Jimmy Kropp

Minwoo Lee

ITCS 5156 - 051

15 December 2021

The Various Methods of Animal Recognition

Endangered species have been growing in concern over the years. With human activity increasing on the planet, extinction of species has become 1,000 to 10,000 times faster because of activities such as deforestation, overhunting, overfishing, and nitrogen pollution. If an unbalance of species is created, it will cause ecosystems to fall behind and impure environments to arise. This could lead to harm to the human ecosystem, causing harm to our population.

Through machine learning, we will be able to record animal population densities in the wild, and more importantly, keep track of endangered species. This will help researchers keep records of thriving populations, and adversely, declining populations. With the data collected and transformed, efficiency in the field will be raised due to more accurate information about animal populations and movements.

To approach this challenge, I created a neural network using a dataset I built that included animals such as cats, cows, dogs, elephants, and squirrels. I did not create a dataset of endangered animals because I wanted to see how the model I created handled random data. If the model could handle the random data, then it can handle more advanced data.

When I was researching about my project, I came across some related works that could be useful to me. The first research paper was about using a CNN for animal recognition through SVM. The first step was to preprocess the data, making sure that the effect of external factors in the images were minimized. This step also included building the predictive model from the training data that was created. Using various models for image classification such as PCA, LDA, SVM, and LBPH, they were able to obtain different results. These results were used to determine which model worked best for their use case (Why Endangered Species Matter).

The second research paper was about a kNN supervising model detecting animals across a road in real time (Animal Recognition System Based on Convolutional Neural Network). First, a server was set up using a Raspberry pi, which was booted up in wait of client requests. Once an animal is detected, the client would send the server a message informing that it found an animal. The server would return that it received the message, and in turn would send a message to a signal on the road that would light up with "STOP" so drivers would stop as to not hit the animal. The thread is then closed by the server, and the connection is closed.

The pros of these articles are that they both identify animals for my use case, and I was able to come up with methods similar to what was presented to create my own model. The first research paper was helpful in outlining the process that they went through in order to train the data and have highly accurate results.

There were also cons to these articles as well. I could not use all of the second research paper because it was identifying animals using a server and client system, but I was able to use part of the video detection. In my own research, I used images, but it was interesting to see how the model might work with video. Even though the first article was helpful in describing the process, it had some weaknesses to it. There was no repository of code attached to it, so I could not see the full functionality of what the model they created was capable of.

The first approach that I took was to create my own model. For this part of the project, I created a CNN using Keras that was able to intake a dataset of my creation and train the model from that. The training data consisted of cats, cows, dogs, elephants, and squirrels. As I mentioned before, these are random animals that are different enough to maybe give the model a hard time. If this model produced a high accuracy, I could potentially use it on more images of endangered species. I created the categories that the test images would be separated into once the model was trained, and after that I created the model. I created the model with 3 hidden layers, each layer containing a Conv2D, BatchNormalization, MaxPooling2D, and a Dropout of 25%. The final activation was a logsoftmax that would have a cross entropy loss and an rmspop optimizer. With this I was able to produce the output below (Fig. 1).

My second approach was to use the Oxford-III Pets database and train a CNN that would shrink the images to all the same size and decipher if the image was a cat or a dog. There was no dataset that I was able to find that had a training and testing set that was not for cats and dogs in the pytorch docs. I was going to use the dataset that I had created for the first approach, but when I tried pip installing a package to my environment, it broke the whole thing and I was not able to fix it. With the initial set having only the preprocessing of image shrink, I tested the images against a basic model with 3 epochs (Fig. 2). Even with this small number of epochs, the output had a decently high accuracy. The next step I took was to grayscale the images, from a suggestion from the presentation of the project. For this, the preprocessing steps I took were to shrink the images to 128 and make all of the images grayscale. Comparing the results was interesting, as I assumed that the images being grayscale would have a higher accuracy. The accuracy was not too much lower than the original dataset, but it still surprised me.

The third approach I took was using transfer learning to compare the results of the same dataset against different ResNets, different loss functions, and different optimizers. For the first 2 stages, I used resnet34, which is a middle class ResNet, both with the Adam optimization algorithm (Fig. 3 and Fig. 4). The loss functions for these, however, were different. The first stage used the negative log likelihood loss function, while the second used cross entropy loss function. Both stages had very high accuracies against the test data, but the cross entropy loss function had a slightly high accuracy (Fig. 5 and Fig. 6). The next 2 stages were using using resnet18 (Fig. 7 and Fig. 8). With the same loss functions and optimizer, the results came out very similar as well, with the accuracies coming out the same at about 95% accuracy. The 2 stages after that went back to using resnet34, but instead these both used the SGD optimizer (Fig. 9 and Fig. 10). The accuracies for these models were way lower than I was expecting, they were also lower than the previous accuracies. However, between the two models, the transfer learning model with resnet34, cross entropy loss, and an SGD optimizer had a higher accuracy. The last 2 stages consisted of using resnet18 with their respective loss functions, but with the SGD optimizer (Fig. 11 and Fig. 12). The transfer learning algorithm with resnet18, NLLLoss, and SGD optimizer just managed to inch out higher than the model using the cross entropy loss. Compared to the other models that were run, these 2 stages lay in the middle. The most accurate model using transfer learning was the model using resnet18, cross entropy loss, and the Adam optimizer. If I were to take a guess, I would have assumed that one of the models using resnet34 would have been more accurate because it was one of the higher ResNets.

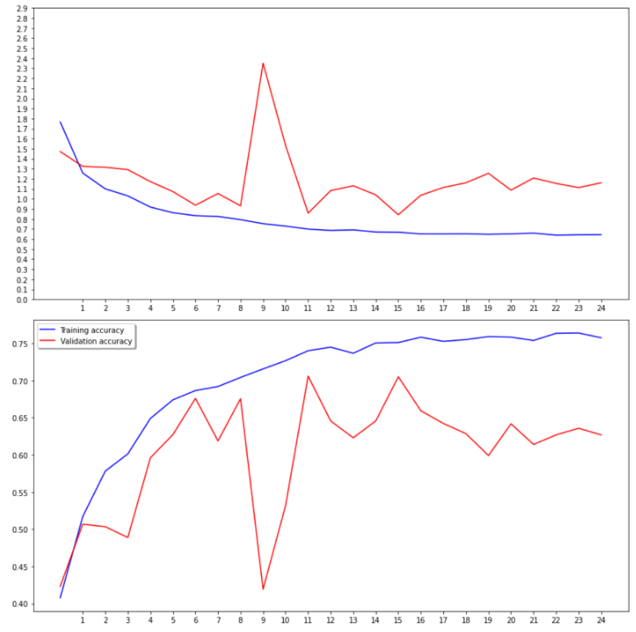
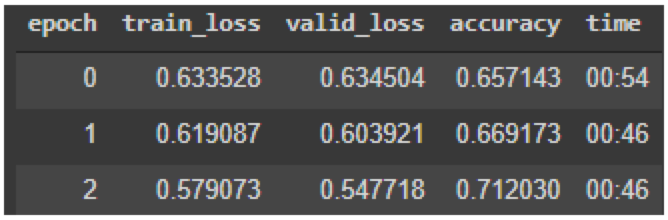


Fig. 1



epoch	train_loss	valid_loss	accuracy	time
0	0.633528	0.634504	0.657143	00:54
1	0.619087	0.603921	0.669173	00:46
2	0.579073	0.547718	0.712030	00:46

Fig. 2

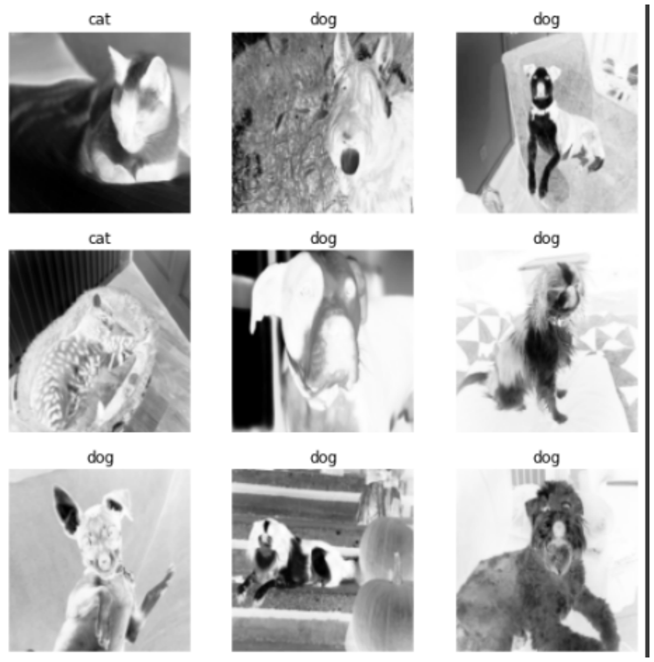
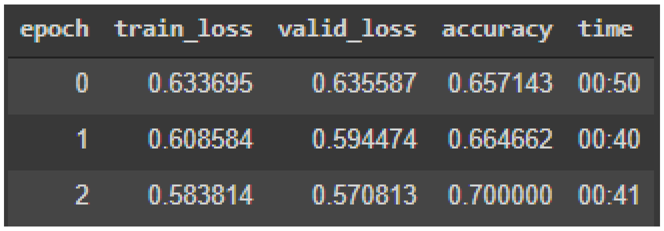
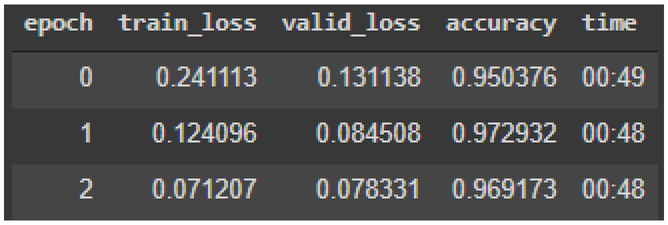


Fig. 3



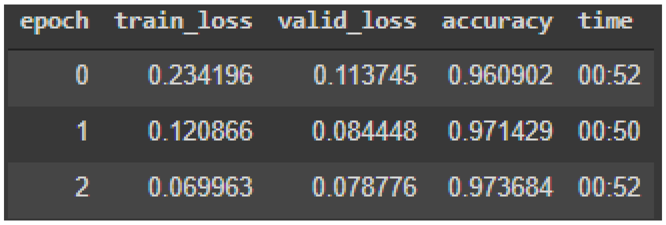
epoch	train_loss	valid_loss	accuracy	time
0	0.633695	0.635587	0.657143	00:50
1	0.608584	0.594474	0.664662	00:40
2	0.583814	0.570813	0.700000	00:41

Fig. 4



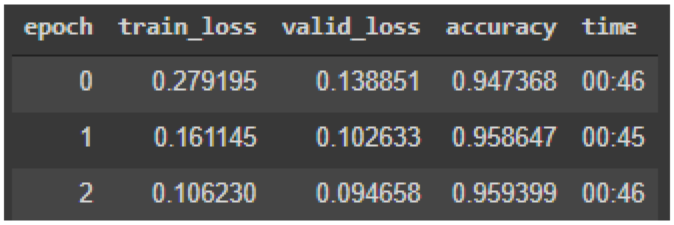
epoch	train_loss	valid_loss	accuracy	time
0	0.241113	0.131138	0.950376	00:49
1	0.124096	0.084508	0.972932	00:48
2	0.071207	0.078331	0.969173	00:48

Fig. 5



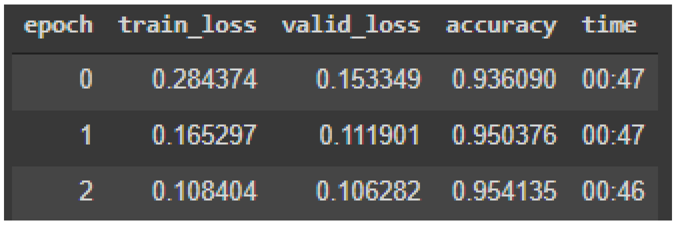
epoch	train_loss	valid_loss	accuracy	time
0	0.234196	0.113745	0.960902	00:52
1	0.120866	0.084448	0.971429	00:50
2	0.069963	0.078776	0.973684	00:52

Fig. 6



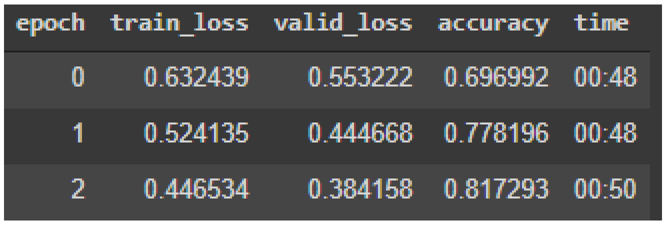
epoch	train_loss	valid_loss	accuracy	time
0	0.279195	0.138851	0.947368	00:46
1	0.161145	0.102633	0.958647	00:45
2	0.106230	0.094658	0.959399	00:46

Fig. 7



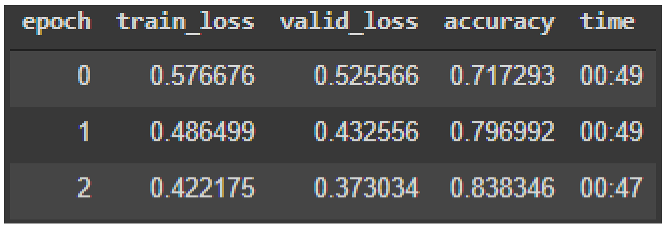
epoch	train_loss	valid_loss	accuracy	time
0	0.284374	0.153349	0.936090	00:47
1	0.165297	0.111901	0.950376	00:47
2	0.108404	0.106282	0.954135	00:46

Fig. 8



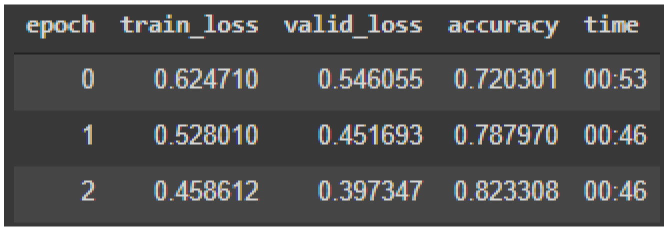
epoch	train_loss	valid_loss	accuracy	time
0	0.632439	0.553222	0.696992	00:48
1	0.524135	0.444668	0.778196	00:48
2	0.446534	0.384158	0.817293	00:50

Fig. 9



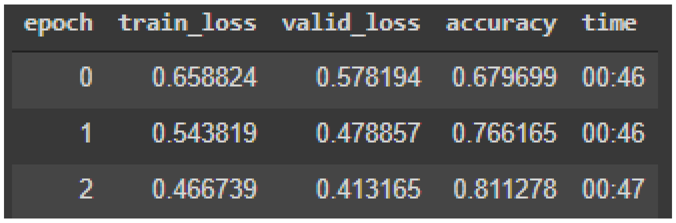
epoch	train_loss	valid_loss	accuracy	time
0	0.576676	0.525566	0.717293	00:49
1	0.486499	0.432556	0.796992	00:49
2	0.422175	0.373034	0.838346	00:47

Fig. 10



epoch	train_loss	valid_loss	accuracy	time
0	0.624710	0.546055	0.720301	00:53
1	0.528010	0.451693	0.787970	00:46
2	0.458612	0.397347	0.823308	00:46

Fig. 11



epoch	train_loss	valid_loss	accuracy	time
0	0.658824	0.578194	0.679699	00:46
1	0.543819	0.478857	0.766165	00:46
2	0.466739	0.413165	0.811278	00:47

Fig. 12

Overall, it was interesting to see the difference between a hand created neural network model vs the ResNet transfer learning models. I was able to compare the different ResNets between themselves while also comparing them to different loss types and different optimizers. Future work on this project will include repairing my broken environment, so that I can use my own workstation. This will make it easier to create my own dataset of endangered species to train with and test with, without using the Google Colab environment GPU up. The new dataset will include a larger variety of images, most of which being endangered species. At the end of the future work, I hope to be able to assist researchers identify endangered species with ease and enable them to help the populations thrive again.

26, Renee Cho |March, et al. “Why Endangered Species Matter.” State of the Planet, 3 Apr. 2019, https://news.climate.columbia.edu/2019/03/26/endangered-species-matter/.

Trnovszky, Tibor, et al. “Animal Recognition System Based on Convolutional Neural Network.” Advances in Electrical and Electronic Engineering, vol. 15, no. 3, Faculty of Electrical Engineering and Computer Science VSB - Technical University of Ostrava, 2017, pp. 517–25, https://doi.org/10.15598/aeee.v15i3.2202.