MSc Project Report

Deep learning techniques applied to hybrid PET/MR medical images

Irene Grone
MSc Data Science

Department of Computer Science and Information Systems
Birkbeck College, University of London

September 2020

Academic declaration

This project proposal is substantially the result of my own work, expressed in my own words, except where explicitly indicated in the text. I give my permission for it to be submitted to the JISC Plagiarism Detection Service.

Table of Contents

List of abbreviations

Abbreviation	Explanation
CT	computed tomography
CNN	convolutional neural network
DICOM	digital imaging and communications in medicine
FDG-PET	fluorodeoxyglucose - positron emission tomography
LOO	leave-one-out
LOR	line of response
MR	magnetic resonance
PET	positron emission tomography
PET/CT	imaging technology that combine positron emission tomography and computed tomography
PET/MR	imaging technology that combine positron emission tomography and magnetic resonance imaging
ReLU	rectified linear unit
ROI	region of interest
RF	radio frequency
RTSTRUCT	radiotherapy structure set
SGD	stochastic gradient descent
STS	soft-tissue sarcoma
SVM	support vector machine
TCIA	The Cancer Image Archive
TOF	time of flight
Voxel	volume element

1. Abstract

This project investigates the applications of deep learning methods on medical images analysis and classification, in particular on hybrid PET/MR images. 2D convolutional neural networks (2D CNN) can be used as algorithm for general feature extraction and for classification of patients with an identified histological condition that are at risk of developing a lung tumour.
Two CNN architectures are proposed and tested on the Soft-Tissue-Sarcoma dataset: data augmentation is performed by extracting patches of size 48 x 48, Leave-One-Out loop are implemented to maintained independence for training and testing.
Previous research in the field shows that a model that is successful for a type of images and medical condition cannot be applied successfully to a different problem. The results obtained show evidence that CNN can be powerful models, but that the adaptation and customisation of these models require considerable time for parameter adjustments and fine tuning to obtain acceptable and useful results.

Supervisor
David Weston

2. Introduction

2.1 Description

This project aims to investigate the application of deep learning methods on medical image analysis and classification for prediction and prevention of tumour development. For this purpose, images acquired with different modalities are analysed: fused PET/MR images, these hybrid images could provide stronger information than single modality images. In particular the project implements convolutional neural networks used as general algorithm for feature extraction method and classification models, instead of common feature extraction techniques used in medical imaging like intensity histograms and filter-based features.

2.2 Report outline

Section 3 - Overview of medical imaging and the methodologies considered in the project: the physics principle on which the techniques are based, acquisition methods, information and characteristics of the images obtained by each technique. Introduction of deep learning and its application for medical image analysis and classification, the chapter includes the revision of similar applications in the field and how the project is related and differentiates from other works.

Section 4 – Description of the technical objectives of the project, the choice of programming language for the development of the project and the analysis or the results, details about Python packages used.

Section 5 – The chapter presents a description of the clinical data and DICOM files. Following extraction, pre-processing and preparation of the image data; the characteristics of the network architectures implemented and the reason for the choice of parameters and hyperparameters. The experiments undertaken are detailed and include diagram and summaries of the convolutional architectures implemented.

Section 6 – This chapter presents the overall results from the experiments. The results are discussed, including: tables and graphs, detailed analysis of classification metrics, comparison of the models implemented.

Section 7 – A critical analysis of the project overall, the technical challenges encountered, the lessons learnt and suggestions about alternative methods that could be implemented are presented. The section concludes with ideas on future developments and further applications of the techniques investigated in the project.

3. Background research

This section describes the physics principles behind the medical imaging technologies, specifically for the image modalities used in the project, their main limitations and challenges. Following a description of digital imaging and communications in medicine (DICOM) file format, then the main stages of image analysis are briefly described. Following a general overview of deep learning, with details regarding convolutional neural network techniques. The section concludes with a description of specific application of deep learning to medical imaging and the differences and similarities with this project.

3.1 Medical Imaging

Medical images have become more and more important in modern medicine thanks to the advancements in technology and their application in Medicine. Medical images are used for the early diagnosis of diseases, for planning treatments and post-treatment follow-up actions. The variety of methods are based on different physic principles: transmission, radiation, reflection/refraction of light, sound or spin (Deserno, 2011). Each imaging technique – radiography, CT, SPECT, MR, PET and others – provides complementary information to understand better the structure and function of the tissue, bones and organs (Dougherty, 2009). The imaging technique that follows are all based on the basic principles of atomic and nuclear physics.

3.1.1 Positron Emission Tomography (PET)

PET is an imaging system that detects two annihilation photons, or gamma rays, originating from the radiotracer compounds administered to the patient. The detection of specific radiotracers allows the retrieval of functional information: biologic activity, blood flow, chemical composition, absorption that allows to identify cancer, fracture in bones, anomalies. For example, the fluorodeoxyglucose (FDG) is used in PET scan to detect glucose metabolism. Positron-emitting radionuclides possess the property that makes PET a unique high-resolution molecular imaging device. When the emitted positron collides with a nearby electron, they annihilate and produce two annihilation photons of 511 keV. The annihilation photons, identical to gamma photons with 511 keV of energy, travel in nearly exact opposite directions of each other. This near collinearity allows to identify the location of the positron emitters through the detection of the two photons by detectors positioned exactly on opposite sides of the event, the photons hit the detector almost at the same time, the detectors convert the high-energy photons into electrical signals that are then electronically processed. The scanners identify the line of response (LOR) and record the time-of-flight (TOF), this information is used for coincident time analysis, to determine if the signals detected are from photons of a true annihilation event or due to random coincidence, attenuation or scattering. Coincidence events are saved in a data set format called sinogram: a line integral projection data obtained from a large number of detectors at different views surrounding the entire object. Analytical methods calculate the radionuclide tracer distribution directly from the measured sinogram data (Cho et al., 2011).

3.1.2 Magnetic Resonance (MR)

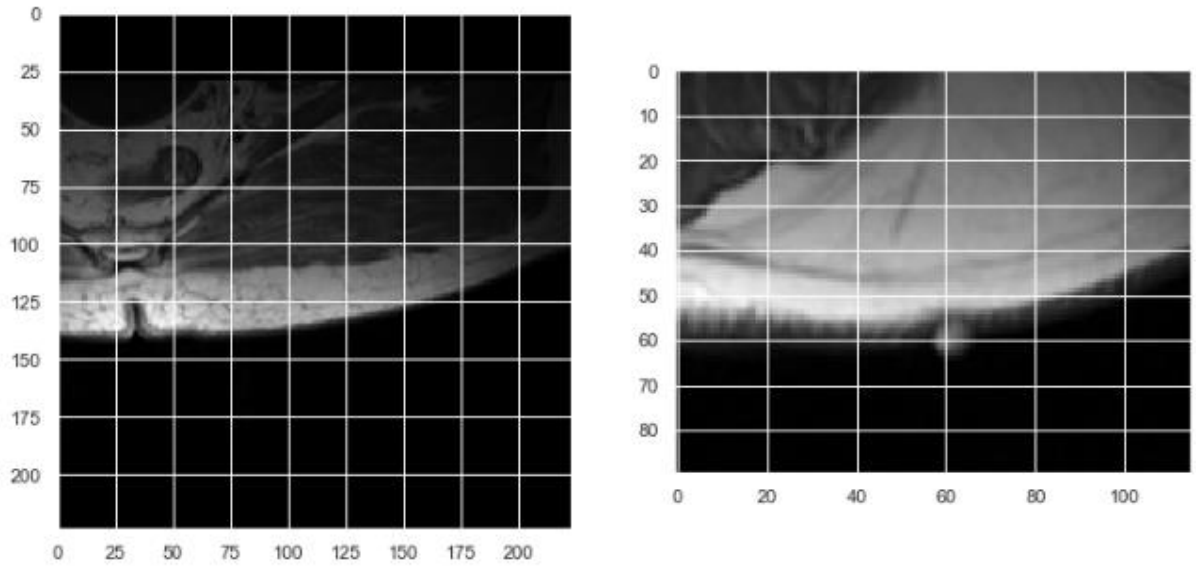
This methodology is based on electromagnetic effects of the nucleus. Considering the nucleus of hydrogen composed of one proton (charged particle) continuously spinning, it has a magnetic moment. When an external magnetic field is applied the atomic magnetic moment is aligned to the external magnetic field and precession start (Deserno, 2011). The main magnets in the MR system generates a strong magnetic field; when the object under investigation is inserted in the main magnetic field, the spins in the object will align either parallel or anti-parallel to the main magnetic field. A slightly larger fraction of these protons will be oriented in the anti-parallel form, leading to a net magnetization. In a given magnetic field, all the spins precess with a specific frequency known as the Larmor frequency, specific to the strength of the magnetic field. When an external oscillating magnetic field at the Larmor frequency is applied to the spins, the spins absorb the applied energy and are excited to a high energy status due to the magnetic resonance absorption phenomena. They remain in the exited status for a time that depends on the relaxation properties of the object. The spins than return to at equilibrium state and will give off the energy in form of electromagnetic signal. The energy is delivered and recorded by radio frequency (RF) system composed by: the transmitter coils which transmit the energy within the bandwidth including the Larmor frequency, and the receiver coils. Each signal received by the RF coils is not encoded to produce the image, the gradient system has the functions of slice selection and special encoding necessary to spatially localize the magnetic resonance signal. The signal is then recorded and analysed in the computer system, then using a mathematical algorithm, as Fourier transform, the image is reconstructed. Many physical parameters are involved in MR, and different properties are taken in consideration: T1 spin–lattice relaxation time, determines the parallel vs. antiparallel alignment of spins when interacting with the surrounding tissue; T2 spin–spin relaxation time, is the dephasing time of the in-phased spins due to spin-to-spin interaction. These properties are weighted in the image by adjusting the pulse sequence and related imaging parameters. MR offers excellent spatial resolution, often more than an order of magnitude better than PET (Cho et al., 2011).

3.1.3 Multimodality images

The functional or molecular images obtained with PET have lower resolution than anatomical images provided by CT and MR. The CT scan provides the anatomical context for display and interpretation of the PET scans. However, soft tissue contrast in CT is poor, and CT images are of limited value for delineating organ and other tissue boundaries in a manner that they can be used to guide PET image formation. MR scanners provide high-resolution anatomical information with excellent soft tissue contrast. (Bai et al., 2013).
MR imaging also can be used to guide PET image reconstruction, partial volume correction, and motion compensation for more accurate disease quantification and can improve anatomic localization of sites of radiotracer uptake, improve diagnostic performance, and provide for regional and global structural, functional, and molecular assessment of various clinical disorders (Torigian et al., 2013). The co-registration performed by separate MR and PET scanners is difficult due to differences in the posture from scan to scan and to the internal motion and non-rigid deformation of organs that occurs in time (Bai et al., 2013). PET/MR combined imaging can be obtained by different approaches: using software-based registration on independently acquired PET and MR image data sets; by using hybrid PET/MR

scanners that sequentially or concurrently acquire the images. The technique has the potential to create robust combined structural, functional, and molecular images to assess a variety of medical conditions (Torigian et al., 2013). Figure 1 show examples of the images in the dataset used for the project: on the left MR T1-weighted and on the right the fused T1 – PET.

Figure 1: the images are example from STS dataset: left MR T1-weighted (224x224) and right fused T1 – PET (90x115). These are the first images in the series for Patient ID: STS_002, they represent the left buttock. Source (TCIA – collection: Soft-Tissue-Sarcoma)



3.1.4 Image analysis

Image processing involve several stages of image analysis used for quantitative measurements and for an abstract interpretation of the medical images. The process starts with the first stage of high-level image analysis is feature extraction, it is followed by the segmentation stage and the results are used in the classification stage. The first two stages are applied to the different abstraction levels of the image. Before proceeding with the analysis, the images usually need pre-processing. Enhancement algorithms are used to reduce image noise and increase the contrast of structures, to amplify the distinction between normal and abnormal tissue. These techniques improve image quality and form a pre-processing step before the automated image analysis (Bankman, 2000).

The objective of feature extraction is the extraction of information from the abstraction levels of the image: data level features depend on the information of all pixel together, pixel level features take into account the values of each individual pixel, edge level features are defined by the great difference of values in adjacent pixels, texture level features try to quantify the uniformity in an heterogeneous structure, region level features are usually calculated after segmentation phase and are used for object identification and classification (Deserno, 2011). When extracting features from images it is important to reduce the dimensionality measuring only essential properties of the objects. Features are higher-level representations of structure and shape, and should be chosen to preserve only the information that is important to the particular application; as an example, features

describing the contents of the objects and features describing the shape of the objects (Dougherty, 2009).

Segmentation is the process of dividing the image into connected regions which are meaningful for a task, most of the times to distinguish objects or regions of interest (ROIs). Segmentation in medical images as CT or MR is applied to detect organs and to define the contours of anatomical structures; to discriminate healthy tissue from pathological tissue. The most basic attribute used in defining the regions is grey level or brightness, properties as colour and texture can be used as well. Segmentation is, most part of the times, the first stage in pattern recognition systems; when the objects or the ROIs are isolated from the rest of the image it is possible to make characteristics measurement that can be used for the classification of the objects (Dougherty, 2009). Segmentation methods are classified in relation to both features and techniques used. Non-contextual techniques ignore the relationships that exist between features in an image: pixels are only grouped together based on a global attribute. For example, the intensity-based thresholding, a pixel is assigned to a specific region based on its grey level value. Contextual techniques exploit the relationships between image features: they group together pixels that have similar grey levels and are close to one another or have similar gradient values. For example, in a region-based technique like region growing, the connected regions are found based on some similarity of the pixels within them; or a boundary-based technique like edge-based methods are used to define the boundaries between regions (Dougherty, 2009).

Classification it is often the last stage of the image analysis process, it involves assigning objects to separate classes. After the image is segmented to isolate different objects from each other and from the background, the objects are labelled. Feature extraction step reduces the data by measuring certain properties of the labelled objects. The values of these properties are then passed to a classifier that evaluates them and makes a decision about the class each object should be assigned. To design a classifier, it is necessary to have a training set of images, the training set should contain a representative sample of the population of objects to classify (objects from each of the classes). Training the classifier is the process of using the data to determine the best set of features such that the in-class variabilities are less than the between-class variabilities (Dougherty, 2009).

3.1.5 Medical imaging data format

Digital imaging and communications in medicine (DICOM) is one of the international standard data formats for medical images and related information. DICOM groups information in a dictionary-like dataset; a DICOM data object consists of a number of attributes, or data elements, referring to: meta data, physician and study information, patient information, image information, etc., and also one special attribute containing the image pixel data. The DICOM attribute is composed of:

- tag: two hexadecimal number, in the format (gggg, xxxx) that indicates (group, element), each tag has a correspondent keyword
- VR: Value Representation, two letter string describing the format of the value
- VM: Value Multiplicity
- Value: actual value of the element

A single DICOM object can have only one attribute containing pixel data, this corresponds to a single image. For modalities such as CT, MR, PET the attribute may contain multiple "frames", allowing the multi-frame data (DICOM Standard, part 5). In Figure 2 we see the part of attributes of a DICOM object that refer to image information, from the left: tag, keyword, VR, value.

Figure 2. DICOM object extract: attributes referring to image information for the MR T1 weighted of Patient ID = 'STS_001', file '1-01.dcm'

```
(0028, 0002) Samples per Pixel                  US: 1
(0028, 0004) Photometric Interpretation         CS: 'MONOCHROME2'
(0028, 0010) Rows                               US: 512
(0028, 0011) Columns                            US: 512
(0028, 0030) Pixel Spacing                      DS: [3.90625e-1, 3.90625e-1]
(0028, 0100) Bits Allocated                     US: 16
(0028, 0101) Bits Stored                        US: 12
(0028, 0102) High Bit                           US: 11
(0028, 0103) Pixel Representation               US: 0
(0028, 0303) Longitudinal Temporal Information M CS: 'MODIFIED'
(0028, 1050) Window Center                      DS: "661.0"
(0028, 1051) Window Width                       DS: "1298.0"
(3253, 0010) Private Creator                    LO: 'Varian Medical Systems VISION 3253'
(7fe0, 0010) Pixel Data                         OW: Array of 524288 elements
```

3.2 Deep Learning and CNN for image analysis

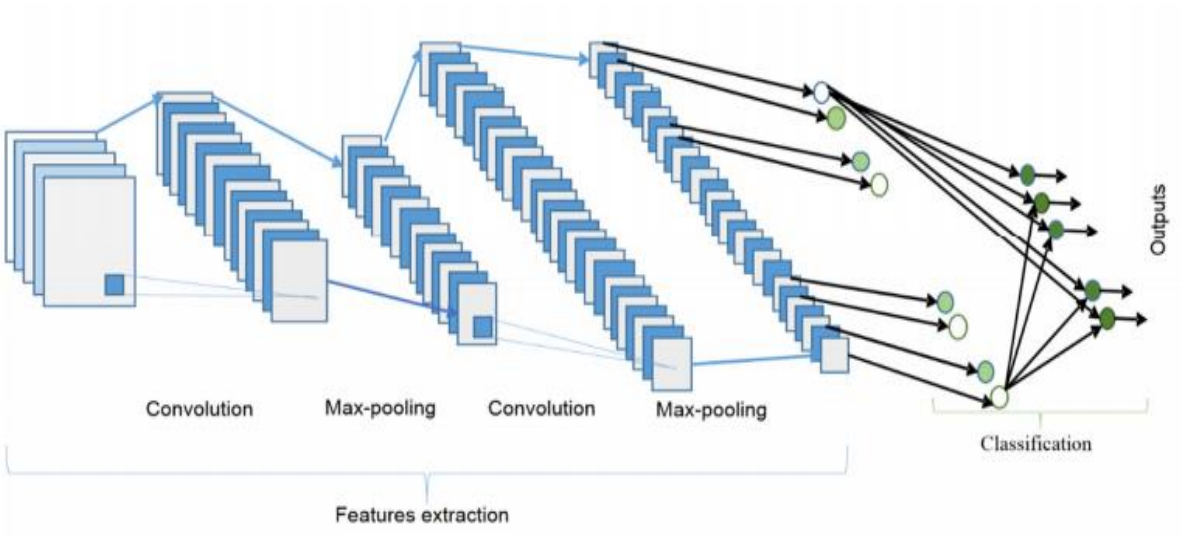
During the last few years there have been significant advances in the field of deep learning (DL) and of neural-network based machine learning methods. These advances in deep methodologies have also entered and have become a prominent framework for medical imaging data analysis (RISP VISION, 2020). The project focus on Convolutional Neural Networks (CNNs) for image data analysis and computer vision. CNNs are able to extract informative, translation invariant features from image data or volumetric data and have shown to be useful for various medical imaging modalities.

Convolutional neural networks have their origin from neuroscientific experiments of the visual cortex of the brain of cats conducted by David Hubel and Torsten Wiesel (1959). The neurons in the visual cortex are arranged in 2D spatial map that mirrors the image in the retina, they discovered that neurons in the visual cortex respond strongly to very specific simple patterns of light, such as precisely oriented bars, but responded scarcely to other patterns (Hubel and Wiesel, 1959). Some of the neurons have a local receptive field, and they react only to stimuli located in a small region of the receptive field, others have a larger receptive field and react to complex patterns that are combination of the simple patterns (Goodfellow et al., 2017). CNNs can learn: translational invariant patterns, lower layers in the model extract local, highly generic feature maps (such as edges, contours, colours and textures), these are generic feature of any image; spatial hierarchies of patterns, higher layers in the model extract more complex and abstract visual concepts, e.g. roof of a building, door of a house (Chollet, 2018).

A convolutional neural network is characterised by the presence of convolutional layers and max-pooling layers, and in general, includes: an input layer, a series of alternating convolutional layers and max-pooling layers, one or more fully-connected (dense) layers and

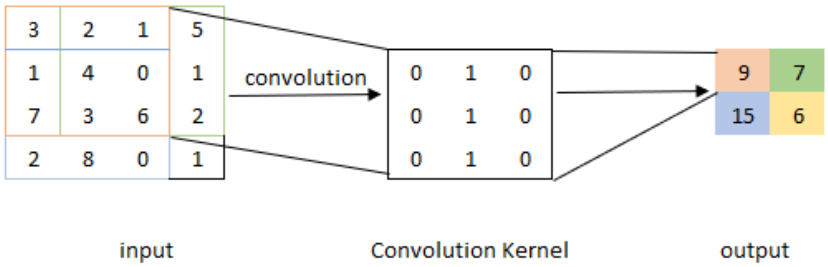
an output layer at the end. Figure 3 shows a general architecture of a 2D CNN for image classification. The network learns through backpropagation performing a gradient descent phase to adjust all the connection weights.

Figure 3. General architecture of a 2D CNN for feature extraction and classification. Source (Alom et al. 2019)



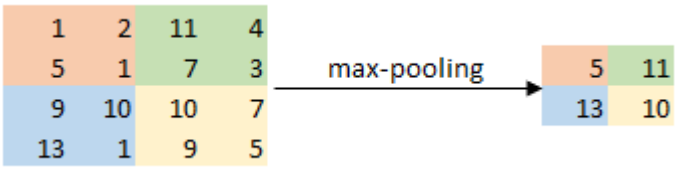
Convolutional layer: it is characterized by the application of the convolution operation. Each neuron is connected only to input pixels from the previous layer in its receptive field (kernel), the neuron's weights are represented as a small image the size of the receptive field called filter (or convolution kernel); the convolution operation extracts patches from its input feature sliding the receptive field across the input image, these 2D matrix are multiplied element-wise with the filter and summed, the output pixels produce an output feature map. Figure 4 presents a diagram of the convolution operation with kernel of size 3 x 3. All neurons in a layer use the same filter and same bias term, a layer full of neurons using the same filter outputs a feature map, which highlights the areas of the image that activate the filter the most. A convolutional layer has multiple filters and each output one feature map, so a neuron receptive field extends in 3D across all the feature maps of the previous layer. During training the convolutional layer will automatically learn the best filters for its task, and the layers above will learn to combine them in more complex patterns.

Figure 4. Convolution operation diagram



Pooling layer: its characteristic is to subsample the input image to reduce the number of parameters and the computational complexity using the pooling operation. Like in the convolutional layer, each neuron is connected to the outputs of neurons located in a small receptive field (kernel) in the previous layer. The neurons in the pooling layer have no weights, they use an aggregation function (max, mean) to aggregate the inputs: as example, a max-pooling layer will pass to the next layer the maximum value found in the receptive field and will drop the other values (Géron, 2019). Figure 4 present an example of max-pooling operation with receptive field size 2 x 2.

Figure 5. Max-pooling operation diagram



Many are the architectures developed during the years, two of the classical architecture used for image classification are LeNet-5 and AlexNet. In 1998 LeCun et al. introduced the LeNet-5 architecture applied to handwritten digit recognition. LeNet-5 comprises seven layers (excluding the input) all containing trainable parameters (weights). The input is a 32 x 32 pixel greyscale image. The pooling layers use a variant of average pooling; each neuron in the output layer outputs the square of the Euclidian distance between its input vector and its weight vector. Table 1 shows a summary of LeNet-5 architecture.

Table 1. Summary of LeNet-5 architecture

Layer type	Size	Feature maps	Receptive field (kernel)	Activation function
Fully connected	10			RBF
Fully connected	84			tanh
Convolution	1 x 1	120	5 x 5	tanh
Pooling	5 x 5	16	2 x 2	tanh
Convolution	10 x 10	16	5 x 5	tanh
Pooling	14 x 14	6	2 x 2	tanh
Convolution	28 x 28	6	5 x 5	tanh
Input	32 x 32		32 x 32	

In 2012 Krizhevsky et al. developed AlexNet, introduced for image classification on a dataset with images of 1000 different classes. It is similar to LeNet-5, but much deeper and larger, with the introduction of series of multiple convolutional layers stacked directly one on top of the other followed by a pooling layer. The first convolutional layer filters the 224×224×3 input image with 96 kernels of size 11×11×3 with a stride of 4 pixels (distance between the receptive field centres of neighbouring neurons in a kernel). The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size 5 × 5 × 48. The third, fourth, and fifth convolutional

layers are connected to one another without any pooling or normalization layers. The third convolutional layer has 384 kernels of size 3 × 3 × 256 connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size 3 × 3 × 192, and the fifth convolutional layer has 256 kernels of size 3 × 3 × 192. The fully-connected layers have 4096 neurons each (Krizhevsky et al., 2012).

With the increase of complexity and depth, the risk of overfitting increases; this is a common problem with medical imaging due to the small size of datasets available, therefore careful consideration is required when defining a CNN architecture for these datasets. Standard methods to reduce overfitting:

- reducing the size of the model, therefore trying smaller model with less layers;
- dropout layers: at every training step every neuron in the layer has a probability of being temporarily "dropped" (ignored) during the step, the neuron so trained are less sensitive to small changes in the input;
- data augmentation: generating variants of each training image by shifting, rotating, and resizing the images and add them to the training set to increase the its size.

3.3 Related work

In recent years many are the studies undertaken that have successfully applied convolutional neural network, 2D or 3D, to medical images for automated feature extraction and classification of diseases.

Vallières, et al. (2015) collected and made available the medical files and clinical data for the Soft-tissue-Sarcoma database. They developed a joint FDG-PET and MRI texture-based model for the early evaluation of lung metastasis risk in STS patients, they defined and extracted nine non-texture features and multiple variations of 41 texture features from the tumour region of single modalities and fused modalities images. They identified a multivariable model that combines four texture features extracted from the fused FDG-PET/MRI pre-treatment scans that help to predict future lung metastases development.

Li et al. (2014) present a study on how a simple customised CNN is used to automatically and efficiently learn the intrinsic image features from high-resolution computed tomography (HRCT) images patches of lung that are suitable for disease classification. They demonstrate that for patches of texture-like images of 32 x 32 pixels, a simple CNN architecture allows to obtain good performance in image classification tasks. The network architecture is composed by: first layer - convolutional layer with kernel size of 7 x 7 and 16 feature maps output; the second layer - max pooling layer with kernel size 2 x 2, followed by three dense layers with 100 – 50 – 5 neurons respectively. The network does not need to be very deep considering that the images do not present high level features; the advantages are the reduced number in parameters to be learned, avoiding overfitting, and the fast training time.

Kesim et al. (2019) in their study on small-size CNN architecture for features extraction on X-ray chest images investigate the advantages of small network compared with well-known pre-trained network. They implement three different architectures: CNN-1, input size 128 x 128 pixels, includes thirteen convolution layers and two fully connected layers of 1024 and

12 nodes (number of classes), at the output of each convolution layer, there are ReLU layers, no max-pooling process between some convolution layers; CNN-2, input size 256 x 256 pixels, has four convolution layers and two fully connected layers of 1024 and 12 nodes (number of classes) , there are ReLU and max-pooling layers; CNN-3 has the same layers as the second architecture, but uses an input size of 128 x 128 pixels and has a different number of feature maps for each layer. Their results show that small networks have better performance in classification rate, there is less generalization loss, they have reduced training time and can be used in real time application on an embedded system.

Trivizakis et al. (2019) evaluate 2D CNN and explore a new 3D CNN for tissue classification for distinguish between primary and metastatic tumours of the liver from MR imaging. Three CNN architecture are proposed and tested: Patch-wise 2D CNN, input image size of 32 x 32 pixels, to learn texture-like features and identify malignant vs normal tissue; a 2D CNN with same structure and adapted to train on the original slice images 256 x 256 pixels, to learn ; a 3D CNN, to learn complex volume features such as tumour structure and shape, it includes four 3D convolutional layer and a fully connected layer. Trivizakis et al. also applied a support vector machine (SVM) classifier to the extracted feature vectors from the CNN architectures. Their result showed an improvement in classification accuracy using features extracted from the 3D CNN in respect to classification from features extracted from the 2D CNN.

This project implements and investigates similarly, to Kesim et al. and Trivizakis et al. studies, the generalisation of feature extraction and classification using deep learning techniques, in specific 2D CNN. The aim is to investigates different CNN architectures applied to medical images on the hybrid PET/MR dataset contained in the STS database and evaluate the results on obtained on this dataset. The project differentiates from the other studies because it investigates the performance of the proposed CNN on fused PET/MR medical images, if time allows MR images will be investigated and the results obtained will be compared to prove that there is difference in classification performance in favour of the multi-modalities image that give stronger features as stated by Vallières, et al. in their study. There is a substantial difference with previous work with CNN because the dataset used includes images that represent different parts of the body instead of the same area, this could allow to find a more generalised model but can also add difficulty due to the small number of images available for each areas of the body.

4. Project specifications

This section describes: the technical objectives of the project, the programming language chosen for development, details about Python packages used.

4.1 Objectives

The objective of the project is to develop two programs in Python: the first script covers data extraction from DICOM files for fused PET/MR images and from Excel file for clinical data, pre-processing and data augmentation of the dataset for input to the convolutional networks, implementation of Leave-One-Out loops to train and test the 2D CNN architectures models proposed; saving output from train/test loops; the second script covers analysis of the results obtained from running the models, computing classification metrics and confusion matrices, evaluate the classification.

4.2 Software

The project has been developed using Python. The choice is based on the knowledge acquired during various modules undertaken during the master course and during personal projects. Python is free and open source and it is becoming the most used programming language in industry for data science. Many well developed and integrated libraries and framework for computer vision and deep learning are available. The program runs on a machine with GPU enabled, by default TensorFlow use the GPU for CNN training.

4.2.1 Details and versions

The project is developed using Python version 3.7.8. Follow a list of specialised Python packages utilised for medical image analysis and deep learning (see Appendix B for a full list and versions).

- Keras – Keras is a deep learning API written in Python, it runs on the machine learning platform TensorFlow. Developed with a focus on enabling fast experimentation; it supports CNN and in general any neural network architecture (Keras)
- TensorFlow – low-level, end-to-end, open-source machine learning software library
- NumPy – pure Python library for scientific computing
- Scikit-image - collection of algorithms for image processing
- Scikit-learn - collection of algorithms for machine learning
- pydicom – is a pure Python package for working with DICOM files, it is used in combination with NumPy to work on Pixel Data

GPU drivers installed:

- CUDA
- cuDNN

5. Data, methods and experiments

This section presents a detailed description of the clinical and image data and pre-processing steps applied, the characteristics and structures of the CNN architectures implemented using Keras library: detailed information on the methodology used for the choice of parameters and hyperparameters; the different experiments undertaken for each model.

5.1 Data

Concurrent PET/MR images from the integrated new scanners are not easily available and they could not be found in open access medical database at the moment of execution of the project. For time constraint and delays in the schedule an alternative source of hybrid images obtained by using software-based registration on independently acquired PET and MR scans. The medical images used for this project are part of the collection "Soft-tissue-Sarcoma" from The Cancer Image Archive (TCIA) (Clark et al., 2013), and the study "A radiomics model from joint FDG-PET and MRI texture features for the prediction of lung metastases in soft-tissue sarcomas of the extremities" (Vallières, et al. 2015). The Cancer Imaging Archive http://doi.org/10.7937/K9/TCIA.2015.7GO2GSKS (see Appendix A for how to download the DICOM dataset and clinical data). The study considered patients that had pre-treatment FDG-PET/CT and MRI scans and that were investigated for lung metastases development in the follow-up period (Vallières, et al. 2015).

5.1.1 Clinical Data

The clinical data file contains the "Clinical Information" sheet where we have the characteristics of the patients included in the study, Table 2 shows the data fields and their description.

Table 2. Clinical data file content

Data field	Description
Patient ID	Unique patient ID
Age	Age of the patient
Sex	Sex of the patient
Histological type	Type of identified STS
MSKCC type	MSKCC classification of identified STS
Site of primary STS	Area of the body with identified STS
Grade	Grade of identified STS
Time – diagnosis to MRI scan (days)	Days between biopsy of STS and MR scan
Time – MRI scan to PET scan (days)	Days between MR and FDG-PET/CT scan
Treatment	Type of treatments received
Outcome (recurrence, mets)	Type of tumour or recurrence developed at follow up visit
Time – diagnosis to outcome (days)	Days between diagnosis of STS and Outcome
Status (NED, AWD, D)	Status of patient at last follow up visit
Time – diagnosis to last follow-up (days)	Days between diagnosis of STS and last follow up

The "Outcome (recurrence, mets)" field indicates what type of metastases, if any, was developed by the patients in the follow-up period after treatment. The outcome shows that 19 patients developed lung metastases, 31 patients resulted negative or developed metastases in other part of the body; this is the criteria used to determine the target label for the data and is reported in the "Outcome vector" sheet. Table 3 summarises how the "Outcome (recurrence, mets)" attribute has been classified and labelled.

Table 3. Clinical data "Outcome (recurrence, mets)" – outcomes at follow up visit

Outcome (recurrence, mets)	Series count	Target Label
--	24	0
Mets – lungs	19	1
Mets – bones	2	0
Recurrence – regional	2	0
Recurrence – local	1	0
Mets – arms	1	0
Mets – spine	1	0
Mets – abdomen	1	0

Table 4 presents a summary of the different area of the body where the soft-tissue-sarcoma has been identified.

Table 4. Clinical data "Site of primary STS" – list of the diverse areas of the body with identified STS and count of the series

Site of primary STS	Series count
left thigh	17
right thigh	11
right buttock	5
right calf	3
right quadricep	2
left calf	2
right groin, left pelvis, left adductur, left buttock, left arm, right hand, parascapusular, right parascapusular, left knee, left poplietal fossa, left biceps	1 each

5.1.2 Image data

The Soft-tissue-Sarcoma collection contains images acquired with different modalities for 51 patients with an identified type of soft-tissue sarcomas (STSs), a type of tumour, of the extremities (arms, legs). For all patients are available several series of scans in different modalities all saved in DICOM standard data format. For the purpose of this project are selected the series of images in the modalities: MR T1-weighted (T1), and the fused FDG-

PET/T1 for all 51 patients (for more details see appendix A). All the FDG-PET/CT scans were acquired with the same scanner, PET scans are already corrected and reconstructed. MR scans were acquired with different medical protocols between patients and are in raw format. The contours of the tumour region were manually drawn by expert oncologist, then were propagated to FDG-PET and MR T1 scans using a commercial software; the fused images (called in the series Aligned_T1toPET_BOX) were obtained performing a 3D discrete wavelet transform (DWT) detailed in Vallières, et al. (2015) and they include only the ROI area. For this reason, the dimensions of the images are different between series. The images acquired refer to different part of the body extremities, a different number of images have been acquired for every series in the study. Table 5 summaries the image data sets composition before pre-processing and data augmentation (see Appendix D for details).

Table 5. Raw images characteristics and labels as in STS database

Imaging type	No. series	Labels positive (1) / negative (0)	Image height range	Image width range	No. images per series	Tot No. images
Aligned_T1toPET_BOX	51	19 / 32	55 - 274	54 - 248	25 - 131	3524

5.1.3 Pre-processing

The dataset is very small and the number of images and their sizes are different for each series, some images are unsuitable for use. To augment the dataset and to standardize the size for input in the model for each image in the series patches have been extracted.
Pre-processing steps applied to raw input images Aligned_T1toPET_BOX:

- Removed unsuitable images: images where the maximum pixel value in the whole image is < 10
- normalised to values in range 0-1 considering the maximum pixel value in the dataset
- extraction of patches of size 48 x 48 pixels: non-overlapping patches starting from top left of images, the patches extracted from right and bottom borders overlap with adjacent patches depending on the exact size of the image

Table 6 summarises the images available after cleaning and after patches extraction and their labels. See the patches set size for each patient ID in any of the result tables, column “test set size” in Appendix B.

Table 6. Images after pre-processing and augmentation

Imaging type	Tot No. images	Labels positive (1) / negative (0)
Cleaned Aligned_T1toPET_BOX	3362	1372 / 1990
patches of Aligned_T1toPET_BOX	49564	19891 / 29673

5.2 Methods

This is a binary classification problem: the patient will present lung metastasis or not at follow up visit. Considering the CNN architectures used in similar studies on medical images and the standard architectures for image classification from literature, this project proposes and analyses the results of customised architectures to the hybrid PET/MR images datasets. Two models are proposed, they differentiate by the number of trainable parameters: first model, named fused_model_1, is deeper and has around 4 times the number of trainable parameters with respect to the second model, named fused_model_2.
The images are not independent: each patient has a series of images; each one is a slice of a 3D volume that represent the area of the patient's body scanned for tumour. Therefore, there are 51 groups of independent images; patches of 48 x 48 pixels are extracted from each image, assigned the corresponding Patient ID and label. The dataset is then very small to use standard train and test sets split for validation of the model; to train and test each model Leave-One-Out (LOO) method is used. Each patient's group of images is considered the 1 sample out for testing, the rest as N-1 sample for training. Loops that iterate over all the patients are implemented: epochs number and batch size are set, then the loop starts extracting in turn all the images for one patient as test set and using the rest as training set for the model. At the end of each iteration the prediction on the test set are calculated: each image on its own is classified using the logistic (sigmoid) function with threshold 0.5, then the overall accuracy on the test set is calculated as number of correctly classified images and proportion of correctly classified images; the results are stored: Patient ID, true label, size of test set, accuracy as number of correctly classified images and proportion of correctly classified images. At the end of the loop these results are saved to file.

5.2.1 2D CNN architectures

The general configuration of the models:

- Network architecture: all the models proposed are sequential with different combination / sequences of convolutional and MAX-polling layers, every model ends with 1 or two fully-connected layers and the output layer
- Weights initialization: in all the model proposed the weights are randomly initialised from Normal distribution with mean 0 and standard deviation 0.05 (default)
- Bias initialization: in all models and layers biases initialized at 0
- activation function:
  - all convolutional layers are followed by a ReLU layer (see Appendix C for details)
  - output layer use logistic (sigmoid) function default threshold = 0.5 (see Appendix C for details)
- Kernel size:
  - convolution layers 3 x 3
  - Max-pooling 2 x 2
- Padding:
  - fused_model_1: same
  - fused_model_2: valid
- Loss function: binary cross-entropy
- Optimizer: RMSprop

The Normal distribution mean and standard deviation used for weights initialization and the optimiser function have been chosen for reproducibility and generally faster computation compared with other optimiser used in standard architectures (e.g. Adam). These are initial choice to test the models, they can be modified for fine tuning after evaluation of the results obtained. Figures 6 and 7 show the specific architecture of fused_model_1 and model fused_model_2 as from output of summary() function, the sequence of layers in the model starts from the top: for each layer are indicated name and type, output shape and number of parameters (the Input layer is present in the code, but the function does not return it by default). At the end it is indicated the total number of parameter and how many of them are respectively trainable and non-trainable.

Figure 6. Summary of fused_model_1

```
Model: "fused_model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv_1 (Conv2D)              (None, 48, 48, 8)         80
_________________________________________________________________
conv_2 (Conv2D)              (None, 48, 48, 8)         584
_________________________________________________________________
pooling_1 (MaxPooling2D)     (None, 24, 24, 8)         0
_________________________________________________________________
conv_3 (Conv2D)              (None, 24, 24, 16)        1168
_________________________________________________________________
conv_4 (Conv2D)              (None, 24, 24, 16)        2320
_________________________________________________________________
pooling_2 (MaxPooling2D)     (None, 12, 12, 16)        0
_________________________________________________________________
conv_5 (Conv2D)              (None, 12, 12, 32)        4640
_________________________________________________________________
conv_6 (Conv2D)              (None, 12, 12, 32)        9248
_________________________________________________________________
pooling_3 (MaxPooling2D)     (None, 6, 6, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
FC_1 (Dense)                 (None, 128)               147584
_________________________________________________________________
FC_2 (Dense)                 (None, 16)                2064
_________________________________________________________________
output (Dense)               (None, 1)                 17
=================================================================
Total params: 167,705
Trainable params: 167,705
Non-trainable params: 0
_________________________________________________________________
```

Figure 7. Summary of fused_model_2

```
Model: "fused_model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv_1 (Conv2D)              (None, 46, 46, 8)         80
_________________________________________________________________
pooling_1 (MaxPooling2D)     (None, 23, 23, 8)         0
_________________________________________________________________
conv_2 (Conv2D)              (None, 21, 21, 16)        1168
_________________________________________________________________
pooling_2 (MaxPooling2D)     (None, 10, 10, 16)        0
_________________________________________________________________
conv_3 (Conv2D)              (None, 8, 8, 32)          4640
_________________________________________________________________
pooling_3 (MaxPooling2D)     (None, 4, 4, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
FC_1 (Dense)                 (None, 64)                32832
_________________________________________________________________
output (Dense)               (None, 1)                 65
=================================================================
Total params: 38,785
Trainable params: 38,785
Non-trainable params: 0
_________________________________________________________________
```

5.3 Experiments

For an initial evaluation of the 2D CNN proposed, the models have been tested for different number of epochs, the batch size has been maintained constant at 200 considering the memory capability available, the training time for each iteration in the loops and time to complete each full loop. All the hyperparameter set at default values for reproducibility and comparison.

LOO experiment completed:

- fused_model_1, epochs = 50, batch size = 200, approximate run time 1h 30min
- fused_model_1, epochs = 100, batch size = 200, approximate run time 2h 45min
- fused_model_2, epochs = 100, batch size = 200, approximate run time 1h 05min
- fused_model_2, epochs = 150, batch size = 200, approximate run time 1h 40min

6. Analysis and evaluation of the results

This section presents the analysis of the classification performance of the models, it includes tables and visualisations for each LOO loops performed.

6.1 Criteria of evaluation

As explained in section 5.2 Methods, each image is classified with logistic function with threshold 0.5, then the overall number of correctly classified images for the patient determines the class the patient is assigned. A separate Python script has been written for the analysis and visualization of results; different metrics and thresholds are used to determine the classification performances of the models.
The aim of the classifier is to indicated patients that are at risk of developing a tumour during the follow up period; to be effective in prevention we want to identify patients "at risk" even if they have a probability of being positive as low as 0.1.
The evaluation of the model is done calculating:

- Confusion matrix, accuracy, precision, sensitivity, specificity calculated for thresholds 0.5 and 0.1;
- ROC curve calculated considering for each patients the proportion of positive images as the probability of the patient of being positive.

Confusion matrix, defined as (see Appendix C for details):

		Actual class	
		0	1
Predicted class	0	TN	FN
	1	FP	TP

Accuracy = (TP + TN) / (P + N)
proportion of correct predictions of true positive and true negative cases among all the cases
Precision = TP / (TP + FP)
proportion of correct true positive cases among all the predicted positive
Sensitivity, or True Positive Rate – TPR = TP / P
proportion of positive cases correctly identified among all positive cases
Specificity, or True Negative Rate – TNR = TN / N
proportion of negative cases correctly identified among all negative cases

ROC curve – defined in the space of Sensitivity vs (1 − Specificity), or TPR vs (1 – TNR). It shows the ability of a binary classifier to distinguish the classes as the probability threshold value is varied.

The metrics have been calculated to the same model for different number of epochs to possibly evaluate improvement in the behaviour of the models in terms of training: has the model ben trained for long enough to learn well the features that allow the classification of images in this dataset? Does the model perform better considering a certain metric or does it seem to overfit / underfit the data? What is the best model and is there a relation / difference in training time and performance between the models?

6.2 Results

Tables 7 and 8 show the values of the metrics calculated for each model by training time and threshold.

Table 7. Summary of metrics for threshold = 0.5

Model	Accuracy	Precision	Sensitivity	Specificity
fused_model_1, ep=50	0.568627	0	0	0.90625
fused_model_1, ep=100	0.607843	0	0	0.96875
fused_model_2, ep=100	0.607843	0	0	0.96875
fused_model_2, ep=150	0.54902	0	0	0.875

Table 8. Summary of metrics for threshold = 0.1

Model	Accuracy	Precision	Sensitivity	Specificity
fused_model_1, ep=50	0.450980	0.333333	0.473684	0.437500
fused_model_1, ep=100	0.549020	0.357143	0.263158	0.718750
fused_model_2, ep=100	0.549020	0.357143	0.263158	0.718750
fused_model_2, ep=150	0.568627	0.421053	0.421053	0.656250

Figure 8 shows the resulting confusion matrices obtained for the predictions for two threshold, 0.5 and 0.1, when model 1 was trained for 50 and then 100 epochs and model 2 was trained for 50 and then 150 epochs.

Figure 8. Confusion matrices for each model and thresholds

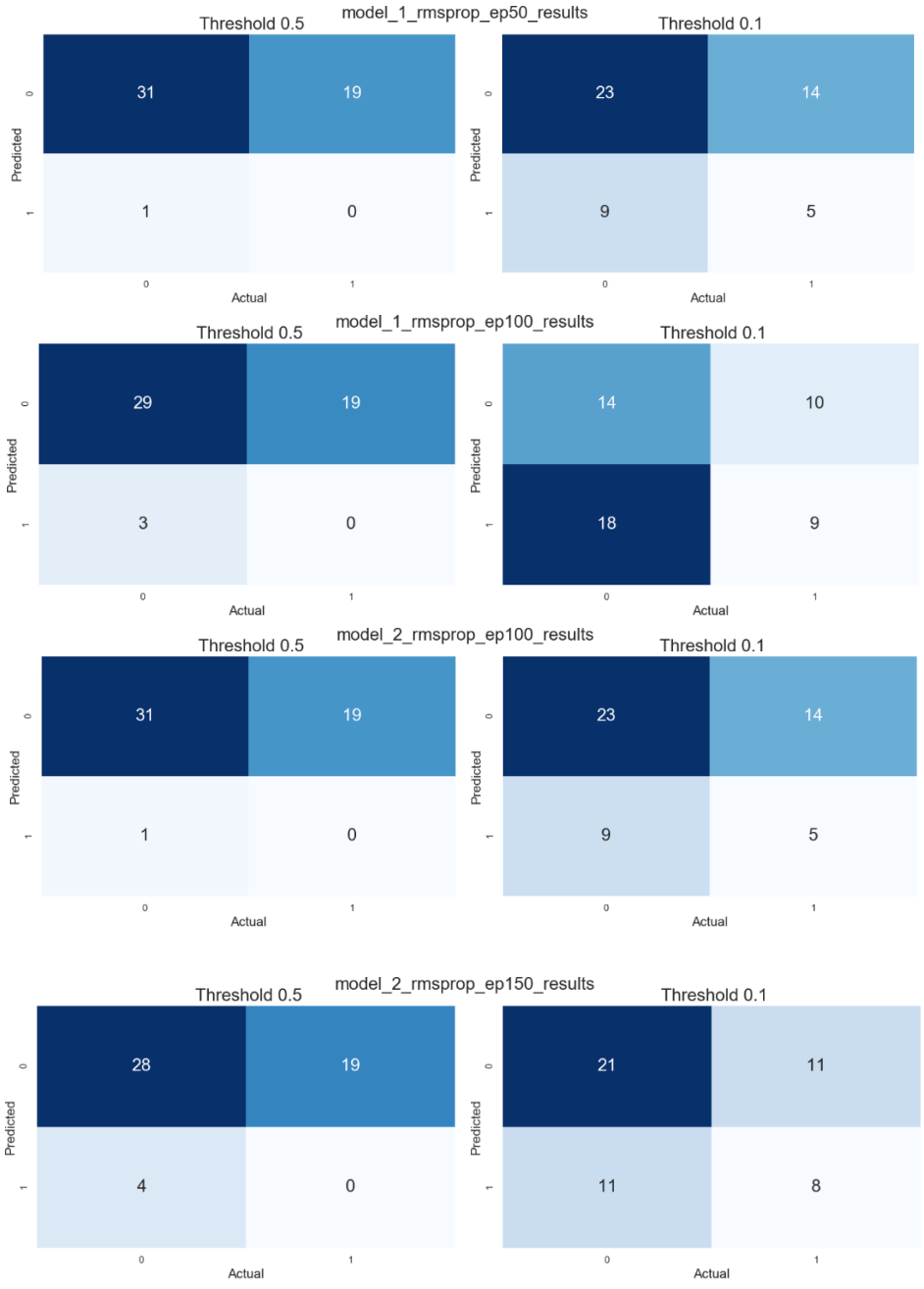
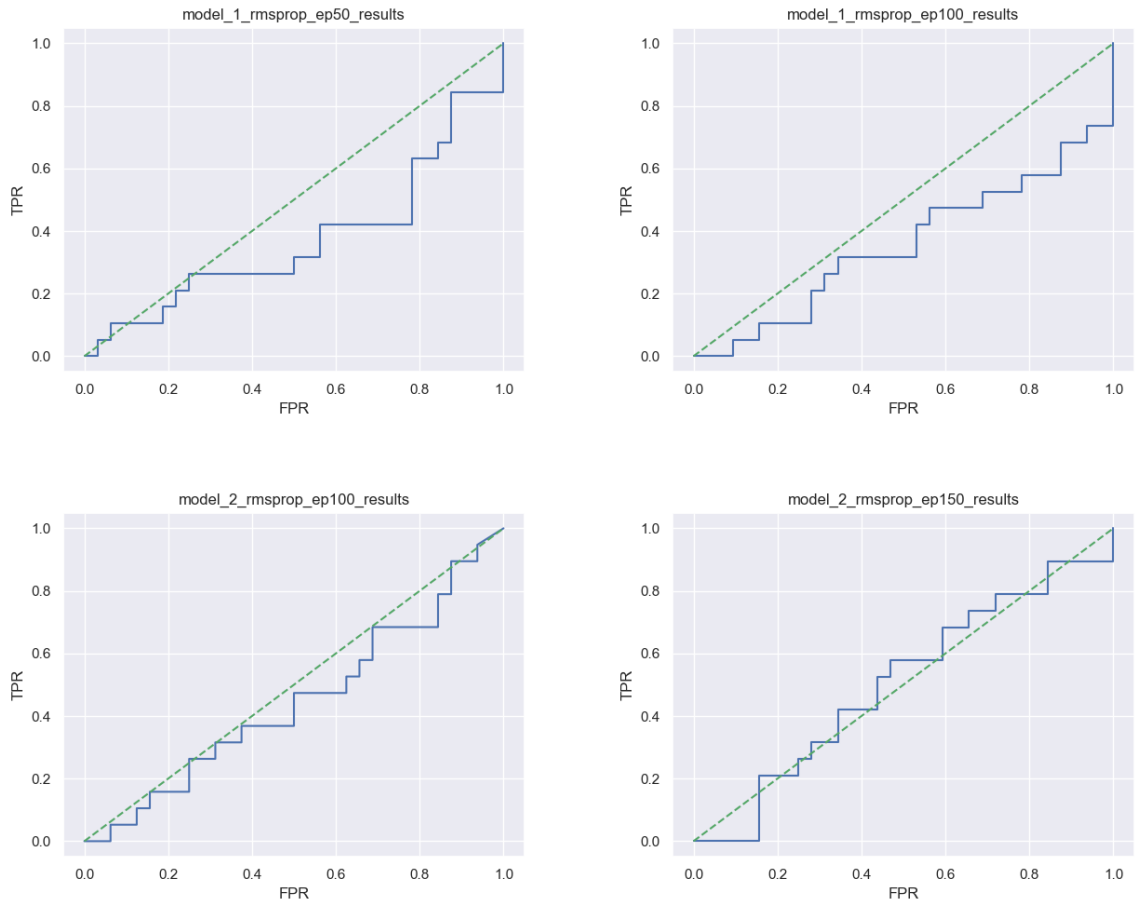


Figure 9 shows the plots of the ROC curves for the models, when model 1 was trained for 50 and then 100 epochs and model 2 was trained for 50 and then 150 epochs.

Figure 9. Plots of the ROC curves for each model: in blue line ROC curve for the classifier, in green no-discrimination line



6.3 Analysis of the results and consideration

The dataset has 32 negative cases and 19 positive cases, and the respective proportions are 0.627 and 0.373.

When considering the threshold for positiveness of 0.5: confusion matrices on the left side of figure 8 and metrics values in table 7 clearly show that both models, independently of the number of epochs they have been trained, do not discriminate between classes, they predict almost all the cases to the negative class and they do not classify any of the true positive cases.

When considering the threshold for positiveness of 0.1: confusion matrices on the right side of figure 8 and metrics values in table 8 show an improvement in classification precision and sensitivity for both models, more when trained for a higher number of epochs. At this threshold the models start to discriminate between classes: they predict both classes and correctly predict some of the true positive cases.

Sensitivity and specificity are independent test characteristics, their values do not depend on the proportion of positive and negative in the dataset; these metrics are used to define the ROC curve that considers the relation of sensitivity against (1 – specificity) for different threshold of positiveness.
Figure 9 shows the plot of the ROC curve for both models and training epochs against the no-discrimination line: the ROC curve below the no-discrimination line indicates that the model performs worse than a random classifier; the ROC curve close or on to the no-discrimination line indicates the model performs like a random classifier; the ROC curve above the no-discrimination line indicates the model can distinguish between classes. Both models show to be worse or equivalent to a random classifier; a slight improvement is noticeable for longer training time: model 2 starts to distinguish the classes.

Possible causes of poor performance of the models:

- Small size of the dataset and the fact that is very varied in the area of the body under investigation;
- Short training time: the models have not been trained long enough to learn the features of the dataset and distinguish the classes;
- The model, as they have been tested, are not yet customised for this dataset, more experiments varying layers and sequence of layers are required;
- Image patches: the patches extracted from the border of the raw images contain a majority of black area and little part of the body, they could prevent the model to learn;
- Weights initialisation and optimisation function to updates the weights: a seed was set for random initialisation of weights for reproducibility and comparison, RMSprop was chosen because fast to compute; this could be a non-optimal combination that does not allow the models to learn.

Both models seem to underfit the data, a slight improvement is shown when they are trained for longer; this suggest that the model did not learn how to classify the images and increasing the number of training epochs could improve their performance. The models are mostly equivalent, model 2 seems to perform slightly better than model 1 when trained for longer with the advantage that its training time is much shorter for the same number of epochs because it has less layers, hence less trainable parameter to update, in the architecture.

7. Summary and Conclusions

This section presents: a summary and an overall evaluation of the project with emphasis on the strong and the weak points, the difficulties encountered, what has been learnt working on the project and possible area of future developments.

7.1 Discussion

The project overall has been successful in part. The aim of investigate deep learning techniques and implement 2D CNN models as general feature extraction and classification algorithms has been achieved: the data preparation pipeline works well, the models proposed and the Leave-One-Out loops have been implemented successfully.
The classification performance of the models, as they have been trained, does not allow to clearly classify the patients and help in the early detection of all patients at risk of tumour development in the future: the models proposed perform in general as random classifier, with a slight improvement for model 2 when trained for higher number of epochs.
Considering the results obtained for classification I would have determined a different way for image patches extraction and definition, remove patches extracted from the border of the raw image that mostly represent area outside of the body, and having had the time I would have trained for longer and fine-tuned model 2.

7.1.1 Difficulties encountered

Technical limitation - Computation capability and GPU dedicated memory available for model training; time available for training the models for high number of epochs and testing of different weights initialization and optimization functions for weights update.
Software compatibilities - Considerable time was spent for installing the compatible versions of GPU drivers and TensorFlow on the machine used; some incompatibilities between libraries were encountered and drivers or libraries re-installed.
Data preparation - Standardisation of image data: due to the different sizes of the images in the raw format and the small size of the datasets; preparation of datasets for input in the model.

7.2 Conclusion and future developments

The project was very interesting, it allowed me: to learn about medical images and their different types and DICOM file format for their storage; to achieve a better understanding of 2D CNN and their architectures for image classification.
The project could be extended and developed in many ways investigating different areas of improvements:

- Increase of the number of epochs for training the models to verify when the models start to overfit the dataset considering the accuracy of overall classification of the patients; fine tune the values of the hyperparameters for the best model;
- Repeat the LOO loops 10-15 times and averaging the results, to check the stability of the output considering the random weight initialisation used;
- Investigate the MR images dataset included in the Soft-Tissue-Sarcoma collection with 2D CNN models and proceed to testing and evaluation as for the hybrid images

and validate Vallières hypothesis that the hybrid images provide stronger attribute for early detection of patients at risk;

- Change the models presented: varying the layers sequence, adding dropout layers, use different optimisers;
- The series of PET/MR images are slices of a 3D volume and the tumour is a 3D object; implementing 3D CNN models on this dataset would allow to capture volume features: extract 3D volume patches, implementing the same LOO loops and recording the same metrics as done for the 2D CNN models. Compare the result and verify if these features allow better classification and hence prediction of patients at risk;
- Ensemble methods: 2D CNN and 3D CNN models prediction, use majority vote on the classification obtained using the accuracy or averaging the probability of being positive and use a threshold of this average to determine if the patient is at risk;
- Continue the research in the field of medical images: test the models proposed on another similar dataset (same type of images, but different type of cancer) and verify if the models are better suited for classification of another type of clinical condition.

8. References

Alom, Z., Taha, T., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, S., Hasan, M., Essen, B., Awwal, A., Asari, V., 2019, "A State-of-the-Art Survey on Deep Learning Theory and Architectures", Electronics, 8, 292, doi: 10.3390/electronics8030292

Bai, B. Li, Q., Leahy, R.M., "MR guided PET image reconstruction", Elsevier Seminars in Nuclear Medicine, Vol. 43, Issue 1, pp 30-44, January 2013

Bankman, I.N., 2000, Handbook of medical imaging processing and analysis, San Diego: Academic Press

Cho, Z., Son, Y., Kim, Y., Yoo, S., 2011, "Fusion of PET and MRI for Hybrid Imaging", in Deserno, T. (ed.), Biomedical Image Processing, Berlin: Springer, pp. 55-79

Chollet, F., 2018, Deep Learning with Python, New York: Manning

Clark K, Vendt B, Smith K, Freymann J, Kirby J, Koppel P, Moore S, Phillips S, Maffitt D, Pringle M, Tarbox L, Prior F., "The Cancer Imaging Archive (TCIA): Maintaining and Operating a Public Information Repository", Journal of Digital Imaging, Volume 26, Number 6, pp 1045-1057, 2013, http://doi.org/10.7937/K9/TCIA.2015.7GO2GSKS

DICOM Standard, Official documentation, available online at: https://www.dicomstandard.org/concepts [last accessed on August 2020]

Deserno, T., 2011, "Fundamentals of Biomedical Image Processing", in Deserno, T. (ed.), Biomedical Image Processing, Berlin: Springer, pp. 1-51

Dougherty, G., 2009, Digital Image Processing for Medical Applications, Cambridge, UK: Cambridge University Press

Géron, A., 2019, Hands-on machine learning with Scikit-Learn, Keras and Tensorflow, 2nd edition, Sebastopol: O'Reilly Media

Goodfellow, I., Bengio, Y. and Courville, A., 2017, Deep learning, Cambridge, Massachusetts: The MIT Press

Hubel, D. H., Wiesel, T. N., Receptive fields of single neurones in the cat's striate cortex, The Journal of physiology, vol. 148(3), pp 574–591, 1959, https://doi.org/10.1113/jphysiol.1959.sp006308

James, G., Witten, D., Hastie, T., Tibshirani, R., 2013, An Introduction to Statistical Learning, New York: Springer

Keras, official documentation, available online at: https://keras.io, [last accessed on September 2020]

Kesim, E., Dokur, S., Olmez, T., "X-Ray Chest Image Classification by A Small-Sized Convolutional Neural Network", Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT), Istanbul, Turkey, 2019, pp. 1-5, doi: 10.1109/EBBT.2019.8742050

Krizhevsky, A., Sutskever, I., Hinton, G., "ImageNet Classification with Deep Convolutional Neural Networks", Neural Information Processing Systems, No.25, 2012, doi: 10.1145/3065386.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791

Li, Q., Cai, W., Wang, X., Zhou, Y., Feng, D.D., Chen, M., "Medical Image Classification with Convolutional Neural Network", 13th International Conference on Control Automation Robotics & Vision (ICARCV), Singapore, 2014, pp. 844-848, doi: 10.1109/ICARCV.2014.7064414.

Numpy, official documentation, available online at: https://numpy.org , [last accessed on September 2020]

RISP VISION, Deep Learning in medical imaging, available online at: https://www.rsipvision.com/deep-learning-in-medical-imaging/ [last accessed April 2020]

Pandas official documentation, available online at: https://pandas.pydata.org/docs/ , [last accessed on September 2020]

PyDICOM, official documentation, available online at: https://pydicom.github.io/pydicom/stable/index.html, [last accessed on September 2020]

scikit-image, official documentation, available online at: https://scikit-image.org/docs/stable/ , [last accessed on September 2020]

scikit-learn official documentation, available on line at: https://scikit-learn.org/stable/modules/classes.html, [last accessed on September 2020]

Tensorflow, official documentation, available online at: https://www.tensorflow.org, [last accessed on September 2020]

Torigian, D.A., Zaidi, H., Kwee, T., Saboury, B., Udupa, J., Cho, Z., Alavi, A., "PET/MR imaging: technical aspects and potential clinical applications", Radiology; Vol. 267, No. 1, pp 26-44, April 2013

Trivizakis, E., Manikis, G., Nikiforaki, K., Drevelegas, K., Constantinides, M., Drevelegas, A., Marias, K., "Extending 2-D Convolutional Neural Networks to 3-D for Advancing Deep Learning Cancer Classification with Application to MRI Liver Tumor Differentiation" in IEEE Journal of Biomedical and Health Informatics, Vol. 23, No. 3, pp. 923-930, May 2019, doi: 10.1109/JBHI.2018.2886276.

Vallières, M., Freeman, C. R., Skamene, S. R., Naqa, I. El., "A radiomics model from joint FDG-PET and MRI texture features for the prediction of lung metastases in soft-tissue sarcomas of the extremities", Phys. Med. Biol., Vol. 60, pp. 5471-5496, 2015

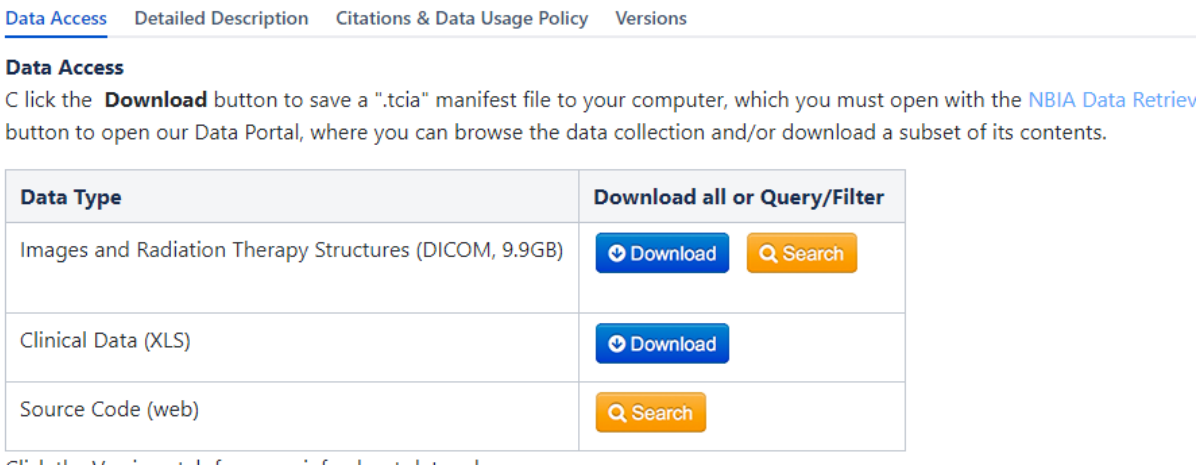
Appendix A – How to download the dataset

The clinical data, and DICOM files used in the project are part of the collections "Soft-tissue-Sarcoma" of the TCIA database.

TCIA home page: https://www.cancerimagingarchive.net/

Soft-tissue-Sarcoma home page:
https://wiki.cancerimagingarchive.net/display/Public/Soft-tissue-Sarcoma#a633484e607e4330ba7e01844a1dc27d

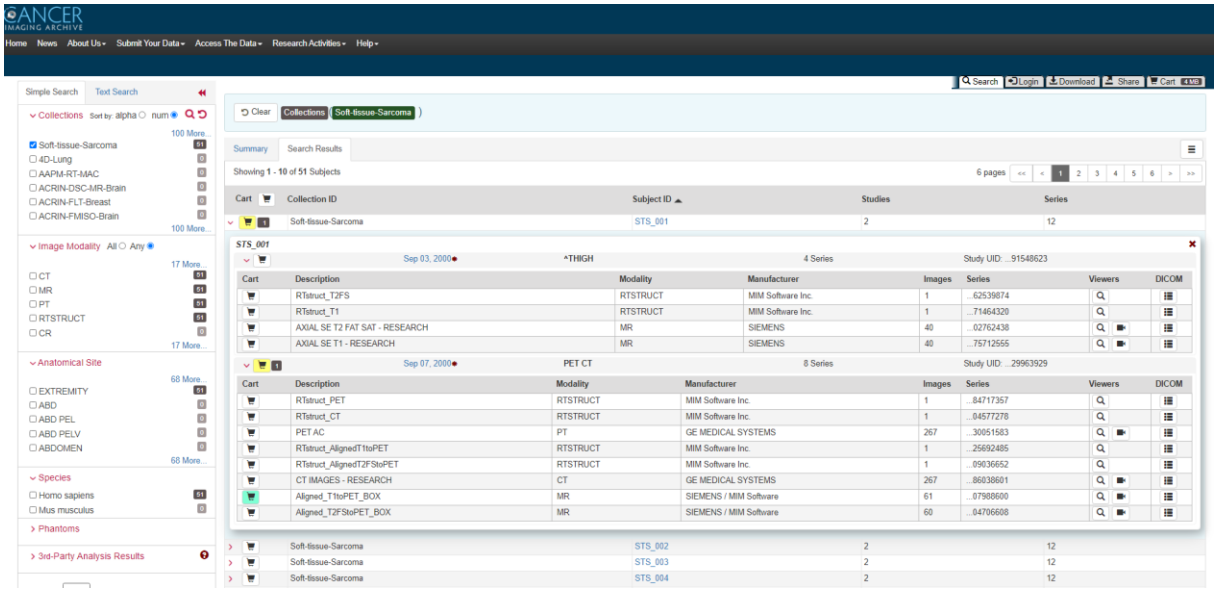
from this page, tab "Data Access" are available DICOM files for images and, Clinical Data files and link to the MATLAB the code used for the data pre-processing, definition and extraction of texture feature and multivariable modelling.



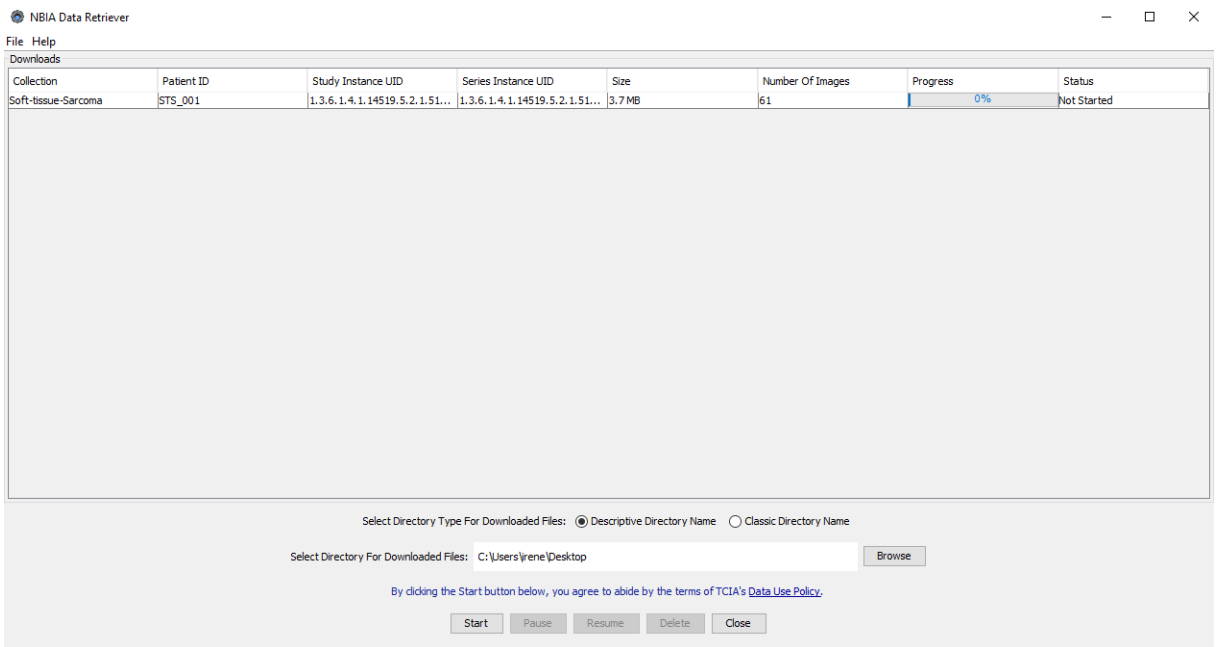
Clinical data are available directly from download button, as showed above, in Excel format as INFOclinical_STS.xlsx.

The images have been downloaded from TCIA database using the NBIA Data Retriever for Windows following the instruction at:
https://wiki.cancerimagingarchive.net/display/NBIA/Downloading+TCIA+Images

Once downloaded the Data Retriever, from the Data Access tab used for clinical data, click on Search button for Images. The system redirects you to the collection archive where it is possible to select which series to download: select tab "Search Results".
The process of selecting and downloading the right images is tedious because of the organisation of the series of images in the collection Soft-tissue-Sarcoma: the overall folders are for Patients ID (indicated as Subject ID), for each patient there are subfolders for the studies, for each study subfolders for the series of images. The series of images needs to be selected singularly for each patient selecting the cart button on the left aside the series description, as showed in the screenshot below. This process has been executed twice: one for each type of image modalities in order to have the series downloaded separately for each patient for easier organisation and data extraction from program code.



Once the selection is complete go to Cart page (from button top right of the screen): a summary of the selection made is present, it is possible to delete part of the selection made. Select download and a NBIA manifest file (.tcia) is provided, prepare the folder for image download and open the NBIA manifest file; the Data Retriever opens (see screenshot below with example of one series only), select or browse the directory for prepared for the images and start the download. From the Cart page select Export spreadsheet to download the file with details of what series are downloaded and how many images are in each in csv format (it is like a download log).



It is possible to reuse the .tcia files to download the images once the NBIA Data Retriever is installed by opening the file.

All the files related to clinical data, NBIA files .tcia and download log have been uploaded in the GitHub repository for the project. Here is the list

File	**Description**
INFOclinical_STS.xlsx	clinical data
NBIA-manifest-1597864614806.tcia	NBIA file to download Aligned T1toPET_BOX
series-data1597864634664_Aligned_T1toPET.csv	download log Aligned T1toPET_BOX
NBIA-manifest-1597861721067.tcia	NBIA file to download MR T1
series-data1597862735284_MR_T1.csv	download log MR T1

The DICOM files are saved only on personal storage and as downloaded from TCIA database in two separate folders, one for each imaging modality:

..\MSc data\Soft-tissue-Sarcoma_MR_T1
..\MSc data\Soft-tissue-Sarcoma_Aligned_T1toPET

It is not possible to upload the DICOM files to the GitHub repository maintaining the folders hierarchy and due to the total size (almost 1 GB), for this reason the .tcia files are provided so the data can be downloaded as done for this project and the code can be tested.

Appendix B – Software and hardware details

Python version 3.7.8

Python packages (excluding base packages installed by default with Python)

Name	**Version**
Keras	2.4.3
matplotlib	3.0.3
numpy	1.19.0
pandas	1.0.5
pydicom	2.0.0
seaborn	0.10.1
scikit-image	0.17.2
scikit-learn	0.23.1
tensorflow	2.2.0

GPU drivers

CUDA	v10.1
cuDNN	v7.6.5

Machine characteristics:

OS: Windows 10
CPU: AMD Ryzen 7 3700
GPU: NVIDIA RTX 2070 super 8GB

Appendix C – Mathematical details

Logistic (sigmoid) function.

The logistic function is used for binary classification, returns values in the interval (0, 1)
it estimates the probability of an instance to belong to a specific class, it usually considers as threshold for the decision p = 0.5: if the probability is greater or equal than 0.5 the model predicts that the instance belongs to the positive class otherwise belongs to the negative class.
Equation:

$$f(x) = \frac{1}{1 + e^{-x}}, \qquad range\ (0, 1)$$

Equation vectorized form:

$$p = f(\mathbf{x}^{\mathrm{T}}\ \boldsymbol{\theta} + \mathbf{b})$$

where *p* is the probability, $\mathbf{x}^{\mathrm{T}}$ is the input vector, $\boldsymbol{\theta}$ is the weight vector and $\mathbf{b}$ is the bias vector. The model predicts the output as:

$$\hat{y} = \begin{cases} 0\ if\ p < 0.5 \\ 1\ if\ p \geq 0.5 \end{cases}$$

Rectified Linear Unit - ReLU

In neural network models the rectifier function is an activation function that is defined as the positive part of its argument, a neuron that uses this function is called a rectified linear unit. It is the default activation used because fast to compute and has no maximum output.
Equation:

$$f(x) = \begin{cases} 0\ if x \leq 0 \\ x\ if\ x > 0 \end{cases} = \max\{0, x\}, \qquad range\ [0, \infty)$$

Equation vectorized form:

$$\mathbf{h} = \ ReLU(\mathbf{x}^{\mathrm{T}}\ \boldsymbol{\theta} + \mathbf{b})$$

where **h** is the output of the layer, $\mathbf{x}^{\mathrm{T}}$ is the input vector, $\boldsymbol{\theta}$ is the weight vector and $\mathbf{b}$ is the bias vector.

Confusion matrix

Defined as:

		Actual class	
		0	1
Predicted class	0	TN	FN
	1	FP	TP

P - positive: number of real positive cases in the dataset
N - negative: number of real negative cases in the dataset
TP - true positive: number of predicted positives cases that are actual positive
FN - false negative: number of predicted negatives cases that are actual positive
TN - true negative: number of predicted negatives cases that are actual negative
FP - false positive: number of predicted positives cases that are actual negative

Appendix D – Image dataset details

Aligned_T1toPET raw image dataset leaned as output from read_dicom_pixel()

Patient ID	height (pixels)	width (pixels)	No. images
STS_001	158	192	61
STS_002	90	115	35
STS_003	139	121	47
STS_004	127	128	55
STS_005	155	126	86
STS_006	117	139	86
STS_007	173	226	73
STS_008	79	77	28
STS_009	218	199	87
STS_010	136	137	88
STS_011	186	193	91
STS_012	98	109	65
STS_013	140	179	97
STS_014	148	187	105
STS_015	100	80	46
STS_016	180	165	85
STS_017	161	178	56
STS_018	94	113	42
STS_019	193	244	53
STS_020	274	248	124
STS_021	174	188	93
STS_022	192	174	63
STS_023	131	151	94
STS_024	83	78	43
STS_025	55	75	23
STS_026	135	120	43
STS_027	189	176	100
STS_028	134	123	68
STS_029	162	154	83
STS_030	263	222	121
STS_031	201	161	70
STS_032	114	85	39
STS_033	155	163	65
STS_034	165	162	52
STS_035	99	93	54
STS_036	166	187	62
STS_037	170	226	66
STS_038	118	131	55
STS_039	158	147	79
STS_040	165	150	56
STS_041	109	90	45
STS_042	97	54	42
STS_043	104	92	53
STS_044	208	180	52
STS_045	163	157	70
STS_046	172	138	99
STS_047	111	113	33
STS_048	144	152	57
STS_049	133	148	73
STS_050	135	141	49
STS_051	84	109	50

Appendix E – Result files

Results LOO on fused_model_1: epoch 50 – run time approximately 1h 30min

Patient ID	True label	test set size	correctly classified	accuracy
STS_001	0	976	909	0.931352459
STS_002	0	210	165	0.785714286
STS_003	0	423	388	0.917257683
STS_004	0	495	478	0.965656566
STS_005	0	1032	1011	0.979651163
STS_006	1	774	40	0.051679587
STS_007	0	1460	1408	0.964383562
STS_008	0	112	106	0.946428571
STS_009	1	2175	70	0.032183908
STS_010	0	792	761	0.960858586
STS_011	0	1820	1751	0.962087912
STS_012	0	585	548	0.936752137
STS_013	0	1164	1107	0.951030928
STS_014	1	1680	302	0.179761905
STS_015	0	276	182	0.65942029
STS_016	0	1360	1289	0.947794118
STS_017	1	896	0	0
STS_018	1	252	4	0.015873016
STS_019	0	1590	1515	0.952830189
STS_020	0	4464	4054	0.908154122
STS_021	1	1488	0	0
STS_022	1	1008	164	0.162698413
STS_023	1	1128	0	0
STS_024	0	172	170	0.988372093
STS_025	0	92	91	0.989130435
STS_026	0	387	377	0.974160207
STS_027	1	1600	33	0.020625
STS_028	1	612	231	0.37745098
STS_029	0	1328	1285	0.967620482
STS_030	0	3630	3276	0.902479339
STS_031	1	1400	20	0.014285714
STS_032	1	234	96	0.41025641
STS_033	1	1040	32	0.030769231
STS_034	0	832	784	0.942307692
STS_035	0	324	320	0.987654321
STS_036	0	992	969	0.976814516
STS_037	1	1320	41	0.031060606
STS_038	1	495	10	0.02020202
STS_039	1	1264	33	0.026107595
STS_040	1	896	45	0.050223214
STS_041	0	270	268	0.992592593
STS_042	0	252	151	0.599206349
STS_043	0	318	258	0.811320755
STS_044	0	1040	867	0.833653846
STS_045	0	1120	1048	0.935714286
STS_046	1	1188	241	0.202861953
STS_047	0	297	262	0.882154882
STS_048	0	684	280	0.409356725
STS_049	0	876	697	0.7956621
STS_050	1	441	25	0.056689342
STS_051	0	300	202	0.673333333

Results LOO on fused_model_1: epoch 100 – run time approximately 2h 45min

Patient ID	True label	test set size	correctly classified	accuracy
STS_001	0	976	710	0.727459016
STS_002	0	210	205	0.976190476
STS_003	0	423	340	0.803782506
STS_004	0	495	485	0.97979798
STS_005	0	1032	894	0.86627907
STS_006	1	774	34	0.043927649
STS_007	0	1460	1261	0.86369863
STS_008	0	112	95	0.848214286
STS_009	1	2175	28	0.012873563
STS_010	0	792	730	0.921717172
STS_011	0	1820	1711	0.94010989
STS_012	0	585	489	0.835897436
STS_013	0	1164	1046	0.89862543
STS_014	1	1680	319	0.189880952
STS_015	0	276	205	0.742753623
STS_016	0	1360	1169	0.859558824
STS_017	1	896	16	0.017857143
STS_018	1	252	4	0.015873016
STS_019	0	1590	1481	0.931446541
STS_020	0	4464	3421	0.766353047
STS_021	1	1488	316	0.212365591
STS_022	1	1008	166	0.16468254
STS_023	1	1128	41	0.036347518
STS_024	0	172	160	0.930232558
STS_025	0	92	70	0.760869565
STS_026	0	387	366	0.945736434
STS_027	1	1600	79	0.049375
STS_028	1	612	298	0.486928105
STS_029	0	1328	1259	0.948042169
STS_030	0	3630	3297	0.908264463
STS_031	1	1400	12	0.008571429
STS_032	1	234	49	0.209401709
STS_033	1	1040	117	0.1125
STS_034	0	832	751	0.902644231
STS_035	0	324	310	0.956790123
STS_036	0	992	896	0.903225806
STS_037	1	1320	102	0.077272727
STS_038	1	495	27	0.054545455
STS_039	1	1264	7	0.005537975
STS_040	1	896	90	0.100446429
STS_041	0	270	259	0.959259259
STS_042	0	252	120	0.476190476
STS_043	0	318	207	0.650943396
STS_044	0	1040	866	0.832692308
STS_045	0	1120	991	0.884821429
STS_046	1	1188	327	0.275252525
STS_047	0	297	281	0.946127946
STS_048	0	684	147	0.214912281
STS_049	0	876	604	0.689497717
STS_050	1	441	47	0.106575964
STS_051	0	300	144	0.48

Results LOO on fused_model_2: epoch 100 – run time approximately 1h 05min

Patient ID	True label	test set size	correctly classified	accuracy
STS_001	0	976	902	0.924180328
STS_002	0	210	210	1
STS_003	0	423	351	0.829787234
STS_004	0	495	494	0.997979798
STS_005	0	1032	1030	0.998062016
STS_006	1	774	24	0.031007752
STS_007	0	1460	1415	0.969178082
STS_008	0	112	101	0.901785714
STS_009	1	2175	2	0.00091954
STS_010	0	792	785	0.991161616
STS_011	0	1820	1783	0.97967033
STS_012	0	585	582	0.994871795
STS_013	0	1164	1164	1
STS_014	1	1680	234	0.139285714
STS_015	0	276	158	0.572463768
STS_016	0	1360	1343	0.9875
STS_017	1	896	10	0.011160714
STS_018	1	252	1	0.003968254
STS_019	0	1590	1527	0.960377358
STS_020	0	4464	4046	0.906362007
STS_021	1	1488	131	0.088037634
STS_022	1	1008	241	0.239087302
STS_023	1	1128	7	0.006205674
STS_024	0	172	166	0.965116279
STS_025	0	92	91	0.989130435
STS_026	0	387	379	0.979328165
STS_027	1	1600	4	0.0025
STS_028	1	612	255	0.416666667
STS_029	0	1328	1318	0.99246988
STS_030	0	3630	3267	0.9
STS_031	1	1400	16	0.011428571
STS_032	1	234	48	0.205128205
STS_033	1	1040	18	0.017307692
STS_034	0	832	825	0.991586538
STS_035	0	324	316	0.975308642
STS_036	0	992	915	0.922379032
STS_037	1	1320	21	0.015909091
STS_038	1	495	3	0.006060606
STS_039	1	1264	0	0
STS_040	1	896	84	0.09375
STS_041	0	270	267	0.988888889
STS_042	0	252	183	0.726190476
STS_043	0	318	250	0.786163522
STS_044	0	1040	830	0.798076923
STS_045	0	1120	1021	0.911607143
STS_046	1	1188	184	0.154882155
STS_047	0	297	292	0.983164983
STS_048	0	684	202	0.295321637
STS_049	0	876	697	0.7956621
STS_050	1	441	15	0.034013605
STS_051	0	300	209	0.696666667

Results LOO on fused_model_2: epoch 150 – run time approximately 1h 40min

Patient ID	True label	test set size	correctly classified	accuracy
STS_001	0	976	802	0.821721311
STS_002	0	210	202	0.961904762
STS_003	0	423	358	0.846335697
STS_004	0	495	477	0.963636364
STS_005	0	1032	1007	0.975775194
STS_006	1	774	50	0.064599483
STS_007	0	1460	1384	0.947945205
STS_008	0	112	101	0.901785714
STS_009	1	2175	100	0.045977011
STS_010	0	792	746	0.941919192
STS_011	0	1820	1804	0.991208791
STS_012	0	585	573	0.979487179
STS_013	0	1164	1131	0.971649485
STS_014	1	1680	215	0.12797619
STS_015	0	276	124	0.449275362
STS_016	0	1360	1278	0.939705882
STS_017	1	896	2	0.002232143
STS_018	1	252	10	0.03968254
STS_019	0	1590	1507	0.947798742
STS_020	0	4464	3888	0.870967742
STS_021	1	1488	220	0.147849462
STS_022	1	1008	349	0.346230159
STS_023	1	1128	16	0.014184397
STS_024	0	172	164	0.953488372
STS_025	0	92	84	0.913043478
STS_026	0	387	378	0.976744186
STS_027	1	1600	21	0.013125
STS_028	1	612	284	0.464052288
STS_029	0	1328	1287	0.969126506
STS_030	0	3630	3119	0.85922865
STS_031	1	1400	35	0.025
STS_032	1	234	101	0.431623932
STS_033	1	1040	62	0.059615385
STS_034	0	832	817	0.981971154
STS_035	0	324	322	0.99382716
STS_036	0	992	900	0.907258065
STS_037	1	1320	105	0.079545455
STS_038	1	495	17	0.034343434
STS_039	1	1264	0	0
STS_040	1	896	105	0.1171875
STS_041	0	270	267	0.988888889
STS_042	0	252	75	0.297619048
STS_043	0	318	164	0.51572327
STS_044	0	1040	800	0.769230769
STS_045	0	1120	1108	0.989285714
STS_046	1	1188	277	0.233164983
STS_047	0	297	294	0.98989899
STS_048	0	684	245	0.358187135
STS_049	0	876	689	0.78652968
STS_050	1	441	74	0.167800454
STS_051	0	300	95	0.316666667