Supplementary Material

1 PARAMETERS FOR THE CODE

1.1 Pre-processing

The parameters for the scripts that extract the patches, with both $grid$ extraction and $multi-center$ extraction:

- -i: an input csv file with the paths of the WSIs.
- -t: the folder where the masks are stored. If the masks are pixel-wise annotated, The folder includes the files, with the name of the WSI and png format suffix. If the masks include only the tissue region: the masks are generated using HistoQC tool. The folder where they are stored includes folders with the name of the WSI.
- -o: the path of the output, where to store the images.
- -p, the amount the threads used in the process.
- -w: the magnification level of the mask (both pixel-wise annotated and with no annotations). The default value is 1.25x.
- : -m: the magnification level selected. If the method used to extract the patch is the grid extraction, then the scale is a single value; otherwise, if the method to extract the patches is $multi-center$ extraction, is it a list.
- -s: the size of the patches, in term of pixels
- -x: the percentage of tissue pixels within a patch to be selected
- -y: the stride between two close patches

1.2 Scale detector

The parameters of the scripts used to generate the input data (Create_csv_from_partitions.py, Create_csv.py) are the following:

- -i: the input folder where the partitions file or the file with the list of images are stored.
- -p: the folder where the patches are stored.
- -o: the path of the folder where to store the csv files generated
- -m: the magnifications wanted to be put in the csv. They will be used to train the model.

The parameters of the scripts used to train (Train_regressor.py) and test (Test_regressor.py) the models:

- -i: the folder where the input data are stored.
- -o: the folder where to store the model weights, some hyperparameters and the metrics file.
- -c: the CNN pre-trained to use (the CNN used in this paper is a ResNet34).
- -verbose: if the script must be verbose or not.
- -m: the magnifications used as classes

The parameters to use the detector as a module (regressor.py):

- -i: the input data. The input data can be of three different types: the path of a folder that includes the patches, the path a single patch or a csv file, that includes the path of the patches.

- -p: the path of the model trained.
- -b: batch size
- -m: the magnifications wanted to be put in the csv. They will be used to train the model.

1.3 Multi-scale CNN for classification

The parameters of the scripts used to generate the input data (Generate_csv_single_scale.py, Generate_csv_multicenter.py, Generate_csv_upper_region.py) are the following:

- -i: the input folder where the partitions file or the file with the list of images are stored.
- -p: the folder where the patches are stored.
- -o: the path of the folder where to store the csv files generated
- -m: the magnifications wanted to be put in the csv. They will be used to train the model.

The parameters of the scripts used to train (Fully_supervised_training.py) and test (Fully_supervised_testing.py) the single-scale CNNs:

- -n: the number of classes.
- -i: the folder where the input data are stored
- -o: the folder where to store the model weights, some hyperparameters and the metrics file.
- -c: the CNN pre-trained to use (the CNN used in this paper is a ResNet34).
- -verbose: if the script must be verbose or not.
- -m: the magnification level chosen.
- -e: the number of epochs
- -f: the structure of the classifier. False means that the features are directly linked with the output probabilities, while True means that there is an intermediate fully connected layer.
- -lr: the learning rate
- -optimizer: the optimizer to use

The parameters of the scripts used to train (Fully_supervised_training_combine_features_multi.py, Fully_supervised_training_combine_probs_multi.py) and test (Fully_supervised_testing_combine_features_multi.py, Fully_supervised_testing_combine_probs_multi.py) the multi-scale CNNs:

- -r: the parameter to choose the model training variant: $multicenter$ means that only one loss function is optimized, while $upper\_region$ means the n+1 loss functions are optimized.
- -n: the number of classes.
- -i: the folder where the input data are stored.
- -o: the folder where to store the model weights, some hyperparameters and the metrics file.
- -c: the CNN pre-trained to use (the CNN used in this paper is a ResNet34).
- -verbose: if the script must be verbose or not.
- -m: the magnification levels chosen.
- -e: the number of epochs.
- -f: the structure of the classifier. False means that the features are directly linked with the output probabilities, while True means that there is an intermediate fully connected layer.
- -lr: the learning rate.

- -optimizer: the optimizer to use.

2 EXPERIMENTAL SETUP

2.1 Pre-processing

Both the pre-processing methods are tested on several tissue datasets, with the same hyperparameters: the same input parameters, the same tissue masks (both with and without pixel-wise annotations) and the same hardware architecture. The input parameters involve the patches features, in terms of patch size (224 x 224 pixels), stride to build the grid (no stride is used), percentage of pixels tissue within a patch to be selected (70%). The methods are tested on three datasets (colon, prostate, lung) to show that they work with WSIs from different tissues. The methods work with tissue masks that include pixel-wise annotations and only tissue. The hardware infrastructure contains 56 CPU cores (Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz), that can host two threads per core.

2.2 Scale detector

The detector is trained with the colon dataset and it is tested in patches that come from three tissues. The training set includes up to 1.5 million patches from seven magnification levels (5x, 8x, 10x, 15x, 20x, 30x, 40x), extracted from a subset of the colon dataset. The CNN is the implementation of the ResNet34, proposed in PyTorch. The model is pre-trained on Imagenet, therefore the classifier head is modified to have as many outputs as the variables to estimate (in this case, one). The network is trained for 10 epochs, with batches of 128 samples, a learning rate of 0.001 and Adam as the optimizer. Class-wise data augmentation is applied during the training, using Albumentations library (Buslaev et al., 2018). The patches are randomly (probabilistic rate of 0.5) horizontally or vertically flipped, rotated and colour augmented. The training is repeated three times, to limit the non-deterministic effects introduced by the stochastic gradient descent on the results. The average and the standard deviation are reported for each of the metrics. The training set includes a subset from the colon dataset (up to 1 million patches). The testing set includes three partitions: a subset of patches from colon dataset, the prostate dataset and the lung dataset. The partition of the colon dataset includes up to 650k patches, the prostate one up to 1.5 million patches and the lung one up to 3 million.

2.3 Multi-scale CNN for classification

Two multi-scale CNNs architectures and optimization variants (and the single-scale CNN with which they are compared) are trained and tested with the same CNN architecture as the backbone, the same hyperparameters, the same data augmentation and the same dataset. All the network have a ResNet34 model (pre-trained on ImageNet) as backbone architecture, proposed in PyTorch. In the multi-scale CNNs architectures, each of the branches is designed with this architecture, while in the single-scale CNN it represents the entire network. The model is pre-trained on Imagenet, therefore the classifier head is modified to have as many outputs as the number of classes (in this case, five classes). The number of classes is a parameter for the scripts. Each network is trained for 10 epochs, with batches of 64 samples, a learning rate of 0.001, a decay rate of 0.0001 and Adam as the optimizer. In the multi-scale optimized with n+1 loss functions, no weights are applied to the losses. Class-wise data augmentation is applied during the training, using Albumentations library (Buslaev et al., 2018). The patches are randomly (probabilistic rate of 0.5) horizontal or vertical flipped, rotated and colour augmented. The training set includes 148 WSIs, with patches from three magnification levels (5x, 10x, 20x). It is split into training, validation and testing partitions, with a proportion of 80/20/20. The partitions are then pre-processed to generate the input data (as mentioned in Section 2.3). The split is repeated five times, to limit the non-deterministic effects introduced by the stochastic gradient descent on the results. The average and the standard deviation are reported for each of the metrics.

2.4 Multi-scale CNN for segmentation

HookNet is trained to segment structures from two tissue, using five different magnifications levels and it is compared with five individuals U-Net. HookNet is trained on breast and lung WSIs. In the breast use-case, the model is trained to segment six tissue types: ductal carcinoma in-situ (DCIS), invasive ductal carcinoma (IDC), invasive lobular carcinoma (ILC) benign epithelium (BE), Other and Fat. The breast dataset is split in training (50 WSIs), validation (18 WSIs) and testing (18 WSIs) partitions. In the lung use-case, the model is trained to segment four tissue types: tertiary lymphoid structures (TLS), germinal centers (GC), Tumor and Other. The lung dataset is split in training (12 WSIs), validation (6 WSIs) and testing (9 WSIs) partitions. The HookNet is fed with a combination of patches from different magnification levels. In the breast use-case, the model is trained using patches from 20x for the target branch and patches from 1.25x or 5x for the context branch. In the lung use-case, the model is trained using patches from 20x for the target branch and patches from 5x for the context branch. The magnification levels used to train the U-Net models are: 1.25x, 2.5x, 5x, 10x, 20x. The CNN is compared with five U-net (one for each of the magnification levels). The architectures share some of the hyperparameters used for training. The models are trained for 200 epochs with patches of size 284x284x3, in batches of 12 samples. The learning rate chosen is 0.000005 and the Adam optimizer is used. Data augmentation is applied during the training, using spatial transformations and the stain transformation proposed by (Tellez et al., 2019). HookNet is trained optimizing two loss functions, to regularize the contribution of the context and the target branch. The loss functions are pixel-wise categorical cross-entropies. A more accurate description of the network and the experimental setup is available in (**?**).

3 METRICS

Two types of metrics are used to evaluate the performance of the tools: regression metrics and classification metrics. The regression metrics are used to assess the performance of the scale detector model. They are the coefficient of determination (R-squared or R^2) and the mean-squared error (MSE). The R^2 metric measures the proportion of the variance between a dependent variable (in this case the estimated scale) and an independent variable (in this case the input patch). The metric values vary between -inf and 1. The higher the value, the better the model estimates a value. The MSE metric measures the average of the squared errors (the difference between the ground truth and the estimations) of the model. The metric varies between 0 and inf. The lower the error, the better the model estimates a value. The classification metrics are used to assess the performance of the regression model, the classification models and the segmentation model. The metrics are the balanced accuracy, the Cohen's $kappa$ (McHugh, 2012) and F1 score. The balanced accuracy measures the average of the single accuracies, individually evaluated on each of the classes. The metric varies between 0 and 1. The higher the value, the better the model predicts a class. The Cohen's $kappa$ measures the agreement between raters (in this case between the ground truth and the predicted classes). The metric varies between -1 and 1. The higher the value, the better the model predicts a class. The F1 score is the average between the precision and the recall. The precision is the proportion of correctly predicted positive values among all the positive values. The recall is the proportion of correctly predicted positive values among all the values that should be classified as positive. The F1 score varies between 0 and 1. The higher the value, the better the model predicts a class. The classification metrics are applied to the scale detector, although it is a regression model. The estimations are discretized to the closer magnification level, among the seven considered in the dataset.

4 SOFTWARE DEPENDENCIES

- python 3.6.9

- openslide-python 1.1.1
- openslide 3.4.1
- numpy 1.18.5
- pillow 6.2.1
- opencv 3.4.1
- scipy 1.4.1
- scikit-learn 0.23.1
- scikit-image 0.15.0

REFERENCES

Buslaev, A., Parinov, A., Khvedchenya, E., Iglovikov, V. I., and Kalinin, A. A. (2018). Albumentations: fast and flexible image augmentations. *ArXiv e-prints*

McHugh, M. L. (2012). Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica* 22, 276–282

Tellez, D., Litjens, G., Bándi, P., Bulten, W., Bokhorst, J.-M., Ciompi, F., et al. (2019). Quantifying the effects of data augmentation and stain color normalization in convolutional neural networks for computational pathology. *Medical image analysis* 58, 101544