Kaggle: Ovarian Cancer Cell Classification

Gregory Sylvester, Oakland University

Abstract

Ovarian Cancer is lethal form of cancer that needs to be treated with a host of solutions. This is where the need to classify the type of ovarian Cancer becomes crucial, as properly classifying it can be a cumbersome and difficult task without the proper knowledge, experts, and equipment[1]. So, developing a model that can properly classify the type of cancer will help facilitate the process of treatment for so many. The model I built was a tf_efficientnet_b2, which is a part of the open source Timm library. To help reduce the complexity of this project from becoming to large the thumbnail png files where classified not the WSI images. This helped to reduce the complexity and scale of the model so it would be reasonable for the project. Using this I was able to hit a validation accuracy of approximately .40.

I. Introduction

Kaggle is hosting a competition to help classify the types of ovarian cancers. The competition started October 6th 2023 and ends on December 27th 2023. The models which best meet the classification criteria will be picked as the winner. The problem breaks down into two criteria, one is the ability to detect outliers in the data, this is the result of certain domain knowledge specific attributes i.e., morphologies, etiologies, molecular and genetic profiles. Since the project for Kaggle requires heavy preprocessing, I scaled back my ambitions and chose to only work with the thumbnails instead of the data heavy original images.

Kaggle provides a large pool of knowledge sharing to help boost the overall performance by allowing individuals to share insights and other important information about the ongoing project[2]. This is where I received most help, such as choosing the model.

Fitting the right model is very crucial to increase accuracy, given our data tiling was required to help scale the dataset. The top performing prebuilt model (which was being shared) was tf-efficientnet-b0[3]. To build this model I used some import libraries such as Timm, Cv2, PyTorch, numpy, pyplot.

II. Related Work

Looking at other fellow Kagglers notebooks helped to resolve a great deal of problems in my model. There are two noteworthy notebooks that helped me. Firstly, being that of *Kera's train and infer on thumbnails[4]*. This notebook helps by providing a structure that of training based on thumbnails. This is the same method I took as it was a relatively fast and easy way to preprocess images. Second noteworthy notebook was *UBCOvarianCancer(@EfficientNetB3) [5]* This notebook helped to provide insight on how the Timm models should be deployed and trained. However, their implementation used TensorFlow instead of PyTorch.

III. Data

The competition initially provided 2 sets of data (sum of 794 GB), source data (WSI and TMA) and thumbnails. The whole source data was a bulk of the data which needed to be preprocessed and washed before using. This whole source data originates from 20 different institutions, each with different procedures for preparing and imaging slides. This means these images have vastly different scales, color palettes and other . While the thumbnails was relatively easier to work with, each thumbnail was only a couple MB.

There are 5 classes in data, CC , EC, HGSC, LGSC, MC. The distribution of the classes is below.

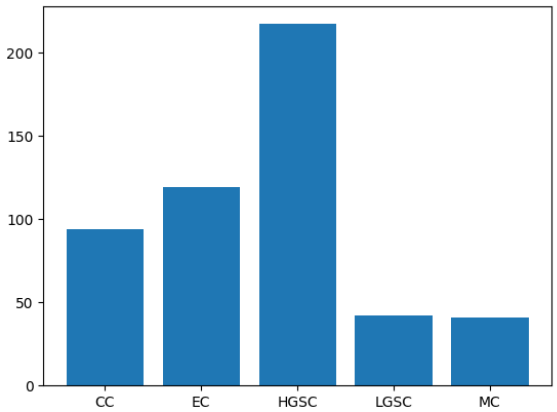

Figure 1 Distribution of Cancers

To help process the thumbnails they were loaded into NumPy arrays. At which point I took random Tiles from the provided array. Using that as the input

IV. Methods

A. Approach

To classify the images using the Timm tf_efficientnet_b2 model seems like the right approach as it was one of the better performing model. It is a CNN architecture with scaling compound. It is designed with the intention of capturing more fine-grained patterns on the bigger picture. Other approaches I thought of were the CNN with attention. However, this would require developing some sort of heuristic attention function to recognize import areas for tiling of the images. This became even more important when on the 16th the host released image annotations, color code highlighting the source data. This would be another approach using these highlighted regions to help train a model to detect these regions and importance in the classification task then deploying that on another model.

B. Other models

Some of the other major important models listed were Resnet, Mobile Net, and vision transformer[6]. All of which were listed as having lower accuracy than the Efficient Net.

C. Tiling

Tilling the images was one of the most discussed topics in the forums as it helped to drastically reduce the training time. As well as since most other participants needed their models to use the source data, they needed a method to reduce file sizes to process in Kaggle. Using random tiling was the easiest and most generalized way to implement this method of data augmentation.

V. Experiments

For the metric part of the measurement of the model I used the validation data set of batch size 16. Running the model for 20 epochs it didn't appear to provide much improvement beyond the 3rd to 4th epoch. For the optimizer I used Adam with a loss rate of .05 and criterion of cross entropy loss. For the hyper parameters not much tinkering has been done. Below is the result of the model produced.

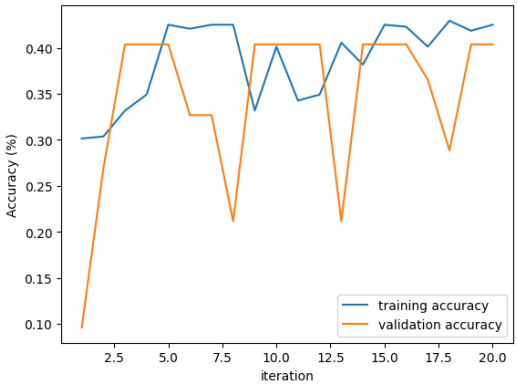

Figure 2 train/validation accuracy (20 epochs)

Common failures in building the model were image resizing and tiling. This caused the model to crash several times because I was messing with the input and output layers on the model. The biggest failure was the runtime environment since I tried using Kaggle's provided GPU, this would almost always overflow. In the end I ran it on my computer at home which has a local GPU, this solved the issue.

VI. Conclusion

For this project I learned how to implement a model from Hugging Face, selecting from the Timm library and using that to get mildly good results of .4 for the competition[]. This is comparable to the results from the 2 mentioned notebooks scored .54 and .43 respectively. In the future some adjustments that could be made would be changing the tiling selection using the highlighted regions in the annotation images. Another major change would make is to shift toward more preprocessing of the whole dataset instead of the just the PNG thumbnails.

VII. References

[1] https://www.kaggle.com/competitions/UBC-OCEAN

[2] https://www.kaggle.com/competitions/UBC-OCEAN/discussion

[3]https://huggingface.co/docs/timm/models/efficientnet

[4] https://www.kaggle.com/code/aritrag/kerascv-train-and-infer-on-thumbnails

[5] https://www.kaggle.com/code/raviiloveyou/ubc-ovarian-cancer-efficientnetb3

[6] https://www.kaggle.com/competitions/UBC-OCEAN/models

[7] https://timm.fast.ai/training