UAB
Universitat Autònoma
de Barcelona

Helicobacter pylori Diagnosis from Microscope Images

Submitted by:

- Luis Domene García, 1673659
- Eric López Cervello, 1672981
- Marino Oliveros Blanco, 1668563

Vision & Learning

15-11-2024

1. Introduction

Helicobacter pylori (H. pylori) is a gastric bacterium classified as a class 1 carcinogen to humans by the World Health Organization since 1994. Its detection is crucial for early diagnosis and prevention of related gastric diseases, including cancer. However, the conventional approach of visually examining these large and complex histological images (up to 120,000 x 16,000 pixels) is labor-intensive and time-consuming, requiring the expertise of skilled pathologists.

This project seeks to address these limitations by leveraging artificial intelligence (AI) techniques to automate and enhance the detection of H. pylori in immunohistochemical histological images. The proposed system will not only reduce the workload on pathologists but also improve diagnostic accuracy and efficiency.

Key challenges include managing small, highly imbalanced datasets, identifying anomalous staining, and ensuring reproducibility of results. Through the integration of machine learning, generative models, and attention mechanisms, this initiative aims to advance the state of medical diagnostics and provide a robust tool for tackling this global health concern. In the next section, we will explain the different methods we use to attack this problem.

2. Methodology

Given a microscope image of the patient cells with potential helicobacter, this is splitted into smaller patches. Our strategy consists of two steps: first, we classify whether each of these patches contains helicobacter or not; then we do an aggregation of the predictions of all patches in order to make a whole diagnosis for the patient. Each of the following sections explains in detail each stage.

2.1 Patch classification

As explained before, in this step we classify each of the patches individually into 2 classes: positive (infected) or negative (sane). In order to do this, two approaches have been implemented: One using a simple binary classifier of the patch image, and another more sophisticated involving an autoencoder for anomaly detection.

2.1.1 Classification model

The objective of this model was to classify images by extracting meaningful features. The architecture incorporates two convolutional layers to progressively learn spatial hierarchies of features. Initially, it captures low-level patterns, such as edges and textures, in the first layer and progresses to more complex structures like shapes and bacterial features in the second layer. Pooling layers follow each convolutional layer to reduce the spatial dimensions, thereby retaining essential information while enhancing computational efficiency and reducing the risk of overfitting.

After high-level spatial features are extracted, fully connected layers transform these features into a compact representation by expanding them into a high-dimensional space (131,072 neurons) and progressively reducing them. The final layer consists of two output

neurons for binary classification (Positive and Negative classes), using a softmax activation function optimized for this task. Figure 1 illustrates the model architecture.

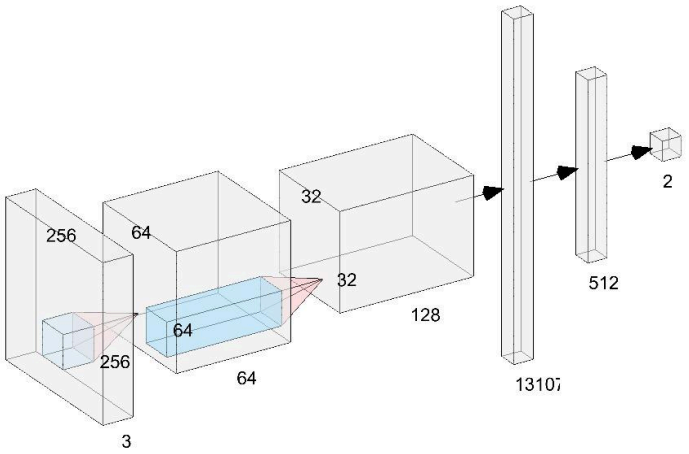

Figure 1: Architecture used for our Classification model, showing the convolutions and layers

Before passing the raw patches to the model, we apply some preprocessing. While most of the images had a size of 256x256, we found some samples that deviated from this size, so we resized them appropriately. Then we transformed them from range 0-255 to 0-1, and further standardized them with mean 0.5 and standard deviation 0.5. Finally, we remove the alpha channel, leaving the images in RGB.

2.1.2 Anomaly Detection + Adaptive Thresholding

The other approach is more sophisticated and includes an Autoencoder to find the quantity of red that the image patch contains, and an adaptive threshold to classify those with the quantity above a specific value.

The first step is to implement an autoencoder with the aim of reconstructing the original image patch, but without the red color characteristic of Helicobacter. This way, we will be able to calculate the difference in red between the original and the reconstructed. The assumption is that, if this is high, it will mean Helicobacter was present in the patch, and thus we consider the patch to be positive, and negative otherwise. The image preprocessing used is the same as for the classification model.

The autoencoder

In a nutshell, an autoencoder is a type of neural network architecture consisting of an encoder, a bottleneck and a decoder. The encoder takes an image and reduces its dimensionality by applying a series of convolutions. The resulting latent vector, located at the bottleneck, is then passed to the decoder, which tries to reconstruct the original image by increasing the dimensionality again using deconvolutions. The purpose of this is to make the neural net learn a summarized information of the image and store it in a much smaller latent vector.

To accomplish the previously explained process, we train the autoencoder with the patches from all the patients whose diagnosis is negative. These do not contain Helicobacter, and thus the cells are mostly blue, so the model will only learn to build blue pixels when decoding from the latent vector. When a patch with red pixels is passed to the model, it will have to reconstruct them as blue, since it has no knowledge of the color red. An example is shown in Figure 4.

In Figure 2, we can see the architecture of the autoencoder we trained. The encoder consists of 4 convolutional layers, of number of channels 3 (RGB), 16, 32 and 64 respectively, kernel size 3, stride 2 and padding 1. Each convolution is followed by a ReLU activation function. Next, the bottleneck is thinner due to the previous convolutions, but has more channels, 128. The decoder is just a mirror of the encoder in the number of layers and channels, and uses the same kernel size, stride and padding. We also used ReLU except for the last layer, which is followed by a Sigmoid in order to fit numbers to a 0 to 1 range, which is approximately the range of the input image values when normalized. The choice of these values is later explained in the Metrics and Experiments section.

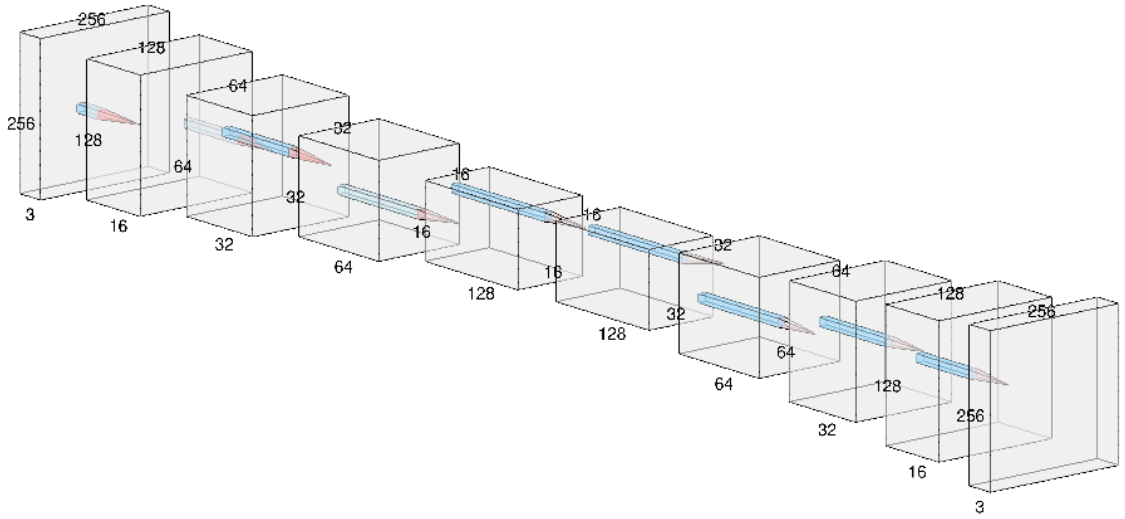

Figure 2: Autoencoder architecture used for patch reconstruction

A final postprocessing is applied to the resulting image. It is first denormalized as: $x * \mu + \sigma$, and then returned back to color range 0-255 as $x * 255$. Additionally, we found reconstructions to have a strange green and red pattern on the bottom and left sides, shown in Figure 3. We thought this could potentially damage the algorithm in charge of counting red pixels. We didn't manage to avoid it by changing the architecture values like output padding or stride, so we applied a center cropping of 4 pixels and constant padding to return the image back to its original size, using an arbitrary blue color from the background of one sample image.

Figure 3: Strange green and red patterns on the border of reconstructions, before and after applying a crop and pad postprocessing.

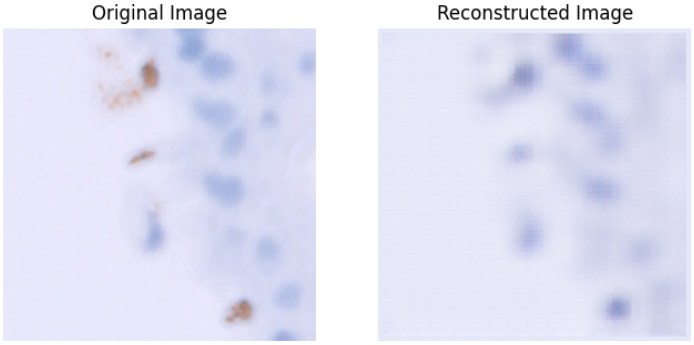

Figure 4: Example of a patch image containing Helicobacter (in red) and the reconstruction by the autoencoder after post processing.

Adaptive thresholding

Once we have reconstructions from the patches, we compute the difference of red pixels between these and the corresponding originals, as a fraction. We convert both samples to the HSV space, where the color can be more easily identified in the Hue channel; we apply a mask on the Hue values in the range [-20, 20]; we count the number of pixels with these values; and finally we compute the fraction of those counts of pixels for the original and reconstructed pair of patches. This is done with the following formula:

$$F_{red} = \frac{\#\{(i,j)\ with\ -20 < H_{Ori}(i,j) < 20\}}{\#\{(i,j)\ with\ -20 < H_{Rec}(i,j) < 20\}}$$

Now, we proceed to find which is the best threshold to decide how much red fraction is necessary to classify a patch as positive. In order to do it, we first compute the ROC curve of the red fractions and the actual patch labels. Next, we find the closest point of the curve to (0, 1). The corresponding red fraction is the threshold we will use. An example is shown in Figure 5. Intuitively, this means that there is no other threshold that provides a better trade-off between True-positive and False-positive rate. Once obtained the optimal red fraction threshold, we can classify patches as positive if their red fraction is above the threshold, and negative otherwise.

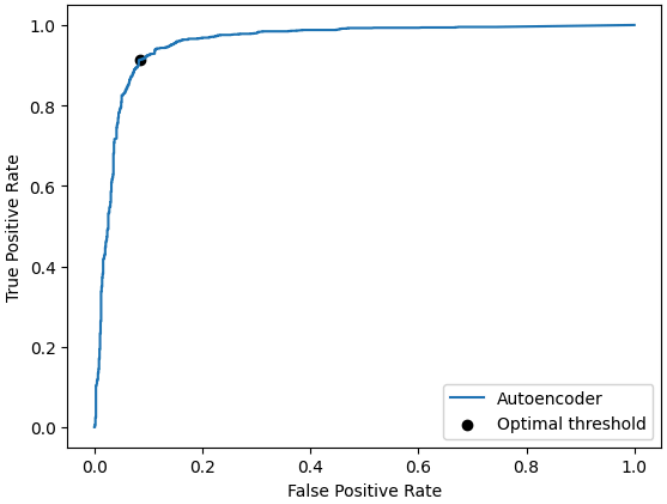

Figure 5: Optimal threshold found on ROC curve

2.2 Patient Diagnosis

After classifying individual patches as potentially containing Helicobacter pylori or not, the next step is to determine whether the patient actually has the bacterium. This requires a binary diagnosis based on the entire microscope image, which is composed of all the patches. A single positive patch does not necessarily indicate the presence of Helicobacter pylori in the patient, as several factors can contribute to false positives in patch classification. For instance, errors can arise from model reconstruction inaccuracies or misinterpretation of red areas in the original image that are unrelated to the bacterium.

To address this, we employ an adaptive thresholding method, similar to the approach used with the autoencoder in anomaly detection. Specifically, we compute a ROC curve based on the percentage of positive patches per patient image across the training set. The optimal threshold is identified as the point closest to (0,1) on the curve. Patients are diagnosed as positive if the proportion of positive patches exceeds this threshold.

However, for the classifier-based approach, we observed that the model typically classified only one patch per patient as positive — either the patch containing H. pylori or, in cases where multiple patches contained the bacteria, the most prominent one. This allowed us to simplify the diagnosis process by setting a manual threshold of 1. Consequently, patient-level diagnoses were derived by aggregating predictions across all patches in an image. If any patch was classified as positive, the patient was assigned a positive diagnosis.

This approach reflects a key distinction between the autoencoder and classifier models. While the assumption that a single positive patch does not guarantee the presence of H. pylori is more applicable to the autoencoder, the classifier's behavior supports a simpler decision rule.

3. Experimental Design

3.1 Dataset Description

The dataset used for this project is the Quiron Dataset, it contains over 600 Histological WSI images from over 300 patients gastric biopsies, these are stained using the Immunohistochemical staining method.

Our dataset is divided into first Cross-Validation and HoldOut. Cross-Validation is used for training and validation, and contains two subsets called Annotated and Cropped. Cropped is the main part of our dataset and contains many cropped patches per patient (ranging from 200 to 3000 approximately) where the H. Pylori is stained as previously mentioned. The Annotated on the other hand contains the same number of patients as the 'Cropped' but for each patient the number of patches is reduced to just a few (ranging from 2 to 116), we know which patches are positive or negative, plus the positive patches are data augmented. We also have two excel and csv style files. The first one contains the Patch ID and the concentration of H. Pylori; -1 if negative, 1 if positive. This is useful for patch classification. The second file contains the Patient ID and the concentration of H. Pylori: BAIXA-low, ALTA-high, and NEGATIVA-negative. This file is useful for patient diagnosis.

Holdout on the other hand is the test set we will be using. It contains patches of images of unseen patients for our trained models. That is, patients in Holdout are completely different from patients in Cross-Validation. The same csv file contains the labels to evaluate this test set on the patient diagnosis task.

3.2 Metrics and Experiments

In this section, we explain how the models were trained, including the training process, hyperparameters used and the design of cross validation to train and validate the methods in multiple folds, to obtain more reliable training evaluations. Also, we provide a description of the metrics used and what divisions of the dataset were used for each step of our methods.

3.2.1 Training configuration

Classifier

The initial set of Annotated patches consisted of 2,695 images. Those where bacterial presence could not be confidently determined by experts (Presence = 0) were excluded. The final dataset comprised 1,181 positive and 899 negative samples. Therefore the classifier model was trained on an annotated dataset of 2,080 images. A stratified k-fold cross-validation approach was used, maintaining class proportions across training and validation sets, and ensuring that images from the same patient did not appear in both splits.

To make predictions, an ensemble method was applied, leveraging the best-performing model from each fold. In case that the validation loss improves during training the model is saved. If during an epoch the validation loss does not improve then patience is increased by 1. When patience reaches 5 that means that in the last 5 epochs the validation loss hasn't improved, then the training is stopped and the model saved is the last model that improved

the validation loss. At the end of the training we have 5 different models. The ensemble demonstrates that selecting the model with higher confidence in its predictions outperforms the approach of simply averaging the probabilities to classify a patch as positive or negative.

The training parameters included 40 epochs (with early stopping), an initial learning rate of 0.001 and a batch size of 256. Early stopping was done to prevent overfitting, where training was stopped if the validation loss failed to improve for 5 consecutive epochs. The best model for each fold was saved based on the lowest validation loss.

At the end of training, five distinct models were obtained (one per fold). Table 1 summarizes the training performance and includes the information of the last epoch saved.

Fold	Train Label Distribution	Validation Label Distribution	Epochs	Loss	Validation Loss	Accuracy (%)	Confusion Matrix	Loss Improvement
0	1: 945, 0: 719	1: 236, 0: 180	6	0.305	0.269	87.5	[[176 4], [48 188]]	0.3347 to 0.2699
1	1: 945, 0: 719	1: 236, 0: 180	7	0.189	0.268	90.625	[[176 4], [35 201]]	0.2931 to 0.2689
2	1: 945, 0: 719	1: 236, 0: 180	9	0.325	0.436	88.701	[[172 8], [39 197]]	0.4503 to 0.4365
3	1: 945, 0: 719	1: 236, 0: 180	6	0.202	0.245	90.865	[[176 4], [34 202]]	0.2604 to 0.2450
4	1: 944, 0: 720	1: 237, 0: 179	8	0.240	0.229	92.307	[[174 5], [27 210]]	0.2855 to 0.2294

Table 1:Training performance of the Classifier across folds

To determine a diagnosis for each patient, all image patches were classified individually. A patient was deemed positive if at least one patch was classified as positive. Initially, we considered implementing a patient-level classifier, using features such as the number of positive and negative patches to infer the bacterial density within the patient. This approach would have allowed us to leverage the entire dataset of patients to train the classifier. However, this idea was ultimately discarded in favor of a simpler binary classification approach (positive or negative) at the patient level. This decision was based on the observation that negative patients inherently lack any positive patches, making the added complexity of a patient-level classifier less valuable for our primary objective. This idea, however, is later discussed in the ablation studies.

Figure 6 illustrates examples of misclassified patches during the training validation process. These errors often occur in patches that differ significantly from the training data.

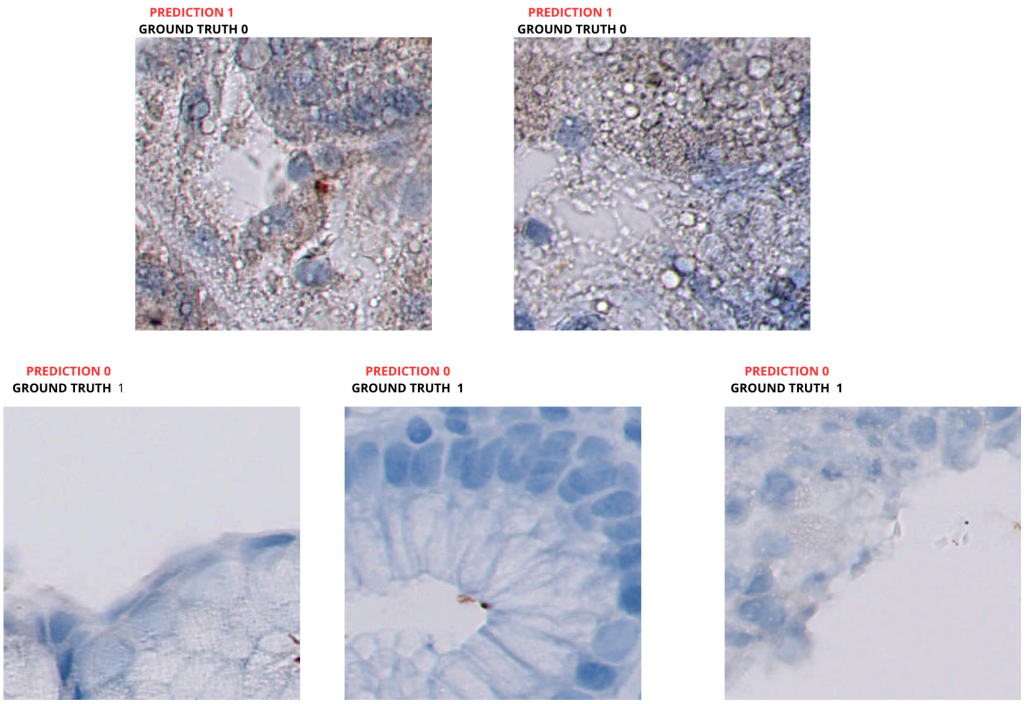

Figure 6: Example of challenging cases from training validation. Top images show false positives, bottom images show false negatives.

Autoencoder

The reason for the selection of the values for the Autoencoder architecture in the methodology section is based on our experience and knowledge on neural networks. We wanted the autoencoder to be small, since its only purpose is to remove the red color from the original image. A kernel size of 3 was used, since it covers as much receptive field as a bigger one if then more layers are introduced. This idea emerges from the VGG neural net design. The number of channels progressively escalates in powers of two as it's the traditional way. ReLU activation functions are fast and enough to introduce some non-linearity. The choice for the rest of parameters was arbitrary, but since the autoencoder worked the first time, and color red was correctly removed from the patches, we found no need to further change the parameters.

The starting learning rate used was 0.001, with a step scheduler to decrease it by 0.1 each epoch, the batch size 256, as much as the GPU memory supported, and the optimizer Adam. We used MSE as loss, simple but effective to find the error between the original and reconstructed images. Even though the first trainings were of 2 or more epochs, we found that the loss had already decreased to a stable value before completing the first epoch, so we ended up training the model for just an epoch.

3.2.2 Cross Validation and final Training

The dataset may contain images of patients very diverse from others. For this reason we decided to perform 5-fold cross validation. In our case, the 5-fold works by dividing all the patients found on the CrossValidation part of the dataset into 80-20% train-validation splits 5 times, in a way that each time the validation split contains a different 20% of the dataset,

and the rest is used for training. Note that the splits are done by patients rather than patches, otherwise there could be a mix of patches from all patients, but we want the model to only see a fraction of the patients and then generalize to others.

We train and set up the whole pipeline in the train set, and then make inferences and evaluate on the validation set. As an example, let P1-P80 and P81-P100 be the patients selected in a fold for the train and validation set, respectively.

- **For the Anomaly Detection**: First, we retrieve the patches from the Cropped subset of the corresponding patients P1-P80, filter out those whose patient diagnosis is positive, and train the Autoencoder on the negatives. Then we retrieve the patches of the same patients from the Annotated subset, where we know labels for individual patches, and we find the optimal threshold for patch classification there by making inference with the model and computing the red fraction, as explained in previous sections. Next, in the Cropped subset of patients P1-P80, we predict patient diagnosis and find the optimal threshold. Finally, we evaluate each of these processes on the subsets of patients P81-P100: the autoencoder on the negative Cropped, the patch classification threshold on the Annotated, and the patient diagnosis on the Cropped.
- **For the Classifier**: For each epoch, the model is trained on patches from patients P1-P80 from Annotated, and validated on patches from P80-P100. This validation is then used for early-stopping, as explained previously. The confusion matrix obtained for the validation of the last epoch is used as the final patch classification confusion matrix of the fold. Next, patient diagnosis is performed on the Cropped set but not including patients that appear in the annotated set, since those have been already used in the training of the classifier. Finally the last evaluation is performed in the Holdout.

These procedures are done in each of the 5 folds, therefore obtaining 5 confusion matrices for both the patch classification and the patient diagnosis. In order to calculate general metrics of the cross validation, we aggregate them by summing the matrices, obtaining a single confusion matrix. We can now compute the metrics:

$$Accuracy = \frac{TN+TP}{TN+FP+TP+FN} \qquad Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN} \qquad F1 = 2\frac{P*R}{P+R}$$

Accuracy: The overall performance of the model. Not very useful since our dataset is unbalanced.
Precision: The ability of the model to correctly identify the true positives.
Recall: The ability of the model to find all the true positives. This is the most interesting, since we don't want the model to miss positive cases, otherwise patients would not be treated and could be dangerous.
F1 score: The harmonic mean between Precision and Recall.

For the metrics commented, we believe that the accuracy is not that meaningful, and the most important metrics are precision and recall, specially the last one, since we don't want the models to diagnose patients with the bacteria as negative, this would be dangerous since no treatment would be applied to them and they could potentially die. It is also important to maintain a good trade-off between precision and recall, as indicated by the F1 score. Therefore, the aim of the evaluation of the models is to achieve as higher recall as possible, but also an arguably high F1 score.

With cross validation we evaluated each part of the pipeline, as well as the result of the whole pipeline (the diagnosis). Now, in order to test the model on the Holdout test set, we trained it once on the whole training set (Cropped and Annotated). We followed a similar procedure as described above for both models, and summarized in Table 2, but this time each step was done on all the patients P1-P100. In the case of the classifier, the ensemble of models of the 5 folds was used to do the patch classification, as explained previously. The final evaluation of the patient diagnosis was done on the Holdout set, with totally different patient images from the training set.

Task	**Dataset division**
Train Autoencoder	Cropped (negative)
Find optimal threshold for patch classification	Annotated
Find optimal threshold for patient diagnosis	Cropped
Train Classifier	Annotated
Evaluate diagnosis	Holdout

Table 2: Dataset divisions used for each of the tasks during the process. For training the Autoencoder, only the patients with negative diagnosis in Cropped were used.

3.2.4 Ablation Studies

Attention mechanisms

One of our main ablation studies was the development of attention mechanisms for the patient diagnosis task. Here we developed two methods over trained patch classification models to experiment if the attention approach was worth pursuing into our final used design. They were tested on the Holdout set.

The first approach used a feature extractor to extract the features from our previously trained classifier model. It uses global self-attention. The feature extractor is quite complex and it adds trainable parameters that along with the lack of regularization causes overfitting. Making nearly every diagnosis negative. Having close to 95% percent of all guesses being negative. A botched method.

The second approach used self attention in addition to a learned weight mechanism, which is, in itself, dot-product attention. It was also built on the combination of ensembles per fold (K-fold version) of our patch classification approach. After computing the scores each feature vector is weighed by attention weights –aggregation. The diagram in Figure 7 shows this idea. This model ran significantly faster, smoother and yielded better results than the first attention approach. With an accuracy of 72%.

Although the performance was better than the first attention approach the results are still worse than those of our other patient diagnosis approaches.

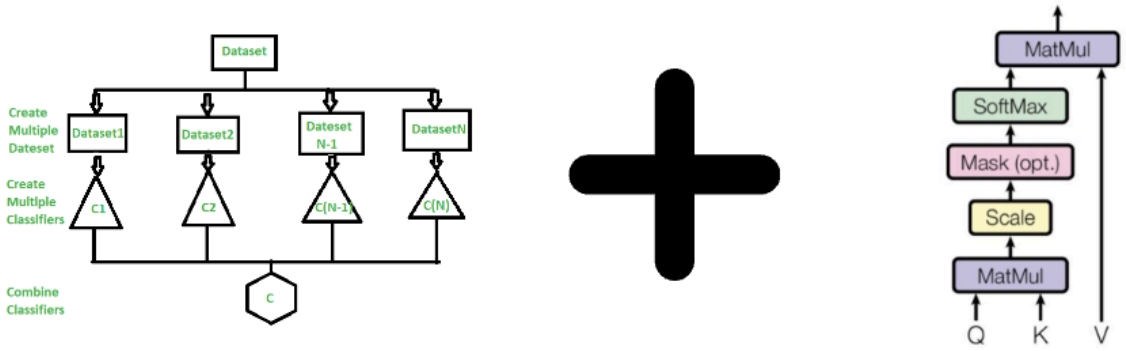

Figure 7: Ensemble with learned weight mechanism in aggregation

We conclude that this method needs more exploration and a better implementation to prove successful or to be able to replace our other patient diagnosis methods. We should perfect our implementations and search for other methods, if we want to pursue this route.

Autoencoder with batch normalization

It is worth mentioning that, for patch classification with anomaly detection, we also trained and evaluated a separate autoencoder which only differs from the first one in that it includes batch normalization between layers, but it did not outperform the simple model. While patch reconstructions seem qualitatively clearer and more precise, the classification of those patches was slightly worse, and the final patient diagnosis was even poorer. We believe more complexity on a such simple task of removing the red pixels in an image is not only unnecessary but also harmful.

4. Results

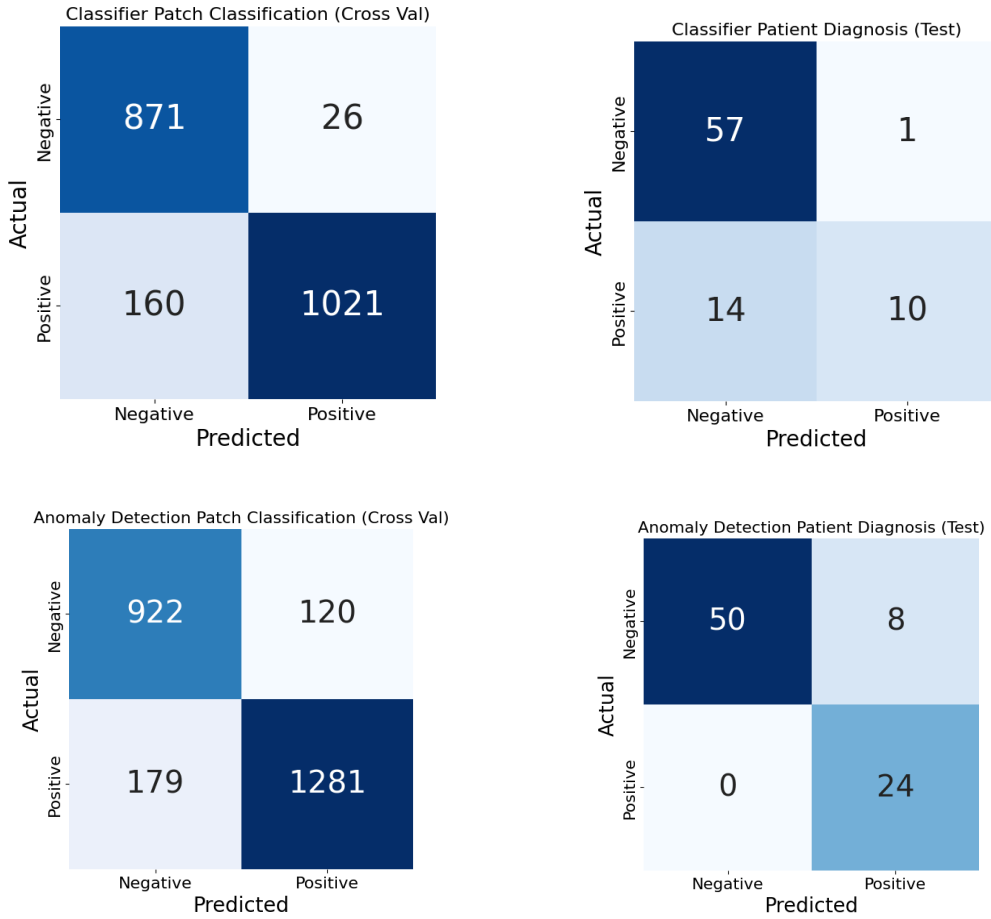

Figure 8: Confusion matrices for Classifier (top) and Anomaly Detection (bottom) approaches, for patch classification in cross validation (left) and then patient diagnosis on test set (right)

Model (step)	Accuracy	Precision	Recall	F1 score
Classifier (Cross Val)	0.91	0.97	0.86	0.91
Classifier (Test)	**0.81**	**0.90**	**0.41**	**0.57**
Anomaly Detection (Cross Val)	0.88	0.91	0.87	0.89
Anomaly Detection (Test)	**0.90**	**0.75**	**1.00**	**0.85**

Table 3: Metrics for Classifier and Anomaly Detection, in cross validation and test set

Other results

- Optimal red fraction threshold: **81**
- Red fraction ROC curve AUC: **0.94**
- Optimal % of positive patches threshold: **7.90**
- % of positive patches ROC curve: **0.97**

5. Discussion and Conclusions

Results show that the models adequately learn to classify patches and provide patient diagnosis for H. Pylori presence in microscope images, specially the anomaly detection system has given satisfactory results.

The confusion matrices in Figure 8 and the metrics in Table 3 demonstrate that both approaches, the Classifier and the Autoencoder with adaptive threshold, have a good performance classifying patches as positive or negative. False positives and false negatives are not a big concern in this step if later can be leveraged in the patient diagnosis.

On the one hand, the Classifier model is highly effective at identifying negative patients (precision: 0.90), with only one false positive. However, it struggles with correctly identifying positive patients (recall: 0.41), see how 14 patients with H. Pylori were wrongly misdiagnosed as negative. This suggests that the model may be biased toward the majority class (negative patients) or that the features extracted from positive patches are insufficiently distinct.

On the other hand, the Anomaly Detection system, while having a lower precision (0.75) than the classifier, the recall dramatically increases (1.00). The F1 score is higher for this second approach, indicating that the trade-off between the two metrics is better. The perfect recall could be explained by a threshold for % of positive patches for diagnosis being low (7.90). That is, with a low threshold, a patient with just a few positive patches would be directly classified as positive. This inevitably results in some false positives, but also in not missing actual positives, which was our initial and most important aim.

An important insight is that the difference in results between the evaluation during cross validation and at test time is very small. This indicates that there was no overfitting of the models, and both methods were robust enough to generalize well for the unseen patients in the Holdout set.

Regarding our future work we are going to discuss the topics and approaches we would do to improve our results and overall quality of our work. First, we would like to apply grid-search on our models to find better parameters, especially on our Autoencoder, with the number of layers, kernel size and channels, to try to improve its precision. Also hyperparameter tuning for all our models, potentially improving the performance of our ensemble of classifiers. A machine learning classifier, like SVMs or Random Forest, for Patient Diagnosis based on classified patches, along with more exploration on Attention approaches, is also considered in our future work.

The main thing mostly is to reduce recall on our Classifier model, as the importance of recall in medical cases is absolute. For this, we would try to increase the dataset size and address class imbalance using methods like SMOTE or more data augmentation and, additionally, feature engineering.

To sum up, our contributions include two approaches for Patch Classification, two approaches for Patient Diagnosis, we have used K-Fold Cross Validation and tested our methods on our Holdout and explained thoroughly our results and findings. The results provided by our models satisfied our aims and the performance of the Anomaly Detection system surpassed our expectations. There is still room for improvement for the false positives that we hope could be properly addressed.

6. References

- VGG: https://arxiv.org/abs/1409.1556
- Knowledge-based Attention Model for Diagnosis Prediction in Healthcare https://dl.acm.org/doi/pdf/10.1145/3269206.3271701
- Deep Learning Attention Mechanism in Medical Image Analysis: Basics and Beyonds https://www.sciltp.com/journals/ijndi/article/view/173
- Diagnosis of Helicobacter pylori using AutoEncoders for the Detection of Anomalous Staining Patterns in Immunohistochemistry Images: https://arxiv.org/abs/2309.16053
- Classification of H. pylori Infection from Histopathological Images Using Deep Learning: https://pubmed.ncbi.nlm.nih.gov/38332407/

7. Annex

In this section we show some additional images of this project.

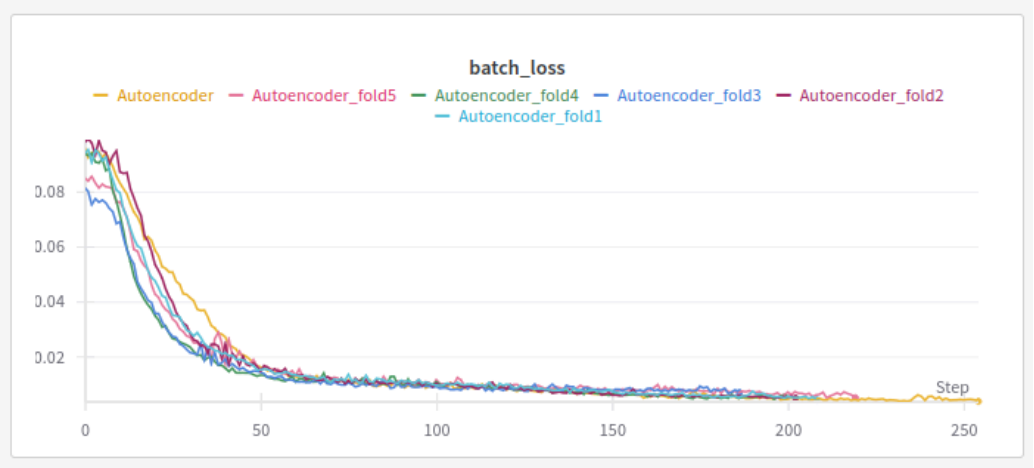

Figure 9: Training loss of the autoencoder models for each fold (Autoencoder_foldX), along with the final model (Autoencoder) trained on the whole training set

Figure 10: Validation loss of the autoencoder models for each fold. See how the model finds easier to fit some folds than others