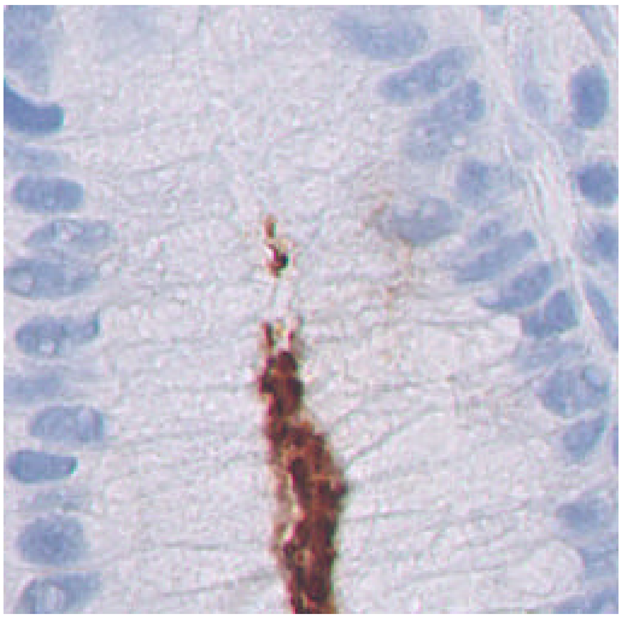
Universitat Autònoma de Barcelona

H. pylori (*Helicobacter*) Detection Medical Imaging



Submitted by:
Mustapha El Aichouni, 1668936
Arnau Solé Porta, 1630311
Josep Bonet Saez, 1633723
Jordi Longaron, 1630483

Vision & Learning

19/11/2024

1. Introduction

This work presents an approach to detecting *Helicobacter pylori* (*H. pylori*) infection in patients using advanced medical imaging techniques. *H. pylori*, a bacterium linked to various gastrointestinal diseases such as peptic ulcers and gastric cancer [1], requires early and accurate detection to improve patient outcomes and prevent complications. A traditional detection method, that is, the examination of histological samples with immunohistochemical staining, is highly time-consuming due to the size of images (120000x16000 pixels) and the different densities of the bacteria. Our work introduces an imaging-based model that uses convolutional autoencoders to reconstruct immunohistochemical histopathological images' patches extracted from healthy patients to detect anomalous staining for rapid *H. pylori* detection. For faster convergence and improved spatial reconstruction quality, we employ a modified version of U-Net [2] that uses skip connections only on the final layer of the encoder, just before the bottleneck. For patch classification and patient diagnosis, we use voting across multiple ML-based classifiers with engineered features, achieving 92% accuracy for classification and 84% accuracy for diagnosis.

1.1. Goals

The primary objective of the challenge is to develop a deep learning-based method capable of diagnosing whether patients are infected with *H. pylori* and identifying abnormal staining in image patches from these patients. To achieve such a goal, we need to know how to structure the data of patients, interact with this data, and split it into folds (Cross-validation) for further evaluation and analysis of performance. A summary of the key steps is given in the following:

- Data Preparation
- Anomaly Detection
    - Modelling
    - Measuring Anomalies
    - Feature Extraction
    - Cross Validation & Models
- Patient Diagnosis
    - Feature Engineering
    - Cross Validation & Models

All of these steps are explained in detail in *Section 2.2*.

2. Methodology

2.1. General Workflow

A 2-step general workflow for this challenge includes:

- **Patch Classification** → Classifying patches from patients as Negative (No *H. pylori*) or Positive (*H. pylori*). Some possible approaches involve using either:
  - **Classification Approach:** Using a set of annotated patches (augmented for class balance) that indicate bacterial presence, perform cross-validation and use a backbone feature extractor like ResNet18 or VGG16. A weighted loss addresses class imbalance during training. After feature learning, apply contrastive learning with triplet loss to distinguish positive and negative samples in a new feature space, which then serves as input to a binary classifier with weighted cross-entropy loss to detect *H. pylori* in each patch.
  - **Anomaly Detector:** This approach uses an Autoencoder trained on patches from healthy patients in the CrossValidation set, consisting of:
    - **Encoder**: A CNN with dimensionality reduction (e.g., max pooling) to capture a visual representation in latent space.
    - **Bottleneck**: A compact block for dimensional reduction, producing a representation Z.
    - **Decoder**: A CNN with upsampling to reconstruct the original content from latent space.

    The model is trained with a similarity measure, like Mean Squared Error (MSE), to compare input and output patches. After training, both the original and reconstructed images are converted to HSV colour space, where red pixels are extracted using specific hue bounds (e.g., -20 to 20) to quantify the red content in each patch.

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

Figure 1: Formula for the ratio of red pixels in original patch and reconstructed

The ratio F_{red} (see Figure 1) of red pixels between the original and reconstructed images serves as an error function to detect *H. pylori* in a patch. We then aim to find an optimal threshold for this function, learned from annotated patches using ROC-based adaptive thresholding. Alternatively, we can leverage the Autoencoder's self-supervised visual representations and classify anomalous patches using a fully connected neural network (FCNN).

- **Patient Diagnosis** →We aggregate the patch outputs for a global diagnosis of the patient. In this stage, the models trained for patch classification have to be evaluated on Cropped patches. For each patient, the patch outputs have to be aggregated to yield the diagnosis. These are some possible approaches:
    - **Adaptive Thresholding**: Use adaptive thresholding based on the percentage of positive patches. A typical approach aggregates patch labels from the classifier and applies majority voting, where the most common label determines the diagnosis. This percentage acts as a score indicating the likelihood of *H. pylori* presence. By calculating these scores for each patient, an optimal threshold can be determined to identify *H. pylori*. This threshold is derived from annotated diagnoses using ROC-based adaptive thresholding, where the rule is: Percentage > Threshold → *H. pylori*. The ROC curve shows the balance between precision and recall at various thresholds, with the optimal threshold being the point closest to (0,1) (FPR=0, TPR=100%).

    - **Attention Mechanism**: We can use self-attention to the percentages and other relevant features, allowing the model to focus on key indicators and enhance the decision-making process for determining whether a patient is affected.

2.2. Our Workflow

Our approach to this problem is inspired by *Cano, P. et al.* [3]. We begin by feeding patches from healthy patients into a convolutional autoencoder to reconstruct them and capture the underlying patterns of healthy samples. When anomalous data is input, the autoencoder struggles to accurately reconstruct it, signalling an anomaly. We evaluate the autoencoder using the validation set of annotated samples, measuring the loss of red pixels in the reconstructed versus original patches.

Next, we perform feature engineering, extracting metrics like the absolute and relative difference in red pixels between reconstructed and original patches and the number of red pixels in the original patch. Using these features and the ground-truth labels, we train a classifier with K-Fold, fitting a range of classifiers and applying voting. We assess the classifier's performance on the annotated data (more info in *Section 4*).

For further diagnosis, we apply the voting model patch classification predictions to all cropped and unseen data (HoldOut) and store all predictions. We then expand on the computed features for the diagnosis, incorporating metrics like the total red pixels in original patches, total red pixel difference between original and reconstructed patches, mean percentage difference, and the percentage of positive patches per patient. We use K-Fold and train our patient diagnosis model on a set of Cropped data with patient labels, then finally perform inference on unseen data.

2.2.1. Data Preparation

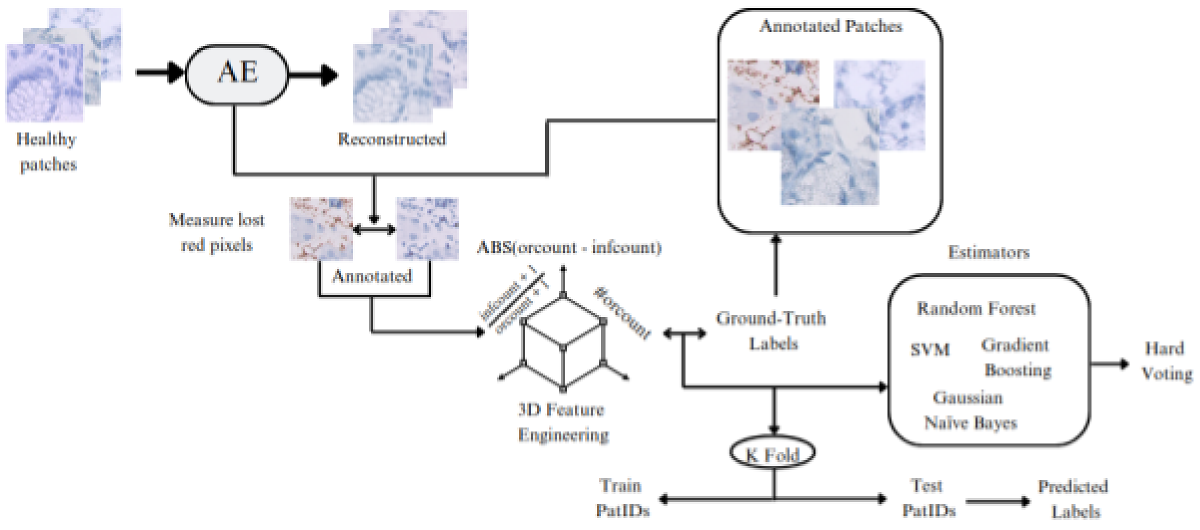


Figure 2: Our Patch Classification Pipeline

Within the CrossValidation folder, there are two data sets: Annotated and Cropped. We also carry a HoldOut set for reproducibility and evaluation. To begin modelling with the Autoencoder, we collect the patches from 20 patients labeled as sane in the Cropped folder.
The way that we split our data is by dividing Cropped patches based on patient IDs where the patches in the train set are all negative (from the 20 patients) and the rest of the patches in the validation set are a mixture of negative and positive patches. This enables the model, when tested with anomalous patches in the validation set, to produce an error signal that indicates abnormality. To simplify data interaction, we create a data structure using dictionaries where each patient's unique code serves as a key. Each patient entry contains a list of dictionaries, with each dictionary holding an image path and the corresponding label of the patient. The final datasets that have been used are managed in several ways:

- The Annotated Data is split based on validation IDs (patient) produced from the split of the AE. It corresponds to annotated patches containing both positive and negative samples.
- The Train Set for the AE uses the train split where we take 20 patients labelled as sane in the Cropped folder and train with their patches.
- A Validation Set for the AE and future patient classification, where we test the model in the validation patient IDs of the AE split that contains a mixture of both positive and negative patches from Cropped.
- A Test Set, based on the HoldOut folder where we just simply use all patient IDs of the HoldOut set.

This is the notation we give to each of the final datasets used for this project.

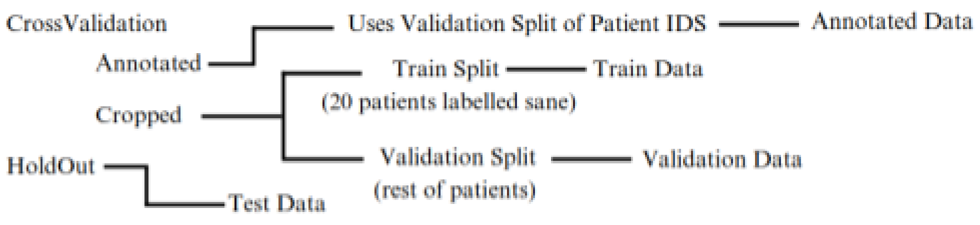


Figure 3: An overview of our data splits

2.2.2. Anomaly Detection

2.2.2.1. Modelling

With the data prepared, we split our input patches into training and validation sets based on the patient IDs that were healthy and started training our autoencoder to reconstruct patches from healthy patient samples (patches from 20 sane patients) from Cropped. To achieve faster convergence, we introduce a skip connection between the encoder's final convolution and the decoder's first deconvolution layer, inspired by U-Net [2]. We are not interested in reconstructing the image but in recolouring the image, because the autoencoder should paint the bacteria blue, producing the reconstruction error. Since colour information is not directly present in the last layer, skipping that connection mostly passes spatial information to the decoder. This transferral reduces the amount of things the model has to learn and reduces blurriness. The model avoids Max Pooling, instead using ReLU activations between each convolutional and deconvolutional layer. For normalised RGB images, a sigmoid activation is applied at the output. Patches are resized to 256x256. An overview of our model architecture is shown in *Figure 4*.

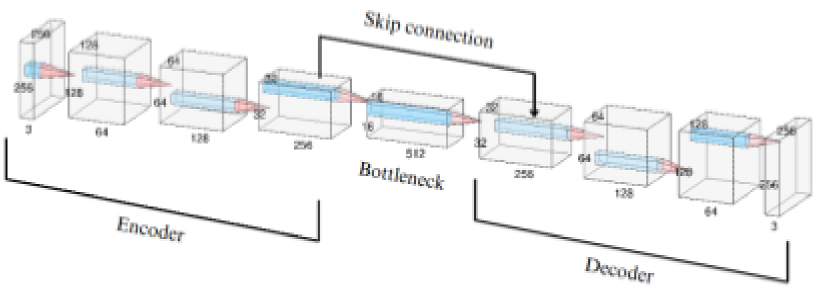


Figure 4: Our Convolutional Autoencoder Architecture, based on U-Net, figure made in NN SVG [4]

2.2.2.2. Measuring Anomalies

To detect anomalies in Annotated patches, we'll avoid using reconstruction error (MSE) as it's not an effective measure. Since red staining in the patches indicates the presence of H. pylori, we'll base anomaly detection on the count of red pixels in the reconstructed and original patches. Specifically, after training, we'll convert the patches to HSV (Hue, Saturation, Value) colour space and measure the loss of red pixels during reconstruction. By setting upper and lower hue bounds to identify red, we can count the number of red pixels in each patch and use this as an anomaly indicator.

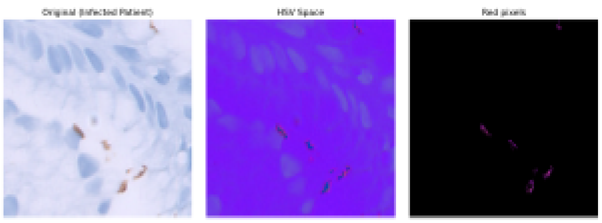


Figure 5: Identifying red pixels in an original patch of an infected patient

After calculating the number of red pixels in both the original and reconstructed patches, we compute a measure called F_{red} by dividing the red pixel count of the reconstructed patch by that of the original patch. A low F_{red} value (closer to 0) indicates an anomaly, while a high value (closer to 1) suggests no anomaly. For numerical convenience, we adjust the calculation by using $\frac{infcount + 1}{orcount + 1}$, where $infcount$ is the count of red pixels in the reconstructed patch, and $orcount$ is the count of red pixels in the original patch. After F_{red}, we also create a set of features that include $orcount$ and $|orcount - infcount|$ which corresponds to the absolute difference between the count of red pixels in the original and the count of red in the reconstructed.

2.2.2.3. Feature Extraction

Now that we have a set of engineered features that can be used for patch classification we can start by computing these features for the Annotated Data, the Validation Data, and the Test Data (refer to Section 2.2.1.). All of the features are then saved in Pickle format for further use.

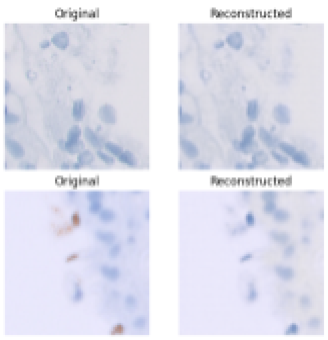


Figure 6: Original and reconstructed patches of the Autoencoder

In this illustration (Figure 6), it is visualised the reconstruction of patches from the Annotated Data where validation ids from the AE are used. We know the labels of the patches, so it's easy to identify how the model reconstructs anomalous data.

2.2.2.4. Cross Validation & Models

We can now begin classifying our patches since all the datasets have been prepared and the features have been computed. For the Annotated Set, we generate X and Y, where X are the features from every patient ID patch in the validation IDs from Annotated and Y are the target ground-truth labels. We use the same procedure for the Validation Set, which is used to test our patient diagnosis techniques. Additionally, we test patient diagnosis in the Test Set. We employ SVM, Random Forest, Gradient Boosting, and Naive Bayes as our classification estimators (refer to *Figure 2*). In addition to using a voting classifier, whose predictions we keep, we further generate predictions for these estimators. We use Sklearn's `cross_val_predict` to calculate predictions after fitting the models to the Annotated Set. The key idea is that we divided the data in the Annotated Set into five folds, then gathered all of the predictions from each fold and combined them to obtain an objective, global picture of how our model predicts our patches. We take into consideration the fact that patches from the same patient cannot be used for both training and testing simultaneously.

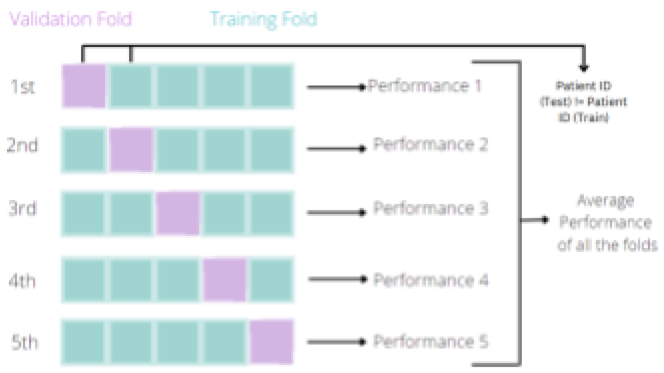


Figure 7: The K-Fold on patch classification. Figure adapted from https://pub.towardsai.net/k-fold-cross-validation-for-machine-learning-models-918f6ccfd6d

Once we have the Voting model which is the one we believe works best we perform inference with it on the Validation Set and Test Set for further use in the patient diagnosis.

2.2.3. Patient Diagnosis

We can use global features calculated from all of the patients' patches to diagnose them after we have predictions for the patches. We address this issue by gathering various engineered features, all of which are derived from the previously calculated patch-level features. We believe that using ML classifiers in conjunction with all of the information that is available from a patient's patches can improve diagnosis.

2.2.3.1. Feature Engineering

For diagnosing patients we have designed several features that can then be used for classifying whether a patient is sane or infected:

1. Total Red Pixel Count: The aggregated number of red pixels in all the patient's patches. Numerically, it will be $\sum_{j=1}^{N} orcount_j$.
2. Total Pixel Difference: The aggregated absolute difference in the number of red between the patches and their reconstruction. $\sum_{j=1}^{N} \left| orcount_j - infcount_j \right|$
3. Mean Percentage Difference: The mean of the aggregated percentage difference in red pixels. $\frac{1}{N} \sum_{j=1}^{N} \frac{infcount_j + 1}{orcount_j + 1}$
4. Percentage of positive patches: The relative amount of positive patches that the patient has based on patch classification prediction. $\frac{positive}{total} X\, 100$

We believe that these features hold important information and are able to explain how the quantity of red pixels is distributed between the patches.

2.2.3.2. Cross Validation & Models

We compute these features for both the Validation and Test Data, using the same definitions for X and Y as in patch classification. Several estimators, including Random Forest, Gradient Boosting, and kNN, are trained on the Validation Data. Additionally, we employ a voting classifier with these estimators, using soft voting.

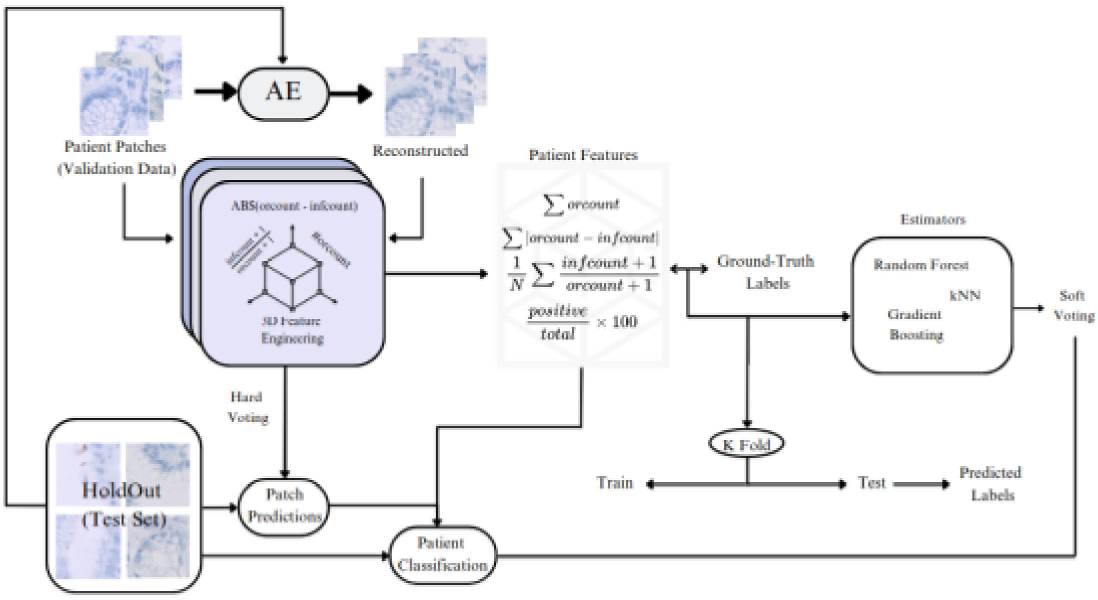


Figure 8: Our Patient Classification Pipeline

For patient classification, we follow the same training procedure as in patch classification. This involves performing K-Fold cross-validation on the Validation

Data, predicting on the test fold of each iteration, and obtaining global predictions for the entire dataset. Unlike patch classification, there is no concern about biased results due to patches from the same patient, as we use global features at the patient level rather than patch-level features. We finally perform inference with the Voting model on unseen data (Test Data).

3. Experimental Design

3.1. Dataset Description

We have obtained the dataset from The Quiron database. It has WSI scored by a pathologist according to H. pylori density. Patients are classified as NEGATIVE, or as POSITIVE (LOW and HIGH DENSITY) for the presence of H. pylori. Each WSI contains 2 sections of several gastric mucosa samples and one section extracted from an external positive control to check the validity of the immunohistochemical staining.

Patches of size 256x256 were cropped along the border of the tissue samples of one of the sections.

We have divided the total data into four distinct splits (as in *Section 2.2.1*), each serving a specific purpose:

- Train Data: Used exclusively to train the autoencoder for image reconstruction.
- Annotated Data: Designed for training in patch classification and visualisation tasks.
- Validation Data: Used for training patient classification models.
- Test Set: Reserved for evaluating performance.

The Annotated Data, Validation Data and Train Data are composed of the data that the expert pathologist annotated. Due to the imbalance of the number of positive annotated patches, they have been augmented by cropping windows along the tissue border from the point the annotated patches were extracted from. Train data is composed of 20 patients with 18329 sane patches. Annotated Data and Validation Data have 135 Patients, 57 sane and 78 infected. Validation Data is a subset of Cropped and Annotated Data contains the patient ids of Validation Data.

The Test Set is composed of the images in the HoldOut set, it only has the patients labelled (not the patches). We used them for the final evaluation of our methods. It has 116 Patients, 58 sane and 58 infected.

3.2 Experiments and Metrics

To evaluate the autoencoder we plotted some of the model's results on Annotated Data (see *Figure 6*). We also have a loss of 0.0004.

For both patch classification and patient diagnosis we performed the following steps:

1. Cross-Validated Predictions: For each classifier, we generated predictions using `sklearn.model_selection.cross_val_predict` with 5 folds. This method produces out-of-sample predictions for each data point, simulating a real-world testing scenario.
2. Confusion Matrix: For each estimator, a confusion matrix was plotted, visualizing the number of true positives, true negatives, false positives, and false negatives. This helps in understanding the distribution of predictions and identifying any bias in the model (e.g., favoring one class).
3. Classification Report: We generated a detailed classification report for each estimator, which includes:
    - Precision: The ratio of true positive predictions to all positive predictions (i.e., how many of the predicted positives are actually correct).
    - Recall (Sensitivity): The ratio of true positive predictions to all actual positive cases (i.e., the model's ability to detect positive cases).
    - F1-Score: The harmonic mean of precision and recall, providing a balanced measure that is particularly useful when dealing with class imbalance.
    - Support: The number of actual occurrences of each class in the dataset.

For our final model, we created an ensemble model using a Voting Classifier. Two configurations were used:

- Hard Voting: The ensemble predicts the class label that receives the majority vote from individual classifiers. We used it for patch classification.
- Soft Voting: The ensemble averages the predicted probabilities of each classifier and selects the class with the highest average probability. We used it for patient diagnosis.

4. Results

Since we use Voting in order to make the final decision, we have a plethora of metrics for both patch classification and patient classification. We concluded that the most relevant metrics, however, are the final metrics after voting.

4.1 Patch Classification

	Precision	**Recall**	**f1-score**	**support**
Sane	0.90	0.92	0.91	1096
Infected	0.94	0.93	0.93	1460
Accuracy			0.92	2556
Macro Avg	0.92	0.92	0.92	2556
Weighted Avg	0.92	0.92	0.92	2556

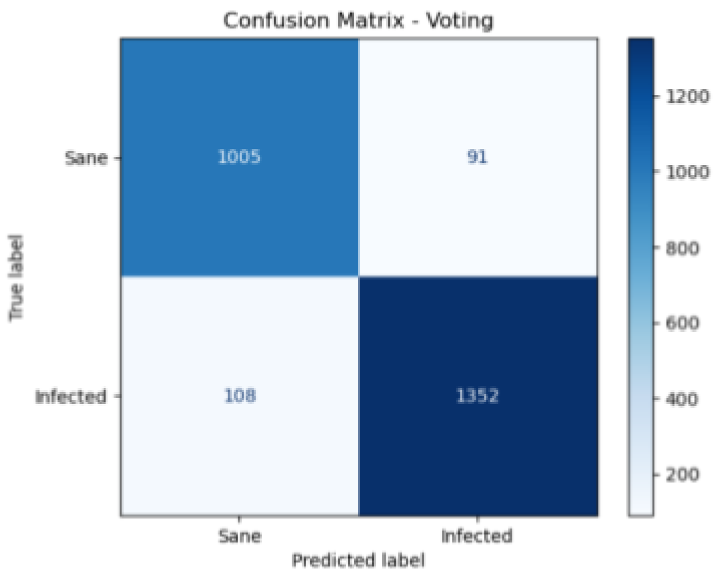


Figure 9: Confusion Matrix for patch classification

4.2 Patient diagnosis

	Precision	Recall	F1-score	Support
Sane	0.83	0.86	0.85	58
Infected	0.86	0.83	0.84	58
Accuracy			0.84	116
Macro Avg	0.85	0.84	0.84	116
Weighted Avg	0.85	0.84	0.84	116

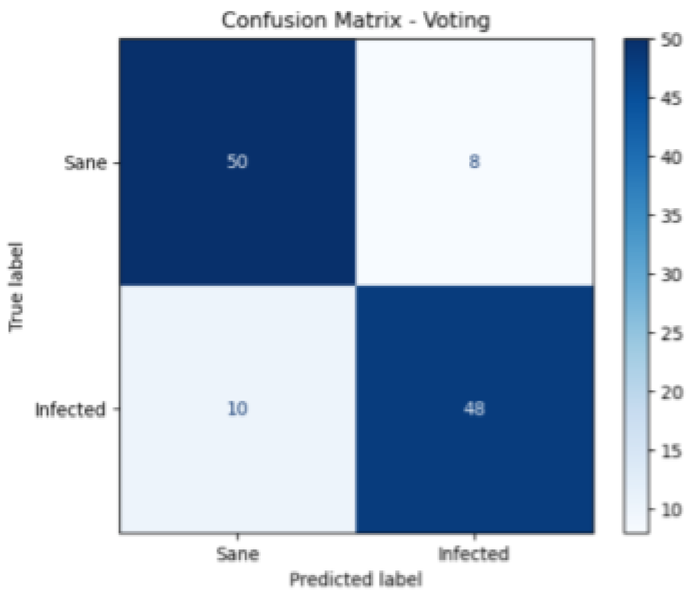


Figure 10: Confusion Matrix on Patient Diagnosis

5. References

[1] *Infección por helicobacter pylori (H. pylori)* (no date a) *Mayo Clinic.* Available at: https://www.mayoclinic.org/es/diseases-conditions/h-pylori/symptoms-causes/syc-20356171 (Accessed: 19 November 2024).

[2] Ronneberger, O., Fischer, P. and Brox, T. (2015) *U-Net: Convolutional Networks for Biomedical Image Segmentation*, *arXiv.org*. Available at: https://arxiv.org/abs/1505.04597 (Accessed: 10 November 2024).

[3] Cano, P. *et al.* (2023) *Diagnosis of helicobacter pylori using autoencoders for the detection of anomalous staining patterns in immunohistochemistry images*, *arXiv.org*. Available at: https://arxiv.org/abs/2309.16053 (Accessed: 10 November 2024).

[4] *NN-svg* (no date) *NN SVG*. Available at: https://alexlenail.me/NN-SVG/ (Accessed: 11 November 2024).