INTERNSHIP REPORT

Zsolt Dobos

Master Programme: Applied Computational Intelligence

E-mail: zsolt.dobos@stud.ubbcluj.ro

Academic Info Code: 4Y125642

Introduction

My internship is based on the Prostate cAncer graDe Assessment (PANDA) Challenge. This was a public challenge published on the Kaggle platform in 2020. Prostate cancer is one of the most common cancer type among the males in worldwide. The goal of this challenge was to motivate the developers to resolve the classification problem of the prostate tissue biopsies whole-slide images. This challenge was published by two institutions, namely Radbound and Karolinska, and they created a large, labeled dataset with around 11,000 whole-slide images, which became of the largest publicly available prostate cancer dataset.

During my work, I tried to analyze and replicate the submitted notebooks on the Kaggle platform and I expanded the existing solutions with my own ideas and approaches to reach a better accuracy and to reduce the computational complexity which would make the algorithms faster.

Prostate cancer

The first step was to understand the prostate cancer, the method of diagnosing the carcinoma and the Gleason score.

The prostate cancer is the deformation of the glands in the prostate and the appearance of the cancer cells based on the description of the International Society of Urological Pathology (ISUP). The computer vision task is to identify the morphological changes of the glands in the prostate. Based on the level of these abnormalities, we can achieve the stage of the cancer. The ISUP uses the Gleason score for this purpose. Donald F. Gleason created this grading system based only on the morphological features of the tumor. The original system went through on a lot of changes. Because of the development of the imaging technologies and the new results on the histopathological field, the ISUP modified the original Gleason grading system to be more accurate. Despite the efforts of the ISUP, the Gleason grading system will remain subjective, the final score will be dependent on the grader, and this is where the Computer Aided Diagnosis (CAD) systems can help. On the Fig. 1, we can see a summary of the Gleason patterns. We can see well separated healthy glands on the Gleason 1 and washed, unrecognizable and “chaotic” glands on the Gleason 5. For more information about the Gleason patterns, I recommend the [1]. A tissue sample will get the Gleason score based on the two most dominant Gleason pattern, so it is marked as a sum (for example 3+4), where the first one is the primary- and the second one is the secondary pattern on the tissue.

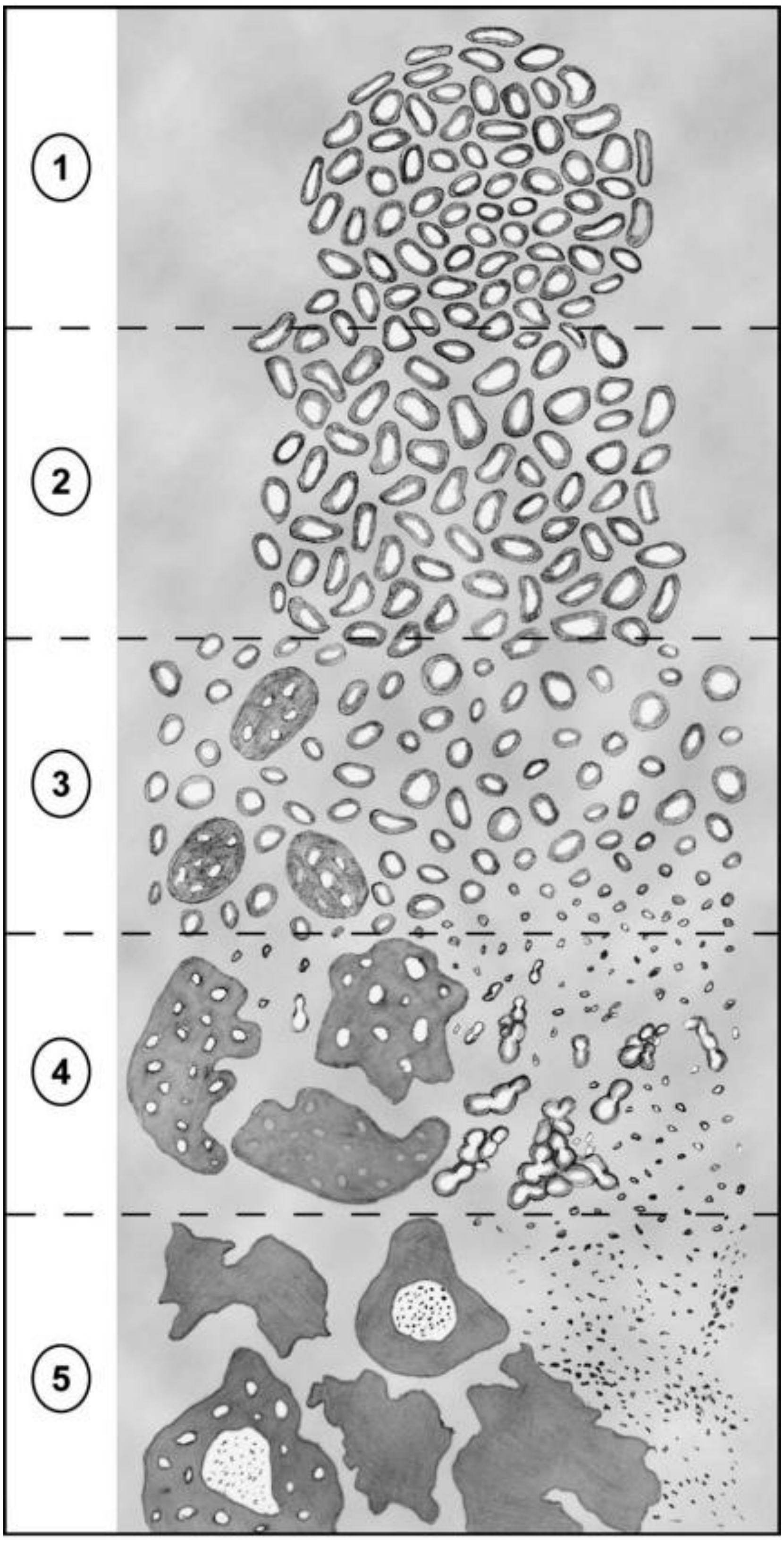

Fig. 1. The Gleason Patterns [1]

Data analysis

The dataset consists of a csv file with the data entries, the images and the masks.

The csv file contains 4 columns: image_id, data_provider, isup_grade and gleason_score. The image_id is the unique ID of the WSIs, the data_provider is one of the two institute, the gleason_score is the described score in the previous section, and the isup_score is a derived score from the gleason_score, which contains only a single number and can be applied better in some implementation.

The shape of the original csv was 10616x4, but I needed to remove 3 entries, because one of them was corrupted file and two of them was missing.

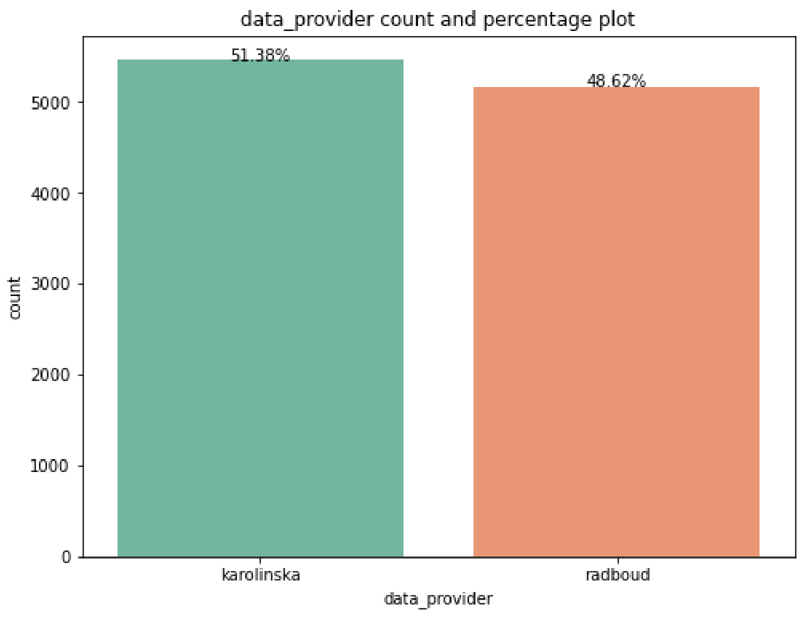

Fig. 2. The distribution of the data by the providers

The two institutes provided almost 50-50% of the data (Fig. 2.) This sounds promising, but on the Fig. 3. we can see that the data is unbalanced by the ISUP grades. I tried to train the models on this data and on a balanced version of the data. We can observe on the Fig. 4., that the institutes provided data in different percentage to the different ISUP grades, but this should give us less problem.

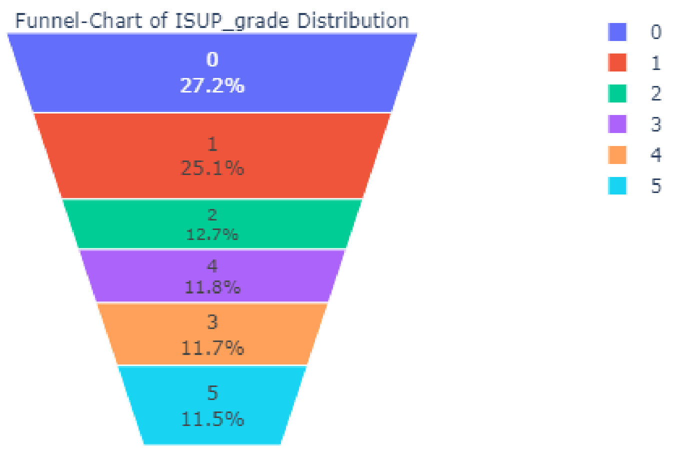

Fig. 3. The distribution of the data by the ISUP score

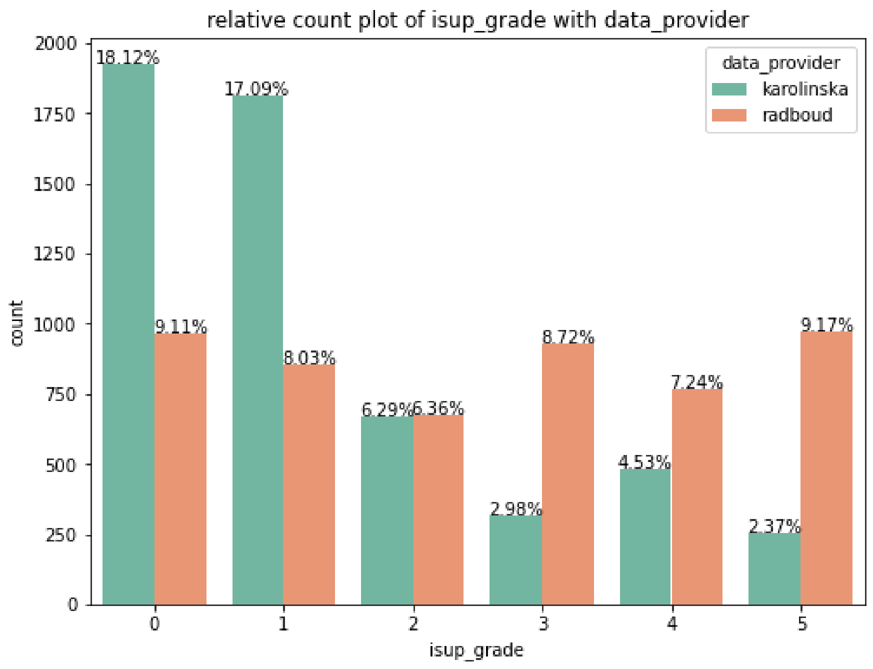

Fig. 4. Relative count plot of ISUP grade with data provider

The masks for the images mark the exact location of the Gleason patterns on the images (Fig. 5.). This can be very useful if we want to create a patch level classification system. During my internship, I tried to create a system which can extract the most relevant patches from the whole-slide image and tries to classify the tissue by running a prediction on this set of patches. This means, I did not need to use the mask images, but in the future works I want to test a patch level classifier too.

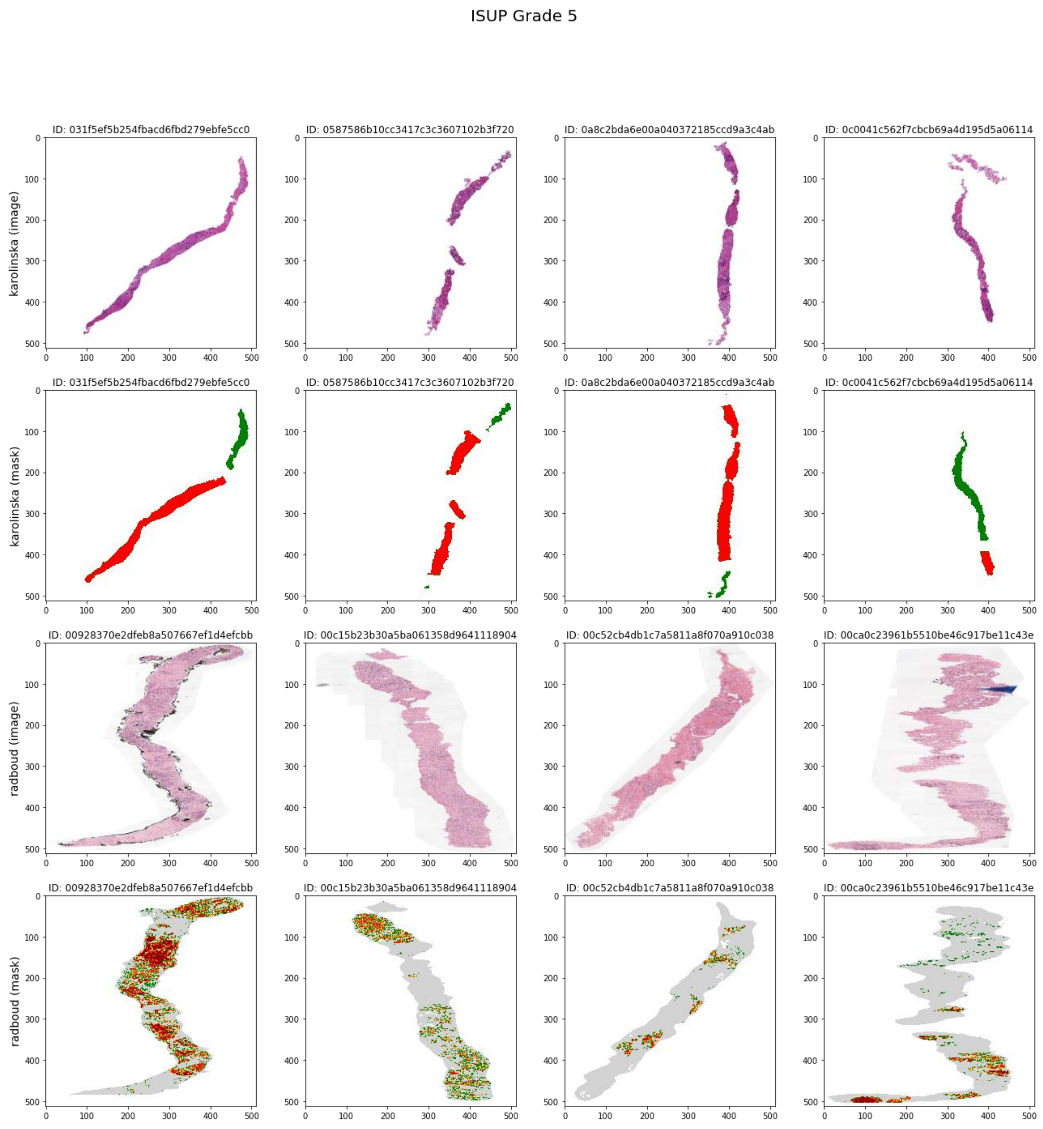
ISUP Grade 5
ID: 031f5ef5b254fbacd6fbd279ebfe5cc0
ID: 0587586b10cc3417c3c3607102b3f720
ID: 0a8c2bda6e00a040372185ccd9a3c4ab
ID: 0c0041c562f7cbcb69a4d195d5a06114
ID: 00928370e2dfeb8a507667ef1d4efcbb
ID: 00c15b23b30a5ba061358d9641118904
ID: 00c52cb4db1c7a5811a8f070a910c038
ID: 00ca0c23961b5510be46c917be11c43e
karolinska (image)
karolinska (mask)
radboud (image)
radboud (mask)

Fig. 5. Masks for WSIs with ISUP Grade 5

Unfortunately, some of the images contains artifacts, as we can see the pen markers on the Fig. 6.

The size of the whole-slide images varies on a large scale, when we develop the preprocessing algorithm, we need to take this into account. On the smaller images, maybe easier to find all the relevant parts, but it possibly does not contain enough information about the prostate. On the other hand, working with the large whole-slide images can be computationally exhaustive and the chance of missing relevant part of the tissue is higher. The algorithm must handle the smaller, the medium and the large samples with the same effectiveness.

Because the size of the dataset is more than 400GB, I developed the algorithms on a small subset. This saved time, but I needed to run the final trainings on the large dataset to achieve good results.

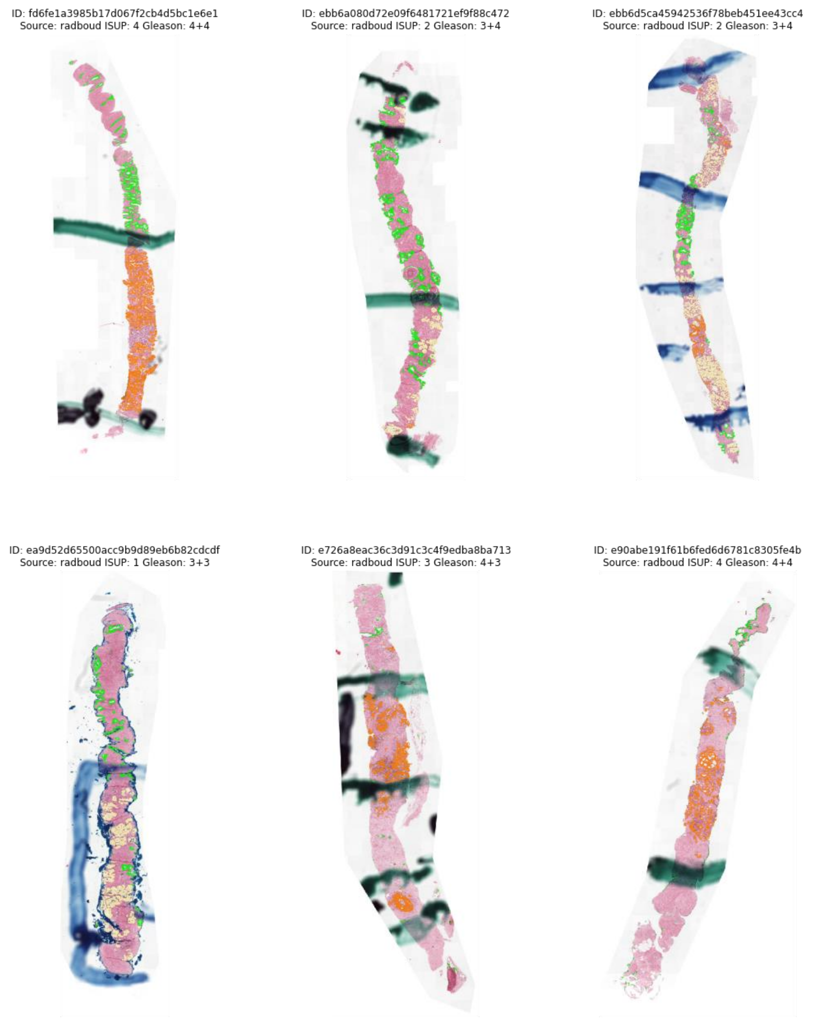
ID: fd6fe1a3985b17d067f2cb4d5bc1e6e1
Source: radboud ISUP: 4 Gleason: 4+4
ID: ebb6a080d72e09f6481721ef9f88c472
Source: radboud ISUP: 2 Gleason: 3+4
ID: ebb6d5ca45942536f78beb451ee43cc4
Source: radboud ISUP: 2 Gleason: 3+4
ID: ea9d52d65500acc9b9d89eb6b82cdcdf
Source: radboud ISUP: 1 Gleason: 3+3
ID: e726a8eac36c3d91c3c4f9edba8ba713
Source: radboud ISUP: 3 Gleason: 4+3
ID: e90abe191f61b6fed6d6781c8305fe4b
Source: radboud ISUP: 4 Gleason: 4+4

Fig. 6. Artifacts on the WSIs

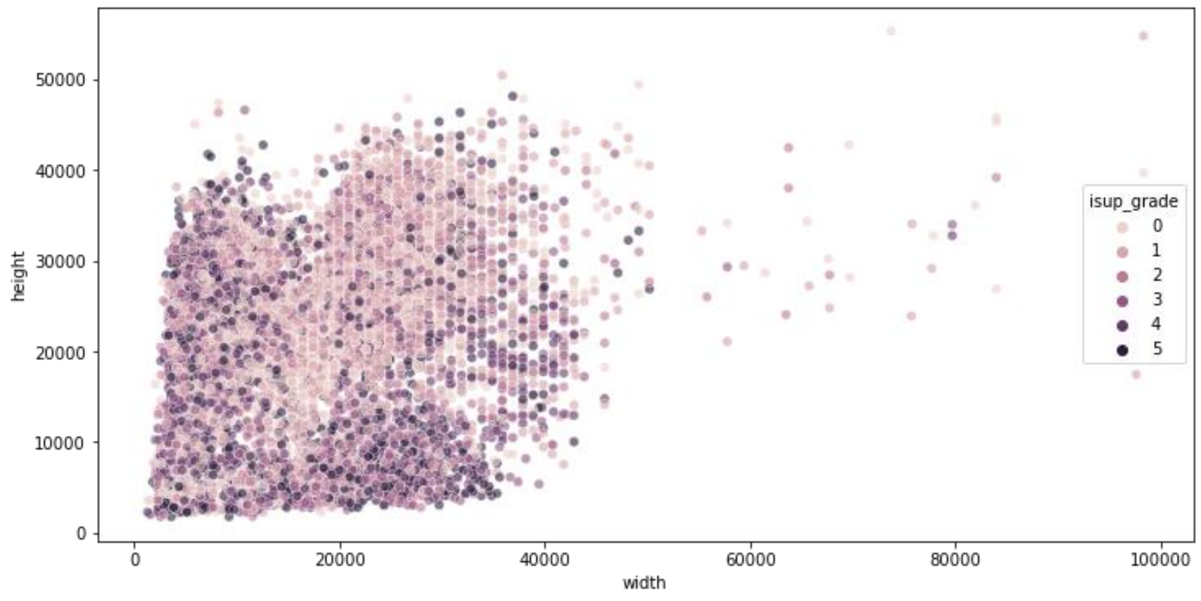

Fig. 7. The sizes of the WSIs

Preprocessing the images

Many solutions tried to create “glued” input images for the neural networks. This means that they extracted the relevant patches from the whole-slide image, then they combined it together and created a new image which is much smaller than the original but can represent the tissue.

The whole-slide image (WSI) provides possibilities to preprocess the image efficiently. Most important thing in the WSI is the image pyramid. This means that the WSI contains the same size image on different resolution, and we can easily walk through the image on the different resolutions and apply computations on the preferred pyramid level (Fig. 8.).

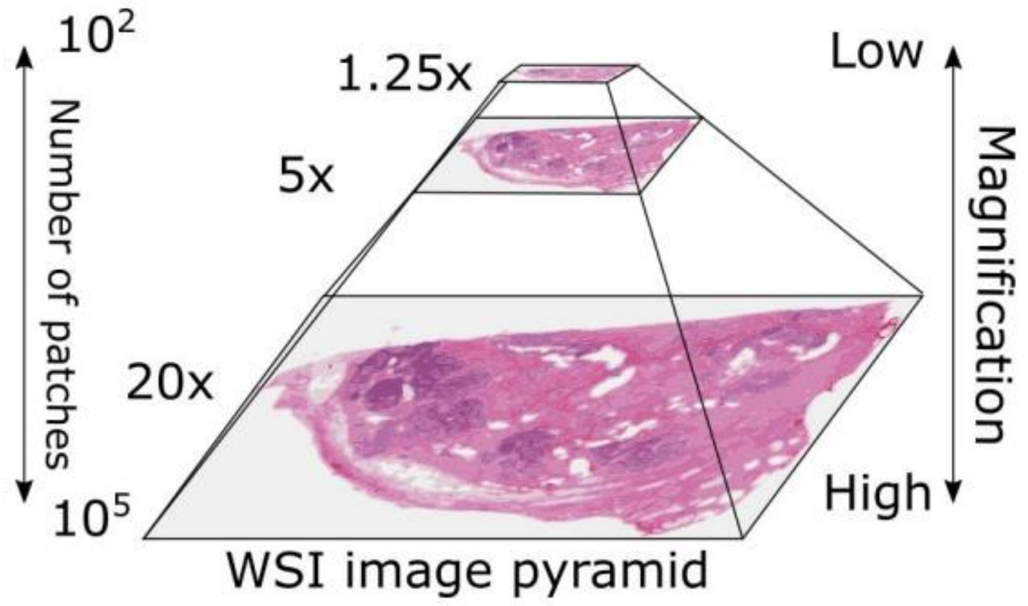

Fig. 8. WSI Image pyramid[2]

OpenSlide is a free and open-source C library used for reading and interpreting digital pathology whole slide images. OpenSlide provides a platform-independent solution for accessing the image data and metadata in a vendor-neutral format. This library supports a wide range of proprietary file formats from various vendors, making it easier for developers and researchers to work with whole slide images without having to worry about file format compatibility. OpenSlide also offers various features like image scaling, region-of-interest extraction, and support for large images, making it a versatile tool for handling digital pathology images. In the Kaggle challenge, many competitors used this library to preprocess the image data. Unfortunately, one part of them was not so effective and they extracted a lot of unusable background parts, another half of them was slow, because they didn’t apply the pyramid feature of the WSI. Therefore, I implemented my own preprocess algorithm, which is based on the WSI pyramid feature. The power of my algorithm is that it does computation on the lower resolution of the tile, and it extracts the highest quality versions.

1. The first step is calculating the mean of the pixel intensities and thresholding the tiles, every tile with lower mean intensities are kept, with this step we can easily filter out the background tiles, because they are fully white.

2. Secondly, we calculate for every tile the standard deviation
3. Thirdly we sort the tiles by the standard deviation
4. Lastly, we return the first *n* tile with the max resolution

To understand the step 2 and step 3, we need to go back to the definition of the Gleason patterns. The higher the Gleason pattern, the poorly formed glands have more rare lumens, and the tile is “flatter”, and this results a lower standard deviation. Theoretically, this means that the algorithm finds the highest Gleason score regions on every WSI. Based on the [1], this is promising, because the currently used Gleason score works in this way too.

I added a data augmentation step to the tile extraction preprocessing part. After we extracted the tiles, we can shuffle and rotate them. Finally, we glue them together. I saved the preprocessed images and ran the preprocessing and the training phase separately, to save time during the training.

The algorithm has two parameters: the size of the tile (it extracts square tiles) and the number of the extracted tiles. The choice of these hyperparameters was heuristically. Unfortunately training the neural networks with every hyperparameter combination would be computationally very exhaustive, so I extracted some sample image with many hyperparameter combination and I choice the size 256 and the number of tiles 6x6, because the human eyes can find easily the Gleason score on these images.

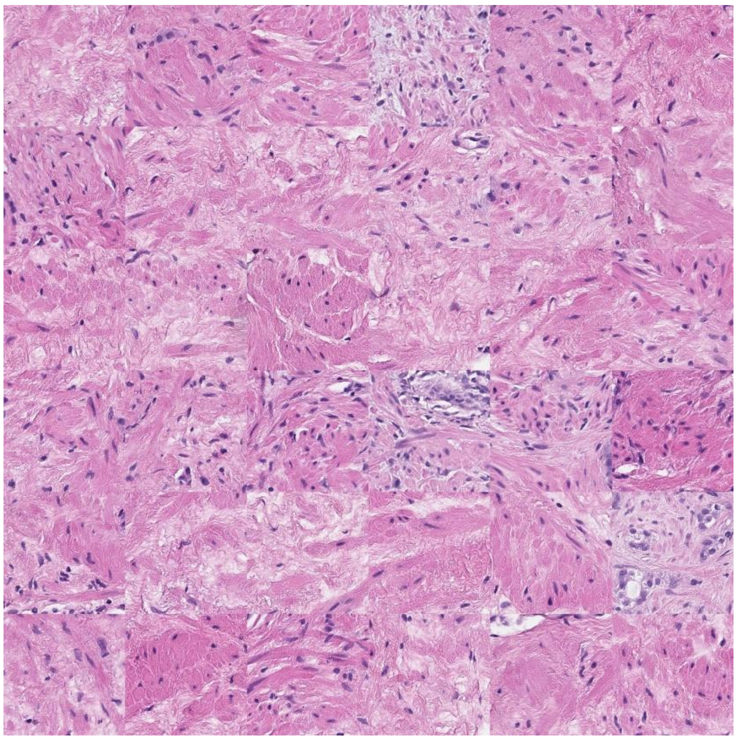

Fig. 9. An example to the extracted image from the WSI with Gleason score 4+4

Training the neural networks

In the first stages I used 4 types of backbone architecture. The VGG16, the DenseNet121, the SEResNext50 and the EfficientNetB2. All models were pretrained on the ImageNet dataset and every trainable parameter was trained during the fine tuning, During the first stages, I trained the models on the small subsets of the dataset. By examining the class activation maps and the results on the small dataset, I decided to train only the SEResNext50 and the EfficientNetB2 on the full dataset. An example of the training on the small dataset (Fig. 10.), the EfficientNet and the SeResNext outperformed the older architectures. The EfficientNet received the best validation loss, 1.55, but it did not improve after a few epochs.

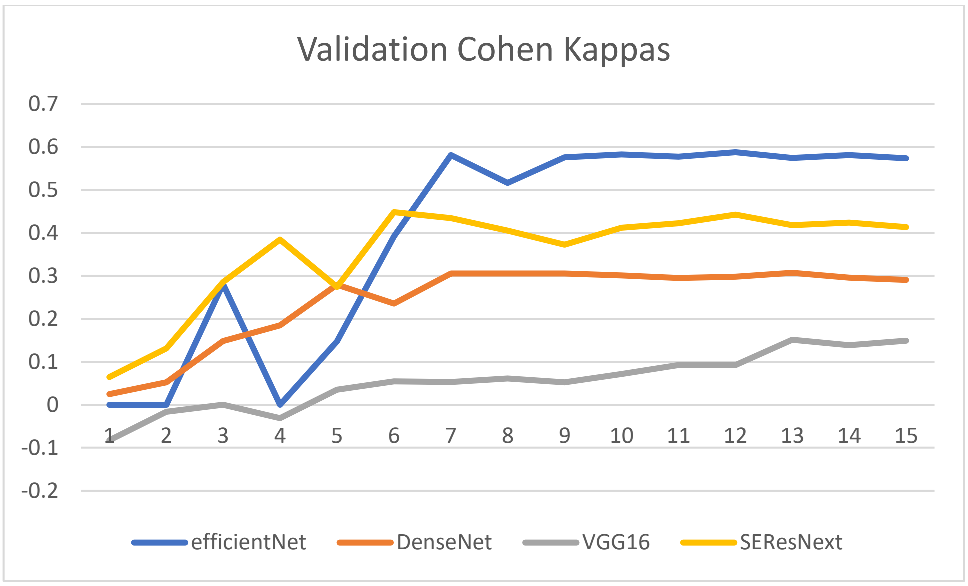

Fig. 10. Cohen Kappas on the 64x5x5 small dataset

For the final experiments, I extracted 36 tiles with size of 256x256 from every WSI and I also augmented every image 4 times. The final dataset contains 53,065 images. I trained every model two times. Firstly, with the unbalanced data and secondly with balanced data. The balanced dataset contains 30,720 images for training and 6000 images for validation. Every epoch was around 40 minutes length during the training of the EfficientNetB2 and around 1 hour 10 minutes during the training of the SEResNext50.

As we can see on the Fig. 11. and Fig. 12., both models started to overfit on training data after few epochs on the balanced dataset and because of this, I trained the models only for a few epochs on the unbalanced dataset.

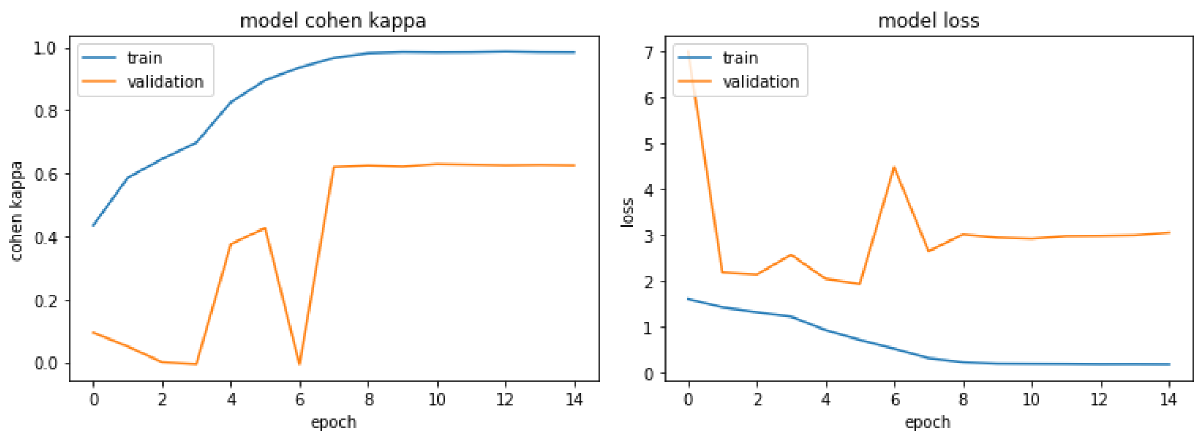

Fig. 11. EfficientNet training on balanced dataset

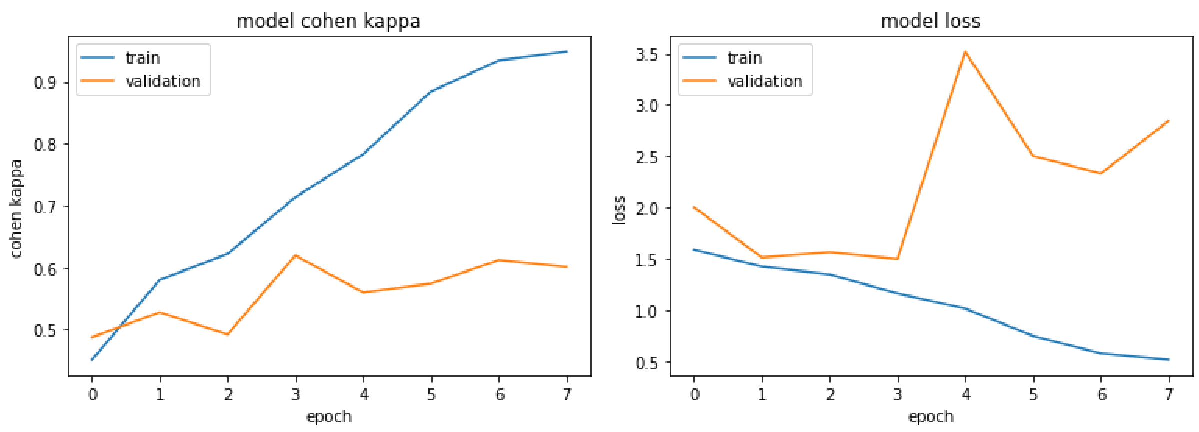

Fig. 12. SEResNext on balanced dataset

	SEResNext50	EfficientNetB2
unbalanced	0.673	0.718
balanced	0.624	0.642

Table 1. Quadratic weighted kappas on the validation dataset

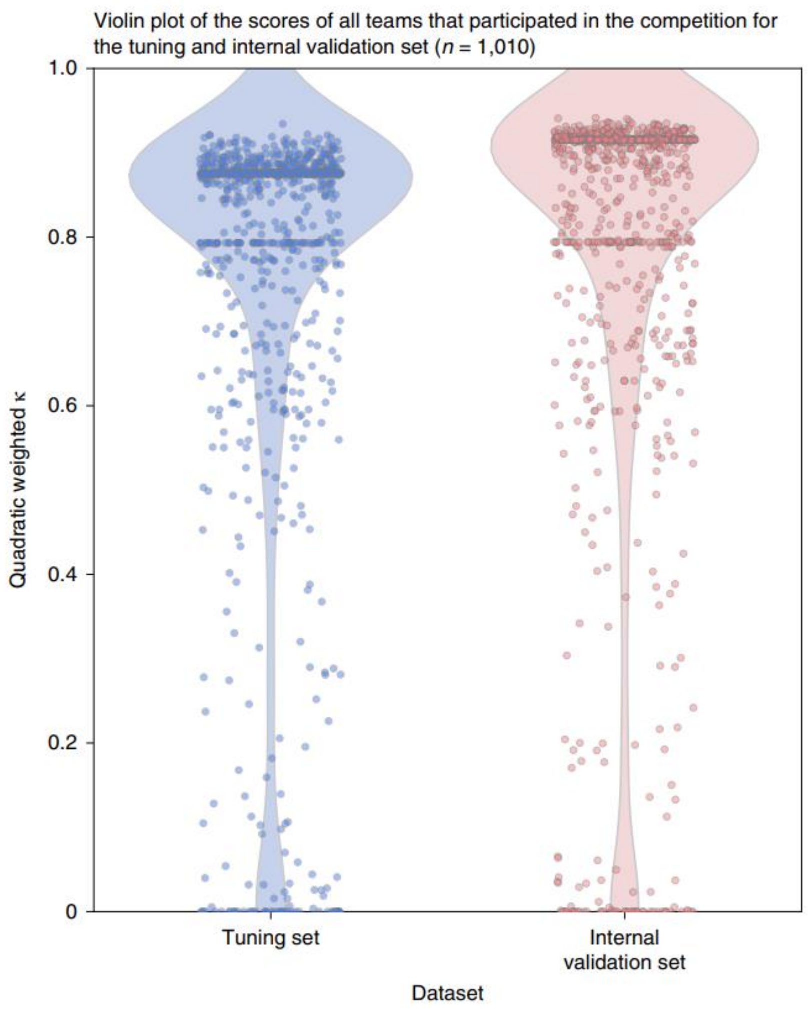

Fig. 13. The results from the original challenge [3]

We can conclude that the performance of these models almost reached the average results by comparing the Table 1. and the Fig. 13., however I could not use the same validation data, because it was not public.

By analyzing the confusion matrices, we can see that both models performed well on the 0 and the 5 ISUP grades but struggled with the 2 and 3 ISUP grades. It is no so surprising because these two grades are on the middle and it is easier to miss them.

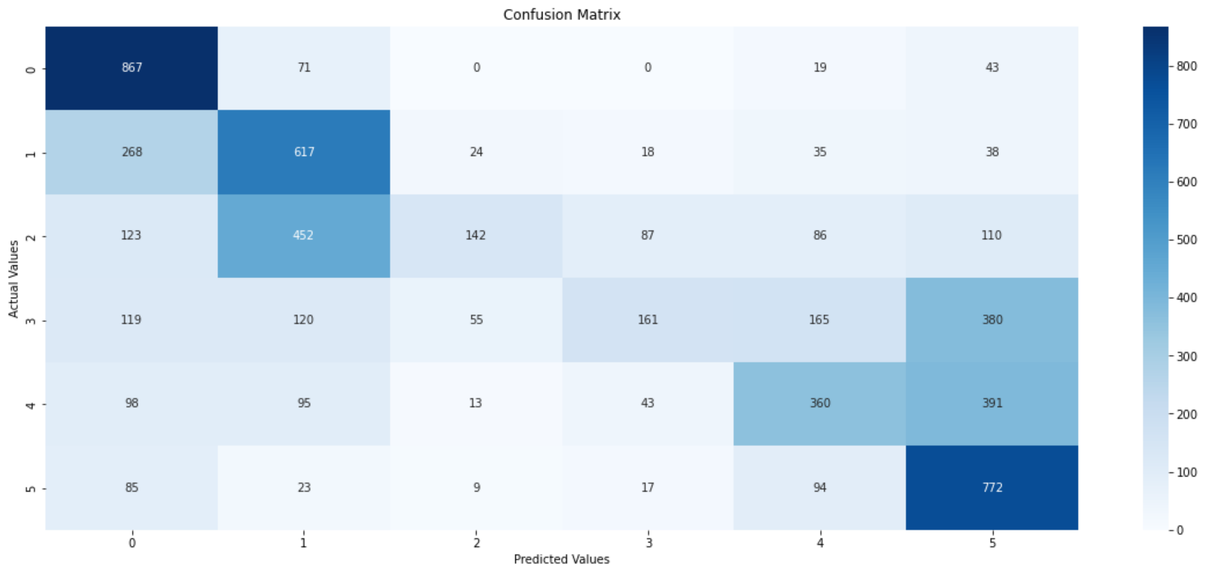

Fig. 14. Confusion matrix, unbalanced SEResNext50

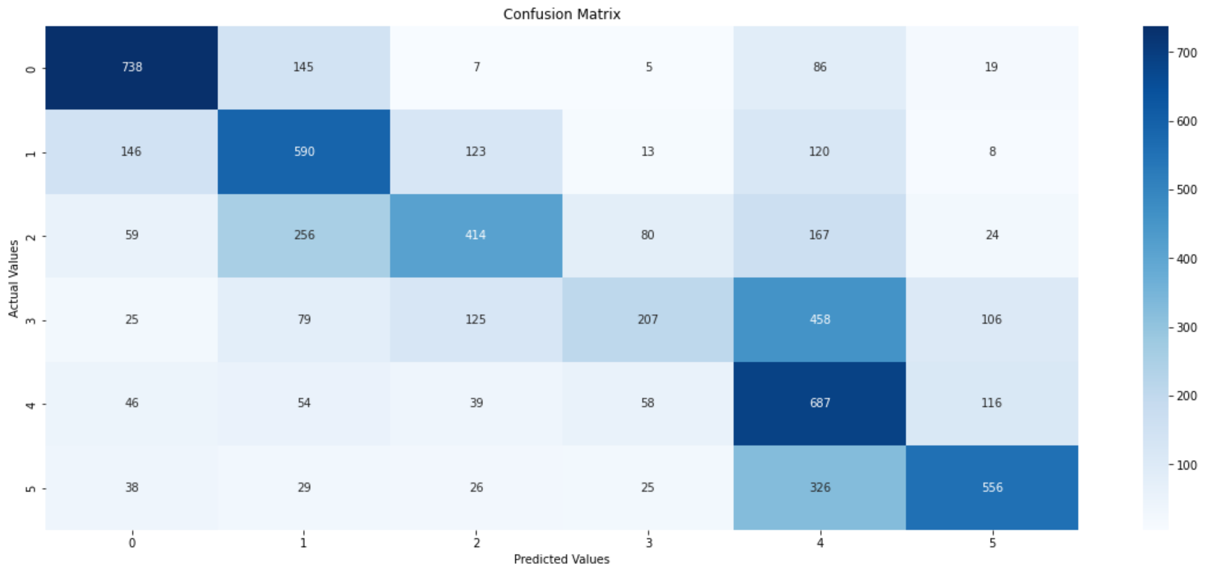

Fig. 15. Confusion matrix, unbalanced EfficientNetB2

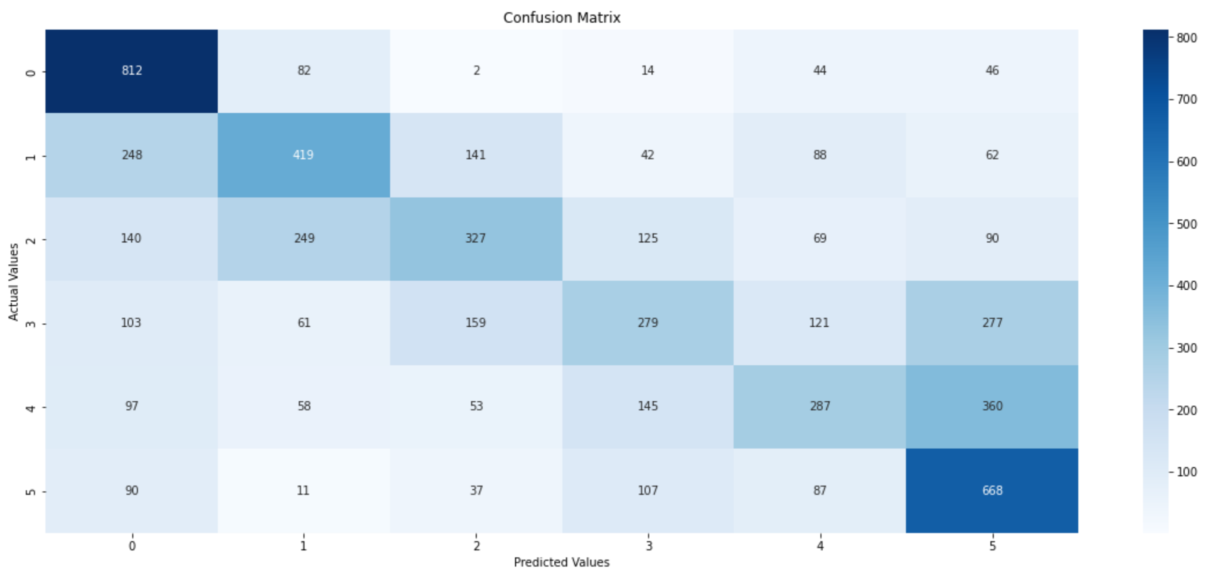

Fig. 16. Confusion matrix, balanced SEResNext50

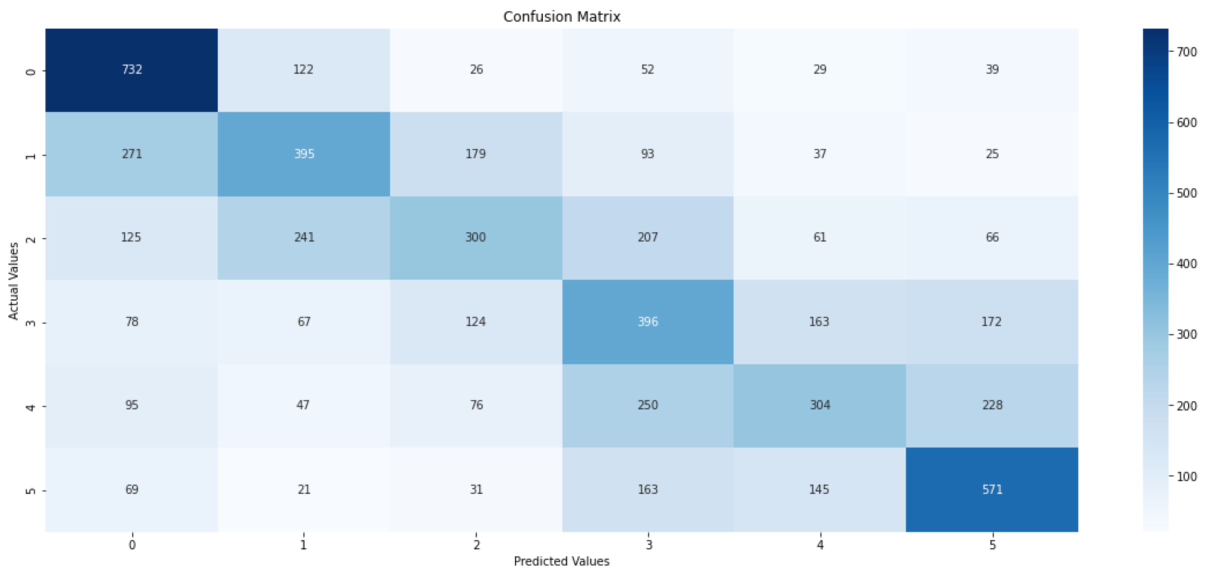

Fig. 15. Confusion matrix, balanced EfficientNetB2

On the Fig. 18., we can see the Class Activation Map of the EfficientNetB2. The images in the first row are the true images with the true labels, the images in the second row shows us which regions of the images were responsible for these predictions. As we can see under activation and over activation both can lead to failure.

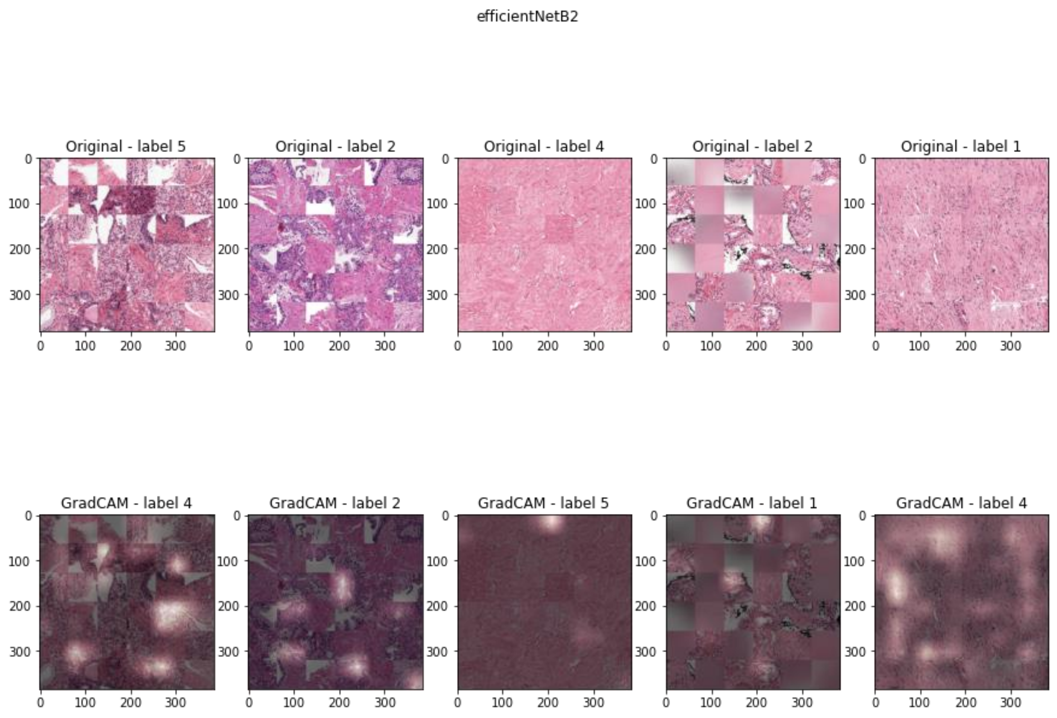

Fig. 18. Class Activation Map of the EfficientNetB2

Conclusion

In conclusion, my pipeline requires further improvements, but there are many parts which can be switched. We could change the patch finding algorithm, the patch sizes, we could classify only the single patches or forward them without “gluing” to the network. Also, we could change the backbone architecture. But the results also showed that there are good components in the algorithm because it didn’t fail completely.

References

[1] Epstein, Jonathan I., et al. "The 2005 International Society of Urological Pathology (ISUP) consensus conference on Gleason grading of prostatic carcinoma." The American journal of surgical pathology 29.9 (2005): 1228-1242.
[2] Li, Zheyu, et al. "Computationally efficient adaptive decompression for whole slide image processing." Biomedical Optics Express 14.2 (2023): 667-686.
[3] Bulten, Wouter, et al. "Artificial intelligence for diagnosis and Gleason grading of prostate cancer: the PANDA challenge." Nature medicine 28.1 (2022): 154-163.