Mammograms classifier

Ensemble of CNN models for microcalcifications clusters detection

Alessia De Ponti Francesco Leonetti

February 2023

Introduction

The aim of this project is to create and train a neural network able to detect the presence of microcalcifications clusters in mammograms.

Methods

A Convolutional Neural Network (**CNN**) hypermodel was designed for this task. Using a **5-fold + hold-out** procedure, 5 instances of the hypermodel were selected, trained and assessed on an internal test set.

Then, an **ensemble** learning strategy was implemented: treating each of these 5 models like an *expert*, the final response comes from a weighted average of the single experts' predictions. These weights are trained to maximize the accuracy of the ensemble.

Dataset

The dataset is made up of **797** images:

- **414** represent normal breast tissue (Label = 0)
- **383** represent microcalcifications clusters (Label = 1)

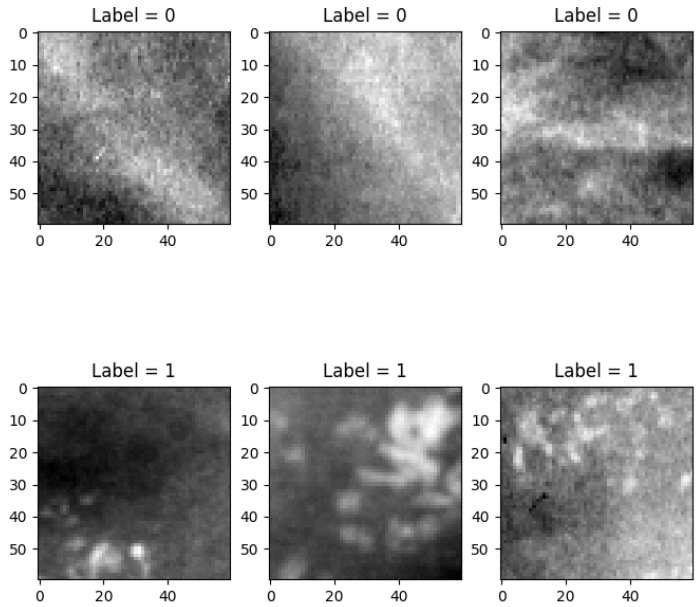


Figure: Random images extracted from the dataset

An external dataset will be used after training to assess the generalization capability of the final models'ensamble.

Dataset transformations

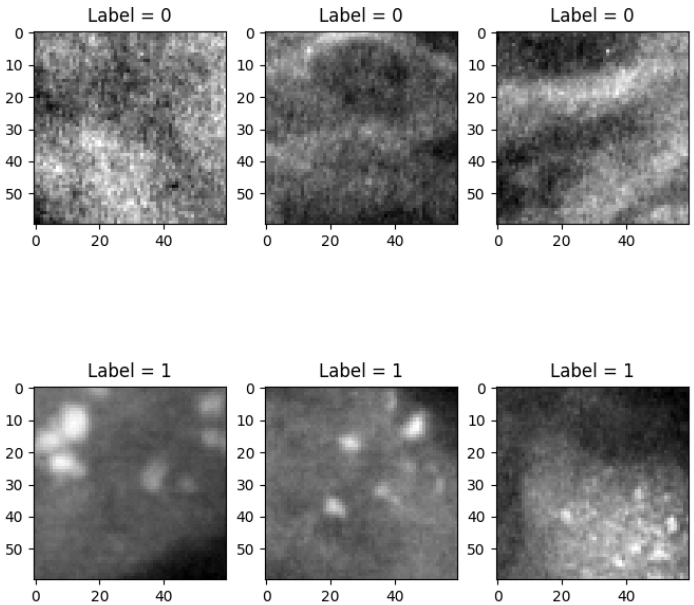
The user can select whether to process the aforementioned dataset as it is or to perform any of the following procedures:

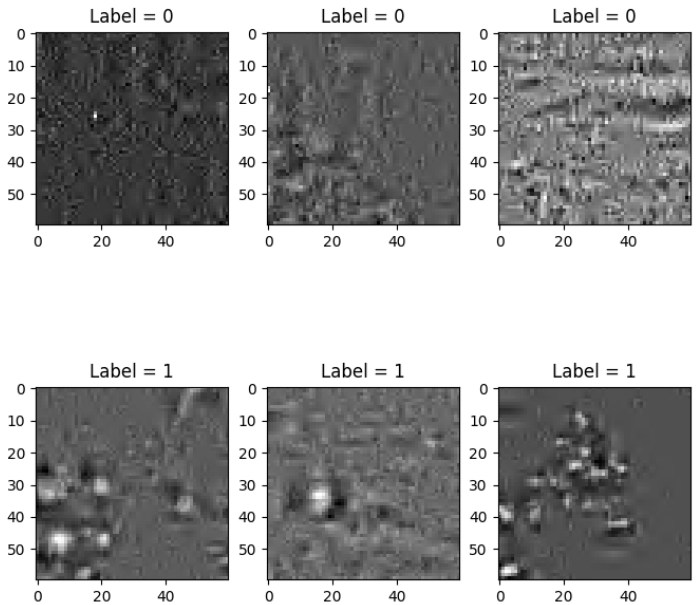
- **Data augmentation**

using `keras.ImageDataGenerator`

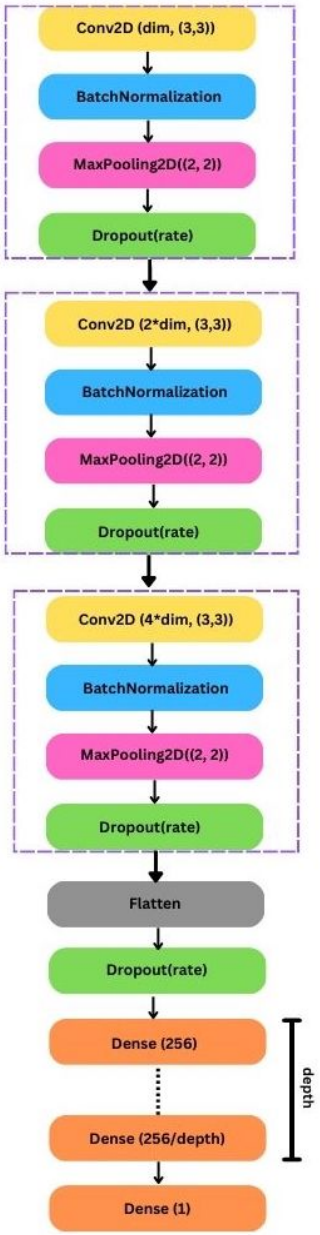
- **Wavelet filtering**

using `matlab.engine`





Hypermodel



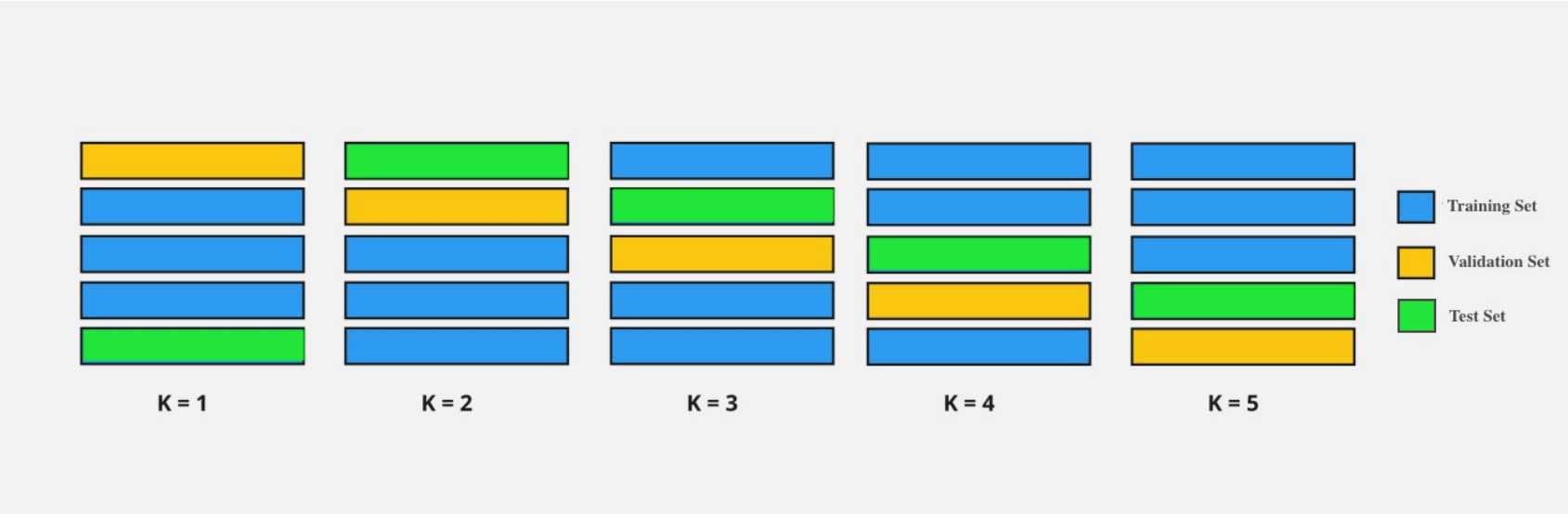
The deisgned hypermodel is made up of three convolutional blocks and a final fully-connected block.
The user-settable hyperparameters are:

- **dim:** the number of output filters in the first convolution (this number doubles after each convolutional layer)
- **rate:** dropout rate
- **depth:** the number of layers in the fully-connected block.

Note that these parameters influence the model's complexity, its generalisation capability and its overall performance.

Model selection and model assessment

Given the hypermodel and an hyperparameters space, the best model is selected with an **internal Hold-out** (Validation set = 25% of development set). A **5-fold cross-validation** procedure is chosen to evaluate the model's performance.



Using `keras-tuner.BayesianOptimizator`, hyperparameters search is performed, exploring the user-selected `searching_fraction` of the hyperparameters' space.

Ensemble

At this point we are left with 5 trained models (one for each fold), so an **ensemble** learning strategy is imlpemented using `PyTorch`. The response of the committee is going to be a **weighted average** of the single predictions. These weights are trained to maximise the ensemble's accuracy and represent the *reliability* of each expert among the committee.
Finally, the ensemble's performance is tested on an external test set (label=1 cases only).

Pros:

- Robustness
- Flexible architecture
- Dynamic control of complexity

Cons:

- Computationally expensive
- Possibly redundant

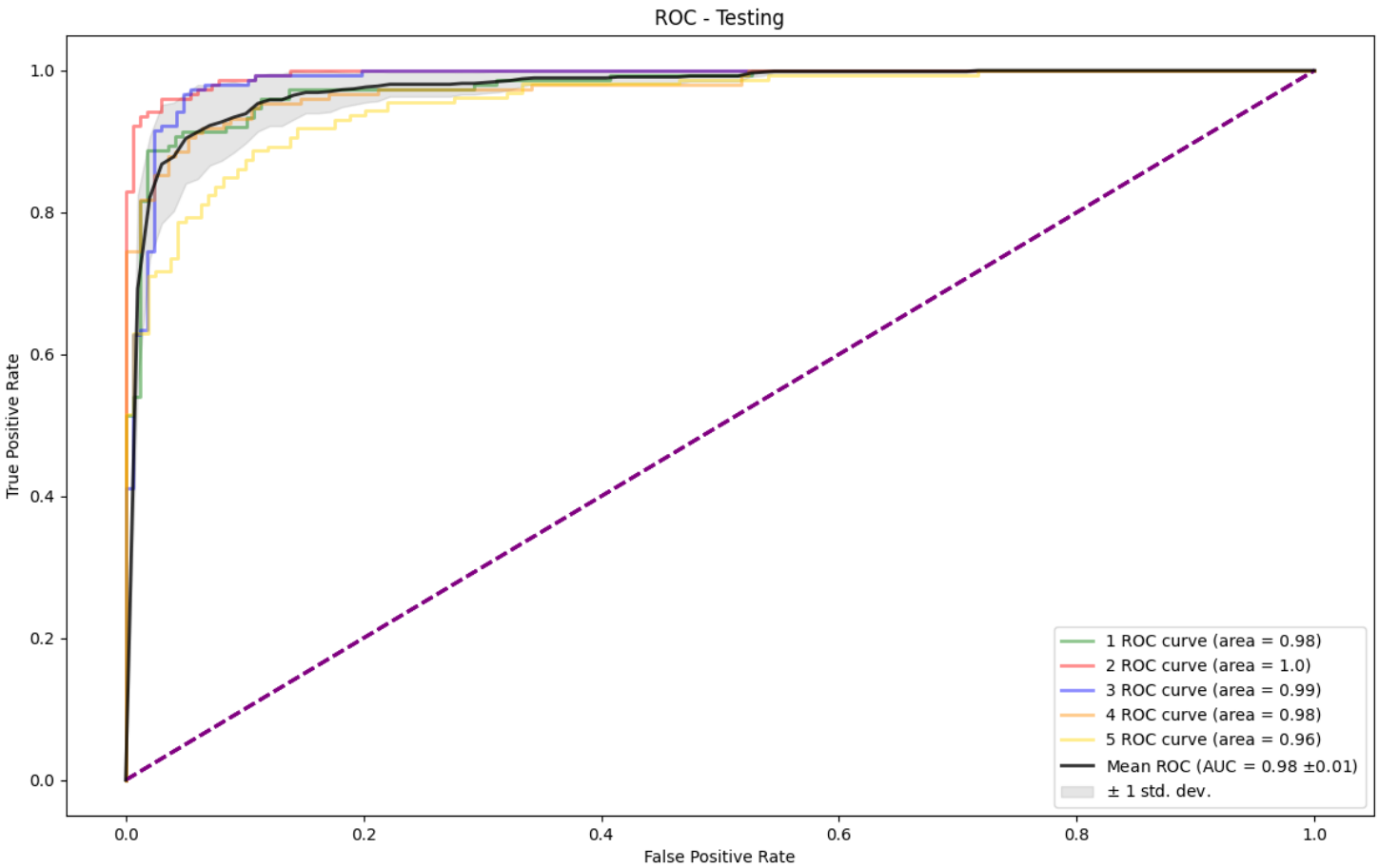
Classes

In order to implement this workflow, two custom-made classes were used:

- `Data`: used to handle dataset
  - performs data augmentation and wavelet filtering
  - `get_random_images`: returns random patterns from one or both classes
- `Model`: used to carry out the aforementioned training strategy
  - `tuner`: performs hyperparameters search
  - `fold`: performs k-fold
  - `get_predictions`: returns each model's prediction for all the patterns (used as input for the ensemble model)
  - `get_ensemble`: trains and saves the ensemble

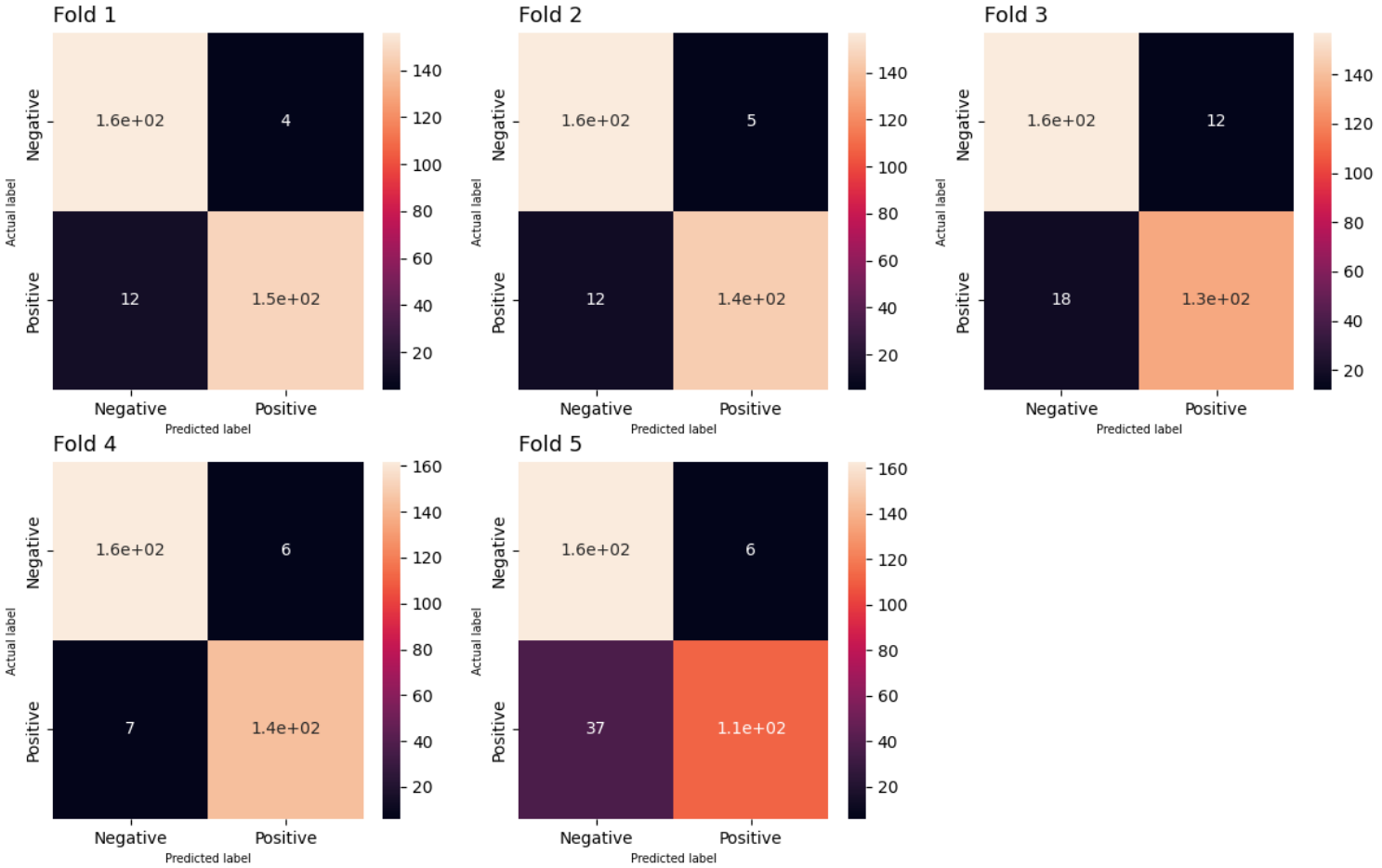
Results (K-fold)

Here is an example of ROC curves and AUCs for each of the 5 folds, togheter with their *mean* and *stdev*.



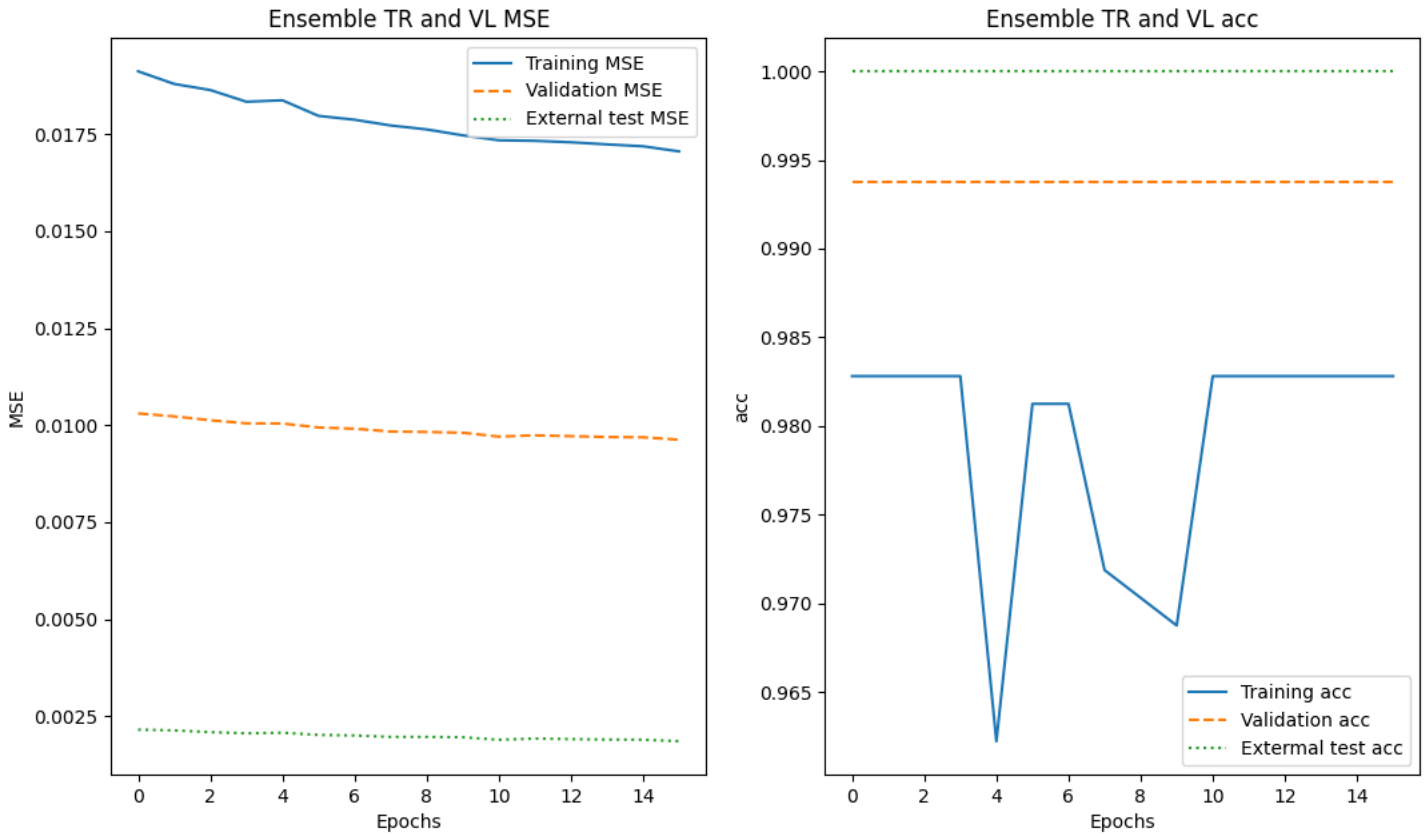
Results (K-fold)

Here are some examples of Confusion Matrices obtained for the 5 folds.



Results (Ensemble)

Here is an example of ensemble's learning curve and accuracy plot.



GradCAM and interpretability

Selecting the most reliable model, according to the ensemble's weights, the algorithm **GradCAM** was employed to highlight which regions of the input images are relevant in the decision making process.

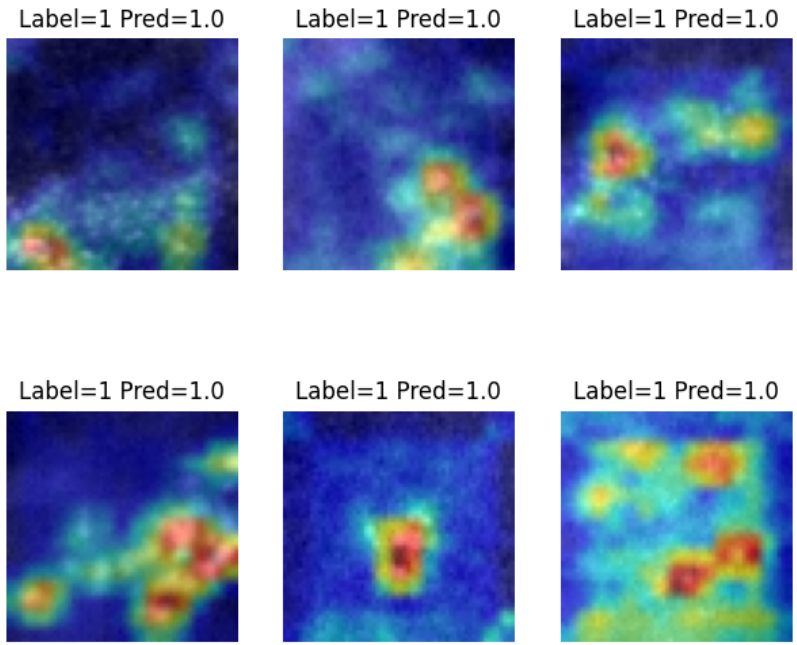


Figure: Random images visualised with gradCAM