Project Machine Learning
— Milestone 2 —

Filip Matysik, usr: pml16

January 3, 2025

GitHub: `https://github.com/filippuo2000/TransMIL_PML_TUB_WS24_25`

1 Introduction

This report describes the work that has been done for milestone 2 of the project that is aiming to replicate the results of the TransMIL (Shao et al. (2021)) paper, specifically on the CAMELYON16 (Ehteshami Bejnordi et al. (2017)).

As for milestone 2, both the model and code are basically fully developed. The model is strictly mimicking the one described in the TransMIL paper - details of this full implementation have been described in the "Changes in the model" section. It is now possible to load the data, train the model and evaluate its performance for training, validation and test stages, with multiple metrics describing and estimating the generalization error. Description of the most important parameters that influence training results can be found in the "Model selection" section. Explanation of the employed metrics has been presented in the "Model evaluation" section. Comparison between different models results on the validation and test set has been included in the "Test results" section.

At this stage the model has been trained and finetuned on the full version of the CAMELYON16 dataset. Training and test results are satisfying, and for some configurations of the model even better than the ones achieved by the authors of the TransMIL paper, thus in my opinion the objective of this stage of the project has been successfully reached.

It is however worth noting that at this stage of the project, no attention has been given to the explainibility of the model. Its analysis will be the main goal of stage 3 of the project. Possible tasks for milestone 3 could be also to find out to what extent different feature extraction method than the one proposed by the TransMIL authors influences model's performance and to perform a more detailed finetuning with libraries like Ray, because in this milestone, the finetuning has not been fully exploited (after finding out that the implemented model outperforms the one described in TransMIL paper, strict search for the maximum test results improvement has been postponed). Training and evaluation of the model are now ran with pytorch lightning.

2 Changes in the model

Generally in comparison with the simple baseline model implemented for milestone 1, two components have been added: 1. Pyramid Position Encoding Generator (PPEG), 2. squaring of the sequence to be able to use the PPEG module. One thing in the model has been changed - the way self-attention is being computed.

2.1 Pyramid Position Encoding Generator (PPEG)

This module is responsible for encoding spatial relationships between patches in the bag, on both local and global level. PPEG ensures that the spatial relationships between patches are explicitly encoded, adding an additional layer of spatial structure to the patch embeddings. Below, on figures 1. and 2. we see the illustration and description of the PPEG module's algorithm.

```
**Input:** A bag of feature embeddings $\mathbf{H}_S^\ell$ after correlation modelling, where $\mathbf{H}_S^\ell \in \mathbb{R}^{(N+1)\times d}$.
**Output:** The feature embeddings $\mathbf{H}_S^P$ after conditional position encoding and local information
        fusion, where $\mathbf{H}_S^P \in \mathbb{R}^{(N+1)\times d}$.
    1) Split: $\mathbf{H}_S^\ell$ is divided into patch tokens $\mathbf{H}_f$ and class token $\mathbf{H}_c$;
    $\mathbf{H}_f, \mathbf{H}_c \leftarrow \text{Split}\left(\mathbf{H}_S^\ell\right)$, where $\mathbf{H}_f \in \mathbb{R}^{N\times d}, \mathbf{H}_c \in \mathbb{R}^{1\times d}$;
    2) Spatial Restore: patch tokens $\mathbf{H}_f$ are reshaped to $\mathbf{H}_S^f$ in the 2-D image space;
    $\mathbf{H}_S^f \leftarrow \text{Restore}\left(\mathbf{H}_f\right)$, where $\mathbf{H}_S^f \in \mathbb{R}^{\sqrt{N}\times\sqrt{N}\times d}$;
    3) Group Convolution: using a set of group convolutions with kernel $k$ and $\frac{k-1}{2}$ zero
  paddings$(k = 3, 5, 7)$ to obtain $\mathbf{H}_t^f, t = 1, 2, 3$;
    $\mathbf{H}_t^f \leftarrow \text{Conv}\left(\mathbf{H}_S^f\right)$, where $\mathbf{H}_t^f \in \mathbb{R}^{\sqrt{N}\times\sqrt{N}\times d}, t = 1, 2, 3$;
    4) Fusion: $\mathbf{H}_S^f$ and the $\mathbf{H}_t^f, t = 1, 2, 3$ obtained from the convolution block processing are
  added together to obtain $\mathbf{H}_S^F$;
    $\mathbf{H}_S^F \leftarrow \mathbf{H}_S^f + \mathbf{H}_1^f + \mathbf{H}_2^f + \mathbf{H}_3^f$, where $\mathbf{H}_S^F \in \mathbb{R}^{\sqrt{N}\times\sqrt{N}\times d}$;
    5) Flatten: $\mathbf{H}_S^F$ are flattened into sequence $\mathbf{H}_{se}$;
    $\mathbf{H}_{se} \leftarrow \text{Flatten}\left(\mathbf{H}_S^F\right)$, where $\mathbf{H}_{se} \in \mathbb{R}^{N\times d}$;
    6) Concat: connect $\mathbf{H}_{se}$ and class token $\mathbf{H}_c$ to obtain $\mathbf{H}_S^P$;
    $\mathbf{H}_S^P \leftarrow \text{Concat}\left(\mathbf{H}_{se}, \mathbf{H}_c\right)$, where $\mathbf{H}_S^P \in \mathbb{R}^{(N+1)\times d}$.
```

Figure 1: PPEG algorithm

As can be seen on figure 3., the PPEG module receives an input from the self-attention layer, of shape [B, N+1, num-features] consisting of N patch-level representation tokens, each of shape [N, num-features], where N is an integer that has an integer square root value. Essentially N represents the number of patches in a bag appended with the amount of patch tokens needed to obtain the number that will have an integer square root value (described in "Squaring of the sequence" subsection), num-features is the number of features each token consists of when passed to the first self-attention layer and B is the batch size. "+1" in input shape N+1 represents the class token. It is worth noting that the input and output shape are identical for the PPEG module.

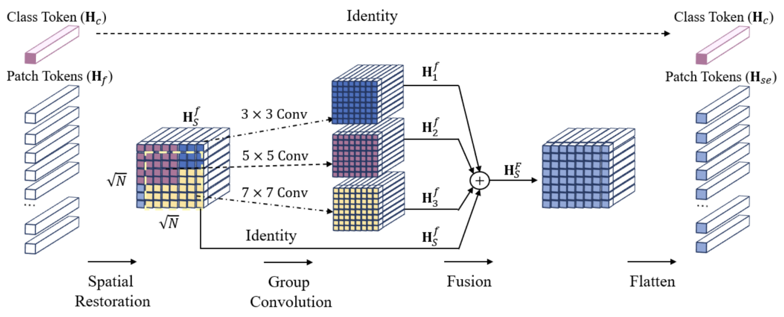


Figure 2: PPEG module ilustrated

As we see in the figure 2., it is not a single token that is being squared into an image-like representation. Instead, elements from the same position in each token are connected to form an image-like, squared representation. The use of multiple convolutional kernels (different sizes) enables encoding position information at various spatial scales, making it robust to differences in patch layouts or image resolution and collecting the information about neighbouring regions between patches. This is the conditional position encoding. Local information fusion is obtained by fusing features from different convolutional layers (as well as the identity representation of the squared representation) after group convolutions to form the output square-like representation that is then flattened to again form a set of patch-level token representations. Class token remains unchanged during this stage of the model.

Overall characteristics of the PPEG module has been described in the TransMIL paper: *"(1) PPEG module uses different sized convolution kernels in the same layer, which can encode the positional information with different granularity, enabling high adaptability of PPEG. (2) Taking advantage of CNN's ability to aggregate context information, the tokens in the sequence is able to obtain both global information and context information, which enriches the features carried by each token."*

Although in this case convolutions operate independently on each channel without summation across channels (group convolution), the multi-scale kernels allow each token's features to encode relationships across increasingly larger neighborhoods. The larger receptive fields approximate a form of global information.

Since the model is not dealing with raw image patch-level extracted from Whole Slide Images (WSIs) that form the dataset, but with precomputed feature representations of those patches, local morphological information has already been extracted when obtaining those representations, thus it would be redundant to use the convolutional layers from PPEG to encode morphological, patch-level information, because it has been already obtained earlier on.

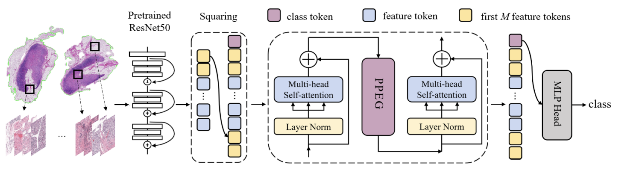


Figure 3: TransMIL model

2.1.1 Squaring of the sequence

As can be seen in the figure 2. in order to be able to obtain the squared representation of the feature tokens, their amount must be an integer that has an integer square root. To assure this is true, it is often necessary to append the original sequence of tokens with additional tokens, which are taken from the beginning of the sequence. As can be seen in figure 3., this operation is applied to the sequence of tokens containing the extracted feature representation.

Input: A bag of feature embeddings $\mathbf{H}_i = \{\boldsymbol{h}_{i,1}, \ldots, \boldsymbol{h}_{i,n}\}$, where $\boldsymbol{h}_{i,j} \in \mathbb{R}^{1\times d}$ is the embedding of the jth instance, $\mathbf{H}_i \in \mathbb{R}^{n\times d}$

Output: Bag-level predicted label $\hat{Y}_i$

1) Squaring of sequence;

$\sqrt{N} \leftarrow \lceil \sqrt{n} \rceil$, $M \leftarrow N - n$, $\mathbf{H}_S \leftarrow$ Concat $(\boldsymbol{h}_{i,class}, \mathbf{H}_i, (\boldsymbol{h}_{i,1}, \ldots, \boldsymbol{h}_{i,M}))$, where $\boldsymbol{h}_{i,class} \in \mathbb{R}^{1\times d}$ represents class token, $\mathbf{H}_S \in \mathbb{R}^{(N+1)\times d}$;

Figure 4: Description of the squaring step

Amount of tokens that have to be appended at the end of the sequence is equal to M (figure 4.). To calculate M one has to first calculate the ceiling of the square root of number of tokens in the sequence, n, which is equal to $\sqrt{N}$. M is then the difference between N and n.

2.2 NystromAttention

For this milestone, Nyström attention (Xiong et al. (2021)) has been implemented in the model as proposed by the authors of TransMIL. It is a technique designed to approximate standard self-attention in transformers with reduced computational and memory complexity, making it more efficient for long sequences. After all the computational complexity is reduced from $O(n^2)$ to $O(n)$. Thanks to this processing of bags (WSIs) that contain thousands of tokens as input through self-attention layers is done more efficiently and faster.

3 Feature extraction - reminder

For feature extraction CTransPath (Wang et al. (2022)) method has been chosen. However, the features have been provided by the project's coordinator, thus the feature extraction process was not explicitly performed. Description of the CTransPath method can be found in appendix A.

It would be interesting to compare the influence of different feature extraction methods on the training results, which could be a part of the workload for Milestone 3.

4 Model evaluation

To evaluate the model during training (on validation set) and after training (on test set) a bunch of classification metrics has been utilized. Motivation for their use has been described below.

4.0.1 Accuracy

Basic classification metrics that essentially describes what percentage of the model's predictions have been correct. It describes the overall performance of the model and is class-agnostic, thus for imbalanced datasets it is not a good choice, cause high accuracy for the dominant class could positively influence overall accuracy, even if the accuracy of the minority class was very small. Taking into consideration that CAMELYON16 is a slightly imbalanced dataset (as can be seen in appendix 7) with the negative class (0 - no cancer) being the dominant class, using only the accuracy metrics could falsify the model's ability to correctly classify the positive class (1 - cancer), thus additional metrics are needed to evaluate the performance of the model on the class-level. Especially since it is more important for the model to be able to correctly detect the positive class, thus it's the accuracy of the positive class that should primarily be maximized.

4.0.2 AUC

AUC here refers to the AUROC metrics (area under the receiver operating characteristic). This metrics evaluates whether the model is able to correctly rank examples. ROC curve shows the trade-off between true positive rate (TPR) and false positive rate (FPR) across different decision thresholds. The goal is to minimize the FPR and maximize the TPR. AUROC is a more informative metrics than accuracy as it states what is the probability that given the positive class sample, the model will assign it a higher probability in classification than it would to a negative class. Using it as a metrics for a problem, where predicting the positive class correctly is more important than negative class, is a reasonable choice. However for a slightly imbalanced dataset this metric could still assign higher scores to models that maximize the negative class accuracy and achieve worse results for the positive class and thus have a higher AUC score than models that perform slightly worse for the negative class, but better for the positive class. Taking this into consideration, in my opinion it is necessary to look at additional metrics in order to choose the best model, which in the case of CAMELYON16 and problem of cancer region detection would be Recall (positive class accuracy) and Specificity (negative class accuracy).

4.0.3 Recall (or class 1 accuracy)

This metrics essentially describes good the model is at predicting true positives and how often it predicts false negatives. Since the false negative error is the worst case scenario and the one to avoid in the cancer region detection task (because it states "no cancer" when the true result is "cancer"), it is absolutely crucial to maximize this metric. This is however difficult since 1) in CAMELYON16 (and probably in real life scenarios as well) there are slightly more negative than positive samples and 2) cancer regions in positive samples in the CAMELYON16 dataset in most cases take up to 10% (appendix A) of the extracted tissue, so it is difficult to make the model distinguish between positive and negative samples with such slight difference between the two.

4.0.4 Specificity (or class 0 accuracy)

This metrics describes how well the model predicts the true negatives with respect to both true negatives and false positives. It is less important than recall in this case, but still worth looking at. Again, scores from this metrics are still important for the model selection and one should not completely trade-off specificity for recall (the model that perfectly predicts "cancer" when the patient really has cancer, but often predicts cancer when the patient is healthy is not necessarily the desired one), even though the probability of such outcome is rather low.

4.0.5 Final model selection

When choosing the final model for the CAMELYON16 dataset task one should primarily and most importantly look at the Recall (positive class accuracy) score and then secondly at AUC score and finally at both Specificity and overall accuracy scores.

5 Model selection

In order to choose the best performing models (and at the end model) several parameters have been tuned. Those were most importantly: optimizer's learning rate, reduction in number of features passed as an input to the TransMIL model. Additionally comparison of two different optimizers has been conducted, as well as the comparison of model's performance with and without the PPEG module.

No cross-validation has been performed. In order to obtain reproducibility of the experiments, seed has been set to control the randomness in all pytorch, numpy and python.random operations, specifically in weight initialization and order of the sample extraction from the dataloaders.

It should be noted that metrics such as Recall and Specificity have been calculated for the positive class, thus not by taking the mean out of those metrics performed separately on each class.

5.1 Training setup

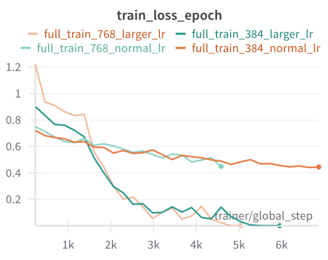
All models have been trained on the distance of 30 epochs, with early stopping callback monitoring the validation loss, and patience set to 5 epochs. Checkpoint for the test run has been chosen based on the minimum validation loss from the training. Thus when monitoring the results presented in the report, it is best to check in which epoch the validation loss was the smallest and look at different metrics values for this specific epoch - this is the version of the model that is being used during test stage.

It is worth noting that in models names: "larger_lr" refers to lr=0.002 and "normal_lr" refers to lr=0.0002. No learning rate scheduler has been utilized during training. All runs have been performed on the gpu-test partition and took less than 15 minutes.

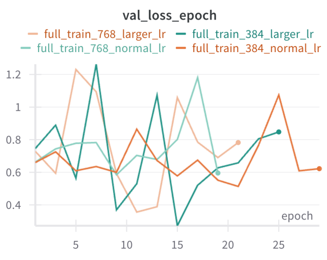
If there's no optimizer name in the model run, the optimizer is SGD. If there's no "no_ fc_layer" caption in the model's name it means that input features are preprocessed by the fully-cpnnected layer. number after the "full_train" indicates the number of input features to the model"

5.2 Learning rate influence - SGD optimizer

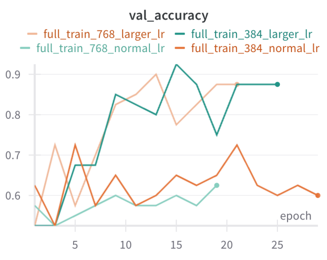
Since Lookahead optimizer that has performed best according to the TransMIL paper, is not available in pytorch library, basic SGD optimizer has been used for comparison at first.



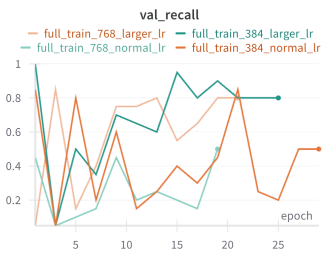
(a)



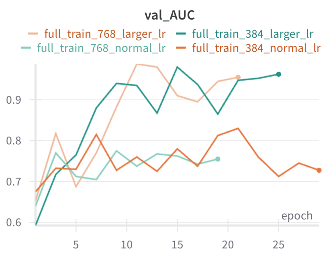
(b)



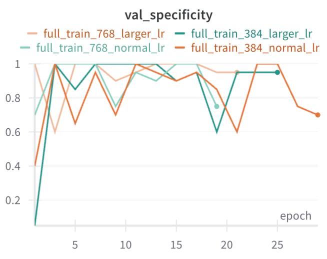
(c)



(d)



(e)



(f)

Figure 5: Comparison of learning rates for SGD optimizer.

Generally it can be seen that for the SGD optimizer larger learning rate (in this case 0.002) enables the model to converge during training and obtain decent results on the validation set, significantly better than the ones obtained by models with smaller learning rate (0.0002). Use of learning rates larger than 0.002 and smaller than 0.0002 did not influence the training positively. One could wonder about the lack of convergence on validation loss for the models with larger learning rate, so in order to achieve it one should probably either increase the patience parameter in early stopping callback or adapt the learning rate scheduler.

Out of these six models clearly the one with 384 features on input performs best on the Recall metrics (80+%) and relatively well on three other important metrics. Thus this model is chosen to be included in top 4 best models that are used for the test evaluation.

Models with raw 768 features input were performing worse than the plotted ones. However later it will be shown that increasing the patience parameter in early stopping helps those models to converge and achieve better results (larger learning rate + SGD).

5.3 Lookahead with Radam optimizer influence

Models with Lookahead+Radam optimizer have similar and for some sets of parameters even slightly better performance than the ones with SGD optimizer.

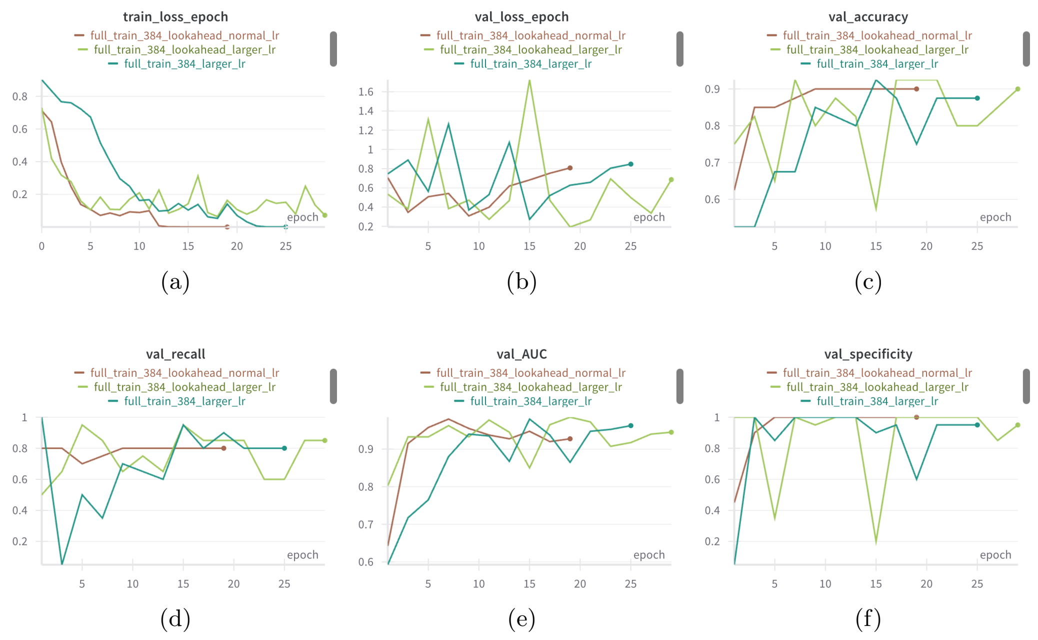


Figure 6: Best SGD model vs Lookahead optimizer model + optimizers comparison.

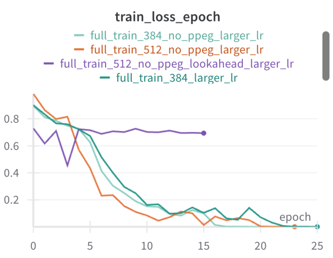
As can be seen on the figure 6. a comparison of the two best models for both optimizers (384 features as input) shows that both models perform similarly on the most important Recall metrics, reaching similar peak values and stabilizing around the 80% level, at the same time maintaining high (over 80%) AUC and specifity values, thus also overall accuracy.

On figure 6. the comparison of the learning rates for the same version of the model with Lookahead+Radam optimizer has also been presented. Model with the smaller learning rate converges more smoothly (train loss), but it's training stops earlier due to the implemented callback. However its values for the both Recall and AUC metrics are smaller compared to the Lookahead model with larger learning rate, thus setting the learning rate to 0.002 instead of 0.0002 is a more reasonable choice for the Lookahead+Radam optimizer and other experiments for different number of input features with this optimizer (not presented here) were conducted with this learning rate.

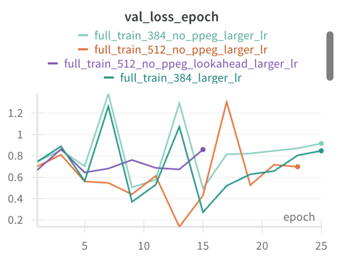
Finally the presented Lookahead model with 384 features on input is chosen for the final top 4 evaluation on the test set since this was the best performing model for this specific optimizer configuration.

5.4 PPEG module influence

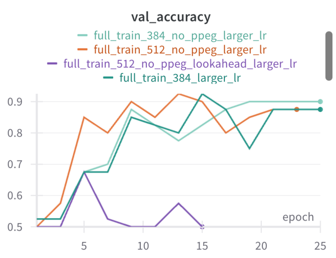
In previous milestone of the project, a simple baseline model has been presented without the PPEG module. Since it hasn't been trained on the full dataset, it is interesting to do so now and see how it performs in comparison with the models that contain PPEG. Especially since input features for the CAMELYON16 dataset have been obtained with a more topic-oriented method (CTransPath - section "Feature extraction - reminder") than the one proposed by the authors of the TransMIL paper (ResNet50 (He et al. (2015)) model pre-trained on ImageNet (Deng et al. (2009)).



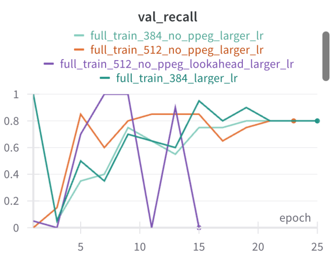
(a)



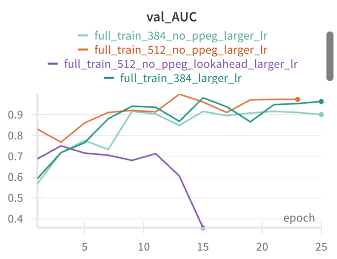
(b)



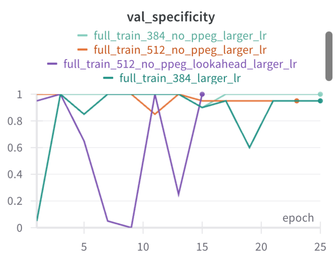
(c)



(d)



(e)



(f)

Figure 7: No PPEG vs best PPEG model + optimizers comparison.

As it interestingly turns out, models without PPEG module have similar and in some cases even better performance than the best ones with PPEG. On figure 7. two best models without the PPEG module have been shown (384 and 512 input features, SGD optimizer with learning rate of 0.002). For most epochs the model with 512 input features achieves better results than the model with 384 input featuers features on both Recall and AUC metrics, which are considered most important for the given problem. Additionaly, for the smallest val_loss of the 512-model the scores of these two metrics are higher than the ones achieved by the 384-model at its minimum epoch val_loss. Thus the 512-model seems to be a better choice for the final top 4 models evaluation the test set.

As can be seen, the SGD is a better choice than Lookahead+Radam in this case, as the performance for these two has been compared. The second one does not really converge and outputs unstable values for most metrics.

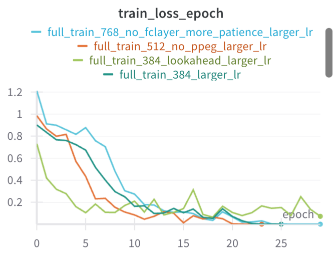
Further analysis on why models without PPEG are able to in some cases even outperform models with PPEG should be conducted in Milestone 3 of this very project. The most intuitive hypothesis would be that the extracted features already contain the spatial information factor about the relationship between neighboring patches, which is provided by PPEG. However the previous analysis of the feature extraction method performed in Milestone 1 concluded that all patches from a WSI are processed through the CTransPath method independently in contrast to PPEG, thus not implying the existence of information about the spatial context for other patches within a single patch features. This has to be further investigated.

5.5 Number of input features influence

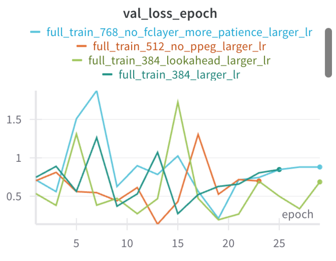
Generally it is difficult to draw clear conclusions about their influence on the training results. It has however been observed that usually models with 384 features as input tend to obtain better results than models with 512 parameters and 768 parameters processed by an fully connected layers. This is not true in all cases, because as we will see in the "Best models" section, for models without the PPEG module, the best configuration was the one with 512 parameters. Models with 768 raw features as input take more time to converge and usually achieve better results than ones with 512 or 768+fclayer, but due to the use of early stopping with patience set to 5, this behavior has not been fully observed here. An exceptional training with patience set to 10 however placed one such model in the top 4 best models, as per section "Best models". This however suggests that for a more fair comparison, the patience parameter should be set slightly larger.

5.6 Best models

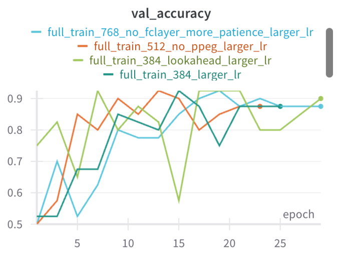
In this section models assesed as "best" from the training-validation stage are presented. Most of them have been presented in previous subsections. However there's an additional model here (768-model without the first fully-connected layer), which has been trained on different terms than the other models (larger patience parameter, to allow model's convergence) and in this case to keep the comparison fair, its results should be treated as a justification for re-performing experiments with a different patience parameter. It also has been placed here to prove that models that are trained on raw 768 feature input can achieve very good results in training and are able to after all converge.



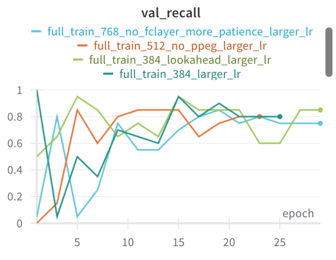
(a)



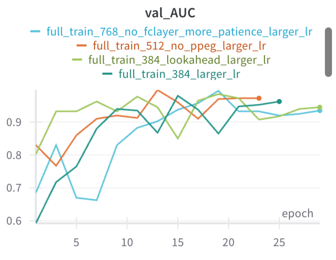
(b)



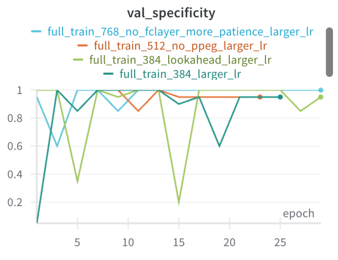
(c)



(d)



(e)



(f)

Figure 8: Best 4 models.

As can be seen on figure 8., all of the 4 chosen best models maintain their Recall score over or at least close to the 80% after a certain epoch at the same time scoring over 80% on the remaining 3 metrics. All these models seem to not trade off low Specificity score for high Recall score (positive class accuracy), which makes them well-balanced and probably best suited for the task of cancer presence detection.

6 Test results

Below table presents results of the inference on the test set for the chosen best 4 models:

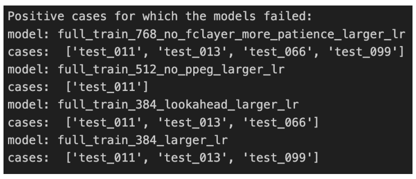
Table 1: Test results for best models

Name	Recall (%)	AUC (%)	Acc (%)	Specificity (%)
1. full_train_768_no_fclayer_more_patience_larger_lr	91.84	**98.39**	**95.35**	**97.50**
2. full_train_512_no_ppeg_larger_lr	**97.96**	97.55	**95.35**	93.75
3. full_train_384_lookahead_larger_lr	93.88	97.45	93.02	92.50
4. full_train_384_larger_lr	93.88	95.51	86.05	81.25
TransMIL paper model	na	93.09	88.37	na

It can be seen that the first 3 models outperform the results of the model presented in the TransMIL paper (all three have higher AUC and accuracy scores). Fourth model has a larger AUC, but slightly smaller accuracy, thus it's performance in comparison with the orginal model could be evaluated as similar. This model however achieves a significantly smaller Specificity score than all three remaining models and at the same time does not achieve an outstanding Recall score, thus it's generalization error is the largest out of the tested models.

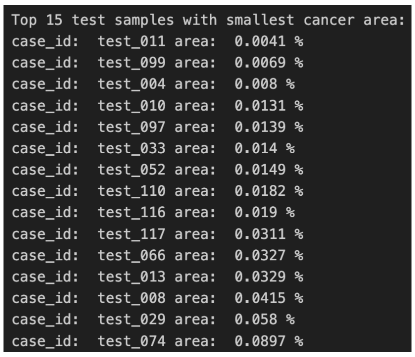
Best results on the test set are achieved by the 512-no-ppeg-model, which as the only model achieves the 95+% Recall score and at the same time maintains the Specificity score at the 90+% level, which implies good performance of the model for both positive and negative classes. Two remaining models with PPEG still achieve very high Recall and at the same time Specificity scores, which also suggests they have a strong generalization ability.

Below, on figures 9. and 10. a brief analysis of the worst cases has been performed (incorrectly classified positive cases - false negative errors). It can be seen that all models failed on the 'test_011' sample, which turns out to have the smallest percentage of the cancer area in the whole test set, thus it is a "difficult case". Same applies to other false negative errors across models - in total there are only 4 positive test samples which different models misclassified, what it means is that if the models failed, they failed on basically the same test samples, all of which are in the top 15 test cases with the smallest cancer region percentage. For a more detailed explanation of why the model is failing for samples that have small cancer region, but not the smallest across the test dataset, one would need to delve into the explainability of the model. It would also be worth checking why the models are misclassifying negative samples.



```
Positive cases for which the models failed:
model: full_train_768_no_fclayer_more_patience_larger_lr
cases:  ['test_011', 'test_013', 'test_066', 'test_099']
model: full_train_512_no_ppeg_larger_lr
cases:  ['test_011']
model: full_train_384_lookahead_larger_lr
cases:  ['test_011', 'test_013', 'test_066']
model: full_train_384_larger_lr
cases:  ['test_011', 'test_013', 'test_099']
```

Figure 9: Uncorrectly classified positive cases



```
Top 15 test samples with smallest cancer area:
case_id:  test_011 area:  0.0041 %
case_id:  test_099 area:  0.0069 %
case_id:  test_004 area:  0.008 %
case_id:  test_010 area:  0.0131 %
case_id:  test_097 area:  0.0139 %
case_id:  test_033 area:  0.014 %
case_id:  test_052 area:  0.0149 %
case_id:  test_110 area:  0.0182 %
case_id:  test_116 area:  0.019 %
case_id:  test_117 area:  0.0311 %
case_id:  test_066 area:  0.0327 %
case_id:  test_013 area:  0.0329 %
case_id:  test_008 area:  0.0415 %
case_id:  test_029 area:  0.058 %
case_id:  test_074 area:  0.0897 %
```

Figure 10: Top 15 test case with the smallest cancer region percentage

7 Discussion

Generally, in author's opinion, the goal of this milestone has been achieved, which was to train and evaluate the fully implemented model and possibly replicate the original results from the chosen scientific paper (which is also the case here). The classifiers were not expensive to train and all experiments have been successfully ran on the single GPU. This allowed for the comparison of multiple parameters. However since not all parameters have been tweaked, there's still potential for improvement on the finetuning field (playing with self-attention layers parameters and more optimizers, as well as adding the learning rate schedulers and extending the patience parameter in the early stopping callback).

Chosen and analyzed Multiple Instance Learning method is suitable for the task, what has been proven by the results obtained in this milestone. However it turned out to be surprising that one of the main improvements in the MIL approach to the cancer detection problem introduced by the authors of TransMIL, which was the PPEG module, was not influencing the performance of the model as much as shown in the TransMIL paper. This should be further investigated in Milestone 3 of the project.

Simplicity of the MIL approach (the final task is the binary classification problem) allows to obtain a confidence measure for the predictions, which is simply the output of the last "classification" fully-connected MLP layer, passed through the softmax function, which results in the prediction probabilities for each of the two classes that sum up to 1.

References

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

B. Ehteshami Bejnordi, M. Veta, P. Johannes van Diest, B. van Ginneken, N. Karssemeijer, G. Litjens, J. A. W. M. van der Laak, , and the CAMELYON16 Consortium. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. *JAMA*, 318(22):2199–2210, 12 2017. ISSN 0098-7484. doi: 10.1001/jama.2017.14585. URL `https://doi.org/10.1001/jama.2017.14585`.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. URL `https://arxiv.org/abs/1512.03385`.

Z. Shao, H. Bian, Y. Chen, Y. Wang, J. Zhang, X. Ji, and Y. Zhang. Transmil: Transformer based correlated multiple instance learning for whole slide image classication. *CoRR*, abs/2106.00908, 2021. URL `https://arxiv.org/abs/2106.00908`.

X. Wang, S. Yang, J. Zhang, M. Wang, J. Zhang, W. Yang, J. Huang, and X. Han. Transformer-based unsupervised contrastive learning for histopathological image classification. *Medical image analysis*, 81:102559, 2022. URL `https://api.semanticscholar.org/CorpusID:251207603`.

Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention, 2021. URL `https://arxiv.org/abs/2102.03902`.

A a) Dataset statistics

In total there are 399 WSI images in the dataset. However features have been provided for 398, with features missing for the "normal-144" slide. Thus there are 238 elements in the dataset of the negative class (0 - no cancer), and 160 positive class elements (1 - cancer). The data has been split in the following ratio for the train, test and evaluation subsets:

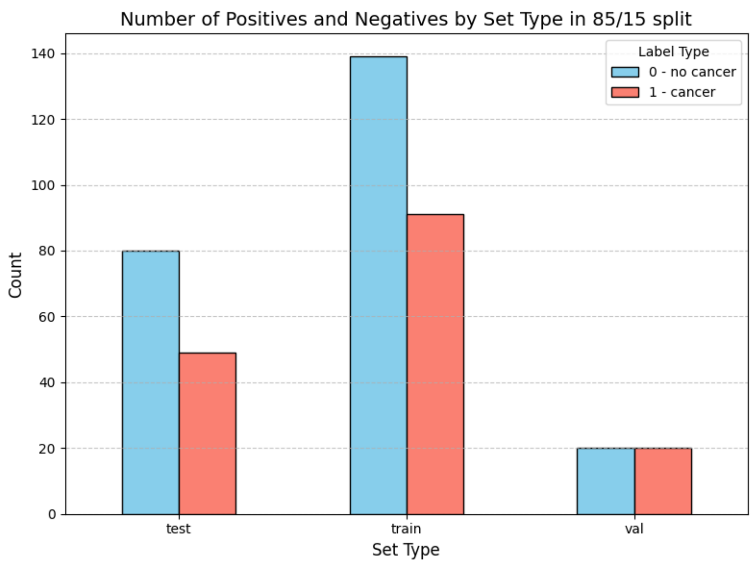


Figure 11: Split of the data for subsets and their within class label ratio

A b) Analysis of the positive samples in CAMELYON16

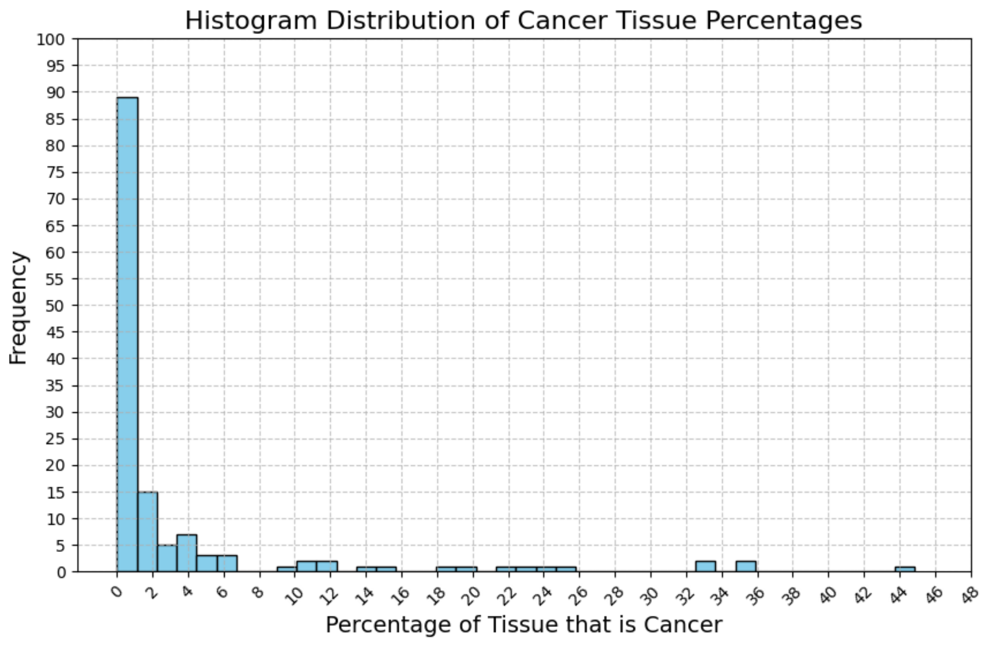


Figure 12: Distribution of the ratio of cancer tissue area with respect to the area of the tissue

A c) Feature extraction process

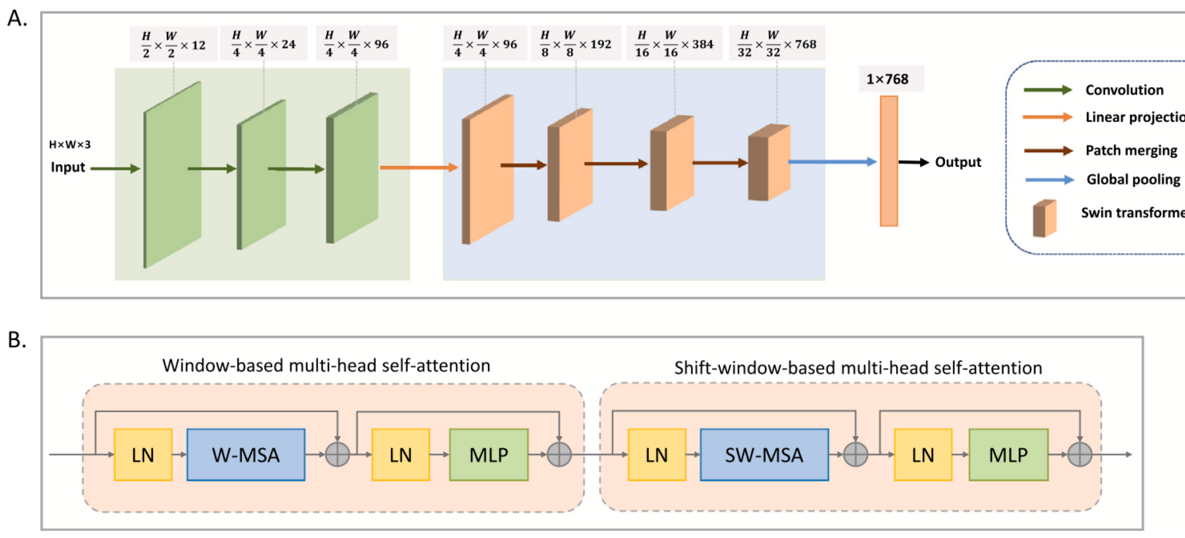

A.
$\frac{H}{2} \times \frac{W}{2} \times 12$
$\frac{H}{4} \times \frac{W}{4} \times 24$
$\frac{H}{4} \times \frac{W}{4} \times 96$
$\frac{H}{4} \times \frac{W}{4} \times 96$
$\frac{H}{8} \times \frac{W}{8} \times 192$
$\frac{H}{16} \times \frac{W}{16} \times 384$
$\frac{H}{32} \times \frac{W}{32} \times 768$
1×768
H×W×3
Input
Output
Convolution
Linear projectio
Patch merging
Global pooling
Swin transforme
B.
Window-based multi-head self-attention
Shift-window-based multi-head self-attention
LN
W-MSA
LN
MLP
LN
SW-MSA
LN
MLP

Figure 13: Architecture of the feature extraction model

As can be seen in the Figure 13., the feature extraction process involves the following:

- CNN backbone: Each patch is passed through CNN but to extract the low-level visual features, such as textures and shapes.

- Transformer Encoder: The feature map from the CNN is flattened and fed into a transformer encoder. The transformer captures long-range dependencies within the patch itself, allowing the model to understand global patterns inside the patch (e.g., structural relationships between different regions of the patch).

- Dimensionality Reduction: After passing through the transformer, the output is processed by dimensionality reduction layers (e.g., linear layers) to produce a fixed-length feature vector for each patch, ([1x768])

In the context of CTransPath and the CAMELYON16 dataset, the network processes one patch at a time as an independent unit. Each patch is fed into the network to extract its local feature representation. This means that while the model excels at identifying patch-level features (e.g., cellular or tissue structures), the direct dependencies or spatial relationships between adjacent patches are not explicitly modeled during the feature extraction process.

Extracted features come out of the feature extractor in the normalized form (centered around the zero value and squashed between the range from -1 to 1.). Shape of the output from the feature extractor vary depending on the input size of the bag (as in how many patches are in each), but for each patch the extracted sequence has the same length - 768. Thus the output for a single bag is of shape: [N, 1, 768], where N is the number of patches in the bag. This representation is then resized to [N, 768], to get rid of the second, redundant dimension. As described above in 3.2, the model can handle inputs of varying size.