Student: Edwin Meyers
Date: 14-Sep-20

Capstone Project Mémoire

Prostate cANcer graDe Assessment (PANDA) Kaggle Challenge
Prostate cancer diagnosis using the Gleason grading system with neural networks

INTRODUCTION

The purpose of this document is to share the main components of the author's Udacity Capstone project (see "proposal.pdf" in github repository here for more background on the project). The project is based on a Kaggle competition named "Prostate cANcer graDe Assessment (PANDA) Kaggle Challenge" (see competition overview here) and the steps that were taken to increase the project's performance, more specifically its models' accuracies and submission scores (see competition leaderboard here). Additional insights on lessons learned are also shared when deemed worthy mentioning.

1_SETTING UP

The project was first being built on AWS but, because of GPU costs and dataset management complications, the project was eventually moved to Kaggle which offers free GPU-enabled kernels and easy access to a variety of data in addition to the competition data.

The pictures were multi-level ".TIFF" files, each level containing very different sized images. Of the three levels available, the middle one was eventually chosen because associated image sizes contained enough detail all while not being so big that they would overload the GPU's memory.

2_FIRST MODEL

Notebook location: main repository
Notebook name:

- *2_first_model_draft.ipynb*

Summary of step:
The objective was to set up an initial model that would serve as a foundation for further features to be built upon. The "foundation" model was largely borrowed from another notebook made by the author for a previous Udacity project (see image classifier here). Once built, the accuracy was measured (28%, not much better than random state = 1/(6 isup categories) = 16.7% accuracy) and next steps were defined according to quick calculations / time measurement / draft work done in the notebook.

Three principal achievements:

1) Understanding how to open a multi-level ".TIFF" file and determining which level to use for the rest of the project.
2) Modifying the dataloader's "__get_item__" class definition so that the dataloader can take excel sheets of specified image id's and labels instead of only loading what is in a specified folder. This was very useful given that there were 10,616 images and having the flexibility to sift through images nimbly was important (especially, as the reader will later see, for the Arutema method).
3) Creating a working model

It was clear that next steps had to be taken to increase the quality of the data inputs to hopefully improve the quality of the predictions.

3_IMAGE AUGMENTATIONS

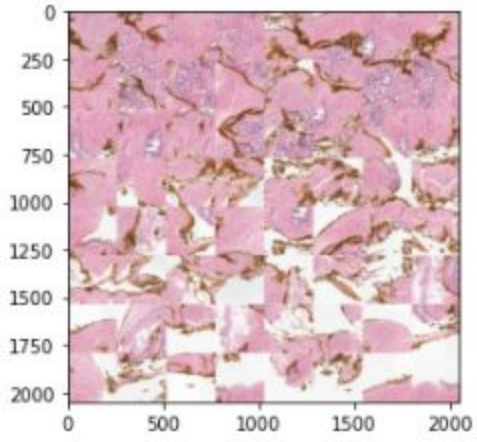
Notebook location: Main repository
Notebook name:

- *3_image-augmentations-masks-and-label-mapping.ipynb*

Summary of step:
From what the author could measure for a small sample of images, biopsy tissues usually only account for less than 5% of the image surface. The rest is background/glass laminate. This means that 95% of the image is essentially noise. The author borrowed algorithms from Kaggle user PAB97 (see notebook here) to implement image augmentations into the data refining process.

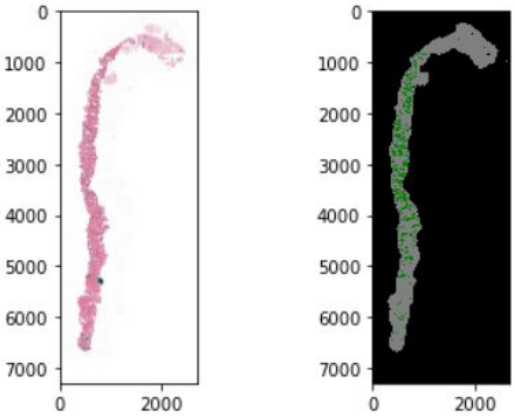
Principal achievements:

1) Learning basic image manipulation methods and how to transform a biopsy tissue into a square composed of high-tissue concentration tiles (image below is an enhanced version of a normal tissue image using a window size of 256, slide length of 166, and 64 tiles)

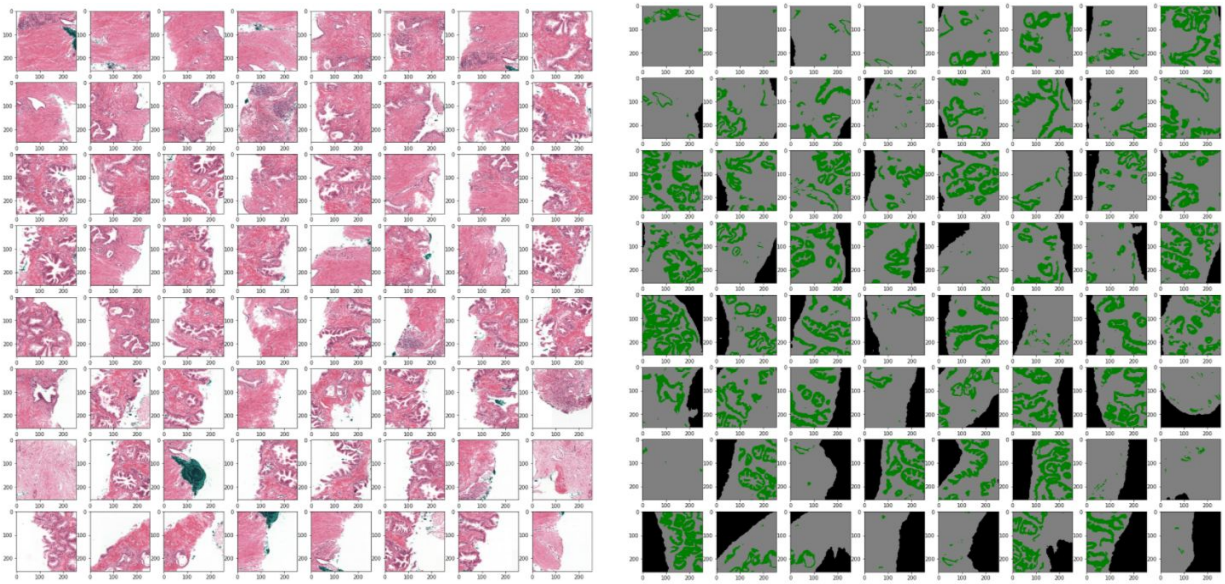


2) Learning how to open masks and map mask labels to image enhancements (first row of images below represents a normal tissue sample and its corresponding label mask, second row corresponds to image enhancements performed on both the normal tissue and the mask biopsy images)

Biopsy picture (left) next to corresponding mask image (right)



Augmented biopsy picture (left) with corresponding label masks (right)



3) Thought exercise of finding ways to make use of the label masks: There didn't seem to be any teams making good use of the label masks. The author's intention, though never implemented due to lack of time, was to experiment creating smaller images (256x256) with ISUP scores that could be determined using the score of the corresponding region in the label maks, and to then train a model with 100,000's of such images produced and, using the predicted label of individual smaller images, determine what the ISUP/Gleason score is of the larger, original biopsy image from which the smaller images came from. It will remain a thought for now, though the notebook has the capacity to save individual tiles images and assign them their corresponding ISUP grade and include such grade in the file name.

It's worth noting that ideally, the model would have been trained on 256x256x64 augmented images. Due to lack of time, the model is trained on a pre-prepared 128x128x16 augmented images database.

4_BASE MODEL AND FIRST SUBMISSION (57.2 submission score)

Notebook location: Main repository
Notebook names:

- *4a_base-model-no-folds.ipynb*
- *4b_base-model-analysis-and-submission.ipynb*

Summary of results:
56.6% accuracy rate with validation dataset
57.2 Kaggle submission score (see how score is calculated in the document named "proposal.pdf")

Model/Training information:
10 epochs
Model = Efficientnet-B4

criterion = nn.NLLLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

Principal achievement:

1) Implementing image augmentation for model training and increasing accuracy from 28% to 57%
2) Figuring out how to submit a model to the Kaggle competition. This was a very complicated part of the competition.

5_K-FOLDS TRAINING AND ANALYSIS

Notebook location: Main repository
Notebook names:

- *5a_Training model with k-folds.ipynb*
- *5b_K-Folds model - Analysis and submission.ipynb*

Summary of results
11% accuracy rate with validation dataset
Not submitted to Kaggle due to low score.

Model/Training information:
10 epochs
5 folds
Efficientnet-B4
criterion = nn.NLLLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

Principal achievement:

1) Learning how to implement k-folds model training method
2) Learn new weights determination and manipulation technique (eg. what weights to assign to different models' outputs depending on their historical accuracy rates with different predicted isup grades)
3) Learning how to save and load checkpoints for all folds

Individual k-models reached accuracy rates as high as 59% (please ignore formatting error in notebook), but the author was unable to find appropriate weights that would allow a weighted (or unweighted) average to yield more accurate results than could any single model.

6_GLEASON SCALED MODEL

Notebook location: Main repository
Notebook name:

- *6_base_model_no folds_10_g_score_learning.ipynb*

Summary of Step:
The objective of this notebook was to see if accuracy rates would increase by increasing the number of output classes (defined by number of possible Gleason scores, 10 classes) with the hope that it would increase the accuracy when mapping Gleason scores to ISUP scores (6 classes). The reason why this was developed was because the original model was having

difficulty accurately predicting biopsies with ISUP scores 4 and 5. These two ISUP scores, however, can be mapped to 5 different Gleason scores (see image below).

ISUP Prostate Cancer Grade Groups

Grade group	Gleason score	Gleason pattern
1	≤6	≤3+3
2	7	3+4
3	7	4+3
4	8	4+4, 3+5, 5+3
5	9 or 10	4+5, 5+4, or 5+5

Summary of results:
42.7% accuracy rate with validation dataset
Not submitted to Kaggle due to low score

Model / Training information:
10 epochs
Model = Efficientnet-B4
criterion = nn.NLLLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

Principal achievement:

1) Learning that mapping based efficiencies may not necessarily increase accuracy rates. It is safe to assume that the model implicitly takes care of these types of category mappings.

7_LOGITS BASE MODEL AND SUBMISSION (61.0 submission score)

Notebook location: Main repository
Notebook names:

- *7a_base_model_no_folds_Logits.ipynb*
- *7b_base_model_w_logits_-_analysis_ and_submission.ipynb*

Summary of results:
59.2% accuracy rate with validation dataset
61.0 Kaggle submission score (see how score is calculated in the document named "proposal.pdf")

Model / Training information:
30 epochs
Model = Efficientnet-B0
criterion = nn.BCEWithLogitsLoss()

scheduler was cosine with gradual warm up
optimizer = optim.Adam(model.parameters(), lr=learning scheduler)

Principal achievement:

1) Implementing new criterion (BCEWithLogitsLoss) and understanding how it implicitly communicates to the model a sense of continuity in the output categories (tissue biopsy samples have varying but continuous grades of cancer vs. classifying different types of animals is a discrete type of classification)
2) Implementing a learning scheduler
3) Using Efficientnet-B0, which is much lighter and faster to train than Efficientnet-B4, which allowed the author to train over many more epochs and experiment with learning rate schedulers.

8_ARUTEMA METHOD WITH LOGITS BASE MODEL AND SUBMISSION (76.6 submission score)

Notebook location: Main repository
Notebook names:

- *8a_base_model_wLogits_retrained_Arutema's method.ipynb*;
- *8b_Arutema base model w logits - analysis and submiss.ipynb*

Summary of step:
The winning team of the competition used a method called the Arutema method (see discussion here), which consists of filtering “noisy” dataset items according to how well a previously trained model can predict their ISUP score within a certain threshold.

The training dataset was divided into two categories: images that were predicted by a previously trained model within an acceptable accuracy threshold (keep_df, represented by the blue dots in the chart below) and those that fell outside of the prediction accuracy threshold (remove_df, represented by orange dots in the chart below). A new model was then created and trained using images from the keep_df dataset only. This resulted in an astonishing 97.9% validation accuracy rate and a 95.7% test validation rate (see snippet of algorithm and chart below).

This step was a highly enlightening experience because it shows how the model needs to be taught with clear and concise examples that are easy to interpret and are likely more representative of the ground truth. That way, the model knows which key features to look for, even in new cases where the features may be more difficult to ascertain than others.

Applying the Arutema method helped increase the final submission score from a previous high of 61.0 submission score to 76.6, representing a +15.5 point increase!.

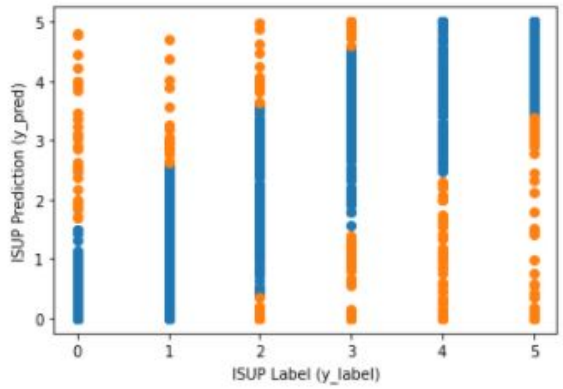
```
# Defining threshold and keeping/removing images accordingly
threshold = 1.6
new_df['thresh_gap'] = [np.abs(x-y) for x, y in zip(new_df['isup_grade'], new_df['pred_y_dec'])]
new_df['passed_thresh'] = [1 if x <= threshold else 0 for x in new_df['thresh_gap']]

new_df_keep = new_df[new_df['thresh_gap']<= threshold]
new_df_remove = new_df[new_df['thresh_gap']> threshold]

new_df_keep.to_csv("sample_keep.csv", sep=",", index=False)
new_df_remove.to_csv("sample_remove.csv", sep=",", index=False)
```

```
new_df.passed_thresh.value_counts()
```

```
1    10342
0      274
Name: passed_thresh, dtype: int64
```

```
plt.scatter(new_df_keep['isup_grade'], new_df_keep['pred_y_dec'])
plt.scatter(new_df_remove['isup_grade'], new_df_remove['pred_y_dec'])
plt.xlabel('ISUP Label (y_label)')
plt.ylabel('ISUP Prediction (y_pred)')
```

```
Text(0, 0.5, 'ISUP Prediction (y_pred)')
```



Summary of results:
97.9% accuracy rate with validation dataset
76.6 Kaggle submission public score, which places me in the 33rd percentile on the public leaderboard (see how Kaggle competition score is calculated in the document named “proposal.pdf”)

Model / Training information:
30 epochs

Model = Efficientnet-B0
criterion = nn.BCEWithLogitsLoss()
scheduler was cosine with gradual warm up
optimizer = optim.Adam(model.parameters(), lr=learning scheduler)

Principal achievement:

1) Understanding and seeing first hand the power of denoising data through the Arutema method. In the end, removing 3% of the dataset resulted in a +16 point increase in the final submission score! Less can certainly mean more when it comes to data wrangling / filtering.
2) Finishing the project knowing that if there were a few more days left, it would be very possible to achieve a significantly higher score by creating a 64x(256x256) pre-tiled image dataset, train new model with the dataset using methods described in this section, and submitting the model for grading.