Prototyping Artificial Intelligent Classifiers using Deep Learning

Herman Njoroge Chege[1]

[1]Affiliation not available

July 27, 2022

Abstract

Point 1: 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.

Point 2: 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. The prototype developer is publicly available and can be adapted for citizen science as well as other applications not envisioned in this paper.

Point 3: 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 superior accuracy. We include additional approaches to understand the misclassified images and to curate the dataset to increase accuracy.

Point 4: 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.

Keywords:

Artificial Intelligence, Deep Learning, Species Classification, Neural Network, Pattern Recognition, Big Data

Introduction

Deep learning, a branch of machine learning, is an artificial intelligence approach which has been used for pattern recognition across multiple domains (Shen et al.; Golden; Min et al.; Heaton et al.; Esteva et al.; Esteva et al.). Whereas other machine learning approaches have been used for acoustic classification (Aide et al.), ecological modelling and studying animal behaviour (Olden et al.; Valletta et al.; Christin et al.), 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 et al.; Christin et al.)(Deng et al.; Yosinski et al.). 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.).

In ecology, however, the use of deep learning is still in its infancy. 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 et al.; Lamba et al.; Miao et al.; Ditria et al.). Several research articles continue to implement new and interesting applications (Terry et al.; Talas et al.; Priyadarshani et al.). 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.

Ecology is particularly 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. 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; Norouzzadeh et al.). Deep learning is specifically better than other methods in dealing with non-linear complex data commonly encountered in ecology (Christin et al.). In fact, all winning methods for the most recent LifeCLEF contests have been deep learning-based (Joly et al.). Reviews and proposals for these have been put forward and the field feels right for disruption (Christin et al.; Lamba et al.). 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 et al.; Christin et al.).

A lot of the challenges that prevented deep learning from having practical applications have been eliminated with advancements research on transfer learning and data augmentation (Shorten and Khoshgoftaar). 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.

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. Therefore, to illustrate to non-experts how they can prototype these previously mysterious techniques this paper takes you step by step on the various stages and offers open-source code in form of an annotated Jupyter Notebook that can be used by anybody in the world to produce expert-level accuracy on whatever supervised species classification they want to carry out. 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`

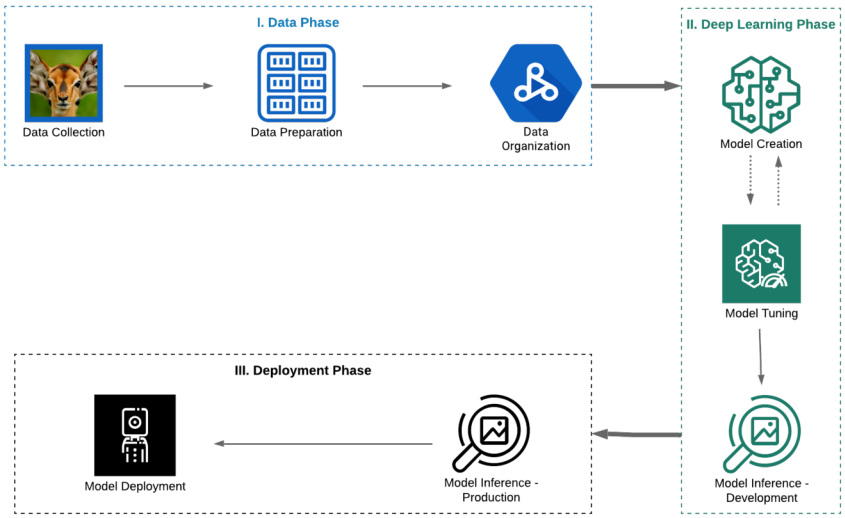

Figure 1: Overview of the various phases for prototyping of image data species classifiers using deep learning

LIBRARIES AND TOOLS

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.; Howard and others). 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.; Randles et al.).

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 et al.; Kayid et al.). 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.

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 et al.). 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*; *INaturalist*). 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. Other interesting methods beyond the scope of this study are unsupervised learning and reinforcement learning where the algorithm discovers the patterns in the data itself.

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). 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:

Implementation

I. PREPARATION

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*

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`

II. DATA PHASE

Importing Data

In this case study we use three sea star species: *Pisaster ochraceus*, *Pycnopodia helianthiodes* and *Solaster dawsonii*. We will simplify the exercise by organizing data in a format similar to this:

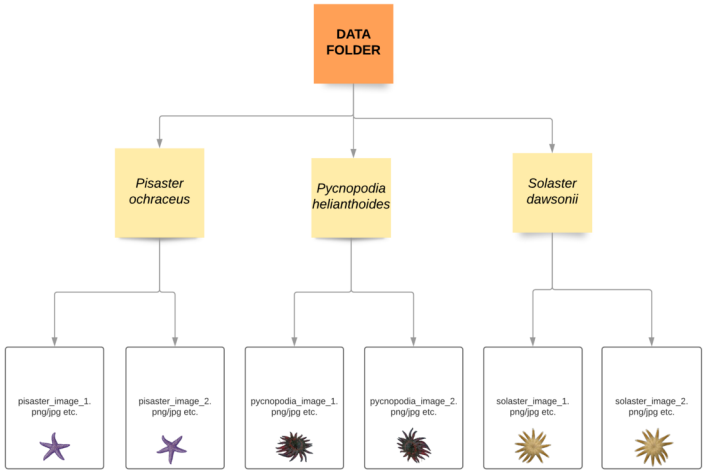

Figure 2: Data organization for the classification of three sea star species: *Pisaster ochraceus*, *Pycnopodia helianthiodes* and *Solaster dawsonii*

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.

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 sea star 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; Ripley).

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 visual of the images and their respective labels.

III. DEEP LEARNING PHASE

Deep Learning Model Creation

In our case study, we have a computer vision problem, so we will use Convolutional Neural Networks (CNN) (Krizhevsky et al.). In our case, we will use the RESNET 34 architecture that is not too complicated but delivers superior results compared to other architectures (He et al.). We use accuracy 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 that we want the model to run for. In our case, we run for 10 epochs and achieved an accuracy of about 85%.

`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. 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.`

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%.

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). 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).

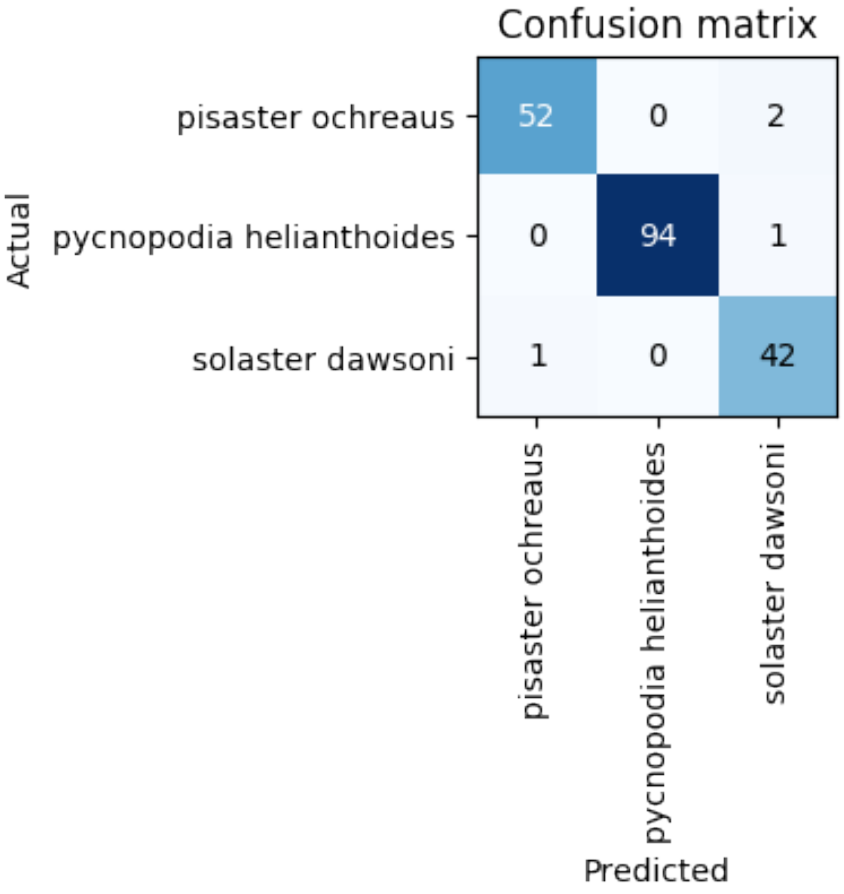

Figure 3: Confusion Matrix of actual vs predicted species from our model, deeper color represents data distribution amongst the species.

`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 loss is 1.75 and the probability is 0.17.

IV. INFERENCE & MODEL DEPLOYMENT PHASE

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
```

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.

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.

Acknowledgments

This work was partially supported by the NSF traineeship: Quantitative & Evolutionary STEM Training (QUEST): An Integrative Training Program for Versatile STEM Professionals to Solve Environmental and Global Health Problems of NSF Award Number: 1735316 and Prof. Melissa Pespeni's lab start up grant at the University of Vermont. Dr. Easton White and Prof. Nick Cheney provided input at the highest level. 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.

Supplementary Material

References

https://www.gbif.org/, https://www.gbif.org/.

https://www.inaturalist.org/, https://www.inaturalist.org/.

Aide, T. Mitchell, et al. "Real-Time Bioacoustics Monitoring and Automated Species Identification". *PeerJ*, vol. 1, PeerJ, July 2013, p. e103, doi:10.7717/peerj.103.

Cantrell, Bradley, et al. "Designing Autonomy: Opportunities for New Wildness in the Anthropocene". *Trends in Ecology & Evolution*, vol. 32, no. 3, Elsevier BV, March 2017, pp. 156–66, doi:10.1016/j.tree.2016.12.004.

Christin, Sylvain, et al. "Applications for Deep Learning in Ecology". *Methods in Ecology and Evolution*, edited by Hao Ye, vol. 10, no. 10, Wiley, July 2019, pp. 1632–44, doi:10.1111/2041-210x.13256.

Coleman, Cody A., et al. *DAWNBench : An End-to-End Deep Learning Benchmark and Competition.* 2017.

Deng, Jia, et al. "ImageNet: a Large-Scale Hierarchical Image Database". *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–55, doi:10.1109/CVPR.2009.5206848.

Ditria, Ellen M., et al. *Automating the Analysis of Fish Abundance Using Object Detection: Optimising Animal Ecology with Deep Learning.* Cold Spring Harbor Laboratory, October 2019, doi:10.1101/805796.

Esteva, A., et al. "A Guide to Deep Learning in Healthcare". *Nat Med*, vol. 25, 2019, pp. 24–29.

—. "Dermatologist-Level Classification of Skin Cancer with Deep Neural Networks". *Nature*, vol. 542, 2017, pp. 115–18.

Farley, Scott S., et al. "Situating Ecology as a Big-Data Science: Current Advances Challenges, and Solutions". *BioScience*, vol. 68, no. 8, Oxford University Press (OUP), July 2018, pp. 563–76, doi:10.1093/biosci/biy068.

Fukushima, Kunihiko, and Sei Miyake. "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition". *Competition and Cooperation in Neural Nets*, Springer Berlin Heidelberg, 1982, pp. 267–85, doi:10.1007/978-3-642-46466-9$_1$8.

Golden, Jeffrey Alan. "Deep Learning Algorithms for Detection of Lymph Node Metastases From Breast Cancer". *JAMA*, vol. 318, no. 22, American Medical Association (AMA), December 2017, p. 2184, doi:10.1001/jama.2017.14580.

Hairston, Nelson G., et al. "Community Structure Population Control, and Competition". *The American Naturalist*, vol. 94, no. 879, University of Chicago Press, November 1960, pp. 421–25, doi:10.1086/282146.

Hampton, Stephanie E., et al. "Big Data and the Future of Ecology". *Frontiers in Ecology and the Environment*, vol. 11, no. 3, Wiley, April 2013, pp. 156–62, doi:10.1890/120103.

He, Kaiming, et al. *Deep Residual Learning for Image Recognition.* 2015.

Heaton, J. B., et al. "Deep Learning for Finance: Deep Portfolios". *Applied Stochastic Models in Business and Industry*, vol. 33, no. 1, Wiley, October 2016, pp. 3–12, doi:10.1002/asmb.2209.

Hinton, Geoffrey, et al. "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups". *IEEE Signal Processing Magazine*, vol. 29, no. 6, Institute of Electrical and Electronics Engineers (IEEE), November 2012, pp. 82–97, doi:10.1109/msp.2012.2205597.

Howard, Jeremy, and others. *Fastai. Making Neural Nets Uncool Again.* GitHub, 2018, https://github.com/fastai/fastai.

Hubel, D. H., and T. N. Wiesel. "Receptive Fields and Functional Architecture of Monkey Striate Cortex". *The Journal of Physiology*, vol. 195, no. 1, Wiley, March 1968, pp. 215–43, doi:10.1113/jphysiol.1968.sp008455.

Jiang, Yang, et al. *Expert Feature-Engineering vs. Deep Neural Networks: Which Is Better for Sensor-Free Affect Detection?.* 2018, pp. 198–211, doi:10.1007/978-3-319-93843-1$_1$5.

Joly, Alexis, et al. "LifeCLEF 2017 Lab Overview: Multimedia Species Identification Challenges". *Lecture Notes in Computer Science*, Springer International Publishing, 2017, pp. 255–74, doi:10.1007/978-3-319-65813-1$_2$4.

Kayid, Amr, et al. *Performance of CPUs/GPUs for Deep Learning Workloads.* May 2018, doi:10.13140/RG.2.2.22603.54563.

Kluyver, Thomas, et al. "Jupyter Notebooks – a Publishing Format for Reproducible Computational Workflows". *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, edited by F. Loizides and B. Schmidt, IOS Press, 2016, pp. 87–90.

Kohavi, Ron. *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection.* March 2001.

Krizhevsky, Alex, et al. "ImageNet Classification with Deep Convolutional Neural Networks". *Communications of the ACM*, vol. 60, no. 6, Association for Computing Machinery (ACM), May 2017, pp. 84–90, doi:10.1145/3065386.

Lamba, Aakash, et al. "Deep Learning for Environmental Conservation". *Current Biology*, vol. 29, no. 19, Elsevier BV, October 2019, pp. R977–R982, doi:10.1016/j.cub.2019.08.016.

LeCun, Yann, et al. "Handwritten Digit Recognition with a Back-Propagation Network". *Advances in Neural Information Processing Systems 2*, edited by D. S. Touretzky, Morgan-Kaufmann, 1990, pp. 396–404, http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf.

—. "Deep Learning". *Nature*, vol. 521, no. 7553, Springer Science and Business Media LLC, May 2015, pp. 436–44, doi:10.1038/nature14539.

Miao, Zhongqi, et al. "Insights and Approaches Using Deep Learning to Classify Wildlife". *Scientific Reports*, vol. 9, no. 1, Springer Science and Business Media LLC, May 2019, doi:10.1038/s41598-019-44565-w.

Min, Seonwoo, et al. "Deep Learning in Bioinformatics". *Briefings in Bioinformatics*, Oxford University Press (OUP), July 2016, p. bbw068, doi:10.1093/bib/bbw068.

Mishra, Pradeepta. "Introduction to Neural Networks Using PyTorch". *PyTorch Recipes*, Apress, 2019, pp. 111–26, doi:10.1007/978-1-4842-4258-2_4.

Norouzzadeh, Mohammad Sadegh, et al. "Automatically Identifying Counting, and Describing Wild Animals in Camera-Trap Images with Deep Learning". *Proceedings of the National Academy of Sciences*, vol. 115, no. 25, Proceedings of the National Academy of Sciences, June 2018, pp. E5716–E5725, doi:10.1073/pnas.1719367115.

Olden, Julian D., et al. "Machine Learning Methods Without Tears: A Primer for Ecologists". *The Quarterly Review of Biology*, vol. 83, no. 2, University of Chicago Press, June 2008, pp. 171–93, doi:10.1086/587826.

Paine, Robert T. "Food Web Complexity and Species Diversity". *The American Naturalist*, vol. 100, no. 910, University of Chicago Press, January 1966, pp. 65–75, doi:10.1086/282400.

Paszke, Adam, et al. *Automatic Differentiation in PyTorch*. 2017.

Priyadarshani, Nirosha, et al. "Wavelet Filters for Automated Recognition of Birdsong in Long-Time Field Recordings". *Methods in Ecology and Evolution*, Wiley, January 2020, doi:10.1111/2041-210x.13357.

Randles, Bernadette M., et al. "Using the Jupyter Notebook as a Tool for Open Science: An Empirical Study". *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, IEEE, 2017, pp. 1–2.

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

Roh, Yuji, et al. *A Survey on Data Collection for Machine Learning: a Big Data – AI Integration Perspective*. 2018.

Russakovsky, Olga, et al. *ImageNet Large Scale Visual Recognition Challenge*. 2014.

Shen, Dinggang, et al. "Deep Learning in Medical Image Analysis". *Annual Review of Biomedical Engineering*, vol. 19, no. 1, Annual Reviews, June 2017, pp. 221–48, doi:10.1146/annurev-bioeng-071516-044442.

Shi, Shaohuai, et al. *Benchmarking State-of-the-Art Deep Learning Software Tools*. 2016.

Shorten, Connor, and Taghi M. Khoshgoftaar. "A Survey on Image Data Augmentation for Deep Learning". *Journal of Big Data*, vol. 6, no. 1, Springer Science and Business Media LLC, July 2019, doi:10.1186/s40537-019-0197-0.

Simonyan, Karen, and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014.

Talas, Laszlo, et al. "CamoGAN: Evolving Optimum Camouflage with Generative Adversarial Networks". *Methods in Ecology and Evolution*, edited by Graziella Iossa, vol. 11, no. 2, Wiley, December 2019, pp. 240–47, doi:10.1111/2041-210x.13334.

Tan, Chuanqi, et al. *A Survey on Deep Transfer Learning*. 2018.

Terry, J. Christopher D., et al. "Thinking like a Naturalist: Enhancing Computer Vision of Citizen Science Images by Harnessing Contextual Data". *Methods in Ecology and Evolution*, edited by Res Altwegg, vol. 11, no. 2, Wiley, January 2020, pp. 303–15, doi:10.1111/2041-210x.13335.

Ting, Kai Ming. "Confusion Matrix". *Encyclopedia of Machine Learning and Data Mining*, edited by Claude Sammut and Geoffrey I. Webb, Springer US, 2017, pp. 260–60, doi:10.1007/978-1-4899-7687-$1_5$0.

Valletta, John Joseph, et al. "Applications of Machine Learning in Animal Behaviour Studies". *Animal Behaviour*, vol. 124, Elsevier BV, February 2017, pp. 203–20, doi:10.1016/j.anbehav.2016.12.005.

Weinstein, Ben G. "A Computer Vision for Animal Ecology". *Journal of Animal Ecology*, edited by Laura Prugh, vol. 87, no. 3, Wiley, November 2017, pp. 533–45, doi:10.1111/1365-2656.12780.

Yosinski, Jason, et al. "How Transferable Are Features in Deep Neural Networks?". *Advances in Neural Information Processing Systems 27*, edited by Z. Ghahramani et al., Curran Associates, Inc., 2014, pp. 3320–28, http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf.

https://www.fast.ai/, https://www.fast.ai/.

Lecun, Y., et al. "Gradient-Based Learning Applied to Document Recognition". *Proceedings of the IEEE*, vol. 86, no. 11, November 1998, pp. 2278–324, doi:10.1109/5.726791.

Szegedy, C., et al. "Going Deeper with Convolutions". *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9, doi:10.1109/CVPR.2015.7298594.

Setting up a website to Access your Classifier

After confirming that your model works and correctly infers your species with reasonable accuracy, it is time to deploy it to a server or mobile app where you can be able to carry out inference from your model. Here i will illustrate how to make a web page where you can upload and classify images - more complicated user design and interface might need the services of a front end software engineer who are now abundant in the software engineering domain.

Depending on the complexity of the project there are many options to deploy including - Google App Engine, AWS Lambda, Amazon Sage Maker, AWS Elastic Beanstalk and Microsoft Azure Functions. There are also options to host it on your own servers for grand projects that might require dedicated devops engineers.

For our purpose we will deploy it to a pay as you go platform that can easily scale and is easy for anyone to setup. The link here leads to a page of how to do this by deploying your trained models to such a platform by the name - Render https://course.fast.ai/deployment_render.html. Below i explore the process step by step with an assumption of basic github skills.

1. Github configurations

The process involves forking a starter app from https://github.com/render-examples/fastai-v3. Signing up for a render account here https://dashboard.render.com/. Then uploading the trained model file created when you run `#CODE BLOCKS 18#` (it is located on your path and might have a name like `export.pkl`) to a cloud service like Google Drive or Dropbox. Then you can copy the download link for the file from these cloud services.

You can then go to the forked github folder above and edit the file `server.py` inside the `app` directory and update the `export_file_url` variable with the URL copied above from the cloud services. In the same file, you should update the line:

```
classes = ['Pisaster ochraceus', 'Pycnopodia helianthiodes','Solaster dawsonii']
```

with the classes, you expect from your model - in our case the three sea star species. You might be going back and forth by committing to your GitHub as you improve your model - Render integrates with your GitHub repo and automatically builds and deploys changes every time you push a change making the process seamless.

2. Configurations on Render

You can now go to Render and create a new **Web Service** and use the repo you created above. You will need to grant Render permission to access your Github repo in this step. On the deployment screen, pick a name for your service and use `Docker` for the Environment. The URL will be created using this service name. The service name can be changed if necessary, but the URL initially created can't be edited. Then click **Save Web Service**.

That’s it! Your service will begin building and should be live in a few minutes at the URL displayed in your Render dashboard. You can follow its progress in the deploy logs.

TESTING

Your app’s URL will look like https://seastar-classifier.onrender.com depending on the name of your service. You can also monitor the service logs as you test your app. You are now ready to go and start classifying species from real life photos that you might have.