Applications of Fast.ai Pretrained Models in Image Classification Problem

Antonio Ahmed Tapia Maldonado
Master in Computer Engineering
Dr. Jeffrey Duffany
Computer Science Department
Polytechnic University of Puerto Rico

Abstract *— Since the inception of* Transformers *and GPTs, artificial intelligence has proliferated. Fast.ai is among the cutting-edge libraries that are leading new advancements in the field. To harness the power of fast.ai and other advancements in the field, we set out to try and evaluate the practicality of the fast.ai library. To achieve this, we choose a given use case for artificial intelligence and then set out to fulfill said use case by leveraging fast.ai. We created three image classification models through fast.ai and then made an application that used those models. The use case we chose was a local wildlife fauna and flora classifier. The results from training the models were models with meager error rates, and these models had little to no data engineering.*

Key Terms *— computer vision, deep learning, fast.ai, image classification*

INTRODUCTION

Fast.ai is a deep learning library that provides practitioners with high-level components that can quickly and easily provide state-of-the-art results in standard deep learning domains, and provides researchers with low-level components that can be mixed and matched to build new approaches [1].

One of the most cutting-edge libraries in Machine Learning is fast.ai. This library includes very sophisticated pre-trained models and tools for fine-tuning them. It also contains the tools to leverage graphical processing units (GPUs) to accelerate the fine-tuning and inference processes.

To our interest, the fast.ai library also has a set of tools to facilitate the implementation of deep-learning neural networks. These deep learning networks come bundled with all the knowledge needed to be fine-tuned into the problem because these networks have already been trained with state-of-the-art hardware. Therefore, they already have a lot of knowledge embedded in them. Note that deep neural networks are defined as those with three or more hidden layers.

Thus, our purpose for this project was to develop a system that included deep learning models to classify flora and fauna in Puerto Rico. We leveraged fast.ai and Jupyter notebooks to train various deep-learning models.

We also developed a React Native application to facilitate testing the model's inferences process. For the application's back-end, the trained models with the fast.ai library were deployed to an inference webserver called TorchServe. To work with TorchServe, we created MAR files and uploaded them to a TorchServe server running on a Microsoft Azure Virtual Machine.

METHODOLOGY

Fast.ai, Image Classification, and Getting the Labeled Image Dataset

The fast.ai library also comes with pre-trained image classification models that facilitate the development of computer vision models. After a preliminary evaluation of all the pre-trained models, we chose the ResNet-50 model, which provided the lowest error percentage for a limited test of three epochs and a preliminary image size of 150 px x 150 px.

Intending to improve the accuracy output of the AI models, we decided to create three separate models instead of training one model to classify among all categories. We also limited the species' categories in each model's output to nine, which should have sufficient scope.

To obtain a set of labeled images, we utilized the Bing Image Search API. These images are what the image model will use to learn how to classify the different images. To feed the images into the model, all images were resized to fit a square of 250 px by 250 px.

Our objective was for the model to learn to discern the most common features of each species type. We removed some of the labeled images we received from Bing Image Search. Specifically, we removed images with the following characteristics:

- Two or more plants in the same image
- Collages
- Sprouts or undeveloped specimens (we are only interested in fully grown specimens)
- Drawings and illustrations
- Images with people or other things not related to the species
- Non-JPG format images
- Non-full color images

Training an Image Classification Model with Fast.Ai

We can obtain the table with much information regarding how the model would result from training an image classification model. One such field is the error rate, which tells us the percentage of errors while testing the model of the test dataset. We also get the time it took to train the model and epoch, which tells us which iteration on the fine-tuning where the results are obtained.

Another output from training the models is the confusion matrices, which tells us exactly how many predictions were wrong for that model. With a confusion matrix, we can see if our model needs help differentiating among categories.

Training Image Classification Model of Trees in Puerto Rico

Our application of image classifications is helping tourists identify trees in the rainforest of Puerto Rico. To do so, a set of labeled images must be gathered with the most common trees on the island, according to [2].

We picked nine of the most common trees from the list: *Artocarpus altilis*, *Bursera simaruba*, *Carica papaya*, *Coccoloba uvifera*, *Delonix regia*, *Mangifera indica*, *Tabebuia heterophylla*, *Terminalia catappa*, and *Thespesia populnea*.

After cleaning the dataset, we utilized a randomly generated image split to train and then tested the training results. From the training, we obtained a model with the characteristics shown on tables 1 and 2.

Table 1

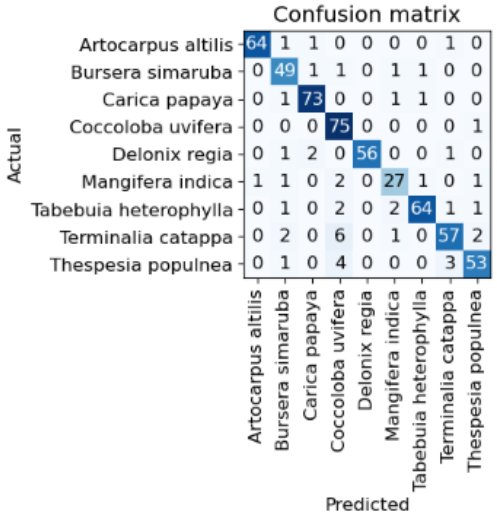
Fast.ai image tree image classifier model training results

epoch	train_loss	valid_loss	error_rate	time
0	2.107763	0.958302	0.261947	04:27

epoch	train_loss	valid_loss	error_rate	time
0	1.093191	0.593663	0.173451	06:26
1	0.824228	0.549795	0.166372	05:20
2	0.648223	0.613604	0.148673	04:57
3	0.518046	0.519078	0.138053	05:04
4	0.393416	0.487118	0.115044	04:57
5	0.304550	0.433191	0.109735	04:57
6	0.234228	0.372403	0.084956	05:16
7	0.169797	0.380457	0.086726	05:30
8	0.126345	0.361392	0.081416	05:53
9	0.109953	0.362511	0.083186	06:19

Figure 2

Fast.ai image tree image classifier model confusion matrix.



Results of Image Classification Models of Trees in Puerto Rico

To manually test the inference capabilities of the model, we utilized a labeled image of the tree species. Below are the pictures and the results.

Figure 1 was fed into the tree classification model, and the result was a correct prediction of the species *Artocarpus altilis* (figure 2) [3].

Figure 1

Labeled image of *Artocarpus altilis* [3]

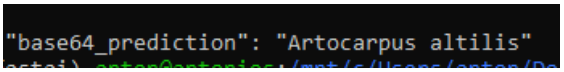


Figure 2

Result from inference server running on TorchServe to infer figure 1

Another example of a previously labeled image is figure 3, for *Coccoloba uvifera* [4]. Figure 4 shows the results of the inference from the model.

Figure 3

Labeled image of *Coccoloba uvifera*

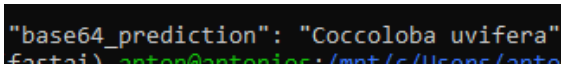


Figure 4

Result from inference server running on TorchServe to infer figure 3

Training Image Classifying Models of Fungi in Puerto Rico

One application of image classifications is helping tourists identify fungi in Puerto Rico. We picked nine of the most common fungi from [5]: false parasol (*Chlorophyllum molybdites*), hairy hexagonia, fringed sawgill (*Lentinus crinitus*), *Cookeina tricholoma*, fairy inkcap (*Coprinellus disseminatus*), *Leiotrametes lactinea*, split gill (*Shizophyllum commune*), magic mushroom (*Psilocybe cubensis*), and *Earliella scabrosa*.

After cleaning the dataset, we utilized a randomly generated image split to train and then tested the training results. From the training, we obtained a model with the characteristics shown on tables 3 and 4.

Table 3

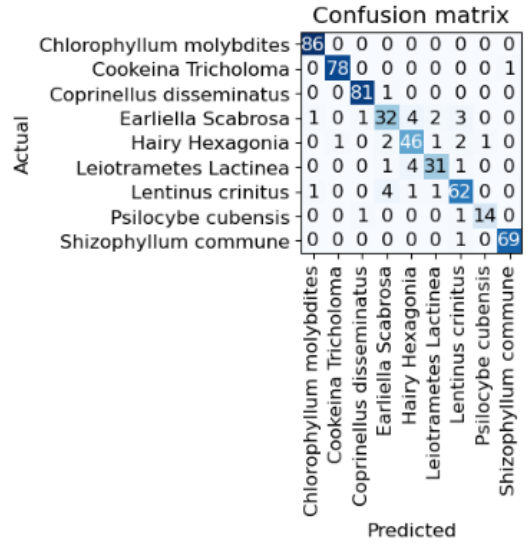
Fast.ai image fungi image classifier model training results

epoch	train_loss	valid_loss	error_rate	time
0	1.584525	0.629121	0.149533	04:16

epoch	train_loss	valid_loss	error_rate	time
0	0.658967	0.515569	0.140187	05:03
1	0.534303	0.417054	0.112150	04:52
2	0.439563	0.452863	0.119626	04:49
3	0.370225	0.387200	0.093458	05:06
4	0.302951	0.382546	0.097196	05:02
5	0.222379	0.373214	0.089720	05:17
6	0.164840	0.347773	0.085981	05:17
7	0.127442	0.295489	0.074766	05:18
8	0.099263	0.305497	0.071028	05:17
9	0.073556	0.311110	0.067290	05:18

Table 4

Fast.ai image fungi image classifier model confusion matrix



Results of Image Classifying Models of Fungi in Puerto Rico

From the previous model, we submitted figure 5 [6] to see if it would be classified correctly. The image was fed into the fungi classification model, and the result was a correct prediction of the *Chlorophyllum molybdite* species (figure 6).

Figure 5

Labeled image of *Chlorophyllum molybdites*

```
"base64_prediction": "Chlorophyllum molybdites"
```

Figure 6

Result from inference server running on TorchServe to infer figure 5

Another example of a previously labeled image was one of *Cookeina tricholoma* (figure 7) [7]. Figure 8 shows the results of the inference from the model.

Figure 7

Labeled image of *Cookeina tricholoma*

```
"base64_prediction": "Cookeina Tricholoma"
```

Figure 8

Result from inference server running on TorchServe to infer figure 7

Training Image Classifying Models of Fauna in Puerto Rico

One application of image classifications is helping tourists identify animals that are native to Puerto Rico. We picked nine of the most iconic animals on the island from [8]: *Anolis evermanni*, *Bufo marinus*, *Chilabothrus inornatus*, *Cyclura stejneger*, Dermochelys coriacea, *Eleutherodactylus juanariveroi*, Rhesus macaques, *Riccordia maugaeus*, and *Trichechus manatus*.

After cleaning the dataset, we utilized a randomly generated image split to train and then tested the training results. From the training, we obtained a model with the characteristics shown on tables 5 and 6.

Table 5

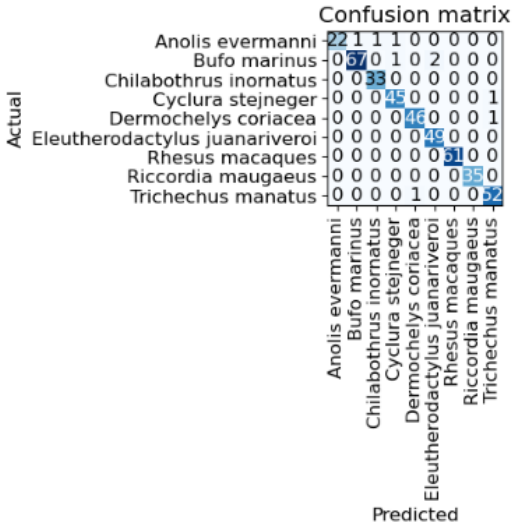
Fast.ai image fauna image classifier model training results

epoch	train_loss	valid_loss	error_rate	time
0	0.888312	0.115969	0.028640	03:41

epoch	train_loss	valid_loss	error_rate	time
0	0.180658	0.091292	0.023866	04:55
1	0.144018	0.096881	0.026253	04:27
2	0.135816	0.101307	0.026253	04:38
3	0.135237	0.184823	0.042959	04:38
4	0.116133	0.140800	0.038186	04:48
5	0.101545	0.145114	0.040573	04:25
6	0.074477	0.108284	0.021480	04:21
7	0.057583	0.098347	0.026253	04:17
8	0.042225	0.096191	0.023866	04:16
9	0.036369	0.096404	0.021480	09:26

Table 6

Fast.ai image fauna image classifier model confusion matrix



Results of Image Classifying Models of Fauna in Puerto Rico

From the previous model, we submitted the figure 7 [9] to see if it would be classified correctly.

Figure 7

Labeled image of *Bufo marinus*

The image was fed into the fauna classification model, and the result was a correct prediction of the *Bufo marinus* species (figure 8).

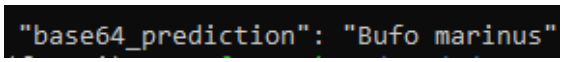


Figure 8

Result from inference server running on TorchServe to infer figure 7

Another example of a previously labeled image was one of *Eleutherodactylus juanariveroi* (figure 9) [10]. Figure 10 shows the results of the inference from the model.

Figure 9

Labeled image of *Eleutherodactylus juanariveroi*

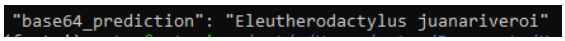


Figure 10

Result from inference server running on TorchServe to infer figure 9

Errors in Classification

The models, as was shown by the confusion matrices, did not have a one hundred percent

accuracy in their predictions. So, we cannot say that the models learned how to classify all characteristics of a given species, which is an erroneous prediction (figure 11).

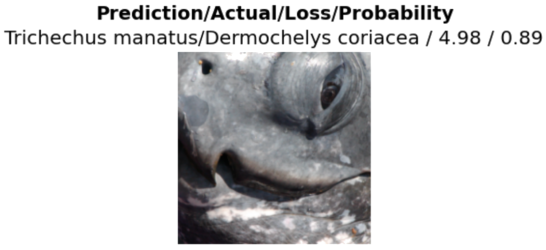


Figure 11
Erroneous classification from the fauna model

We can also say that the models struggle to classify species from a distance. It also needs help when they are not the image's central focus (figure 12).

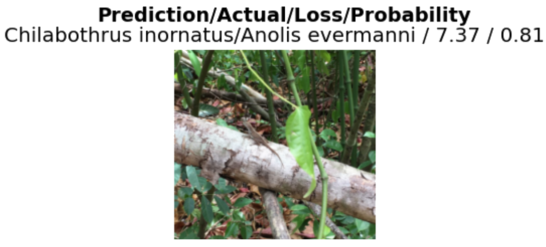


Figure 12
Erroneous classification from the fauna model

The models also struggle with images of the animals hidden in their natural habitat, where the manati is hidden underwater (figure 13)

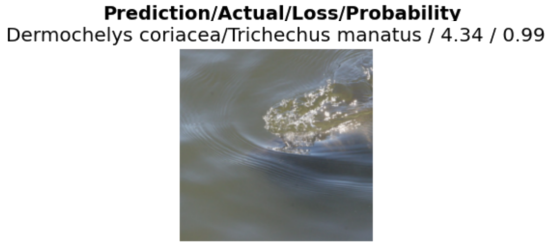


Figure 13
Erroneous classification from the fauna model

We note that these are severe problems with the application, and a larger dataset is needed to evaluate the shortcomings of the models trained with fast.ai. However, that is beyond the scope of this paper with our current dataset. From the results, we can say the AI learned enough to classify clear images of each species similar to the photos in our dataset.

Serving Models with TorchServe

TorchServe is a highly scalable and efficient web server specialized in serving inferences from exported machine-learning models. It functions as an easy way to serve inferences from the web. The model was deployed to a virtual machine on the Microsoft Azure Cloud.

TorchServe receives packaged files as input in the MAR file format. To generate the MAR files, we utilized the program Torch Model Archiver, which received the weights file from training the model, a handler for TorchServe, and a model class that contained the architecture for reconstructing the model. We utilized this program to generate the MAR files that TorchServe uses.

TorchServe automatically resizes all images it receives to 250 px by 250 px and then makes an inference on the image. This is because we configured the MAR file handler to resize the photos and how to send back the inferences.

The React Native Client App

To facilitate the process of submitting images for classification, a React Native client app was developed. It consisted of a simple client holding the inferences downloaded from the TorchServe server. It came with a login page to get the credentials to access the TorchServe API. This login screen was not used in this project, but it was added in case security is ever added to the app (figure 17).

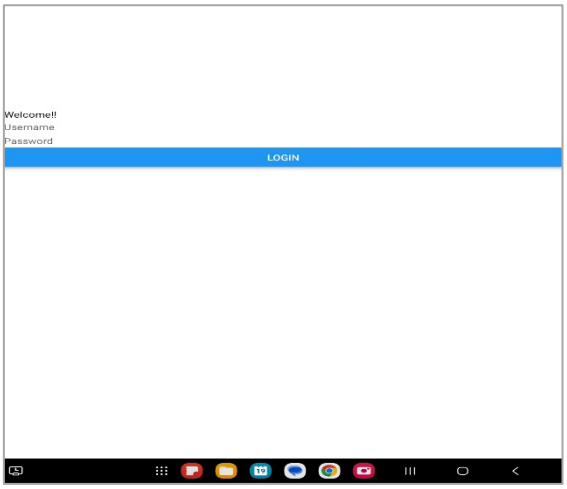


Figure 14
Login screen for React Native application

The main menu screen (figure 15) was designed so that users could snap a quick picture while in the field. It also gives the option to upload a previously taken picture by clicking the upload an image option.

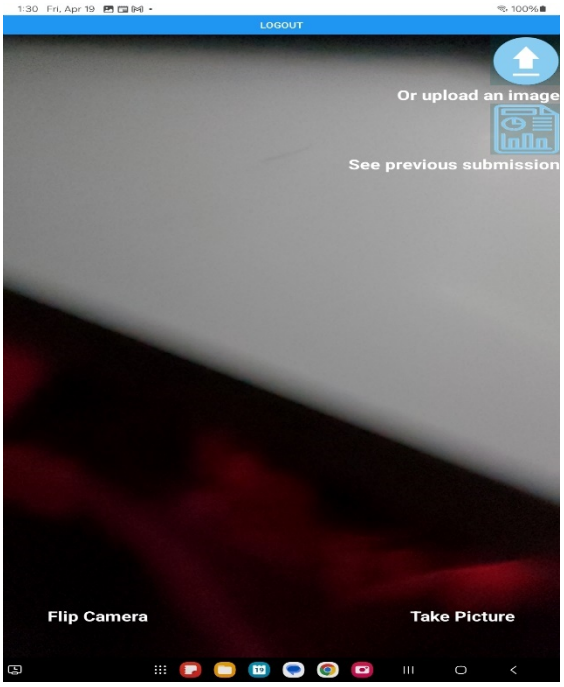


Figure 15
Main menu screen for the React Native application

Alternatively, in case an user wishes to see previous results from previous submissions, they may click on the "See previous submissions" button to see the current report with all previous submissions. Alternatively, users may select which camera (front-facing or back-facing) they want to use in the Android device by clicking on the "Flip Camera" button.

On the Image Select screen (figure 16), users may either select one image by tapping on it or, by long-pressing, choose multiple photos to select for classification.

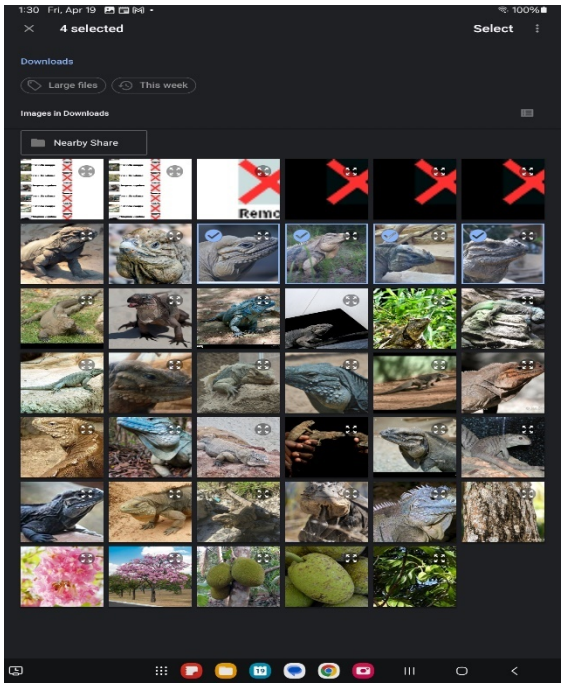


Figure 16
Image Select screen

Users may enter the image submission screen (figure 17) from the Image Select screen, where they may select which of the selected images you will submit to the TorchServe server. This TorchServe server is web-based. It is hosted in an Azure Virtual Machine. Users also get three separate inference models at the bottom of the image selection.

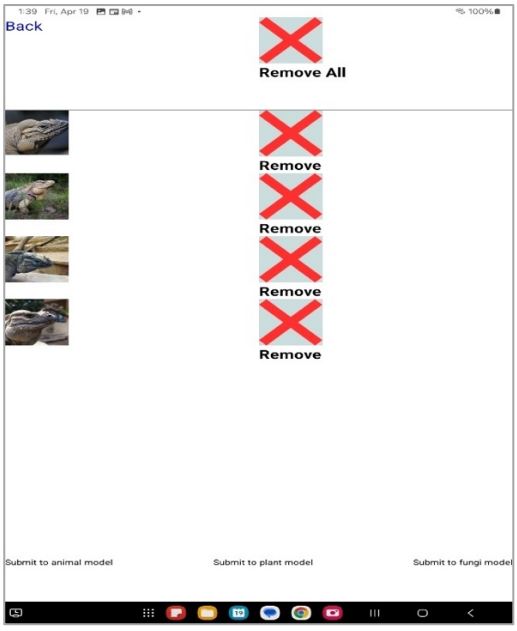


Figure 17
Image submission screen

From the image submission screen, users may click on one of the bottoms below the images to select an inference model to submit the selected images (figure 18).

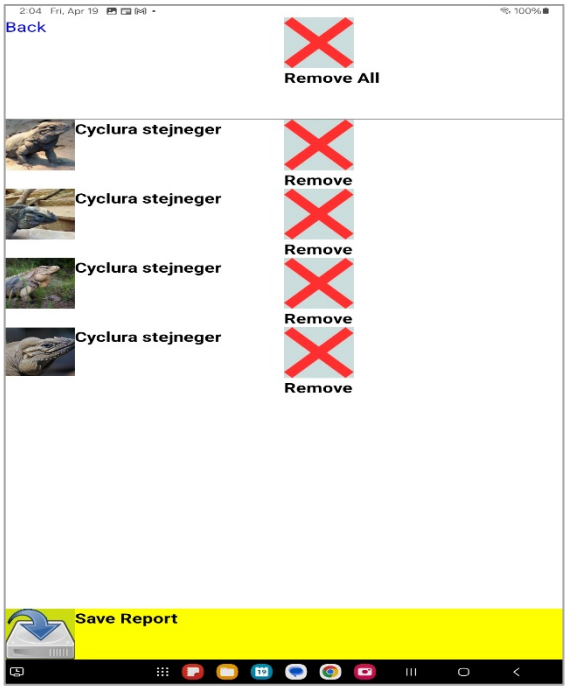


Figure 18
Report screen

After the user submits the images to the inference model on the web, they will receive a classification of the image. This forms what is called a report of all the classified images. This report may be saved on the downloads folder of the Android device.

The user may save it by pressing on the yellow button at the bottom of the report page. The user may even delete individual inferences if they want to delete only a few of the inferences from the report. The user may also delete all inferences on the report by clicking on the "Remove All" button on the top right of the report screen.

Figure 19 shows a saved report. It will be in JPG format and contain all the report's images and their inferences.

Figure 19
Full report saved

RESULTS AND DISCUSSION

From the model standpoint, we obtained a correct prediction from each of the three models on both images we obtained to infer. This tells us that fast.ai and pre-trained models are comparable to specific models, which would require much more time and data engineering to make.

With TorchServe, we succeeded in deploying our three models into a highly efficient inference-serving web server. We also successfully managed to generate a MAR file for each of our models.

The React Native app allows us to successfully upload and classify images and store the results. The app is ideal for submitting a massive amount of pictures into an inference model.

CONCLUSION

The impact of fast.ai and other pretrained libraries like Hugging Face should be significant. Pre-trained models require little to no expertise in machine learning, making it almost trivial to train a model. With tools like TorchServe to serve our models, integrating machine learning into our applications is now easier. Image classification may be done with little to no data engineering.

REFERENCES

[1] fast.ai. (2019). Fast.ai. https://docs.fast.ai/

[2] Picture This, “Top 20 most common trees in Puerto Rico.” Accessed May 12, 2024. Available: https://www.picturethisai.com/region/tree/Puerto-Rico-Puerto-Rico.html

[3] M. S. Sikarwar, B. J. Hui, K. Subramaniam, B. D. Valeisamy, L. K. Yean, and K. Balaji, A review on *Artocarpus altilis* (Parkinson) Fosberg (breadfruit),” *Journal of Applied Pharmaceutical Science*, vol. 4, no. 08, pp. 91–97. August 2014. Available: DOI: 10.7324/JAPS.2014.40818

[4] M. Calderón-Santoyo, E. M. González-Cruz, M. Íñiguez-Moreno, O. Ramos-Martínez, A. Burgos-Hernández, and J. A. Ragazzo-Sánchez. “Microencapsulation of phenolic extract from sea grape (Coccoloba uvifera L.) with antimutagenic activity,” *Chemistry and Biodiversity*, vol. 19, no. 11. November 2022. Available: DOI: 10.1002/cbdv.202200806

[5] Picure Mushroom, “Top 20 most common mushrooms in Puerto Rico.” Accessed May 12, 2024. Available: https://picturemushroom.com/region/Puerto-Rico-Country.html

[6] J. Wu et al., “Bioactive compounds from the mushroom-forming fungus *Chlorophyllum molybdites*,” *Antibiotics*, vol. 12, no. 3, p. 596. Available: DOI: 10.3390/antibiotics12030596

[7] R. Hermawan, I. P. Putra, and M. P. Amelya, “*Cookeina tricholoma* of West Java (Indonesia) based on morphological and molecular identification,” *Philippine Journal of Science*, vol. 151, no. 5, pp. 1807–1812. Available: DOI: 10.56899/151.05.22

[8] Puerto Rico Activities, “Puerto Rico wildlife: 9 animals you can see in El Yunque National Forest.” Accessed May 12, 2024. Available: https://puertoricoactivities.com/blog/el-yunque-national-forest-animals/

[9] National Invasive Species Information Center, “Cane toad.” Accessed May 12, 2024. Available: https://www.invasivespeciesinfo.gov/aquatic/fish-and-other-vertebrates/cane-toad

[10] U. S. Fish and Wildlife Service, “Llanero coqui.” Accessed April 17, 2024. Available: https://www.fws.gov/species/llanero-coqui-eleutherodactylus-juanariveroi?aggregated_content_type=%5B%22Image%22%5D