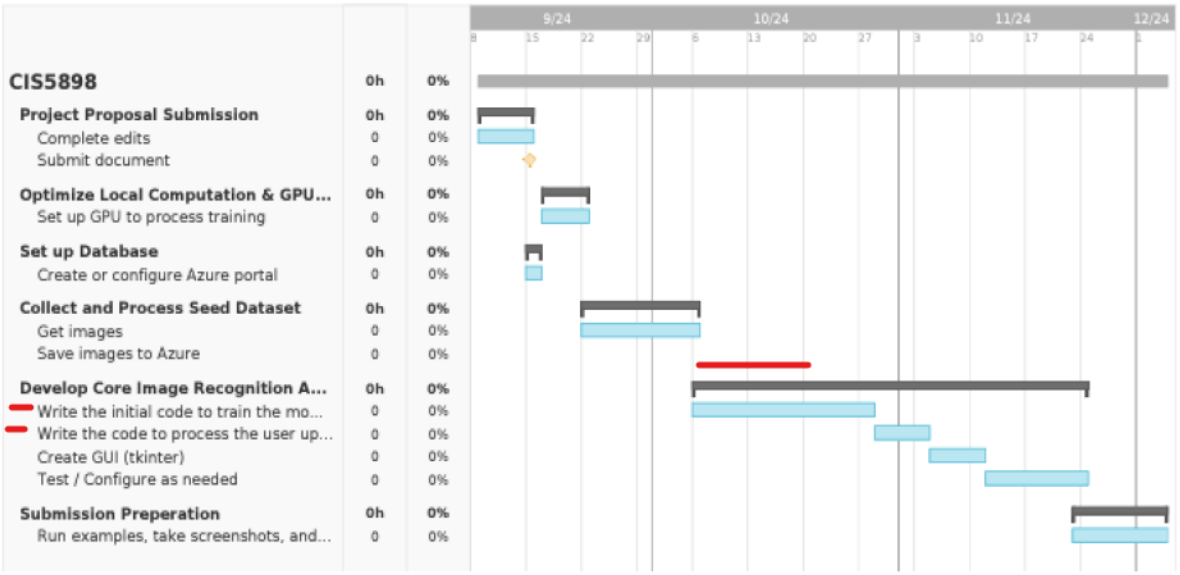
Matt Morrow

CIS5898

Third Progress Report

During this progress period, I attempted to implement the training process outlined in the article, *"Is it a Bird? Creating a Model from Your Own Data."* The program was to connect to the Azure database, retrieve the stored images, load them into a proprietary data loader, and train the model using the **vision_learner()** function.

Schedule (Gantt Chart)

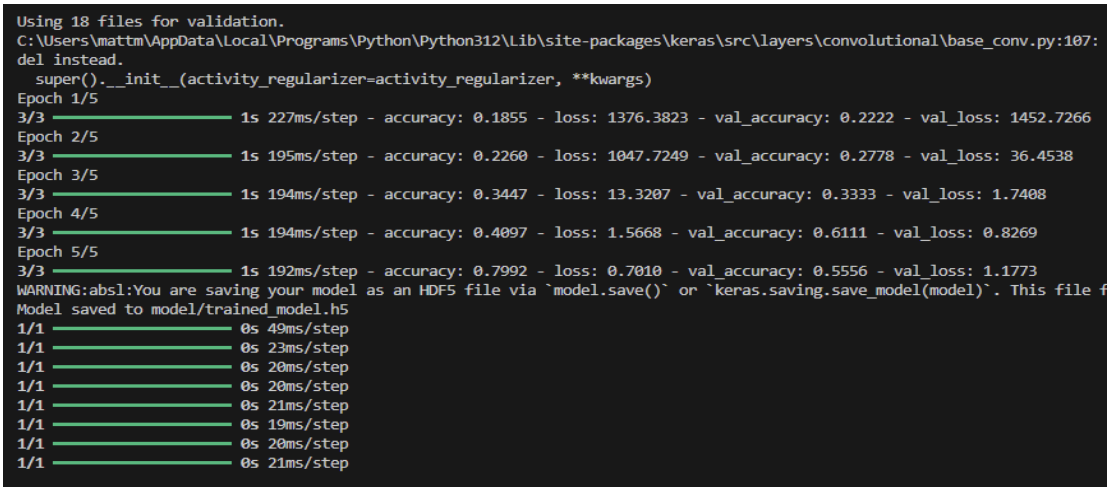


However, due to what seems to be an unexplainable compatibility issue with PyToch/FastAi and the latest Python version (3.12.4) with Windows 11; I needed to resort to an alternative method: Tensorflow.

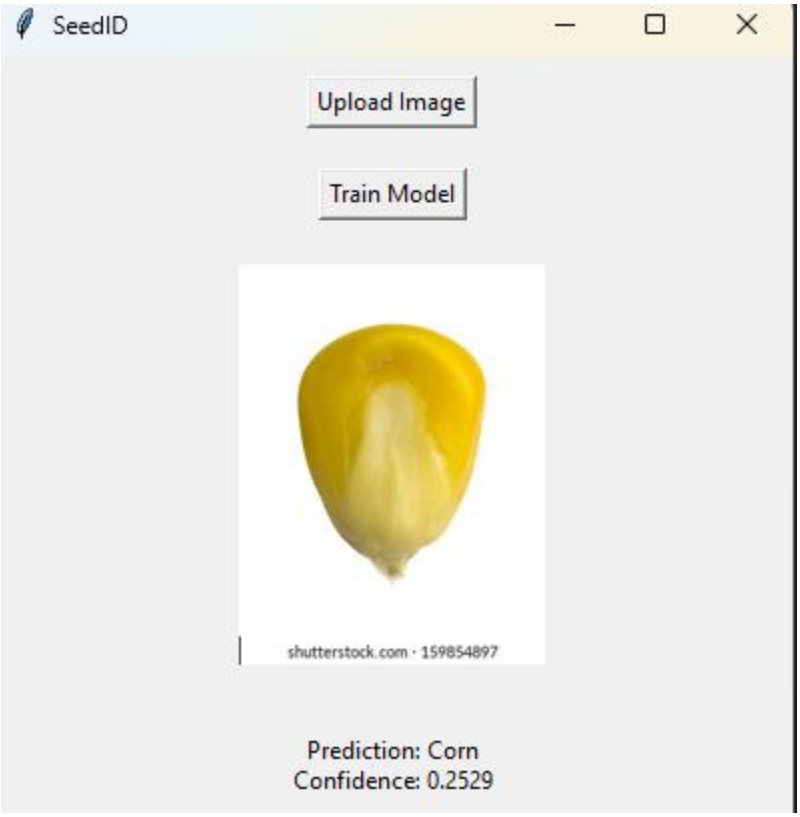
As you can see in the below screenshot the output was generating “nan” and 0.00 scores during the training of the custom model. Days were spent troubleshooting this process but to no avail.

After installing Tensorflow, I introduced a new class defined as, TrainModel, that prepares image data and trains a machine learning model using three functions, load_data, build_model and train_model. The load_data method loads and preprocesses image data from a directory, splitting it into training and validation sets. The build_model method constructs a convolutional neural network (CNN) with multiple layers, including convolutional layers, max-pooling layers, and fully connected layers, which is compiled with an optimizer and loss function suitable for multi-class classification. Lastly, in the train_model method, the model is trained using the provided datasets for a specified number of epochs, and the trained model is saved to a file for future use. These epochs can be altered to enhance the overall prediction accuracy of the model but it does require more processing power as the epoch increases.

When launching the train model action, the following output is displayed. Showing semi-successful results. I would like to see higher rates of accuracy, but it is a start in the right direction.



A basic GUI was designed using the Tkinter framework. This UI contain the buttons which allow the user to perform key actions such as uploading an image, training the model, and reloading the model after training. Additionally, it features an image display area that shows the uploaded image and a label that informs the user of the prediction score generated by the model.



When the user selects “Upload Image”, a method defined as, upload_image, opens a file dialog for the user to select an image, resizes the image for display in the GUI, and updates the image label to show the selected image. Once an image is uploaded, it calls predict_image method to process and predict the class of the image. If an error occurs during prediction, an exception is caught, and a relevant message is displayed.

Next Steps

1). Refine and improve the accuracy of the model.

2). Expand the custom image model database to include more seeds.

3). Improve logging/exception handling.

4). Include a log report that is saved as a file in the project’s directory.

5). Create comprehensive unit tests and run regression testing