # on Jupyter Notebook
```

```
Refer to #CODE BLOCKS 19,20 # on Jupyter Notebook
```

2.5.2 *Model Deployment*

In some cases, the user might want to develop an online application such as a website or application where users in the field, such as citizen scientists, can use a previously-developed model for real-time classification.

We illustrate how to build this workflow in the supplementary material.

3 Conclusion

Although there have been many papers prototyping interesting applications of artificial intelligence in life science particularly biology and ecology, very few outline beginner-friendly approaches to classifying species. To address this issue, we have illustrated the tools, steps, and resources required to build a image classifier. Despite the existence and use of these tools, it is when citizens and scientists alike have access to readily available cost-effective and intuitive tools that domain ecologists can start utilizing the potential of these powerful algorithms to solve and discover otherwise challenging problems. We hope this paper can help spur a new approaches to species classification. As a next step, we will use this methodology to build a more comprehensive sea star image classifier for the Western Coast of the United States. The classifier will be integrated into a phone application to allow citizen scientists to monitor both population counts and also sea star wasting disease.

4 Acknowledgments

The workflow is adapted from the Fast.ai course run by Jeremy Howard and Rachel Thomas of the Data Science Institute of the University of San Fransisco.

5 Supplementary Material

Hosted file Guide for online deployment1.pdf available at https://authorea.com/users/274158/articles/429253-rapid-prototyping-of-species-classifiers-using-deep-learning-a-guide-for-non-experts

References

. https://www.gbif.org/. Retrieved from https://www.gbif.org/

. https://www.inaturalist.org/. Retrieved from https://www.inaturalist.org/

Aide, T. M., Corrada-Bravo, C., Campos-Cerqueira, M., Milan, C., Vega, G., & Alvarez, R. (2013). Real-time bioacoustics monitoring and automated species identification. *PeerJ*, *1*, e103. doi:10.7717/peerj.103

Alber, M., Bello, I., Zoph, B., Kindermans, P.-J., Ramachandran, P., & Le, Q. (2018). Backprop Evolution.

Cantrell, B., Martin, L. J., & Ellis, E. C. (2017). Designing Autonomy: Opportunities for New Wildness in the Anthropocene. *Trends in Ecology & Evolution*, *32*(3), 156–166. doi:10.1016/j.tree.2016.12.004

Canziani, A., Paszke, A., & Culurciello, E. (2016). An Analysis of Deep Neural Network Models for Practical Applications.

Christin, S., Hervet, E., & Lecomte, N. (2019). Applications for deep learning in ecology. *Methods in Ecology and Evolution*, *10*(10), 1632–1644. doi:10.1111/2041-210x.13256

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Li, F. F. (2009). ImageNet: a Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 248–255). doi:10.1109/CVPR.2009.5206848

Ditria, E. M., Lopez-Marcano, S., Sievers, M. K., Jinks, E. L., Brown, C. J., & Connolly, R. M. (2019). Automating the analysis of fish abundance using object detection: optimising animal ecology with deep learning. doi:10.1101/805796

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, *542*, 115–118.

Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., ... Dean, J. (2019). A guide to deep learning in healthcare. *Nat Med*, *25*, 24–29.

Farley, S. S., Dawson, A., Goring, S. J., & Williams, J. W. (2018). Situating Ecology as a Big-Data Science: Current Advances Challenges, and Solutions. *BioScience*, *68*(8), 563–576. doi:10.1093/biosci/biy068

Golden, J. A. (2017). Deep Learning Algorithms for Detection of Lymph Node Metastases From Breast Cancer. *JAMA*, *318*(22), 2184. doi:10.1001/jama.2017.14580

Hampton, S. E., Strasser, C. A., Tewksbury, J. J., Gram, W. K., Budden, A. E., Batcheller, A. L., ... Porter, J. H. (2013). Big data and the future of ecology. *Frontiers in Ecology and the Environment*, *11*(3), 156–162. doi:10.1890/120103

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition.

Heaton, J. B., Polson, N. G., & Witte, J. H. (2016). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, *33*(1), 3–12. doi:10.1002/asmb.2209

Howard, J., & others. (2018). fastai. Making Neural Nets Uncool Again. GitHub.

Jiang, Y., Bosch, N., Baker, R., Paquette, L., Ocumpaugh, J., Andres, A., ... Biswas, G. (2018). Expert Feature-Engineering vs. Deep Neural Networks: Which Is Better for Sensor-Free Affect Detection? (pp. 198–211). doi:10.1007/978-3-319-93843-$1_1$5

Joly, A., Goëau, H., Glotin, H., Spampinato, C., Bonnet, P., Vellinga, W.-P., ... Müller, H. (2017). LifeCLEF 2017 Lab Overview: Multimedia Species Identification Challenges. In *Lecture Notes in Computer Science* (pp. 255–274). Springer International Publishing. doi:10.1007/978-3-319-65813-$1_2$4

Kayid, A., Khaled, Y., & Elmahdy, M. (2018). Performance of CPUs/GPUs for Deep Learning workloads. doi:10.13140/RG.2.2.22603.54563

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., ... Willing, C. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. In F. Loizides & B. Schmidt (Eds.), *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (pp. 87–90). IOS Press.

Kohavi, R. (2001). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection, *14*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. doi:10.1145/3065386

Lamba, A., Cassey, P., Segaran, R. R., & Koh, L. P. (2019). Deep learning for environmental conservation. *Current Biology*, *29*(19), R977–R982. doi:10.1016/j.cub.2019.08.016

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. doi:10.1038/nature14539

Miao, Z., Gaynor, K. M., Wang, J., Liu, Z., Muellerklein, O., Norouzzadeh, M. S., ... Getz, W. M. (2019). Insights and approaches using deep learning to classify wildlife. *Scientific Reports*, *9*(1). doi:10.1038/s41598-019-44565-w

Min, S., Lee, B., & Yoon, S. (2016). Deep learning in bioinformatics. *Briefings in Bioinformatics*, bbw068. doi:10.1093/bib/bbw068

Norouzzadeh, M. S., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M. S., Packer, C., & Clune, J. (2018). Automatically identifying counting, and describing wild animals in camera-trap images with deep learning. *Proceedings of the National Academy of Sciences*, *115*(25), E5716–E5725. doi:10.1073/pnas.1719367115

Olden, J. D., Lawler, J. J., & Poff, N. L. R. (2008). Machine Learning Methods Without Tears: A Primer for Ecologists. *The Quarterly Review of Biology*, *83*(2), 171–193. doi:10.1086/587826

Paine, R. T. (1966). Food Web Complexity and Species Diversity. *The American Naturalist*, *100*(910), 65–75. doi:10.1086/282400

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., Devito, Z., … Lerer, A. (2017). Automatic differentiation in PyTorch.

Randles, B. M., Pasquetto, I. V., Golshan, M. S., & Borgman, C. L. (2017). Using the Jupyter notebook as a tool for open science: An empirical study. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)* (pp. 1–2). IEEE.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks.* Cambridge University Press. doi:10.1017/CBO9780511812651

Roh, Y., Heo, G., & Whang, S. E. (2018). A Survey on Data Collection for Machine Learning: a Big Data – AI Integration Perspective.

Shen, D., Wu, G., & Suk, H.-I. (2017). Deep Learning in Medical Image Analysis. *Annual Review of Biomedical Engineering*, *19*(1), 221–248. doi:10.1146/annurev-bioeng-071516-044442

Shi, S., Wang, Q., Xu, P., & Chu, X. (2016). Benchmarking State-of-the-Art Deep Learning Software Tools.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1). doi:10.1186/s40537-019-0197-0

Talas, L., Fennell, J. G., Kjernsmo, K., Cuthill, I. C., Scott-Samuel, N. E., & Baddeley, R. J. (2019). CamoGAN: Evolving optimum camouflage with Generative Adversarial Networks. *Methods in Ecology and Evolution*, *11*(2), 240–247. doi:10.1111/2041-210x.13334

Terry, J. C. D., Roy, H. E., & August, T. A. (2020). Thinking like a naturalist: Enhancing computer vision of citizen science images by harnessing contextual data. *Methods in Ecology and Evolution*, *11*(2), 303–315. doi:10.1111/2041-210x.13335

Ting, K. M. (2017). Confusion Matrix. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning and Data Mining* (pp. 260–260). Boston, MA: Springer US. doi:10.1007/978-1-4899-7687-1_50

Valletta, J. J., Torney, C., Kings, M., Thornton, A., & Madden, J. (2017). Applications of machine learning in animal behaviour studies. *Animal Behaviour*, *124*, 203–220. doi:10.1016/j.anbehav.2016.12.005

Weinstein, B. G. (2017). A computer vision for animal ecology. *Journal of Animal Ecology*, *87*(3), 533–545. doi:10.1111/1365-2656.12780

White, E. R., & Bahlai, C. A. (2019). Experimenting with the past to improve environmental monitoring programs. *EcoEvoRxiv Preprints.* doi:10.32942/osf.io/cz5va

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks?. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27* (pp. 3320–3328). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf