Applied AI in Biomedicine - Project Report
Team Name: RagazzAI

Juliette Treyer, Tanguy Rolland, Vincent Villeneuve, Matthieu Varenne

242842, 242816, 276573, 242843

April 7, 2025

1 Introduction

In this project, we address a *classification* problem on a lung cancer CT scan dataset. More than simply image classification tasks, this challenge focuses on both **model architecture and confidence estimation**, requiring robust methods to handle the clinical significance of the data. Our approach was to design models capable of producing accurate classifications of malignancies while estimating the reliability of their predictions.

2 Problem Analysis

The dataset comprises 2363 patients, each providing two types of CT slices and a malignancy score ranging from 1 to 5. Nodules are labeled in two ways:

- **Binary classification**: benign (classes 1-2-3) vs. malignant (classes 4-5).
- **Multi-class classification**: exact malignancy score (1 to 5).

The dataset is imbalanced, with benign nodules being far more frequent than malignant ones, and the class 3 being overrepresented, while 1 and 5 are clearly underrepresented. Addressing this imbalance represents a significant challenge in developing effective models.

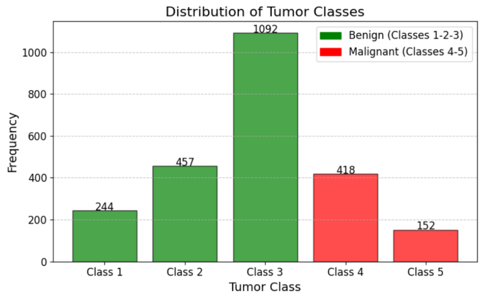

Figure 1: Class distribution of the dataset

No missing labels were identified. We also checked for potential duplicates, and none were found.

We performed PCA on the image features extracted by a pre-trained VGG19 (imagenet weights) in order to visualize the classes in their latent representation.

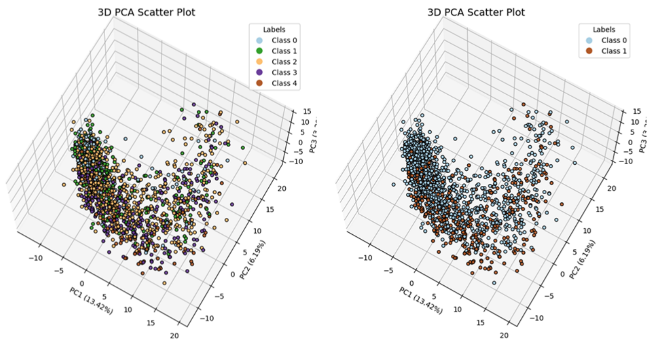

Figure 2: Images features from VGG19 in the first principle plane of the PCA for zoomed images

This representation shows how the multiple classes on the left of the figure might be especially difficult to separate since their representing clusters in the latent space of the VGG are intertwined. We also point out the issue of class imbalance since we can immediately see how the Malignant class (1) is underrepresented.

3 Method

3.1 Data Preparation and Augmentation

In this section, we describe the preprocessing and augmentation steps applied to the dataset to improve the model's generalization ability.

• **Resizing the images:** First, all images were resized to **224×224** pixels, a standard size for deep learning models, especially when using pre-trained architectures. This ensures that the images are compatible with the model input while balancing computational efficiency and retaining essential image features.

• **Normalisation:** All images were also **normalized** to a range of **[0, 1]** to facilitate the training process. Both for the dataset containing the full images and the one containing the nodule images, the images have very variable intensity distributions. Therefore, we will chose to apply min-max scaling image by image, and not on the entire datasets, during preprocessing.

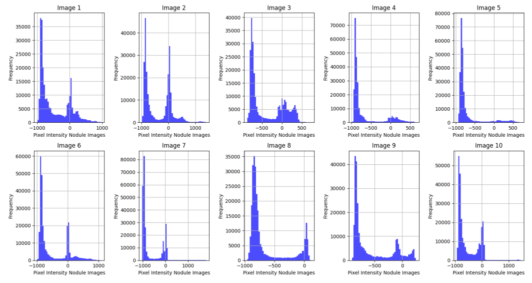

Figure 3: Pixel intensity distributions for nodule images

• **Data Augmentation:** Given the potential for overfitting and to improve robustness, we applied **data augmentation**. This included flips, Gaussian noise, random zoom as well as brightness and contrast modifications. These transformations helped to diversify the training set and address class imbalance by increasing the representation of underrepresented classes.

Two separate strategies were adopted for data augmentation. First, data augmentation was performed on the entire dataset before splitting, therefore resulting in training, validation and test sets which all contained augmented data and almost balanced classes. In a next part, data augmentation was performed after the splitting and only on the training set. While this second strategy yielded less good results, it is supposed to be more accurate, as the imbalance of the provided dataset is probably representative of most datsets of the kind.

To balance class distribution, we also used from scikit-learn the library *compute class weight* function. The weights were assigned inversely proportional to the frequency of each class in the dataset. The weights were then incorporating into the loss function for the training.

• **Splitting of the dataset:** After applying preprocessing and augmentation, we split the dataset into training, validation, and test sets. During the splitting, stratification was performed on the classes or on the lables for the multiclass and the binary models respectively.

These steps ensured the model was trained on a diverse and well-represented dataset, improving its ability to generalize to unseen data.

• **Conversion to RGB images:** In order to prepare the grayscale images for transfer learning, it is also necessary to convert the images into RGB format. This is done using a Gray2RGB custom TensorFlow layer which converts grayscale images into RGB format and normalizes them using ImageNet mean values.

3.2 Compilation

• **Loss Function:** `categorical-crossentropy` is suitable for multi-class as well as for binary classification. Here is the final formula for this loss function:

$$\mathcal{L}_{\text{batch}} = -\frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{C} y_{ij}\log(\hat{y}_{ij})$$

Where:

$$\begin{aligned}
\mathcal{L}_{\text{batch}} &: \text{Batch categorical cross-entropy loss}\\
N &: \text{Number of samples in the batch}\\
C &: \text{Number of classes}\\
y_{ij} &: \text{True label for class } i \text{ of the } j\text{-th sample (one-hot encoded)}\\
\hat{y}_{ij} &: \text{Predicted probability for class } i \text{ of the } j\text{-th sample}\\
\log &: \text{Natural logarithm function}
\end{aligned}$$

- **Optimizer:** An Adam optimizer with an initial learning rate of 0.001 was chosen for its fast convergence and robustness to hyperparameters.

3.3 Training Process and Callbacks

For the training process, the data were divided into training (80%), validation (10%) and test sets (10%) sets.
Several callbacks were implemented:

- **ReduceLROnPlateau:** Dynamically reduces the learning rate by a factor of 0.1 if the validation accuracy plateaus for 5 consecutive epochs, with a minimum learning rate of 10^{-6}.
- **ModelCheckpoint:** Saves the model's weights to a file (`best_weights.keras`) whenever the validation accuracy improves, ensuring the best model is preserved.
- **EarlyStopping:** Halts training if the validation accuracy does not improve for a specific number of consecutive epochs and restores the best weights to prevent overfitting.

4 Experiments

4.1 Handling class imbalance

4.1.1 First strategy - Augmentation on the entire dataset

In this first strategy, augmentation was performed on the entire dataset before splitting. 500, 400, 0, 400, and 500 samples were added to the classes 1 to 5 respectively. The resulting training, validation and test sets have the following sizes:

- Training set: 3170 samples
- Validation set: 397 samples
- Test set: 396 samples

However, even though this strategy improved the results, especially for binary classification, it was clear that there was an underlying overfitting issue, since the less represented classes would end up having even more augmented samples than original ones, thus overspecializing the model on these samples.

4.1.2 Second strategy - Augmentation on the training set only

In order to reduce overfitting, we thought about augmenting only the train set (X_train) such that the model would still be validated and tested over sets which had the real-world distribution.

The result of this strategy was difficult to compare with the previous strategy since the test sets were different, but we argue that such trained model would generalize better on the future test set.

4.1.3 Last strategy - Class weights

Rather than augmenting the dataset, we finally tried class weights, meaning weighing each class by its inverse-frequency in the dataset. Less represented classed are weighted more in the loss to make sure the model does not overfits to the majority class.

4.2 Models

4.2.1 Basic CNNs

We initially experimented with simple Convolutional Neural Networks (CNNs) to establish a baseline for both binary and multi-class classification tasks. These models consisted of several convolutional and pooling layers, followed by fully connected layers with dropout for regularization. While these networks were able to capture essential image features, their performance quickly reached a plateau due to the simplicity of the architecture. This limitation prompted us to explore more complex models, including pre-trained architectures and advanced regularization techniques, to improve prediction accuracy and robustness.

4.2.2 Transfer Learning

Transfer learning was performed using several different pre-trained models. Following models were used:

- VGG19
- MobileNetV2
- EfficientNetB0
- ResNet50

For each one of those models, we designed a custom classification head which replaces the one from the pretrained model. We used Imagenet weights. We first train the classification head by freezing the rest of the model, then we can unfreeze some of the last layers of the base model to further fine-tune it.

4.2.3 Model from the paper

We also used an improved CNN model developed in [1]. The CNN model is designed from scratch to optimize the learning of visual and spatial features of image blocks. The process includes pooling layers to reduce parameters, fully connected layers with a dropout (loss rate: 0.3) to avoid overfiting, and a softmax layer to classify images into two categories. These innovations are designed to reduce computations and improve model efficiency.

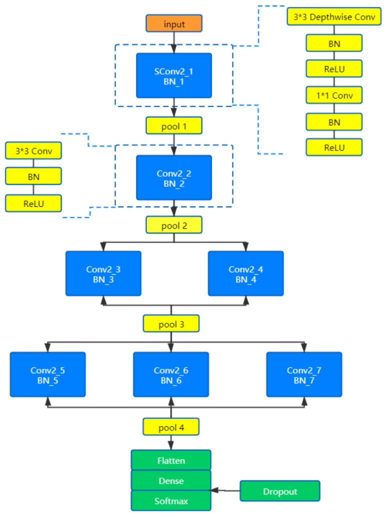

Figure 4: The improved CNN network architecture

4.3 Explainability - GradCAM

Since we are facing a Biomedical classification problem which is linked diagnosis problem, we gave some thoughts to the explainability of our models. We therefore implemented a Grad-CAM explaination pipeline for the MobileNetV2. The goal was to try to see whether the model was focusing on the right areas of the image, especially we wanted to see if the model was capable of identifying the nodule area in the full slices. Unfortunately, we were only able to implement it for an early version of our MobileNetV2 which was not the best-performing one.

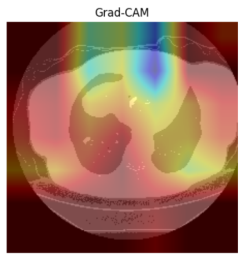

Figure 5: Grad-CAM heatmap of MobileNetV2 (low performance)

As we can see the focus area do not really make sense at this point. We would have expected our best models to show much clearer focus points.

5 Results

The performances of each model are reported in the tables below. Each table corresponds to either multi-class or binary classification, and either with full images or zoomed images.

Overall, classifying zoomed slices is always easier than classifying full slices, and binary classification is always easier than multi-class classification.

For example, our best results for the binary classification of zoomed slices were 81% of accuracy without augmentation (Finetuned MobileNetV2), and 89% for the VGG19. Both models perform well, but for different reasons (see Discussion).

We also provide here the 4 confusion matrices for the MobileNetV2 training, which was performed only with class weights and no augmentation.

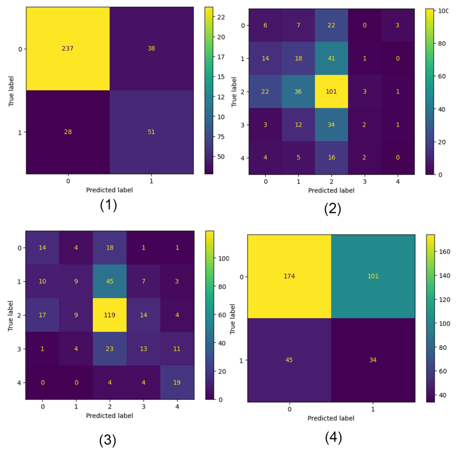

Figure 6: Confusion matrices for Finetuned MobileNetV2 on zoomed slices (1)&(3) and full slices (2)&(4)

Table 1: Metrics obtained for zoomed slices and binary classification

Model	Balancing strategy	Accuracy	Precision	Recall	F1
Basic CNN	-	0.82	0.81	0.81	0.81
MobileNetV2	class weights	0.81	0.73	0.75	0.74
VGG19	Train set	0.86	0.86	0.79	0.81
IMCNN	Full dataset + class weights	0.86	0.86	0.82	0.83
VGG19	Full dataset + class weights	0.89	0.89	0.88	0.89

Table 2: Metrics obtained for zoomed slices and multi-class classification

Model	Balancing strategy	Accuracy	Precision	Recall	F1
Basic CNN	-	0.48	0.42	0.48	0.44
MobileNetV2	class weights	0.49	0.41	0.43	0.40
VGG19	Full dataset	0.64	0.69	0.64	0.65
EfficientNetB0	Full dataset	0.59	0.58	0.60	0.58
ResNet50	Full dataset	0.21	0.13	0.20	0.16
VGG19	Train set	0.51	0.54	0.50	0.52

Table 3: Metrics obtained for full slices and binary classification

Model	Balancing strategy	Accuracy	Precision	Recall	F1
Basic CNN	-	0.71	0.51	0.71	0.59
MobileNetV2	class weights	0.59	0.52	0.53	0.51
VGG19	Train set	0.71	0.61	0.56	0.56
IMCNN	Full dataset + class weights	0.68	0.65	0.60	0.60

Table 4: Metrics obtained for full slices and multi-class classification

Model	Balancing strategy	Accuracy	Precision	Recall	F1
Basic CNN	-	0.42	0.17	0.42	0.25
MobileNetV2	class weights	0.36	0.21	0.21	0.20
VGG19	Full dataset	0.49	0.56	0.47	0.48
VGG19	Train set	0.15	0.06	0.18	0.09
VGG19	Full dataset + class weights	0.64	0.66	0.64	0.64

6 Discussion

Our top models demonstrate strong performance, but we identified two key concerns:

The handling of class imbalance during training significantly affects how the results can be interpreted. For instance, a model trained on a fully balanced dataset (e.g., VGG19) may achieve a high F1-score and strong performance on the current dataset but could under-perform on the future test set. This is because the class distribution in the test set will differ significantly, potentially revealing overfitting on the less-represented classes.

On the other hand, a model trained without augmentation but with class weights faces greater challenges during training, as the class imbalance skews it toward the majority classes. However, this approach could better preserve performance on the future test set.

The second point is about the fact that zoomed slices are easier to classify than full slices because the important information (the nodule size, shape...) is more focused. A full slice can be seen as a "noisy" image from the point of view of the model, since it has to additionally learn how to identify the important area in the image along with determining the class of the nodule.

This is specifically what we could have been able to highlight with grad-CAM.

7 Conclusions

In conclusion, we were satisfied with our results but believe we could have explored the imbalance issue further, potentially by implementing custom weighted losses. Additionally, we think exploring deeper into explainability would have been a valuable focus.

References

[1] Research on the Classification of Benign and Malignant Parotid Tumors Based on Transfer Learning and a Convolutional Neural Network