Deep Learning for Automatic Classification of Multi-Modal Information Corresponding to Chest Radiology Reports

Nelson Filipe Rodrigues Nunes

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Bruno Emanuel da Graça Martins
Prof. Mário Jorge Costa Gaspar da Silva

Examination Committee

Chairperson: Prof. Alexandre Paulo Lourenço Francisco
Supervisor: Prof. Bruno Emanuel da Graça Martins
Member of the Committee: Prof. João Miguel da Costa Magalhães

October 2019
Acknowledgments

I would like to thank my supervisors, Prof. Bruno Martins and Prof. Mário J. Silva, for their constant guidance during the development of my M.Sc. research project. Always available to share their knowledge, to read preliminary versions of my dissertation, and to give me advice on how I could improve the results of my research.

I’m very much obliged to INESC-ID for providing me and my research colleagues a room where we could gather and work together on our projects.

I also express my gratitude to Luz Saúde for providing me with a stimulating working environment and the necessary hardware support. Special mention to Nuno André da Silva, for making this collaboration possible and always trying to make it as successful as possible.

To all my close friends, I am truly grateful for always being available along the way, either for academic or friendship reasons. This 5-year journey would not have been the same without you.

To my family, who have always provided me with emotional and financial support I needed while studying in Lisbon. Dedicated to the memory of my mother who always believed in my value. You are gone, but your belief in me has made this journey possible. I hope you are proud of me.
Abstract

Hospitals frequently use chest X-ray images, which can provide a detailed view of a patient's thorax, as a radiology examination technique for the diagnosis of cardiac and pulmonary diseases. However, the interpretation of these images, which often results in a free-text radiology report and/or a classification, requires specialized medical professionals, leading to high labor costs and waiting lists. Automatic inference of thoracic diseases from the results of chest radiography exams is still a challenging task, although several recent studies have proposed machine learning methods for this particular task. Deep neural network architectures can contribute to a more efficient indexing of radiology exams (e.g., associating the data to diagnostic codes), providing inexpensive, accurate, and interpretable classification results that can guide the domain experts. This dissertation addresses this task by introducing a novel multi-modal end-to-end neural network for automatically classifying chest radiology exams, combining state-of-the-art pre-trained Convolutional Neural Networks (CNNs) architectures, together with Recursive Neural Networks (RNNs) units, with clinical pretrained word embeddings and attention units. The experimental results show interesting patterns, e.g. validating the high performance of the individual components, particularly when pre-training these components of the model with large pre-existing datasets, and showing promising results for the multi-modal processing of radiology examination data, confirming the usefulness of using multi-modal data. The best model achieves 0.949 and 0.987 in macro and micro average AUROCs, respectively.

Keywords

Classification of Chest Radiology Exams, Deep Learning, Learning from Multi-Modal Data
**Resumo**

Hospitais recorrem frequentemente a imagens de raio X, que fornece uma vista detalhada do tórax do paciente, como um método de radiologia para o diagnóstico de doenças cardíacas e pulmonares. Contudo, a interpretação destas imagens, que frequentemente resultam num relatório de radiologia em texto livre e/ou numa classificação, necessita de profissionais médicos especializados, originando altos custos de mão-de-obra e longas listas de espera. A inferência automática de doenças torácicas de radiografias ao tórax é ainda uma tarefa desafiante, embora alguns estudos recentes tenham proposto métodos de aprendizagem automática para esta específica tarefa. Esta dissertação aborda esta tarefa, introduzindo um novo método multimodal para classificação automática de exames de radiologia torácica, combinando recentes redes neurais convolucionais pré-treinadas, juntamente com redes neurais recursivas com word embeddings pré-treinados e atenção neuronal. Os resultados experimentais indicam padrões interessantes, e.g. validam o alto desempenho de cada componente individual, particularmente quando são pre-treinadas usando grandes datasets pre-existentes, e assinalam resultados promissores para o processamento multimodal de dados de exames de radiologia, confirmando a utilidade do uso de data multimodal. O melhor modelo alcançou 0.949 e 0.987 em termos da média macro e micro das AUROCs, respectivamente.

**Palavras Chave**

Classificação de Relatórios de Radiologia, Classificação de Radiografias torácicas, Aprendizagem com Redes Neuronais Profundas, Aprendizagem com dados multimodais
## Contents

1 Introduction ................................................. 2  
1.1 Objectives ............................................. 4  
1.2 Methodology .......................................... 4  
1.3 Contributions ......................................... 5  
1.4 Thesis Outline ......................................... 6  

2 Concepts and Related Work .............................. 7  
2.1 Introduction to Deep Learning ......................... 8  
2.1.1 The Perceptron .................................... 8  
2.1.2 The Multi-Layer Perceptron ....................... 9  
2.1.3 Convolutional Neural Networks .................... 12  
2.1.4 Recurrent Neural Networks ....................... 13  
2.2 Deep Learning for Text Classification .................. 17  
2.3 Deep Learning for Image Classification ................. 20  
2.4 Deep Learning for Medical Text Reports Classification .... 25  
2.5 Deep Learning for Medical Image Classification ........ 29  
2.5.1 Classification of Radiology Reports ................ 29  
2.5.2 Overview ........................................ 34  

3 The Multi-modal Deep Neural Model for X-ray report classification 36  
3.1 The Multi-modal Deep Neural Architecture ............... 37  
3.1.1 Representing the Textual Inputs .................... 38  
3.2 Modeling Visual Contents ................................ 42  
3.3 Per-Modality Model Pre-Training ....................... 45  
3.4 Summary ............................................. 46  

4 Experimental Evaluation ................................. 47  
4.1 Datasets and Experimental Methodology .................. 48  
4.2 Experimental Results .................................. 49  
4.2.1 Radiography Classification Task ................... 49
4.2.2 Radiology Reports Classification Task ............................................. 50
4.2.3 Radiology Exams Multi-Modal Classification Task .......................... 52
4.3 Interpreting Results by Visualizing mechanisms ................................. 53
4.4 Summary .......................................................................................... 55

5 Conclusions and Future Work .............................................................. 56
5.1 Summary of Main Conclusions .......................................................... 57
5.2 Future Work .................................................................................... 58
List of Figures

1.1 Multi-modal chest radiology exam from Open-i (Demner-Fushman et al., 2016a) dataset classified with the MeSH label: Pneumonia. ........................................... 3

2.1 The perceptron model (left) and the multi-layer perceptron model (right). ......................... 8
2.2 Convolutional operation with 1 kernel of size 3x3, without padding (left), and an application of 10 different filters, using padding, to create an stack of feature maps (right). ........... 12
2.3 Diagram with the LeNet5 architecture, adapted from Lecun et al. (1998). ......................... 13
2.4 Convolution and pooling over a sentence, adapted from Goldberg (2017). ....................... 14
2.5 Graphical representations of an RNN, recursive (left) and unrolled (right), adapted from Goldberg (2017). .......................................................... 14
2.6 The hierarchical attention network from Yang et al. (2016). ........................................ 17
2.7 Diagrams illustrating Context-Word Region Embedding (left) and Word-Context Region Embedding (right), both inspired on figures from the original paper by Qiao et al. (2018). . 19
2.8 Diagram with the VGG-11 architecture, adapted from Simonyan and Zisserman (2014). 21
2.9 Image Transformer, as described in the original paper by Parmar et al. (2018). ............... 22
2.10 Encoder used for both encoding the fields and documents, in the hierarchical network from Duarte et al. (2018). .................................................... 26
2.11 The memory augmented neural network architecture described by Rios and Kavuluru (2018). .......................................................... 27
2.12 Framework of cluster-based semantic expansion model described by Shen et al. (2018). 28
2.13 TieNet architecture as described by Wang et al. (2018). ............................................ 30
2.14 Architecture of the multi-label fine-grained network from Ge et al. (2018). ................... 33

3.1 General overview on the proposed neural network architecture. The blue boxes represent the components of our general architecture for which we test using different pre-trained approaches. ........................................... 37
3.2 ELMo’s architecture illustrating the creation of contextual embeddings through BiLM. ... 39
3.3 BERT’s architecture illustrating the creation of contextual embeddings though the MLM task. ................................................................. 40
3.4 Illustrations describing LSTM and mLSTM recurrent cells. ......................... 41
3.5 ResNet-50 convolutional neural network architecture. ................................ 42
3.6 The DenseNet convolutional neural network architecture. .............................. 43
3.7 The Dual Path Network block. ............................................................ 44
3.8 Illustration of the depthwise separable convolution block. .............................. 44
3.9 The mobile inverted bottleneck convolution (MBConv) block, enhanced with the squeeze-and-excitation mechanism. ............................ 45

4.1 Plots comparing the classification performance of the (ii) experimental setting. ..... 51
4.2 A comparison of classification performance with different word embeddings. .......... 52
4.3 A comparison of classification performance with different modality and pre-training settings models. .................................................. 53
4.4 Multi-modal visualization mechanisms computed for three example instances from the Open-i test split. ................................................. 54
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Comparing a capsule vs a traditional neuron.</td>
<td>24</td>
</tr>
<tr>
<td>2.2</td>
<td>Summary of previous work presented in this report.</td>
<td>35</td>
</tr>
<tr>
<td>4.1</td>
<td>Statistical characterization of the datasets used in the experiments.</td>
<td>48</td>
</tr>
<tr>
<td>4.2</td>
<td>Results for the MIMIC-CXR/CheXpert dataset experimental settings.</td>
<td>50</td>
</tr>
<tr>
<td>4.3</td>
<td>Results for the MIMIC-III dataset experimental settings.</td>
<td>50</td>
</tr>
<tr>
<td>4.4</td>
<td>Results for the Open-i dataset experimental settings.</td>
<td>53</td>
</tr>
</tbody>
</table>
1 Introduction

Contents

1.1 Objectives ................................................................. 4
1.2 Methodology .............................................................. 4
1.3 Contributions ............................................................ 5
1.4 Thesis Outline ............................................................ 6
Chest radiography is globally the most common medical imaging examination. The interpretation of chest X-rays, involving specialized medical professionals, is critical for the screening, diagnosis, and management of many diseases. In recent years, leveraging the availability of large datasets, several authors have explored deep learning methods for the automated interpretation of chest radiography images, arguing that having automatic methods working at the level of practicing radiologists can provide substantial benefits in many medical settings, from improving workflow prioritization and assisting clinical coders, to supporting clinical decisions and large-scale screening initiatives.

Hospitals nowadays invest in clinic coders to abstract relevant information from the non-structured data (e.g. medical images or narrative descriptions of clinical encounters) present in clinical records of patients, deciding which diagnosis and procedures to codify under the patterns of standardized taxonomies such as: the International Statistical Classification of Diseases and Related Health Problems (ICD-10), and Medical Subject Headings (MeSH) (i.e. thesaurus for indexing articles for PubMed). Coding requires the knowledge of experts, with high experience in clinical terminology and codification rules. This requirement leads to high labour costs, combined with the possibility of human error (i.e., over or under-coding) and long waiting lists.

Accurate models for data classification (e.g., for assigning images and/or text (Duarte et al., 2018; Laserson et al., 2018) to codes within standard clinical taxonomies) can significantly speed-up the coding process, decrease labor costs, increase coding consistency, and standardize the coding of legacy data. The picture archiving and communication systems (PACSs) and the radiology information systems (RISs) of many clinics/hospitals have large amounts of chest radiology images, stored together with free-text radiology reports summarizing their interpretation. Figure 1.1 provides an illustration.

Many previous studies have, for instance, reported good results with methods based on Convolutional Neural Networks (CNNs) (Baltruschat et al., 2018), in the task of coding chest radiography images.
according to classes conforming to the Fleischner Society’s recommended glossary of terms for thoracic imaging (Hansell et al., 2008). Thus, we argue that the combination of state-of-the-art methods for image analysis (Khan et al., 2018) and natural language processing (Goldberg, 2017) can be particularly interesting in the context of coding legacy data, directly considering multi-modal contents (i.e., images together with existing text reports) as a way to improve the classification results.

1.1 Objectives

This dissertation presents a method for the automatic classification of chest radiology exams, a multi-label prediction according to 14 thoracic diseases classes. The research main’s objective was to study how the multi-modal radiology exams data can be classified using a deep neural network, leveraging state-of-the-art models proposed not only for image classification but also for natural language processing (NLP). This study also aimed to verify the impact of using multi-modal data, and leveraging single modality pre-existing datasets to pre-train individually each modality path of the architecture. Finally, the last objective focused on the explainable Artificial Intelligence (AI) field, providing distinct interpretability views, such that the predictions of the method can be understood by human experts.

1.2 Methodology

Preliminary, a general revision was conducted on the areas of single and multi modality image and text classification tasks, focusing on biomedical domains such as: thorax radiography classification, and biomedical text coding classification. Designed the general architecture for the multi-label multi-modal deep learning architecture and implemented the different sub-models leveraging the keras\textsuperscript{1} deep learning library, and the source code supporting our experiments was also made available\textsuperscript{2} on GitHub.

Our experiments relied on four different datasets, including two datasets containing frontal chest X-ray images labeled according to 14 observation classes (i.e., CheXpert (Irvin et al., 2019) and MIMIC-CXR (Johnson et al., 2019)), one dataset of full-text radiology reports (i.e., a subset of the data from MIMIC-III (Edward William Johnson et al., 2016) with the ICD labels converted to the same 14 observation classes), and the Open-i (Demner-Fushman et al., 2016a) multi-modal radiography dataset containing frontal chest X-ray images together with full-text reports, also labeled according to the same 14 observation classes. Taking into account the multi-label nature of the datasets, we considered stratification when dividing the data (i.e., the merging of MIMIC-CXR/CheXpert, and the subsets from MIMIC-III and open-i) into training (64%), validation (16%), and testing (20%) splits. The stratification method balances

\textsuperscript{1}http://keras.io/
\textsuperscript{2}http://github.com/nfrn/Multi-Modal-Classification-of-Radiology-Exams
the assignment of instances into splits, attending to the distribution of single classes and multi-class pairs (Sechidis et al., 2011; Szymański and Kajdanowicz, 2017).

Model training when leveraging the text datasets was made with batches of 64 instances, while models leveraging image and multi-modal inputs were trained with batches of 16 instances, in both cases leveraging back-propagation together with the Adam optimization method (Kingma and Ba, 2015). Chapter 4 goes into further detail regarding the experimental evaluation employed in this research.

To assess the quality of the predictions, we used the following metrics: accuracy (i.e., average number of correct labels per instance), coverage error (CE) (i.e., how many labels, ranked according to prediction scores, need to be checked to cover all the true labels), label ranking average precision (LRAP), and micro- and macro-averaged scores for precision, recall, F1, and areas under ROC curves (AUROCs).

### 1.3 Contributions

The main ideas and contributions of this M.Sc thesis work can be summarized as follows:

- A novel multi-modal deep learning approach to classify chest radiology exams is proposed. For processing image data, the convolutional neural networks (CNNs) experimented were: Dual Path Networks (DPNs) (Chen et al., 2017) with random initialized weights, and Residual Neural Networks (ResNets) (He et al., 2016), Densely Connected Convolutional Networks (DenseNets) (Huang et al., 2017), and EfficientNet (Tan and Le, 2019) architectures pre-trained in the ImageNet (Deng et al., 2009) image dataset. For modeling free-textual data, three different biomedical pre-trained word embeddings were tested, namely BioWordVec (Chen et al., 2018), BioELMo (Jin et al., 2019), and BioBERT (Lee et al., 2019), together with Bidirectional Multiplicative Long Short-Term Memory (bi-mLSTM) units (Krause et al., 2017) for creating intermediate representations, which are finally combined through a multi-head attention mechanism (Vaswani et al., 2017). The complete model concatenates the representations from the convolutional and recurrent parts, and it can be trained end-to-end (or fine-tuned, in case we leverage pre-trained components) from data combining both modalities (e.g. using the standard back-propagation algorithm together with an optimization method such as Adam (Kingma and Ba, 2015)).

- One of the main contributions of this thesis consists in quantifying the impact of leveraging single modality pre-existing datasets to pre-train individually each modality path of the architecture. The complete multi-modal model with pre-trained weights (i.e. image processing path pre-trained with CheXpert and MIMIC-CXR, and text modeling path pre-trained with MIMIC-III dataset) and fine-tuned with the open-i training split, achieved a very high performance in terms of the differ
ent metrics. Pre-training, in particular, contributed significantly to the overall performance of the complete model.

- In connection to the classification results, the proposed model also allows us to explore two distinct interpretability views. Using a gradient-weighted class activation map (Grad-CAM), it is possible to visualize which areas of the X-ray were more important to the classification decision (Selvaraju et al., 2017). Simultaneously, using the weights from the multi-head attention mechanism, it is possible to visualize which words in the full-text report were more important to the classification.

- During the development of this research project, two articles were produced, namely: A Multi-modal Deep Learning Method for Classifying Chest Radiology Exams (Nunes et al., 2019) published in the EPIA 2019 conference proceedings and awarded with “Best Paper” award, and Deep Learning for Automatic Classification of Multi-Modal Information Corresponding to Chest Radiology Reports, also planning to be published in the near future.

### 1.4 Thesis Outline

The remainder of this dissertation is organized as follows: Chapter 2 presents fundamental concepts and related work that support the theme of this dissertation. Chapter 3 describes the multi-modal architecture developed and details the methods used. Then, Chapter 4 details the datasets, experimental methodology and evaluation of the proposed approach. Lastly, Chapter 5 summarizes the main conclusions of this work, and points to possible future contributions for future work.
## Concepts and Related Work

### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Introduction to Deep Learning</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Deep Learning for Text Classification</td>
<td>17</td>
</tr>
<tr>
<td>2.3</td>
<td>Deep Learning for Image Classification</td>
<td>20</td>
</tr>
<tr>
<td>2.4</td>
<td>Deep Learning for Medical Text Reports Classification</td>
<td>25</td>
</tr>
<tr>
<td>2.5</td>
<td>Deep Learning for Medical Image Classification</td>
<td>29</td>
</tr>
</tbody>
</table>
This section clarifies some of the fundamental concepts required for the reader to fully comprehend the dissertation and overviews relevant related work, starting with general text and image classification, and then describing studies related to the classification of radiology reports and biomedical text.

Section 2.1 introduces the field of deep learning explaining multi-layer perceptrons, convolutional and recurrent neural networks. Then, Section 2.2 and Section 2.3 overview relevant related work using deep learning for the Text and Image classification tasks, respectively. Lastly, Section 2.2 and Section 2.3 describe studies related to the classification of radiography images and bio-medical text in, accordingly.

2.1 Introduction to Deep Learning

2.1.1 The Perceptron

Deep learning has a long history. Frank Rosenblatt’s paper on the perception model Rosenblatt (1957) was the precursor to modern neural networks, describing an electronic device inspired in biological principles that exhibits an ability to learn.

As exemplified in Figure 2.1 (left), in a perceptron, a positive weight represents an excitatory connection, while a negative weight an inhibitory one. The output is one if the weighted sum of the inputs is above a threshold, and zero otherwise. By varying the weights, we can change the function that the perceptron computes. The learning mechanism relies on increasing and decreasing the weights of the active inputs when the model misclassifies a positive or negative example, respectively. Although a simple model, only capable of classifying linearly separable inputs, the perceptron was able to recognize printed letters and speech sounds. Mathematically, the perceptron model can be written as shown in the following equation:

\[
y = f(x) = g \left( \sum_{i=1}^{n} x_i w_i + b \right) = g(x \cdot w + b)
\]  

(2.1)

, where \(y\) denotes the output prediction, \(x = (x_1, \ldots, x_n)\) refers to the vector of input features, \(w\) is

![Figure 2.1: The perceptron model (left) and the multi-layer perceptron model (right).]
the vector of weights, \( b \) is a bias term, and \( g(\cdot) \) is the activation function (e.g., the original paper used a binary step function for the binary classification task, although nowadays a logistic sigmoid or an hyperbolic tangent are frequently used as activation functions within more complex networks, due to their nonlinear nature). Given a training set of inputs \( x \) together with the corresponding outputs \( y \), training the perceptron corresponds to adapting all the weights and biases (e.g., the parameters \( w \) and \( b \), expressed in the Equation 2.1) to their optimal values. The equation with optimal parameters is referred to as \( f^*(x) \).

Regarding extensions to the perceptron model, we have for instance that non-linearly separable patterns can be learned by layers of interconnected neurons (e.g. multi-layer perceptrons). In this case, the learning mechanism is required to compute the change in the weights of the hidden layers, noting that every hidden neuron can influence the output by multiple paths. In multi-layer perceptrons, training is achieved through a training procedure known as backpropagation, in combination with gradient descent optimization of the parameters. The next subsections detail these approaches. multi-layer perceptrons are frequently also referred to as feed-forward networks, where information flows through the function being evaluated from \( x \), through the intermediate computations used to define \( f \), and finally to the output \( y \).

### 2.1.2 The Multi-Layer Perceptron

Although these single neural network nodes have a limited mapping power, we can stack these nodes as the primary building block of more complex models. These multi-layered perceptrons can be seen as nested composite functions, as exemplified in Figure 2.1, for the case of a feed-forward network, with a single hidden layer:

\[
y = f(x) = g(g'(x \cdot A + a) \cdot B + b)
\]  

(2.2)

In Equation 2.2, \( A \) and \( B \) represent the weights of the first and second layer, respectively. The functions \( g'(\cdot) \) and \( g(\cdot) \) both denote an element-wise non-linearity, resulting of the activation functions associated to nodes in the hidden and out layers of the network.

During neural network training, we drive \( f(x) \) to match \( f^*(x) \). The training data has examples of \( x \) accompanied by a label \( y \approx f^*(x) \), specifying the value that the output layer must produce for each value \( x \) of the input layer. The model must decide on how to use the layers to produce the desired output (e.g., the approximation of \( f^* \)).

In most cases, neural networks are trained by using iterative gradient-based optimizers, that drive the cost function to lower values. In order to apply gradient-based learning, we must first choose a cost function. A cost function is a measure of similarity between the true distribution and the predicted distribution (e.g., mean squared error for regression problems or a cross-entropy for classification). Gradient
descent is then used to minimize the cost function, by updating the parameters in the opposite direction of the gradient.

There are several variants of gradient descent, and the amount of data required to compute the gradient of the cost function differentiates between these variants. Considering the size of the training dataset, a trade-off between the accuracy of the parameter update and the time required to compute it is often required.

Batch gradient descent, also known as vanilla gradient descent, uses the entire training dataset, to compute the gradient of the objective function $J$, regarding the parameters $\theta$, and the learning rate $\eta$.

$$
\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta)
$$

(2.3)

In order to compute just one update, this method has to compute the gradient of all the dataset. Depending on the amount of data, this can be very time consuming and therefore is only viable for datasets that can fit in memory. Moreover, this variant cannot update the model online with new live examples. On the other hand, batch gradient descent is guaranteed to converge to the global minimum for convex error surfaces, and to a local minimum for non-convex surfaces.

Stochastic gradient descent computes and updates the parameters for each training example $x^{(i)}$ and correspondent label $y^{(i)}$:

$$
\theta = \theta - \eta \cdot \nabla_{\theta} J \left( \theta; x^{(i)}; y^{(i)} \right)
$$

(2.4)

This approach provides better computational efficiency and the ability to learn online. One characteristic of this variant is that it often performs high variance updates that lead the objective function to fluctuate heavily. This property causes this variant not to guarantee convergence. However, if we lower the value of the learning rate, this variant presents the same convergence behaviour of batch gradient descent, almost certainly converging to a local minimum for non-convex optimization and to a global minimum for the convex one.

Mini-batch gradient descent combines the best of both approaches and performs an update for every mini-batch of $n$ training examples:

$$
\theta = \theta - \eta \cdot \nabla_{\theta} J \left( \theta; x^{(i:i+n)}; y^{(i:i+n)} \right)
$$

(2.5)

Choosing a proper learning rate or a suitable learning rate schedule is an fundamental decision. Different types of gradient descent optimization algorithms have been proposed to address this challenge, and several alternatives have also been proposed (Ruder, 2016).

Momentum and Nesterov methods help accelerate gradient descent, by guiding it in the relevant direction and smoothing oscillations. The momentum factor builds up by adding a fraction $\gamma$ (e.g., a
value such as 0.9) of the update vector of the last time step to the current update vector.

\[ \theta = \theta - \gamma v_{t-1} + \eta \nabla_{\theta} J(\theta) \]  

(2.6)

In turn, Nesterov provides a notion of correction to the momentum. After making a significant jump in the direction of the previously accumulated gradient, this approach calculates the gradient in the next position and then applies a correction.

Adagrad adjusts the learning rate taking into consideration the actual parameters, performing smaller updates for parameters associated with frequent features, and significant updates for parameters associated with infrequent features. Adadelta is an expansion of Adagrad that attempts to mitigate its aggressive and consistent decreasing learning rate. Rather than inefficiently accumulating all previous gradients, this approach restricts the window of collected past gradients to some predefined value. Therefore, the result gradients are recursively defined as a decaying average of the previous ones.

Adaptive Moment Estimation (Adam) calculates adaptive learning rates for each parameter. As Adadelta, Adam also stores an exponentially decaying average of past squared gradients \( v_t \). This approach also keeps an exponentially decaying average of past gradients \( m_t \), similar to momentum, as seen in Equation 2.7.

\[
\begin{align*}
    m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t, \\
    v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2
\end{align*}
\]  

(2.7)

In the previous equations, \( m_t \) and \( v_t \) are estimates of the first moment and the second moment of the gradients, respectively. With these values, Equation 2.8 describes the Adam update rule, identical to the Adadelta and RMSprop update rules.

\[ \theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t \]  

(2.8)

The typical algorithm used to apply gradient descent in deep neural networks is frequently denominated as backpropagation, and it consists of two steps. In a forward step, the algorithm keeps a stack of the function calls and their parameters. Using the derivatives of these functions, we are later able to backpropagate the errors. In a backward pass, the partial derivatives of a given cost function, with respect to the different parameters, are propagated back through the network. The chain rule of differentiation can be used to compute the derivatives associated to nested composite functions, and those values are then used by a gradient-based optimization algorithm to adjust the corresponding weights.
Convolutional Neural Networks (CNNs) are specialized neural networks for processing data that has a known grid-like topology (e.g., image or sequential time-series data). These networks use convolutions in place of general matrix multiplications in at least one of their layers.

Traditional neural network layers use a multiplication by a matrix of parameters in each of the layers, this way describing the interaction between each input unit and each output unit. This corresponds to a dense interaction, in the sense that every output unit interacts with every input unit. In contrast, by using a kernel with smaller size than the input, convolutional networks have sparser interactions. This provides the ability to detect small but meaningful features, such as edges in images with thousands of pixels. Multiple convolutions are often performed on an input, each using different kinds of filters and resulting in a distinct feature map. The final output of a convolution layer is a stack of all these feature maps, as seen in Figure 2.2. Through the feature maps, the network can efficiently describe complicated interactions between many variables, constructing such interactions from simple building blocks that each describe only sparse interactions. This technique exploits local spatial coherence (e.g., spatial close inputs are correlated) and spatial invariance in recognition (e.g., an object can appear anywhere in the image).

More formally, a convolution is an operation on two functions of a real-valued argument. Consider as an example the following equation:

\[
s(t) = (x \ast w)(t) = \int_{-\infty}^{\infty} x(\tau)w(t-\tau)d\tau
\]

, where the convolution is denoted as \( \ast \), the first argument to the convolution, \( x \), is often referred to as the input and the second argument, \( w \), as the kernel. By modifying the kernel, different kinds of convolved data can be obtained, often referred as the feature maps.

A Convolutional layer applies the convolutional operation, specifying the size and the number of kernels to be applied, how much the convolution filter at each step is moved (e.g., choosing bigger strides, leads to less overlap between the receptive fields), and the use of padding (maintain the same dimensionality by surround the input with zeros or the values on the edge). Padding is commonly used
in CNNs, in order to preserve the size of the feature maps, preventing the representations from shrinking at each layer.

Convolutional layers are often followed by a pooling layer. In order to reduce the number of parameters, pooling layers reduce the dimensionality by downsampling each feature map independently, reducing both the height and width, while persevering the depth intact. The most common type of pooling is max-pooling, which returns the maximum value in the pooling window.

The LeNet5 architecture, shown in Figure 2.3, was one of the first applications of CNNs for image recognition (Lecun et al., 1998). In LeNet5, pooling is performed with 2x2 windows, with a stride of 2 and no padding, while convolutions are done with 3x3 windows, with a stride of 1 and with padding.

Convolutional layers are often used together with a non-linear activation function, providing non-linearity to the network. Most often, the activation function is the rectified linear unit (ReLU) (Nair and Hinton, 2010), which is defined as the positive part of its argument. The final feature map values are not actually the sums of the inputs, but the result of the ReLU function applied to the summed inputs.

CNNs are also extensively used for natural language processing, in this case leveraging 1D convolution operations, together with vectorial representations for the words (e.g., word embeddings (Li and Yang, 2018)). Figure 2.4 illustrates how CNNs can produce fixed-length text representations, while focusing on the most important features. In this case, a sliding window of three words, corresponding to the concatenation of the three word embeddings is put through a 6 x 3 kernel, followed by the use of the hyperbolic tangent function. Finally, a max-pooling is applied, resulting in a final 3-dimensional vector.

2.1.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) take as input an ordered list of input vectors \( x_1, \ldots, x_n \), together with an initial state vector \( s_0 \), and returns an ordered list of state vectors \( s_1, \ldots, s_n \), as well as an ordered list of output vectors \( y_1, \ldots, y_n \). The state vector \( s_i \) and output vector \( y_i \), both represent the state of the RNN after observing the inputs \( x_1: i \). Thus, the RNN model provides a framework for conditioning on the entire history \( x_{1:i} \) without resorting to the Markov assumption (i.e., the conditional probability distribution

![Figure 2.3: Diagram with the LeNet5 architecture, adapted from Lecun et al. (1998).](image)
of next states depends only upon the present state, not on the sequence of previous states), which is traditionally used for modeling sequences.

Consider that $R$ is a recursively defined function that, given as input a state vector $s_i$ and an input vector $x_{i+1}$, results in a new state vector $s_{i+1}$. Consider also that $O$ is a function that maps a state vector $s_i$ to an output vector $y_i$. The symbol $\theta$ represents the parameters that are shared across all time steps, and $s_n$ can be thought of as an encoding of the entire input sequence. The job of network training is to set the parameters of $R$ and $O$ such that the state encodes useful information for the task we are aiming to solve.

By analyzing the unrolled RNN in Figure 2.5 (right), we can interpret it as a very deep neural network, in which the same parameters are shared across many parts of the computation. Network training can be made through a variation of the backpropagation algorithm that is typically referred to as backpropagation through time (BPTT), and is composed of the following steps: given an input sequence, create the unrolled computation graph, append a loss node, and then apply the backpropagation algorithm to
compute the gradients concerning that loss.

There are various ways in which the supervision signal can be applied, namely by considering only the final output vector $y_n$ as an encoding of the information in the sequence, or seeing the RNN as a transducer, generating an output for each input it reads in (i.e., for each of the outputs, we can compute a local loss signal, and the total loss can use several combinations (e.g., sum or average).

There are several different instantiations of the abstract RNN architecture, each providing concrete definitions for the functions $R$ and $O$. A simple RNN instantiation, often referred to as the Elman RNN (Elman, 1990), can be defined as follows:

\begin{equation}
    s_t = g \left( x_t \cdot W^x + s_{t-1} \cdot W^s + b \right)
    
y_t = s_t
\end{equation}

In the previous equation, the state at position $t$ is a linear combination of the input at position $t$ with the previous state $s_{t-1}$, passed through a non-linear activation function $g$, usually the hyperbolic tangent. The output at position $t$ is the same as the hidden state in that position.

This simple approach suffers from the vanishing gradient problem (i.e. the repeated multiplications of the gradient while back-propagating to earlier layers, causes its value to become significantly small, saturating the performance of the network) and therefore it cannot capture well long-range dependencies. Other RNN architectures attempt to address this limitation.

Long Short-Term Memory (LSTM) units are perhaps the most successful type of RNN architecture, achieving state-of-the-art results on several sequence modeling tasks (Hochreiter and Schmidhuber, 1997). In this specific architecture, the main idea to preserve gradients across time is to introduce as part of the state representation. The access to cells is controlled by gating components, deciding for each input state, the amount of the new input that should be transferred to the memory cell. The gate $g$, returning values in $[0, 1]^n$, is designed to be close to either 0 or 1 by using the sigmoid function. When multiplying this result component-wise with another vector $v \in \mathbb{R}^n$, the indices in $v$ corresponding to near-one values in $g$ are allowed to pass, while the others are blocked. The gating mechanisms allow for gradients related to the memory part to stay high across long time ranges. LSTM units can be formally defined as follows:
The symbols $i$, $f$, and $o$ represent three gates controlling the input, forget, and output, respectively. The values for these gates are calculated based on linear combinations of the current input $x_t$ and the previous state $h_{t-1}$, passed through a sigmoid activation function. An update candidate $g$ is computed as a linear combination of $X_t$ and $h_{t-1}$, passed through an hyperbolic tangent activation function. Then, the memory $c_t$ is updated, and the forget gate controls the amount of the previous memory that is kept by $c_{t-1} \odot f$, while the input gate controls how much of the proposed update is kept through $g \odot i$. At last, the value of $h_t$ is calculated based on the values of the memory $c_t$, passed through an activation function, and controlled by the output gate.

Gated recurrent units (GRUs) make use of a gating mechanism similarly to LSTMs, although using fewer gates and without separating the memory component, thus reducing the computational complexity when compared with LSTM (Cho et al., 2014a). GRUs use two gates: the first, $r$, controls both the access to the previous state $s_{t-1}$ and the value of the proposed update $\tilde{s}_t$; and the second gate $z$, determines the value of the updated state $\tilde{s}_t$, by applying a gate controlled interpolation on the previous state $s_{t-1}$ and the proposal $h$. They can be formally defined as follows:

$$s_t = (1 - z) \odot s_{t-1} + z \odot \tilde{s}_t$$
$$z = \sigma(x_t \cdot W^{xz} + s_{t-1} \cdot W^{zs})$$
$$r = \sigma(x_t \cdot W^{xr} + s_{t-1} \cdot W^{zs})$$
$$\tilde{s}_t = \tanh(x_t \cdot W^{xz} + (s_{t-1} \odot r) \cdot W^{zs})$$
$$y_t = s_t$$

Recent studies suggests better results with LSTM cells when the bias term of the forget gate is close to one and when the dropout mechanism is only applied on the non-recurrent connection (Jozefowicz et al., 2015).

The previous examples use the hyperbolic tangent function as the activation function. An alternative is the penalized hyperbolic tangent function (Xu et al., 2016), this function penalizes the negative part of
the saturated activation function. The equation describing this behaviour can be defined as follows:

\[ f(x) = \begin{cases} 
\tanh(x) & \text{if } x > 0 \\
 a \cdot \tanh(x) & \text{otherwise} 
\end{cases} \] (2.13)

In the previous equation, \( a \in (0, 1) \). Due to its finite range, it can take the role of a gate and therefore be employed in more sophisticated neural network units (e.g., LSTM).

### 2.2 Deep Learning for Text Classification

The objective of improving the interpretability of the results produced by the classification models, can be addressed with attention mechanisms. They provide better interpretability by giving insight into which words and sentences contributed to the classification decision.

Yang et al. (2016) explored the use of attention mechanisms in the document classification task, as an approach to improve classification results and, simultaneously, provide interpretability by giving insight into which words and sentences contributed to the classification decision. The authors also studied the hypothesis that more meaningful representations can be created by including knowledge of the document structure in the model architecture. The proposed neural network has a hierarchical structure that mirrors the hierarchical structure of textual documents (e.g., textual documents are composed of sentences, which in turn are composed of words), applying attention mechanisms both at the word and the sentence level.

As shown in Figure 2.6, the Hierarchical Attention Network (HAN) has four main parts, namely a word encoder, a word-level attention layer, a sentence encoder, and a sentence-level attention layer. Both the
word and sentence encoders use bidirectional GRUs to learn meaningful and contextual representations of the respective words or sentences.

Using a pre-trained embedding matrix, HAN first embeds each word to a vector. This vector is then used as input into the GRU-based sequence encoder, producing the word annotation $h_{it}$.

The word attention layer processes the representation $h_{it}$ through a one-layer MLP, creating the hidden representation $u_{it}$. The importance of each word is measured as the similarity of $u_{it}$ with a word level context vector $u_w$. Through a softmax function we get the normalized importance weights $\alpha_{it}$ and, through the weighted sum of the word representations, leveraging these weights, we get a sentence vector $s_i$.

The same approach is applied for both the sentence encoder and attention mechanisms. A document vector $v$ is produced as the output of the sentence attention mechanism, summarizing all the information of sentences in the document. This high level representation of the document can be used as a feature for document classification.

Through experiments with a variety of text classification datasets, the authors showed that the hierarchical attention network can achieved good results, outperforming non-hierarchical models based on LSTMs or GRUs. More recently, other studies have also successfully applied the same model on other text classification tasks, for instance in the domain of clinical text (Duarte et al., 2018).

In terms of model interpretability, visualizing the weights associated with each word in a sentence can demonstrate a coherent context-dependent word importance in the classification decision. For instance, in the experiment reported by Yang et al. (2016) concerning in classifying user reviews, the word good has a higher weight for reviews with higher rating, and the inverse trend is also observed for the word bad.

Qiao et al. (2018) instead proposed two methods for learning task-specific distributed representations of $n$-grams, referred to as region embeddings. The goal is to learn representations of small text regions, preserving the local internal structural information and supporting text classification.

The model utilizes both the word embeddings $e_{wi}$ and their local contexts $K_{wi}$ to produce meaningful region embeddings. Let $\text{region}(i, c)$ represent a fixed length contiguous sub-sequence of the document, centered in the word $w_i$ and with region length of $2 \times c + 1$ (e.g., region(3, 2), represents the sub-sequence food is not very good of the sentence The food is not very good in this hotel). The word embedding $e_{wi}$ represents the embedding of the word $w_i$, obtained from a look up layer for the representation of the vocabulary matrix $E \in \mathbb{R}^{h \times v}$. The local context $K_{wi}$ represents a distinctive linear projection function on $e_c$ in the local context $\text{region}(i, c)$, utilizing a local ordered word information instead of each word. $P_{\omega_i}$ represents the projected word embedding of the word $w_i$, combining both word embeddings and local contexts with an element-wise multiplication.

As shown in Figure 2.7, the authors proposed two distinct methods to perform the region embedding,
leveraging different perspectives.

The word-context region embedding composes the semantics of a given region by only taking in consideration the influence of the middle words on the context words. This approach combines the local context unit of the middle word of the region with the original word embeddings of the region, using a element-wise multiplication. A max pooling operation is then applied to the resulting projected embeddings, extracting the most predictive features present in the region.

Alternatively, the context-word region embedding composes the semantics of a given region taking in consideration the influence of the local context words on the middle word. In this case, the projected embeddings are calculated by combining the original word embedding of the middle word, with the local context units of the words in the region. The max pooling operation explained in the first approach is also applied in this case. To represent variable-sized documents, the authors propose to sum up the embeddings of all regions, and then feed the result to a fully-connected layer for document classification. Both models were evaluated in eight text classification datasets, and the obtained results showed that the word-context region embeddings outperformed the context-word region embeddings. A multi-size combination of region sizes lead to better accuracy than single-size, increasing word embedding dimensionality lead to consistently better accuracy, showing that this model is more robust to overfitting than previous approaches (e.g., FastText (Joulin et al., 2017), and simple CNN and RNN models). The
obtained results were supported by visualization experiments, showing that the proposed local context unit can capture the semantic and syntactic information for each word.

Cer et al. (2018) studied two different methods for creating sentence embeddings, designed for transfer learning to other NLP tasks. The first method uses the Transformer (Vaswani et al., 2017) encoding architecture to create a meaningful representation of each sentence. The encoder is composed by six layers, each one composed by two sub-layers. The first sub-layer is a multi-head self-attention mechanism, while the second is a position-wise fully connected feed-forward network, creating a fixed length sentence encoding vector. The second method uses a deep averaging network (DAN) (Iyyer et al., 2015). In this case, the sentence embeddings are the result of passing averaged word and bi-gram embeddings through a feedforward neural network. The results suggest that transfer learning from the Transformer based sentence encoder performs better than the transfer learning from the DAN encoder. Models that use sentence level transfer learning achieved better performance than models that only use word level transfer learning.

2.3 Deep Learning for Image Classification

Simonyan and Zisserman (2014) presented the Visual Geometry Group network (VGGNet) architecture for the Image Net Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2015) in 2014. This architecture achieved first place on the image localization task (find where in an image a certain object is) and a second place on the image classification task.

As shown in Figure 2.8, the architecture is characterized by its simplicity, adopting only $3 \times 3$ convolutional layers stacked with $2 \times 2$ pooling throughout the network, followed by two fully-connected layers of 4096 nodes and a softmax classifier. The main drawback of this architecture is that it contains an elevated number of parameters, therefore making it challenging to train.

Deep Learning architectures for modeling visual contents consist in stacking convolutional and pooling layers throughout the network, followed by fully-connected layers. As Convolutional Neural Networks (CNN) become deeper, the number of parameters continues to grow, therefore making it challenging to train (e.g. vanishing-gradient problem).

He et al. (2016) presented the Residual Network (ResNet) architecture, aiming to avoid the vanishing gradient problem in very deep neural networks, this architecture won the ILSVRC (Russakovsky et al., 2015). Given the repeated multiplications of the gradient while back-propagating to earlier layers, its value becomes significantly small, saturating the performance of the network. The authors introduced the concept of identity shortcuts, a direct connection that skips one or more layers, forming a residual block. This architecture is explained further in Chapter 3.

Huang et al. (2017) presented Dense Convolutional Networks (DenseNets) for image classification.
The authors argue that CNNs can be significantly deeper, more accurate, and efficient to train if they hold shorter links between initial and later layers. The authors propose to use additional connections to alleviate the vanishing-gradient problem by strengthening the feature propagation and reuse. Each layer’s input consists in all the output feature-maps of all the preceding layers. Thus, the output feature-maps of a given layer will be used as part of the inputs of all the subsequent layers. Therefore we can interpret this flow of feature-maps as the global state of the network. Each layer contributes with \( k \) feature-maps of its own to the global state. Thus, the value of \( k \) represents the growth rate of the network, adjusting the quantity of new information each layer provides to the global state.

In order to improve the computational efficiency of the network, the authors suggest the use of two different mechanisms. The bottleneck layer reduces dimensionality of the layer’s input by adding a \( 1 \times 1 \) convolution before each \( 3 \times 3 \) convolution. The compression layer reduces the dimensionality of the layer’s output feature maps by generating \( \lfloor \theta m \rfloor \) output feature maps, where the compression factor can have the values: \( 0 < \theta \leq 1 \).

The proposed architecture was evaluated on four object recognition benchmark datasets. In general, although requiring less computation, the model was able to achieve improvements over the state-of-the-art on most datasets. The results also showed that without compression and bottleneck layers, DenseNets seem to perform better as the number of layers and growth factor increases. In turn, DenseNets with bottleneck and compression were shown to be more parameter-efficient, achieving robust results per number of parameters of the network. This characteristic can also be interpreted as being less prone to overfitting. This architecture is explained further in Chapter 3.

Parmar et al. (2018) presented the Image Transformer, an extension of the recently proposed model architecture named Transformer (Vaswani et al., 2017) for sequence modeling, to image problems with
Figure 2.9: Image Transformer, as described in the original paper by Parmar et al. (2018).

a tractable likelihood. The Transformer architecture is based on sequence-to-sequence model for Statistical Machine Translation (SMT) (Cho et al., 2014b). This architecture includes two RNNs, namely the encoder, which processes the input sequence, creating a continuous meaningful representation, and the decoder, which generates the output, one symbol at a time. This encoder-decoder model is designed to re-utilize previously generated symbols as extra input, while generating the next symbol.

The Transformer relies on self-attention mechanisms, previously shown to be an effective way of modeling textual sequences. Specifically, the Transformer model uses a multi-headed attention mechanism, an attention weight matrix is calculated separately in $n$ different attention heads. The result values are then concatenated and element-wise multiplied by a final weight matrix. The Transformer architecture also attempts to capture a sense of order in the sequence, by using position encoding vectors, also added to each input embedding. Thus, the model is able to learn a specific pattern determining the distance between different words in the sequence. Similarly to the ResNet approach, the architecture of the encoder has a residual connection around each sub-layer of self-attention mechanism and a feed-forward block.

The Image Transformer represents an image by encoding each of the input pixels’ three color channels as a channel-specific set of 256 $d$-dimensional embedding vectors of the intensity values in the range $[0; 255]$. An image with width $w$ and height $h$ is represented as a tensor with shape $[h \cdot w \cdot 3, d]$, and this representation is processed through a convolution operation that combines the three channels per
pixel, generating a representation with shape \([h, w, d]\). From this representation, an encoder generates a contextualized, per-pixel-channel representation of the source image, while a decoder autoregressively produces an output image, one channel per pixel at each time step. For these both tasks, the image transformer uses stacks of self-attention and feed-forward layers. In this context, the attention distribution is then used to weight the contribution of the other pixels’ representations to the next representation for the pixel at hand.

Parmar et al. (2018) introduced two ways of applying the local attention to images, by varying the query blocks. In 1D local attention, the input tensor is flattened with positional encodings in raster-scan order. While being contiguous in the linearized image, these query blocks can be discontinuous in image coordinate space, not exploiting local spatial coherence. The 2D local attention approach tries to overpass this disadvantage by partition the input tensor with positional encodings into rectangular query blocks, contiguous in the original image space. The experimental results show that increasing the receptive field improves perplexity significantly, highlighting an advantage of local self-attention over CNNs: the number of parameters is independent of the size of the receptive field. In the task of recovering a high-resolution image given a low resolution one, while conserving realistic and credible details, the results reveal a significant improvement on the percentage of humans deceived. The previous state of the art was 10.2% in terms of percentage of fooled humans, and using 2D local attention, the image transformer achieved 36.11 ± 2.5%.

Hinton et al. (2011) introduced capsule networks, as an improvement over traditional CNNs. The authors note that CNNs lack in considering the pose relationship, both rotational and translational, between the straightforward features that form up a more complex one, arguing that to perform classification and object recognition tasks correctly, it is essential to take into account the hierarchical pose relationships between object parts.

In capsule networks, as originally stated by the author, each capsule learns to recognize an implicitly defined visual entity over a limited domain of viewing conditions and deformations and it outputs both the probability that the entity is present within its limited domain and a set of instantiation parameters that may include the precise pose, lighting and deformation of the visual entity relative to an implicitly defined canonical version of that entity (Hinton et al., 2011).

The length of the capsule’s output vector encodes the probability of recognition of a specific feature. A non-linear squashing function is used to guarantee that short vectors get shortened to almost zero length, while long vectors get resized to a length near 1. The state of the perceived feature is encoded as the direction the vector points to.

Leveraging the idea of capsules, the authors proposed a CapsNet architecture with two parts: the encoder and the decoder. The encoder takes as input an image and learns to encode it into a \(n\)-dimensional vector of instantiation parameters. It contains three layers, namely a convolutional layer, a
PrimaryCaps layer and ClassCaps layer. The PrimaryCaps layer has 32 primary capsules whose work is to take the basic features perceived by the convolutional layer and produce combinations of them. The ClassCaps layer has a number of capsules equal to the number of target classes. The decoder has three fully connected neuron layers and takes as input the 16-dimensional vector from the correct ClassCap and learns to decode it into an image of the target class, forcing the capsules to learn features that are helpful for reconstructing the original image.

Capsule Networks cannot be trained with the standard backpropagation algorithm, but Sabouret al. (Sabour et al., 2017) presented a solution, referred to as the dynamic routing between capsules algorithm. This algorithm, which is outlined in the listing shown below, explores the idea that lower level capsule will send its input to the higher level capsule that agrees with its input.

Algorithm 1 Dynamic Routing Algorithm

1: procedure ROUTING(\(\hat{u}_{j|i}, r, l\))
2: All \(b_{j|i} = 0\)
3: for \(r\) iterations do
4: for all capsule \(i\) in layer \(l\) do
5: \(c_{j|i} = \text{softmax}(b_{j|i})\)
6: end for
7: for all capsule \(j\) in layer \((l+1)\) do
8: \(s_j = \sum_i c_{j|i} \hat{u}_{j|i}\)
9: end for
10: for all capsule \(j\) in layer \((l+1)\) do
11: \(v_j = \text{squash}(s_j)\)
12: end for
13: for all capsule \(i\) in layer \(l\) and capsule \(j\) in layer \((l+1)\) do
14: \(b_{j|i} = b_{j|i} + \hat{u}_{j|i} \cdot v_j\)
15: end for
16: end for
17: return \(v_j\)
18: end procedure

For each layer of capsules, the input to a capsule \(s_j\) is a weighted sum over all prediction vectors \(\hat{u}_{j|i}\) from the capsules in the previous layer. This value is the result of the multiplication of the output \(u_i\) of each capsule in the previous layer, by a weight matrix \(W_{ij}\), as shown next:

\[
s_j = \sum_i c_{j|i} \hat{u}_{j|i}, \quad \hat{u}_{j|i} = W_{ij} u_i
\]  

Table 2.1: Comparing a capsule vs a traditional neuron.

<table>
<thead>
<tr>
<th></th>
<th>Capsule</th>
<th>Traditional neuron</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input and output from nodes</td>
<td>vector</td>
<td>scalar</td>
</tr>
<tr>
<td>Weighting</td>
<td>(s_j = \sum_i c_{j</td>
<td>i} \hat{u}_{j</td>
</tr>
<tr>
<td>Non-linear activation</td>
<td>(v_j = \frac{\hat{u}_{j</td>
<td>i}}{|\hat{u}_{j</td>
</tr>
</tbody>
</table>
In the previous equation, the \( c_{ij} \) coupling coefficients between each capsule and all the capsules in the following layer sum to 1, and are calculated by a routing softmax as shown in the next equation.

\[
c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})}
\]  

(2.15)

In Equation 2.15, \( b_{ij} \) are the log prior probabilities that capsule \( i \) should be coupled to capsule \( j \) learned simultaneously as all the other weights. These values are adjusted taking into account the agreement between the current output \( v_j \) of each capsule \( j \) in the layer above and the prediction \( \hat{u}_{ji} \) made by capsule \( i \). This value is computed by the scalar product \( a_{ij} = v_j \cdot \hat{u}_{ji} \), and then added to the initial logit \( b_{ij} \). Then, the new values for all the coupling coefficients linking capsule \( i \) to following level capsules are computed as follows:

\[
L_k = T_k \times \max(0, m^+ - \|v_k\|^2) + \lambda (1 - T_k) \times \max(0, \|v_k\| - m^-)^2
\]

(2.16)

In the previous equation, \( T_k = 1 \) if class \( k \) is present in the input. The parameter \( \lambda \) controls the shrinking of the lengths of the activity vectors of all the capsules, by down-weighting the loss for absent classes.

The CapsNet model trained on the handwritten digit images of the MNIST dataset achieved higher test classification accuracy than a baseline convolutional model. On the image CIFAR10 dataset, the ensemble of capsule models achieved a 10.6% test error, a value almost equal to the standard convolutional networks.

### 2.4 Deep Learning for Medical Text Reports Classification

Duarte et al. (2018) demonstrated that deep learning methods previously proposed to general text classification problems can also be applied successfully in the bio-medical context. The authors evaluated the performance of a hierarchical attention network on the task of classifying death certificates according to ICD-10, leveraging both the free-text descriptions in death certificates, and text from the associated autopsy reports and clinical bulletins. The authors proposed a deep neural network that both combines word embeddings, recurrent units and attention mechanisms, generating meaningful intermediate representations for the textual contents. Leveraging co-occurrences between classes in the training data, the authors also explored different mechanisms for initializing the weights of the last nodes of the network. The backpropagation algorithm was used for training, together with the Adam optimization method.

The inputs of the model have a hierarchical structure (i.e., the words form different fields, and all the fields form the input entry), and the model first builds an individual representation of each field, later aggregating these representations into an encompassing representation for each death certificate. This
two-level hierarchical approach, inspired on the previous method from Yang et al. (2016), is illustrated in more detail in Figure 2.10, showing that the field encoders have the same architecture as the document encoders.

The recurrent unit used in both levels is a bi-directional GRU. While in the first level (e.g., the word encoder), the model leverages word embeddings as input, the second level (e.g., the document-encoder) uses as input the field representations generated at the first level. The result representation is then concatenated with an alternative representation built through a simpler mechanism that computes the average of the embeddings for all words in the input. This simpler mechanism eases the direct propagation of errors back into the embeddings, speeding up its learning. The word embeddings are not pre-trained, instead they are initialized randomly and then learned during model training.

Finally, the result of the concatenation is then passed to feed-forward output nodes. The model considers three different outputs, namely a softmax node that outputs the ICD-10 full-code, a softmax node that outputs the ICD-10 block, and a sigmoid node that outputs multiple ICD-10 codes, corresponding to all contributing and auxiliary conditions, together with the main cause of death. The softmax nodes use the categorical cross-entropy loss function, while in the sigmoid node, the authors choose the binary cross-entropy loss function because of its performance in multi-label classification problems (Nam et al., 2014).

The authors also tested two different approaches to initialize the weights of the output nodes, with basis on co-occurrences between the class labels. The first approach leverages the Apriori algorithm (Agrawal et al., 1994) to find significant and frequent label co-occurrence patterns (i.e., frequent item-sets). The second approach uses non-negative matrix factorization (Lee and Sebastian Seung, 1999) over a label co-occurrence matrix.

Experiments were made with a dataset of the death certificates obtained from the mortality surveillance system of the Portuguese Ministry of Health. In this dataset, most classes are sparsely used. Despite the classes being unbalance, the model still obtained a high classification accuracy. The results
showed that frequent itemset initialization achieved the best accuracy, respectively 89.320%, 81.349%, and 76.112% for chapter, block and full-code classification.


Memory Augmented Neural Networks proposed by Weston et al. (2014) have access to external memory that the model may use to make predictions. More specifically, in a key-value memory network, the keys are the training instances, while the values are the correspondent labels. The model searches for the most similar training instance to find its respective value. Rios and Kavuluru (2018) also leveraged this idea, combining it with CNNs, forming a semi-parametric multi-head matching network.

An overview of the model is shown in Figure 2.11. The first main component is an CNN layer augmented with external memory, over a support set $S$ (i.e., a small subset of the training dataset). Related instances are chosen automatically by joining ideas from metric learning and neural attention. Unlike in traditional $k$-nearest neighbor classifiers, the support set instances just enrich and complement the features derived from the input instance, and therefore the model does not explicitly use the labels. An output layer for $L$ labels, using the sigmoid activation function, uses the concatenation of the CNN representations of the input and the support set, $h$, to form the value $\hat{y}$, and is described in the following equation:

$$\hat{y} = P(y|x, S) = \sigma(W_c h + b_c) \quad (2.17)$$

In the previous equation, $W_c$ is a matrix representing the multiple representations of a given instance, as result of the multi-head mechanism, similar to self-attention (Lin et al., 2017).

The second main component, called metalabeler, ranks and select the top $k$ labels for each instance.
This approach is proposed as solution to the highly unbalanced classification problem, i.e., when training using binary cross-entropy the predictions are biased towards 0, for threshold of 0.5. In order to fix this problem, the authors optimized the threshold for each class, by training a regression based output layer, and is described in the following equation:

\[ \hat{r} = \text{ReLU}(W_g(x) + b_r) \]  

(2.18)

In the previous equation, \( \hat{r} \) estimates the number of classes \( x \) should be assigned to.

The multi-head matching network ranks each label by its score in \( \hat{y} \). Then \( \hat{r} \) is rounded to the closest integer and then predicts the top \( \hat{r} \) ranked labels.

The MetaLabeler training leveraged the multi-label cross-entropy loss and the mean squared error loss as loss functions, training together using the multitask learning paradigm (Collobert et al., 2011).

Experiments where performed using the the Medical Information Mart for Intensive Care (MIMIC) dataset. The results show several improvement, over simple CNNs and matching networks, in most metrics for both frequent and infrequent classes in the dataset. However, in terms of general accuracy the model considerably performed worse than the previous state of the art. Comparing to traditional matching networks, the proposed method performs significantly better across every evaluation metric.

Shen et al. (2018) introduced the cluster-based word vector expansion method, exploring hierarchical agglomerative clustering (HAC) to cluster words similar in the semantic space. The cluster-based word vectors represent the implicit topic information of the cluster. This approach targets the problem of features sparseness and semantic ambiguity in medical short text classification field.

As shown figure 2.12, the framework starts by applying the unsupervised skip-gram model to the unlabeled medical corpus, obtaining the respective word embeddings. The skip-gram model objective is to predict a context, the surrounding words, of a given a center word. This mechanism will be formally explained in Chapter 3.

Second, HAC cluster the words in the labeled corpus. Starting with every single sample as a single cluster, then progressively merging two most similar clusters, until reaching only one cluster or a predefined cluster limit criteria. Adopting average linkage and Euclidean distance-based similarity, the cluster embeddings of each cluster is calculated with the following equation:
\[ C = \frac{1}{m} \sum_{i=1}^{m} V_i \]  

where \( V_i \) represents the vector of the \( i \)th word, and \( m \) represents the number of words in the cluster.

Finally, the Word-Cluster Embeddings can be calculated as the concatenation of the Word Embeddings and the Cluster Embeddings.

The authors experimented with the word-cluster embeddings, by passing it through simple CNN and LSTM models to classify short texts on five medical datasets. Both CNN and LSTM models using HAC-based word-cluster embeddings achieved better results than models using different clustering algorithms. The authors of HAC-LSTM model achieved 0.947 accuracy, improving over the previous state-of-the-art RCNN by Lai et al. (2015).

### 2.5 Deep Learning for Medical Image Classification

#### 2.5.1 Classification of Radiology Reports

Wang et al. (2018) presented a deep learning architecture for thorax disease classification in chest X-ray images, named TieNet. As shown in Figure 2.13, the text-image embedding network (TieNet) is an end-to-end trainable CNN-RNN architecture enhanced with attention-encoded text embeddings (AETE), saliency weighted global average pooling (SW-GAP), and a joint learning loss function.

The network takes as input an image \( I \) and a sequence of \( 1 - \) of \( V \) encoded words. The initial CNN component uses layers borrowed from an ImageNet pre-trained model for image classification, including an extra convolutional layer (i.e., a transition layer) to manipulate the spatial grid size and feature dimension. The RNN is based on the visual image spatial attention model by Xu et al. (2015). A long short-term memory (LSTM) decoder on different parts of the input image during each decoding step, and the attention signal is determined by the previous hidden state and CNN features.

The convolutional activation in the transition layer generates the convolutional features, denoted as \( X \) with dimensions \( 16 \times 16 \times 1024 \). This value initializes the RNN’s hidden state, \( h_0 \), where a fully connected embedding, \( \phi(X) \), maps the size \( d_X \) transition layer activation to the LSTM state space of dimension \( d_h \). The RNN’s input consists of the current word plus the multiplication element-wise of \( X \) with a deterministic and soft visual spatial attention \( a_t \) outputting at each time the subsequent attention map. AETE uses attention to combine the most salient portions of the RNN hidden states. Let \( H = (h_1, \ldots, h_T) \) be the \( d_h \times T \) matrix of all the hidden states. The attention mechanism outputs a \( r \times T \) matrix of weights \( G \), computed as follows:

\[
G = \text{softmax} (W_{s2} \cdot \tanh (W_{s1} \cdot H))
\]  (2.20)
Then, the model computes an $r \times d_h$ embedding matrix, $M = GH$, which in essence executes $r$ weighted sums across the $T$ hidden states, aggregating them together into $r$ representations. Each row from $G$ indicates how much each hidden state contributes to the final embedded representation of $M$, adding an extra layer of interpretation. In the end the model uses an max-over-$r$ pooling operation across $M$, producing an embedding vector $\hat{X}_{\text{AETE}}$ with size $d_h$.

SW-GAP improves visual embeddings for classification by re-using the attention mechanism, $G$, except that this part performs a max-over-$r$ operation, producing a sequence of saliency values $g_t(t = 1, \ldots, T)$ for each word $W_t$. This value will help weight and select the subsequent spatial attention maps, generated at each time point:

$$a_{ws}(x, y) = \sum_t a_t(x, y) \ast g_t$$  \hfill (2.21)

The result from Equation 2.21 is then used to highlight the spatial regions of $X$ with more meaningful information:

$$\hat{X}_{\text{SW-GAP}}(c) = \sum_{(x,y)} a_{ws}(x, y) \ast X(x, y, c)$$  \hfill (2.22)

In sum, $\hat{X}_{\text{SW-GAP}}$ represents the global visual information, guided by both text and visual attention. The joint learning mechanism combines the $\hat{X}_{\text{AETE}}$ and $\hat{X}_{\text{SW-GAP}}$ global representations, first by
concatenating both $\hat{X} = [\hat{X}_{AETE}; \hat{X}_{SW-GAP}]$ and then by feeding them to a final fully-connected layer, to produce the output for multi-label classification. This way, the model explores the intuition that the connection between the CNN and RNN network can help the training of both. More specifically, the image activations can be adjusted for the text embedding task, and the salient image features could be extracted by pooling based on high text saliency.

As the TieNet can also generate text reports, the authors also optimize an RNN generative model loss, $L_R$, by adding weights to instances associated with different categories. The model overall loss has two parts, namely a sigmoid cross entropy loss $L_C$ for the multi-label classification, and the loss $L_R$ from the RNN generative model (Xu et al., 2015).

The authors evaluated the proposed model on 3 different datasets, namely ChestX-ray14 (Wang et al., 2017), Open-i (Demner-Fushman et al., 2016b), and a not public hand-labeled dataset. The pre-trained word embedding vectors were learned on PubMed articles using the word2vec of the genism\(^1\) package with feature size of 200. The evaluation consisted of analyzing receiver operating curves (ROC) for each category, to measure the image classification performance, and analyzing the results of BLEU (Papineni et al., 2002), METEOR (Lavie and Agarwal, 2007), and ROUGE (Lin, 2004) scores for the report text generation task.

In image classification, the model achieved a high performance on all datasets both ChestX-ray14, Open-i and Hand-labeled (i.e., over 0.87\% in AUC with only reports as the input, and over 0.90\% in AUC with both image and associated report, using sample number weight average). In the classification and reporting of Chest X-rays, the model outperformed previous work of Xu et al. (2015), achieving 0.2860 in BLEU, 0.1024 in METEOR, and 0.1988 ROUGH.

One interesting finding is that some false predictions (e.g., Mass and Consolidation) can actually be recognized in the images (i.e., verified by a radiologist), although not being noted in the report. The proposed network can thus, to some extent, correlate the image pixels, with the text description.

Baltruschat et al. (2018) presented an analysis on the task of pathology detection as a multi-label classification problem for the ChestX-ray14 dataset. The paper studied the effect of weight initialization, model pre-training and transfer learning, considering network architectures such as ResNet-50 (He et al., 2016) with a large input size. Moreover, the authors also assessed the addition of non-image features on the performance of the model. Specifically, the authors experimented with two distinct initialization strategies: initialize randomly, and thus training the models are trained from scratch, or use pre-trained weights from different but similar domains and tasks. The transfer-learning approaches that were used considered either off-the-shelf without retraining the weights, and fine-tuning approaches consisting on using pre-trained models with small retraining of the weights in the new domain.

The network architectures used by the authors included the original ResNet-50 with 3 RGB input

\(^1\)https://radimrehurek.com/gensim/models/word2vec.html
channels (He et al., 2016), ResNet-50 with 1 input channel, and ResNet-50 with increased input size of (428,428) pixels, and also with an extra pooling layer after the first bottleneck block. The non-image features included age, gender, and view position. Age was linearly scaled to [0, 1], while view position and gender were encoded as 0 or 1. Later, the new non-image vector is concatenated with the image feature vector. In this multi-label chest x-ray classification task, the evaluation metrics used by the authors were the Spearman correlation coefficient (i.e., measuring a rank correlation between models) and the standard AUC. Surprisingly, the off-the-shelf (OTS) model has a higher correlation to the from scratch models, when compared with the fine-tunned (FT) model. The results indicated a high variability of the outcome with respect to the selected dataset split. FT models achieved better AUC that OTS models and non-image features slightly increased the AUC on average. The best overall performance achieved 0.822 average AUC, the ResNet-50 was trained from scratch, with an enlarged input size and integrated non-image data.

Jiménez-Sánchez et al. (2018) explored the application of Capsule Networks in the medical imaging domain, which typically involves small amounts of annotated data and class-imbalance. In this domain, the paper compares the performance of CapsNet, LeNet (Lecun et al., 1998) and a baseline ConvNet, in two medical datasets that targets, mitosis detection (TUPAC16) and diabetic retinopathy detection (DIARETDB1). Reducing the amount of training data up to 1%, while keeping the original class distribution, showed consistent better F1-Scores for the CapsNet model. As the amount of training data increases, the difference in performance reduces. For different class-imbalance scenarios, the mean F1-scores show that the CapsNet surpasses the results of ConvNets.

Karimi et al. (2017) identified optimal parameters for setting up a convolutional neural network for coding radiology reports. The authors used a grid search procedure to test different values of batch size, number of epochs, application of activation functions separately on convolutional layers and fully connected layers, dropout rate, filter size, depth, learning rate, word representation vector size, and stride. The models were evaluated two datasets: ICD9 (Pestian et al., 2007) and IMDB (Maas et al., 2011), with stratified 10-fold cross-validation. For the ICD-9 dataset, using the CNN architecture proposed by Kim (2014) with optimal parameters, the model achieved 83.84% accuracy, an 2.29% improvement over the model with default parameters. The results suggest that pre-trained word vectors improve the classification accuracy, that trainable word embedding vectors are better than static ones, that in-domain word vectors are better than generic ones and that larger embedding size does not always lead to higher accuracy.

Ge et al. (2018) introduced a novel error function called Multi-label Softmax Loss, applying it in a multi-label fine-grained network, in order to leverage the particularity that in the ChestX-ray14 dataset, many pathologies are visually similar and that can be multiple present in a single scan.

The multi-label softmax loss (MSML) considers the relationship of multiple labels explicitly, and is
described in the following equation, where $Y_i$ defines the positive class indices $y_c = 1$ of the current sample, and where $|Y_i|$ is used for normalization and represents the cardinality of $Y_i$.

$$E_{MSML} = \frac{1}{|Y_i|} \sum_{l \in Y_i} \frac{\exp(x_i^l)}{\sum_{k \in Y_i} \exp(x_i^k)}$$ (2.23)

Each positive response $x_{i \in Y_i}$ is fed to the exponential function. The nominator contains the positive activation, while the denominator contains both the positive and the negative activation (i.e $x_{k \notin Y_i}$).

In order to better capture the relationship between the images and text representations, the authors used bilinear pooling. The bilinear pooling idea consists of performing, in one of the pooling layers, an outer-product at each spatial location $(i,j)$ of two networks to generate second-order statistical discriminative local feature representations. The bilinear pooling can be mathematically computed with the following equation:

$$p_{i,j} = \text{vec} (f_{1,i,j}^1, f_{2,i,j}^2)$$ (2.24)

In the previous equations, $f_{a,i,j}^1 \in \mathbb{R}^d$ is a local feature descriptor from one of the pooling layers in network $a$, while $p_{i,j}$ is the outer product of two vectors, vec() is the vectorization operation.

As seen in Figure 2.14, in order to break the symmetry, both CNN components have different auxiliary loss functions. The first, with the MSML function as loss, focuses on learning the label correlations, while the second, with a normal cross-entropy function as loss. The bilinear pooling outputs are used to calculate the fine-grained cross-entropy loss (FCE), corresponding to a weighted sum of the previous two. Experiments were performed in the ChestXRay14 dataset and the results show consistent improvement over models using only a cross-entropy loss, when applied in both Residual Networks, Dense Networks and Ensemble Methods. The ensemble of DenseNet-121 and VGG-F using the MSML loss function achieved an AUC of 0.8537, outperforming previous state-of-the-art methods.
2.5.2 Overview

The work presented leveraged different approaches for creating and combining meaningful representations of both text and image data, with focus in medical imaging. Addressing common particularities of medical domain datasets (e.g., class unbalance) and exploring mechanisms to provide more interpretability to the models. Several ideas from the work surveyed in this chapter have been considered, namely the multi-head attention mechanism, ResNet, and the TieNet architecture, all the details are explained in Chapter 4. Table 2.2 provides a compact summary of the papers presented in this section.
Table 2.2: Summary of previous work presented in this report.

<table>
<thead>
<tr>
<th>Work</th>
<th>Task</th>
<th>Intuition</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Yang et al., 2016)</td>
<td>Text classification</td>
<td>Representation incorporating knowledge of document structure</td>
<td>Word and sentence level attention mechanisms</td>
</tr>
<tr>
<td>(Qiao et al., 2018)</td>
<td>Text classification</td>
<td>Learn local context units for each word</td>
<td>Context-word region and word-context region embeddings</td>
</tr>
<tr>
<td>(Cer et al., 2018)</td>
<td>Text Classification</td>
<td>Learn text representations for transfer learning</td>
<td>Transformer encoder and DANs</td>
</tr>
<tr>
<td>(Simonyan and Zisserman, 2014)</td>
<td>Image classification</td>
<td>Convolution network architecture for image classification</td>
<td>VGGNet</td>
</tr>
<tr>
<td>(He et al., 2016)</td>
<td>Image classification</td>
<td>Direct connection of distant layers, forming a residual block</td>
<td>ResNet</td>
</tr>
<tr>
<td>(Huang et al., 2017)</td>
<td>Image classification</td>
<td>Hold shorter connections from early layers to later layers</td>
<td>DenseNet</td>
</tr>
<tr>
<td>(Parmar et al., 2018)</td>
<td>Image classification</td>
<td>Self-attention in image data: 1D and 2D local attention</td>
<td>Image transformer</td>
</tr>
<tr>
<td>(Hinton et al., 2011)</td>
<td>Image classification</td>
<td>Consider the pose relationship between simpler features</td>
<td>Capsule Networks</td>
</tr>
<tr>
<td>(Duarte et al., 2018)</td>
<td>Bio-Medical text classification</td>
<td>Word embeddings, recurrent units and neural attention mechanisms</td>
<td>Combined model initialization leveraging class co-occurrences</td>
</tr>
<tr>
<td>(Wang et al., 2018)</td>
<td>Radiology reports classification</td>
<td>Attention-encoded text embedding, Saliency weighted global average pooling, and Joint learning loss function</td>
<td>Text-image embedding network</td>
</tr>
<tr>
<td>(Baltruschat et al., 2018)</td>
<td>Radiology reports classification</td>
<td>Weight initialization, Pretraining and Transfer Learning</td>
<td>ResNet-50 with increased input size</td>
</tr>
<tr>
<td>(Jiménez-Sánchez et al., 2018)</td>
<td>Radiology reports classification</td>
<td>Benefits of equivariance over invariance</td>
<td>CapsNet</td>
</tr>
<tr>
<td>(Karimi et al., 2017)</td>
<td>Radiology reports classification</td>
<td>Optimal parameters for setting up a CNN</td>
<td>Pre-trained dynamic word vectors</td>
</tr>
<tr>
<td>(Ge et al., 2018)</td>
<td>Radiology reports classification</td>
<td>Many pathologies are visually similar and can be multiple present in a single scan</td>
<td>Multi-label softmax loss function</td>
</tr>
</tbody>
</table>
3

The Multi-modal Deep Neural Model for X-ray report classification

Contents

3.1 The Multi-modal Deep Neural Architecture ........................................... 37
3.2 Modeling Visual Contents ................................................................. 42
3.3 Per-Modality Model Pre-Training ....................................................... 45
3.4 Summary ......................................................................................... 46

36
This chapter describes a multi-modal deep neural network for classifying chest radiology exams. Section 3.1 provides an overview of the general multi-modal neural architecture, illustrated in Figure 3.1. Section 3.1.1 details the free-text processing path of the network, exploring different biomedical pre-trained word embeddings, together with recursive and attention units. Section 3.2 presents image processing path of the network, exploring different CNN architectures, for extracting meaningful features from the radiography images. The process of pre-training individually each modality path of the architecture is described in Section 3.3. Lastly, Section 3.4 presents a summary of the chapter.

3.1 The Multi-modal Deep Neural Architecture

As illustrated in Figure 3.1, the proposed neural network architecture has two separate branches that extract meaningful representations from the distinct information modalities, namely from X-ray images and from the associated radiology reports. The text classification branch has three main components: pre-trained biomedical word embeddings, Bidirectional Multiplicative Long Short-Term Memory (bi-mLSTM) units (Krause et al., 2017), and a multi-head neural attention (Vaswani et al., 2017) mechanism. In turn, the image classification branch uses modern CNN models, such as ResNet (He et al., 2016), DenseNet (Huang et al., 2017), DPN (Chen et al., 2017), or EfficientNet (Tan and Le, 2019). The complete network concatenates the representations from both branches, using the resulting features to inform a 14 nodes prediction layer. Due to the multi-label nature of the task of classifying radiology exams, the sigmoid activation function was chosen to convert each node’s score between 0 and 1, independently of the other node’s scores. If a score, for a specific class is bigger than 0.5, the model classifies that class as positive, therefore multiple classes can be classified as positive for the same instance. Although this fusion method is much simpler than the one used in the TieNet model (Wang et al., 2018), it allows us easily to pre-train the two branches of the network.
3.1.1 Representing the Textual Inputs

The increased number of biomedical text records publicly available, supported the development of pre-trained language representations (i.e. pre-trained word embeddings and contextual word representations), improving the performance of deep learning models in different biomedical domains (Chiu et al., 2016; Peng et al., 2019a,b). Thus, for representing text, we explore three pre-trained word embeddings, namely: BioWordVec (Chen et al., 2018), BioELMo (Jin et al., 2019), and BioBERT (Lee et al., 2019).

BioWordVec are FastText (Joulin et al., 2017) word embeddings pre-trained over large bio-medical corpora, specifically PubMed abstracts and clinical notes from MIMIC-III (Chen et al., 2018). The skip-gram model aims at predicting the surrounding words given a center word. For a sequence of words \( w_1, w_2, \ldots, w_N \), representing a textual corpus \( N \) words, and a \( c \) context windows size, the method attempts to maximize the following average log probability:

\[
\frac{1}{N} \sum_{n=1}^{N} \left( \sum_{-c \leq i \leq c, i \neq 0} \log (p(w_{n+i}|w_n)) \right)
\]

(3.1)

, where \( w_n \) represents the word at position \( n \). The standard definition for \( p(w_{n+i}|w_n) \) applies a softmax function, as described formally in the following equation:

\[
p(w_{|w_O}) = \frac{\exp \left( V(w_I) \top V(w_O) \right)}{\sum_{w=1}^{W} \exp \left( V(w) \top V(w_O) \right)}
\]

(3.2)

, where \( W \) represents the vocabulary size, and \( V(w_I) \) and \( V(w_O) \) are vector representations of an input center word and of an output context word, respectively. FastText word embedding method extends the Skip-gram model by representing each word as an \( n \)-gram of characters. The FastText approach involves training embeddings for individual word tokens and character \( n \)-grams, allowing us to compute embeddings for out-of-vocabulary words, by averaging the embeddings for the corresponding \( n \)-grams. This aspect is particularly interesting for clinical text, where orthographic variations and typographical errors are often present.

BioELMo is an extension of the Embeddings from Language Model (ELMo) model introduced by Peters et al. (2018), providing contextual word embeddings pre-trained on PubMed abstracts. The original ELMo approach uses a multi-layer bi-directional recurrent architecture to produce the contextualized word embeddings. BioELMo vocabulary consists 1M most frequently tokens from the pre-training corpus. In particular, the RNN architecture uses to stacked layers of Bi-directional RNNs leveraging Long Short-Term Memory (LSTM) cells, i.e. a particular type of RNN that will be explained latter, together with multiplicate LSTMs (mLSTMs). The model was pre-trained using an unsupervised task named Bidirectional Language Model (BiLM), concerned with predicting if a sequence of words is an authentic
sentence. BiLM involves a forward and a backward Language Model (LM). The forward LM takes as input a sequence of tokens and applies a softmax layer to the last LSTM layer output, in order to predict the next token. Similarly, the backward LM receives a reversed sequence and tries to predict the previous token given the future context. Note that the parameters of the token representations and the softmax layer are connected in both directions, while the parameters of the LSTMs are not connected in both directions, becoming independent. The word representations are then calculated as a linear weighted combination of the two hidden layers of Bi-LSTMs, together with the context-independent token representation. As described in Figure 3.2, the context-independent token representation is produced by a 2048 channel char-gram CNN followed by two highway layers (i.e., an extension of the residual connections idea, modulating the quantity of input signal to be added to the output, by passing the input through fully connected layers with sigmoid and ReLU as non-linearity activation functions) and a linear projection down to 512 dimensions (Kim et al., 2016). Due to computational limitations, we follow ELMo’s feature-based approach, opposed to the jointly fine-tuned on a downstream task approach. Thus, the word representations are extracted and calculated by using the learnt downstream task layer weights to average the following layers: one token embedding, and two Bi-LSTM layers.

BioBERT exploits the pre-trained Bidirectional Encoder Representations from Transformers (BERT) model introduced by Devlin et al. (2018), fine-tuning a model pre-trained on general texts on PubMed abstracts and PubMed central full-text articles. Instead of RNNs for modeling sequences of words, BERT leverages a different neural architecture named the Transformer (Vaswani et al., 2017), a stack of encoder and decoder blocks. The encoder block has one layer of a multi-head attention (i.e., layer that will be explained latter) followed by a feed forward layer. The decoder block has this same structure with an extra masked multi-head attention. Contrasting with directional models that process a text sentence sequentially (i.e., left-to-right or right-to-left), BERT is considered a bidirectional model, as the Transformer encoder is fed with the entire sequence of words simultaneously, allowing the model to learn the context

Figure 3.2: ELMo’s architecture illustrating the creation of contextual embeddings through BiLM.
of a word based on all of its surroundings. BERT’s multi-layer bidirectional Transformer (Vaswani et al., 2017) architecture was originally pre-trained on Wikipedia and books corpora using two unsupervised tasks, namely the masked language model (MLM) and the next sentence prediction (NSP) tasks. MLM consists in predicting a randomly masked word based on the sentence context words, while NSP consist in predicting, for a given pair of sentences, if one follows the other in the original corpora. BERT segments sentences using the WordPiece model with a 30,000 token vocabulary, introduced by Wu et al. (2016). For a given corpora and a pre-defined maximum number of tokens, the WordPiece model is trained to select a set of word pieces that minimizes the total number of word pieces generated while segmenting the entire the corpus.

As illustrated in Figure 3.3, the segmented sequence starts with the [CLS] token, followed by the WordPiece tokens wrapped with two [SEP] tokens representing the division of the two sentence pair. The input embeddings are the sum of the token embeddings, the segmentation embeddings, and the position embeddings involved in the Transformer architecture. BioBert uses the original BERT vocabulary. Due to computational limitations, we follow BERT’s feature-based approach, opposed to the jointly fine-tuned on a downstream task approach. Thus, the word representations are extracted and combined using the hidden layer representations generated from last 4 Transformer encoders.

Leveraging word representations produced by methods such as one of the three models outlined before, typical approaches for representing textual data involve Long Short-Term Memory (LSTM) units for performing the analysis of sequences of embeddings, i.e. neural networks that model sequential data by having a recurrent hidden state regulated by gates. This units were previously described in Chapter 2. Bi-directional LSTM units (bi-LSTMs) can be used to process a sequence of words both in a forward (\( h_{it}^{\rightarrow} \)) and in a backward direction (\( h_{it}^{\leftarrow} \)). By concatenating the states from two independent LSTM units (i.e., \( h_{it} = [h_{it}^{\rightarrow}, h_{it}^{\leftarrow}] \)), bi-LSTMs can provide a more wide-ranging summary of the information at each position \( i \) the aforementioned ELMo approach for generating word embeddings uses two Bi-LSTMs.
In our model, instead of standard bi-LSTMs for processing the sequences of word embeddings, we used the recently proposed Multiplicative LSTM units (mLSTMs), also within a bi-directional arrangement. These units combine factorized hidden-to-hidden transitions (i.e., an idea taken from multiplicative recurrent neural networks) with the gating logic from LSTMs (Krause et al., 2017). Figure 3.4 provides an illustration comparing both recurrent cells. For a given input, mLSTMs can have different recurrent transition functions, and this approach can be formally defined as follows.

\[
\begin{align*}
    h_t &= \tanh(c_t) \odot o_t \\
    c_t &= c_{t-1} \odot f_t + g_t \odot i_t \\
    m_t &= (x_t \cdot W^{xm}) \odot (h_{t-1} \cdot W^{mh}) \\
    i_t &= \sigma(x_t \cdot W^{xi} + m_t \cdot W^{mi}) \\
    f_t &= \sigma(x_t \cdot W^{xf} + m_t \cdot W^{mf}) \\
    o_t &= \sigma(x_t \cdot W^{xo} + m_t \cdot W^{mo}) \\
    g_t &= \tanh(x_t \cdot W^{xh} + m_t \cdot W^{mh}) \\
    y_t &= h_t
\end{align*}
\] (3.3)

where \(m_t\) represents an intermediate state.

Moreover, instead of the standard sigmoid or hyperbolic tangent activation functions, we used the penalized hyperbolic tangent (Xu et al., 2016) as the activation function within the recurrent units, penalizing negative inputs to create an additional non-linearity that facilitates model training, as shown in the next equation:

\[
f(x) = \begin{cases} 
    \tanh(x) & \text{if } x > 0 \\
    0.25 \times \tanh(x) & \text{otherwise}
\end{cases}
\] (3.4)

Previous experiments showed improved results when combining the penalized hyperbolic tangent with standard LSTM units, in a variety of NLP tasks (Eger et al., 2018).

The results from the bi-directional mLSTM units (i.e., the hidden states) can be combined through attention mechanisms, i.e. functions that map a context vector and a set of hidden state vectors into a single output vector (Vaswani et al., 2017; Yang et al., 2016). The output vector is the result of a
weighted sum of the hidden states, where the weights are computed by a score function that takes as input the context vector and each hidden state vector. For defining the weight that should be given to the representation at each time step $t$ (i.e., for weighting the values at each position $t$ for a given input sequence), a commonly used attention mechanism can be formally defined as follows:

$$s = \sum_t \left( \frac{\exp(\tanh(W \cdot y_t + b_t) \cdot u)}{\sum_{t'} \exp(\tanh(W \cdot y_{t'} + b_{t'}) \cdot u)} \right) \cdot y_t$$  \hspace{1cm} (3.5)

where, $s$ corresponds to the text representation resulting from the attention weighting scheme, while $W$, $u$ and $b$ are learned parameters, respectively a matrix of weights, the context vector, and a bias term. Multiplying the softmax outputs (i.e., the part in brackets) with each hidden state keeps the values of the words that are more significant for the task, and reduces the values for less important words. In our model, we used multiple attention heads, thus considering multiple representations built from the input sequence (i.e., different weights, learned independently in each attention head, associated with the different word positions). The output that is produced from the multi-head attention mechanism corresponds to the concatenation of the weighted vectors produced by each head.

\section{3.2 Modeling Visual Contents}

Deep learning architectures for processing visual inputs often involve stacking convolutional and pooling layers in a network for producing an intermediate representation, followed by fully-connected layers that produce a final classification. As Convolutional Neural Networks (CNNs) become deeper, the number of parameters continues to grow, therefore making it challenging to train.

For instance, (He et al., 2016) introduced Residual Networks (ResNets), i.e. CNNs that aim to avoid problems such as vanishing gradients during the training of deep models, by introducing direct identity connections that skip one or more convolutional layers, forming a residual block. As described in Fig-
These skip connections consist in adding the output of previous convolutional layers together with the output of the current layer, creating paths propagating information from earlier to later layers.

The blocks used in ResNet architectures also apply a bottleneck operation, which consists in a $1 \times 1$ convolution before each $3 \times 3$ convolution to reduce the number of input feature-maps, and thus to improve computational efficiency. Using the aforementioned block, He et al. also proposed complete CNN architectures for image classification. For instance ResNet-50 which is illustrated on Figure 3.5, consists of a BN-Relu-Conv block, followed by a max-pooling layer, and 4 stages of bottleneck residual blocks, before a final average pooling layer.

In turn, (Huang et al., 2017) presented Dense Convolutional Networks (DenseNets), i.e. CNNs that use additional connections to alleviate vanishing gradient problems by strengthening feature propagation and reuse. In their approach, each convolutional layer input consists in the concatenation of all the output feature maps of all the preceding convolutional layers. This way, the output feature maps of a given layer will be used as part of the inputs of all the subsequent layers. This shorter links between initial and later layers enable CNNs that can be significantly deeper, more accurate, and efficient to train. Figure 3.6 provides an illustration for the main block used in DenseNet architectures, together with the DenseNet-201 architecture which consists of a BN-ReLU-Conv block, followed by a max-polling layer, and 4 stages of dense and transition blocks (i.e.,), before a final average pooling layer. A transition block is a BN-ReLU-Conv block followed by an average pooling layer.

More recently, (Chen et al., 2017) proposed Dual Path Networks (DPNs), i.e. CNNs that combine ideas from both the ResNet and DenseNet architectures. As described in Figure 3.7, the orange feature maps from DenseNet are mostly reused and refined, while the blue feature maps from ResNet keep stacking new features. The blocks used in DPN architectures combine both approaches, by extending blocks used in ResNet architectures with a slice and concatenation operation.

(Tan and Le, 2019) introduced the Efficient Network (EfficientNet) neural architecture, combining a particular block with a novel CNN architecture scaling method named compound scaling. The original main building block combines squeeze-and-excitation operations, as described by (Hu et al., 2018), to-
Figure 3.7: The Dual Path Network block.

Figure 3.8: Illustration of the depthwise separable convolution block.

gether with a mobile inverted bottleneck convolution (MBConv) block, originally described by Tan et al. (2019) and Sandler et al. (2018) and which uses separable depthwise convolutions. Figure 3.9 provides an illustration of the MBConv block followed by the squeeze-and-excitation mechanism (Hu et al., 2018). Opposing the original residual block approach, where a high number of channels is compressed and then expanded with an $1 \times 1$ convolution for better computational efficiency, the inverted residual block expands and then compresses a low number of channels with an $1 \times 1$ convolution, since the Depthwise separable convolution already significantly reduces the number of parameters. The Depthwise convolution used differs from standard convolution by splitting both the input feature maps and kernel filters into $n$ channels. For each channel, convolves the corresponding input and filter split, stacking all the split output tensors. Depthwise separable convolution, extends the depthwise convolution by adding an extra step: a $1 \times 1$ convolution across channels. Figure 3.8 provides an illustration of this procedure.

The Squeeze-and-Excitation mechanism opposes the standard mechanism that equal weights each feature map channel when creating the final output feature maps. Using average pooling, followed by a nonlinear fully connected layer and a sigmoid activation, this procedure provides each channel a smooth gating mechanism similar to a content aware mechanism that weights each feature map channel individually.

The multi-objective neural architecture search optimizes both FLOPS and accuracy, while scaling the architecture depth, width, and resolution with a constant ratio $\phi$ (e.g. EfficientNet-B5 scaled from EfficientNet-B0 with $\phi$ equal to 5). This approach can be formally defined as follows, where $\alpha, \beta, \gamma$ are...
constants that were determined by a small grid search on the original small EfficientNet-B0 model, with
value 1.2, 1.1 and 1.15, respectively.

\[
\begin{align*}
d &= \alpha^\phi \\
w &= \beta^\phi \\
r &= \gamma^\phi \\
\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 \\
\alpha &\geq 1, \beta \geq 1, \gamma \geq 1
\end{align*}
\] (3.6)

EfficientNet-B5 results from scaling the EfficientNet-B0 model using 1.6 and 2.2 as the width and
depth coefficients.

### 3.3 Per-Modality Model Pre-Training

As an alternative of randomly initializing all the parameters of the proposed multi-modal architecture,
both the text and image processing paths were pre-trained using single modality large datasets. More
concretely, MIMIC-CXR (Johnson et al., 2019) and CheXpert (Irvin et al., 2019) datasets were shuffled
and used to pre-train and evaluate each CNN architectures described in Section 3.2. In order to pre-train
and evaluate the different methods for free-text classification proposed in Section 3.1.1, we leveraged
the MIMIC-III (Edward William Johnson et al., 2016) dataset, filtered to contain only chest radiology
reports. After pre-training this models, the final sigmoid layer is ignored and the remaining components
are re-used on the complete multi-modal architecture.
3.4 Summary

This chapter described the novel multi-modal deep learning approach for classifying chest radiology exams, detailing each component. Section 3.1 provided an overview of the general multi-modal architecture. In summary, the text path uses word embeddings to represent each word, this vectors are then processed by recurrent units, and an attention mechanism weights the importance of each word, generating a vector representation of the report; and the image path generates a vector representation of the radiography, by leveraging a deep convolutional neural architecture. This multi-modal representations are then concatenated to support the final layer of classification. Section 3.1.1 detailed the free-text processing path of the network, exploring pre-trained biomedical contextual and non-contextual word embeddings, Bidirectional Multiplicative Long Short-Term Memory (bi-mLSTM) units, and a multi-head attention mechanism. Section 3.2 presented image processing path of the network, exploring different deep convolutional neural architectures, such as: ResNet-50, DenseNet-201, Dual Path Network-92, and EfficientNet-b5. The process of pre-training individually both text and image paths of the architecture, leveraging different single-modality datasets(e.g., MIMIC-CXR, CheXpert, MIMIC-III, and Open-i), was described in Section 3.3.
Experimental Evaluation

Contents

4.1 Datasets and Experimental Methodology ........................................... 48
4.2 Experimental Results ........................................................................... 49
4.3 Interpreting Results by Visualizing mechanisms ................................. 53
4.4 Summary ............................................................................................. 55
Table 4.1: Statistical characterization of the datasets used in the experiments.

<table>
<thead>
<tr>
<th>Label</th>
<th>Images</th>
<th>Text</th>
<th>Multi-Modal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MIMIC-CXR</td>
<td>CheXpert</td>
<td>MIMIC-III</td>
</tr>
<tr>
<td>No Finding</td>
<td>83,336</td>
<td>19,765</td>
<td>133,563</td>
</tr>
<tr>
<td>Cardiomegaly</td>
<td>18,240</td>
<td>19,578</td>
<td>608</td>
</tr>
<tr>
<td>Lung Opacity</td>
<td>56,012</td>
<td>30,158</td>
<td>351</td>
</tr>
<tr>
<td>Lung Lesion</td>
<td>60,196</td>
<td>98,759</td>
<td>62</td>
</tr>
<tr>
<td>Edema</td>
<td>8,315</td>
<td>8,149</td>
<td>9,741</td>
</tr>
<tr>
<td>Consolidation</td>
<td>43,812</td>
<td>61,535</td>
<td>4,635</td>
</tr>
<tr>
<td>Pneumonia</td>
<td>16,614</td>
<td>37,396</td>
<td>43,006</td>
</tr>
<tr>
<td>Atelectasis</td>
<td>61,108</td>
<td>59,658</td>
<td>287</td>
</tr>
<tr>
<td>Pneumothorax</td>
<td>12,953</td>
<td>20,408</td>
<td>16,241</td>
</tr>
<tr>
<td>Pleural Effusion</td>
<td>65,449</td>
<td>86,541</td>
<td>29,978</td>
</tr>
<tr>
<td>Pleural Other</td>
<td>3,009</td>
<td>4,318</td>
<td>896</td>
</tr>
<tr>
<td>Fracture</td>
<td>5,675</td>
<td>7,935</td>
<td>16,862</td>
</tr>
<tr>
<td>Support Devices</td>
<td>74,970</td>
<td>108,184</td>
<td>10,504</td>
</tr>
<tr>
<td><strong>Total Instances</strong></td>
<td>250,044</td>
<td>191,229</td>
<td>261,091</td>
</tr>
</tbody>
</table>

This chapter explains the experimental evaluation of the proposed deep learning architecture. Section 4.1 describes the different datasets that supported the experiments, jointly with the respectively experimental methodology. Section 4.2.1 details the obtained results over the MIMIC-CXR and CheXpert datasets, while Section 4.2.2 presents the obtained results over MIMIC-III dataset. Section 4.2.3 lists the obtained results over Open-i dataset. Section 4.3 is centered on the visualization mechanisms proposed in this work, illustrating its relevance in improving the interpretation of the results of the model. Lastly, Section 4.4 presents a summary of the chapter, providing an overview of the obtained results.

### 4.1 Datasets and Experimental Methodology

Our experiments relied on four different datasets, including two datasets containing frontal chest X-ray images labeled according to 14 observation classes (i.e., CheXpert (Irvin et al., 2019) and the first version of the MIMIC-CXR (Johnson et al., 2019) dataset), one dataset of full-text radiology reports (i.e., a subset of the data from MIMIC-III (Edward William Johnson et al., 2016) with the ICD labels converted to the same 14 observation classes), and the Open-i (Demner-Fushman et al., 2016a) multimodal radiography dataset, collected from multiple institutes by Indiana University and containing frontal chest X-ray images together with full-text reports, also labeled according to the same 14 observation classes. Table 4.1 provides general data characterization statistics.

In more detail, the MIMIC-CXR dataset includes 371,920 chest X-rays, associated with 227,943 studies involving patients admitted to the Beth Israel Deaconess Medical Center, between 2011 and 2016. In turn, CheXpert contains 224,316 chest X-rays from 65,240 patients, collected from Stanford
Hospital between October 2002 and July 2017. In both cases, we considered only frontal view X-ray images (i.e., 250,044 instances from MIMIC-CXR, and 191,229 instances from CheXpert), merging both datasets for the experiments involving image data (i.e., when evaluating the different CNN architectures alone, or for pre-training the complete model in the case of multi-modal tests).

MIMIC-III is a freely accessible critical care database containing, among other elements, radiology reports associated with ICD diagnostic codes (taken from patient discharge notes). We filtered the radiology reports according to the occurrence of key-phrases such as chest, lungs or thorax, and using the ICD codes that result from matching the 14 labels of the MIMIC-CXR dataset with the correspondent set of ICD codes. This resulted in a set of 261,091 textual documents.

Finally, the complete Open-i dataset includes 3,851 radiology reports and the associated 7,784 chest X-ray images. We filtered the dataset to consider only 3,689 instances containing full-text reports and frontal X-ray images, labeled according to the 14 labels from MIMIC-CXR/CheXpert.

Taking into account the multi-label nature of the datasets, we considered stratification when dividing the data (i.e., the merging of MIMIC-CXR and CheXpert, or the subsets from MIMIC-III and Open-i) into training (64%), validation (16%), and testing (20%) splits. The considered multi-label stratification method balances the assignment of instances into splits, attending to the distribution of single classes and multi-class pairs (Sechidis et al., 2011; Szymański and Kajdanowicz, 2017).

Model training when leveraging the text datasets was made with batches of 64 instances, while models leveraging image and multi-modal inputs were trained with batches of 16 instances, in both cases leveraging back-propagation together with the Adam optimization method (Kingma and Ba, 2015). In order to use pre-trained ImageNet models, the input images were re-scaled to $256 \times 256$ matrices. The learning rate was initially set to $10^{-3}$, and then refined through a cyclic pattern (Smith, 2017). The number of epochs was also defined through a criteria based on a validation loss, stopping when the variation between consecutive epochs was less than $10^{-6}$.

To assess the quality of the predictions, we used the following metrics: accuracy (i.e., average number of correct labels per instance), coverage error (CE) (i.e., how many labels, ranked according to prediction scores, need to be checked to cover all the true labels), label ranking average precision (LRAP), micro and macro-averaged scores for multi-label precision, recall, F1 scores, and areas under ROC curves (AUROCs).

4.2 Experimental Results

4.2.1 Radiography Classification Task

For the radiography chest X-ray image classification task, the tests considered two different settings: (i) use the DPN-92, ResNet-50, DenseNet-101, and EfficientNet-B5 architectures to classify X-ray
Table 4.2: Results for the MIMIC-CXR/CheXpert dataset experimental settings.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>LRAP</th>
<th>CE</th>
<th>Precision Micro</th>
<th>Macro</th>
<th>Recall Micro</th>
<th>Macro</th>
<th>F1-score Micro</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPN-92</td>
<td>0.8174</td>
<td>0.5033</td>
<td>6.2743</td>
<td>0.1106</td>
<td>0.0728</td>
<td>0.5158</td>
<td>0.1702</td>
<td>0.1822</td>
<td>0.0694</td>
</tr>
<tr>
<td>DenseNet-201</td>
<td>0.8286</td>
<td>0.5996</td>
<td>6.3484</td>
<td>0.2364</td>
<td>0.1301</td>
<td>0.5838</td>
<td>0.3275</td>
<td>0.3365</td>
<td>0.1307</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>0.8333</td>
<td>0.6116</td>
<td>6.3310</td>
<td>0.3973</td>
<td>0.2071</td>
<td>0.5665</td>
<td>0.2891</td>
<td>0.4670</td>
<td>0.1922</td>
</tr>
<tr>
<td>EfficientNet-95</td>
<td>0.8467</td>
<td>0.6874</td>
<td>5.5840</td>
<td>0.3062</td>
<td>0.1889</td>
<td>0.6600</td>
<td>0.3996</td>
<td>0.4234</td>
<td>0.2267</td>
</tr>
<tr>
<td>ResNet-50 w/ ImageNet pre-train</td>
<td>0.8435</td>
<td>0.6440</td>
<td>5.9882</td>
<td>0.3895</td>
<td>0.2075</td>
<td>0.6180</td>
<td>0.3252</td>
<td>0.4778</td>
<td>0.2294</td>
</tr>
<tr>
<td>DenseNet-201 w/ ImageNet pre-train</td>
<td>0.8466</td>
<td>0.6867</td>
<td>5.6045</td>
<td>0.2915</td>
<td>0.1656</td>
<td>0.6986</td>
<td>0.3687</td>
<td>0.4114</td>
<td>0.1864</td>
</tr>
<tr>
<td>EfficientNet-B5 w/ ImageNet pre-train</td>
<td>0.8637</td>
<td>0.7248</td>
<td>5.2662</td>
<td>0.4529</td>
<td>0.2734</td>
<td>0.6998</td>
<td>0.4669</td>
<td>0.5499</td>
<td>0.3075</td>
</tr>
</tbody>
</table>

grayscale images trained with the combined MIMIC-CXR/CheXpert data, and (ii) use the ResNet-50, DenseNet-101, and EfficientNet-B5 models pre-trained with ImageNet to classify X-ray RGB images trained with the combined MIMIC-CXR/CheXpert data.

Table 4.2 presents the results obtained over the testing data splits. The first four rows correspond to Setting (i), the following three rows correspond to Setting (ii). In the first experimental setting’s results, EfficientNet-B5 achieved consistently better results in terms of almost all the different metrics, while the DPN-92 achieved consistently worst results in terms of almost all the different metrics. The second experimental setting’s results demonstrate that leverage public available ImageNet pre-trained weights, instead of randomly initialized weights, and leverage the 3-channels image data (e.g. RGB) instead of 1-channel image data (e.g. grayscale), can improve the overall performance of these models. For instance, EfficientNet-B5 achieved consistent better results in terms of almost all the different metrics with settings (ii), when compared with settings (ii). Figure 4.1 provides extra details, by illustrating the AUROC values of each class for the different models. EfficientNet-b5 outperformed the remaining models, accomplish 0.869 and 0.792, in micro and macro average AUROC.

4.2.2 Radiology Reports Classification Task

Table 4.3 presents results for the free-text radiology report classification task in the MIMIC-III dataset, including results from the model with Bi-directional Multiplicative LSTMs followed by the Multi-Head Attention mechanism, leveraging as word embeddings: (a) BioWordVec, (b) BioELMo, and (c) BioBERT. The third experimental set of results show that the model with BioWordVec embeddings achieved better results then those achieved with BioELMo and BioBert contextual embeddings. Although these models have shown superior performance in other tasks, it is worth mentioning that methods such as ELMo take character information to construct word representations, therefore attending the problem of out-of-

Table 4.3: Results for the MIMIC-III dataset experimental settings.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>LRAP</th>
<th>CE</th>
<th>Precision Micro</th>
<th>Macro</th>
<th>Recall Micro</th>
<th>Macro</th>
<th>F1-score Micro</th>
<th>Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-mlLSTM-A-BioBERT</td>
<td>0.9385</td>
<td>0.8131</td>
<td>2.1081</td>
<td>0.5864</td>
<td>0.3318</td>
<td>0.7153</td>
<td>0.5640</td>
<td>0.6463</td>
<td>0.4006</td>
</tr>
<tr>
<td>Bi-mlLSTM-A-BioELMo</td>
<td>0.9441</td>
<td>0.8309</td>
<td>2.0336</td>
<td>0.6239</td>
<td>0.3770</td>
<td>0.7482</td>
<td>0.6207</td>
<td>0.6805</td>
<td>0.4504</td>
</tr>
<tr>
<td>Bi-mlLSTM-A-BioWordVec</td>
<td>0.9544</td>
<td>0.8654</td>
<td>1.8718</td>
<td>0.6995</td>
<td>0.4471</td>
<td>0.7973</td>
<td>0.7433</td>
<td>0.7452</td>
<td>0.5325</td>
</tr>
</tbody>
</table>
vocabulary words to some extent. However, the individual character information is an insufficient and unnatural linguistic unit for word representation, when compared with FastText embeddings character $n$-grams to model out-of-vocabulary tokens. The better performance of the BioELMo model, when compared with BioBERT in this classification task, goes in line with the conclusions of Jin et al. (2019), which showed that although fine-tuned BioBERT is better than BioELMo in biomedical named entity recognition and natural language inference tasks, as a fixed feature extractor BioELMo outperforms BioBERT in their probing tasks (i.e., tasks that probe token representations for linguistic properties). Figure 4.2 illustrates the AUROC values of each class for the different models. Bi-mLSTM-A-BioWordVec outperformed the remaining models, accomplish 0.969 and 0.922, in micro and macro average AUROC.
Figure 4.1: A comparison of classification performance with different word embeddings.

4.2.3 Radiology Exams Multi-Modal Classification Task

Table 4.4 presents results for the multi-modal chest radiology exam classification task in the Open-i dataset, including results from: single modality classification: (a) Bi-mLSTM-A-BioWordVec, (b) Efficient Network B5 with ImageNet pre-trained weights, and multi-modality classification: (c) Multi-modal approach without pre-trained weights, and (d) Multi-modal approach fine tuned and with pre-trained weights.

The complete model, leveraging both modalities with pre-trained weights and fine-tuned with the Open-I training split, achieved a very high performance in terms of the different metrics, outperforming methods using just a single modality, and/or methods without model pre-training. Pre-training, in particular, contributed significantly to the overall performance of the complete model (e.g., 2% improvement in accuracy, and 14% improvement in macro-AUROC). Figure 4.3 illustrates the AUROC values of each class for the different models. Multi-modal approach fine tuned and with pre-train weights outperformed
Figure 4.3: A comparison of classification performance with different modality and pre-training settings models.

the remaining models, accomplish 0.987 and 0.951, in micro and macro average AUROC.

Table 4.4: Results for the Open-i dataset experimental settings.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>LRAP</th>
<th>CE</th>
<th>Precision Micro</th>
<th>Precision Macro</th>
<th>Recall Micro</th>
<th>Recall Macro</th>
<th>F1-score Micro</th>
<th>F1-score Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-mLSTM-A-BioWordVec</td>
<td>0.9713</td>
<td>0.6788</td>
<td>3.9652</td>
<td>0.4096</td>
<td>0.1267</td>
<td>0.5964</td>
<td>0.2594</td>
<td>0.4857</td>
<td>0.1342</td>
</tr>
<tr>
<td>EfficientNet-B5</td>
<td>0.9223</td>
<td>0.6852</td>
<td>4.1379</td>
<td>0.4397</td>
<td>0.1676</td>
<td>0.6329</td>
<td>0.3324</td>
<td>0.5189</td>
<td>0.1909</td>
</tr>
<tr>
<td>BioWordVec-EfficientNet w/o pre-train</td>
<td>0.9634</td>
<td>0.8976</td>
<td>2.0134</td>
<td>0.7299</td>
<td>0.2770</td>
<td>0.8664</td>
<td>0.3660</td>
<td>0.7919</td>
<td>0.3067</td>
</tr>
<tr>
<td>BioWordVec-EfficientNet</td>
<td>0.9795</td>
<td>0.9452</td>
<td>1.6653</td>
<td>0.8654</td>
<td>0.5538</td>
<td>0.9150</td>
<td>0.7125</td>
<td>0.8895</td>
<td>0.6023</td>
</tr>
</tbody>
</table>

4.3 Interpreting Results by Visualizing mechanisms

In addition to near real-time surveillance of specific lung and heart diseases, the proposed architecture can also be useful for assisting human coders, by providing coding suggestions that are interpretable. In connection to the classification results, the proposed model also allows us to explore two distinct in-
terpretability views. Using the gradient-weighted class activation mapping (Grad-CAM), i.e., a technique that leverages the gradients of a chosen target class, flowing into the final convolutional layer to construct a localization map emphasizing the important regions in the image for that specific class classification, it is possible to visualize which areas of the X-ray were more important to the classification decision (Selvaraju et al., 2017). Simultaneously, using the weights from the multi-head attention mechanism, it is possible to visualize which words in the full-text report were more important to the classification.

Figure 4.4 portrays three example instances from the Open-i test split, correctly assigned to the classes No Findings, Cardiomegaly, and Fracture, respectively. Figures 4.4(a), 4.4(b), and 4.4(c) presents the original X-ray images overlayed with the Grad-CAM heatmaps, emphasizing the areas that were more important to support each classification decision. In both examples, the heatmaps adopt a thorax shape and highlight the areas around the lungs, heart and ribs, regions relevant to the corresponding true labels analysis. Figures 4.4(d), 4.4(e), and 4.4(f) illustrates the attention weights computed by the model by coloring boxes over the texts, denoting word attention weights according to two of the attention heads (i.e., blue and green colours). In both examples, the free-text reports contain words that can be highly indicative of the disease, and that the method assigned a higher weight. Figure 4.4(e) instance with true label: Cardiomegaly, gave higher weights to words describing the cardiomegaly condition, such as heart size is mildly enlarged, and the concept itself: cardiomegaly. Another example, Figure 4.4(f) instance with true label: Fracture, gave higher weights to words describing the fracture, such as rib fractures.
4.4 Summary

This chapter explained the experiments performed for the evaluation of the proposed deep learning architecture. Section 4.1 presented the different datasets that supported the experiments, jointly with the respectively experimental methodology. Section 4.2.1 displayed the obtained results over the MIMIC-CXR and CheXpert datasets, the ImageNet pre-trained models achieved better performance, while Section 4.2.2 presented the obtained results over MIMIC-III dataset, BioWordVec outperformed other models leveraging the benefit of using n-gram embeddings over character embeddings. Section 4.2.3 listed the obtained results over Open-i dataset, the complete model outperformed other methods using just a single modality, and/or methods without model pre-training. Lastly, Section 4.3 presented some multi-modal examples of Open-i dataset, explaining how the visualization and attention mechanisms proposed in this work can be used to provide some interpretability to the obtained results, illustrating how different words and pixels were attended in each prediction.
Conclusions and Future Work

Contents

5.1 Summary of Main Conclusions ........................................... 57
5.2 Future Work ................................................................. 58
This dissertation achieved as the main contribution: a multi-modal deep learning approach to automatically classify chest radiology exams. Currently, several deep learning architectures leveraged convolutional neural networks Ge et al. (2018); Laserson et al. (2018); Wang et al. (2017); Yan et al. (2018), and recurrent and attention units Duarte et al. (2018); Karimi et al. (2017); Peng et al. (2019b) for classifying X-ray images, and biomedical records, although very few studies have specifically addressed the multi-modal classification of chest radiology exams Wang et al. (2018). This chapter summaries the main conclusions and contributions, and highlights possible directions for future work on this multi-modal classification task.

5.1 Summary of Main Conclusions

This dissertation presented a novel deep learning method for the multi-label classification of data from radiology exams according to 14 observation classes, combining both X-ray images and full-text reports.

For processing image data, from the convolutional neural networks (CNNs) experimented: Dual Path Networks (DPNs) Chen et al. (2017) with random initialized weights, and Residual Neural Networks (ResNets) He et al. (2016), Densely Connected Convolutional Networks (DenseNets) Huang et al. (2017), and EfficientNet Tan and Le (2019) architectures pre-trained in the ImageNet Deng et al. (2009) image dataset, EfficientNet-b5 outperformed the remaining models, accomplish 0.869 and 0.792, in micro and macro average AUROC in the MIMIC-CXR/CheXpert dataset.

For modeling free-textual data, from the three different biomedical pre-trained word embeddings tested, namely BioWordVec Chen et al. (2018), BioELMo Jin et al. (2019), and BioBERT Lee et al. (2019), together with Bidirectional Multiplicative Long Short-Term Memory (bi-mLSTM) units Krause et al. (2017) for creating intermediate representations, which are finally combined through a multi-head attention mechanism Vaswani et al. (2017), Bi-mLSTM-A-BioWordVec outperformed the remaining models, accomplish 0.969 and 0.922, in micro and macro average AUROC in the MIMIC-III dataset.

We validated the high performance of the individual components of the multi-modal end-to-end architecture. Pre-training, in particular, contributed significantly to the overall performance of the complete model. The complete model fine-tuned and with single modality pre-trained weights outperformed the remaining models, accomplish 0.987 and 0.951, in micro and macro average AUROC.

One of the major drawbacks of applying machine learning in clinical settings is the capability to explain the models. In our approach, special focus was given to the attention mechanisms to allow visual interpretation of the results, building the basis to allow translating the developed work to clinical validation. This is required in order to understand if the model is learning the correct features from the images and text. For example, if the classified image as fracture is actually detecting the fracture in the correct place. This must be performed with multiple physicians in order to be able to capture difference
in performance between experts (Brady, 2017).

The obtained results indicate that pre-train the models have a high impact on models performance. The proposed approach used ImageNet pre-trained weights, ImageNet is a public dataset that does not consider any medical imaging data. The impact on the task might indicate that the learned features to classify day-to-day images might be similar to the ones classifying X-rays. The same dataset was also used to explore image reconstruction in MRI (Zhu et al., 2018).

5.2 Future Work

Despite the interesting results, there are also other approaches to explore for future work. For example, as an alternative of using BioBERT and BioELMo as fixed feature extractors, fine-tune both while training on the downstream task of classifying chest radiology reports. For instance, instead of mLSTM units, the recently proposed Mogrified LSTM units from Melis et al. (2019) can perhaps be used instead. This particular approach has achieved results in language modeling tasks similar to state-of-the-art Transformer models, having the benefit of having considerably fewer parameters. In order to increase the performance of recurrent models such as mLSTMs, Gu et al. (2019) proposed gating mechanisms that robustly increase the performance of LSTMs on tasks requiring long temporal dependencies. The recent release of MIMIC-CXR (Johnson et al., 2019) dataset, which incorporates X-ray images together with the full-text reports, can be used to further evaluate the proposed model and enables new experiments using multimodal bitransformers (Kiela et al., 2019), which have achieved promising results on classification of images and text, and Unicoder-VL (Li et al., 2019), a universal encoder that aims to learn joint representations of vision and language in a pre-training manner. Guendel et al. (2019) proposed a multi-task deep learning architecture for classifying X-ray images that concurrently support disease classification, lung and heart segmentation, and disease’s location classification task. The study demonstrated that training concurrently a multi-task neural network, one can improve the classification performance of the model.
Bibliography


61


