UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

TESI DI LAUREA MAGISTRALE IN

INGEGNERIA DELL'AUTOMAZIONE

A Deep Learning approach for the classification of neonatal respiratory status

Relatore

Ch.ma Prof.ssa Luisa Verdoliva

Candidato

Luigi D'Amico
M58/238

Correlatore

Dott. Ing. Diego Gragnaniello

Contents

List of Figures

List of Tables

Introduction

Assessing the presence of Respiratory Distress Syndrome (RDS) in newborn babies is of major importance since the patient goes through permanent damages that could be avoided with appropriate methods if the diagnosis is timely. Although there exist few methods to assess the presence of lung diseases, the majority of them do not achieve good performance or they are too invasive for babies prematurely born.

In the last few years, Lung ultrasound (LUS) image analysis is emerging as a possible solution thanks to its high description power and harmlessness. Although there are several works on this topic that highlight the most relevant image features that could characterise the patient status, it is still a challenging problem to objectively classify an image through visual inspection. Thanks to the enormous potential of the LUS images, the research community is investing in quantitative analysis using computer vision techniques that automatically assess the patient's status. The goal is to build methods that should be free from the subjective medial opinion which may be affected by personal experience and perception.

This work proposes a Deep Learning computer vision technique to automatically assess the presence of RDS by LUS videos. Deep Learning models are the most promising techniques used nowadays in computer vision to address complex problems. These models are able to learn by themselves to detect and classify the discriminant features present in the image given a collection of data examples. The dataset used is the same considered in [7] and it is composed of LUS videos from

different hospitals and patients for which the true classification is known.

The thesis is structured as follows.

The first chapter presents lung ultrasound imaging and the major features used by physicians to make a diagnosis. In addition, the need for quantitative computer-aided analysis is emphasised. Then there are described some works present in the scientific literature concerning this topic.

In the second chapter, the main concepts about Deep Learning are described such as the training strategy and the evaluation of the model performance. Beyond this, it is presented a description of the main building blocks of Deep networks used for computer vision tasks. These are used to build some state-of-the-art networks that will be used in this work, such as EfficientNet and ResNet.

Chapter three is focused on the main objective of this thesis. Firstly, the dataset is presented in detail and there will be some discussion about the image features that the model should learn to represent. Furthermore, there is a formal definition of the problem followed by a description of Convolutional Neural Networks architectures proposed to solve it. Beyond this, there will be a discussion on the loss function, the optimiser and the metrics used in the experiments. Finally, the adopted pre-processing and data augmentation strategy is investigated.

Chapter four is devoted to the experiments. Firstly, the training and evaluation strategies are presented, a cross-validation technique is carried out to overcome the scarcity of the data. An overview of each proposed model performance is presented followed by a detailed description of the best model performance. Finally, the results of a model prediction explanation technique (Integrated Gradient) are shown, highlighting the pixels which contributed most to some prediction use cases.

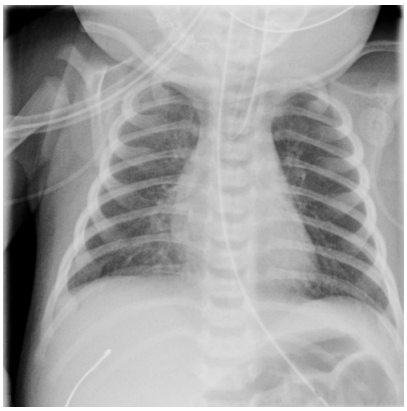
The work concludes with chapter five where conclusions are drawn as well as proposals of possible future works on this topic.

Chapter 1

Lung ultrasound imaging

Techniques that analyse lung diseases and status are changing over time thanks to the growth of technologies and the interest of the research community. One of the most used technique is the Chest X-ray (CXR), which proved to be fundamental for all patients admitted with suspected pneumonia [15]. However, it manifests to be not so efficient in case of emergency due to patient condition, waste of time and interobserver availability [1]. In fact it has been shown that promptness is an important factor for the success of the therapy [16]. Figure 1.1 shows an example of CXR for a neonatal pneumonia. Meanwhile, when the patient condition is critical, the computed tomography (CT) is recommended as it manifests to be more accurate although it is not a technique available in all emergency departments and is limited by exposure risks and cost [26] [6].

Lung ultrasound (LUS) is another approach that attracted a growing interest in the research community. Beside the fact that it can be used to describe a large variety of diseases, it is low-cost, relative safety, repeatable, harmlessness, highly descriptive and free of radiation hazards. All of these properties make this techniques very suitable in case of emergency setting, when the diagnosis has to be fast and the resources are scarce. Despite the lungs ultrasound can be very challenging because



Case courtesy of Dr Jeremy Jones, Radiopaedia.org, rID: 23898

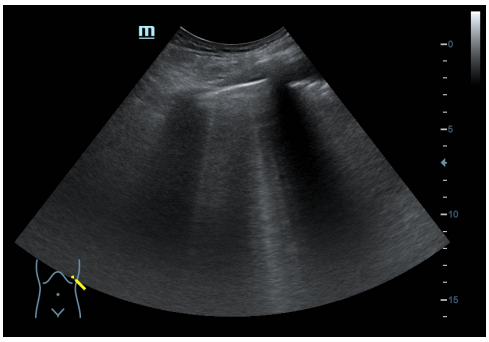
Figure 1.1: Chest X-ray for neonatal pneumonia

of the presence of air in the lungs, the discriminant features used for the pulmonary pathology detection is carried out using patterns generated by the pleural line (or pleura) [23]. The pleural line is the lung border and indicates the interface between the soft tissues and the lung tissue. In case of healthy patient, the the pleura presents a very different impedance than the alveolar air, which causes ultrasound waves to be completely reflected [19]. Thanks to the high energy retained, the waves are reflected several times between the acoustic discontinuity impedance surface and the ultrasound transducer producing a very characteristic pattern displayed on the monitor called "A lines" [13][28]. The A lines are horizontal path reverberation artifacts that occur beneath the pleural line with regular intervals and increasingly greater depths with decreasing intensity (figure 1.2a). On the other hand, in case lungs are not in normal conditions, the subpleural alveoli may contains fluid, semifluid or solid material (inflammatory exudate) and the ultrasound waves are not reflected back but they pass through the pleural line reaching the area of parenchymal consolidation. In such case a different type of artifact is generated, the "B lines". B lines artifact is a series of very closely spaced reverberations which is interpreted as a confluent vertical echo which does not fade with increasing depth [24] (figure

Case courtesy of Dr David Carroll,

Radiopaedia.org, rID: 61382

(a) A-lines artifact



Case courtesy of Dr Balint Botz,

Radiopaedia.org, rID: 65567

(b) B-lines artifact

Figure 1.2: LUS ultrasound typical images

1.2b).

The elements mentioned above constitute baseline features which should help to analyse and make a diagnosis using the LUS images. Despite all, it is still a challenging problem to objectively evaluate this kind of images and it often requires specialised equipment and trained personnel to perform and interpret them. For these reasons, an automatic system to assist the analysis of such images would be of great impact.

1.1 Quantitative LUS analysis

The quantitative LUS analysis aims to develop automatic systems to classify lung status. A computer-aided approach would help to make a more timely and accurate diagnosis especially in emergency setting, when expert physicians and advanced resources are absent.

The challenge is to quantitative detect and classify the main discriminating features present in the images. Such system can be build up using computer vision

techniques, in particular two main approaches have been experimented during the years. First, a more statistical and model-based approach, where the distinctive image features are analysed together with their statistical distribution and a specifically tailored model is developed. The second approach is completely based on Deep Learning that automatically learn the discriminating textural pattern based on given training examples.

Thanks of the high importance that such system would have, lot of papers have been published about QLUS and, in this paragraph, few of these will be cited and briefly discussed highlighting studies concerning the newborn and preterm newborns although very few research work are present nowadays about these cases probably due to the scarcity of available data.

1.1.1 Model based and statistical approach

One of the first promising QLUS examples was given by [3] where a simple but effective strategy was developed. Thirty-two patients with suspected community-acquired pneumonia were enrolled and the computed tomography (CT) results was used as a gold standard. Each hemithorax was evaluated by both chest X-ray and visual ultrasonography. The LUS images have been used to develop and test the proposed quantitative analysis showing better results prediction compared to either chest X-ray and visual ultrasonography. QLUS analysis was performed selecting a region of interest area extending from the pleural line to the bottom of the image. The echo intensity was determined for horizontal slices to compute a frequency distribution for the whole image as shown in figure 1.3. The present study showed that the mean value of the distribution allow to separate cases of different classes i.e. affected or non affected by pneumonia.

In 2018, [4] analysed a dataset composed by children under 5 years of age, 15 with a diagnosis of local pneumonia and 6 without any pulmonary comorbidity. A Deep

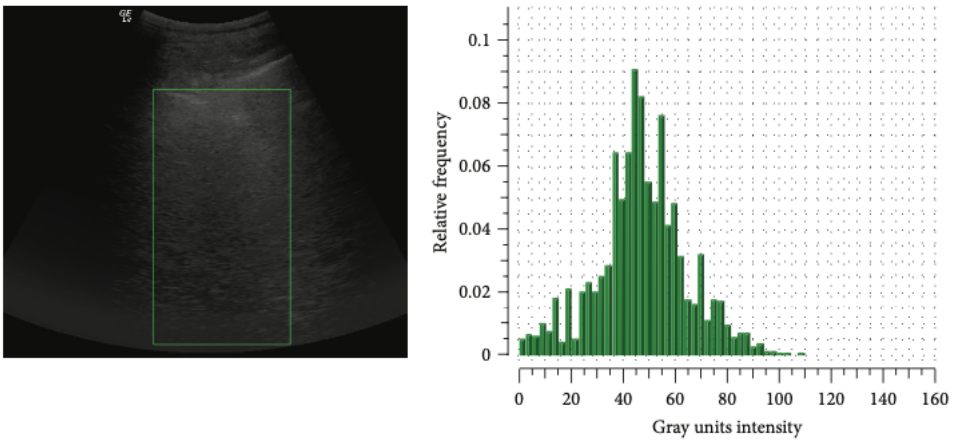


Image taken from [3]

Figure 1.3: LUS image and frequency distribution

Learning model was trained to classify the presence of pneumonia by some hand-crafted features obtained from the lung ultrasound digital images. They developed an interesting pipeline of image processing for building the so-called "characteristic vectors" used as input for the Deep Learning architecture. In particular, the pre-processing was carried out in two steps. The first step was a pleural line detection using its brightness and first-order statistics. The pixels associated to the area above the pleural line was changed to black because of the absence of information present in this area. The second steps was the features extraction, each ultrasound frames was subdivided into 400 linear rectangular sections that extend from the surface of the skin to the bottom of the frame and are called characteristic vectors as shown in figure 1.4. Three class label were attributed to the sampled vectors, namely pneumonia, healthy lung and rib bone, depending if they corresponded to a region of pneumonia infiltrate, a normal lung, or to the acoustic shadow of the rib respectively. Note that essential information is carried out in the intercostal spaces between two shadow stripes. The extracted features are represented by the average brightness

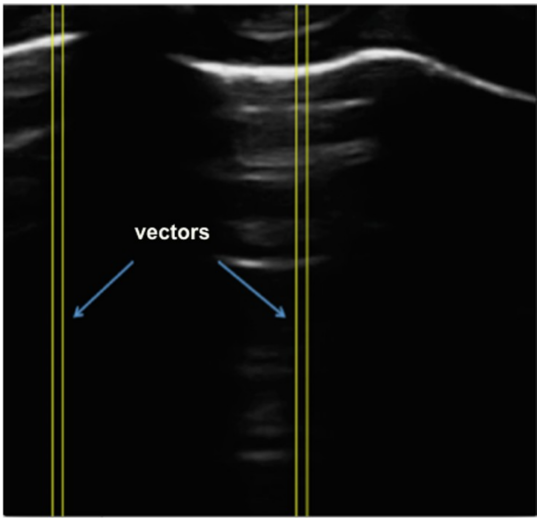


Image taken from [4]

Figure 1.4: Separation of a LUS image in characteristic vectors

profiles of the pixels contained in the sections of the vectors corresponding to the 25, 50, 75 and 100 percentile of the length. The 4 hand-crafted features vector was then used as input to train a simple neural networks composed of 3 layers in order to classify the label categories associated to the characteristic vector. In conclusion, a high sensitivity and specificity were achieved for the classification of characteristic vectors.

In [17] a dataset containing 600 ultrasound frames was built from seventy-five neonates with different respiratory status, the objective was to describe the severity neonatal respiratory distress. In this work both visual scores and computer-aided scores was given and tested with two oxygenation indexes, namely the oxygen ratio and the alveolar arterial oxygen gradient, both considered as reference standards. A significant correlation was found both from visual assessment and quantitative assessment. For what concern the visual score, two neonatologists with different degree of experience in lung ultrasound independently scored every frames of a video and a sum value from every video was used for the correlations with oxygenation indexes

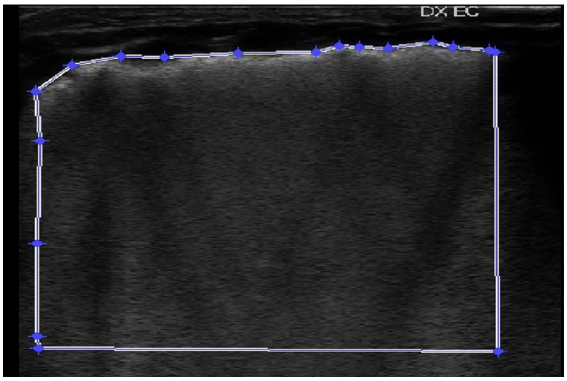


Image taken from [17]

Figure 1.5: Region of interest selection in a LUS image

associated to the patient. The same frames dataset was then scored by a grey-scale analyser with the help of a MATLAB interface built specifically for this work, which allow the user to select a region of interest (ROI) using the mouse and in particular to exclude the area above the pleural line (figure 1.5). Two groups of features was selected, a first group comprising of 10 features was simply based on the computation of global and local first-order statistics such as grey-level histogram, mean and variance; a second group comprising of more sophisticated analysis accounting for second-order statistics, some of these features were able to capture spatial dependence in the image, the idea here was to describe local micro- and macro-pattern present in the frame. The 44 output features extracted for every images was used to train a Support Vector Machine regressor and a significant correlation with the two oxygenation indexes was found. This study represents a first endeavor for the quantitative analysis of neonatal lung ultrasound.

1.1.2 Data-driven approach

There has been an intense research on QLUS using Deep Learning during the last few years especially because of the impetuous advent of COVID-19 pandemic. In

[2] notably results has been obtained, a lung ultrasound (POCUS) dataset has been gathered and a Deep Learning approach has been used in order to classify the presence of COVID-19, bacterial pneumonia and healthy patients. The effectiveness of Deep Leaning techniques for LUS quantitative analysis was confirmed by [20] that used the POCUS dataset to jointly detect and localise COVID-19.

In 2021, [14] uses a dataset consisting by infants in different status, in particular 33 healthy infants, 22 infants with bronchiolitis and 7 infants suffering from bacterial pneumonia. A fully Deep Learning approach is adopted i.e. no handcrafted features were considered, leaving the algorithm autonomously learn all the features necessary to perform the classification. They considered two different task, one binary classification (healthy vs. Bronchiolitis) and one three-class classification (healthy vs. Bronchiolitis vs. Bacterial pneumonia). Different cutting-edge Deep Learning architecture were trained, VGG19, Exception, Inception-v3 and Inception-ResNet-v2. In order to avoid overfitting, they applied data augmentation process composite by two operation, horizontal flip and width shift. Using Explainable Artificial Intelligence (XAI) [5] they also investigated what kind of features the model learnt and how the images are processed inside it. In particular, they used three XAI approaches specifically tailored for convolutional neural networks, namely visualisation of CNN convolutional filters, visualisation of activation maps produced by every layer and the visualisation of gradient-weighted class activation mapping (Grad-CAM). Visualising the filters created by the network, they learned to detect simple edged shapes while the texture patterns become more complex the deeper the layer becomes. For what concern the activations, in the first layers they essentially split the information present in the image i.e. background, contours and texture-like patterns while the activations associated to the last layers become more and more abstract. The last explanatory technique they inspected was Grad-CAM [22]. This technique is able to show where the features (pixels) that characterise a diagnosis

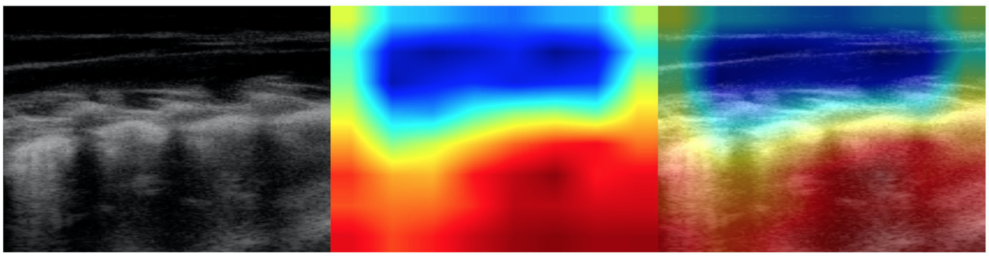


Image taken from [14]

Figure 1.6: Class activation mapping using Grad-CAM visualisation

are located in the image. Grad-CAM localised the domain-specific features that are typically taken into account by physicians when formulating a diagnosis such as larger consolations with air and/or liquid bronchograms, pleural effusions, pleural line irregularities, small and long artefacts. In figure 1.6 there is a class activation mapping produced by Grad-CAM for a correctly predicted bronchiolitis, the model highlights diagnostic-relevant features as irregular short vertical artifacts, irregular pleural line and white-lung. This work testifies the power of Convolutional Neural Network in this domain to learn complex discriminatory features essential to make predictions from the input images only.

The same dataset of [17] was extended in [7] with videos acquired by three centers with three class labels representing the patient status namely healthy, affected by Respiratory Distress Syndrome (RDS) and effected by Transient Tachypnea (TTN). Differently from the previous work, each video was associated with a non-invasive reference standard i.e. the pulse oximetric saturation ratio (SF) that represents a sort of healthy status index. A large dataset has made possible to follow an analysis based on Convolutional Neural Networks (CNNs). In particular, two major tasks has been considered: a regression task where the target variable was the SF value and a binary classification task to predict the health status of a patient i.e. healthy or sick. They firstly investigated the use of pre-processing and data augmentation

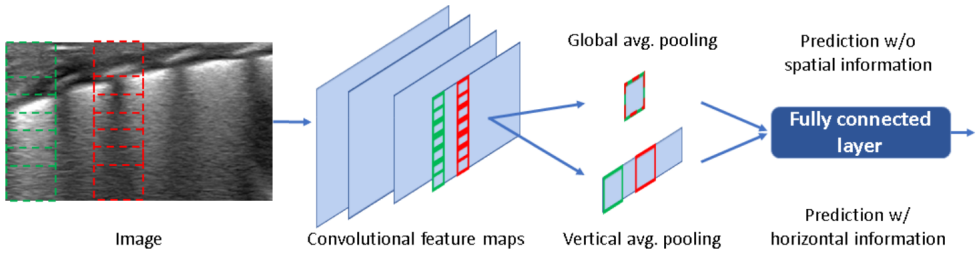


Image taken from [7]

Figure 1.7: Difference between global avg. pooling vs. proposed vertical avg. pooling

techniques for the video frames proposing a cropping strategy leaving out the bottom side of the image, where no useful information should exist. Moreover, a random horizontal flip, a random rotation and a random modification of the brightness and the contrast was considered to make the whole procedure more robust to different incidence angles of the probe and different instruments and calibrations.

Few CNNs were considered, namely AlexNet, ResNet, EfficientNet-B0, EfficientNet-B1 and EfficientNet-B2. Both for the regression and classification task, the frame-level results were aggregate to obtain a video- and session-level prediction score. In this work two advanced training strategies was applied in order to better adapt the model and the training strategy to the specific problem. Firstly, they replaced the global average pooling, commonly located after the last convolutional layer and before the classification stage, with a row-wise average pooling in order to avoid mixing up feature maps extracted by different vertical stripes of the image (see figure 1.7). Moreover they considered a curriculum learning training strategy for the classification task, the model is thus trained firstly on "easy examples" and then fine-tuned using "hard examples"; the regression task implicitly receives such information through the target SF which represents a sort of severity value of the patient condition. Results confirm the superiority of deep learning approaches with the re-

spect to the texture analysis made with hand-crafted features. The enlargement of the dataset and the use of more sophisticated computer vision techniques has lead to better performance with respect to the previous work and narrow the gap with the visual score predicted by the human experts.

Chapter 2

Convolutional Neural Networks

An Artificial Neural Network (ANN) is a computational model based on "artificial neuron" which loosely model the behaviour of neurons in a biological brain. Like the synapses, the neurons are linked together transmitting signals from one to another, the connection strength can be adjusted by the weight associated with. Each neuron processed its input in some way and transmit its output to another neuron.

The main idea behind ANN is that it represents a very complex and general nonlinear mathematical function ideally having the description power to address the problem being faced. The generality of the function is then particularised when the ANN is trained on the specific problem adjusting its weights i.e. the connection strengths between neurons.

ANNs are trained by processing examples composed by known input and output. The training is performed mainly by determining the error between the predicted output of the network and the known output. The training procedure then adjust the network weight using a learning rule to produce outputs which are increasingly similar to the target output: as examples and repetition can increase the model prediction will become closer and closer to the target output. This type of training procedure is known as supervised learning where, giving an input, the model

parameters is adjusted based on a known output. Such system learns to perform tasks without being programmed with task-specific rules and this is particular important especially for very complex problem as computer vision where specifically programming a model-based algorithm can be very cumbersome.

A convolutional neural network (CNN) is a class of ANN most commonly applied to analyse visual imagery. They are characterised for the presence of Convolutional Layers which have properties suitable to deal with images as shift-invariance, it means that if the input shift the output also shift but stays otherwise unchanged.

This chapter is divided in four sections. In section 2.1 the basic concept to train a neural network will be introduced, section 2.2 provides a description of the some layers used in the modern state-of-the-art CNNs, 2.3 presents methodologies to interpret the network inference phase.

Here, there will be introduced some formal definitions that will be used during this work.

1. **Dataset** consisting of input-output pairs (x_i, y_i), where x_i is the input and y_i is the true ouput i.e. the desired output of the network on input x_i. The set of input-output pairs of size N is denoted as

$$X = \{(x_1, y_1), \ldots, (x_N, y_N)\} \tag{2.1}$$

   For a classification problem the output y_i has shape C x 1, where C is the number of classes to predict, y_i is a 0 valued vector except in correspondence of the true class index where the value is 1.

2. A **feedforward neural network** whose parameters are collectively denoted by $\theta \in \mathbb{R}^d$ where d represents the number of parameters. A neural network can be interpreted as a very complex function that, give some input x_i produce a

vector of probability $\hat{y}_i$ for each class.

$$F(x_i; \theta) = \hat{y}_i \tag{2.2}$$

For a classification problem the output $\hat{y}_i$ has shape C x 1 where each element represents the predicted probability that x_i belongs to the available classes. The goal of a classification problem is to make $\hat{y}_i = y_i$

3. An **error function**, $L(X, \theta)$, which defines the error between the desired output y_i and the calculated output $\hat{y}_i$ of the neural network on input x_i for a set of input-output pairs $(x_i, y_i) \in X$ and a particular value of the network parameters θ. The aim of the training phase is minimise this error function (loss function).

2.1 Training a neural network

A neural network, as already pointed out, is a very complex mathematical function that should be adapted to the specific problem adjusting its weight (parameters). Supposing that the model parameters are randomly chosen at beginning, the training procedure is a phase where the model weights are continuously adjusted to let the model prediction be closer and closer to the true output.

The training of a neural network is usually carried out defining a loss function (error function) that gives a measure of the error between the model prediction and the true output. An optimisation procedure, based on the gradient descent, has to minimise the loss function adjusting the parameter weights of the network. Because of the need to compute the derivative of the loss function w.r.t. every model parameters, the high complexity of the networks leads to the use of backpropagation method. In this section all this aspects will be discussed.

2.1.1 Loss function

In deep learning, a loss function or cost function (sometimes also called an error function) is a function that maps the network prediction and the true label into a real number intuitively representing some "cost" associated with the event. The goal of the training procedure is to minimise the loss function. A Loss function has to be differentiable w.r.t. model parameters because the optimisation algorithm is based on Gradient Descent that needs the derivative of the function to optimise.

The most common loss functions used for classification problem is the Categorical Cross Entropy Loss. Entropy is the measure of randomness in the information being processed, and cross entropy is a measure of the difference of the randomness between two random variables. Supposing that y_i is the true output and $\hat{y}_i$ is the output prediction probabilities, the mathematical form of Categorical Cross Entropy Loss is

$$L_{CE} = -\sum_{i=1}^{N} y_i log(\hat{y}_i) \tag{2.3}$$

that is 0 only if the network predict the correct label with a probability of 1.0 .

2.1.2 Optimisation

Optimisation is the problem of finding a set of inputs to an objective function that results in a maximum or minimum function evaluation. In Deep Learning, training optimisation algorithms are used to update layer's parameters in order to find the best parameters that minimise the Loss function $L(\theta)$. The research of the best model parameters is a complex problem because of the high-dimensionality and non-linearity of the space. Some techniques has been developed during the years.

This section provides an overview of the most used algorithm in Deep Learning. Firstly, it will be introduced the most basic optimiser i.e. the Gradient Descent

and the reason why more sophisticated algorithms has been developed during the year until reaching Adam optimiser that is the most used algorithm nowadays for training neural networks.

Gradient Descent

Gradient descent is a way to minimise an objective function $L(\theta)$ parameterised by a model's parameters $\theta \in \mathbb{R}^d$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$ w.r.t. to the parameters. The learning rate η determines the size of the steps taken to reach a (local) minimum.

The Batch gradient descent computes the gradient of the cost function w.r.t. to the parameters θ for the entire training dataset:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t) \tag{2.4}$$

where t denotes the current time-step. For this equation is necessary to compute the gradient for the whole dataset to perform just one update, batch gradient descent can be very slow and is intractable for datasets that don't fit in memory. This fact lead to the use of a stochastic loss function computed on a reduced size sample of the entire dataset called mini-batch. An update is performed for every mini-batch of n training samples. The batch size n depends on the number of samples that can be fit in memory.

Momentum: SGD with momentum

Due to the stochastic nature of the Loss function when mini-batches are used to compute it, a momentum term can be very useful to regularise the changes in the parameters space. The moment is a term that depends on the previous iteration, thus the updating rule utilises a linear combination of the gradient in the current iteration and the update in the previous iteration:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t) + \gamma \theta_t \tag{2.5}$$

This method introduces a extra hyper-parameter γ used to control the moment effect i.e. the importance of the previous iteration.

Adaptive Learning algorithms: AdaGrad and RMSprop

The learning rate hyperparameter controls how quickly the parameters are updated for each iteration step. The trade-off between a large and a small learning rate is an important factor to consider. Small learning rates could cause the model to learn slowly with the risk to get stuck, whereas large learning rates cause the parameters to change too fast and the algorithm can never reach the optimum. A simple rule can address this problem: at begin of the training the learning rate is set to high values to ensure fast convergence and it is slowly decreased along with the training for fine-tuning. The drawback of this procedure is that the learning rate is the same for all parameters, given a decreasing learning rate it might end up in a situation where the parameters for infrequent features are weakly updated. The ideal situation is to ensure that the learning rate decrease more for frequent features and less for sparse features.

Adaptive Gradients (AdaGrad) provides a simple approach for changing the learning rate over time and adaptively scaling it for each dimension. The updating rule is

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\epsilon + G_t}} \odot g_t \tag{2.6}$$

$\odot$ indicates the element-wise matrix-vector multiplication, η is the initial learning rate, ϵ is some small quantity to avoid the division of zero, g_t is the gradient estimate in time-step t i.e. $g_t = \nabla L(\theta_t)$, $G_t \in \mathbb{R}^{d \times d}$ is a diagonal matrix where each diagonal

element is the sum of the squares of the gradients w.r.t. θ_i up to time step t. G_t allows the learning rate to be different for every parameter, in particular it results in a high value for frequent features and a low value for infrequent features. G_t can be fully utilised in the updating rule 2.6 but computing the square root of a full matrix is computationally infeasible in a high-dimensional case.

Root Mean Squared Propagation (RMSprop) is very close to Adagrad, except for it does not update the learning rate using the sum of previous gradients, but instead it is more restricted to gradients for recent time-steps.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\epsilon + E[g^2]_t}}\, g_t \tag{2.7}$$

where

$$E[g^2]_t = (1-\gamma) g_t^2 + \gamma\, E[g^2]_{t-1} \tag{2.8}$$

The term $E[g^2]_t$ is computed using Momentum, it means that the new observation is balanced on the previous observation though γ. RMSprop allows the learning rate to increase and decrease during the training despite AdaGrad where the learning rate is monotonously decreasing.

Adam

Adaptive Moment Estimation (Adam) is probably the optimiser that performs the best on average. When introducing the algorithm, the authors of original paper [12] list the attractive benefits of using Adam on non-convex optimisation problem as straightforward to implement, computationally efficient, little memory requirements, invariant to diagonal rescale of the gradients, well suited for problems that are large in terms of data and/or parameters, appropriate for non-stationary objectives, appropriate for problems with very noisy/or sparse gradients, hyper-parameters have

intuitive interpretation and typically require little tuning. Here is the update rule for Adam:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2.9}$$

where

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{2.10}$$

$$\begin{aligned} m_t &= (1-\beta_1)g_t + \beta_1 m_{t-1} \\ v_t &= (1-\beta_2)g_t^2 + \beta_2 v_{t-1} \end{aligned} \tag{2.11}$$

The gradient used to the parameter update m_t is similar to the one used for SGD with momentum, a momentum term controlled by β_1 is present to regularise the parameter's change. The updating of the learning rate can be regarded as RMSProp, in which the running sum of the squared gradient is kept and a momentum term controlled by β_2 is present. The final update equation can be seen as a combination of RMSProp and SGD with Momentum. In other words, the algorithm updates an exponential moving average of the gradient m_t and the squared gradient v_t. Since the moving averages are estimates of the mean and the raw moment of the gradient and they are initialised to 0, in the initial time-steps, the moment estimates can be biased toward zero. This is counteracted using a bias-corrected estimate $\hat{m}_t$ and $\hat{v}_t$.

As previously said, this algorithms does not require much effort to tune hyper-parameters, authors suggest $\epsilon = 10^{-8}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$.

2.1.3 Backpropagation

A neural network can learn its parameters using gradient descent based algorithms but all of these need to compute the gradient of the cost function w.r.t. the network

parameters which number can be of the order of millions.

Backpropagation is an algorithm for supervised learning of artificial neural networks which calculates the gradient of the error function with respect to the neural network's weights. This approach is much more efficient compared to trivially computing the gradient of each weight separately. In backpropagation the calculation of the gradient proceeds backward through the network, the gradient of the final layer is calculated first and the gradient of the first layer is calculated last. Once the gradient of the loss function is computed w.r.t. one layer's parameters, using the chain rule, the same computation is used for the previous layer. This method is very effective, especially for deep structured networks and it takes more advantage of specialised hardware to further improve performance.

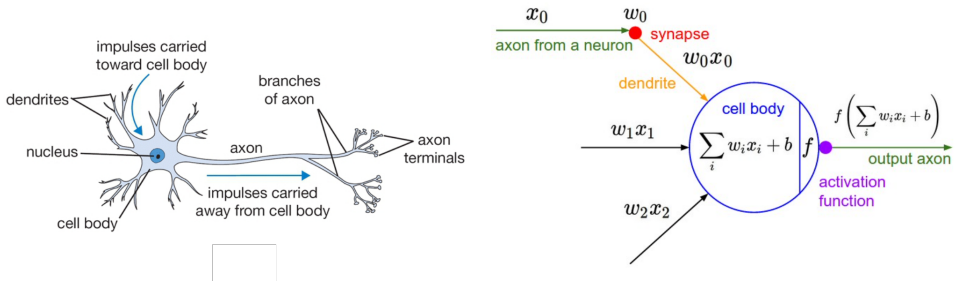
2.2 Neural Network layers

Every neural network is composed of blocks called layers where the information goes through. Every layer can be seen as a function that processes an input signal and gives out an output signal. There are many types of layers that perform different operations, most of them are parameterised function in which parameters has to be learned during the training phase. Due to the derivative of the functions that the optimisation algorithm requires, the function representing a layer must be differentiable w.r.t. its trainable parameters.

In this section some layers will be explained, how they process the information and the reason they are used for. Particular attention is given to layers widely used in computer vision which constitute the main blocks of network architectures used in this work.

2.2.1 Fully Connected Layer

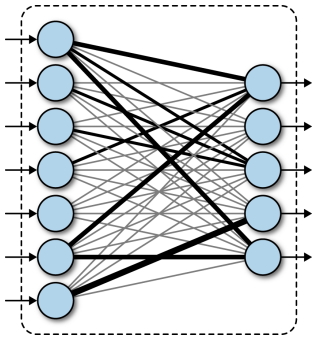
The basic operation of a Fully Connected layer is performed by a Neuron cell that computes a weighted sum over an input vector $x = \begin{bmatrix} x_0 & x_1 & \dots & x_n \end{bmatrix}$ plus a bias as showed in Figure 2.1.



Source: cs231n by Stanford

Figure 2.1: A biological neuron (left), its mathematical form in Deep Learning (right)

A Fully Connected Layer is typical composed by more output Neuron cell as depicted in Figure 2.2 where all the inputs from one layer are connected to every activation unit of the next layer, that's why is also called Dense layer. Mathematically,



Source: O'Reilly Media

Figure 2.2: A dense layer representation

we can think of a linear layer as a function which applies a linear transformation W

on a vectorial input x and output a vector with different dimension y,

$$y = Wx \tag{2.12}$$

where the bias parameter is omitted for simplicity. The matrix W is the weight matrix that has to be trained. Each value represents the connection strength between a single input signal to a single output signal.

2.2.2 Activation or Nonlinearity

The capacity of the neural networks to approximate any functions, especially non-convex, is directly the result of the non-linear activation functions. Every kind of activation function takes a vector and performs a certain fixed point-wise operation on it. The choice of activation functions in Deep Neural Networks has a significant impact on the training dynamics and task performance, as in the case of vanishing or exploding gradients problems.

The two most promising activation functions used nowadays are described below and shown in figure 2.3.

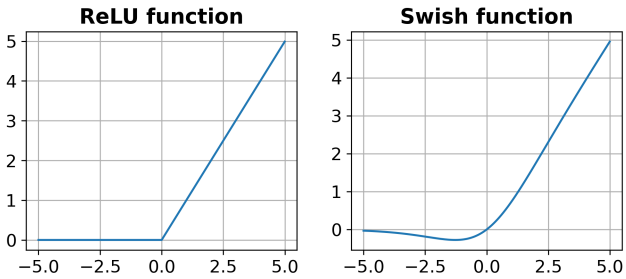


Figure 2.3: ReLU and Swish activation functions input-output relation

Rectified Linear Unit: The ReLU has the following mathematical form

$$y = max(0, x) \tag{2.13}$$

During the years, the ReLU activation was found to greatly accelerate the convergence of the training procedure due to its linear non-saturating form. It does not suffer from vanishing or exploding gradient and involves cheap operations compared to other activation functions. However, ReLU removes all the negative information resulting in a destructive operation for certain datasets and architectures.

Swish activation function: The Swish has the following mathematical form

$$f(x) = x\sigma(x) \tag{2.14}$$

where $\sigma(x)$ represents the Sigmoid function. The swish function is a smooth, non-monotonic activation function that results to work better when ReLU on deeper models across many challenging datasets. Despite the ReLU, its derivative is not 0 for half of the values of the input x avoiding vanishing gradient problems.

2.2.3 Spatial Convolution

In image processing, a different type of layer named Convolutional layer is used instead of a fully connected layer. A convolutional layer is composed by a set of N_k filters $F = f_1, ..., f_{N_k}$ which are convolved spatially with the input image x, they produce a set of N_k 2D features maps:

$$z_k = f_k * x \tag{2.15}$$

where * is the convolution operator. Spatial convolution layers take advantage of the fact that the input exhibits spatial relationships, neighboring pixels are not affected by their location within the image. When the filter correlates well with a region of the input image, the response in the corresponding feature map location is strong. Unlike conventional linear layers, weights are shared over the entire image reducing the number of parameters per response and equivariance is learned (i.e. an object

shifted in the input image will simply shift the corresponding responses in a similar way). A spatial convolution layer is not defined by the spatial size of the input features maps, neither by the size of the output feature maps, but by the number of filters (number of output channels), the properties of its filters (number of input channels, wide, high) and the properties of the convolution (padding, stride).

2.2.4 Spatial Pooling

In Convolutional Neural Networks, a pooling layer is used to downsample the feature maps to reduce the dimension of the feature maps and to provide spatial invariance. Spatial pooling in a hierarchical feature model makes each feature resilient to minor shifts in position since multiple positions are pooled together. If this technique is applied repeatedly and hierarchically, then an object can be recognised despite substantial spatial distortions (like shift, tilt, rotation), making object recognition more robust and less fragile. Pooling involves selecting a pooling operation. The region of the images where the operator must be applied is called patch, it is almost always 2x2 pixels applied with a stride of 2 pixels. The pooling operation is specified, rather than learned.

Two common functions used in the pooling operation are:

- Average Pooling: Calculate the average value for each patch on the feature map.
- Maximum Pooling (or Max Pooling): Calculate the maximum value for each patch of the feature map.

2.2.5 Batch Normalisation

The training of deep neural networks can be challenging as it can be very sensitive to various factors i.e. the random initial weights, the optimisation algorithm config-

uration and the input statistical parameters. A batch Normalisation layer is used to stabilise through normalisation of the layers' input by re-centering and re-scaling. Ideally, the normalisation step would normalise with parameters computed on the entire dataset but this is impractical because global information is unknown, thus the normalisation step is restrained to each mini-batch is in the training process.

Denoting as B the mini-batch of size m, the mean and variance of B could be computed as

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.16}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{2.17}$$

that represent a sort of empirical mean and variance of the entire dataset. Define with $x = (x^{(1)}, \ldots, x^{(d)})$ a layer of the network with d-dimensional input, each dimension is normalised separately using the following transformation:

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\sigma_B^{(k)^2} + \epsilon}} \tag{2.18}$$

where $k \in [1, d]$ and $i \in [1, m]$; $\mu_B^{(k)}$ and $\sigma_B^{(k)}$ are the mean and variance of the channel k, ϵ is a constant for numerical stability. The resulting normalised activation $\hat{x}^{(k)}$ have zero mean and unit variance. In order to recover the representation power of the network layer, a learnable transformation step to shift and rescale the input channels is performed as

$$y_i^{(k)} = \gamma^{(k)}\hat{x}_i^{(k)} + \beta^{(k)}, \tag{2.19}$$

where the parameters $\gamma^{(k)}$ and $\beta^{(k)}$ are learned in the optimisation process.

2.2.6 Depthwise separable convolutions

The Depthwise separable convolution block was introduced by MobileNetV1 [11] with the purpose to replace the traditional convolution layers in a more efficient

way, in fact, this block is mainly used as building block for mobile architectures reaching high accuracy while keeping the parameters and mathematical operations relatively low. The main idea is to factorise traditional convolution by separating spatial filtering (spatial convolution keeping the channels separate) from the feature generation mechanism (depthwise convolution).

For example, in the case of a 3x3 convolutional layer on 16 input channels and 32 output channels, a normal convolution layer would require 3x3x16x32=4608 parameters. In a depthwise separable convolution, the spatial filtering is performed by traversing the 16 channels input features map with only 1 3x3 kernel each giving again a 16 channels features map as output. Consequently, the depthwise information is extracted by 32 1x1 kernel size convolution giving a 32 channels features map as output. The total number of parameters is 16x3x3 + 16x32x1x1 = 656, a number much smaller than the previous one obtained with the traditional convolutional layer. The two-step convolution works thanks to the hypothesis that spatial and depthwise information can be decoupled and processed separately.

2.2.7 Mobile Inverted Residual Bottleneck Block - MBConv

The block treated in this section comes from MobileNetV2 [21] which is the next version of MobileNetV1. MobileNetV2 hardly uses Depthwise separable as its main building block but extends its predecessor with two main ideas: inverted residuals block and linear bottlenecks. In the following the block is built starting from the main concept of residual block applied to depthwise convolutional layer.

Residual block

A Residual block uses a skip connection to connect the input features map to the output features map simply adding them together. This turned out to be very effective with deep network where the loss of information between layers is a very

challenging problem. The skip connection leads to easy access to earlier features maps that have not been modified by the convolutional block. In a depthwise residual layer, a residual block can be applied as in figure 2.4.

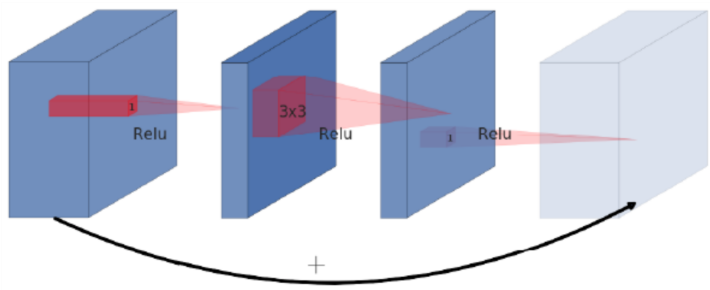


Image taken from [21]

Figure 2.4: Depthwise separable convolutions Residual block

A residual block usually follows a wide, narrow, wide depth of features map, where the skip connection is performed at wide final features map.

Inverted Residual block

On the other hand, the main building block in MobileNetV2 uses an apposite approach in terms of features map depth, it is a narrow, wide, narrow approach. The first step widens the network using 1x1 convolutions and then a 3x3 convolution is performed to the deep features map, afterwards, another 1x1 convolution squeezed the network to obtain the original number of channels and to sum up with the input throught the skip connection.

Linear BottleNecks

A linear BottleNecks is made by removing the nonlinear activation to the last convolution layer before adding it to initial activation. In the original paper the authors show empirically that a non-linear activation function like ReLU destroy informa-

tion especially when the input's channels are less that the output's. In figure 2.5 there is the main building convolutional block of MobileNetV2.

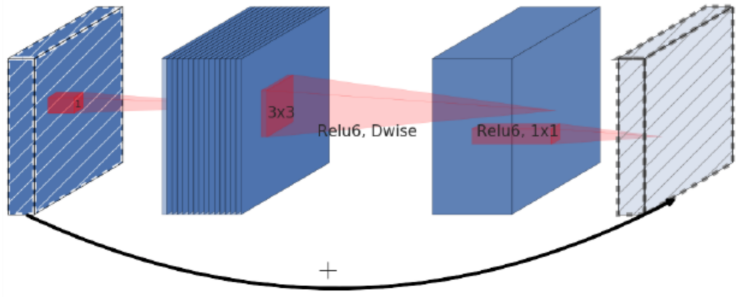


Image taken from [21]

Figure 2.5: Inverted Residual and Linear Bottleneck - MobileNetV2 layer

Squeeze-and-Excitation layer

In [10] the authors built up a MobileNetV2 introducing lightweight attention modules based on squeeze-and-excitation into the bottleneck structure as shown in figure 2.6. The basic concept of Squeeze-and-Excitation layer is adding up a parameter to each channel of a convolutional block so that the network can adaptively adjust the weighting of each feature map. The feature recalibration is performed in the following way. The output features of dimension H x W x C are passed through a squeeze operation, which aggregates the feature maps across spatial dimension H x W averaging the value of each feature maps to produce a channel descriptor of size 1 x 1 x C. In order to let the module be flexible i.e. learn nonlinear interaction between channels, and to limit the model complexity, the channel description is passed through a dense layer of reduced dimension and then another dense layer to recover the original dimension. The output channel description is then used to weigh the original features map.

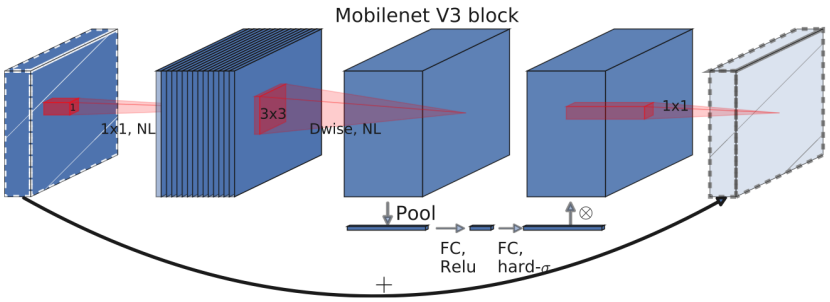


Image taken from [10]

Figure 2.6: Inverted Residual and Linear Bottleneck with Squeeze-and-Excite

2.2.8 Deep Architectures

According to the universal approximation, theorem [9], given enough capacity, a feedforward network with a single layer has the ability to represent any mathematical function. However, the layer might be massive and the network is prone to overfitting the data. Therefore, a common trend in the research community is to build deep networks.

Deep architectures are built stacking up layers in a way that the information flows from the input through all the hidden layers and then to the final layer. The use of deep architecture rather than shallow networks is justified for many reasons and it is proven to be more effective for modeling complex problems. The key concept is the modularisation of the information. Layers increasingly build more abstract representations of the input and thus the information built from one layer is used by successive layers to build a higher level representation.

2.3 Model interpretability

Deep learning models are generally black-box because it is very difficult to interpret the reason why a particular model made a certain choice. In the last few years, some

methods were proposed to explain how these models work and thus try to turn the black-box into a grey-box.

Methods for explaining neural networks generally falls within two categories, namely Saliency Methods and Feature Attribution (FA). Saliency methods are used to understand what is going on inside the network, they allows to detect which weights are being activated given some input or what regions of an image is being detected by a particular layer. Otherwise, there are Feature Attributions methods that indicate how much each input feature contributed to the model prediction, in fact, the prediction usually comes along with feature attribution information.

2.3.1 Integrated Gradients

Integrated Gradient is an attribution method presented in the paper Axiomatic Attribution for Deep Networks [25] where two axioms for the correctness of attribution methods are presented in order to avoid errors that stem from the misbehavior of the model versus the misbehavior of the attribution method. The Integrated Gradient method satisfies both the axioms listed below:

- Sensitivity
- Implementation invariance

An attribution method satisfies Sensitivity if for every input and baseline that differ in one feature but have different predictions then the differing feature should be given a non-zero attribution. Furthermore, attribution methods satisfy Implementation invariance if the attributions are always identical for two functionally equivalent networks; two networks are functionally equivalent if their outputs are equal for all inputs, despite having different implementations.

Integrated Gradient, in addition to satisfy both axioms, has the important property that it does not need any instrumentation of the network and can be computed

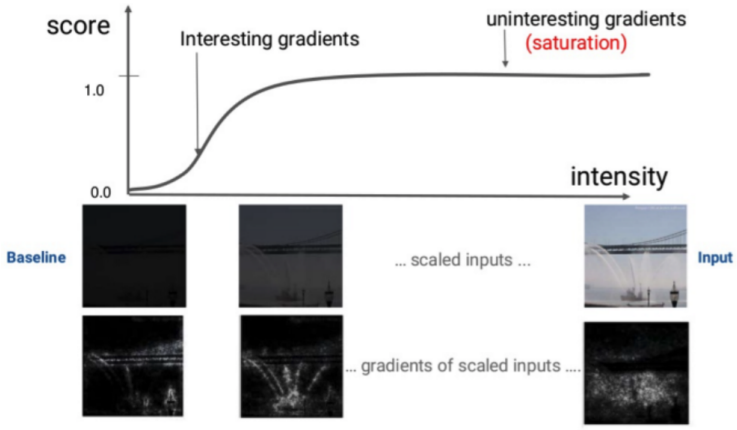


Image by Krishnaram Kenthapadi (Explainable AI in Industry (KDD 2019 Tutorial))

Figure 2.7: Integrated Gradient descent

easily with a few calls to the gradient operation. The method basically performs a (features x gradient) operation which tells how much a particular feature influences the network prediction but this simple approach fails when the network saturates i.e. in the vicinity of the input where the gradient is close to 0. The particularity of Integrated Gradient is that it desaturates the network defining a baseline in order to see the effect of the input features on the input predictions. Following the figure 2.7 the image intensity is firstly set to 0, nextly the intensity is scaled up until to get the original image. The image shows that the interesting part of the gradient is when the image is scaling up and not when it has the full intensity because the network is saturated, meaning that the model already took the decision. The black image is called "baseline", an input with a neutral prediction. The introduction of a baseline is the major improvement brought by this method w.r.t. other features attribution methods.

Chapter 3

Problem definition & experimental setup

In this chapter, we introduce problem is introduced and it is provided a description of the experimental setup, i.e. the deep learning models and metrics. In addition, there is a detailed description of the available dataset. The experiments are carried out entirely using PyTorch, an open-source Deep Learning framework for Python programming language.

3.1 Problem definition

Consider the definition of the dataset as in equation 2.1 where x_i represent a single image of our dataset with size (H x W), the single image corresponds to a frame of the available videos; y_i represent the true label associated to the video where the frame x_i is taken from, y_i contains 2 values which represent the true possible classes to predict. The classes are indicated as

- $BEST$: negative class

  - Image x_i is associated to an healthy patient, $y_i = \begin{bmatrix} 1 & 0 \end{bmatrix}$

- *RDS*: positive class

  - Image x_i is associated to a patient affected by RDS syndrome, $y_i = \begin{bmatrix} 0 & 1 \end{bmatrix}$

The networks model is defined as in equation 2.2, where $\hat{y}_i$ represents the predicted probabilities for each class. Given an input image x_i the problem is to predict the class to which it belongs. Different metrics will be used to evaluate the model performances.

3.2 Neural network architectures

This section is dedicated to a theoretical description of the models that will be used in the experiments. Every model is trained using the Cross-Entropy Loss and the Adam optimiser both provided by PyTorch framework.

The proposed architectures to address the present problem are EfficientNet-B0, EfficientNet-B1, EfficientNet-B2, ResNet18 for which a theoretical discussion will be provided. For every architecture, the last dense layer is replaced with a new dense layer composed of 2 neurons, representing the score associated with each class. A Softmax layer is then added to normalise the scores into probabilities. Table 3.1 shows the number of parameters for each proposed neural network.

A transfer learning approach is adopted, in fact, the models' parameters are pretrained on ImageNet dataset. No layers have been frozen for the training phase due to the very different nature of the current problem with respect to ImageNet dataset scenario. The pretrained parameters are just used as a baseline on which the models are trained again.

Model	# parameters
EfficientNet-B0	$4,01$ M
EfficientNet-B1	$6,52$ M
EfficientNet-B2	$7,79$ M
ResNet18	$11,18$ M

Table 3.1: Number of parameters for different networks

3.2.1 ResNet

In 2015, [8] presented ResNet, an artificial neural network that utilises skip connections to jump over some layers. The advantage was the ability to train deep neuronal network of hundreds layers achieving compelling performances.

The difficulties of training deep neural networks is a well-known phenomenon, it causes the performance to degrade even in the training phase, a problem resulting different from over-fitting. The authors of the original paper simply showed it in the following way. They took two same structured architectures, a shallow one and a deeper one. Although the latter one has more parameters and representation power, the performance was lower than the shallower one, testifying a network optimisation issue due to a vanishing gradient phenomenon. In such a situation, trivially, it would be sufficient that the deeper model learns identity functions in the latest layers in order to achieve, at least, the same performance of the shallower network. This simple fact led to a key concept: it is easier, for a trainable layer, to learn the residual input-output mapping than letting them directly fit the desired underlying mapping. Saying so, they introduced the residual block as depicted in figure 3.1 which constitutes the base block structure of the entire ResNet architectures which can be of different sizes. In the present work a ResNet composed of 18 layers, named ResNet18, is considered and it is shown in figure 3.2.

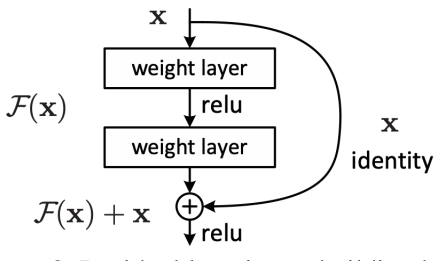


Image taken from [8]

Figure 3.1: A residual block

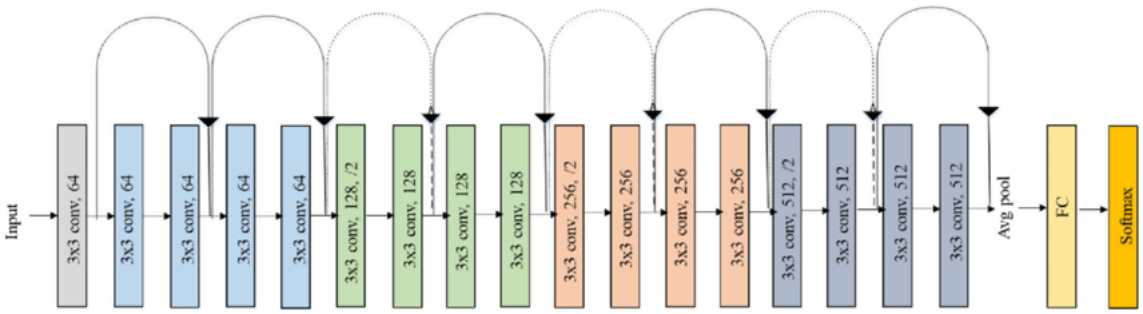


Image taken from [18]

Figure 3.2: ResNet18 architecture representation

3.2.2 EfficientNet

EfficientNets are a family of image classification models which differ, from other state-of-the-art models, by achieving the same accuracy being an order of magnitude smaller and faster than other models as pointed out in the original paper [27]. In this work, the problem of network scaling was studied, in particular how to balance network depth, width and resolution when the size of the network has to be increased. EfficientNets was built to prove the effectiveness of the proposed method. The baseline on which the family is built is called EfficientNet-B0 and its main building block is mobile inverted bottleneck MBConv from MobileNetV2, to which squeeze-and-excitation optimisation is added: table 3.2 lists the architecture

Stage	Operator	Resolution	# Channels	Layers
1	Conv3x3	224x224	32	1
2	MBConv1, k3x3	112x112	16	1
3	MBConv6, k3x3	112x112	24	2
4	MBConv6, k5x5	56x56	40	2
5	MBConv6, k3x3	28x28	80	3
6	MBConv6, k5x5	14x14	112	3
7	MBConv6, k5x5	14x14	192	4
8	MBConv6, k3x3	7x7	320	1
9	Conv1x1 & Pooling & FC	7x7	1280	1

Table 3.2: EfficientNet-B0 baseline network

layers.

The networks used in this work are EfficientNet-B0, EfficientNet-B1, EfficientNet-B2 which have the same implementation of the original paper except for the use of swish non-linearities despite the ReLU activation function.

3.3 Metrics

Metrics are used to evaluate model performances in an interpretable way. They must be specific and significant for the problem being faced. Formally, in a binary classification problem, positive and negative classes are defined. Here we define the RDS class as positive and the BEST class as negative.

We define the number of positive cases as P and the number of negative cases as N, the number of correctly classified as positive as TP (true positive) and the number of correctly classified as TN (true negative), the number of wrongly classified as positive as FP (false positive) and the number of wrongly classified as negative as

FN (false negative). Moreover, TPR (true positive rate) is the ratio of true positives out of the positives and the FPR (false positive rate) is the ratio of false positives out of the negatives,

$$TPR = \frac{TP}{P}, \qquad FPR = \frac{FP}{N}. \tag{3.1}$$

Accuracy

Classification accuracy is the simplest metrics one can imagine, it is defined as the number of correct predictions divided by the total number of predictions.

$$Accuracy = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.2}$$

Confusion matrix

A confusion matrix is a metric to summarise the performance of a classification algorithm dividing the right and wrong classification for different classes. It is substantially a table which diagonal represents correct classification for each class and in the other element there are wrong classifications. In figure 3.3 there is an example with binary classification where TN and TP are placed on the diagonal, and the FP and FN are placed on the off-diagonal.

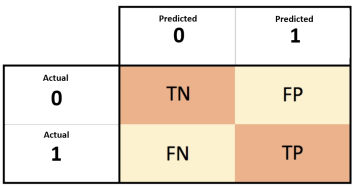


Figure 3.3: Confusion Matrix representation

Sensitivity and specificity

Sensitivity and specificity are metrics usually used in diagnostic tests, sensitivity is a measure of how well a model can identify true positive cases and, on the other side, specificity is a measure of how well a model can identify true negative cases. In particular, sensitivity is the probability that, given a positive case, the test results as positive

$$Sensitivity = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{3.3}$$

Specificity is the probability that, given a negative case, the test results as negative

$$Specificity = \frac{TN}{N} = \frac{TN}{TN + FP} \tag{3.4}$$

There is a trade-off to consider between sensitivity and specificity, in fact, higher sensitivities mean lower specificities and vice versa. The trade-off is directly related to the chosen classification threshold which can be tuned depending on the clinical importance that predicting false positive or negative have.

Receiver operating characteristic (ROC-AUC)

A receiver operating characteristic (ROC) is a plot that illustrates the performance of a binary classifier as its discrimination threshold is varied. Area under the curve (AUC) is a threshold-free classification metric and it represents the area under the ROC curve, it can vary between $[0, 1]$. This metric indicates the degree of measure of separability between classes. Higher AUC, the better the model is at distinguishing between different classes.

Spearman correlation coefficient

The Spearman rank correlation coefficient is a non-parametric measure of correlation. It describes how well the relationship between two variables can be described

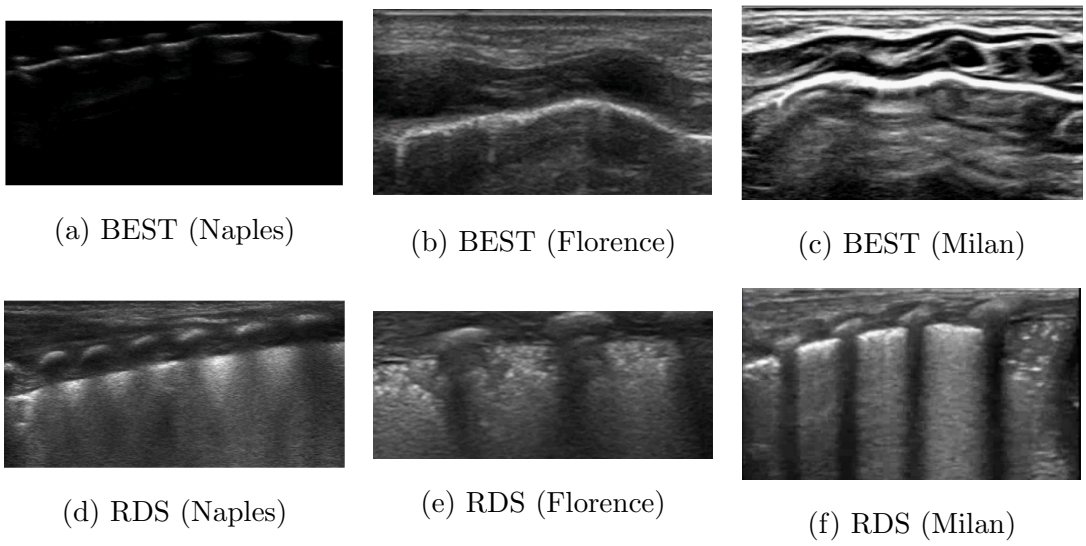
using a monotonic function. This coefficient can vary in the range $[-1 + 1]$. Being a metric used for a regression problem, here is used to correlate the positive class model output probability with the SF coefficient (SF should increase as the healthiness status of the patient increases).

3.4 Dataset

The available dataset contains 745 LUS videos of 72 patients with different respiratory statuses i.e. healthy patients and patients affected by Respiratory Distress Syndrome (RDS). Their gestational age ranges between 25 and 40 weeks. Videos are acquired from different centers, namely Naples, Florence and Milan, and different US instruments have been used, yielding different image properties, like contrast, brightness, resolution and frame rates. In particular, each video contains 6 up to 10 frame with different resolutions. Figure 3.4 shows examples of video frames for different centers and respiratory status.

Every video is associated with a true label and a pulse oximetric saturation ratio (SF) i.e. a value between 101 and 476. The true label is binary, i.e. $BEST$ or RDS, while the SF value is numeric and gives a measure of the patient's health, the higher the value, the healthier the patient is likely to be. SF values are clipped to 476, higher values distinguish healthy or completely healed patients representing no useful information for this work. Details of the dataset are in table 3.3 which shows videos and patient counts for different centers and different statuses.

In general, the dataset contains relatively few samples for a classical image classification problem using deep learning techniques. This fact will be addressed using data augmentation, different frames selection for each video and a modified version of training-test procedure as will be pointed out in the next chapter.



(a) BEST (Naples) (b) BEST (Florence) (c) BEST (Milan)

(d) RDS (Naples) (e) RDS (Florence) (f) RDS (Milan)

Figure 3.4: Some frames obtained by videos of different respiratory status and different hospital center

3.4.1 Images interpretation

In this section some insight of the images are explained with the purpose of understanding some property characterising frames belonging to a healthy or sick patients. First of all, the pleural line assumes very importance for the classification, it is typically smooth and regular for healthy patients otherwise it is irregular for unhealthy patients testifying pleural line inflammation. Another very important aspect to consider for classification is the presence of A-line and B-line underneath the pleural lines which are artifact made by the LUS instruments. The A-lines are horizontal artifact repetitions of the pleural line displayed at regular intervals, their presence is usually associated to an healthy status. B-lines consists of vertical artefact and testify the presence of liquid therefore they are usually associated to an unhealthy status. B-lines are absent under normal conditions. Moreover, in some LUS image ribs are present which leads to a partial occlusion of the pleural line as in figure 3.4d where the pleural line is dashed. In addition, the ribs can be confused by the

Status	Number of patient	Number of videos	Patients per center Naples	Patients per center Milan	Patients per center Florence	Videos per center Naples	Videos per center Milan	Videos per center Florence
Healthy	43	328	23	10	10	188	80	60
RDS	29	417	11	8	10	175	128	114
Tot.	72	745	34	18	20	363	208	174

Table 3.3: Collected Dataset

model with the pleural line; the model should be able to distinguish between this two lines. Usually pleural line presents higher pixel values but this not holds for all images, like the one in figure 3.4c where the image is saturated.

3.5 Preprocessing and Data Augmentation

Data augmentation is the practice of using data already present into the dataset to create new training examples to help the machine learning models to generalise better. In this work, augmentation and preprocessing have been performed using Albumentation library, an open-source Python library for fast and flexible image augmentations. The preprocessing and data augmentation is performed differently for the train and for the validation/test phase.

For what concern the training phase, the training videos dataset is modified accordingly to the following two operations:

- Balancing: the smallest class videos are replicated up to the size of the biggest class,
- Replicate by a factor R: all video samples are replicated by factor $R = 3$,

a random frame is then selected by each video. Balancing is performed to weigh

Operation	Parameters			
1 Rotate	range		probability	
	$[-10°, +10°]$		50%	
2 Resize Crop	height	width	scale	ratio
	224	461	$[0.6, 1.0]$	$[0.8, 1.5]$
2 Horizontal flip	probability			
	50%			
3 Color jitter	brightness	contrast	probability	
	25%	25%	50%	

Table 3.4: Train image augmentations

equally the two classes in the training phase. Due to the random selection of the frame, video samples are then replicated.

Once the frame is selected from the video, the image is row-cropped on the upper side because the lowest pixel rows contain no information: the pleural line is always on the top and the artifacts intensity is practically absent in the bottom. Experimentally a number of 224 row-pixels are chosen resulting in a good compromise for all video frames. Afterward, a random rotation of the image is performed in the range $[-10°, +10°]$ in order to simulate different incidence angles of the probe. After there is a Random resize crop. RandomResizedCrop is a type of image data augmentation where a crop of a random size of the original size and a random aspect ratio of the original aspect ratio is made. This crop is finally resized to a given size (height and width). In the next step, there is a random horizontal flip that simulates different sides of inspection, a vertical flip is not applied because the pleural line would result upside-down. The final image augmentation is a random adjustment of

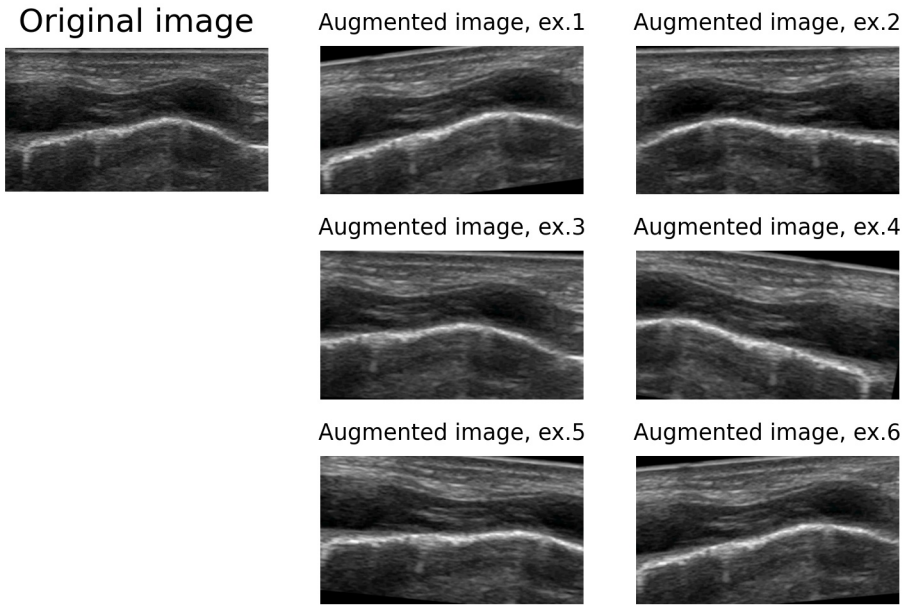


Figure 3.5: Data augmentation for a BEST image

brightness and contrast of 25% to simulate different instrument usage. Furthermore, some preprocessing steps are performed in order to adapt the image to the network input. The resulting image is scaled into range $[0, 1.0]$ and then is normalised using parameters computed on the entire dataset, specifically a mean of 0.1250 and a standard deviation of 0.1435 are used. The image is then transformed into a PyTorch tensor. In table 3.4 there is the pipeline of the applied augmentations.

For what concerns the validation and test phases, a number of 6 frames are taken for each video. A fixed number is chosen because different videos can contain a different number of frames and thus weigh different on the total scores. Preprocessing of the validation/test images is performed in order to adapt them to the inputs the network is trained with; in order, an image cropping, an image resize, a value

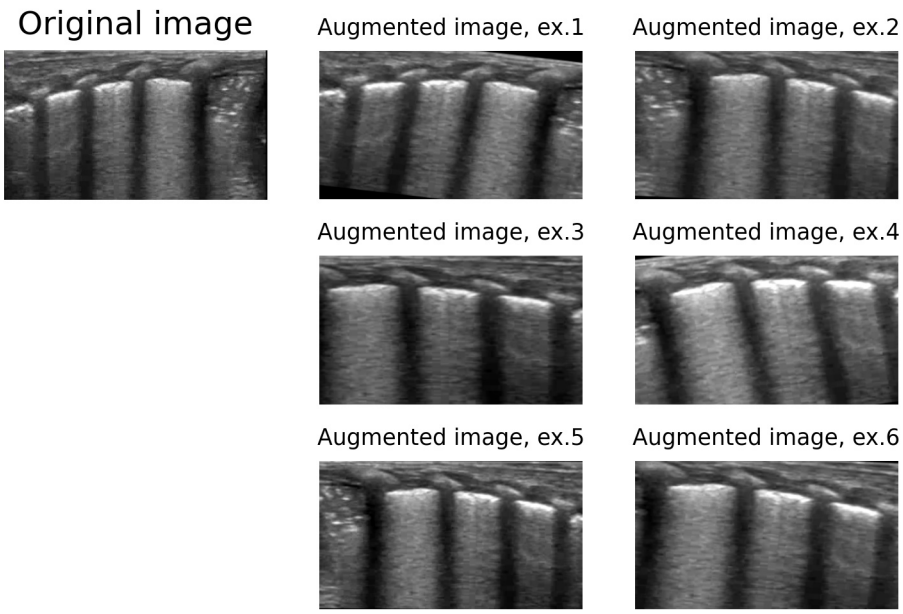


Figure 3.6: Data augmentation for an RDS image

scaling, a normalisation and a transformation to a PyTorch tensor.

Chapter 4

Model performance and results

In this chapter, the numerical experiment results are shown. Firstly, the strategy of sample selection is detailed described where a cross-validation approach is adopted. The sample selection is performed at video level and subsequently, frames are extracted by each video. The results for the proposed architectures are reported, the prediction for each fold are aggregated together in order to compute the metrics using the overall dataset. The best model is chosen and few considerations about its predictions are presented. For this model, there is a section of features attribution exploring the image region where the model focuses more to perform its prediction.

4.1 Group k-fold cross-validation

Cross-validation is a statistical method used to estimate the skill of machine learning models on unseen data. It is commonly used to prevent problems like overfitting and it generally results in a less optimistic estimate of the model skill. This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining k-1 folds. To reduce the variability of the model performance,

the method is repeated for each fold (cross-validation) and the validation results are combined (e.g. averaged) over the folds to give an estimate of the model's predictive performance.

Group k-fold is a variation of k-fold which ensures that samples belonging to the same group are not present in both testing and training sets. The model could learn specific features from samples belonging to the same group generating training overfitting. Here, videos associated with the same patient are grouped together yielding the model to don't use the video associated with the same patient in both training and testing. This approach ensures better performance esteem when the model has to classify new patients' videos.

Due to the poor number of samples, a different approach is used for evaluation and testing. Usually, after the cross-validation, the final model is tested on a single never-seen test set and final performance scores should be given. Because of the small number of samples, the risk that the test sample is not representative of the entire population is high, therefore a little different approach is adopted in which the test set changes for different training cycles and the performance results are evaluated by aggregating predictions obtained on singles test sets.

In particular, the following procedure is adopted:

1. Shuffle the dataset randomly.
2. Split the dataset using group k-fold with $k = 10$, grouping videos associated with the same patient.
3. For each unique fold s with $s \in [0, .., (k-1)]$:
   (a) Take fold s as test fold, $(s-1)$[1] as validation fold and the remaining as training folds,

[1]Consider the module operation on k, e.g. if $(s-1) = -1$, $(s-1)\ mod\ k = (k-1)$

(b) Fit the model using a number of 6 epochs on the training set and evaluate it on the validation set,

- Early stopping: Consider the epoch which have the lowest validation loss. Save the model trained until this epoch and call it *best-model-s*.

4. Summarise the skill of the model using the predictions on the entire dataset:

(a) For each test fold s with $s \in [0, .., (k-1)]$, consider the *best-model-s* predictions.

The figure 4.1 shows the class and hospitals counts for different fold.

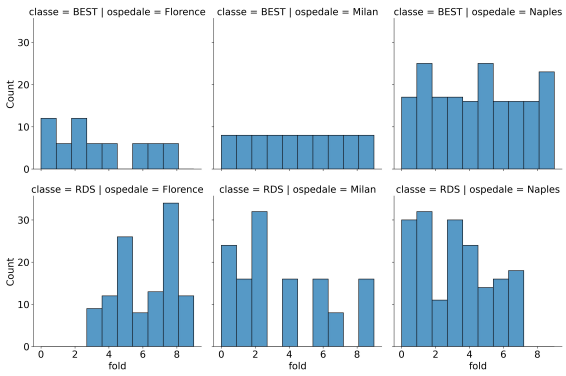


Figure 4.1: Counts for different folds

4.2 Numerical experiment results

Different neural networks were trained, namely EfficientNet-B0, EfficientNet-B1, EfficientNet-B2 and ResNet18. The models' predictions are available for each test fold and are thus merged together to obtain predictions on the entire dataset.

Setting a threshold of 0.5 to compute the predicted class, table 4.1 reports the frame-level metrics scores for different models. Moreover the video-level metrics are

Frame-level results					
Model	Accuracy	Sensitivity	Specificity	Confusion matrix	Spearman correlation
EfficientNet-B0	0.9447	0.9099	0.9888	$\begin{bmatrix} 1946 & 22 \\ 225 & 2272 \end{bmatrix}$	0.8262
EfficientNet-B1	0.9498	0.9387	0.9639	$\begin{bmatrix} 1897 & 71 \\ 153 & 2344 \end{bmatrix}$	0.8044
EfficientNet-B2	0.9315	0.8827	0.9934	$\begin{bmatrix} 1955 & 13 \\ 293 & 2204 \end{bmatrix}$	0.7977
ResNet18	0.9471	0.9143	0.9888	$\begin{bmatrix} 1946 & 22 \\ 214 & 2283 \end{bmatrix}$	0.8171

Table 4.1: Aggregate results frame-level for different model

Video-level results					
Model	Accuracy	Sensitivity	Specificity	Confusion matrix	Spearman correlation
EfficientNet-B0	0.9490	0.9137	0.9939	$\begin{bmatrix} 326 & 2 \\ 36 & 381 \end{bmatrix}$	0.8292
EfficientNet-B1	0.9557	0.9376	0.9787	$\begin{bmatrix} 321 & 7 \\ 26 & 391 \end{bmatrix}$	0.8104
EfficientNet-B2	0.9396	0.8945	0.9969	$\begin{bmatrix} 327 & 1 \\ 44 & 373 \end{bmatrix}$	0.7993
ResNet18	0.9517	0.9185	0.9939	$\begin{bmatrix} 326 & 2 \\ 34 & 383 \end{bmatrix}$	0.8210

Table 4.2: Aggregate results video-level for different model

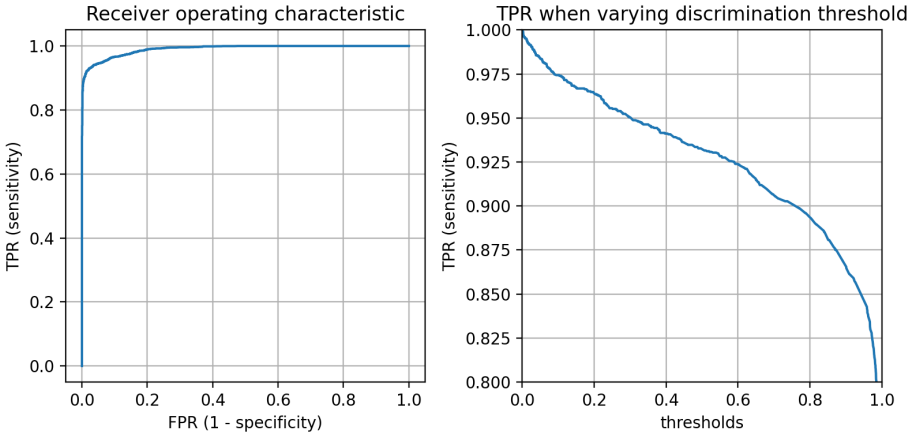


Figure 4.2: Receiver operating characteristic

computed and reported in table 4.2. Video level prediction probabilities are computed averaging the output probability associated with the video frames and, again, a threshold of 0.5 is used to get the predicted class. In both tables, EfficientNet-B1 is chosen for further analysis as it shows better accuracy and more balanced wrong predictions.

Figure 4.2 shows the receiver operating characteristic plot together with a sensitivity plot vs thresholds. The ROC has an AUC score of 0.9905. Beyond this, the plot on the right demonstrates how to improve sensitivity by decreasing the threshold value at expense of specificity. Clearly, the value threshold can be tuned depending on the situation in which the model prediction is used. For example, if negative results avoid the hypothesis of a disease presence for a patient and no further analysis will be done, it would be better to increase the sensitivity; the positiveness of model prediction could be used as the first of other steps to diagnose a disease. On the other hand, if only the model prediction is available and the therapy is too invasive, greatly decreasing the specificity could not be a good choice.

Table 4.3 reports, for each test fold, the validation loss associated and the test accuracy associated with the epoch in which the model training phase is early-

Test fold	Validation loss	Test accuracy	Best epoch
0	0.0647	0.7253	3
1	0.4910	0.9425	3
2	0.0784	0.9875	1
3	0.0030	0.9929	2
4	0.0160	0.9856	1
5	0.0566	0.9612	2
6	0.0783	0.9976	1
7	0.0066	0.9831	4
8	0.0261	0.9974	5
9	0.0007	0.9944	6

Table 4.3: EfficientNet-B1: Training results for different folds

Frame-level results		
Hospital	Accuracy	Confusion matrix
Naples	0.9388	$\begin{bmatrix} 1059 & 69 \\ 79 & 971 \end{bmatrix}$
Florence	0.9904	$\begin{bmatrix} 360 & 0 \\ 10 & 674 \end{bmatrix}$
Milan	0.9469	$\begin{bmatrix} 478 & 2 \\ 64 & 699 \end{bmatrix}$

Video-level results		
Hospital	Accuracy	Confusion matrix
Naples	0.9421	$\begin{bmatrix} 181 & 7 \\ 14 & 161 \end{bmatrix}$
Florence	0.9942	$\begin{bmatrix} 60 & 0 \\ 1 & 113 \end{bmatrix}$
Milan	0.9471	$\begin{bmatrix} 80 & 0 \\ 11 & 117 \end{bmatrix}$

Table 4.4: EfficientNet-B1: aggregate results for different hospitals

stopped i.e. it has the lowest validation loss. The values show relatively high

Hospital \ Class	BEST	RDS
Naples	10	4
Florence	0	3
Milan	1	1

Table 4.5: EfficientNet-B1: Number of different patient's frames wrongly classified

Predicted \ True	BEST	RDS
BEST	475 ± 2	472 ± 4
RDS	360 ± 60	294 ± 82

Table 4.6: EfficientNet-B1: SF index mean and std. dev. for right and wrong predicted class

accuracy scores except for the fold 0, both for testing and validation. It testifies the presence of hard samples in this fold. Table 4.4 reports results divided for each hospital centers in case of frame-level and video-level predictions respectively showing lower results for RDS samples from Milan and all samples from Naples. Table 4.5 reports the number of different patients wrongly classified for each hospital and class: if at least one patient's frame is wrongly classified it is added to the count. The tables show that only 1 RDS patient from Milan has 64 wrongly-classified frames, a patient belonging to the fold 0; this fact proves the low performances obtained on this test fold.

Table 4.6 shows SF index mean and standard deviation for right and wrongly predicted cases of each class: particular interest is for RDS class where the model tends to fail for patients having an intermediate health status whereas it correctly

predicts extreme SF index cases, the same cannot be seen for BEST cases because the SF indexes are clipped for very high values.

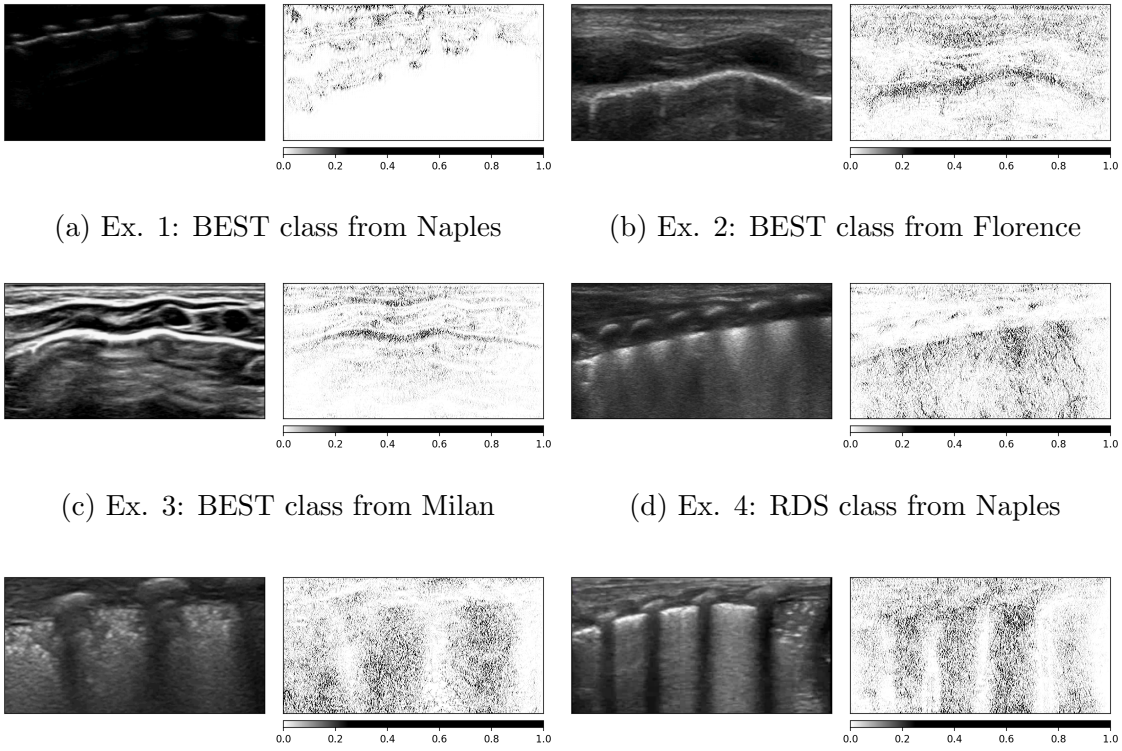
4.3 Features attribution

With the use of Integrated Gradient method, the goal of this section is to get some insight about which pixels influence most in the prediction and understand why the model decides for a particular class. The model should learn to discriminate the image patterns pointed out in section 3.4.1 i.e. the pleural line and the region underneath where a-line and b-line could be present: these are the most important factors that should guide the model prediction. In the beginning, some right prediction attribution will be shown to ensure that the model learned to discriminate the correct features, after that some wrong prediction attributions will be analysed to understand what made the model mislead the prediction.

In following the Figure 4.3 will be analysed representing the feature attribution maps for correct predicted frames for both BEST and RDS classes.

- Figure 4.3a: Model prediction is affected by the pleural line and the region above and below it. The region below shows a slight presence of a-lines and absence of liquid that leads the model to the right prediction of BEST class. Otherwise the region above should not be considered because there is no useful information.

- Figure 4.3b: Here the image can be a little tricky because of the presence of another line above the pleural line caused by the presence of ribs. The model can distinguish between these two lines in fact the attribution associated to the pleural line pixels is stronger.



(a) Ex. 1: BEST class from Naples

(b) Ex. 2: BEST class from Florence

(c) Ex. 3: BEST class from Milan

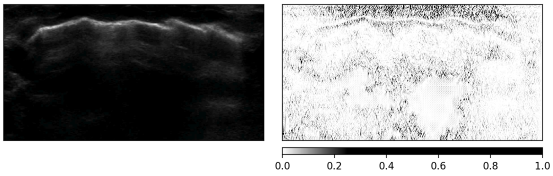
(d) Ex. 4: RDS class from Naples

(e) Ex. 5: RDS class from Florence

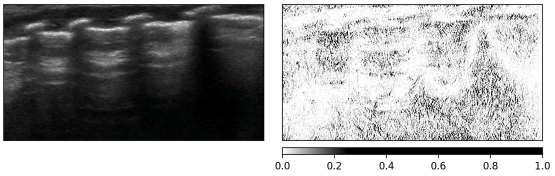
(f) Ex. 6: RDS class from Milan

Figure 4.3: Attribution features maps for right predicted classes.

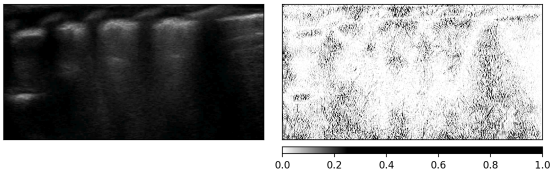
- Figure 4.3c: In the present image there is more contrast, the image is saturated in some regions. The model can still assign strong attribution to the pleural line.
- Figure 4.3d: In this images the presence of ribs is clearly visible. The model can understand where the pleural line is and assigns a strong attribution to the pixels underneath. The detection of b-lines leads to the RDS class prediction.
- Figure 4.3e: This image is more zoomed than the others. The model recognise the presence of b-lines.
- Figure 4.3f: The pleural line is not totally smooth. The b-spline artifact are



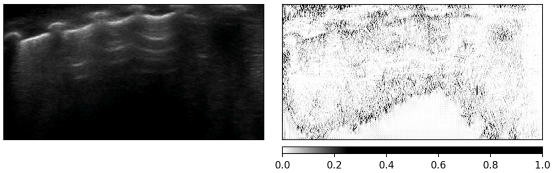
(a) Ex. 1: BEST class from Naples



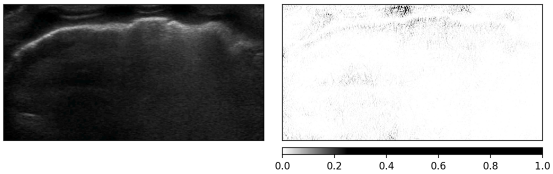
(b) Ex. 2: BEST class from Naples



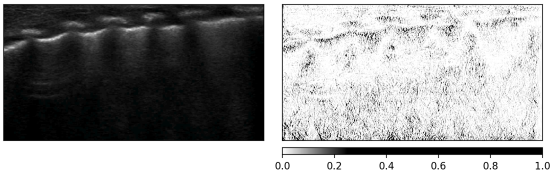
(c) Ex. 3: BEST class from Naples



(d) Ex. 4: RDS class from Naples



(e) Ex. 5: RDS class from Naples



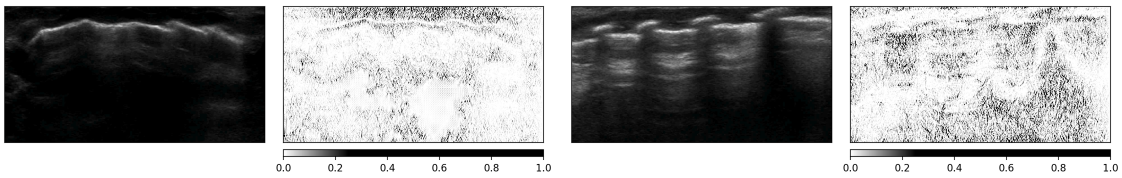
(f) Ex. 6: RDS class from Naples

Figure 4.4: Attribution features maps for wrong predicted classes from Naples hospital.

clearly distinguished by the model.

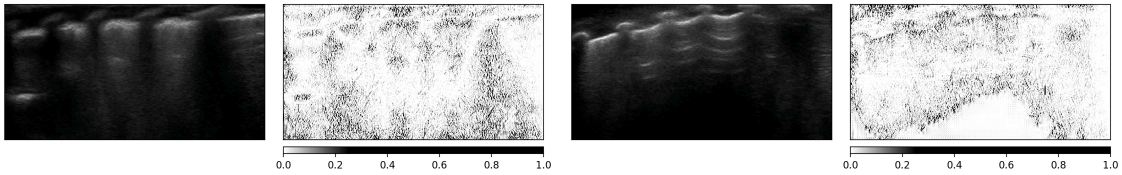
In figure 4.5 there are some frames wrongly classified by the model from Naples. From table 4.4 model have a high probability of going wrong with these patients. Here are some comments:

- Figure 4.5a: Only the edges of the pleural line are detected. The model assign a strong attribution to the pixels above the pleural line.
- Figure 4.3b, 4.3c, 4.3d, 4.3f: The pleural line and a-lines/b-lines are detected but the model assign a strong attribution in the bottom of the image that



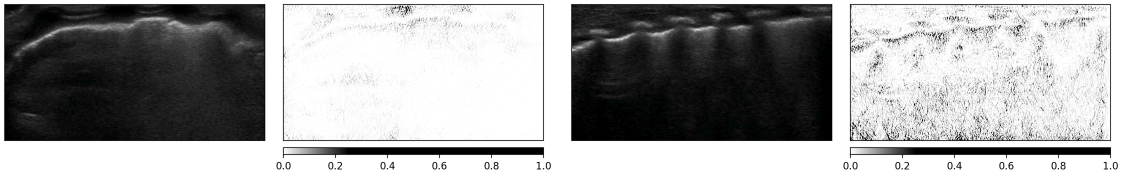
(a) Ex. 1: BEST class from Naples

(b) Ex. 2: BEST class from Naples



(c) Ex. 3: BEST class from Naples

(d) Ex. 4: RDS class from Naples



(e) Ex. 5: RDS class from Naples

(f) Ex. 6: RDS class from Naples

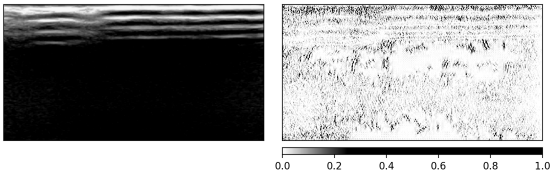
Figure 4.5: Attribution features maps for wrong predicted classes from Naples hospital.

should carry no useful information.

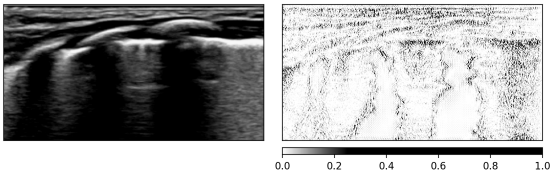
- Figure 4.3e: The pleural line is detected but the model assign a strong attribution to some pixel in the top.

In figure 4.6 there are wrong classified frames for RDS class from Milan. The table 4.5 shows that all the misclassification for this case is associated with the same patient.

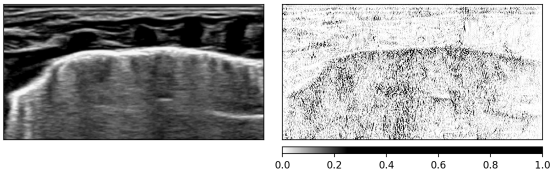
- Figure 4.6a: No meaning frame. It could be due to the initial insertion phase of the LUS instrument.



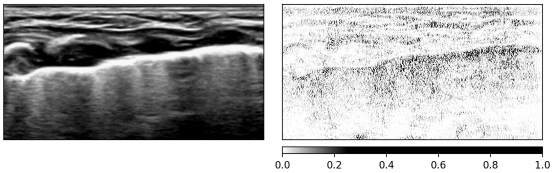
(a) Ex. 1: RDS class from Milan



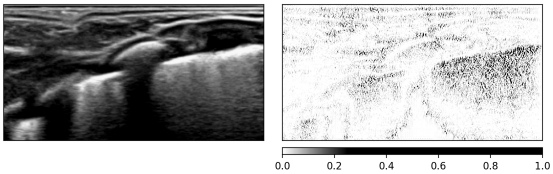
(b) Ex. 2: RDS class from Milan



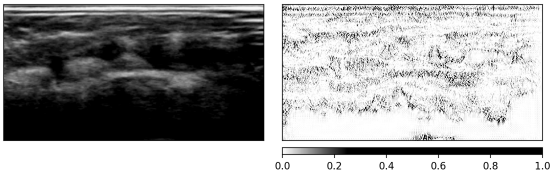
(c) Ex. 3: RDS class from Milan



(d) Ex. 4: RDS class from Milan



(e) Ex. 5: RDS class from Milan

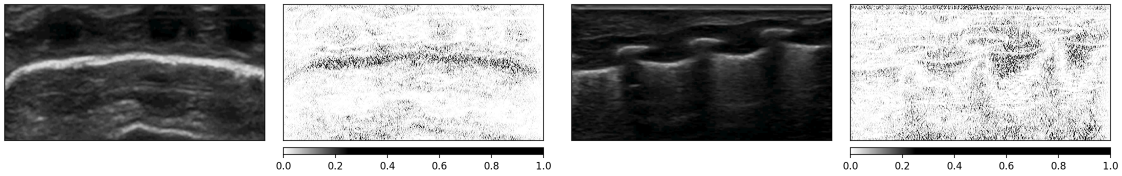


(f) Ex. 6: RDS class from Milan

Figure 4.6: Attribution features maps for wrong predicted classes from Milan hospital. All the frames are associated to the same patient's videos

- Figure 4.6b, 4.6d: The pleural line is detected but the model associate a strong attribution to the pixels above the pleural line.
- Figure 4.3c: The pleural line is detected and the model associate a strong attribution the liquid below it. It is hard to understand why the model goes wrong for this frame.
- Figure 4.3e, 4.3f: These frames are in general hard to interpret.

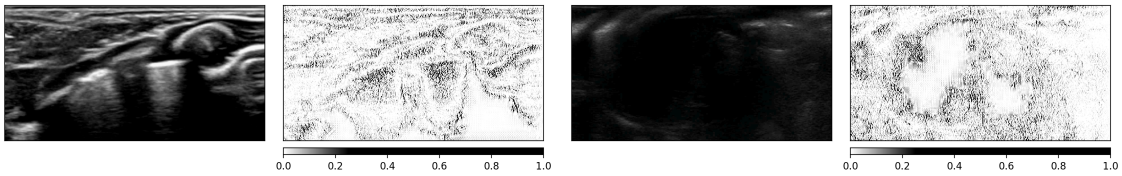
In figure 4.7 there are misclassification attribution feature maps for patients from Florence.



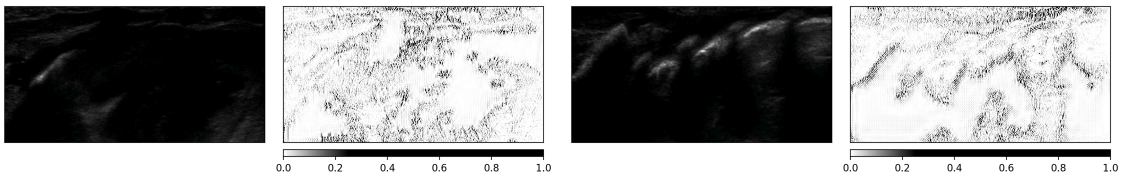
(a) Ex. 1: RDS class from Florence (b) Ex. 2: RDS class from Florence

Figure 4.7: Attribution features maps for wrong predicted RDS class from Florence hospital.

In figure 4.8 there are misclassification for frames very hard to interpret. The presence of this kind of frame could be due to the insertion phase of the instrument.



(a) Ex. 1: RDS class from Milan (b) Ex. 2: RDS class from Naples



(c) Ex. 3: BEST class from Naples (d) Ex. 4: BEST class from Naples

Figure 4.8: Attribution features map for wrong predicted class. Images are hard to classify

Chapter 5

Conclusions

Thanks to the availability of 745 labeled LUS videos of 72 patients from different hospital centers, it was possible to build an algorithm based on Deep Learning for automatically assessing the presence of RDS in patients. Some networks were experimented showing, for all of them, relative high scores for both frame-level and video-level predictions. The results show that EfficientNet-B1 performs better than the other networks, achieving a video-level accuracy of 95.57%, a sensitivity of 93.76% and a specificity of 97.87%, all of these metrics are computed with a simple classification threshold of 0.5. The threshold can be adapted depending on the importance that sensitivity and specificity can assume, based on clinical situation and resources in the specific case.

The model predictions were then analysed per single class (using confusion matrix) and per different hospital centers. It turned out that, for the majority, the model wrong classified patients from Naples of both classes. Low scores are also present for the RDS class from Milan but this misclassification was caused by a single patient, probably due to the quality and the high contrast present in the videos.

No hand-crafted features were built, on the contrary, the neural network learned

the discriminant pattern present in the LUS image by itself. This aspect was investigated with the use of a features attribution technique that allowed to detect, for each image, the pixels which contribute most to the final prediction. As expected, the model base its prediction mainly on the pixels associated with the region of the pleural line and below it, where a- or b-lines artifacts are present. Besides the right prediction, particular attention was given to wrong classified frames in order to understand the bad decision made by the model. It turned out that the reasons are multiple, i.e.:

- Some images have no interpretable information. These are probably frames associated to the initial insertion phase in the region of interest.
- Some images are clearly hard to interpret and classify.
- Strong attribution to pixel above the pleural line which should give no useful information.
- Strong attribution in the bottom of the image where in the majority of the image there are no information, a-line and b-lines are usually extinguished in that region.

While the first two points are dataset failures that could be addressed by a cleaning procedure, the second two are model failures. To overcome to the problem highlighted in the third point, a pleural line model detector can be build to crop the region above it and feed a model similar to the one built here, using the resulting image.

A factor that was completely neglected is the temporal correlation between video frames. In this work, the video-level prediction was carried out predicting frame by frame separately and then averaging the scores to obtain the video prediction. The temporal information could carry, as the experts says, useful information for

the detection of the lung status. Such temporal correlation can be exploited using video-classification deep learning networks that works using samples with one more input dimension i.e. the temporal dimension.

Bibliography

[1] Michael Albaum et al. "Interobserver Reliability of the Chest Radiograph in Community-Acquired Pneumonia". In: *Chest* 110 (Aug. 1996), pp. 343–350. DOI: 10.1378/chest.110.2.343.

[2] Jannis Born et al. "POCOVID-Net: Automatic Detection of COVID-19 From a New Lung Ultrasound Imaging Dataset (POCUS)". In: *ArXiv* (Apr. 2020).

[3] Francesco Corradi et al. "Quantitative Analysis of Lung Ultrasonography for the Detection of Community-Acquired Pneumonia: A Pilot Study". In: *BioMed Research International* 2015 (Feb. 2015). DOI: 10.1155/2015/868707.

[4] Malena Correa et al. "Automatic classification of pediatric pneumonia based on lung ultrasound pattern recognition". In: *PLOS ONE* 13 (Dec. 2018), e0206410. DOI: 10.1371/journal.pone.0206410.

[5] D. Doran, S.C. Schulz, and T. R. Besold. "What Does Explainable AI Really Mean? A New Conceptualization of Perspectives". In: *CEUR Workshop Proceedings* 2071 (Mar. 2018). URL: https://openaccess.city.ac.uk/id/eprint/18660/.

[6] Deepak Garg et al. "Clinical Value of Chest Computerized Tomography Scans in Patients Admitted With Pneumonia". In: *Journal of Hospital Medicine* 9 (July 2014). DOI: 10.1002/jhm.2190.

[7] Michela Gravina et al. “Deep learning in the ultrasound evaluation of neonatal respiratory status”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021, pp. 10493–10499. DOI: 10.1109/ICPR48806.2021.9412732.

[8] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[9] K. Hornik, M. Stinchcombe, and H. White. “Multilayer Feedforward Networks Are Universal Approximators”. In: *Neural Netw.* 2.5 (July 1989), pp. 359–366. ISSN: 0893-6080.

[10] Andrew Howard et al. “Searching for MobileNetV3”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 1314–1324. DOI: 10.1109/ICCV.2019.00140.

[11] Andrew G. Howard et al. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”. In: *ArXiv* abs/1704.04861 (2017).

[12] Diederik Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* (Dec. 2014).

[13] Daniel Lichtenstein. “Ultrasound in the management of thoracic disease”. In: *Critical care medicine* 35 (June 2007), S250–61. DOI: 10.1097/01.CCM.0000260674.60761.85.

[14] Silvia Magrelli et al. “Classification of Lung Disease in Children by Using Lung Ultrasound Images and Deep Convolutional Neural Network”. In: *Frontiers in Physiology* 12 (Aug. 2021). DOI: 10.3389/fphys.2021.693448.

[15] Lewis Mandell et al. "Infectious Diseases Society of America/American Thoracic Society Consensus Guidelines on the Management of Community-Acquired Pneumonia in Adults". In: *Clinical infectious diseases : an official publication of the Infectious Diseases Society of America* 44 Suppl 2 (Apr. 2007), S27–72. DOI: 10.1086/511159.

[16] T Meehan et al. "Quality of Care, Process, and Outcomes in Elderly Patients with Pneumonia". In: *JAMA : the journal of the American Medical Association* 278 (Jan. 1998), pp. 2080–4. DOI: 10.1001/jama.278.23.2080.

[17] Francesco Raimondi et al. "Visual assessment versus computer-assisted gray scale analysis in the ultrasound evaluation of neonatal respiratory status". In: *PLOS ONE* 13 (Oct. 2018), e0202397. DOI: 10.1371/journal.pone.0202397.

[18] Farheen Ramzan et al. "A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer's Disease Stages Using Resting-State fMRI and Residual Neural Networks". In: *Journal of Medical Systems* 44 (Dec. 2019). DOI: 10.1007/s10916-019-1475-2.

[19] Angelika Reissig et al. "Lung Ultrasound in the Diagnosis and Follow-up of Community-Acquired Pneumonia". In: *Chest* 142 (June 2012). DOI: 10.1378/chest.12-0364.

[20] Subhankar Roy et al. "Deep Learning for Classification and Localization of COVID-19 Markers in Point-of-Care Lung Ultrasound". In: *IEEE Transactions on Medical Imaging* PP (May 2020). DOI: 10.1109/TMI.2020.2994459.

[21] Mark Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[22] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

[23] Carolyn Sofka. "Ultrasound Examination of the Lungs in the Intensive Care Unit". In: *Ultrasound Quarterly* 26 (June 2010), p. 116. DOI: 10.1097/01.ruq.0000377228.49570.4f.

[24] Gino Soldati et al. "On the Physical Basis of Pulmonary Sonographic Interstitial Syndrome". In: *Journal of Ultrasound in Medicine* 35 (Aug. 2016). DOI: 10.7863/ultra.15.08023.

[25] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic Attribution for Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 3319–3328. URL: https://proceedings.mlr.press/v70/sundararajan17a.html.

[26] Hannu Syrjälä et al. "High-Resolution Computed Tomography for the Diagnosis of Community-Acquired Pneumonia". In: *Clinical infectious diseases : an official publication of the Infectious Diseases Society of America* 27 (Sept. 1998), pp. 358–63. DOI: 10.1086/514675.

[27] Mingxing Tan and Quoc Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 6105–6114. URL: https://proceedings.mlr.press/v97/tan19a.html.

[28] Giovanni Volpicelli, Fernando Silva, and Michael Radeos. "Real-time lung ultrasound for the diagnosis of alveolar consolidation and interstitial syndrome

in the emergency department". In: *European journal of emergency medicine : official journal of the European Society for Emergency Medicine* 17 (Apr. 2010), pp. 63–72. DOI: 10.1097/MEJ.0b013e3283101685.