Eduardo de Souza Pais

INTELLIGENT DOCUMENT VALIDATION USING NATURAL LANGUAGE PROCESSING AND COMPUTER VISION

INTELLIGENT DOCUMENT VALIDATOR

UNIVERSIDADE D COIMBRA

1 2 9 0

UNIVERSIDADE D
COIMBRA

Eduardo de Souza Pais

Intelligent Document Validation

INTELLIGENT DOCUMENT VALIDATION USING NATURAL LANGUAGE PROCESSING AND COMPUTER VISION

Internship report in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems made at Critical Software, advised by Doctor Tiago Baptista (Critical Software) and by Professor Doctor António Dourado (DEI-FCTUC) and presented at Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2021

Faculty of Sciences and Technology

Department of Informatics Engineering

Intelligent Document Validation

Intelligent Document Validation using Natural Language Processing and Computer Vision

Eduardo de Souza Pais

Internship report in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems made at Critical Software, advised by Doctor Tiago Baptista (Critical Software) and by Professor Doctor António Dourado (DEI-FCTUC) and presented at Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2021

UNIVERSIDADE D COIMBRA

Abstract

Processes in organizations over the past few years have been increasingly automated. In order to make them more efficient and practical. However, one area in which manual work is still common is document analysis. In this area, due to the ubiquity brought by electronic means, the submission of documents has been made, primarily, in digital format. Human intervention is still frequent in the analysis of these documents for tasks such as validation, information extraction and classification. This manual analysis has high costs in terms of time, performance, and possibility of human error which could have serious consequences in critical environments.

Critical Software (CSW) has currently under development a solution that addresses this problem, using technologies in the area of Computer Vision (CV), Machine Learning (ML) and Natural Language Processing (NLP). The solution consists of an Intelligent Document Validation (IDV) system that validates the authenticity of the submitted documents and also extracts useful information from them in order to make the process more efficient and less susceptible to errors.

At this internship the objective was to develop a hybrid IDV solution, which uses both textual and visual characteristics for document classification, and that ensures performance in relation to current models and, simultaneously, ensure robustness in the training of the IDV for new types of documents. This challenge is the main focus of the internship at CSW company lasting one academic year. In summary, the stages of development are: individual training and optimization of textual and image-based models using transfer learning with convolutional neural networks and transformer models, followed by the creation of ensemble models using voting and finally hybrid modelling using a late fusion method, ending on results analysis.

Keywords

Machine Learning, Deep Learning, Computer Vision, Natural Language Processing, Optical Character Recognition

Resumo

Processos em organizações ao longo dos últimos anos têm sido cada vez mais automatizados de forma a torná-los mais eficientes e práticos. No entanto, uma área em que o trabalho manual ainda é comum é a de análise de documentos. Nesta área, devido à ubiquidade trazida por meios eletrónicos, a submissão de documentos tem sido feita principalmente em formato digital. Intervenção humana ainda é frequente na análise destes documentos para tarefas como validação, extração de informação e classificação. Esta análise manual tem custos elevados em termos de tempo, desempenho e possibilidade de erro humano que pode ter consequências graves em ambientes críticos.

Critical Software (CSW) tem atualmente em desenvolvimento uma solução que aborda este problema, utilizando tecnologias da área de Visão Computacional (CV), Aprendizagem Computacional (ML) e Processamento de Linguagem Natural (NLP). A solução consiste num sistema de Validação Inteligente de Documentos (IDV) que valida a autenticidade dos documentos submetidos e também extrai informação útil dos mesmos de forma a tornar o processo mais eficiente e menos suscetível a erros.

Neste estágio o objetivo foi de desenvolver uma solução IDV híbrida, que utilizou características textuais e visuais para a classificação de documentos de forma a melhorar o desempenho em relação aos modelos atuais e que assegure robustez em relação a novos tipos de documentos. Este desafio foi o foco principal do estágio na empresa CSW com duração de um ano letivo. Em resumo, as etapas de desenvolvimento foram: treino e otimização individual de modelos textuais e visuais recorrendo a aprendizagem por transferência com redes convolucionais e transformadores, seguido da criação de modelos de *ensemble* a partir de métodos de votação e por último a criação de modelos híbridos recorrendo a um processo de fusão tardia, e terminou com uma análise dos resultados.

Palavras-Chave

Aprendizagem Computacional, Aprendizagem Profunda, Visão Computacional, Processamento de Linguagem Natural, Reconhecimento Ótico de Caracteres

Acknowledgements

I would like to acknowledge and give thanks to my CSW adviser Doctor Tiago Baptista for his guidance, especially for his patience and understanding in times of struggle during this project. This thanks also extends to my colleague at CSW, Ana Guarino whose guidance was a contributing factor to make this work possible. Both were always available to discuss ideas and solutions and advise whenever needed, throughout all stages of the project.

I would like to acknowledge and give thanks to my advisor and professor Doctor António Dourado for guiding me but also for teaching me the skills needed prior to the development of this project, when I was his student.

I would also like to give my warmest thanks to my family, especially my mom, Janete de Souza, whose constant support made it possible for me to continue my studies – without her, getting here would have been nigh impossible.

Contents

Acronyms

AI – Artificial Intelligence

ANN – Artificial Neural Network

CNN – Convolutional Neural Network

CSW – Critical Software

CV – Computer Vision

DL – Deep Learning

FC – Fully Connected

FFNN – Feed-forward neural network

GRU – Gated Recurring Unit

IDV – Intelligent Document Validation

LSTM – Long Short-Term Memory

ML – Machine Learning

NLP – Natural Language Processing

OCR – Optical Character Recognition

OOV – Out of Vocabulary Words

ReLu – Rectified Linear Unit

RNN – Recurrent Neural Network

SVM – Support Vector Machines

BERT – Bidirectional Encoder Representations from Transformers

List of Figures

List of Tables

Chapter 1 Introduction

This project is integrated in the Master of Informatics Engineering course with specialization in Intelligent Systems with estimated duration of one academic year.

The presented curricular internship took place at Critical Software (CSW) and is supervised by advisor Doctor António Dourado, professor at University of Coimbra and by Doctor Tiago Baptista, tutor at CSW.

In this internship I will integrate the current Intelligent Document Validation (IDV) Team and help develop new models to classify documents using both its image and extracted text.

In this first chapter it is presented a general overview of the document: in section 1.1 the context in which the internship takes place as well as the motivations for tackling its respective challenges; in section 1.2 it is described the internship's main objectives; in section 1.3 it is detailed the work done as well the knowledge gained during the academic year, as well as the planning; in section 1.4 it is described the general structure of the document.

1.1 Context and Motivations

Following the mass digitalization of information in many organizations, there has been a growing need for documentation management. This process involves two major challenges: document classification and extraction. Document classification consists of labelling documents in a certain category based on their textual or visual attributes. Document or data extraction consists of the extraction of a document's relevant data which is often unstructured because of the common use of natural

language on them. A need for the automation of these processes has risen throughout the years due to the fact that they are still, largely, laboured manually which is prone to significant errors. This manual labour is also highly repetitive and often slow. In the effort of reducing error susceptibility and increasing efficiency and confidence, many of these processes are gradually becoming automated, complementing or completely replacing the previous manual labor.

Documents, unlike general images, have varied and structured forms which makes difficult the task of accurately extracting information from different types of documents. A lot of research has been done on this issue and has resulted in the development of applications such as intelligent document classifiers and processors with the purpose of accurately classifying and analyzing documents such as CSW IDV.

1.2 Goals

The main objective of this internship is to create and integrate in the current IDV platform, new machine learning solutions and models that improve the existing ones and allow automation of the training process as well, if possible, for new types of documents, in regard to their classification.

As a brief overview of CSW IDV platform, the IDV has three major services: classification, extraction and validation. Classification service is achieved by relying on either the document's visual or textual features to predict and classify a document's category. While document data extraction services are achieved through the collection of relevant data, often structured and categorized. Document validation, the final service, checks if the last two procedures were successful.

At the moment the platform has many document classification algorithms ranging from textual classifiers, image classifiers as well as rule-based classifiers and has the versatility to integrate new and different types of classifiers.

The CSW IDV platform is currently being used, mainly, on structured documents, where data is organized and its relations are well defined, such as in a form format which is the case of passports, certificates, invoices, citizen cards and other types of documents. Having said this, the platform is also able to deal with unstructured data through the use of natural language processing algorithms.

The main goal is to provide this service with a classifier that follows a hybrid feature learning approach using both textual and visual data to achieve document classification. This hybrid approach has been the target of research in recent works with promising results, detailed in section 2.5. The models, if successful, will be considered for integration into the current IDV classification service.

1.3 Internship Plan

In the first semester the first task, defined in the initial Sprint session, was to define the scope of the project. This included establishing an initial plan of work as well as getting adapt with CSW tools of communication, resource management and development. The second task was to acquire theoretical and practical knowledge about the fundamental areas of study regarding document classification, more specifically on machine learning, computer vision and natural language processing. The third task was to study the state of the art of document classification. This included the study of works regarding textual document classification, visual document classification and also existing hybrid classification approaches. The fourth task, to be done simultaneously, was to study the current IDV platform in how it operates, its features and functionalities and how the created machine learning models would need to be fit for integration. The fifth task was to start writing the intermediate dissertation, including the technical specification. In Figure 1 it is presented the GANTT diagram, created during planning, regarding the previous first semester.

For the second semester, in first sprint, a planning meeting was held to discuss future work. Followed by another review sprint and the main task during these initial sprints will be to set up the research and development environments. The next task was to develop a baseline model to be reviewed in the following sprints. Afterwards development of the models will begin alongside with simultaneous experimentation. Three phases of development are expected with each representing the development of one of three main components: textual, visual and hybrid components. In the first phase it is expected the development of one of two main components (textual or visual). The second version consists of the development of the next component and finally the third consists of the development of the hybrid component. All sprints in between will serve to review work done and dynamically plan the development of each component. The next step planed after development is a final round of testing and validation. If successful, possible integration into CSW IDV was considered. The final task was to write the internship report. In Figure 2 it is presented the GANTT diagram, created during planning, regarding the second semester.

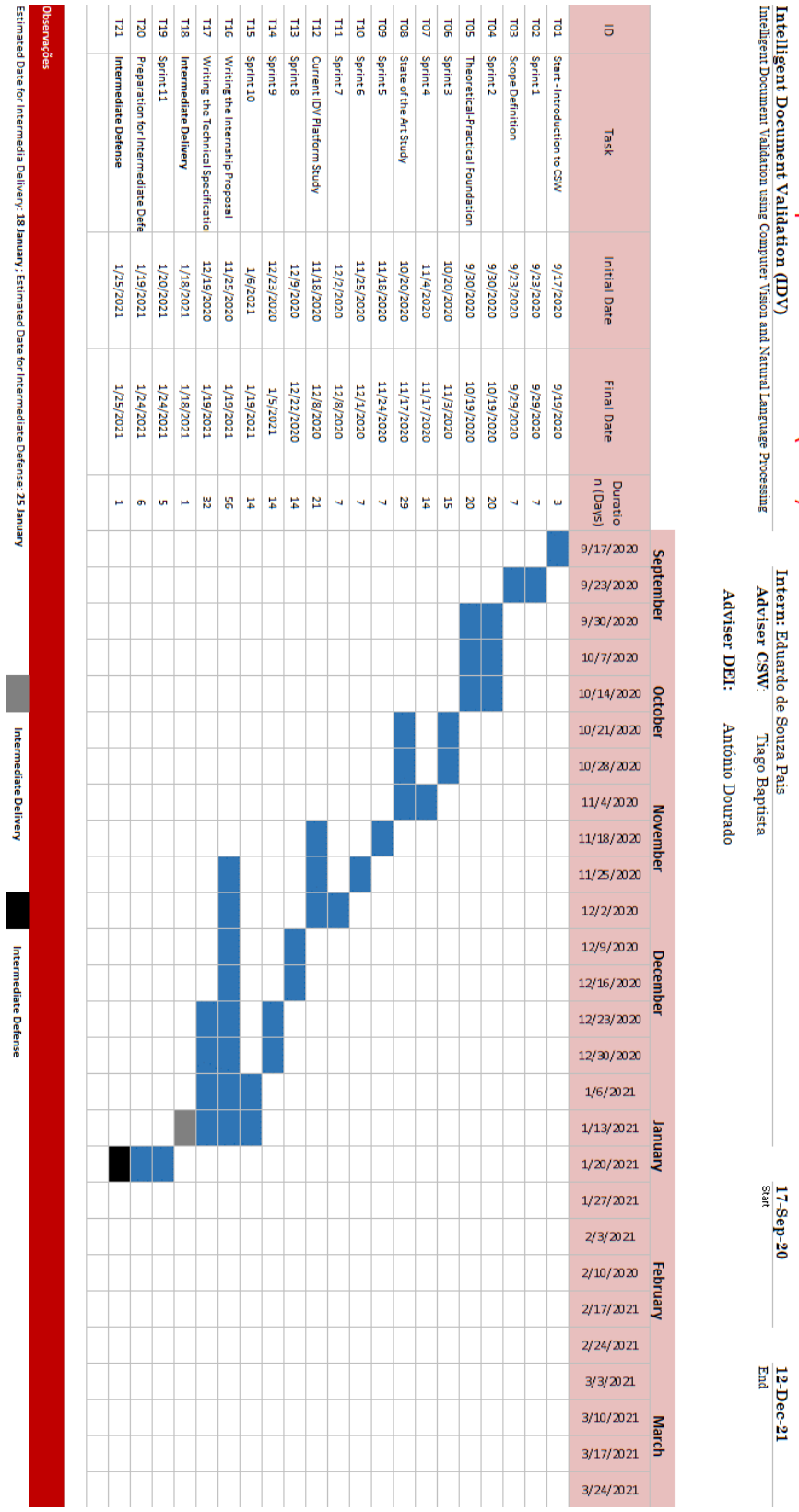

Figure 1 - GANTT 1st Semester

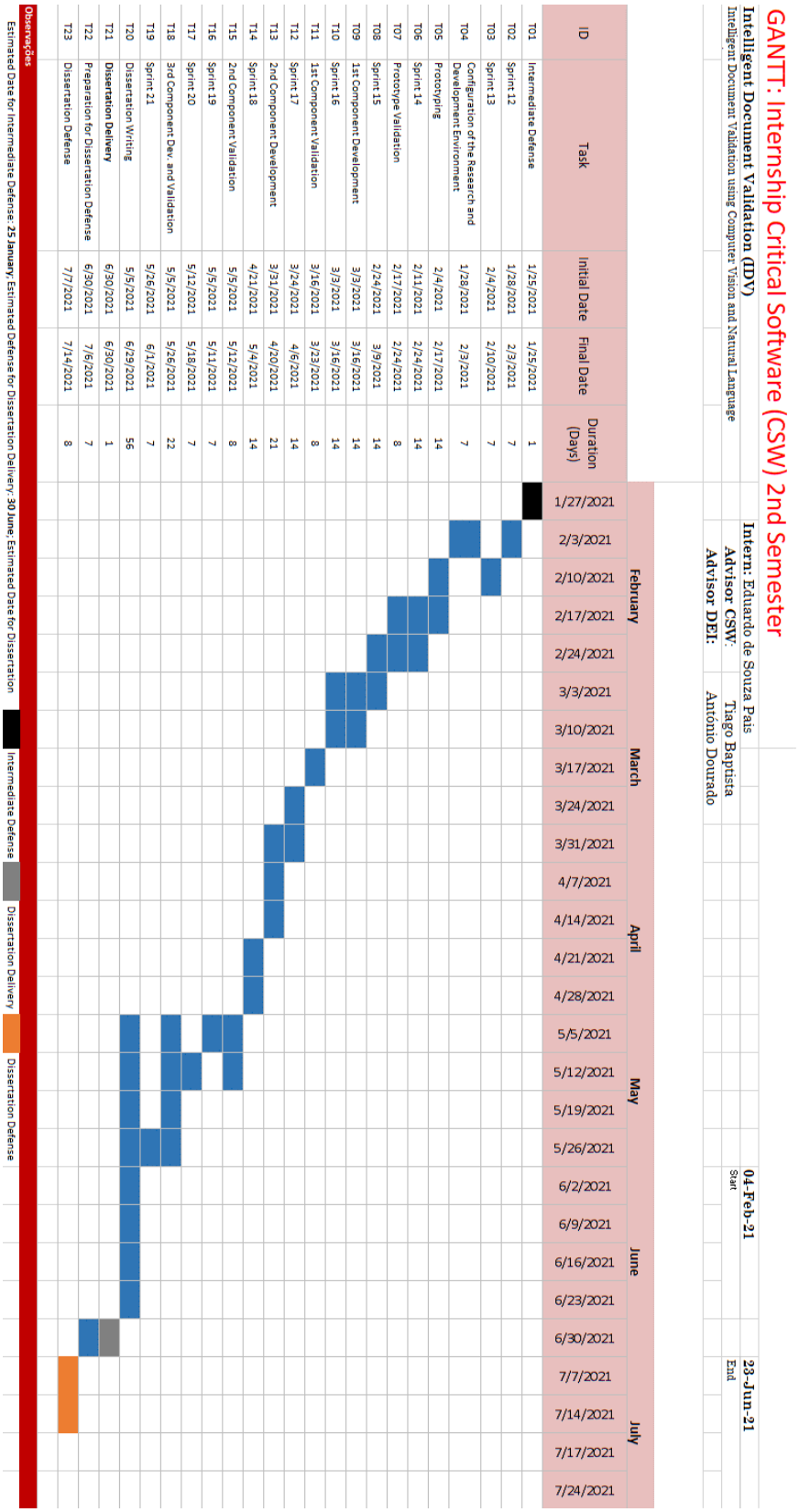

Figure 2 - GANTT 2nd Semester

1.4 Thesis Outline

The remainder of the dissertation is structured as follows: Chapter 2, Theoretical Foundation and State of the Art, presents fundamental theoretical concepts and related work regarding the main supporting themes of this project; Chapter 3, Technical Specification, presents an overview of CSW IDV platform followed by the analysis of the risks and high-level requirements; Chapter 4 – Experimentation and Results detailing the development of the overall project as well as results and respective analysis and lastly Chapter 5 – Conclusion and Future Work which concludes with some last remarks regarding the project as well as possible contributions that can be done in the immediate future regarding the project.

Chapter 2 Theoretical Foundation and State of the Art

In this chapter it is presented the theoretical foundation regarding document classification and the state of the art. Starting with the theoretical foundation sections 2.1, 2.2, 2.3, 2.4, these provide an overview of the concepts and fields of study surrounding document classification and 2.5 consists of the study of related work. More specifically, in section 2.1 it is presented an overview of ML, including its most common concepts and its implementation in this project's context, document classification. In section 2.2 is presented an overview of deep learning methods. In section 2.3 an overview of transformer models. In section 2.4 an overview of natural language processing methods, including optical character recognition. Lastly, in section 2.5 it is presented related works ranging from text-based document classification, image-based document classification and existing hybrid approaches.

2.1 Machine Learning

Machine learning (ML) is a field of Artificial Intelligence (AI) that allows a computer program to learn and adapt to new data without human intervention (Mitchell, 1997). ML algorithms are generally composed of three components: a representation method for knowledge such as decision trees, sets of rules, instances, graphical models, neural networks, support vector machines (SVM); an evaluation method in which to evaluate and compare candidate programs (hypotheses), examples include accuracy, prediction, recall, squared error and others and an optimization method, for example convex optimization and constrained optimization.

There are many different types of ML algorithms, the most common ones are: supervised learning, unsupervised learning, semi-supervised learning and

reinforcement learning. In supervised learning (also known as inductive learning), the training data includes the desired outputs which the main task of the learning function is to map an input to the output based on the examples given with input-output pairs (Russell & Norvig, 2020 (4th Ed.)), inferring a function from labelled training data consisting of a set of trained examples. In unsupervised learning the data does not include the desired outputs and instead relies on detecting patterns in the dataset, typically with no human supervision, and models probability densities over inputs (Hinton & Sejnowski, 1999). Semi-supervised learning is a ML approach that combines where the data is only partially labelled, often being used when the acquisition of fully labelled proves to be unfeasible due to its cost (Ratner, et al., 2017). The final type is reinforcement learning, usually employed in software agent models, in which the objective is to teach agents to take actions in an environment in order to increase a defined reward focusing on finding a balance between exploration and exploitation (Kaelbling, et al., May 1, 1996).

Several machine learning methods have been used for document classification and related tasks such as text classification (Lim, 2019). Examples of ML methods are Naïve Bayes, decision trees, support vector machines, k-Nearest Neighbours, Hidden Markov models, Maximum entropy, Rocchio's algorithm and Deep Learning (DL) methods which are covered individually in section 2.2. Most non deep learning methods used for document classification follow a two-step procedure of hand-crafted feature extraction from documents followed by a prediction step where the extracted features are fed into a classifier (Minaee, et al., 2020).

This two-step approach has limitations such as the reliance on hand-crafted features which requires heavy feature engineering and analysis. Also, dependence on domain knowledge for feature designing makes the created models more difficult to generalize to new data, in this project's context, to new document types. Another significant limitation is the fact that these models cannot benefit from the large amounts of training data available because of the features or its templates being pre-

defined. This has led to the rise of ML methods based on artificial neural networks (ANN) and also DL methods to address some of these limitations.

2.2 Deep Learning

This following section covers relevant deep learning methods and techniques used throughout the project as well as some theoretical basis regarding the relation between deep learning and document classification task. Section 2.2.1 introduces basic concepts of Neural Networks; section 2.2.2 covers convolutional neural networks and details its relevancy to the area of Computer Vision as well as provides some relevant CNN concepts discussed throughout the project; finally, section 2.2.3 covers Recurrent Neural Networks and its ability to learn long term dependencies and its relevancy to the project.

2.2.1 Neural Networks

One of the main ML methods are ANN whose main utility has been to model complex patterns and in solving prediction and classification problems.

One of the first successful models was the perceptron model by Frank R., it had a significant impact as it is essentially a precursor to neural networks and it is still used and taught today (Lopez, 2020). It presented a device inspired in biological principles that had the ability to learn. A generalized version of the model is exemplified in Figure 3 (left), it is composed by inputs, a bias, weights, a combination function, an activation function and an output. Learning relies on correctly varying the weight values of the active inputs, when the model misclassifies a given examples, in order to change the learning function parameters. This model, although only capable of classifying linearly separable inputs, was able at the time to tackle real world problems such as recognizing printed letters. The model is represented mathematically in Equation (1):

$$y = f(x) = g\left(\sum_{n=1}^{n} x_i w_i + b\right) \qquad (1)$$

, where $\boldsymbol{y}$ represents the output result, $\boldsymbol{x} = (x_1, \ldots, x_n)$ refers to the vector of input features, $\mathbf{w}$ the vector of weights, $\boldsymbol{b}$ is the bias term and $g(.)$ is the activation function.

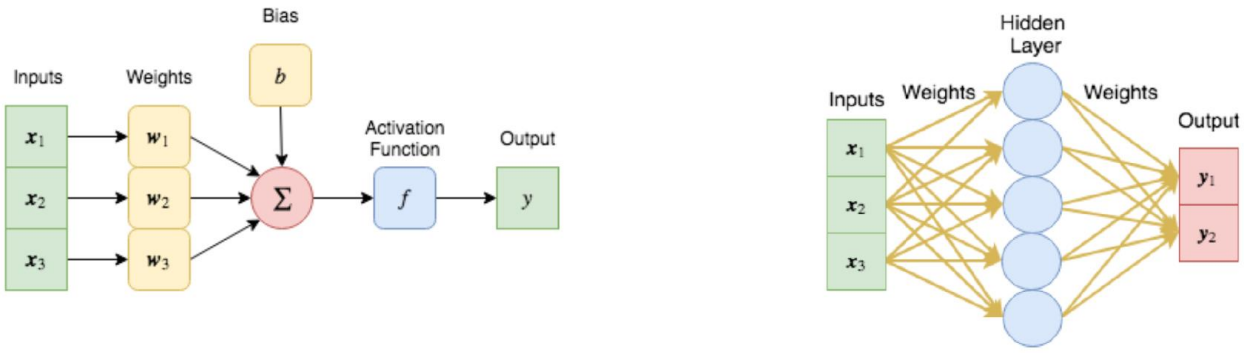

Figure 3 - The perceptron model (left) and the multi-layer perceptron model (right)

The training process of the perceptron consists of given a training set of inputs x and the respective outputs y, the model adapts the weights, w, and bias, b, to their optimal values through a learning function, resulting in an optimal equation $f^*(x)$.

To address the limitation of only solving linear models, an extension to the perceptron model was made to include layers of interconnected neurons resulting in multi-layer perceptron's. In this multi-layer architecture, the learning mechanism relies on the weights of the perceptron's hidden layers where every node can influence the output. In order to optimize the learning procedure, methods such as backpropagation and gradient descent are used. Multi-layer perceptrons are more commonly referred to as Feed-forward networks (detailed further in the following section 2.2.2.).

Multi-layer perceptrons, or feed-forward neural networks (FFNN), are composed of multiple nodes and allow the representation of more complex and non-linear models. Mathematically, a feed-forward network, with just one hidden layer (2 layers in total), can be represented by the following Equation (2):

$$y = f(x) = g(g'(\mathbf{X}' \cdot \mathbf{A} + \mathbf{a}) \cdot \mathbf{B} + \mathbf{b}) \quad (2)$$

A and **B** represent the weights of the first and second layer, respectively; functions $g(.)$ and $g'(.)$ represent an element-wise, the result of the activation functions associated to specific nodes in the layers of network (hidden and output layers).

Neural network training in this case is supervised and the goal consists of driving *f(x)* to correspond to $f^*(x)$. The training data is organized in a way, that for every example *x*, exists a corresponding label $y \approx f^*(x)$. This value *y* is the value that output needs to correctly produce for each example x value in the input layer. The model through adequate methods must obtain the desired value or an approximation of it. The most standard methods consist of iteratively apply optimization via gradient-based methods. These methods are linked to a cost function. A cost function is a measure of similarity between the true (desired) distribution and the predicted distribution. Examples of cost functions are the mean squared error for regression problems and cross-entropy for classification. The goal of the optimization method, like gradient descent, is to decrease the value produced by the cost function, minimizing it by updating the parameter in the opposite way of the gradient, making the overall model more precise.

There are many types of optimization methods, generally referred to as "backpropagation", and many sub-types of them as well, such as gradient descent having many variants. These variants differ in training data required to produce a gradient and time necessary to produce precise results among others. One of the most common optimization methods is the Adaptive Moment Estimation, or simply Adam, which is used recurrently throughout this project's model's training. This method computes the adaptive learning rates for each parameter. Adam, stores an exponentially decaying average of past squared gradients v_t. This method also keeps an exponentially decaying average of past gradients m_t, like momentum, represented in Equation (3) (Ruder, 2017).

$$\mathrm{m}_t = \beta_1 \mathrm{m}_{t-1} + (1 - \beta_1) - \mathrm{g}_t, \qquad (3)$$
$$\mathrm{v}_t = \mathrm{B}_2 \mathrm{v}_{t-1} + (1 - \beta_2)\mathrm{g}_t^2$$

In Equation (3), m_t and v_t are estimates of the first and second moment of the gradients, respectively while g_t is the gradient of the objective function at time step t. Equation (4) represents the Adam rule.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t}+\epsilon} m_t \qquad (4)$$

The algorithm used to apply gradient descent is typically backpropagation, fully known as backpropagation of sensitivities, or of gradients, which calculates the gradient of the error function with respect to the neural network's weights. The calculation of the gradients proceeds backwards through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last. Computations of the gradient from one layer are partially reused in the computation of the gradient of the previous layer. This approach allows for efficient computation of gradient at each layer (McGonagle, et al., 2020).

2.2.2 Convolutional Neural Networks and its relation to Computer Vision

One of the main fields in AI currently being researched is Computer Vision (CV). This field is one of many that aims to bridge the gap between the capabilities of humans and machines by enabling machines to view the world as closely as a human would perceive it. CV allows for a great number of tasks to be performed such as image and video recognition, media recreation, recommendation systems and natural language processing. Computer Vision and Deep learning have been linked in recent times with CV solutions being developed based on DL methods, one such method being convolutional neural networks.

A convolutional neural network (CNN), also known as ConvNet, is a Deep Learning algorithm (Saha, 2018), being a specialization of neural networks, that processes data that has a grid-like topology, e.g., images or sequential time-series data. Given an input in the form of an image or equivalent, the CNN assigns

relevance, such as weights and biases, to various objects in the input data in order to differentiate between them.

A CNN is preferred over a traditional neural network such as FFNN for image classification. A FFNN might be able to correctly classify simple binary images but it doesn't have the means necessary to handle complex images with pixel dependencies throughout. This is because most traditional neural networks use a dense interaction, where every output unit interacts with every input unit through the multiplication by a matrix of parameters in every layer. CNN use a kernel, also known as filter, with smaller size than the input and because of it has sparser interactions which results in the ability to detect smaller but more meaningful features. Through the application of relevant filters, a CNN is able to capture spatial and temporal dependencies in an image. These filters allow for the model to fit to an image dataset due to reduced number of parameters and reusable weights.

The typical input example for a CNN is an image with RGB (e.g., Figure 4) model and the goal with a CNN is to reduce an image into a form that is more easily processable but at the same time capturing or keeping features that are relevant for prediction as well as to provide scalability to the learning model.

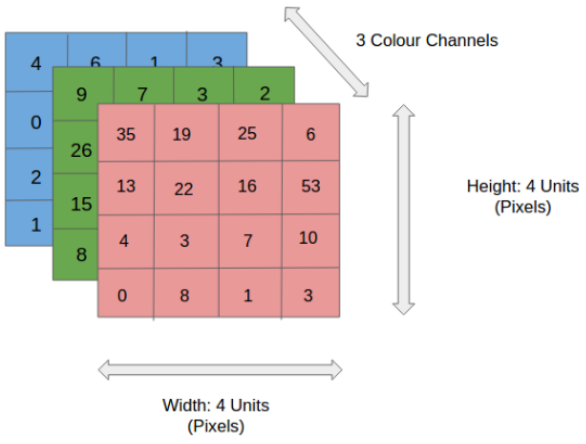

Figure 4 - Input Image of CNN (Saha, 2018)

Given an input image, multiple convolutions are performed on it through the use of different filters for each one (kernels) typically on layers called a convolutional layer resulting in a feature map. The final output of a convolutional layer is a stack

of all those feature maps. With these feature maps the network is able to describe complex relations between variable by building them upon simpler building blocks describing sparce interactions.

Mathematically, a convolution can be represented by the following Equation (9), where images are discrete, which represents an operation of two functions of a real-valued argument. In Equation (9) x is the input and w the kernel and by changing kernel value or matrix it is possible to obtain different kinds of convoluted data, or feature maps.

$$\mathrm{s(t)} = (x * w)(t) = \int_{-\infty}^{\infty} x(\tau)w(t-\tau)d\tau \qquad (9)$$

The convolutional process in a Convolutional Layer is exemplified in Figure 5, with a 5x5x1 image (e.g., greyscale) being convoluted using a 3x3x1 kernel to get a 3x3x1 convolved feature (or feature map). The way the kernel traverses through the image is dependent on a previously defined stride and padding. The kernel shifts 9 times because the stride's length is 1 (non-stride), also exemplified in Figure 5 (right). Generally, in the convolutional layer it is possible to specify the size and the numbers of kernels to be applied, the stride as in how much the convolutional filter at each step is traverses with bigger strides meaning less overlap. In images with multiple channels (e.g., RGB) the kernel has the identical depth to the input image. The general purpose of a convolutional operation is to extract high level features such as edges from the input image and there can be multiple convolutional layers in a CNN. The first convolutional layer typically extracts edges, gradient orientation colours and others while additional layers contribute to the extraction of more high-level features thus providing the model with the ability to understand images in a dataset.

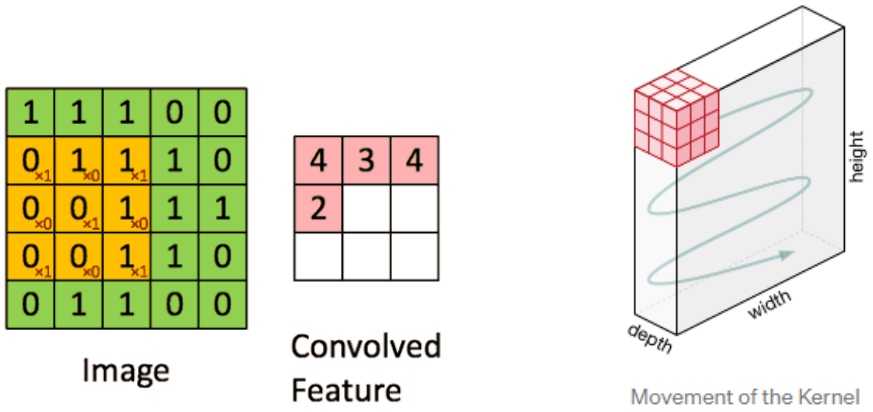

Figure 5 - Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature (left); three-dimensional representation of the movement of the kernel with stride = 1 (right) (Saha, 2018)

Another important aspect of CNN is padding. Padding is used in order to gain control over the resulting size of the feature map throughout the network, e.g., to preserve the size of the features maps as to not have them shrink at each layer. If goal is to have a resulting feature map with a smaller dimensionality from the input than valid padding needs to be applied; if the goal is to retain the original dimensionality than same padding is applied. Figure 6 illustrates the different types of padding.

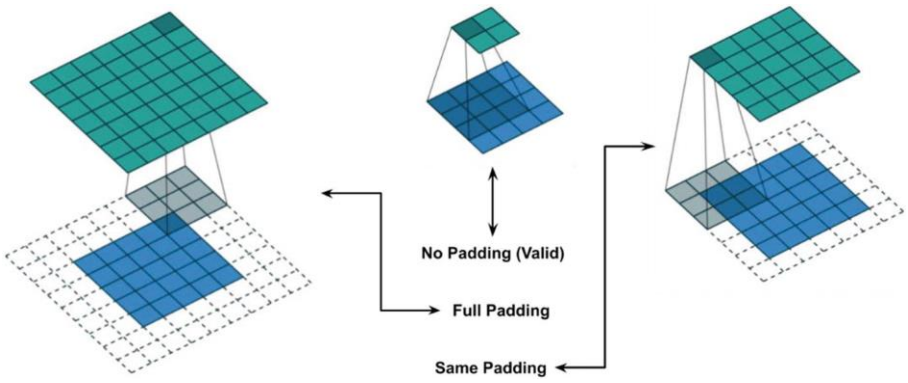

Figure 6 - Different types of padding (El-Amir & Hamdy, 2019)

After a convolutional layer there is usually a pooling layer. This layer's goal is to reduce the dimensionality, reducing the number of parameters, by down sampling

each feature map individually, reducing the width and height but keeping the depth (number of channels) intact. This reduction is useful because it decreases the computational power required to process the data and at the same time extracting the more discriminative features. There are two main types of pooling: max pooling and average pooling. Max pooling returns the maximum value from the portion of the image covered by the kernel while average pooling returns the average of all those values, exemplified in Figure 7. Max pooling also discards unnecessary activations and performs de-noising as well dimensionality reduction. Average pooling only performs dimensionality reduction and usually max pooling performs betters than it.

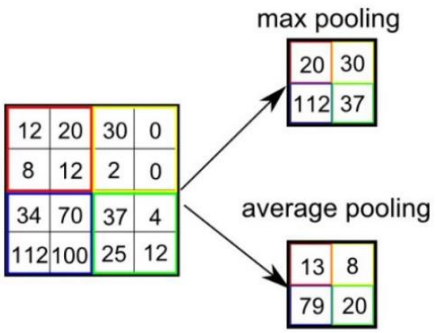

Figure 7 - Types of pooling (Saha, 2018)

With convolutional layers and pooling layers, the CNN is able to understand image features. They are followed by a flatten layer which flattens the final output into a column vector in order for it to be compatible as an input for a neural network. After the conversion the output is fed into a regular neural network to perform classification or prediction. Finally, the output of the flatten layer can be introduced to a fully connected (FC) layer, associated with an activation function such as the rectified linear unit (ReLU) (Nair & Hinton, 2010) to learn non-linear combinations of the high-level features extracted with the previous convolutional layer. Backpropagation can then be applied to every iteration of training over a series of epochs, meanwhile, the algorithm is distinguishing between discriminative and lower-level features and classifying them using an activation function (e.g., softmax) in a final layer. Figure 8 exemplifies this architecture. The last soft-max layer would then be followed by a classification layer.

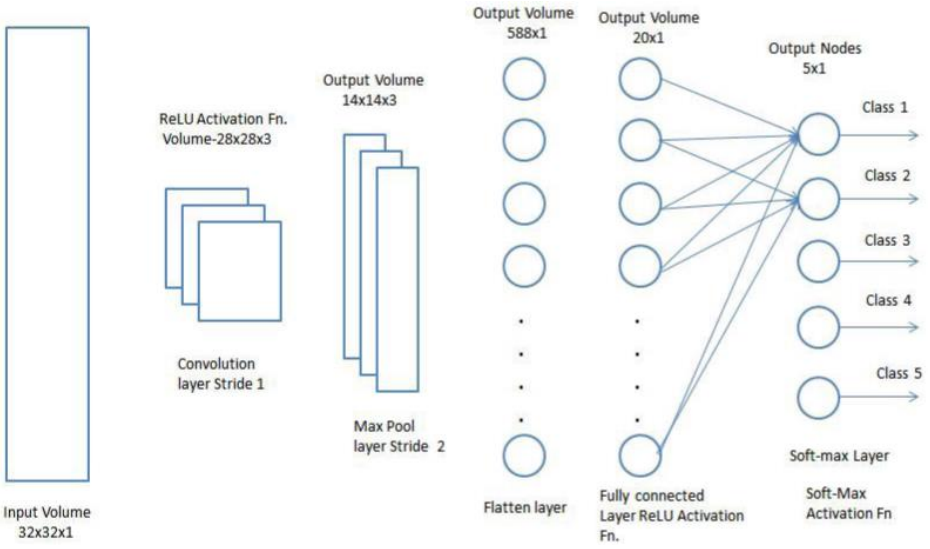

Figure 8 - Example of a full CNN (Saha, 2018)

The first major CNN architecture wasLeNet5 (Lecun, et al., 1998). In LetNet5, pooling is performed with 2x2 windows, a stride of 2 and no padding while convolutions are done with 3x3 windows, stride of 1 and with padding. This first major CNN was designed for classification of digits and letters. Other common architectures are: AlexNetl, VGGNet, GoogLeNet, ResNet and ZFNet (Saha, 2018). CNN have also been applied to natural language processing (Goldberg & Hirst, 2017) by resorting to 1D convolution operations and using vectorial representation of words through the use of word embeddings techniques (Li & Yang, 2017).

Recently, CNNs have been employed for image classification, mostly resorting to convolutional neural networks. Simonyan and Zisserman (2014) presented the Visual Geometry Group network (VGGNet) architecture for the Image Net Large Scale Visual Recognition Challenge (Simonyan & Zisserman, 2014) in 2014. This architecture achieved first place on the image localization task and a second place on the image classification task. Generally speaking, deep learning architectures for modelling 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, and thus making the training more difficult (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. 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.

2.2.3 Recurrent Neural Network – Long Short Term Memory

Recurrent neural network (RNN) is a type of neural networks that allow previous outputs to be used as inputs while having hidden states (Amidi & Amidi, 2020) making it possible to learn long-term dependencies. RNNs are commonly used for ordinal or temporal problems such as language translation, natural language processing (NLP), speech recognition and image captioning. In similarity to other neural network architectures, such as FFNN and CNN, RNN uses training data in order to learn. What makes them distinct, is the usage of "memory" as they take information from prior inputs to influence the current input and output. Elements within a sequence, in a RNN are dependent of previous elements.

RNNs take as input an ordered list of input vectors $x_1, \dots, x_n$, an initial state s_0 and returns an ordered list of state vectors $s_1, \dots, s_n$ and the list of outputs $y_1, \dots, y_n$. The resulting vectors, state vector s_i and output vector y_i, both represent the state in which the RNN is in after observing inputs $x_{1:i}$. The RNN provides a framework for conditioning on the history $x_{1:i}$ for use in modelling sequences.

RNNs shares parameters across each layer of the network, unlike traditional neural networks. RNN share the same weight parameters within each layer of the network and these are adjusted with resource to processes such as backpropagation and gradient descent. Two main parameters are R and O. R is a recursively defined function that, given as input a state vector s_i and an input vector x_{1+1}, results in a new state vector $s_{i:1}$. Parameter O is a function that maps a state vector s_i to an

output vector y_1. The shared parameters are denoted by the symbol θ. These parameters are shared across time-steps and s_n encodes the entire input sequence. The goal of a RNN is to set the parameters R and O such that the state encodes relevant information within the context of the current problem.

RNNs when unrolled can be seen as deep neural network (Figure 9) with parameter sharing. The training of the network can be done by using a variant of the backpropagation algorithm referred to as backpropagation through time (BPTT). Given an input sequence, it is created the unrolled computation graph, then appended a loss node and then BPTT is applied to compute the gradients concerning the loss.

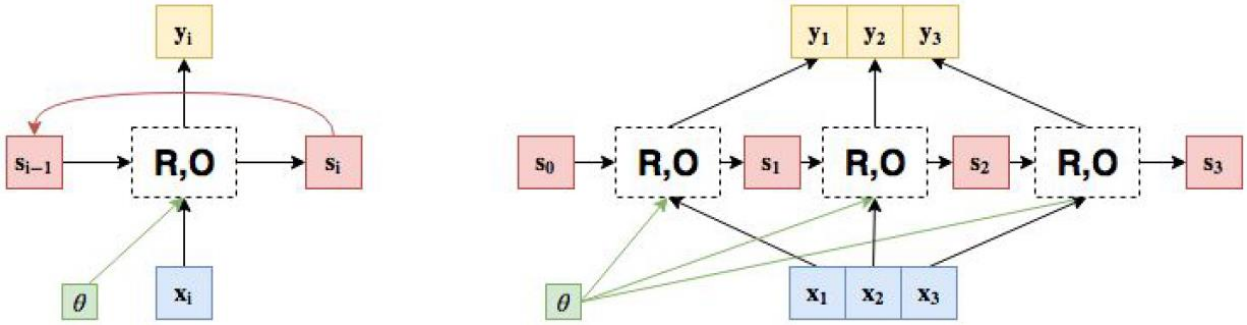

Figure 9 - Graphical representations of an RNN, recursive (left) and unrolled (right) (Nunes, 2019)

Some advantages of using RNNs are: possibility of processing input of any length; model size not increasing with the size of input; computation takes into account historical information; weights are shared across time. Some of its drawbacks are: computation may be slow; difficulty of accessing long past information; does not consider future input for the current state (but can be bidirectional, where in this case it considers, during training, the future in the current state).

There are several RNN architectures. The architecture of a traditional RNN is represented in Figure 10 (for each time step t where a^t represents the activation and y^t is the output). Equations (10) and (11) show the activation and output functions, respectively, for each time-step t where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are shared parameters and g_1, g_2 activation functions (visualized in Figure 11).

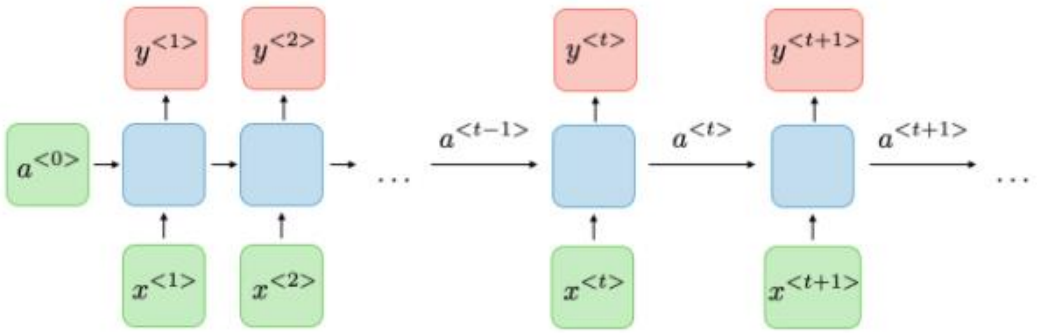

Figure 10 - RNN Architecture (Amidi & Amidi, 2020)

$$a^t = g_1(W_{aa}a^{t-1} + W_{ax}a^t + b_a) \quad (10)$$

$$y^t = g_2(W_{ya}a^t + b_y) \quad (11)$$

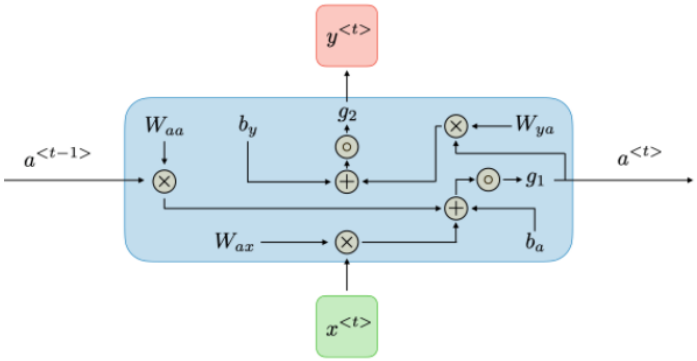

Figure 11 - Single time step RNN (Amidi & Amidi, 2020)

Traditional RNNs typically suffer from the vanishing and exploding gradient phenomena. These problems happen because of the difficulty to capture long term dependencies of multiplicative gradient that can be exponentially decreasing/increasing with respect to the number of layers. There are some solutions to avoid or cope with these problems such as Gradient clipping and "gating". Gradient clipping is a technique used to cope with the exploding gradient problem encountered when performing backpropagation. This technique caps the maximum value of the gradient in the attempt of controlling it. "Gating" refers to the creation of specific gates with the purpose of remedying the vanishing gradient problem and are implemented in some RNN variants' architectures such as Long Short-Term Memory (LSTM) and Gated Recurring Unit (GRU).

LSTM is a state-of-the-art RNN architecture for several sequence modelling tasks (Hochreiter & Schmidhuber, 1997). In order to address the issue of long-term dependencies LSTMs have cells in the hidden layers of the neural network, which have four gates: an input gate, an output gate, a forget gate and an update gate. These gates control the flow of information which is needed to predict the output in the network. LSTM units are formally defined in Equation (12) ($\odot$ denotes component-wise product or Hadamard operations). A state is composed of the memory component c_t and the output h_t.

$$\begin{aligned} cs_t &= [\boldsymbol{c}_t; \boldsymbol{h}_t] \\ \boldsymbol{c}_t &= \boldsymbol{c}_{t-1} \odot \boldsymbol{f} + \boldsymbol{g} \odot \boldsymbol{i} \\ \boldsymbol{h}_t &= \tanh(\boldsymbol{c}_t) \odot \boldsymbol{o} \\ \boldsymbol{i} &= \sigma\left(\boldsymbol{x}_t \cdot \boldsymbol{W}^{xi} + \boldsymbol{h}_{t-1} \cdot \boldsymbol{W}^{hi}\right) \\ \boldsymbol{f} &= \sigma\left(\boldsymbol{x}_t \cdot \boldsymbol{W}^{xf} + \boldsymbol{h}_{t-1} \cdot \boldsymbol{W}^{hf}\right) \\ \boldsymbol{o} &= \sigma\left(\boldsymbol{x}_t \cdot \boldsymbol{W}^{xo} + \boldsymbol{h}_{t-1} \cdot \boldsymbol{W}^{ho}\right) \\ \boldsymbol{g} &= \tanh\left(\boldsymbol{x}_t \cdot \boldsymbol{W}^{xg} + \boldsymbol{h}_{t-1} \cdot \boldsymbol{W}^{hg}\right) \\ \boldsymbol{y}_t &= \boldsymbol{h}_t \end{aligned} \tag{12}$$

Symbols i, f and o represent the 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 $\mathbf{X}_t$ and h_{t-1}, passed through a 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.

2.3 Attention-based and Transformer Models

Sequence-to-sequence deep learning models have risen in popularity in the last five years, being used in many NLP tasks such as machine translation and text summarization. Such models take as input a sequence of items (such as words or letters) and outputs another sequence of items.

This type of models consists of an encoder and a decoder (Sutskever, et al., 2014). The encoder processes each item in the input sequence, going through all of them compiling the captured information of each into a vector (known as the context). This context is usually, in NLP tasks, treated as an array of numbers (a vector). The model proceeds to send this context over to the decoder which produces a respective output sequence. The encoder and the decoder are typically RNNs with the size of the context vector being the number of hidden units in the network.

As seen in the previous section, RNNs typically take two inputs at each time step: an input (e.g.: a word from a sentence) and a hidden state. To transform a particular word into a vector, an algorithm called word embeddings is used. These can capture the semantic information of words (i.e., its meaning Figure 12). Word embeddings can be used as a pre-trained embedding, or they can be created and trained using data. Since the encoder and decoder are both RNNs, each time step one of the RNNs does some processing, it updates its hidden state based on its inputs and previous inputs it has seen.

Context, in the form of vectors, made it difficult to deal with long sentences and a solution as proposed by (Luong, et al., 2015) in the form of a technique called "Attention". It allows the model to focus on relevant words as is necessary, increasing overall model performance. Instead of the RNN encoder passing one hidden state at a time to the decoder, it passes all of them and before producing an output it computes each hidden state softmax score based on their original value (amplifying hidden states with high scores) and produces a context vector by summing up the weighted vectors. This vector is concatenated (C4 - Figure 13) with the last hidden

state (H4) and passed to a FFNN (Forward Neural Networks) thus producing an output.

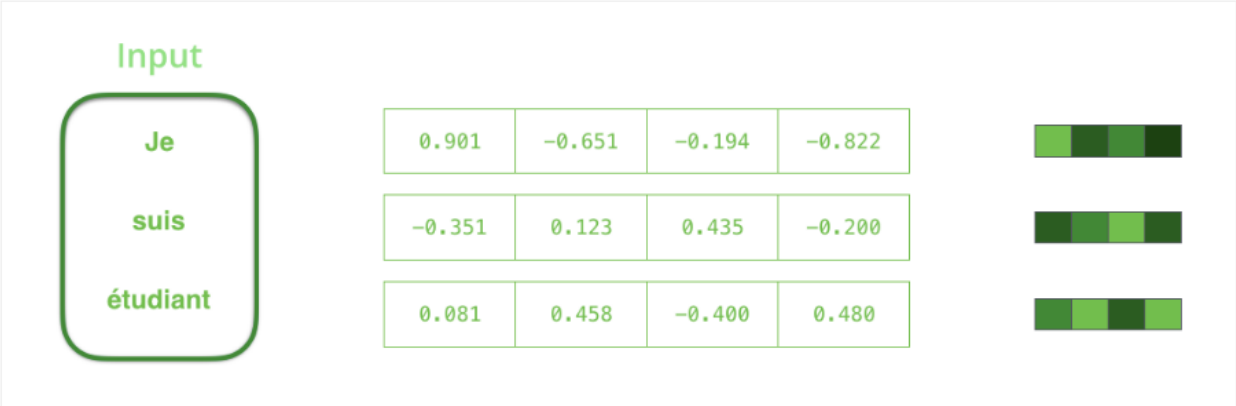

Figure 12 - Individual words conversion into vectors (Alammar, 2021)

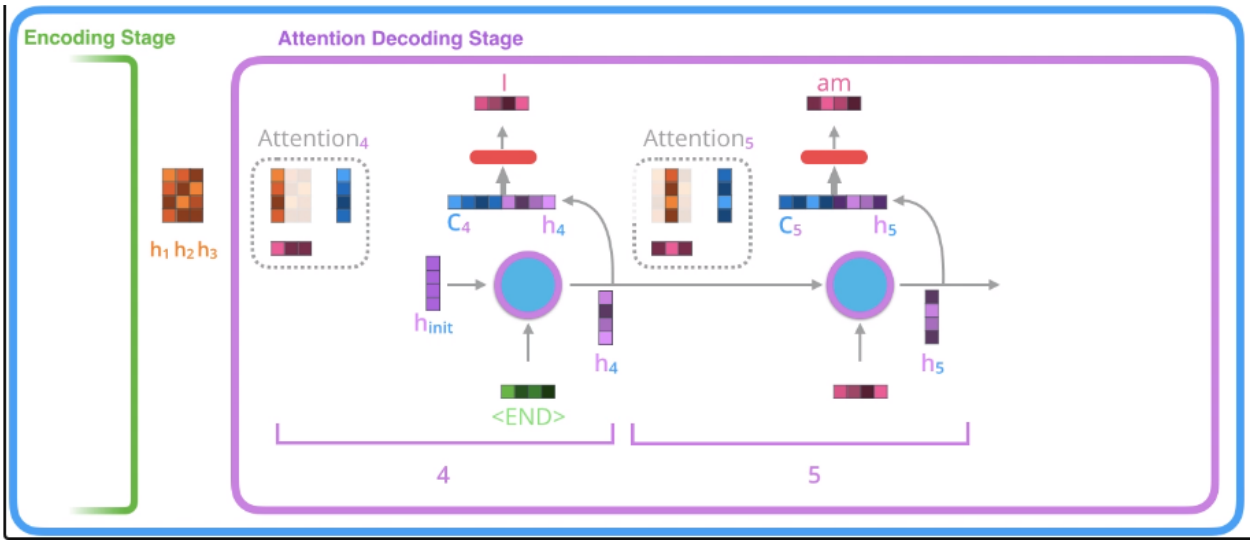

Figure 13 - Sequence to Sequence model with Attention (Alammar, 2021)

Transformer models also use attention techniques, specifically to increase their respective training speed and were first proposed in the work of Vaswani et al. (Vaswani, et al., 2017). These models follow the encoder-decoder stacks format. Each encoder is made of a Self-Attention layer and a FFNN layer. The self-attention layer allows the encoder to take into context other words as it encodes a

specific word and The FFNN layer receives its output. The decoder along with the two mentioned layers also possesses between them an Encoder-Decoder Attention layer, which similarly to Sequence-to-sequence models, helps the model focus on relevant words. Also, similarly to sequence-to-sequence models the input is converted to vectors using an embedding algorithm. The embedding only happens in the bottom-most encoder.

The embedded words flow through the two layers of the encoder and in parallel in the case of the FFNN layer. The word (x1, x2, etc.) at each position passes through a self-attention process (z1, z2, etc.) - this process allows the algorithm to contextualize a word regarding the others in a sentence. Then, they each pass through a feed-forward neural network (r1, r2, etc.) - the exact same network with each vector flowing through it separately (Figure 14).

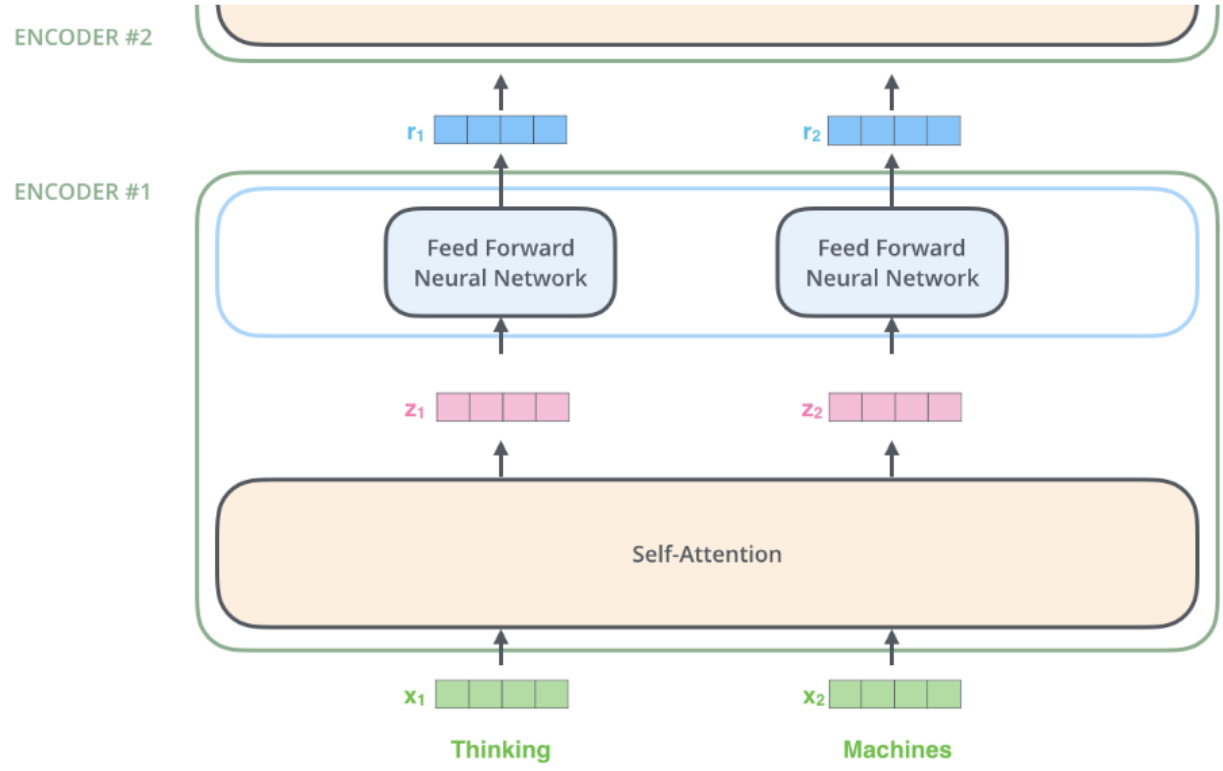

Figure 14 - Transformer encoding process (Alammar, 2021)

Transformer model also uses positional encoding to track the order of words in an input sequence. It does this by adding a vector to each input embedding which the pattern followed by these vectors are learned by the model allowing to track the position of each word (Figure 15).

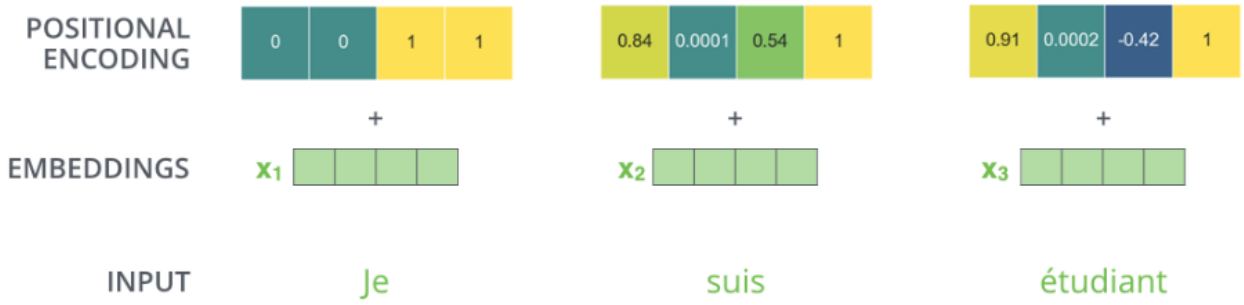

Figure 15 - Example of positional embedding with an embedding of size '4' (Alammar, 2021)

The output of the encoder is a set of attention vectors K and V which are used by each decoder in its encoder-decoder attention layer to focus on particular words in the context of others.

The final layers of a transformer models are usually a FFNN layer and a softmax layer. The Linear layer is a fully connected neural network that projects the vector produced by the stack of decoders, into a larger vector called a logits vector. Each vector value corresponds to a score of a unique word in the model's vocabulary. The softmax layer turns those scores into probabilities, adding to 1.0. The cell with the highest probability is chosen, and the word associated with it is produced as the output for this time step.

One example of a widely used transformer model is BERT. BERT is a contextualized bidirectional word embedding capable of capturing the meaning of words in a sentence in their respective context. Models such as word2vec or GloVe don't take into account context. These generate a single word embedding representation for each word in the vocabulary while BERT takes into account the context for each occurrence of a given word. BERT will provide a contextualized embedding that will be different according to the sentence. This transformer model

was used in this project to both individually train a textual-based model as well as to serve as input to a hybrid model.

2.4 Natural Language Processing

Natural Language Processing (or NLP) is a field of Artificial Intelligence that gives machines the ability to read, understand and derive meaning from human languages (Yse, 2019). The ultimate objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable. Through natural language, a person can generate dozens, hundreds or thousands of words in a declaration, each sentence with its corresponding complexity, ambiguity and interpretation by others. Scaling and analysing several examples of natural language results in a very complex problem.

Data generated from conversations, declarations or even informal situations such as in social media (e.g.: tweets) are examples of unstructured data. Unstructured data represents the vast majority of data in the real world. It doesn't fit into traditional relational databases often requiring pre-processing in order to be useful. NLP along with ML have paved the way for a new way to understand text or speech, it being no longer dependent on keywords (previous, mechanical way) instead being about understanding the text's meaning (cognitive way). This allows for more advanced analysis such as detecting figures of speech, irony and sentiment analysis.

Some of NLP applications are: disease prediction and diagnostics based on electronic records and patient's speech; sentiment analysis for marketing purposes or recommendation systems; language translation; word processors that employ NLP to check for grammatical accuracy of texts; interactive voice responses (e.g.: call centres); personal assistant applications (e.g.: Siri, Cortana).

Most basic tasks in NLP can be divided between syntactic and semantic tasks. Syntax refers to the arrangement of words in a sentence such that they make

grammatical sense. In NLP, syntactic analysis is used to assess how the natural language aligns with the grammatical rules. Semantics refers to the meaning that is conveyed by a text. It is one of the main challenges in NLP and it involves applying computer algorithms to understand the meaning and interpretation of words and how sentences are structured.

Some examples of syntactic NLP tasks are: tokenization, Part-of-speech tagging and lemmatization. Tokenization is a syntax approach and consists of segmenting text into tokens[1]. It is usually a required step for other base NLP tasks. Part-of-speech tagging is the process of automatically assigning a part-of-speech (PoS) to a word in context. Words with the same PoS likely have similar grammatical properties. Lemmatization reduces a word to its base form and groups together different forms of the same word. For example, verbs in past tense are changed into present (e.g., "went" is changed to "go") and synonyms are unified (e.g., "best" is changed to "good"), hence standardizing words with similar meaning to their root.

A few basic NLP tasks related to semantic analysis are: Named Entity Recognition and Word Sense disambiguation. Named entity recognition (NER) is a semantic task and it consists of determining the parts of a text that can be identified and categorized into categorical groups. These categories can be names, locations and organizations. Word sense disambiguation involves giving meaning to a word based on the context, more specially, identifying which sense of a word is used in a sentence. It does this by maps an occurrence of a word with the most suitable entry in a sense repository (e.g., dictionary or an ontology[2]).

NLP has been widely used for text classification. This includes automatic document classification based on textual data, typically through machine learning algorithms (Tolpygo, 2016). One simple way of achieving this is through a bag of

[1] Token: A token is the smallest meaningful unit of information in a natural language string. E.g.: "Modern NLP.", tokens: |Modern| |NLP| |.|

[2] Ontology: An ontology, in NLP or in a semantic web context, is a formal collection of terms used to describe a particular area of interest.

words representation. This method allows to transform raw text into features by creating a count for the number of times each word appears. This method can be complemented by other NLP related methods such as N-grams and term frequency-inverse document frequency (TF-IDF). These features can then be introduced as variables in a classification problem in order to train a model for automatic document classification. Another way NLP has been used is for data preparation, typically in the form of a list of documents, for training or feature extraction. Common NLP tasks for data preparation are n-grams[3], punctuation and stop word removal, lemmatization and stemming.

NLP also allows for a vectorized representation of words. This process is commonly achieved through word embeddings. Word embeddings allows words with similar meaning to have a similar representation, being represented by real-valued vectors in a predefined vector space. This approach to representing words and documents may be considered one of the key breakthroughs of deep learning on challenging natural language processing problems (Brownlee, 2019). Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often lumped into the field of deep learning. The distributed representation is learned based on the usage of words. This allows words that are used in similar ways to result in having similar representations, naturally capturing their meaning. This can be contrasted with the crisp but fragile representation in a bag of words model where, unless explicitly managed, different words have different representations, regardless of how they are used.

Before applying the above-mentioned NLP tasks to a textual input derived from a document, first the textual data from that document needs to be extracted. This process is referred to as Optical Character Recognition (OCR). It is a subset of pattern

[3] The text is often split into words or grams. The grams, that are often referred to as **n-grams** are a sequence of n words from the text (Tolpygo, 2016).

recognition area and computer vision which deals with the problem of recognizing optically processed characters (Eikvil, 1993). OCR electronically converts typed, handwritten or printed text images into machine-encoded text. With OCR it is possible to digitalize a varied number of paper-based documents, across multiple languages and formats, into machine-readable text improving these documents storage, accessibility, editability, translatability and searchability.

Similarly, to pattern recognition, the main principle is to train a machine to identify and classify patterns such as letters, numbers or symbols and their respective features. This teaching is performed, during training, by showing the machine examples of characters of all the different classes. Based on these examples the machine builds a prototype or a description of each class of characters. Then, during recognition, the unknown characters are compared to the previously obtained descriptions, and assigned the class that gives the best match (Eikvil, 1993).

Recent OCR technology is capable of recognizing both hand-written and printed characters but are heavily dependent of the submitted document's structural complexity and quality as well as the artifacts introduced during scanning (Abdulkader & Casey, 2009). Regarding document classification, OCRs are capable of recognizing scanned or photographed documents, images and documents in *Portable Document Format (PDF)*, in each specific pre-processing techniques may be required. Most OCR tools are able to achieve a value of recognition precision of 90-99% and so it is not a perfect process as some words or symbols are not well recognized still (Abdulkader & Casey, 2009) with human intervention still being necessary in some situations.

2.5 Document Classification – Overview and Related Work

With the rise of digitalization, understanding and analysing data, in digital form, from documents has become a common activity, mostly done by humans, in order to validate the documents, extract information from its fields, check structure and its

submissions. This manual management of documents is time consuming and expensive and has resulted in the need for automation, with many of its tasks such as document classification and document recognition to become automated in order to help with making this process faster and more efficient while also assisting and making lighter the burden of this slow and repetitive work for human workers.

To differentiate documents based only on their broad class (such as banking documents from passports) a visual-only approach may provide decent results but in order to perform fine-grained recognition, textual content of the documents is often necessary. For example, in the case of a particular type of document, such as a tax form, documents of this type may share the same structure or layout, logos, templates or forms and differ mostly by their textual content. Deep learning has in recent years been used for both image classification and natural language processing, both areas that can be applied to document classification. Even though some multimodal approaches have used both visual and textual content in this task, it still remains an open problem. Documents classification can therefore be achieved through image classification, text classification (using an OCR) and multimodal classification. The model proposed in this project aims to leverage both textual and visual information for document classification for possible integration into CSW IDV platform.

Early studies, from the field of Computer Vision, have been using deep neural networks for document analysis tasks focused primarily on a document's visual features and structure. Byun & Lee, 2000, used a dynamic programming algorithm for partially matching, in which form structure recognition and form classification are performed for only some areas of the input form. Kumar et al., 2012 used a method based on statistics of patch-codewords over different regions of image for document classification. Kumar, et al., 2014, proposed a method to study structural similarity for document classification based on spatial features. Most studies for document classification that use visual features only treat the task as a conventional image classification. However, visual features alone tend to have problems such as low inter-class discrimination, i.e., images of different classes may share the same structure,

and high intra-class structural variations of highly overlapped document images, i.e., images in the same category have similar structures and therefore visual features alone have difficulty discriminating them (Afzal, et al., 2015).

From a natural language processing perspective, Yang, et al., 2017, presented a multimodal neural network to extract semantic information based on word embeddings from pretrained natural language models. This and others multimodal approaches have achieved promising results. The need for both a document image and its textual content have given rise to other multimodal approaches, with the textual component commonly being extracted by an OCR with post semantic analysis and its visual content most commonly being treated by a convolutional neural network.

In the following sections it will be presented related work regarding the three main types of document classification: in section 2.5.1 related work about textual document classification; in section 2.5.2 about image document classification and in section 2.5.3 related work about known multimodal approaches.

2.5.1 Textual Document Classification

Document images can be characterized based on their textual content. Their textual content can be extracted via OCR techniques. Initial textual based approaches, such as Shin, et al., 2001, worked with structure-based features to classify document pages. Their work was based on visual similarity of the layout structure of documents. They used image features such as percentages of text and non-text (graphics, images, tables and rulings) content regions, column structures, sizes of fonts, etc. In order to achieve this classification, they relied on supervised classifiers in the form of decision trees and self-organizing maps.

More recently, in regards to natural language processing, the appearance of learned word embeddings approaches such as Word2Vec (Mikolov, et al., 2013), Glove, ELMO (Peters, et al., 2018), FastText, XL-Net (Yang, et al., 2019) have led to

significant improvements in document classification. The different types of static and dynamic word embeddings aim to learn lexicon related to the words or vocabulary of a language, syntactic to create well-formed sentences in a language, semantic related to meaning in language, and pragmatic approach related to proximity between words and documents.

Word embedding approaches have led to numerous works based on recurrent and attention mechanisms for text classification. One example of such works is Yang, et al., 2016, "Hierarchical Attention Networks for Document Classification", which proposes a bidirectional recurrent network with a hierarchical attention mechanism that learns both at a word and sentences level, mirroring the natural hierarchical structure of most documents (e.g., documents are composed of sentences, sentences are composed of words). The reasoning behind this approach was to increase classification performance and, simultaneously, provide interpretability by giving insight into which words and sentences contributed to the classification decision.

2.5.2 Image Document Classification

Computer vision and deep learning have been recently suggested as a first solution to classify documents based on their visual appearance. Document image analysis was one of the first areas where modern deep learning has been applied with the first CNN originally being designed for classification of digits and letters (Lecun, et al., 1998). From a computer visions perspective, it has been one of their general goals in this area to achieve image-based document classification without the need for textual content extracted by an OCR as stated in 2007 survey (Chen & Blostein, 2006). Other early attempts in document image classification focused on region-based analysis by detecting and analysing certain parts of a document (as in the work of Hao et. al., 2016).

Deep learning approaches specific to image document classification also became even more prevalent, inspired by the success in of these networks on the ImageNet[4] dataset introduced by Harley et al. (Harley, et al., 2015). This work sought to demonstrate the effectiveness of deep CNNs over handcrafted alternatives in document image classification and retrieval tasks. It also showcased many advantages of deep learning in this task such as the following: CNN's features robustness to compression; how CNNs trained on non-document images transfer well to document analysis tasks and how region-specific feature-learning is unnecessary given sufficient training data.

The way to approach an image document classification has also been a topic of discussion, with transfer learning of a model trained with regular images being common. A study on this was made, on whether general CNN architectures employed for image classification were appropriate for document image classification (Tensmeyer & Martinez, 2017). They performed a large empirical study to discover what aspects of CNNs most affect performance on document images. They exceeded previous results on the RVL-CDIP dataset by using data augmentation and an architecture designed for a larger input image. A major discovery was evidence that CNNs trained on RVL-CDIP learn region-specific layout features and also that by using transfer learning they were able to improve accuracy in relation to previous architectures on the RVL-CDIP. In this approach they used an AlexNet, Krizhevsky, et al.'s (2012) architecture pre-trained on ImageNet dataset.

2.5.3 Hybrid / Multimodal Classification

Visual features alone tend to have problems such as low inter-class discrimination and high intra-class structural variations of highly overlapped document images and therefore visual features alone have difficulty discriminating them. Because of this,

[4] The **ImageNet** project is a large visual database designed for use in visual object recognition software research.

recently there has been a rise of multimodal approaches in order to achieve fine-grained document classification. These hybrid approaches combine textual and visual features in a multimodal network to perform document image classification.

An approch designed to be a baseline for others is that of Audebert, et al., 2019. They propose a baseline approach with a hybrid deep model. In order to classify post-OCR document images, the authors present a pragmatic pipeline to perform visual and textual feature extraction using typical architectures. To leverage the complementary information present in both modalities, they designed an end-to-end network that jointly learn from text and image while trying to keep computation cost at its minimum. They build on existing deep models, MobileNet and FastText, and demonstrate significant improvements using their fusion strategy on two document images dataset.

Regarding visual features, Audebert, et al., fine-tuned a CNN pretrained on ImageNet in order to extract visual features on their images as it had been suggested in previous related works (e.g.: Tensmeyer & Martinez, 2017). In order to reduce time and cost constraints they decided to use a lightweight architecture with competitive classification performance, the MobileNetV2 model (Sandler, et al., 2018). This architecture consists of a stack of bottleneck blocks. each bottleneck block transforms a feature map first by expanding it by increasing its number of channels with a 1∗1 convolutional layer with identity activation. Then, a 3∗3 depth wise convolution is performed, followed by a ReLU and a final 1∗1 convolution with ReLU. To improve efficiency this block inverts the traditional residual block since the expansion is performed inside the block, whereas residual blocks compress and then re-expand the information (illustrated in Figure 16). The MobileNetV2 used had 19 residual bottleneck layers.

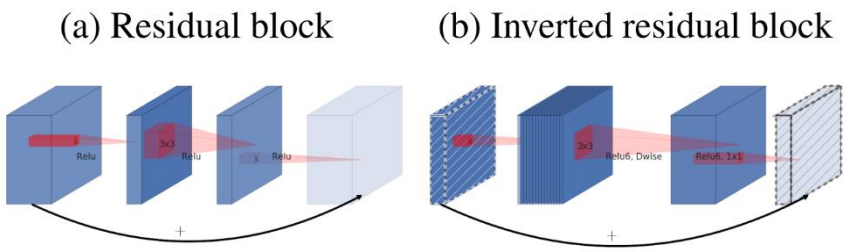

Figure 16 - MobileNetV2 uses inverted residual blocks (Sandler, et al., 2018)

In this work text had not been transcribed and therefore the authors had to resort to an OCR to extract textual data from the documents. They used Tesseract OCR engine version 4 which is based on a LSTM neural network. Tesseract is able to detect text orientation and rotation and act accordingly, if necessary. It also deals with image binarization to identify black text on a white background. However, it does have limitations such as its output being noisy if proper conditions are not met such as an unflatten page, curved text or exotic fonts but a crucial issue is that they do not deal with out-of-vocabulary words (OOV). The author's solution was to use character-based word embeddings that are able to deal with OOV words by assigning them plausible word vectors that preserve both a semantic and a spelling similarity. They went with FastText since no OCR extracted OOV were absent from this embedding. They then convert the word embeddings into a document embedding using the open-software SpaCy[5] to also perform tokenization and punctuation removal. Individual word embeddings are then inferred using FastText pretrained on the Common Crawl dataset.

After textual and visual features extraction they proceed to feed these to a multi-layer perceptron, combining both feature vectors into one. They tested two fusion methods: one involving an adaptive averaging of both feature vectors and the other the concatenation of both vectors. The latter proved to perform best when compared

[5] SpaCy: open-source software library for advanced natural language processing, written in the programming languages Python and Cython (Honnibal, 2015).

to the pure image model. The authors conclude that aligning text and image feature spaces resulted at a loss of discriminative power.

The datasets used by the authors are the Tobacco3482[6] dataset with 3482 black and white documents with annotations for 10 classes of documents (e.g.: Email, Form, Letter, Memo, etc.). They perform training with 800 documents and the rest for testing. They also used the RVL-CDIP dataset[7] with 400000 grayscale digitized documents with annotations for 16 classes documents (e.g.: Email, Letter, Invoice, etc.) each containing 25000 examples. They used the standard split with 320000 documents for training, 40000 for validation and 40000 for testing. For text generation, they considered Tesseract OCR library to extract text from greyscale images from both datasets and is publicly available[8].

The models were implemented using Tensorflow 1.12 and Keras API with manual selection of hyperparameters. Their text model consists of one-dimensional convolutional neural network. The CNN is 4-layers deep and interlaces 1D convolutions with a window of size 12 with max pooling with a stride of 2. Each layer consists in 512 channels with ReLU activation. The final feature map is processed by a max pooling-through-time layer that extracts maximal features on the sequence with Dropout regularization. A fully connected layer then maps the features to the softmax classifier. The input word sequence is zero-padded up to 500 words for documents with less 500 words. Their image model is based on the MobileNetV2, a lightweight CNN architecture that focuses on computing efficiency. The CNN is trained on grayscale document images resized at 384 by 384 with grayscale channel being duplicated three times to adjust to the model's RGB style of images. The CNN is initialized with pretrained weights on ImageNet which accelerated converge and

---

[6] **Tobacco3482:** a subset from the Truth Tobacco Industry Documents archives of legal proceedings against large American tobacco companies. https://www.kaggle.com/patrickaudriaz/tobacco3482jpg

[7] **RVL-CDIP:** a subset from Truth Tobacco Industry Documents. https://www.cs.cmu.edu/~aharley/rvl-cdip/

[8] The extracted text dataset (**QS-OCR dataset**) is available at: https://github.com/Quicksign/ocrized-text-dataset

improving accuracy through transfer learning. The final fusion model considers the previous two models but with the final layers cut-off in order to fuse them together. For the text-based model, the last layer produces an output vector of dimension 128 instead of the number of classes. For the image-based model, an aggregation of the last convolutional features using global average pooling on each channel was made, which produced a feature vector of dimension 1280. Finally, they then map this feature vector using a fully connected layer to a representation space of dimension 128. The full hybrid model's architecture is illustrated in Figure 17. This work achieved an overall accuracy result value of 90.6% on the RVL-CDIP dataset and an overall F1-score of 0.86 (0 to 1 scale) and an overall accuracy value of 87.8% in the Tobacco3482 datasets confirming the proposed idea that the two sources, text and image, give complementary information in for document classification.

Some limitations and observations on this approach can be made. The model was tested on public document image datasets. Real world derived images may not be as clean or as well orientated as the images presented on datasets such the Tobacco subsets. The scanning process was also performed by experts and real-world experience may generate poor quality scans. Data augmentation was not performed and may be necessary in order to classify documents that may have deteriorated due to natural or artificial reasons previous to their digitalization. Post-OCR word embeddings can create noise and a semantic analysis of its output may be necessary.

One of the more recent approaches is that of Bakkali, et al., 2020. They proposed a cross-modal network to perform image and text feature extraction relying on off-the shelf image-based deep networks and word embedding algorithms. The authors attempted to bridge the two modalities in an end-to-end network to simultaneously learn from image and text features. The built-in network is based on the performance of lightweight, heavyweight architectures used in their experiments for image stream, and static, dynamic word embeddings used to perform text classification.

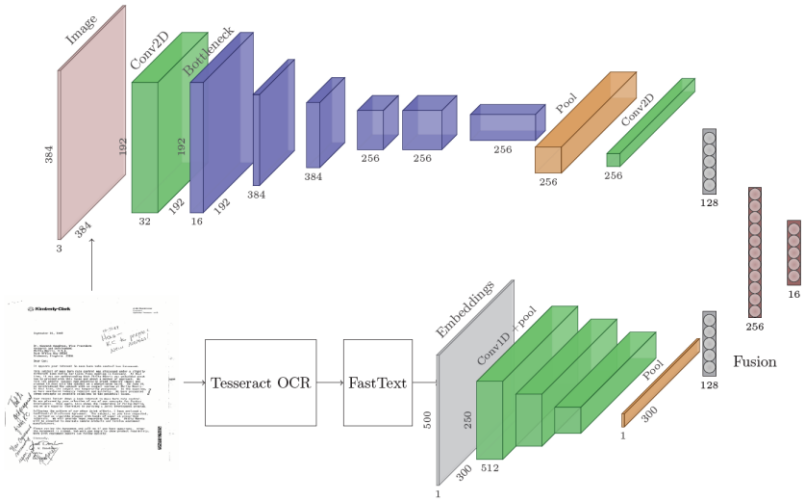

Figure 17 – Multimodal end-to-end, classifier for hybrid text/image (Audebert, et al., 2019)

In regards to image data, they explored two architectures: NasNet and Inception-ResNet. The NasNet (Zoph, et al., 2018) architecture is composed of two types of layers: Normal layer, and Reduction layer. The Normal layer is a convolutional layer that returns a feature map of the same dimension, where the Reduction layer is a convolutional layer that returns a feature map, where the feature map height and width is reduced by a factor of two. Two variations of this network were experimented on with 88.02 million (M) and 4.23 M parameters, respectively. Inception-ResNet-v2 (Szegedy, et al., 2017) is a convolutional neural network that achieved state-of-the-art results. Inception-ResNet-v2 is a variation of the earlier Inception V3 model by introducing the bypass connection as in ResNet. The model has 54.36 M parameters. All three CNNs are pre-trained on the ImageNet weights. For the text component they explore the usage of three recent word embedding approaches: Glove, Bert and FastText.

The proposed network consists of two streams: an image stream and a text stream. In the first, they feed input document images and in the second they extract textual data using an OCR of which the text strings produced are then fed as input to the word embedding algorithm. They considered a late fusion process.

The image stream extracts visual features that are passed to a global average pooling layer to reduce the spatial dimensions of a three-dimensional tensor. It performs also a more extreme type of dimensionality reduction. For the final layers of the three deep CNNs, the global average pooling layer is passed to the last fully connected layer to perform classification with a softmax layer. In regards to textual features, all documents were processed by Tesseract's OCR engine. They analysed the three different words embeddings and focused on Bert's because of the advantage of dynamic word embeddings over static ones in capturing semantic meaning. Bert also has means to deal with OOV. The input embeddings are passed to the attention based bidirectional transformer. After pre-processing the textual content extracted by the OCR engine form document images, they pass the input embeddings of both Glove and FastText to a GRU network of 32 nodes and 3 hidden layers. The final layers of the three models are passed to a softmax layer with categorical cross-entropy loss function.

They explored two different approaches to achieve a late fusion process. In the first, equal concatenation, they add a fully connected layer to the image stream, having the same dimensional output vector as the text stream. In the second, they employ a pixelwise addition between the image and text embedding features, *i.e.,* superposing directly the two embeddings to generate the cross-modal features. The resulting cross-modal features have the same dimension as the text or image text embedding features. The network's parameters are learned through the learning of the individual stream's parameters and a later optimization of joined parameters by the global cross-entropy loss function. They used the RVL-CDIP dataset as well, being a popular dataset for research on document classification. The full model architecture is illustrated in Figure 18.

They used many pre-processing methods in order to increase performance. In order to minimize intra-class similarity, they applied a shear transform process to augment the data. Random image shifting was also used for better generalization.

Cutout data augmentation[9] was also used to improve regularization. As a final pre-processing step, they also convert the grayscale images to RGB images.

Their hybrid model achieved an overall accuracy of 96.94% using the equal concatenation modality for fusion and 97.05% using the average ensemble modality for fusion. The results are higher than previous state-of-the-art results by a margin of 2.63%. They were also higher than the single modal approaches tested by the authors as well proving that a combined feature approach improves the result of using only either visual or textual features.

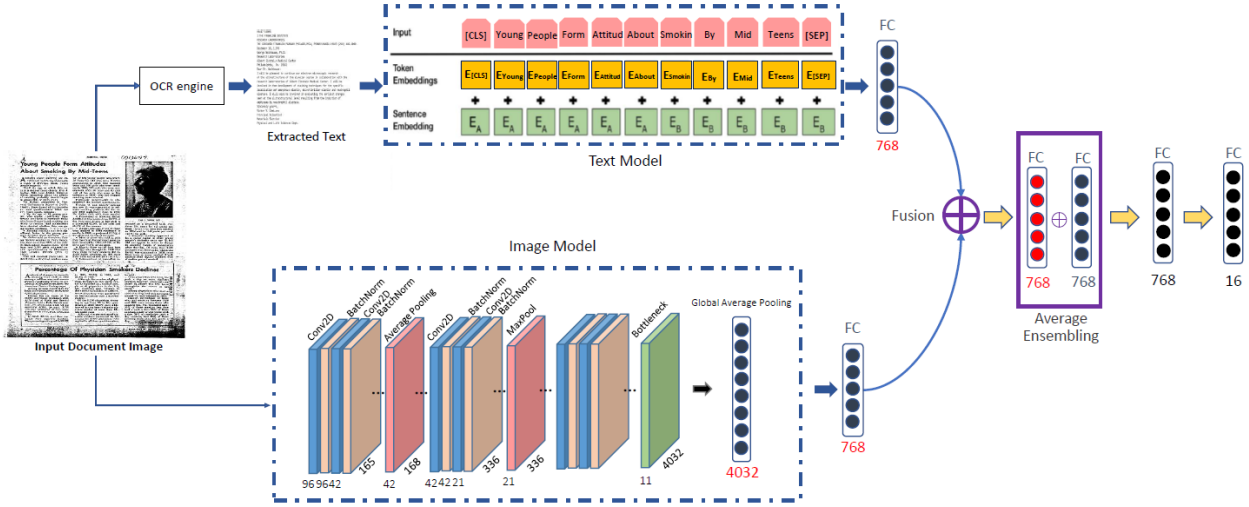

Figure 18 - Cross-modal network with a text stream and an image stream (Bakkali, et al., 2020)

Another work is that of Dauphinee, et al., 2019. This work consists of modular multimodal architecture is presented in for document classification followed with a XGBoost meta-classifier. Being also a modular archicteture, it allows for the swapping of different classifiers to be more acessible.

Similar to previously mentioned works they also use an OCR to perform text extraction, specifically Tesseract OCR. Overall there are three model components in this work: an image classifier, a text classifier and a meta-classifier that joins the two

[9] It consists of randomly masking a square region in an image at every training step, thus removing the redundancy of the images and augmenting the dataset by partially occluded versions of existing samples. (Bakkali, et al., 2020)

prior components into one. Also similarly to the aforementioned works they use a late fusion method. To achieve this their meta-classifiers map two outputs, of dimension equal to the number of classes, to one.

For their image model they experimented with two different CNN architectures: AlexNet and VGG16 both with input dimensions of 277 by 277. The output layer is a 16 neuron softmax layer (corresponding to the 16 classes of the RVL-CDIP dataset). With the images being greyscale from RVL-CDIP they convert to RGB and rescale the data to fit the range [-1,1]. Their textual model differs from the previous two mentioned works as the authors of this work do not use word embeddings. They opt for the raw text to be preprocessed into one-hot vectors, i.e., a document is represented by binary vector whose components indicate the presence of the word corresponding to that index, denoted by the authors as a Bag-of-Words followed by the input dimension (e.g.: BoW-100K). These document vectors are fed into a relatively shallow network. Their meta-classifier is a XGBoost model. This model is based on decision tree ensembles. They do not use any regularization parameters instead opting to limit the depth of the trees to control overfitting (to a maximum depth of 3). The authors justify their choice for an XGBoost model by pointing out the minimal tuning required for the classifier. The abstract model architecture is illustrated in Figure 19.

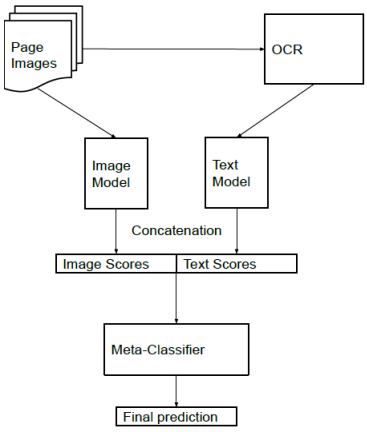

Figure 19 - Abstract Model Architecture (Dauphinee, et al., 2019)

They perform shear transformations and small rotations during training for a better generalization performance. For otimization they use SGD with warm restarts adjusting the learning rate over batches. This otimization method, also used by previous work, tends to find a strong local minimum however it can accelerate past a global minimum.

Their best model was a multimodel consisting of a VGG16 image component and bag of words model with 200 000 vocabulary words used as features. This model achieved a 93.03% test accuracy on the RVL-CDIP dataset, a result higher than many state-of-the-art single model architectures and their own tested single model architectures, with this work also proving that combining both types of features, image and text, improves performance in the document classification task. A final observation by the authors is that the transcriptions errors that ocorred in textual classification component may be accounted for by using more recent word embeddings methods instead of a bag-of-words approach.

Chapter 3 Technical Specification

This chapter covers the general structure and services of the CSW IDV platform, high-level requirements followed by a risk analysis and the tools and methodology adopted during development.

3.1 CSW IDV Platform

The platform, *Intelligent Document Validation* (IDV), developed by CSW, provides services in the area of document analysis. It uses as a *framework,* Flask Python and has a REST API as application programming interface that allows the integration of services in *backend* or *frontend*.

This platform was created with the objective of facilitating processes of organization and management of documents, processes that are still done, largely, manually. It is therefore beneficial that the platform is as versatile as possible, so that it can be integrated in several different situations, being able to analyze multiple types of documents. The format or type of file must become redundant aspects for execution of the services, that is, the execution of these must be possible and efficient for different types of documents.

The services provided by the IDV platform, and a brief description of them, are as follows:

- ***Classification Service*:**
    - Categorizes a document relative to its type and tells the confidence of the forecast (prediction). Tells whether the document is handwritten or printed, if it is in text or form (document with spaces, named fields or placeholders) format.
    - Allows a binary or multi-class classification. In binary classification it is possible to check (binarily) whether the document is of a particular type; in multi-class classification, the

service classifies the document according to a given set of possible categories and indicates confidence for each.

- The service has text classifiers, image classifiers, rule-based classifiers and a composite classifier. As a text classifier, it has a classifier developed using the spaCy library; the image classifier was developed on *Keras* and *Tensorflow*. The service can also be adjusted to include and add different types of classifiers.

- ***Extraction Service*:**
  - Extracts relevant fields for each document type and uses techniques, used according to context. The techniques used are techniques based on the definition of standards and rules as in the case of rules-based systems; NLP and OCR techniques.
  - The service also has several features of image processing and natural language processing.
- ***Validation Service*:**
  - Validates classification and extraction services. Analyzes the other two services in terms of their performance in their tasks. The process consists of validating the classification of the document into a given category or whether the extraction of all relevant characteristics of a submitted document has been carried out.

3.2 High-Level Requirements

The developed hybrid system, and its models, after being validated is intended to integrate in the current IDV Classification Service.

From a high-level point of view the main goals of the internship are:

- The hybrid, multimodal, classifiers must improve the performance upon currently in use single modal classifiers by CSW at document classification;

- Allow, whenever possible, for the automatic training of the IDV platform for new document types, regarding their classification, effectively making the platform more robust to new and different types of documents.

In order to achieve these goals, it is proposed the creation, training and fine-tuning of a multimodal classifier architecture. This multimodal classifier must use current and/or competitive state-of-the-art techniques to be able to apply its multimodal architecture efficiently in a critical environment. The model must also be fine-tuned in order to better classify specific types of documents of interest to CSW while at the same time, if possible, generalize to new and different types of documents.

Given the research heavy nature of this project, high level requirements have proven hard to define, especially given that hybrid architectures are still quite novel in the document classification task, with most relevant works being at most 3 to 4 years old. Still some early (and possibly flexible) high-level requirements can be defined by following the example of recent research done on multimodal classifiers for document classification. The requirements are presented as follows:

- The model(s) must be able to receive, as input data, annotated document images in order to perform supervised learning;
- The multimodal classifier needs to have at the least 3 components: a textual component, a visual component and a hybrid component;
- The textual component must be able to process textual data and extract relevant textual features and feed them to a text-based classifier (the textual data can be obtained via text extraction using CSW own OCR, currently in use in the IDV platform);
- The visual component must be able to process document images and extract from them relevant visual features and feed them to an image-based classifier (e.g., via deep learning methods such as CNNs);

- The hybrid component must be able to join both textual and visual features previously obtained (e.g.: through a late fusion process) in order to accurately use both for document classification;
- The model(s) must be able to output the predicted class for each document example as well the respective confidence in each prediction.

The requirements presented above may not apply to all different approaches that were tested during development. In section 3.4 Methodology and Tools a further detailing of how to achieve these goals and requirements will be presented supported by research and related works on the document classification task.

3.3 Risk Analysis

Risk analysis is an important step in project development in order for it to be successful and a few have been identified and must be taken into consideration during the next phase of this project. Next, it is presented the main risks that have been identified for this project along with a description and a possible mitigation plan:

- **Dataset**: The datasets used for training, validation and testing present a possible risk especially considering that most architectures used for document classification are deep nets and therefore heavily rely on a good quality dataset. Publicly available datasets often consider an ideal, but not realistic, scenario. As an example, in some datasets used in recent literature, images are typically well orientated. This is not a guarantee in a real scenario, among other unique scenarios such as images having curved text or be even slightly folded in some areas. Most of these datasets are made with benchmark and research purposes in mind and may not fare as well in a more dynamic and real scenario. As a counter-measure, pre-processing of the data must be taken into account in order to deal with unique variations of the documents (this includes data augmentation such as including shear-transforming or rotating

images slightly). Experimenting in the future with more than one dataset may also prove useful, possibly with a CSW dataset;

- **Inexperience with some of the more recent and complex architectures and methods**: Most of the development techniques currently being implemented in recent hybrid approaches are relatively novel and inexperience using them is highly possible. A counter-measure for this is to study the available literature, practice with these techniques in both the project and in other applications and also talking to fellow CSW colleagues with more experience in these subjects as well as in professional application development.
- **Lack of information in literature and/or related work**: Most multimodal systems studied in section 2.5.3 provide relevant detail about their hybrid approach and architecture but in a few areas, they can be rather vague about specific topics. Fortunately, most of the individual techniques used by the authors have been studied in detail and there is relevant information online so this risk is possibly not as severe as the others. Mitigation may involve studying the techniques or other aspects individually to better understand and tune them correctly when combining them during the classification task.

3.4 Methodology and Tools

The project methodology was based on SCRUM (Schwaber, 1997). This development methodology (also known as a management framework) has been adopted by CSW for multiple projects including the IDV platform development.

SCRUM was created because of the unpredictable nature of systems development. It does not assume a development cycle can be entirely planned and estimated from the start. SCRUM defines the systems development process as a loose set of activities that combines known, workable tools and techniques with the best that a development team can devise to build systems. An overview of the process is illustrated on Figure 20.

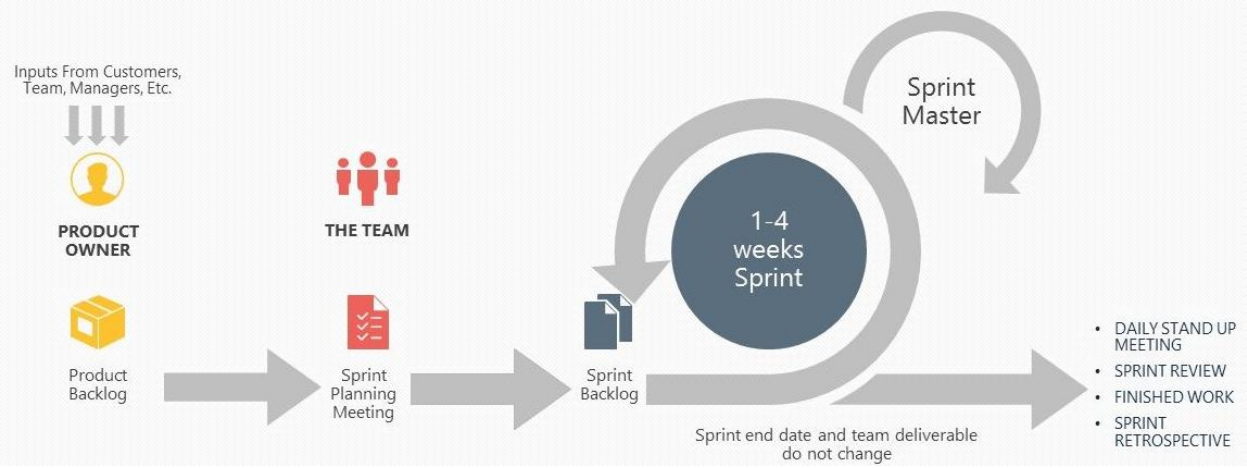

Figure 20 - SCRUM software development process Overview (Anon., 2017)

SCRUM is an iterative (the product is produced during small development cycles) and incremental (product functionality increases during each iteration by adding new features) methodology that aims to make project management a more agile process and adaptable to new and unpredictable situations that may eventually occur during development.

There are three major roles in SCRUM: product owner, scrum master and team member. Product owner acts as a bridge between the client and the development team. This person is responsible for documenting and prioritizing product requirements as well as producing the product backlog. This role is performed, in this project, by CSW's Doctor Tiago Baptista. Scrum master is responsible for team management including team coordination of the team activities, processes implementation and solving problems that may obstruct accomplishing sprint goals, also performed by Doctor Tiago Baptista. Team members consists of the group of developers responsible for completing project tasks. This role is performed by engineers Ana Guarino, Diana Mendes, me and also our technical leader Doctor Rui Lopes.

SCRUM has three main phases: planning and architecture, development and closing. In the first phase, planning and architecture, the product backlog is built. This artifact represents a list of project requirements with descriptions of desired

functionalities as well as risk analysis and initial design of the project's architecture. Development phase consists of an iterative cycle, known as Sprint. Each Sprint starts with a planning meeting where a discussion is had by the team members about ongoing and completed tasks as well as the discussion of any problems or unique situations that rose during the previous Sprint and possible solutions. It is also defined a list of tasks to be performed in the next Sprint (known as the sprint backlog). A Sprint's duration is typically 2 weeks but may be shorter (1 week) or longer (3 to 4 weeks) if necessary. The closing phase consists steps are taken to prepare for the launch of the main deliverable (a product or service) including documentation and development data (such as testing). Daily meetings are also a unique component of SCRUM. They consist of 10 to 15min meetings with all team members and Scrum master where team members discussed tasks performed as well as problems encountered during the previous day and tasks to be performed during the current day. This meeting ensures agreeability between team members in regards to the tasks, problems and effort during a regular basis. Other complementary meetings are the review meeting and the retrospective meeting. In the review meeting, a demonstration of the work is shown to the product owner, at the end of a sprint, and the product owner reviews the work done. In the retrospective meeting, usually taken place at the end of a sprint or at the end of the project, the team reviews the work done and decides on possible strategies, if necessary, to increase effectiveness and improve the current work.

A list of the tools used for project management are as follows:

- **Jira** – Agile projects management software that allows monitoring of development work. This tool is used throughout CSW by most teams for project management;
- **Bitbucket** – Version control tool for Git repositories. This tool is also widely in-use by CSW for version control.
- **Confluence** – Project documentation management tool. Also, a common tool used by CSW for documentation management.

- **Docker** – Tool that allows running applications in containers. This tool contains a virtual development environment that uses resource isolation, which allows developers to develop and run their applications more easily. This tool is used by the CSW IDV team.

The list of technical tools to be used in this project are as follows:

- **Python** – Interpreted, high-level and general-purpose programming language. Common language in AI projects because of its vast collection of development libraries including specific Data Science and Machine Learning libraries.
- **TensorFlow and Keras** – Two open-source libraries available in Python known for their vast number of features related to ML. They're widely used by AI scientific community. They contain many useful tools to create and train a hybrid document classification model, specifically, they allow the creation and use of deep learning methods useful for both text-based and image-based classification. These libraries are also used by CSW IDV team.
- **SpaCy** – Open-source software library for advanced natural language processing, available in Python. This tool allows for a number of basic and advanced NLP tasks to be performed and will be useful for textual data processing. This tool is also already in-use in CSW IDV platform.
- **Tesseract OCR –** Open-source optical character recognition tool. It is one of most widely used OCR tools, primarily in Python. It uses many AI related techniques, such as CV techniques, to extract textual data from scanned images. This tool will be useful to extract and prepare data for later textual feature extraction. This tool is also already in-use in CSW IDV platform.

Chapter 4 Experimentation and Results

This chapter will detail the experiments performed during the development of the hybrid model. It focused mainly on two principal stages: development of a baseline ensemble model detailed in section 4.1 and development of the multi-input, mixed data (hybrid), model detailed in section 4.2.

In order to achieve the goals and high-level requirements set out in section 3.2, it is necessary to create, train and fine-tune a multimodal classifier. From the analysis of literature work it is possible to observe that the implementation of multimodal classifiers has been done successfully in the recent past. All three models studied in section 2.5.3 have used different architectures and yet it is possible to denote a shared high-level structure – all three architectures have three components: a text-based component, a visual-based component and finally a hybrid component. These components work as individual models that can be trained, fine-tuned and tested, especially against their hybrid fusion model as well as the later mentioned baseline model. This allows for direct comparison between the models and to assure the ensemble process for the baseline model and fusion process for the hybrid model are well performed.

4.1 Ensemble Model (Baseline)

The first step in development was to build a baseline. The purpose of this baseline is to represent a solution currently possible to implement using already in-use methods in CSW such as using Tensorflow Keras created image-base models and Spacy text-based models, both already in use in CSW's IDV platform.

A natural approach was then to create an ensemble solution of the problem. Ensemble learning is a process involving multiple models, such as classifiers, where

models are generated and combined in order to improve an original single model-based task. They can improve these tasks by increasing its performance or simply by reducing the number of model candidates for a specific problem. This approach is a simple alternative to constructing a full hybrid classifier since it is not required to fuse textual features with visual features (which is the main challenge when building a hybrid classifier with mixed types of data) and allowed for the use of platforms already in use by CSW.

4.1.1 Dataset Selection

In order to build this ensemble solution and make it comparable to a hybrid approach, the annotated dataset, Tobacco3482, was chosen. This dataset is part of the bigger collection Truth Tobacco Industry Documents. This collection has two major datasets: RVL-CDIP and Tobacco3482. There are significant reasons to consider these candidate sets. The first being the datasets size – annotated document data is usually scarce but these sub-sets (jointly) provide a large number of image examples with over 400000 document image examples. Another reason is class coverage. Tobacco3482 covers 10 different document categories (ADVE, Email, Form, Letter, Memo, News, Note, Report, Resume, Scientific) and RVL-CDIP covers 16 categories (letter, memo, email, file folder, form, handwritten, invoice, advertisement, budget, news article, presentation, scientific publication, questionnaire, resume, scientific report and specification). This high number of classes is important to provide the model with generalization capabilities. A third reason has been their adoption in recent literature to benchmark and compare many document image classification approaches. CSW IDV platform classifiers have in the past been applied to the following document types: passports, receipts, invoices, birth certificates, citizen cards (personal identification), green cards (insurance) and “accident-friendly statements”. Even though there is no overlap between CSW IDV’s tested document types these are still one of the best publicly available options. It is also important to

note that textual content of these datasets was also made publicly available[10] which was useful during the textual model development for either the baseline or hybrid approaches.

Both sub-sets above mentioned were considered but ultimately only one was used, the Tobacco3482 sub-set, in order to minimize training time and resource consumption overall but also because this dataset is unbalanced and more akin to a "real" dataset derived from an IDV application scenario. Due to class unbalance, classification bias for a ruling class was expected to occur which can in turn cause overfitting, especially with an image-based CNN approach which requires a high amount of data. This allows to test whether a hybrid or an ensemble approach is better for a scenario such as this.

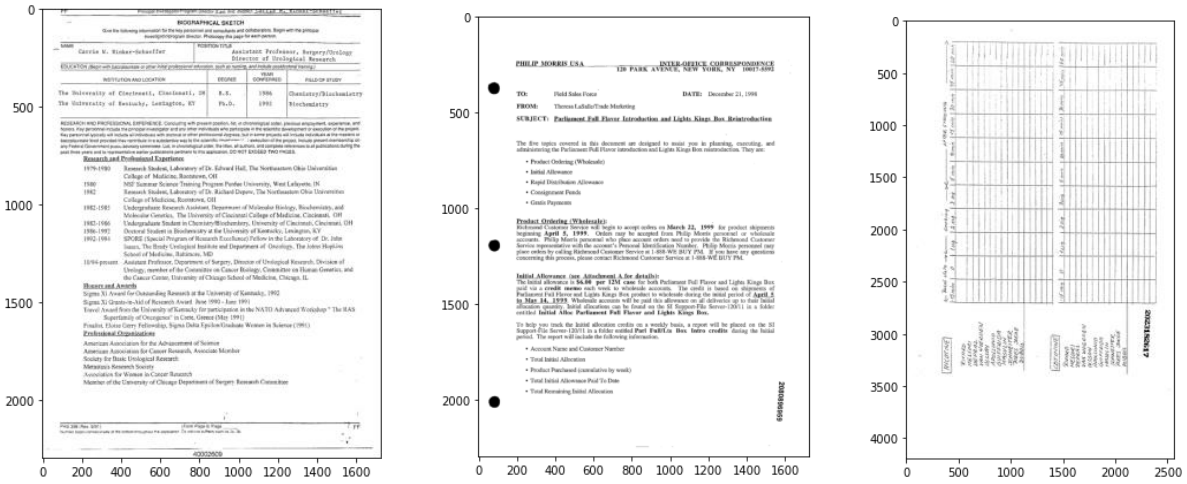

Figure 21 - Sample of the Tobacco3482 dataset, from left to right: examples of 'Resume', 'Memo' and 'Form'

4.1.2 Ensemble Text-based Model

Before producing the ensemble model, previous generation of the individual models is required, starting with the text-based model. The textual content of the Tobacco3482 dataset was extracted via Tesseract OCR, the same OCR used by CSW.

[10] Textual data from the Tobacco3482 and RVL-CDIP datasets using Tesseract OCR v4.0: https://github.com/QuickSign/ocrized-text-dataset/releases/tag/v1.0

With Tesseract it is possible to detect text orientation, perform transformations and rotations if needed and can also identify black text on a white background. To create a text-based model it was used the relatively new Spacy3 library. This library contains a pre-built model of BERT. BERT (Bidirectional Encoder Representations from Transformers) is a powerful transformer that analyses both sides of the sentence with a randomly masked word to make a prediction. In addition to predicting the masked token, BERT predicts the sequence of the sentences by adding a classification token [CLS] at the beginning of the first sentence and tries to predict if the second sentence follows the first one by adding a separation token [SEP] between the two sentences. Its architecture is illustrated in Figure 22.

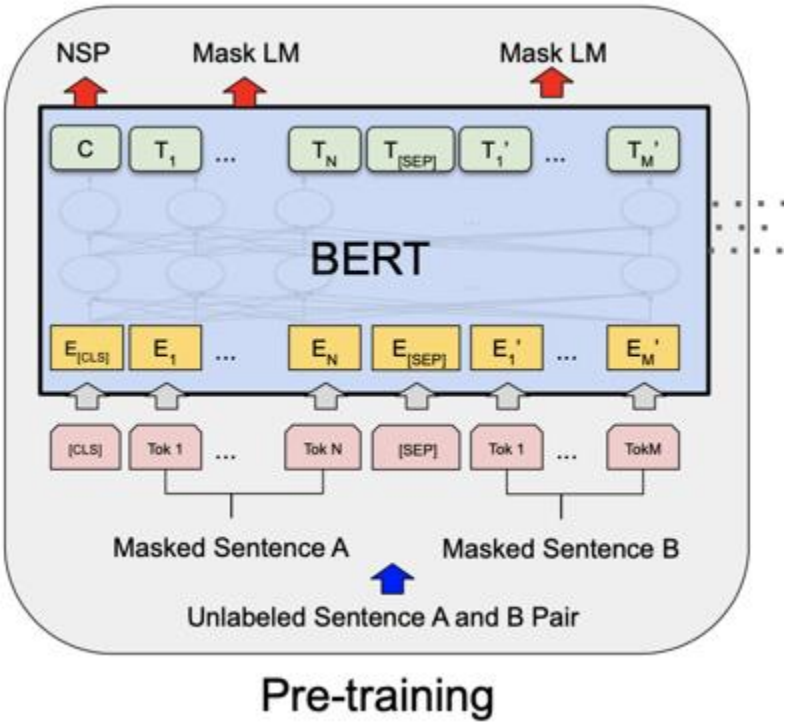

Figure 22 - BERT architecture (Fierro, 2020)

In order to perform classification with Spacy's BERT, pre-processing the data was necessary. The textual data is initially presented in .txt files in directory format. Using a Python script, it was possible to load the textual data into .csv format and

then converting from .csv to the Spacy own's binary file to serve as input for a pre-trained BERT model. Using Spacy Command Line interface, it was possible to train the model with the data being split as such: 80% for training, 10% for validation and 10% for testing. For training it was used a batch size of 16, a pre-built bag of words architecture to process the resulting input vectors from the BERT model, Adam optimizer and a total of 20000 steps. This model achieved an accuracy result of 91.12% for the validation set and 85.91% for the test set.

A second textual classifier was also built using Spacy3 but this time using the RoBERTa transformer variant of BERT. It builds on BERT and modifies key hyperparameters, removing the next-sentence pretraining objective and training with much larger mini-batches and learning rates. This model configured the same way as BERT and resulted in an accuracy 89.31% for the validation set and 82.53% for the test set, worse results than BERT.

4.1.3 Image-based Model

For image classification, two main configurations were used, both using pre-trained CNN architecture models as a base model. The first uses the lightweight MobileNetV2 while the second uses the VGG16 architecture, both pre-trained on the ImageNet dataset. Using these models with transfer learning it is possible to classify images. Given these are both deep neural networks it was predicted that these model's performances were going to be inferior in comparison to the text model configurations described in the previous section 4.1.2. This was expected because the deep neural networks used for image classification are dependent on a high number of image examples for which the dataset size does not suffice. Overfitting was also expected given the class unbalance present in the dataset but such a phenomenon was helpful to analyse whether ensemble could mitigate this issue in a document classification task.

For both configurations raw image data from the Tobacco3482 was loaded and split into an 80% for training, 10% for validation and 10% for testing distribution. Data augmentation was also performed to compensate slightly for the lack of data and to also take into account that in a real scenario resulting scanned images wouldn't be in the "adequate" conditions that the original Tobacco3482 presents. This augmentation consisted of techniques such as shear, zooming randomly 20% of the dataset and vertically flipping the images in order to artificially increase the dataset size.

Starting with MobileNetV2 based model (Table 1), the model consists of the pre-trained MobileNetV2 with its trainable parameters frozen (non-trainable) and fed to a CNN consisting of GlobalAveragePooling2D layer, three Dense layers with 'relu' activation and a 50% chance dropout layer, a Fully Connected Layer layer with 'softmax' activation followed by a classification layer. Its accuracy performance was of 77.79% on the training set, 72.41% for the validation set and 65.35% for the testing set. The number of trainable parameters is 2.8 M along 8 layers. The model's loss and accuracy plots represented in Figure 25, for the first 25 epochs where the loss function stabilizes for both training and validation while accuracy stops increasing for the validation set but not for training (overfits).

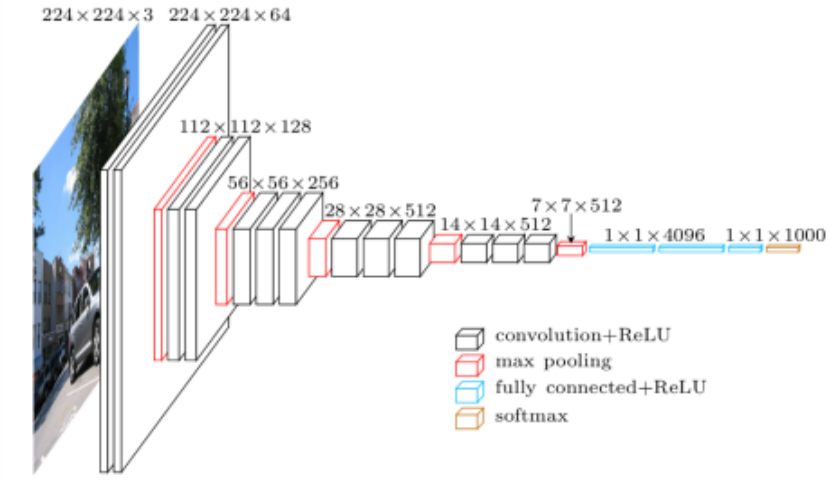

Figure 23 - VGG16 CNN architecture (Thakur, 2019)

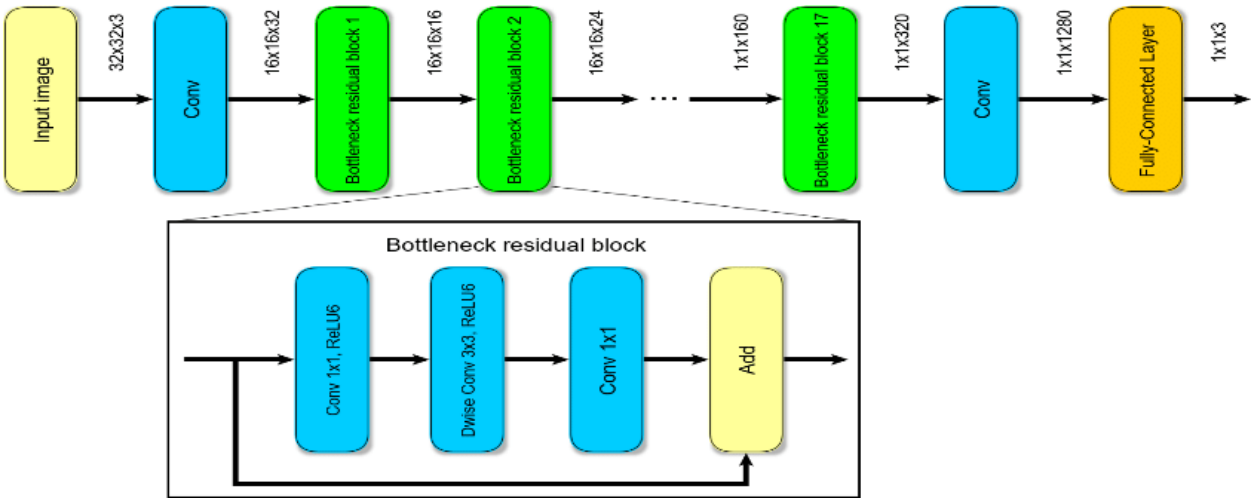

Figure 24 - MobileNetV2 architecture (Seidaliyeva, et al., 2020)

Table 1 - Image-based model #1 archicteture

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functi	(None, 7, 7, 1280)	2257984
global_average_pooling2d_3 (	(None, 1280)	0
dense_9 (Dense)	(None, 1024)	1311744
dense_10 (Dense)	(None, 1024)	1049600
dense_11 (Dense)	(None, 512)	524800
dropout_3 (Dropout)	(None, 512)	0
predictions (Dense)	(None, 10)	5130

Total params: 5,149,258
Trainable params: 2,891,274
Non-trainable params: 2,257,984

Number of trainable weights : 8

VGG16 variant model also had its base model's layers frozen and was followed by a Flatten layer, Dense layer with 'relu' activation and 50% Dropout and

a Fully Connected Layer with 'softmax' activation followed by a classification layer. Its accuracy performance was 94.53% for training set, 74.14% for the validation set and 67.88% for test set, slightly better results than MobileNetV2 but still with noticeable overfitting.

Both models were trained in 100 epochs, learning rate of 0.00005 and with Adam optimizer using Tensorflow Keras.

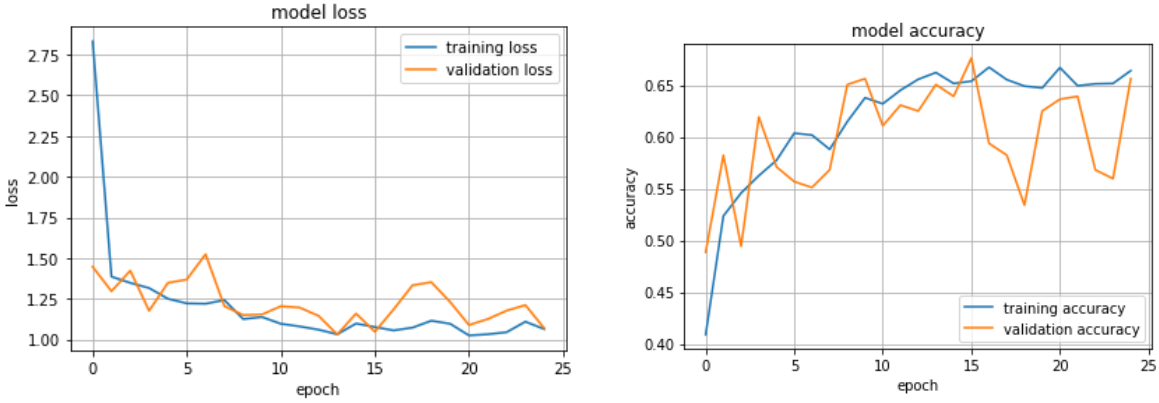

Figure 25 - Model Loss and model Accuracy during training using transfer learning with MobileNetV2 as the base model for 25 epochs

4.1.4 Ensemble Results

Finally, having generated and tuned the individual models, it was possible to perform an ensemble method. The ensemble method used was the classification voting ensemble method. It has two variants: hard-voting and soft-voting. In classification, a hard voting ensemble involves summing the votes for crisp class labels from both the models (a text-based model and an image-based model) and predicting the class with the most votes. A soft voting ensemble involves summing the predicted probabilities for each of the class labels and predicting the class label with the largest sum probability.

Two configurations were used to provide a clearer analysis. The first being an ensemble with MobileNetV2 based image model and the BERT based text model. The second configuration consisted of the VGG16 base image model and Roberta text-

based model. Only the soft voting ensemble was used since it is the appropriate option when the models used in the voting ensemble predict the probability of class membership.

The full results of this baseline approach can be viewed in Table 2.

Table 2 - Ensemble Full Results - Training / Validation Accuracy / Text Model Testing Metrics / Visual Model Testing Metrics / Ensemble Testing Metrics / Accuracy per class for testing set for image-based, text-based, ensemble models, respectively.

MODEL / DATASET	Train / Val Accuracy	Text Model (Test)	Visual Model (Test)	Ensemble (Test)	Accuracy per Class (Test) (Img | Txt | Ensemble - Category (Effect))
- Text Model: roberta_spacyMode_base_custom2 - Image Model: VGG16_Tobacco3482_100epochs_Adam_lr1e-05 - Dataset: Tobacco3482	Image: 0.9453 / 0.7414 (overfit) Text: n.d / 0.9112	Acc: 0.8591 Precision-Score (macro): 0.8391 Precision-Score (micro): 0.8591 Precision-Score (weighted): 0.8632 Recall-Score (macro): 0.8207 Recall-Score (micro): 0.8591 Recall-Score (weighted): 0.8591 F1-Score (macro): 0.8259 F1-Score (micro): 0.8591 F1-Score (weighted): 0.8581	Acc: 0.6788 Precision-Score (macro): 0.7233 Precision-Score (micro): 0.6788 Precision-Score (weighted): 0.6975 Recall-Score (macro): 0.6126 Recall-Score (micro): 0.6788 Recall-Score (weighted): 0.6788 F1-Score (macro): 0.6286 F1-Score (micro): 0.6788 F1-Score (weighted): 0.6666	- Soft Voting: Acc: 0.8732 Precision-Score (macro): 0.8558 Precision-Score (micro): 0.8732 Precision-Score (weighted): 0.8776 Recall-Score (macro): 0.8410 Recall-Score (micro): 0.8732 Recall-Score (weighted): 0.8732 F1-Score (macro): 0.8425 F1-Score (micro): 0.8732 F1-Score (weighted): 0.87121	0.83 | 0.75 | 0.75 - Scientific (Neutral) 0.86 | 0.91 | 0.91 - Letter (Improved) 0.73 | 0.97 | 0.97 - Email (Improved) 0.69 | 0.62 | 0.86 - Note (Improved) 0.79| 0.94| 0.98 - Memo (Improved) 0.75| 0.92| 0.92 - Resume (Improved) 0.48| 0.67| 0.63 - Report (Reduced) 0.52| 0.86| 0.91 - Form (Improved) 0.25| 0.75| 0.75 - News (Neutral) 0.19| 0.72| 0.68 - ADVE (Reduced/Neutral)
- Text Model: bert_spacyMode_base_custom2 - Image Model: models/MobileNetV2_Tobacco3482_100epochs_ Adam_lr1e-05.h5 - Dataset: Tobacco3482	Image: 0.7779 / 0.7241 Text: n.d / 0.8931	Acc: 0.8253 Precision-Score (macro): 0.8105 Precision-Score (micro): 0.8253 Precision-Score (weighted): 0.8389 Recall-Score (macro): 0.7851 Recall-Score (micro): 0.8253 Recall-Score (weighted): 0.8253 F1-Score (macro): 0.7866 F1-Score (micro): 0.8253 F1-Score (weighted): 0.8223	Acc: 0.6535 Precision-Score (macro): 0.6576 Precision-Score (micro): 0.6535 Precision-Score (weighted): 0.6824 Recall-Score (macro): 0.6255 Recall-Score (micro): 0.6535 Recall-Score (weighted): 0.6535 F1-Score (macro): 0.6196 F1-Score (micro): 0.6535 F1-Score (weighted): 0.6514	- Soft Voting: Acc: 0.8169 Precision-Score (macro): 0.7762 Precision-Score (micro): 0.8169 Precision-Score (weighted): 0.8318 Recall-Score (macro): 0.7741 Recall-Score (micro): 0.8169 Recall-Score (weighted): 0.8225 F1-Score (macro): 0.7590 F1-Score (micro): 0.8169 F1-Score (weighted): 0.8142	0.75 | 0.86 | 0.75 - Scientific (Neutral) 0.81 | 0.88 | 0.94 - Letter (Improved) 0.61 | 0.97 | 0.97 - Email (Improved) 0.68 | 0.81 | 0.86 - Note (Improved) 0.66 | 0.98 | 0.98 - Memo (Improved) 0.80 | 0.92 | 0.92 - Resume (Improved) 0.90 | 0.48 | 0.41 - Report (Reduced) 0.48 | 0.75 | 0.66 - Form (Reduced) 0.33 | 0.52 | 0.65 - News (Improved) 0.16 | 0.60 | 0.52 - ADVE (Neutral)

An analysis of Table 2 for both Configuration #1 and Configuration #2 is as follows:

Configuration #1

- Ensemble overall improved performance across categories but had a neutral effect on others, resulting in a model with higher performance than both text (no overfitting or underfitting) and much higher than image (overfitting).
- Scientific, News and ADVE performance had a neutral effect. News and ADVE have both poor results with image classifier but high with text, ensemble has similar performance to text.
- Report had its performance slightly reduced.

- All other 6 categories had their performance increased even if just slightly. In all these categories the image classifier performance was lower than text classifier performance.
- Other relevant notes: News and ADVE have very low performance with image classifier and low performance with text classifier. Report has low performance on both with image being the worst.

Configuration #2

- Ensemble overall improved performance across categories but also had it reduced for a small number of categories resulting in a model which is better than the image-based model (which suffers from overfitting) but similar in performance to the text classifier (which does not suffer from overfitting or underfitting)
- Scientific and ADVE had a neutral impact, i.e., the ensemble performance was not the highest
- Report and Form had their performance reduced. They both share the fact that one native model does very well and the other does poorly
- The remaining 6 categories had their performance increased. Significant improvement is in the News category which both models provide poor results.
- Other relevant notes: ADVE, News very low perf. on Image Classifier and low on Text Classifier; Report in uniquely low on Text but high on Image

From the analysis above it is possible to conclude that ensemble helped mitigate the overfitting phenomenon happening with the image-based and managing to achieve higher performances than the powerful transformer models.

4.2 Hybrid (Multi-Input) Model

To generate a hybrid model resulting from both textual and visual features it was chosen to adopt the literature stance on the use of late fusion process. This method

consists of both models, after individual training and fine-tuning, join together through a fusion process, i.e., both text and visual feature vectors need to be combined into a single feature vector and once again trained by a classifier.

To achieve late fusion two main methods, exist: concatenation of feature vectors or averaging of feature vectors. Averaging is dependent of the alignment of both feature vectors but does tend to produce slightly better results while concatenating is more straightforward and leaves it up to the hybrid classifier to combine both domains. The method used was concatenation because averaging the alignment of both classifiers would prove a complex task, given not only the different nature of the features used but also different layer types used to produce each input.

The dataset used for the hybrid model training, validation and testing was also Tobacco3482. The image-based model used was the same configuration used for the baseline ensemble model, which used MobileNetV2 base model with transfer learning described in section 4.1.2.

Tensorflow and respective Keras API provided the required methods in order to generate end-to-end hybrid models, i.e., models with mixed types of data, textual and visual, as well as with multiple-input, single-output. An obstacle though was that a new text-based individual model needed to be generated in order to be compatible with the concatenation fusion process available in Tensorflow's Keras which will be detailed in the following section 4.2.1.

4.2.1 Hybrid Text-based Model

The new text-based model needed to create the hybrid model was generated via Tensorflow Keras, more specifically, the repository Tensorflow Hub. Tensorflow Hub is a repository with available ready-to-use models. One of these models is the BERT model, more specifically, the bert english cased variant. The model loaded from this repository. Next, the text input data was transformed into numeric token arranged in several Tensors before being put into BERT. After being processed by the BERT

model, tensorflow returns a custom keras layer with the model's output. By doing this it is possible to use this layer, as transfer learning, to have the result BERT model output as input for a new text-based classifier. The model is then added upon with a 10% Dropout layer and a Dense layer with 'softmax' activation ().

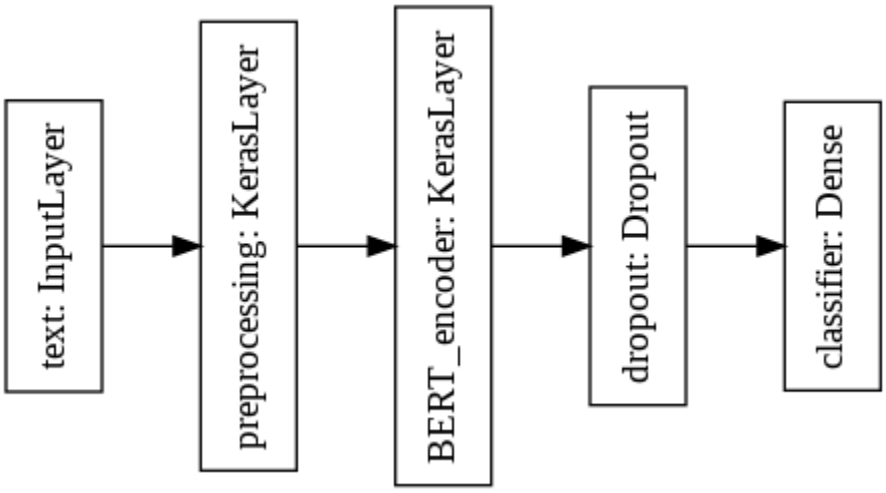

Figure 26 - Hybrid Text-based model

Training of this model was performed using a custom Adam with weigh decay optimizer, learning rate of 0.00003 and only 5 epochs.

The model's performance was 93.78% accuracy on training, 86.67% on validation and 79.15% on the test set. Good results given the relatively simple architecture used. The training and validation loss and accuracy plots are represented in Figure 27. In this figure, for training, it is possible to see that the loss consistently decreases over time and stabilizes at the end while the accuracy consistently increases over time also stabilizing at the end. For the validation set both the loss and accuracy stabilize much sooner and decrease and increase, respectively, at a much slower pace, a possible sign of overfitting due to lack of data.

4.2.3 Late Fusion process

As stated above, the late fusion process was achieved via Tensorflow Keras library which allowed for the generation of an end-to-end hybrid model on mixed data

inputs. The way this achieved is by loading both models and setting its parameters to non-trainable (freeze layers). This way the model only trains the new introduced layers. Removal of the last layer of both models (Dense layer with 'softmax' activation) was also performed in order to concatenate both feature vectors resulting on the penultimate layer. The concatenated output of both these feature vectors can then be used as the input for a follow-up layer able to perform inference on combined feature vector input, effectively producing a hybrid end-to-end classifier. The resulting model architecture is illustrated in Figure 28. The presented architecture is then followed by a SoftMax layer and a classification layer.

Unfortunately, due an error regarding the input data for the hybrid model, resulting in the model failing to adapt both inputs into the required format specificized by the TensorFlow API it was not possible to proceed with the hybrid training and consequent testing. This error was left unfixed due to time. Further discussion and conclusion of results will made taking this fact into consideration.

Figure 27 - Loss and Accuracy plots for both training and validation using the Tensorflow Keras Pre-trained BERT Model

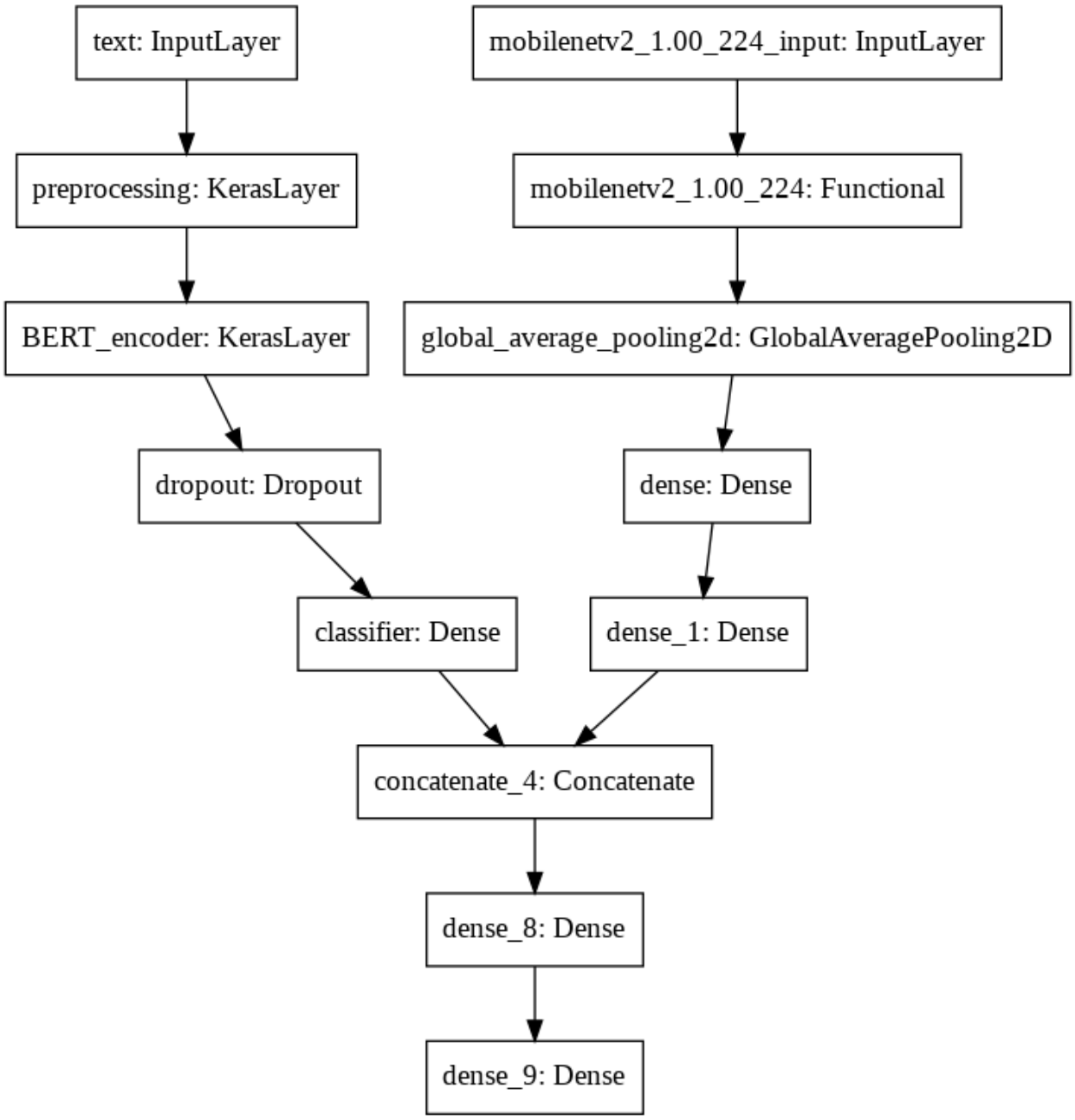

Figure 28 - Hybrid Model Architecture With Multiple Inputs

Chapter 5 Conclusion and Future Work

In this work it was presented multiple ways to consider the use of both textual and visual data for document classification. The first being individual training of each model in regards to its feature type followed by the employment of an ensemble method and finally with an end-to-end multiple-input, hybrid, model. It was possible to conclude by analysing the ensemble classifier results that this technique is simpler and yet effective in achieving good results if at least one of the models is well tuned such as BERT textual model. Therefor it presents itself as good choice if there's not a significant amount of available data and if a deep learning model might prove insufficient, allowing for the use of an ensemble method paired with, per example, a text-based model generated via powerful transformers such as the ones used – which were able to achieve good results with low fine-tuning and with the small, unbalanced, dataset provided. Even tough unable to prove it in this work, the literature in section 2.5.3 concludes that hybrid models are effective in increasing individual model's performance via a fusion process. This process however is much more complex than a simple ensemble and requires both models to be, at least, acceptable for it to concatenate both feature vectors in a way that will favour performance. This said, hybrid models are ideal over ensemble models when using deep learning, if high amount of data is available, and both models can benefit from it.

Some final remarks: it is regrettable that training and evaluation of the hybrid model was not possible in this project lifespan and the immediate future work would be to fix the error stated in section 4.2.3. Other possible future work would be to test both ensemble and hybrid models on a large dataset such as RVL-CDIP which was originally intended but due to time and other reasons stated in section 4.1.1 was scrapped in favour for the Tobacco3482 smaller dataset. And finally, also testing both models with a CSW specific dataset would be interesting to see how the model performs in a "real" scenario and compare it to Tobacco3482 and RVL-CDIP.

Bibliography

Abdulkader, A. & Casey, M. R., 2009. Low Cost Correction of OCR Errors Using Learning in a Multi-Engine Environment. *10th International Conference on Document Analysis and Recognition, ICDAR*, 26-29 July, pp. 576-580.

Afzal, M. Z. et al., 2015. Deepdocclassifier: Document classification with deep Convolutional Neural Network. *13th International Conference on Document Analysis and Recognition (ICDAR)*, 23-26 August.

Alammar, J., 2021. *Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention).* [Online]
Available at: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/
[Acedido em 1 September 2021].

Amidi, A. & Amidi, S., 2020. *Recurrent Neural Networks.* [Online]
Available at: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
[Acedido em 26 December 2020].

Anon., 2017. *https://pslides.com/templates/scrum-software-development-process/*. [Online]
Available at: https://pslides.com/templates/scrum-software-development-process/
[Acedido em 15 January 2021].

Audebert, N., Herold, C., Slimani, K. & Vidal, C., 2019. Multimodal deep networks for text and image-based document classification. *Conférence Nationale sur les Applications Pratiques de l'Intelligence Artificielle (APIA) hal-02163257*, 2019 July.

Bakkali, S., Ming, Z., Coustaty, M. & Rusiñol, M., 2020. Visual and Textual Deep Feature Fusion for Document Image Classification. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 14-19 June.

Brownlee, J., 2019. *What Are Word Embeddings for Text?*. [Online] Available at: https://machinelearningmastery.com/what-are-word-embeddings/ [Acedido em 11 January 2021].

Byun, Y. & Lee, Y., 2000. Form Classification Using DP Matching. *Proceedings of the 2000 ACM symposium on Applied computing - Volume 1*, January.

Chen, N. & Blostein, D., 2006. A survey of document image classification: problem statement, classifier architecture and performance evaluation. *International Journal of Document Analysis and Recognition (IJDAR) 10 1–16*, 03 August.

Cho, K. et al., 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078v3*, 3 June.

Daly, M., 2019. *Natural Language Processing Classification Using Deep Learning And Word2Vec*. [Online] Available at: https://towardsdatascience.com/natural-language-processing-classification-using-deep-learning-and-word2vec-50cbadd3bd6a [Acedido em 11 January 2021].

Dauphinee, T., Patel, N. & Rashidi, M., 2019. Modular Multimodal Architecture for Document Classification. December.

Eikvil, L., 1993. Optical Character Recognition. *citeseer. ist. psu. edu/142042. html.*.

El-Amir, H. & Hamdy, M., 2019. Convolutional Neural Network. *Deep Learning Pipeline*, 21 December, pp. 367-413.

Fierro, C., 2020. *A Light Introduction to BERT*. [Online] Available at: https://medium.com/dair-ai/a-light-introduction-to-bert-2da54f96b68c

Goldberg, Y. & Hirst, G., 2017. Neural Network Methods in Natural Language Processing. *Morgan & Claypool*.

Harley, A. W., Ufkes, A. & Derpanis, K. G., 2015. Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval. *13th International Conference on Document Analysis and Recognition (ICDAR)*, August, pp. 991-995.

He, K., Zhang, X., Ren, S. & Sun, J., 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV*, pp. 770-778.

Hinton, G. & Sejnowski, T. J., 1999. *Unsupervised Learning: Foundations of Neural Computation*. s.l.:The MIT Press.

Hochreiter, S. & Schmidhuber, J., 1997. Long Short-term Memory. *Neural Computation*, December, pp. 9(8):1735-80.

Honnibal, M., 2015. *Introducing spaCy*. [Online] Available at: https://explosion.ai/blog/introducing-spacy [Acedido em 13 January 2021].

IBM Cloud Education, 2020. *Recurrent Neural Networks*. [Online] Available at: https://www.ibm.com/cloud/learn/recurrent-neural-networks [Acedido em 27 December 2020].

J. Kumar, P. Y. a. D. D., 2012. Learning document structure for retrieval and classification. *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), Tsukuba,*, pp. 1558-1561.

Kaelbling, L. P., Littman, M. L. & Moore, A. W., May 1, 1996. *Reinforcement Learning: A Survey*. Vol. 4 (1996) ed. s.l.:JAIR - Journal of Artificial Intelligence Research.

Kahle, B., 2020. *FOSS wins again: Free and Open Source Communities comes through on 19th Century Newspapers (and Books and Periodicals…)*. [Online] Available at: https://blog.archive.org/2020/11/23/foss-wins-again-free-and-open-source-communities-comes-through-on-19th-century-newspapers-and-books-and-

periodicals/
[Acedido em 28 December 2020].

Karandish, F., 2019. *The Comprehensive Guide to Optical Character Recognition (OCR).* [Online]
Available at: https://moov.ai/en/blog/optical-character-recognition-ocr/
[Acedido em 28 12 2020].

Krizhevsky, A., Sutskever, I. & Hinton, G. E., 2012. ImageNet classification with deep convolutional neural networks. *NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, December, Volume 1, p. 1097–1105.

Kumar, J., P. Y. & D. D., 2014. *Pattern Recognition Letters Vol. 43*, 1 July, pp. 119-126.

Kumar, J., Ye, P. & Doermann, D., 2014. Structural similarity for document image classification and retrieval. *Pattern Recognition Letters*, 1 June, Volume 43, pp. 119-126.

L. Hao, L. G. X. Y. a. Z. T., 2016. A table detection method for pdf documents based on convolutional neural networks.. *12th IAPR Workshop on Document Analysis Systems*, p. 287–292.

Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P., 1998. Gradient-based learning applied to document recognition. *in Proceedings of the IEEE*, Nov, Volume 86, pp. 2278-2324.

Lim, C., 2019. An Evaluation of Machine Learning Approaches to Natural Language Processing for Legal Text Classification. September.

Li, Y. & Yang, T., 2017. Word Embedding for Understanding Natural Language: A Survey. *Guide to Big Data Applications*, May.

Lopez, R., 2020. *Understanding the perceptron neuron model.* [Online]
Available at: https://www.neuraldesigner.com/blog/perceptron-the-main-component-

of-neural-networks
[Acedido em 22 December 2020].

Luong, M.-T., Pham, H. & Manning, C. D., 2015. Effective Approaches to Attention-based Neural Machine Translation. *Computation and Language (cs.CL)*, 20 September.

Manevitz, L. M. & Yousef, M., 2001. One-Class SVMs for Document Classification. *Journal of Machine Learning Research 2*, pp. 139-154.

McGonagle, J., Shaikouski, G. & Williams, C., 2020. *Backpropagation.* [Online] Available at: https://brilliant.org/wiki/backpropagation/ [Acedido em 12 23 2020].

Mikolov, T., Chen, K., Corrado, G. & Dean, J., 2013. Efficient estimation of word representations in vector space. *arXiv:1301.3781*, January .

Minaee, S. et al., 2020. Deep Learning Based Text Classification: A Comprehensive Review. 6 April.

Mitchell, T. M., 1997. *Machine Learning*. New York: McGraw-Hill.

Nair, V. & Hinton, G. E., 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the International Conference on Machine Learning*.

Nunes, N. F. R., 2019. Deep Learning for Automatic Classification of Multi-Modal Information Corresponding to Chest Radiology Reports, Instituto Superior Técnico de Lisboa. October.

Peters, M. E. et al., 2018. Deep contextualized word representations.

Ratner, A., Bach, S., Varma, P. & Ré, C., 2017. *Weak Supervision: The New Programming Paradigm for Machine Learning.* [Online] Available at: https://dawn.cs.stanford.edu/2017/07/16/weak-supervision/

Rosenblatt, F., 1957. *The Perceptron, a Perceiving and Recognizing Automaton Project Para.* s.l.:Cornell Aeronautical Laboratory.

RUDER, S., 2016. *An overview of gradient descent optimization algorithms.* [Online] Available at: https://ruder.io/optimizing-gradient-descent/ [Acedido em 23 December 2020].

Ruder, S., 2017. An overview of gradient descent optimization algorithms. *CoRR, abs/1609.04747*, 15 June.

Russell , S. J. & Norvig, P., 2020 (4th Ed.). *Artificial Intelligence: A Modern Approach.* s.l.:Prentice Hall.

Saha, S., 2018. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way.* [Online] Available at: towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Sandler, M. et al., 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18-23 June.

Schwaber, K., 1997. SCRUM Development Process. *Business Object Design and Implementation*, pp. 117-134.

Seidaliyeva, U., Akhmetov, D., Matson, E. T. & Ilipbayeva, L., 2020. Real-Time and Accurate Drone Detection in a Video with a Static Background. July.

Shin, C., Doermann, D. & Rosenfeld, A., 2001. Classification of document pages using structure-based features. *International Journal on Document Analysis and Recognition*, p. 232–247.

Simonyan, K. & Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *CoRR, abs/1409.1556.*.

Strachnyi, K., 2019. *Brief History of Neural Networks.* [Online] Available at: https://medium.com/analytics-vidhya/brief-history-of-neural-networks [Acedido em 11 Novembro 2020].

Sutskever, I., Vinyals, O. & Le, Q. V., 2014. Sequence to Sequence Learning with Neural Networks. *arXiv:1409.3215*.

Szegedy, C., Ioffe, S., Vanhoucke, V. & Alemi, A. A., 2017. Inception-v4, inception-ResNet and the impact of residual connections on learning. *AAAI'17: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, February, p. 4278–4284.

Tensmeyer, C. & Martinez, T., 2017 . Analysis of Convolutional Neural Networks for Document Image Classification. *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 9-15 November, pp. 388-393.

Tensmeyer, C. & Martinez, T., 2017. Analysis of Convolutional Neural Networks for Document Image Classification. *14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto*, pp. 388-393.

Thakur, R., 2019. *Step by step VGG16 implementation in Keras for beginners.* [Online] Available at: https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

Tolpygo, A., 2016. *Natural Language Processing: Document Classificiation.* [Online] Available at: https://www.sflscientific.com/data-science-blog/2016/4/17/natural-language-processing-document-classification
[Acedido em 11 01 2021].

Vaswani, A. et al., 2017. Attention Is All You Need. 6 December, p. 15 pages.

Willis, N., 2006. *Google's Tesseract OCR engine is a quantum leap forward.* [Online].

Yang, X. et al., 2017. Learning to Extract Semantic Structure from Documents Using Multimodal Fully Convolutional Neural Network. *arXiv:1706.02337*, 7 June.

Yang, Z. et al., 2019 . XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv:1906.08237*, 19 Jun.

Yang, Z. et al., 2016. Hierarchical Attention Networks for Document Classification. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June, p. 1480–1489.

Yse, D. L., 2019. *Your Guide to Natural Language Processing (NLP) How machines process and understand human language.* [Online] Available at: https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1
[Acedido em 11 January 2021].

Zoph, B., Vasudevan, V., Shlens, J. & Le, Q. V., 2018. Learning Transferable Architectures for Scalable Image Recognition. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 8697-8710*, June.