جامعة مولاي إسماعيل
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵎⵓⵍⴰⵢ ⵉⵙⵎⴰⵄⵉⵍ
UNIVERSITÉ MOULAY ISMAÏL

كلية العلوم
ⵜⴰⵖⵔⵎⴰⵜ ⵏ ⵜⵓⵙⵙⵏⴰ
FACULTÉ DES SCIENCES

Filière : Sciences de la Matière Physique

MÉMOIRE DE PROJET TUTORÉ

Présenté

Par:

Ayoub EL MHAMDI
youssef MADANE

UTILISATION DU DEEP LEARNING POUR IDENTIFIER LES NODULES PULMONAIRES CANCÉREUX SUR LES IMAGES DE TDM

Soutenu le **/07/2023 devant la Commission d'Examen :

Pr RAJAE Sebihi	**Encadrant**

REMERCIEMENTS

Nous tenons à remercier d'abord toutes les équipes pédagogiques de **la Filière Science de la Matière Physique** de la Faculté des Sciences à Meknès, ainsi que les professeurs ayant contribué activement à notre formation.

Nous profitons de cette occasion, pour remercier vivement notre Professeur **RAJAE SEBIHI** qui n'a pas cessé de nous encourager tout au long de l'exécution de notre Projet de Fin d'Études, ainsi que pour sa générosité et ses compétences en matière de formation et d'encadrement. Nous lui sommes reconnaissants pour ses aides et conseils précieux qui nous ont permis de mener à bien le présent projet.

Nos vifs remerciements vont aussi aux membres de jury pour avoir accepté de juger ce travail.

A la même occasion, nous voudrons également remercier chaleureusement nos parents qui nous ont toujours encouragés durant notre cursus de formation.

Enfin, nos vifs remerciements sont adressés à toutes ces personnes qui nous ont apporté leur aide précieuse et leur soutien inconditionnel. ■

TABLE DES MATIÈRES

RÉSUMÉ.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

INTRODUCTION GÉNÉRALE.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

RÉFÉRENCES BIBLIOGRAPHIQUES.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

Chapitre I.

APERÇU SUR DEEP LEARNING

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do.

Chapitre II.

DETECTING LUNG CANCER NODULES

II.1. introduction

The project is to create a detector for lung cancer, and based on the **LUNA dataset** luna16.grand-challenge.org that is a collection of CT scans of patients with lung nodules, which are small growths in the lungs that may indicate cancer. The dataset is part of a Grand Challenge, which is a competition among researchers to develop and test methods for detecting and classifying nodules. The dataset is open and publicly available,

LUNA16 It contains 1,186 lung nodules annotated in 888 CT scans. The LUNA dataset has two tracks: one for finding the locations of nodules in the scans, and another for reducing false positives by distinguishing benign from malignant nodules.

Automating this process will provide an experience in dealing with difficult scenarios where solving problems is challenging. Automatic detection of lung cancer is challenging, and even professional specialists face difficulty in identifying malignant tumors. Automating the process with deep learning will be more demanding and require a structured approach to succeeding.

Detecting lung cancer early is essential for increasing the patient's survival rate, but it's tough to do manually, especially on a large scale. The problem space of lung tumor detection is important because it is an active research area with promising results. However, it is also unsolved, which satisfies the authors' objective of using PyTorch to tackle state-of-the-art projects.

In large-scale project, it will be working with 3D data and require data manipulation, as no pre-built library is available for suitable training samples. The project will involve using convolutional layers followed by a resolution-reducing downsampling layer. To handle the computational requirements, you will need access to a GPU with at least `8 GB` of RAM or `220 GB` of free disk space for raw training data, cached data, and trained models.

Instead of analyzing the entire CT scan, it will break down the problem into simpler tasks. CT scans are 3D X-rays consisting of a three-dimensional array of single-channel data, with each voxel having a numeric value that approximately corresponds to the average mass density of the matter contained inside.

As for choosing the batch size, it depends on your specific situation. For example, with an image size of 2400x2400x3x4, a single image takes 70 MB, so a batch size of 5 might be more realistic. However, this depends on the available GPU memory, and using 16-bit values instead of 32-bit can help double the batch size

II.1.1. Definitions.

- **CNN**: The design of a convolutional neural network for detecting tumors are based on alternative image recognition that can be used as a starting point. This Convolutional neural networks typically have a tail, backbone, and head. The tail processes the input, while the backbone contains most of the layers arranged in series of blocks. The head converts the output from the backbone to the desired output form.

- **Epoch Training**: The epoch is divided into 20193 steps called batches, each containing 256 data points. Once the data is loaded and the training process begins, monitoring the performance of the computing resources is crucial to ensure that resources are being used effectively.

- **Metrics**: displays training and validation metrics in a graphical format, making it easier to interpret the data. We can adjust the smoothing option to remove noise from trend lines if our data is noisy.

  - Recall is the ability to identify all relevant things.
  - while precision is the ability to identify only relevant things.

The logging output are include the precision by including the count of correctly identified and the total number of samples for both negative and positive samples.

- **Overfitting**: Overfitting occurs when a model learns specific properties of the training set, losing the ability to generalize, and making it less accurate in predicting samples that haven't been trained on. For instance. To avoid overfitting, we must examine the right metrics. Looking at our overall loss, everything might seem fine, but that's because our validation set is unbalanced, and the negative samples dominate, making it hard for the model to memorize individual details. To prevent overfitting, we use data augmentation, which involves modifying a dataset by applying synthetic alterations to individual samples, resulting in a new dataset with a larger number of effective samples. Five specific data augmentation techniques are discussed, including mirroring the image, shifting it by a few voxels, scaling it up or down, rotating it around the head-foot axis, and adding noise to the image.

- **Data Automating**: This technique are designed to create new training samples from the existing ones by applying simple transformations. The transformations include shifting/mirroring, scaling, rotation, and adding noise.
- Thresholding is a simple and common method of segmentation that works by selecting a pixel value (called a threshold) that separates the foreground (the region of interest) from the background (the rest of the image)[^3^]. For example, if you want to segment the bone from a CT scan, you can choose a threshold that corresponds to the intensity of bone pixels and ignore the pixels that are lower or higher than that value. However, thresholding is not always accurate or robust, especially when dealing with complex or noisy images.

II.1.2. Approach for Training our Model involves five main steps

The goal of this project is to create an end-to-end solution for detecting cancerous nodules in lung CT scans using PyTorch. The approach involves five main steps:

1. Loading the CT data and converting it into a PyTorch dataset.
2. Segmenting the image to identify potential tumors.
3. Grouping interesting voxels to form candidates.
4. Classifying the nodules using a classification model.
5. Diagnosing the patient based on the malignancy of the identified nodules, combining segmentation and classification models for a final diagnosis.

II.2. step 1: Manipulation the Data

II.2.1. Data Conversions

To process the data, it is necessary to convert raw data files into a format that is usable by PyTorch, which means converting the row data from 3D array of intensity data to `Tensors` pyTorch format. This data is around 32 million voxels, which is much larger than the nodules. To make the task more manageable, the model will focus on a relevant crop of the CT scan. There are various steps involved in processing the data, including understanding the data, mapping location information to array indexes, and converting the CT scan intensity into mass density. Identifying the key concepts of the project, such as nodules, is crucial.

II.2.2. Data loading

The first step in creating a neural network for detecting lung cancer using PyTorch is handling the dataset. The goal is to produce a training sample from raw CT scan data and a list of annotations, by following thsi topics:

- Loading and processing raw data files
- Implementing a Python class to represent the data
- Converting the data into a format usable by PyTorch
- Visualizing the training and validation data

Overall, the quality of the data used to train the model has a significant impact on the project's success.

II.2.3. Raw CT Data Files

Loading CT data and processes the information to produce a 3D array, and transforms the patient coordinate system to the index, row, and column coordinates of each voxel in the array.

Annotation data from LUNA with nodule coordinates and malignancy flags are also loaded. It contains information about all lumps that look like nodules, whether they are malignant, benign, or something else. We'll use this to build a list of candidates that can be split into training and validation datasets. This Dataset contains information about some of the candidates that have been flagged as nodules, including the diameter. This information is useful for ensuring a representative range of nodule sizes in the training and validation data.

II.2.4. Training and validation sets

Splitting a dataset into training, validation, and test sets is a crucial step in building a machine learning model. It allows for the model to be trained on one set, tuned on another, and evaluated on a final set. This helps prevent overfitting and gives an accurate measure of the model's performance.

We want to ensure that both sets represent the real-world input data that we expect to see and handle normally. If either set is significantly different from our actual use cases, it's highly likely that our model will behave differently than we expect. This split helps us evaluate and improve the model's performance before we deploy it on production data.

II.2.5. Loading individual CT scans

We need to understand how to load and understand CT scan data, which is usually stored in a DICOM file format. The MetaIO format is suggested for easier use, and the Python SimpleITK library can be used to convert it to a NumPy array. The Hounsfield Unit (HU) scale is used to measure CT scan voxel density, with air at -1000 HU, water at 0 HU, and bone at least 1000 HU.

II.2.6. Data Ranges and Model Inputs

Starting with adding channels of information to our samples. To prevent the overshadowing of the new channels by raw HU values, we must be aware that our data ranges from -1,000 to +1,000. We won't add more channels of data for the classification step, so our data handling will remain the same.

Fixed-size inputs are necessary due to a fixed number of input neurons. We want to train our model using a crop of the CT scan that accurately centers the candidate, making identification easier for the model by decreasing the variation in expected inputs.

II.2.7. The Patient Coordinate System

The candidate center data expressed in millimeters, not voxels. We need to convert our coordinates from the millimeter-based coordinate system (X, Y, Z) to the voxel-address-based coordinate system (I, R, C). The patient coordinate system defines the positive X to be patient left, positive Y to be patient behind, and the positive Z to be toward patient head. The patient coordinate system is often used to specify the locations of interesting anatomy in a way that is independent of any particular scan.

II.2.8. CT Scan Shape and Voxel Sizes

The size of the voxels varies between CT scans and typically are not cubes. The row and column dimensions usually have voxel sizes that are equal, (...) and the index dimension has a larger value, but other ratios can exist.

II.2.9. Converting Between Millimeters and Voxel Addresses

(...) Converting between patient coordinates in millimeters and (I, R, C) array coordinates, we define some utility code to assist with the conversion. Flipping the axes is encoded in a 3×3 matrix. The metadata we need to convert from patient coordinates to array coordinates is contained in the MetaIO file alongside the CT data itself.

(…) In CT scan images of patients with lung nodules, most of the data is not relevant to the nodule (up to 99.9999%). To extract the nods, an area around each candidate will be extracted, so the model can focus on one candidate at a time.

II.3. step 2: Segmentation

The process of segmentation to identify possible nodules, which is step 2 of the project's plan. The segmentation model is created using a **U-Net**. The objective is to flag voxels that might be part of a nodule and use the classification step to reduce the number of incorrectly marked voxels.

II.3.1. Semantic segmentation: Per-pixel classification

U-Net network is a popular architecture for semantic segmentation. It was originally proposed for biomedical image segmentation, but since then it has been widely used for several other domains such as self-driving cars, satellite imagery, and more. The U-Net architecture includes an encoder that down-samples the input image, which is then followed by an up-sampling decoder. This allows the network to learn high-level semantic features while preserving the spatial information, making it suitable for semantic segmentation.

Semantic segmentation identifies different objects and where they are in a given image. If there are multiple cats in an image, semantic segmentation can identify each cat's position. The existing classification models can't pinpoint where the cat is; they can only predict whether or not a cat is present in the image.

Semantic segmentation requires combining raw pixels to develop specific detectors for items like color and then building on this to create more informative feature detectors to finally identify specific things like a cat or a dog. Nonetheless, the segmentation model will not give us a single classification-like list of binary flags like classification models since the output should be a heatmap or mask.

II.3.2. Why we need heatmap or mask as output of segmentation

The output of a U-Net network in biomedical image segmentation is typically a heatmap or a mask because these formats provide a clear visual representation of the boundaries that the network has identified in the image. A heatmap is a colored image that highlights the regions of the input image that are most important for the output classes, whereas a mask is a binary image that indicates which pixels belong to which class.

In biomedical image segmentation, it is important to accurately identify the areas of interest, such as tumors or blood vessels, to aid in diagnosis and treatment. The heatmap or mask allows for easy visualization of these areas and can be used by medical professionals to make more informed decisions. Moreover, the heatmap or mask output can be used as input for further processing and analysis, such as quantifying the size or volume of a segmented region.

II.3.3. UNet Architecture for Image Segmentation

U-Net is a convolutional neural network that can produce pixelwise output for image segmentation. It has a U-shaped encoder-decoder structure that operates at different resolutions. The encoder network reduces the spatial dimensions and increases the number of filters at each block, while the decoder network does the opposite. The key innovation of U-Net is the use of skip connections that link the encoder and decoder blocks at the same level, allowing the network to capture multi-scale features and produce more precise segmentations.

We use the same source data as before: CT scans and annotation data. Our goal is to create bounding boxes that cover the whole nodules based on their annotated centers. This will help us with segmentation, which is the process of identifying regions of interest in images. To do this, we search for voxels with high density around the center of each nodule on the row and column axis. We stop when we reach voxels with low density, which indicate normal lung tissue. Then we repeat the search in the third dimension.

We have seven input channels for UNet: three context slices before and after the focus slice, and one focus slice that we segment. We have one output class indicating whether a voxel is part of a nodule.

II.3.4. Visualizing CT Image with Predicted Nodules

The U-Net model has an encoder that captures the context of the image and a decoder that produces the segmentation map. To evaluate the performance of the model, we create an empty image with 512x512 pixels and three color channels.

We overlay the predicted segmentation map on the original CT image and use different colors to indicate the errors. We use red for false positives (pixels that are predicted as nodules but are not), green for true positives (pixels that are predicted as nodules and are), and orange for false negatives (pixels that are not predicted as nodules but are).

The pixel values are normalized between 0 and 1. The goal is to have a clear visualization of the nodules and the errors in the prediction.

II.4. step 3: Grouping nodules.

We use segmentation to find ROIs that might be nodules in CT images. Then we group pixels that are next to each other and above the limit. Each group is a nodule candidate with a center point (index, row, column). We use these points to classify the candidates. Grouping makes the search easier and removes noise.

II.5. step 4: Classification

This step involves dividing the CT scan into individual slices. The output of the segmentation step is an array of per-pixel probabilities, indicating whether the pixel is part of a nodule. These slice-wise predictions are collected in a mask array with the same shape as the CT scan input, and a threshold is applied to the predictions to obtain a binary array. A cleanup step which shortens the flagged area and removes small components.

II.6. step 5: Diagnosing the patient

We use the LIDC annotations to decide if a nodule (a small lump) in the lung is cancerous or not. The LIDC annotations are labels that up to four doctors gave to each nodule based on how it looks in a CT scan. They used a scale from 1 to 5, where 1 means the nodule is very unlikely to be cancerous and 5 means it is very likely to be cancerous. These labels are not based on other information about the patient, such as their medical history or symptoms. To make a final decision, we will use a rule that says a nodule is cancerous if at least two doctors gave it a 4 or a 5. This rule is not very precise and there are other ways of using the labels, such as taking the average or ignoring some nodules.

II.7. Conclusion

The identifying nodule candidates in CT scans for possible cancer detection. A connected-components algorithm is used for grouping the suspected nodule voxels. The labeled chunks are passed on to a classification module to reduce false positives. Finally, the identified regions in the CT scan are cropped and passed onto the classification module using DataLoader.

We use a data loader, to loop over a candidate list to threshold. The output probabilities to get a list of things our model thinks are actual nodules, which would be output for a radiologist to inspect while adjusting the threshold to err a bit on the safe side. A single CT scan from the validation set is run, and 16 nodule candidates are found.

The task of identifying malignant nodules from benign ones in CT scans after implementing the nodule-detection task of the LUNA challenge. Even with a good system, diagnosing malignancy would need a more comprehensive view of the patient, additional non-CT context, and a biopsy instead of just looking at a particular nodule in isolation on a CT scan. This task is likely to be performed by a doctor for some time to come.

- Splitting training and validation (and test) sets between patients is important to avoid errors.
- Converting pixel-wise marks to nodules can be achieved using traditional image processing.
- The diagnosis performs both segmentation and classification.
- TensorBoard can help us visualize and identify network anomalies.
- There is no magic bullet when training neural networks.

This system is not ready to replace a human radiologist, but it could be a useful tool to help them find suspicious areas in the scans. And would need more data and validation from experts, as well as regulatory approval from authorities. The system would also need to run in a scalable environment that can handle different cases and situations.

RÉFÉRENCES BIBLIOGRAPHIQUES.

Chapitre I.

RÉSULTATS ET DISSCUSSION

CONCLUSION