Deep Learning in Medical Imaging 2022 Kaggle Challenge - Team F&A Classification of ISUP grades from Whole Prostate tissue biopsies

Antoine Cadiou* ANTOINE.CADIOU@ENS-PARIS-SACLAY.FR
Fabien Merceron* FABIEN.MERCERON@ENS-PARIS-SACLAY.FR

1. Introduction

Prostate cancer (PCa) is the second most frequent disease in men globally, with over 1 million new diagnosis recorded each year and over 350,000 deaths annually. The development of more precise diagnoses is critical to lower mortality. The grading of prostate tissue samples is used to diagnose PCa. A pathologist examines the tissue samples and grades them using the Gleason grading method. We will describe a method for identifying PCa on slides of prostate tissue samples in our solution, and we will assess the severity of the disease using the largest multi-center dataset on Gleason grading currently available.

The grading process consists of finding and classifying cancer tissue into so-called Gleason patterns (3, 4, or 5) based on the architectural growth patterns of the tumor. After the biopsy is assigned a Gleason score, it is converted into an ISUP grade on a 1-5 scale. The Gleason grading system is the most important prognostic marker for PCa, and the ISUP grade has a crucial role when deciding how a patient should be treated. There is both a risk of missing cancers and a large risk of overgrading, resulting in unnecessary treatments. However, the system suffers from significant inter-observer variability between pathologists, limiting its usefulness for individual patients. This variability in ratings could lead to unnecessary treatments, or worse, missing a severe diagnosis.

The goal of this challenge is to predict the ISUP Grade using only Histopathology images. To do this, we will need to handle Whole Slide Images as huge gigapixel images and deal with the limited number of patients provided in the train set.

2. Dataset overview

- **Training set**
  It is composed of 340 gigapixel images in a specific '.tiff' format. Because of their size, we will not use a whole 'end to end' solution because we cannot just load one after the other these images in the pipeline and process them. We explain how we faced this challenge in section 3.1. For each of these images (almost, expected 2 of them), we know their segmentation masks (based on the Gleason patterns score). Finally, we know their ground-truth ISUP grade, as well as their provider ('karolinska' or 'radboud'). Besides, classes (ISUP grades) are not very imbalanced.
- **Test set**
  It is composed of 86 gigapixel images in the same format as the ones in the training-set. Nevertheless, we don't know their segmentation masks.

* Contributed equally

3. Our Solution

3.1. Pre-processing & "Patch-based" dataset creation

First, we had to deal with the huge gigapixel images since it is impossible to directly load them in a DataLoader for instance. We have created a dataset of patches where patches have been extracted as follow:

- Given a full slide image, of shape (H, W), first we down-sampled it into (H/2, W/2) because of a lack of time and memory.
- From our down-sampled image, we split it into all non-overlapping 384*384 patches.
- We randomly chose 50 patches in the ones that contained at least more than 30% of prostate tissue.

We repeated this operation for all images and their corresponding masks (because we need to have the masks of patches too).
Finally, our 'patchified' train-set is composed of $\approx 50 * 338 = 16900$ tissue patches & 16900 tissue masks patches.
The operation is the same for the slides in the test-set, so our 'patchified' test-set contains $\approx 50*86 =$ 4300 tissue patches and 0 tissue mask patch.

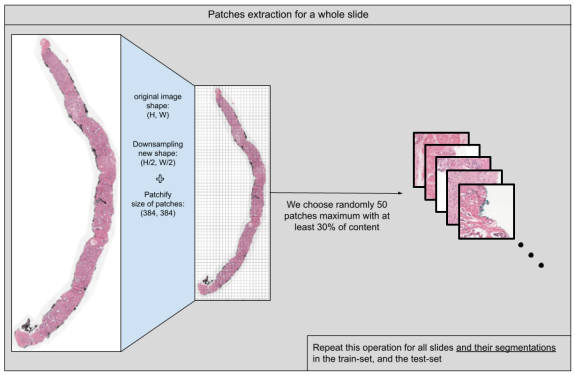


Figure 1: Patchified dataset creation

3.2. Segmentation

Given a tissue patch, we would like to have a model capable to create the associated mask in which gleason patterns are detected. To do this, we first thought about basic models such as UNet or nnUNet as seen in the course, but we have finally used a Vision-Transformer based model that works exactly like a UNet: the encoder is made from a ViT_base_16 and the decoder is simply based on interpolations followed by 2D Convolutions. The complete architecture of this model is presented in appendix 5, and the original paper is this one: (Ranftl et al., 2021)
Initially, we were using a simple Cross-Entropy Loss, but we have implemented our own loss since we have observed that classes are graded (they are not 'independant' in the sense that if a pixel's gleason score is 4 and if we classify it as a 1, it should be more penalized that if we classify it as a 3 for instance), so we use an L1 Loss pixel-wise.
Also, we need to weight the error L1 because gleason's regions are hugely imbalanced in the ground-truth masks, and we want to give importance to misclassification between under-represented classes

such as the 4th or the 5th. So this is the main motivation for our Weighted L1 - CE Loss:

$$\mathcal{L}(y, \hat{y}) = \left[\frac{1}{patch\_size} \sum_{c \in classes} (\mathbb{1}_{y=c} |y - \hat{y}| * W_c) \right] + \mathcal{L}^{CE}(y, \hat{y}) \quad (1)$$

Where y is the ground truth mask (an image of size 384*384) and $\hat{y}$ is the output mask given by our model.
$classes = [0, 1, 2, 3, 4, 5]$ are the possible gleason patterns.
$patch\_size = 384 \times 384$ and it is used to take the mean of all penalized predictions for each pixels.
W is an array of size $\#classes = 6$, and $\sum_c W_c = 1$
$\mathcal{L}^{CE}(y, \hat{y})$ is the cross entropy loss.

This model has been trained with and Adam optimizer, where learning_rate $= 3.10^{-4}$, during 10 epochs and with a batch_size of 8. The evolution of the training loss and the validation loss is visible on figure 6. We can see that our model converged after a few epochs. We can also have a look at the logs of the training on the wandb website: https://wandb.ai/antocad/dlmi_fod/runs/3hsvyhsc

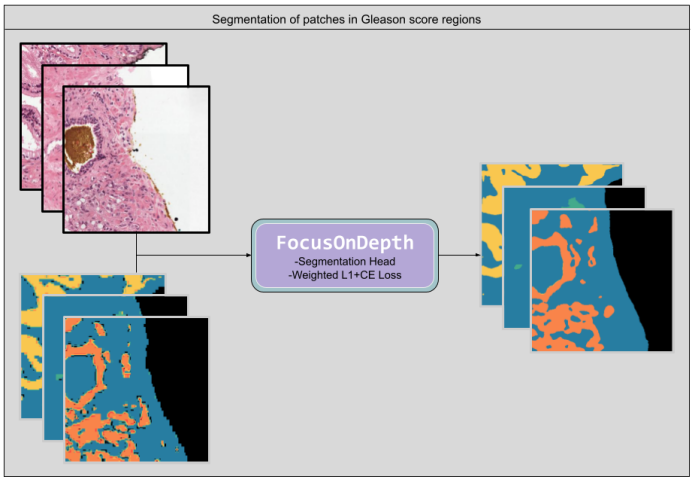


Figure 2: Segmentation model

3.3. Classification

Once we have our masks with gleason scores' regions, it could be interesting to use them to classify a whole slide to an ISUP grade between 0 and 5 (the classification pipeline is presented in figure 3).

The first problematic was to find a way to 'merge' the patches from a same slide together. In fact, this is not that easy because we have lost the natural order of the patches so we are unable to reconstruct the full mask image. Our solution, based on that paper (Nagpal et al., 2019), consists in counting the number of pixels for each Gleason value and to sum these values for all patches in one slide. So that, given a slide, we can extract 6 features (#number of pixels where gleason = 0, 1, 2, 3, 4 & 5). We repeat this operation for each slide in the train/test-set and we build our dataframe. After that, we apply first a min-max normalization by rows, and finally a standard normalization by columns. In addition, we add 2 features that are the data_provider columns in a one-hot-encoded format.

We will train diverse classifiers (namely : a KNN, a Random Forest, a SVM and a MLP), the results are shown in Table 1. Hyperparameters such as *max_depth* for RandomForest, k for KNN or *kernel* for SVC have been selected with a grid-search. For the SVM, we have applied a polynom of degree 2 to our rows, the main idea behind it was to use the 'kernel trick'. Finally, we tried to ensemble these models. To do this, we have multiplied probabilities returned by each model with a given weight vector :

$$probs_{ensemble} = \begin{bmatrix} W_{RandomForest} & W_{SVC} & W_{NN} \end{bmatrix} \begin{bmatrix} probs_{RandomForest} \\ probs_{SVC} \\ probs_{NN} \end{bmatrix}$$

$$\hat{y} = argmax(probs_{ensemble})$$

Where $probs_{...}$ is a vector of size $(1, \#classes = 6)$, $\sum W_i = 1$ and bests W_i have been found using a grid-search. The selected weights are: $W_{RandomForest} = 0.3$, $W_{SVC} = 0.5$, $W_{NN} = 0.2$ and $W_{Knn} = 0$

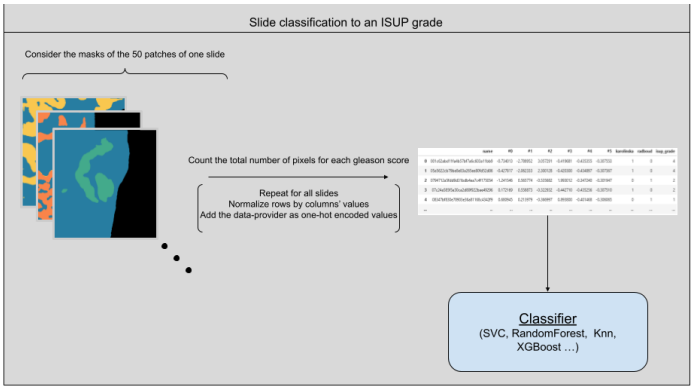


Figure 3: Classification model

4. Results

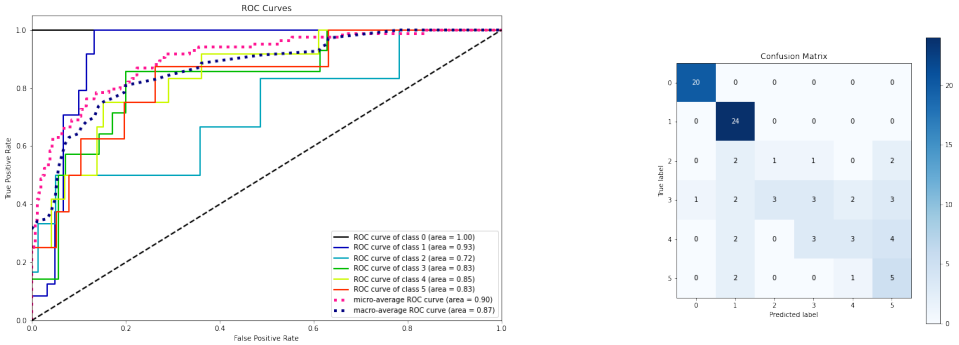
Since the test-set is only composed of 86 images (and the public score on Kaggle is obtained on 43 predictions), all the results presented in the following table should be carefully interpreted because of the overfitting risk which is strong.

The Cross-validation score has been obtained by training our model on 80% of the train-set and evaluating it on the 20% remaining, and we repeated this operation 5 times, with a different validation-set at each time.

Considering the 5th model (the ensemble), we can have a look at the confusion matrix after having trained our ensemble on 70% of the train-set and evaluating it on the rest. As we can see on figure 4, our model has difficulties to separate ISUP scores between {2,3,4,5}. However it performs well to say if a slide is benign.

	Model	Cross_validation score (5 folds)	Kaggle public score
1	KNN (k=5)	0.759	0.874
2	RandomForest (max_depth=7)	0.837	0.948
3	SVM (kernel=rbf + we have applied a polynom of degree 2 on each row)	0.811	0.941
4	Shallow MLP	0.835	0.941
5	Ensemble (0.3 * **2** + 0.5 * **3** + 0.2 * **4**)	0.866	**0.956**

Table 1: Results (scores ROC-AUC)



(a) ROC Curves for each class

(b) Confusion matrix

Figure 4: Graphics obtained on the validation-set ($30\% \approx 100 slides$)

5. Conclusion

To conclude, we can say that our solution obtains rather good results. Our pipeline is based on 3 distinct elements: the creation of a dataset of patches that we have selected in a very interesting way, an innovative segmentation model that uses visions transformers with a loss that we have thought for this problem, and finally, a set of various classifiers (Random Forest, Linear regression, ...) that allows us to obtain a ROC_AUC score of 95.6% and to position us at the 1st place of the challenge (on the public score).

References

Kunal Nagpal, Davis Foote, Yun Liu, Po-Hsuan Cameron Chen, Ellery Wulczyn, Fraser Tan, Niels Olson, Jenny L. Smith, Arash Mohtashamian, James H. Wren, Greg S. Corrado, Robert MacDonald, Lily H. Peng, Mahul B. Amin, Andrew J. Evans, Ankur R. Sangoi, Craig H. Mermel, Jason D. Hipp, and Martin C. Stumpe. Development and validation of a deep learning algorithm for improving Gleason scoring of prostate cancer. *npj Digital Medicine*, 2(1):1–10, June 2019. ISSN 2398-6352. doi: 10.1038/s41746-019-0112-2. URL https://www.nature.com/articles/s41746-019-0112-2. Number: 1 Publisher: Nature Publishing Group.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *CoRR*, abs/2103.13413, 2021. URL https://arxiv.org/abs/2103.13413.

Appendix A. FocusOnDepth(DPT) model architecture

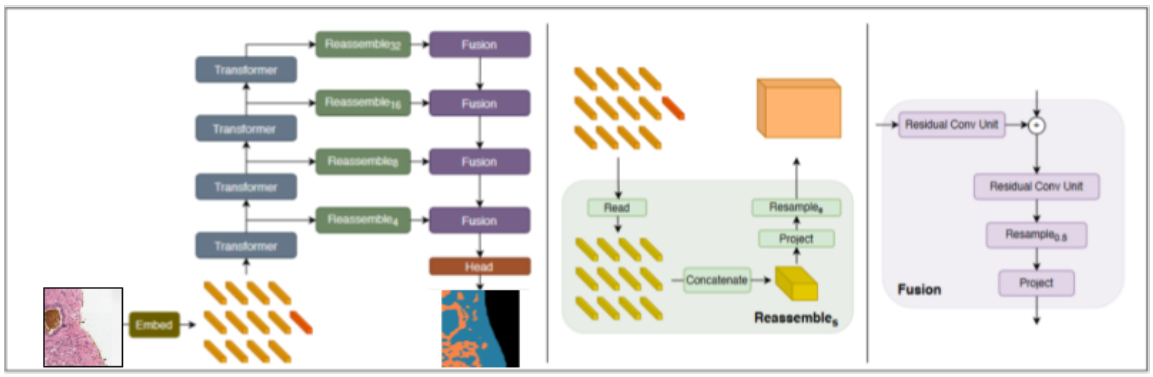
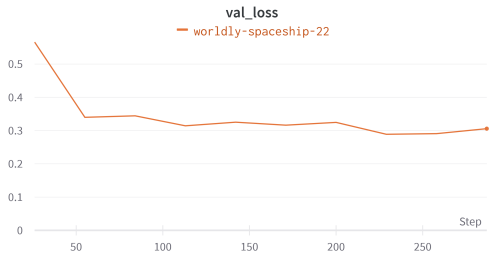


Figure 5: Left: Architecture overview. The input image is transformed into tokens (orange) either by extracting non-overlapping patches followed by a linear projection of their flattened representation (DPT-Base and DPT-Large). The image embedding is augmented with a positional embedding and a patch-independent readout token (red) is added. The tokens are passed through multiple transformer stages. We reassemble tokens from different stages into an image-like representation at multiple resolutions (green). Fusion modules (purple) progressively fuse and upsample the representations to generate a fine-grained prediction. Center: Overview of the Reassembles operation. Tokens are assembled into feature maps with 1 s the spatial resolution of the input image. Right: Fusion blocks combine features using residual convolutional units and upsample the feature maps.

Appendix B. Training/Validation Losses

(a) training loss



(b) validation loss

Figure 6: Loss during training steps