Title

Reimplementation of CNN's for Insect Image Recognition by use of Stock insect images within a Kaggle Database.

Introduction

Our paper wanted to create a way to streamline the process of identifying insects. The process in which this happens becomes time-consuming, monotonous, and utilizes an unnecessary amount of resources that can be used on other things. Hence why the intention was to utilize a convolutional neural network (a model that inputs photos) and be able to identify insects that way. We chose this paper as a way of applying our knowledge, and give us a hands-on opportunity to utilize a neural network to gain experience. This particular project would be considered a classification problem. We are giving our model one particular insect, and we are having it classify what type of insect that was given to the model.

Methodology

What is the architecture of your model? (Mention input space and target space!)

At the moment we are planning on doing a VGG16 as our intent is to just classify insects, since this is our first Neural Network we are working on, this will be the most manageable architecture we can use at the moment. From the looks of our Kaggle Dataset, the photos vary widely between stock photos of insects, drawings, zoomed in photos and black and white images. It will be difficult to try and account for these varied images, but we feel a good input space for our data will be RGB, and 224x224 this will help account for the amount of non-uniformity of our dataset. As far as our target space goes, it will be represented as K=5, because there are 5 classes of insects, Butterfly, Dragonfly, Grasshopper, Ladybird/

Ladybug,and Mosquito within our dataset. We also added batch norm, and drop to the classifier layers.

How are you training the model?

We'll train the model using images from the Kaggle dataset, leveraging pre-trained weights from the initial convolutional layers while training the final linear (classifier) layers to classify 5 different insect types. The convolutional layers will have frozen weights, while the classifier will either be replaced with a smaller dense network or retain the existing architecture.

What loss function are you using, and why is it appropriate for your problem?

We are using Categorical Cross-Entropy for our loss function because this measures how well the model's predicted probabilities match the true class labels by penalizing confident incorrect predictions. It is especially appropriate for insect classification because each image belongs to one specific insect species, and categorical cross-entropy works perfectly with softmax outputs to help the model learn to assign the highest probability to the correct insect class while suppressing the others, leading to more accurate species identification.

What optimizer are you using, and how did you choose it?

We'll use the Adam optimizer because it's well-suited for image classification tasks due to its adaptive learning rates, efficient handling of sparse gradients, and fast convergence. Adam combines the benefits of momentum and RMSprop, making it a robust default choice for training convolutional neural networks.

If you are implementing an existing paper, detail what you think will be the hardest part about implementing the model here.

I think the hardest part of implementing the model here will be the difference in the data we use. The model was initially created to look at images of insects that were specially taken in a lab setting where they were properly displayed and the backgrounds of the images were blank. Our dataset has clearly made drawings of the insects, as well as mostly photos with background elements present. We may have to end up cleaning the images that aren't of real animals out of the dataset,

Metrics:

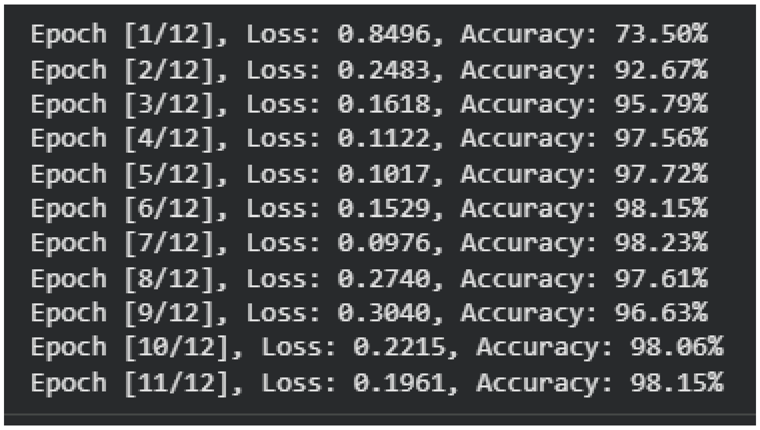
Our main goal is to focus on model's ability to distinguish between insects accurately, for example, something like an ant should never end up classified as a butterfly, we hope to also get it precise enough to be able to recognize something that may look semi-similar at first glance, like a moth versus a butterfly or ground insects like ants versus beetles. Accuracy is the most important metric here, we will likely look at how it might classify false positives or false negatives more depending on how it functions, but we do not need to worry as much as we would if it was a model on tracking what a tumor looks like when it is cancerous, or other matters where you want to avoid specifically telling someone they don't have cancer when they do.

We are going to run the model on a new dataset that includes very different images, and see what other ways the model can be trained, like retraining the last linear layers or replacing them, and seeing if it reacts differently to the odd elements in our dataset, like the graphics of bugs versus actual images, and any different poses and backgrounds the images may have. The authors were looking to see how they could tune the VGG16 model to recognize differences in lab-taken precise images to classify specific insects. We intend to use it on a larger scale with some different tuning and training to classify base categories rather than specific species and have it work with images that real people would likely submit, as most don't have access to a lab and a specimen of an insect that will stay still for a picture.

Results

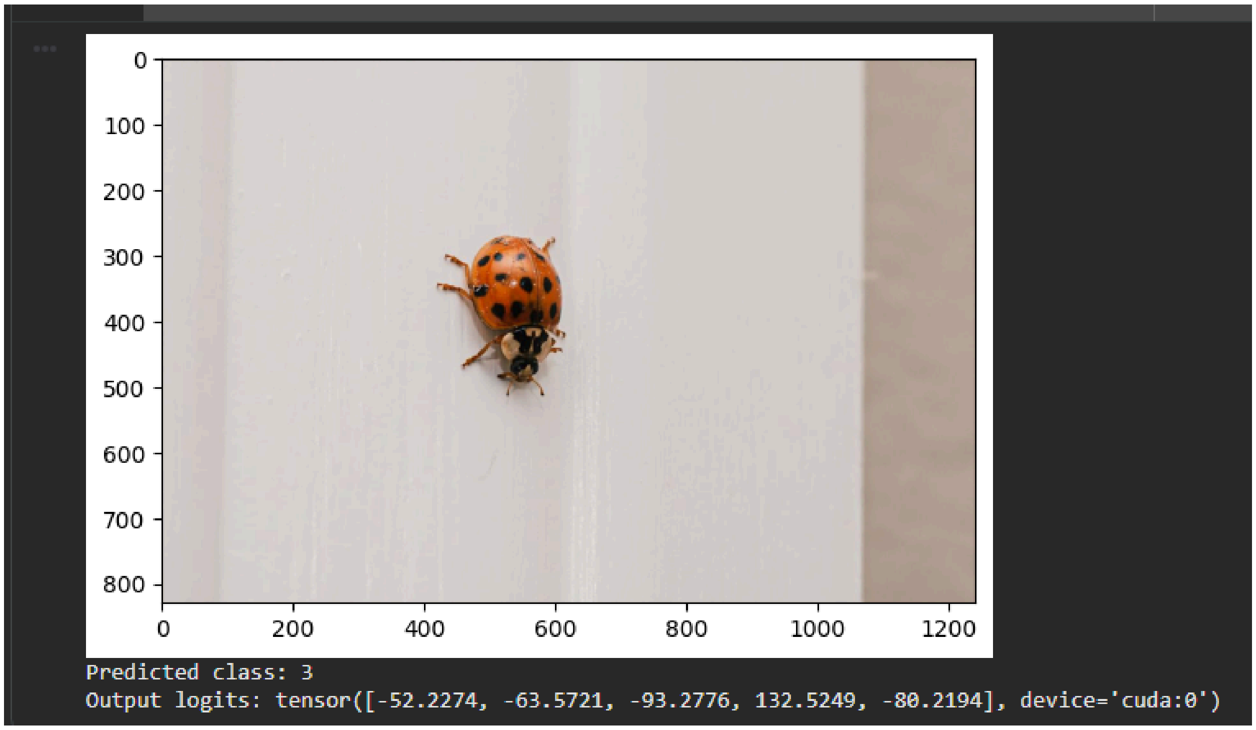
These are updated results after cleaning the data, and removing irrelevant data.

Our training results:



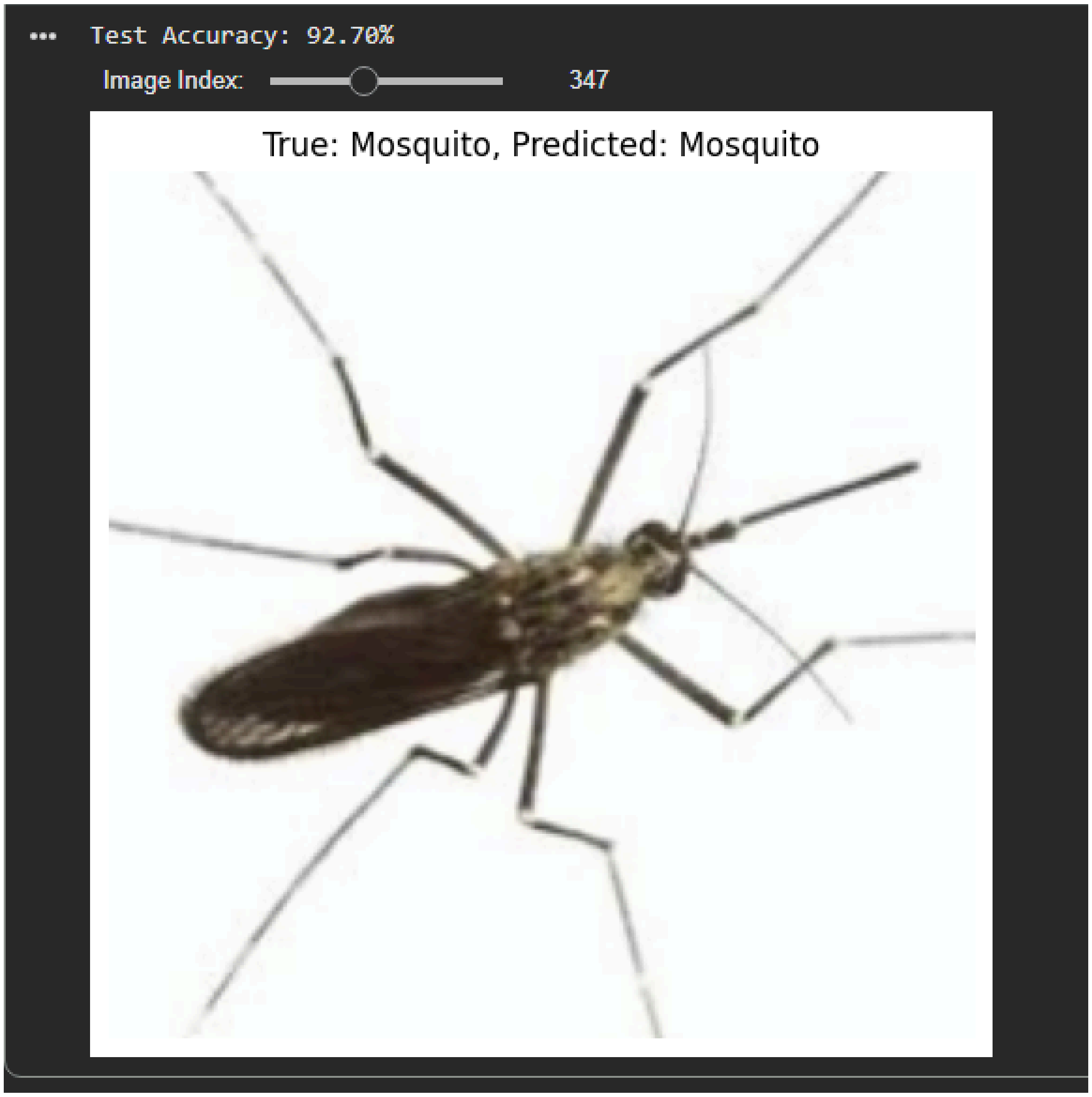
```
Epoch [1/12], Loss: 0.8496, Accuracy: 73.50%
Epoch [2/12], Loss: 0.2483, Accuracy: 92.67%
Epoch [3/12], Loss: 0.1618, Accuracy: 95.79%
Epoch [4/12], Loss: 0.1122, Accuracy: 97.56%
Epoch [5/12], Loss: 0.1017, Accuracy: 97.72%
Epoch [6/12], Loss: 0.1529, Accuracy: 98.15%
Epoch [7/12], Loss: 0.0976, Accuracy: 98.23%
Epoch [8/12], Loss: 0.2740, Accuracy: 97.61%
Epoch [9/12], Loss: 0.3040, Accuracy: 96.63%
Epoch [10/12], Loss: 0.2215, Accuracy: 98.06%
Epoch [11/12], Loss: 0.1961, Accuracy: 98.15%
```

Taking a picture the model hasn't seen and testing if it is able to categorize it correctly or not.



```
Predicted class: 3
Output logits: tensor([-52.2274, -63.5721, -93.2776, 132.5249, -80.2194], device='cuda:0')
```

Our testing results:



```
Test Accuracy: 92.70%
```

Challenges

After cleaning the data, by getting rid of pictures not relevant to insects within the kaggle dataset of things not related to insects we began having issues with overfitting. We were receiving accuracy of 95%> and tried using early stopping, and norm 1 dimension and dropout of .5 to fix that problem in the classifier layers to prevent overfitting.

Reflection

How do you feel your project ultimately turned out?

We felt that our project came out surprisingly very well. At first when reading the initial research paper we were concerned with the complexity of the model building process and going about fixing the images within the kaggle dataset. With how the research paper took lab-based pictures, we thought it would've involved a lot of adjustments to reach those same level of results.

How did you perform relative to your base, target, and stretch goals?

We had a couple of steps from training, and testing the network and improving it with translations. We focused on the models ability to categorize between insects effectively, and look at how we classify false positives. With that being said we've reached the target, and reached our goal like cleaning the dataset, and performing early stopping and adding to the classifier layers.

Did your model behave as expected?

Our expectation going into this project was very low, we didn't think our model would be able to recognize the insects of the kaggle dataset because of its varied nature. However, it exceeded our expectations, and managed to perform very well and reached very high performance metrics.

How did your approach evolve over time? What pivots did you make, if any?

It evolved through changing, and removing certain images from the dataset. Some of the images were just too varied, where some images had multiple insects, or not even having insects at all. While our initial running of the model performed well, we thought it was a noticeable issue when just looking at the dataset at first. It showed us the importance of data cleaning, and going through the pre-processing portion of your model building process.

If you could redo the project, what would you do differently?

If we were to redo the project, we would just try to recognize more labels in general, perhaps classes like Praying Mantis', or other species of grasshoppers, specifically more granular data with more classes. Also, we would start out with batch_norm and dropout after seeing the success we had with it.

<u>What areas could you further improve if you had more time?</u>

We would try to improve on the model training, and test accuracy by changing the convolutional layers with a minimal learning rate. To get the network to understand the images better.

<u>What are your biggest takeaways from this project? What did you learn?</u>

VGG architecture is very powerful, the pre-learned filters really help the classifier layer classify images it has not seen before. It is actually doable on a 15 GB GPU, because the sentiment on Neural Networks you would think we would need large GPU's to run these models.