Multi-class Image Classification for Mosquito Families Using Convolutional Neural Networks and Decision Trees

Group 600 (SDSE Fall 2023)

Rocklin Duong, Athena Lewis, Charlize Niswanger,
Masayuki Ohashi, Aryan Sood, Abigail Wood

December 6, 2023

1 Introduction and Project Description

Mosquitoes are ectoparasitic insects that can transmit infectious diseases to humans and animals. They belong to a diverse group of about 3,500 species that have different vectorial capacities and ecological preferences. Some of the most important diseases that mosquitoes can carry include malaria, dengue fever, and Zika virus, which pose serious threats to public health in many regions of the world. Therefore, accurate identification of mosquito species is essential for disease surveillance, prevention, and control, as well as for understanding the epidemiology and ecology of vector-borne diseases.

However, mosquito identification is not a trivial task, as it requires specialized knowledge and skills, as well as access to reliable morphological or molecular tools. Moreover, mosquitoes are small and delicate insects that may have subtle or cryptic morphological differences among species. These factors make it difficult for non-experts or people living in resource-limited settings to identify the mosquitoes they encounter and assess their potential health risks.

To address this challenge, our project aims to develop a user-friendly and accessible tool that can automatically identify the species of a mosquito from an image captured by a smartphone or a digital camera. Our tool is based on image classification, a machine learning technique that can assign a label to an image based on its visual content. We selected six mosquito species that are of high medical importance and have a wide geographic distribution: Anopheles, Aedes

aegypti, Aedes albopictus, Culex, Culiseta, and Ochlerotatus japonicus. To implement our image classification tool, we trained two different models: a convolutional neural network (CNN) and a decision tree. Our objective was to compare the performance and feasibility of both models and evaluate their strengths and weaknesses for our problem. Both models are widely used for image classification tasks, but they have distinct mechanisms and characteristics. Our CNN model uses a softmax activation function to assign a probability to each class based on the features extracted by the convolutional layers, while the decision tree model uses a series of binary splits to partition the feature space and assign a class to each leaf node. The decision tree model is simpler and more interpretable, but it has fewer parameters to tune. The CNN model is more complex and flexible, but it requires more data and computational resources.

One of the main difficulties of this project was the high degree of similarity among the mosquito species, which makes them hard to differentiate even by human experts. Unlike dog breeds, which have obvious morphological variations, mosquito species have subtle and cryptic features that require specialized training and knowledge to recognize. Therefore, we faced a challenging problem of image classification, with limited time and resources to train, test, and evaluate our models. We expected that our models would have low accuracy rates, given the complexity and ambiguity of the problem. We also wanted to compare the advantages and disadvantages of two different types of models, namely a convolutional neural network (CNN) and a decision tree, and see how they performed on our data set. We used Google Colab as our platform for developing and running our project, as it allowed us to collaborate easily and efficiently, as well as to leverage the GPU provided by Google to speed up the training process and reduce the computational burden on our personal computers.

2 Data Description

The data is sourced from Kaggle and contains photos of six families of common mosquitoes. The photos come from the phone app iNaturalist, which is a community-based platform for sharing photos of flora and fauna for identification and collection. The photos are broken up by family, each with its own capacity for carrying disease. Each family contains a varying amount of photos in the original data set, most having over one thousand photos and only japonicus has less, with 887. Albopictus and culex have 5442 and 6841, respectively.

Initially, 20 photos of each species was chosen for training. The photos were chosen based on

quality and clarity of the specimen within the frame of the photo, but initial runs of the models struggled to accurately label the testing data. Likely reasons for this are that each family of mosquito is unfortunately similar enough to each other that the models are not able to fully identify unique qualities between families. While the human eye might be able to identify that only the culex family has a large furry abdomen, the model may only be able to identify features that most families share, such as basic shape, length, and size. As such, the only strong factor between identification would likely be color, which, while it can be differentiated between species with enough information, can be easily mistaken across species with similar coloring. This caused our initial model to struggle with basic identification. Our initial model was only two classes, aegypti or not aegypti, and when fed a test image of anopheles, another black mosquito, it only barely identified that it was not aegypti.

Moving forward, we starting using a larger number of photos in each class, but we refrained from using the whole collection. While the accuracy of the validation and testing data has not reach a desired state, it has still improved the quality of the models involved. An unfortunate drawback is that the time of the model running has increased by a significant amount. The CNN model takes around an hour to an hour and a half to run 40 epochs. Luckily, the history of the model implies that increasing the number of epochs would not significantly aid in the unusual loss patterns. Another drawback is that due to the sheer size of the data set, manually picking out high quality photos is impossible among the time frame. In an expanded project, an ideal start to the models would be to have the model comb through the images and omit any with high compression, any with the specimen cut off, and any where there are bright colors cast on to the mosquito such that the true color of is obscured. However, this is high level computer vision and not something the team was able to accomplish. It was also important that we keep the number of images per class roughly equal, so that the model did not skew towards any family with a higher number of images. The end data set we are using has 700 training images per class and 150 validation images per class. There are 50 images used for the final testing of the model, and 6 specific images, one from each family, were also set aside for observation. The same data set was used for each model.

3 Methodology

The first model we will be looking at is our convolutional neural network. We chose a CNN so that the model could identify the unique features of each family in order to classify each image.

Important aspects a CNN could identify would be color, abdomen shape, leg length, and wing shape. Observing section 1.3.1 in the CNN model, we will briefly discuss the choices made for the model parameters. The model uses three 2-dimensional convolutional layers. The first layer creates a 5 by 5 output from the input with the image's dimensions, and padding is added to the resulting feature map to make it the same dimensions as the input, which, in our testing, found slightly aided accuracy. The following two layers have a smaller kernel size so that the finer details are captured and analyzed by the model, which is important when distinguishing the details of very similar looking specimen. The number of parameters, as a result, increases as seen in the CNN model summary within 1.3.1. The l2 functions within the second and third layers are the loss penalty with a strength of 0.00005. The layers are then flattened in order to be processed by the dense layers. 40 epochs were chosen for the final training parameters because this is the point where any further training results in a consistent plateau for validation and accuracy losses. The first dense layer has a dropout function to help mitigate over-fitting, and the final dense layer uses a softmax activation function to map the likelihood of each family against each image.

An important parameter to note is that of the random seed. In section 1.3.1, the second line has a commented out section setting up a random seed. This seed ensures that across runs, random variables such as the dropout rate and weight initialization do not cause inconsistent results across runs. For our model, we have decided to not include this seed. While the results do come out more consistent, they also come out far less accurate and with more plateaus across validation and losses. As a result, upon running the results, the model history is likely to be different than the results analyzed here, but the end accuracy should roughly be the same.

One interesting quirk of the model that was noticed was that the loss would actually increase with each epoch. To try and reduce this phenomenon, we implemented a learning schedule. In section 1.3.2, the ReduceLR0nPlateau function reviews the validation loss and reduces the learning rate when this value does not decrease for 5 epochs. This ideally would allow the learning rate to change until a value is found that works consistently across the model without sacrificing run time or accuracy.

The second model we employed was a decision tree. We chose this model because an ideal model would analyze the most important features of each family and assign a hierarchy to best analyze the appearance of a mosquito specimen. We considered between a decision tree and a random forest model. We ended up deciding on a decision tree due to the already long training times. A random forest model is made up of multiple decision trees and is a very in-depth, training-heavy model.

Therefore, it did not seem in the best interest for a model built mainly for comparison of procedures and model history and not one that was interested primarily in accurate results.

The model for the decision tree is set up in the same file, and begins below the results of the CNN model at 2.2.1. In section 2.1.2 the images are first compiled so that the specimen and its respective family labels are held in arrays. Section 2.2.2 prepares the test data for use after the decision tree has been trained. Section 2.3.1 is where the decision tree is fully run, and the resulting map is displayed below. There are no parameters here to explain due to the simplistic nature of the model. In summary, the model is reviewing the patterns within the data of each image and the resulting family label and then attempts to rank the features it finds as important for identification. In the analysis section we will review the resulting tree and its model history.

4 Results and Analysis

4.1 CNN Results

The convolutional neural network model features two test data set predictions. The first is predictions with random test photos and the second displays predictions with set test photos. For ink consideration purposes, the results from 1.5.3 are shown below and the 1.5.2 results can be viewed in our code.

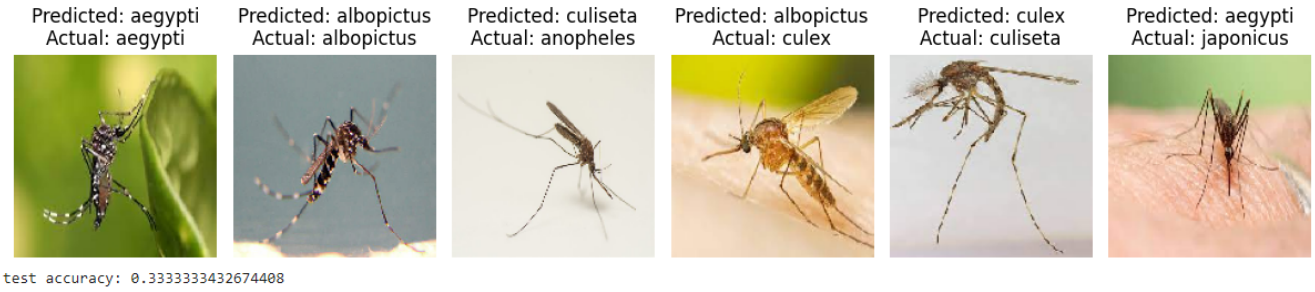

Figure 1: Section 1.5.3, 6 set test photos.

Using random test photos, the CNN executed at 30.33% accuracy when identifying the correct mosquito species. As can be seen in the output of 1.5.2, ten random test photos were chosen. These photos consisted of two photos for the albopictus and aegypti species and one photo for the other four. The model correctly identified both photos of albopictus and one of the photos of aegypti. For the second photo of aegypti the model predicted albopictus. This is not entirely shocking as the two species are similar with their black and white coloring. As far as correctly identifying the

species, this is where the accuracy ends. The other four species were not correctly identified as can be seen in the output. Interestingly, the model primarily predicts aegypti and albopictus.

For the predetermined test photos, the accuracy was 33.33% and the predictions and actual species can be seen in the output of 1.5.3. There are six set test photos, or rather one chosen photo for each species. Once again, the correctly identified species are aegypti and albopictus. With anopheles, culex, culiseta, and japonicus being predicted as culiseta, albopictus, culex, and aegypti respectively.

Our CNN portion also features a series of graphs where we plotted Training/Validation Accuracy, Training/Validation Loss and the Learning Rate across the number of Epochs. The corresponding graphs from 1.4.1 are provided below.

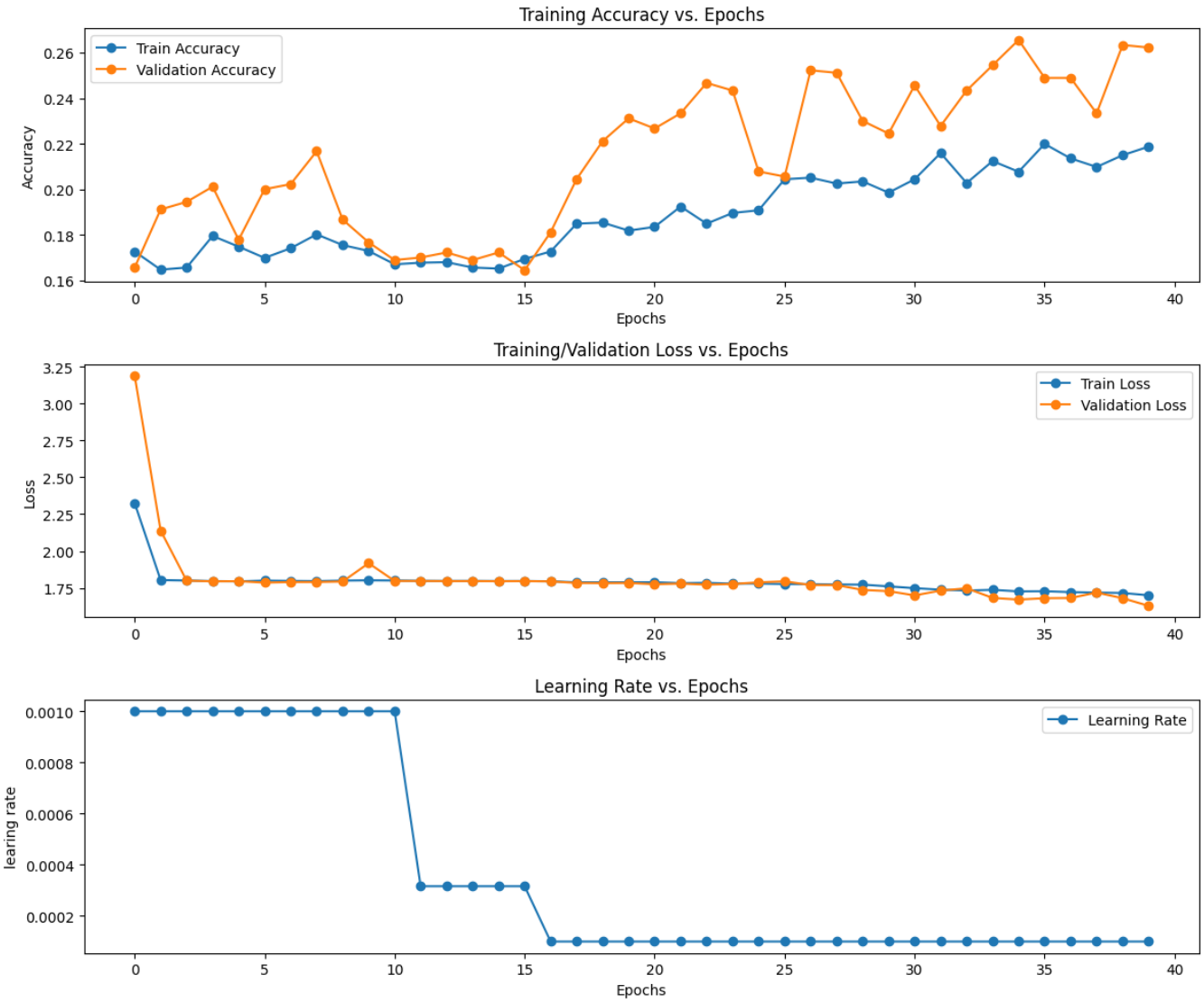

Figure 2: Section 1.4.1 Tables

Analyzing the Learning Rate vs. Epochs graph reveals a decreasing trend as epochs progress,

ultimately stabilizing around the sixteenth epoch and plateauing. This suggests that the model has reached a stable configuration. Simultaneously examining the Training Accuracy vs. Epochs graph, both training and validation accuracy show an upward trend, indicating ongoing learning with increasing epochs. Despite the upward trajectory, neither accuracy metric converges, and the overall accuracy remains relatively low, consistent with our earlier findings. Looking at both graphs together leads to the finding that around epoch 15, both training and validation accuracy experience a notable jump as the learning rate stabilizes. This suggests a correlation between the convergence of the learning rate and an increase in accuracy. Turning attention to the Training/Validation Loss vs. Epochs graph, a distinct plateau is evident in both training and validation loss starting around epoch 2,. This continues with a slight fluctuation until around epoch 27. While there is a modest dip in losses at epoch 27, the plateau remains relatively stable.

Another take away from our loss graph is that there are very large losses in our model. This could be the result of having a noisy data set. There are a lot of similar characteristics between the six species, and this could make it difficult for the model to differentiate between the photos. Another possibility is that our model requires additional layers to better capture the patterns in the data, however, adding more layers may result in over-fitting and would likely extend the training time beyond a time that would make it easy for us to test and run the code.

4.2 Decision Trees Results

The decision tree model was tested with two parameters: as a single tree, and as 150 trees. When running the model, we strongly recommend refraining from running section 2.4 as it can take multiple hours. This section was left in for reference and not as functional parts of the code.

	precision	recall	f1-score	support
0	0.06	0.07	0.07	41
1	0.02	0.50	0.04	2
2	0.00	0.00	0.00	1
3	0.76	0.17	0.28	223
4	0.00	0.00	0.00	5
5	0.20	0.36	0.26	28
accuracy			0.17	300
macro avg	0.17	0.18	0.11	300
weighted avg	0.59	0.17	0.24	300

(a) Single decision tree.

	precision	recall	f1-score	support
0.0	0.00	0.00	0.00	1
1.0	0.00	0.00	0.00	0
2.0	0.26	0.25	0.25	52
3.0	0.00	0.00	0.00	3
4.0	0.80	0.16	0.27	244
5.0	0.00	0.00	0.00	0
accuracy			0.18	300
macro avg	0.18	0.07	0.09	300
weighted avg	0.70	0.18	0.27	300

(b) 150 decision trees.

The macro average being close to the weighted average for the single decision tree matches expectations as each class has roughly equal numbers of training data. However, the discrepancy between the macro average and weighted average in the results for the 150 decision trees is odd.

This could be ascribed to the fact that most of the data was labeled as culiseta, although the single tree has the same issue with culex and yet still maintains symmetric averages. Both models have low f1 scores, which makes sense as the accuracy and precision is rather low across the board.

Comparing the two results, it is apparent that the decision tree model does not fall too short of the CNN's performance. Despite the model's far different consideration of the data, it ends up with an accuracy in the 0.15 to 0.20 range. This is an expected result if the model applies one label to every input, there are 6 classes so it will get roughly 1/6 correct. It should also be noted that the 150 decision trees does not perform much better than the single decision tree, certainly not enough to justify the lengthy run times, as the single decision tree can easily run in a minute or so. The reasons for this low accuracy is likely similar to the CNN model's struggles in that the data is likely far too noisy and the families too similar to be able to raise certain features above the rest. Its tendency to fit to one family could come from a tree that is weighted towards one family first before running through the features of the image. While we used the same exact data set for each model for comparison, the decision tree may actually benefit from a smaller data set. This may allow us to increase the number of trees without sacrificing hours of time while hopefully not compromising accuracy.

4.3 Comparison

Because neither model ever reached strong accuracy scores on their own, the reason to use one or the other does not end up relying on accuracy. Both struggle to consistently label the families, but the decision tree is held back by its over-reliance on certain families when applying labels. It is also held back by its lengthy training time that will only grow as the model is refined. Further adding layers, epochs, and data sets will all extensively add to the amount of time it takes to train the model, which makes it more difficult to go back and forth to refine and test. That being said, the high tunability of the parameters means that there are many avenues to tweak the model to perfection. Perhaps changing the kernel size of the layers, or the weight of the loss penalty could improve the overall history of the model. The decision tree excels in its speed and simplicity in use, but the main controllable factor is in the number of trees trained, which in our case has not resulted in higher accuracy, and its dependence on single family assignment rather than pattern recognition holds it back.

Performance could be further improved by altering the data set or the way it is analyzed prior to being fed into the models. A data set that is not photo based but rather feature based would work

well, such as a set broken up by family and the associated features (brown, long, fuzzy, skinny, for example). The model would also be improved if we could analyze those features within the photos ourselves, by running the photos through an imaging algorithm to detect contours and pulling out the respective features and then storing that for later use.

References

[1] *Ahmadjaved.* (2022, February 6). Multiclass Image Classification using CNN. Kaggle. https://www.kaggle.com/code/ahmadjaved097/multiclass-image-classification-using-cnn

[2] *V, N.* (2023, April 5). Image Classification using Machine Learning. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/01/image-classification-using-machine-learning/

[3] *W*hat Is a Convolutional Neural Network?: 3 Things You Need to Know. MathWorks. www.mathworks.com/discovery/convolutional-neural-network-matlab.html

[4] *Chatterjee, Chandra* (2019, July 31). Basics of the Classic CNN. Medium. https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add

[5] *W*hat is a Decision Tree? IBM. https://www.ibm.com/topics/decision-trees