AWS Machine Learning Engineer Nanodegree

Capstone Project

Noah Rizika

Capstone Proposal

Image Classification Model for Drosophila Embryonic Development Stages

The Project's Domain Background

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.

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.

The Data and Inputs

The project will be an image classifier model, and the data are images. Each image is of one cell: the image is black with faint gray lines marking the cell and its structure. They are .tif files of size 1110x420. Before training, different image augmentation techniques such as rotation and masking will be conducted to maximize the number and quality of data. See below for a chart of the data distribution after image augmentation.

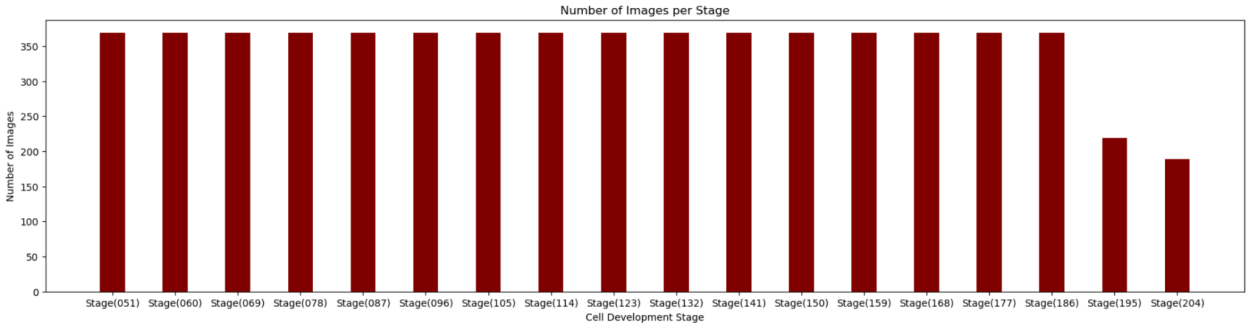

Figure 1. Bar plot showing the number of images per stage after image augmentation

Solution Statement

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.

Benchmark Model

While no previous model exists to compare my model's success to, the provided data are labeled by developmental stage. 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.

Evaluation Metrics

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 stages.

Outline of the Project Design

Given the organization and labeling of the provided data, the notebook will handle moving data into folders that correspond to the cell's identified stage. Image augmentation is necessary to increase the number of data and improve the quality. The PyTorch, cv2, imutils, aspose, shutil and os libraries, as well as others, will be used. There will be a separate python script for both tuning and training. The trained model will be saved to S3 and deployed through the Jupyter notebook. All test images and new, incoming images will be uploaded to SageMaker and converted to png (if they weren't already) and sent to the deployed model for classification. The image will be renamed with the top three inferences appended to its previous name. This is because the model's accuracy is ranked on if it can correctly classify the cell in its three most confident class predictions. Done for all test images, the percent accuracy will be determined by total number of correct inferences (existing in the model's most confident three inferences) / total number of inferences, as emitted by the deployed, trained model. See below for an outline of the Jupyter notebook.

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
    - -Move images to different folders throughout the image augmentation process using os
2. Image Augmentation
    - -Image Rotation using cv2: rotated and saved at 45º, 90º, 135º, 180º, 270º)
    - -Image Masking using cv2: turn the areas that are irrelevant to classification black
    - -Image Normalization using cv2 for image access, then pixel multiplication
    - -Image Conversion from TIF to PNG using aspose
3. Upload to S3
4. Tuning
    - -Learning Rate (continuous integer: 0.1 to 0.001)
    - -Batch Size (categorical integer: 32, 64, 128, 256)
5. Training

-images will be proportionally resized to a smaller size using torchvision.transforms

6. Deploying
7. Testing