AWS Machine Learning Engineer Nanodegree

Capstone Project

Noah Rizika

Project Report

Image Classification Model for *Drosophila* Embryonic Development Stages

DEFINITION

Project Overview

This project is inspired by an issue that arises in a specific research field in cell biology, specifically research that deals with *Drosophila* embryos. This project plays a part in *Drosophila* embryonic development research that studies how genes work together to regulate cell behavior, thereby generating different tissue forms. *Drosophila* embryos are a great model for this research because of their genetic components, live-imaging accessibility and biophysical analysis. In terms of this project, the relative ease of imaging allows for researchers to photograph and classify a cell's stage at different intervals.

The project will be an image classifier model, and the data are images. Each image is a .tif file of size 1110x420 and is of one cell. Before training, various image augmentation techniques such as rotation will be conducted to maximize the number and quality of data. See below for charts showing the data distribution before and after image augmentation.

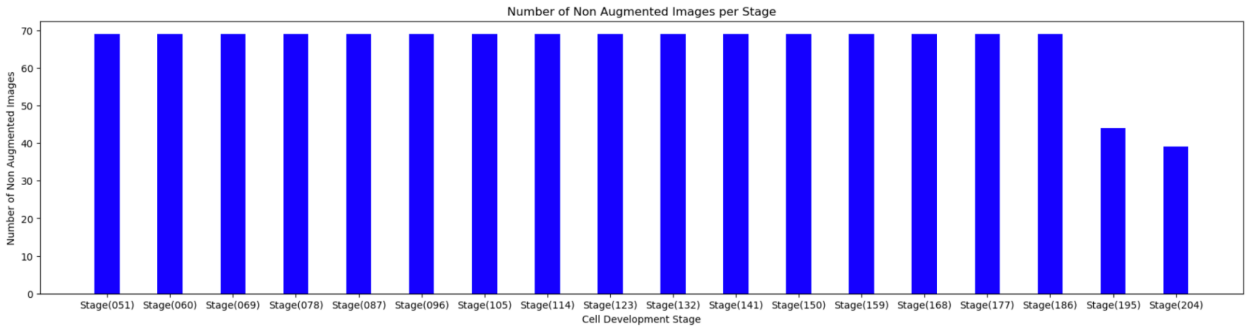


Figure 1. Bar plot showing the number of images per stage before image augmentation.

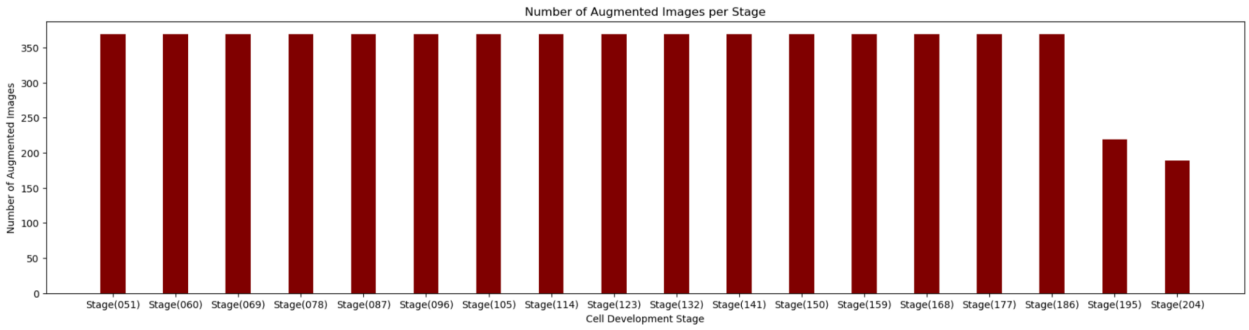


Figure 2. Bar plot showing the number of images per stage after image augmentation

As a high level overview, the project consists of data preprocessing, image augmentation and conversion to png, tuning the learning rate and batch size hyperparameters, training the model using these tuned hyperparameters, deploying the model and testing the model's accuracy.

Problem Statement

Classifying the stage of embryonic stages of development for *Drosophila* cells is difficult, tedious and time consuming. In this research, many images of individual cells were captured and manually classified in order to be further analyzed for working toward understanding the research focus explained above. Manually classifying the cell's stage of development is very time consuming, which slows the research's progress.

In order to expedite the research process, I will build an image classification model to classify the stage of embryonic development for *Drosophila* cell images. The project will be created entirely in AWS Sagemaker, using a python Jupyter notebook and python training scripts. After manually uploading data to Sagemaker, the data will be augmented and uploaded to S3 via the Jupyter notebook. The notebook will run off of a ml.m3.medium instance, training and tuning the model will use a ml.m5.xlarge instance, although the instance count will be greater than one.

Metrics

Some of the data will not be used by the model to train on, and instead will only be used to test the model's accuracy. If the model also trained on this test data, then the accuracy rating would be inaccurate, as it will not showcase the model's generalizability. I emphasize this because

relative to the needs of an image classification model, the number of data is sparse. However, the need to measure the model's accuracy exists nonetheless.

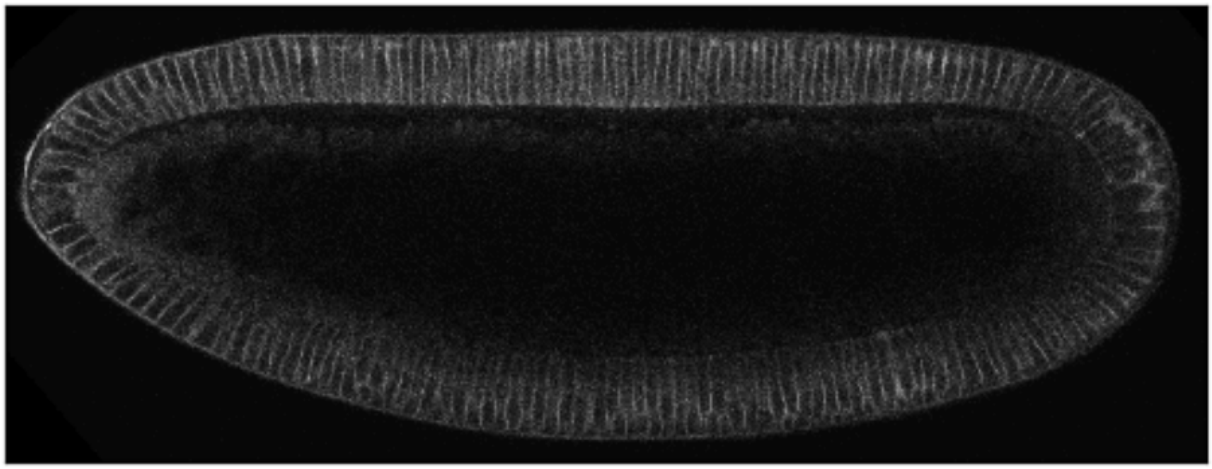
In order to account for inaccurate predictions and tailor the model's output to best help the researchers, the model will output its top three predictions per classification request. The model will be evaluated on if the cell's stage is one of the three printed (ie: most confidently predicted) stages.

ANALYSIS

Data Exploration

The data are black and white .tif images of size 1110x420. Each image contains one cell, which is black with faint gray lines marking the cell and its structure. The cell is centered in the image.

Exploratory Visualization



```
dtype: uint8, shape: (420, 1110, 3), min: 0, max: 217
```

Figure 2. Image and image details of an unaugmented cell. Image was read using cv2 and shown using matplotlib.pyplot.

In the screenshot above, we can see the height and width, but also that the max pixels is a relatively small number. This means that normalization will be greatly beneficial, as the model will be better able to differentiate aspects of the image when pixels differ greatly, resultanting in more accurate inferences.

Algorithms and Techniques

Mentioned in the paragraph above, one useful technique will be normalizing the images. This will be done manually by multiplying each pixel by 6. This number is sufficient to increasing the range of pixel values to span the entire range handled by cv2. The rationale for doing so is discussed in the previous paragraph.

Due to the relatively low number of labeled data, more images will need to be generated. This will be done by rotating and saving each image at different degrees of rotation. While training on identical images is an easy way to overtrain the model, non-rotated and rotated image(s) are diverse enough to train the model with both images, without risking heavy overfitting. It is a delicate balance, but given the small number of data, it results in improved accuracy. Of course, this technique is only notably beneficial to a certain extent. If there were more data available, the model may be more generalizable without training off of any duplicated, yet rotated, images.

The tuning algorithm uses sagemaker's tuner and a python script to learn the best specified hyperparameters, which are learning rate and batch size. These best hyperparameters are used to train the model, which uses its own, separate python script.

Benchmark

Some of the data was not used by the model to train on, and instead was used only to test the model's accuracy. If the model also trained on this test data, then the accuracy rating would be inaccurate, as it will not showcase the model's generalizability. I emphasize this because relative to the needs of an image classification model, the number of data is sparse. However, the need to justly measure the model's accuracy exists nonetheless.

METHODOLOGY

Data Preprocessing

Below is an outline of the two steps involved in data preprocessing. After image augmentation, images were uploaded to S3.

1. Data Organization

-Upload images from multiple folders shared over email
-Remove images of certain stages unrelated to the model using os
-Move images into folders that correspond to their stage using os
-Move and split images from uploaded folders into train, valid, and test folders using os and shutil
-Move images to different folders throughout the image augmentation process using os and shutil

2. Image Augmentation

-Image Rotation using cv2: rotated and saved at 45º, 90º, 135º, 180º, 270º)
-Image Normalization using cv2 for image access, then multiply each pixel
-Image Conversion from TIF to PNG using aspose

Implementation

I used the os and shutil libraries to create and move images and folders, the cv2 library for all image augmentation, and the aspose library for converting images to PNG. The metric type was set to minimize, and the metric definition used Regex and ([0-9\\.]+). The model's accuracy, however, was determined after deployment. While most errors when training the model weren't incredibly tricky to understand and work around, I hadn't known or performed any image augmentation techniques before this project, so I first had to understand the concepts and code before applying and developing the code. Additionally, I had to ensure the dtype, shape, and min and max values did not accidentally change throughout this process, which was especially tedious due to normalization, the original .tif format, and converting the images to .png.

One error common with moving and creating many folders and files is the automatic creation of "invisible" checkpoint data. Using os and shutil libraries to navigate SageMaker's storage, I had to account for the possible existence of these checkpoint data and ensure they were not accessed or manipulated in preprocessing or uploading to S3. Another issue is importing the necessary packages to use the cv2 and aspose libraries. The requirements can vary, so finding the correct packages to install given the python version, etc, was time consuming.

A third issue was the image's file type, in that it was unknown if the images were grayscale, BGR or otherwise. I tried to deduce the type by using cv2 to see how an image responds to being read as a grayscale, BGR, etc. When showing the image, in all cases the image would be unrecognizable, resembling that of a static on a TV. To work around this, I read all the images with cv2 as IMREAD_UNCHANGED. Lastly, while not an error, in order to verify images weren't misread, I had to convert the image from TIF to PNG in order to view it in SageMaker. This extra step caused the preprocessing phase to take more time, but I worked around it by developing the code using only one image for each step, then automating the process for each image.

Refinement

I discussed how I worked around some of the problems in the section above. As for the tuning and training scripts, there were many tedious errors, such as mislabeling different folders or metrics, using the best metric, ensuring all the necessary libraries were imported, etc. I had originally planned to also include a horizontal and/or vertical flipped copy of each image, but after better understanding the data, this would not be helpful. Each stage has specific and universal—albeit, difficult to identify—criteria. However, because the criteria will always be the same and always in similar locations, by flipping the images, the criteria would be reflected, resulting in different-looking criteria. Since "flipped criteria" will never appear in a cell, training partially on flipped images likely would not help, and instead could hurt the model's accuracy. For this reason, I decided not to include this augmentation in the preprocessing steps.

RESULTS

Model Evaluation and Validation

The ResNet50 model was used since it has over 23 million trainable parameters, which allows it to learn the specific attributes needed to classify each cell. The model's output is set to the input of a CNN with two hidden layers, which narrow down the ResNet50's output of many nodes (over 2000) to a final output layer of 17 nodes, the number of cell classifications. The model's solution was tested by making inferences on a small set of .tif images that were separated from the images used for training the model. If the correct answer was one of the model's six most confident inferences, the model would be deemed accurate for this prediction. Originally, the output would only be considered accurate if the correct stage was predicted within the model's three most confident inferences. However, due to the model's great inaccuracy with this as the metric, in order to be somewhat usable, the model will output its six most confident classifications.

Justification

The final model is not significant enough to adequately solve this problem. Unfortunately, the accuracy, measured as explained in the above paragraph, was 55.56%. This rating does not qualify the model to be confidently used, however it may help researchers by recommending they consider certain cell stages first when attempting to classify the images. It may also be used as an additional confirmation to a researcher's manual classification.

Although the model would greatly benefit from more data, there is still much more in my control that I could do to improve the model. For instance, I could try importing different image classification models, such as VGG16 which can be more accurate on smaller datasets, or further refine the image augmentation steps by changing or randomizing the angles of rotation. Also, as time passes in the research, more images will be classified and can be used in addition to the current dataset to retrain the model.