FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Rotated Filters and Learning Strategies in Convolutional Neural Networks for Mammographic Lesions Detection

Eduardo Meca Castro

MASTER THESIS

Mestrado Integrado em Bioengenharia

Supervisor: Jaime S. Cardoso, Phd

Co-Supervisor: José Costa Pereira, Phd

July 19, 2017

Rotated Filters and Learning Strategies in Convolutional Neural Networks for Mammographic Lesions Detection

Eduardo Meca Castro

Mestrado Integrado em Bioengenharia

July 19, 2017

Resumo

O cancro da mama é o mais letal entre as mulheres. Estima-se que 520 mil mortes anuais sejam causadas por esta doença. Por este motivo, programas de rastreio mamário foram implementados em vários países, o que permitiu reduzir significativamente a taxa de mortalidade entre as mulheres mais velhas, antecipando o diagnóstico. Contudo, o número de falsos negativos nestes programas é ainda elevado.

Alguns autores sugeriram sistemas de diagnóstico assistido por computador (CAD) que, em conjunto com os especialistas, permitem aumentar a precisão da decisão. Embora esteja demonstrado que estes sistemas melhoram o diagnóstico, ainda são encontrados muitos falsos positivos.

O renovado interesse da comunidade científica em métodos de Deep Learning tem demonstrado a sua eficácia em reconhecimento de imagens naturais. Relativamente a imagens médicas, o estudo destes modelos é recente mas já demonstrou alguns bons resultados. Contudo, existem diferenças significativas entre datasets de imagens naturais e médicas.

Neste trabalho, um sistema de detecção automática de lesões mamográficas é proposto no contexto do cancro da mama. Para isso foram utilizadas redes convolucionais. Para lidar com diferenças entre dados naturais e médicos três novas metodologias foram estudadas.

Primeiro foram testadas mudanças na arquitectura dos modelos, que reduzem o número de parâmetros a optimizar, funcionando como uma estratégia de regularização. Estas alterações são baseadas no conhecimento prévio de que lesões não têm orientação e é aplicada individualmente a cada camada convolucional e a primeira camada densa dos modelos. Os ganhos em performance obtidos são muito pequenos o que motiva um estudo mais detalhado deste método, em cenários diferentes.

De seguida uma estratégia de Rank Learning é estudada como forma de aumentar o número de dados disponível para treinar modelos convolucionais. Assim é imposta uma aprendizagem baseada em diferenças entre casos positivos e negativos. Os resultados obtidos são semelhantes a métodos de classificação convencionais, apesar de ser necessária mais memória durante o processo de treino. Concluiu-se que esta estratégia pode ser utilizada como uma alternativa para datasets onde existe desproporção entre o número de casos de cada classe.

Finalmente, uma estratégia Cascade é proposta. Neste caso são usados dois modelos, o primeiro optimizado usando todos os casos de treino, enquanto o segundo apenas vê os casos classificados como positivos pelo modelo anterior. Juntamente com o oversampling de casos positivos é possível desta forma criar modelos com um bias cada vez mais pequeno a favor da classe minoritária. Esta estratégia revelou-se eficaz.

O sistema proposto tem uma performance ligeiramente inferior aos sistemas CAD tradicionais (80% de lesões detectadas com 2 falsos positivos por imagem). Maior sensibilidade para lesões malignas e pequeno custo computacional durante o teste são alguns aspectos positivos desta abordagem ao problema. Formas de melhorar a performance do sistema são discutidas como trabalho futuro.

Abstract

Breast cancer is the most lethal form of cancer among women. It is estimated that 520 thousand deaths are caused by this disease each year. Due to this, breast screening programs were put in place. These have significantly reduced the mortality rate among older women, by early diagnosis. However, the rate of false negative diagnosis is still high.

To help deal with this, some authors have suggested computer aided diagnosis systems, which assist specialists increasing decision accuracy. Although these have been shown to improve diagnosis, these methods still deliver many false positives.

The recent surge of Deep Learning methods has shown how effective these can be in natural image recognition tasks. The research in deep learning applied to medical image is recent and has already yielded some positive results. However, several differences exist between natural images datasets and medical ones.

In this work we propose a lesion detection scheme for mammographic images in the context of breast cancer. We focus on the use of deep convolutional neural networks as detectors for this problem. To deal with differences between natural and medical data we study three different methodologies.

First, architectural changes are made to the model as a regularization strategy. For this, we use the field knowledge that lesions do not have orientation, to design a version of the convolutional layer with less parameters. We show that this technique can also be applied to the first dense layer of the model. The gains from using this method are very small, motivating further experiments in different settings to assess how well they are doing.

Second, a rank learning strategy is used to increase the amount of data available for training. Conceptually, this works by forcing the model to learn differences between positive and negative examples. The performance of this method was similar to conventional classification, although it required more memory. This strategy can be seen as an alternative to train models in heavily unbalanced data.

Third, a cascade approach was proposed. This learning strategy works by optimizing a sequence of two models, where the first learns with all initial data, while the second only sees true positives and false positives from the previous model. By oversampling the minority class we are able to create models sequentially less biased towards positives. This strategy proved very effective.

The proposed scheme performed slightly worse than typical CAD systems (80% detection rate at two false positives per image). Good aspects of the final model include increased sensitivity towards malignant lesions and small computational cost when running for new images. We also discuss future work that could be used to enhance system performance.

Agradecimentos

Este trabalho apenas foi possível com vários apoios, pelos quais estou eternamente grato.

À Universidade do Porto, à Faculdade de Engenharia e ao INESC TEC por me terem permitido realizar este trabalho.

Aos meus orientadores Jaime dos Santos Cardoso e José Costa Pereira que foram fundamentais para toda a aprendizagem conseguida ao longo do semestre.

Aos meus pais e irmãos, sempre presentes.

Aos meus amigos, António, Cafeses, Vítor, Vanessa e Jorge pelo apoio constante durante o desenvolvimento deste trabalho.

Eduardo Meca Castro

"It is not that the meaning cannot be explained. But there are certain meanings that are lost forever the moment they are explained in words."

Haruki Murakami, 1Q84

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Breast Cancer Worldwide

Breast cancer poses a massive health problem worldwide. Alone, it accounts for 15% of all cancer deaths in females, being the most lethal in this group (Torre et al., 2015). It's estimated that 1.7 million new cases and 520 thousand deaths happen every year. Less than one percent of cases are developed by man (Breastcancer.org, 2017). The incidence of the disease is unevenly distributed across countries of different development levels with westernized areas having an age-standardized incidence rate of 74.1, more than two times superior to that of less developed areas. This disparity is often attributed to lifestyle differences as women with less and later births and exposed to exogenous hormones (oral contraceptives and hormone replacement therapy) have a bigger risk of developing breast cancer (Torre et al., 2015; Bray et al., 2004). Many countries in South America, Africa and Asia have seen an increase in incidence and mortality, often attributed to women behavioral changes. This could indicate that the problem will become even more demanding as developing countries transition culturally. In Portugal, breast cancer ranks second in terms of age-standardized incidence, right after prostate cancer (men specific), with 85.6 and fourth in mortality with 18.4. If we only account women, this is the most common and lethal form of cancer in our country (World Health Organization, 2012).

In addition to its social impact among older women, there is also a big economic burden associated with the disease. The mean allowed cost per patient in the United States is 80 715€, for the year after diagnosis, and 20 822€ for the second (Blumen et al., 2016). There are also opportunity costs associated with loss of productivity, in the form of morbidity and mortality. In fact, it is estimated that in the European Union, only 40% of cancer associated costs are related with health-care. In the case of Portugal this number drops to nearly 25% with productivity loss in the form of mortality taking more than half the total costs (Luengo-fernandez et al., 2008).

One of the forms of reducing the mortality of breast cancer is early diagnosis. The 5-year survival rates for stage 0 or stage I detected cancers is nearly 100% . For stage II the number drops to 93% (American Cancer Society, 2016). At these stages, the cancer is asymptomatic and so, the only way to detect it is through screening the population with higher risk. For more advanced

stages the survival rate is only 73% (III) and 22% (IV). This means that an early detection of the disease could reduce its social impact among women and their families. In economic terms there is also much to gain, as an early detection would decrease treatment and loss of productivity costs (Blumen et al., 2016).

There is enough evidence today to say that screening mammography reduces the mortality rate for women who are 40 years of age or older. Women in the 50 to 69 years age group, who are in screening programs, had its mortality rate reduced 16% to 35%. For the group of 40 to 49 years old, the reduction is between 15% and 20%. For younger women the benefit is much smaller, due to lower incidence, faster tumor growth and harder diagnosis (Pisano et al., 2005). In the United states, the cost of screening the Medicare Population is 1.08 billion, a little smaller than the total cost of treatment of 1.36 billion. However, as argued previously, the gains of screening reduces the costs of treatment and loss of productivity opportunity cost. More importantly, the social aspect of reduced mortality, treatment years and quality of life is enough to justify the investment in screening (Killelea et al., 2014).

In Portugal, mammography screening every 2 years is indicated for asymptomatic women with more than 49 years. For women older than 69 years, this screening can alternatively be done every 3 years (Direção-Geral de Saúde, 2011).

1.2 Breast Cancer Physiology and X-ray Imaging

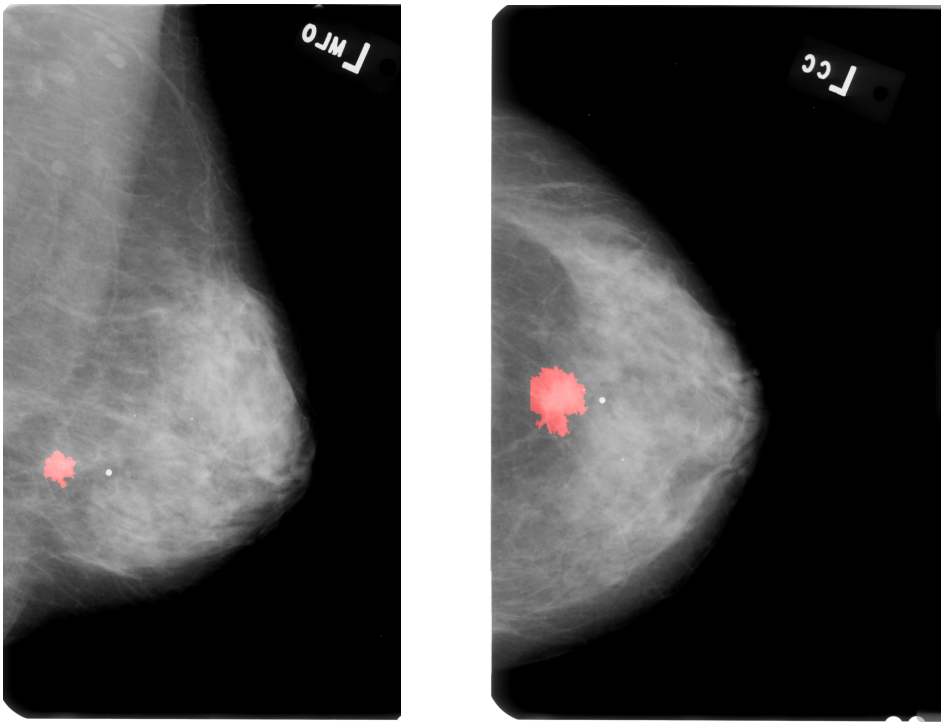
The breast is a very particular organ for three reasons: 1) its primary function is nutritional support to the infant, 2) its structure changes in adulthood and 3) it has a cultural, social and personal significance. These factors influence the diagnosis and treatment of breast diseases (Drake et al., 2009). Anatomically the breast is constituted mostly by adipose tissue, with epithelial structures, ducts and lobules, linked to the nutritional function.

Breast carcinomas are characterized by an excessive proliferation of epithelial cells confined to the ducts and lobules. This proliferation, in most cases, creates lesions which are detectable by mammography. Without this imaging technology, the chance of early detection is only 5%. In some rare cases, cancer cells are detached from each other, thus they do not cause lesions and the mammogram exam looks normal (Drake et al., 2009).

If not treated in an early stage, the disease often evolves into invasive carcinoma, by spreading to nearby tissues. From this point on, the prognostic becomes worse, with the tumor size and spreading radius being the main factors for prognostic.

Mammography is the most common screening method in asymptomatic woman. In this exam, the patient's breast is exposed to a short X-ray pulse which travels through the breast, and is then captured by a detector. Then, an image is produced based on the energy absorbed by each section. Adipose tissue is seen as dark (radio transparent) while epithelial tissue, including lesions, is normally white. The common identifiable changes in breast structure are divided in three groups: masses, microcalcifications and architectural distortions. All of these can be either benign or malignant, depending on multiple factors (Hela et al., 2013).

In the vast majority of cases, mammography exams consist of two views of the breast, one craniocaudal (CC) and one mediolateral oblique (MLO) with vertical and lateral orientations, respectively. Rarely, patients are screened using different views to cope with their characteristics (Radswiki, 2016). Fig. 1.1 shows the same breast watched from MLO and CC views. The red mark identifies the same mass.



(a) MLO view (b) CC view

Figure 1.1: Example of two views from the same exam, done to the left breast. Red regions denote the same mass. Image taken from CBIS-DDSM dataset (Lee et al., 2016).

1.3 Improvement of Diagnosis using CAD Systems

The interpretation of exams by specialists is far from ideal. Using standard mammography, 6-20 false positives are diagnosed for each true positive (Svahn et al., 2015). Each false diagnose leaves the patient anxious and requires further unnecessary exams (Brennan and Houssami, 2016). The percentage of exams with a positive result, which are then confirmed with breast cancer by another more precise method, is called recall. Systems with higher recalls would have an high social and economic impact by reducing the number of exams healthy patients are submitted to.

Also, false negatives have nefarious implications for patients, by delaying diagnosis, which, as seen, diminishes survival rates. Depending on the author, the estimation of cases can be very different. HD et al. (2016) argued this number is around 1-1.5 per thousand exams. Dromain et al. (2013) points to 10% to 15% of the cases. Either way, a screening system should be able to

approximate this number as closer to zero as possible, even if leads to poorer recall rates. The rate of breast cancers detected is called sensitivity.

To address these issues, the scientific community has proposed many algorithms that could, in conjunction with specialists, improve diagnosis by increasing accuracy in interpretation (Hela et al., 2013). Some of these computer-aided detection (CAD) methods currently being used, usually focus on finding microcalcifications and masses then analyzed by a specialist. It is still unknown whether the use of CAD methods yields better results than traditional diagnosis (Azavedo et al., 2012), as they return a high number of false positives, which diminishes specialists' confidence in the system.

The development of more robust algorithms, designed specifically to address the problem of low recall rate, could have a huge impact by reducing the huge workload of screening, increasing survival rates and decreasing health care costs.

1.4 Deep Learning

Deep learning is a group of algorithms that is able to model patterns directly from raw data. This concept is different from many other machine learning methods, which require features to be extracted first. As such, they can be used even in scenarios where we have very small field knowledge about the problem.

Although some works already used these algorithms in the 1990s, in the last few years we have seen very impressive results. Some authors point to the fact that, due to the existence of more available data, as well as computational power, these methods have been able to achieve human or super human performance in many fields. A recent well known example of this, is the Alpha Go a computer program that bested Lee Sedol, one of the best Go players in the world (DeepMind, 2017).

One of the main arguments, that algorithms like this could perform so well on hard tasks is the fact that, with enough computational power and data, these can receive much more training than specialists can in their whole lives.

Authors have explored deep learning in the context of automatic diagnosis. However, several problems arise when working with medical data. In most datasets, the amount of patients is too small, due to legal and ethical restrictions. Another common problem is the fact that, in the case of mammograms, images contain very subtle important details, which increase the difficulty of recognition tasks.

Researchers have tried to address these issues, in order to create robust CAD systems, using deep learning methods. If the results obtained in other fields could be replicated for medical imaging, we would see immediate applications, that would improve our health care systems, while reducing costs and specialists workload.

1.5 Objectives

The main goal of this work is to develop an automatic system for breast lesions detection in mammography images, based on a deep learning approach. For this, we focus on Convolutional Neural Networks, a particular type of deep models, especially well suited to deal with image data. Deep learning models have shown high performances in natural images. However, their application directly to medical datasets does not yield the same results, due to significant differences between natural and medical data.

To address these issues, we propose and study the impact of architectural changes and alternative learning strategies. Specifically, we try to solve the unbalance between healthy and unhealthy tissue in mammograms and the small number of medical images.

1.6 Contributions

In this work, we show Convolutional Neural Networks can be used to detect lesions in mammograms, in the context of breast cancer. We propose a system for this problem based on relatively small models, easy to train. These analyze a local region only, instead of the whole image, which allows to keep some detail information. The system is able to detect 80% of lesions at 2 false positives per image. Important aspects of this method include a small running time for new images and increased sensitivity towards malignant lesions.

Additionally, we study the impact of new methodologies to deal with the problems of low amount of data and unbalance between healthy and unhealthy tissue. First, the introduction of rotated filters in the network architecture as a regularization method is tested. This technique is inspired by the field knowledge that lesions do not have orientation. It works by reducing the number of optimizable parameters in a model. A very small gain in performance was obtained, for some configurations, which motivates additional experiments in different settings.

Second, a rank learning strategy was tested to deal with the low number of unhealthy regions in the dataset. This technique focuses on learning the difference between positive and negative cases. Similar performance to standard classification was obtained, with the disadvantage of requiring more GPU memory for optimization. We show the model fits training data better than validation in a very early stage of optimization. Although this method was not helpful as a regularization, it can serve as an alternative in unbalanced datasets.

Finally, a cascade approach was proposed. This consisted in learning two sequential models where the first trains on the initial data and the second only "sees" the previously classified positives by the first model. Essentially, at each stage a big number of negatives is discarded, while keeping as much positives as possible. This strategy was effective at increasing performance, when compared to a single model system.

All this experimental work was tested in the recent publicly available CBIS-DDSM dataset, and the results obtained can be used as a baseline performance for future methods tested in the same data.

1.7 Dissertation structure

This document is organized in the following way. The next chapter summarizes the past research and current state-of-the-art in the fields of breast cancer CAD systems and deep learning. In chapter 3 and 4, a formal description of Convolutional Neural Networks and other image processing methods is presented. Chapter 5 gives a formal description of the main contributions of this work. In chapter 6 the experiments performed are explained, results are presented and discussed. Finally, we conclude the main ideas about the work.

Chapter 2

Previous work

In this chapter, important previous work is exposed, divided in three sections. First, an introduction to conventional CAD systems is done. We do not attempt to do an exhaustive description of the methodology, but to characterize the variety of strategies used and present some of the most common. In the second section, we portray an historical perspective on Convolutional Neural Networks, the deep models used in this work. This allows a more robust understanding of what were the conditions that led to development of these algorithms and the challenges ahead. A more formal description of the methods is shown on the next chapter. Finally, we review other authors' work on deep learning methods applied to mammography.

2.1 Automatic Lesion Detection in Mammograms

Automatic image analysis of the mammograms started in the 1960s. One example is the work of Winsberg et al. (1967). Authors utilized simple methods to compare images from both breasts, with the objective of finding lesions. However, the minimal resources in terms of computational power, data and imaging techniques, limited the chances of success. Later, in 1990s, more complex approaches followed. Brzakovic et al. (1990) used detection methods, based on thresholding and fuzzy pyramid linking, to identify high intensity homogeneous regions in mammograms, that could be related to cancer. After this, Bayes classification was performed using area, shape and edge features, to discriminate between benign, malignant and non lesions. What is important about this early work is that the main steps of most CAD systems were already shown: 1) detection of suspicious regions; 2) characterization of each region and 3) a classification stage, which predicts for each region a probability of malignancy. Works like this paved the way for interesting developments in CAD systems, which is a globally increasing trend (Giger et al., 2008). In terms of nomenclature, frameworks like this are refereed as CADx, while CADe is used to describe algorithms which only perform detection.

As seen in the introduction, we are not sure about the extent to which false negatives are a problem. However, when these cases are analyzed retrospectively, signs of malignancy can be identified in the majority of missed cases. This has been attributed to many factors: 1) breast

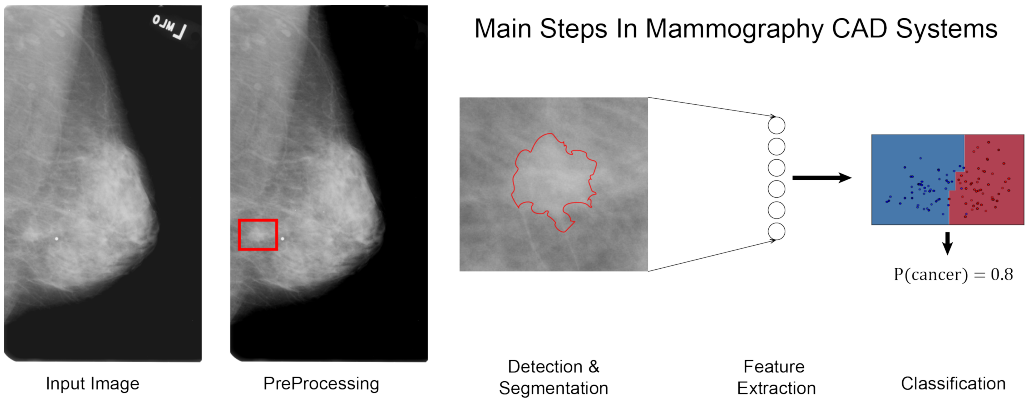


Figure 2.1: Main steps in mammography CAD systems.

complexity, 2) workload, 3) low probability of cancer and 4) subtleties in the image. All of these indicate that a computer assisting radiologists could decrease the number of false negatives, due to their consistent output for the same input and the ability to keep attention. Nowadays, the aim of developing CAD software is to offer objective evidence and increase radiologists confidence in the diagnosis (Giger et al., 2008).

In mammography, when it comes to digital image analysis there are two options: Screen-Film and Full-Field Digital Mammography (FFDM). In the first case, images are physically obtained from the X-ray machine, and need to be digitalized. This leads to lower signal-to-noise ratio, hindering algorithms' performance. Additionally, it is more costly and less practical, as results are not immediately obtained. With the introduction of FFDM, CAD systems can perform better and output a result immediately on the screen (Dromain et al., 2013).

There are many image analysis techniques used in mammography. However, most CAD system usually carry out the same steps. These are preprocessing, detection, segmentation, characterization and classification. Fig. 2.1 shows a diagram which illustrates this. As such, in this section we will individually address each of these.

2.1.1 Preprocessing

Preprocessing, aims at enhancing certain structures in images, to increase the chance of success of following algorithms. Common tasks at this stage include a segmentation of the breast and pectoral muscle, contrast enhancement or image denoising.

Breast segmentation is a simple task. Thresholding and morphological operations usually perform well. This allows for the following algorithms to look for lesions only inside the breast. It also helps to remove some label artifacts found on screen-film mammography (Pereira et al., 2014; Armen Sahakyan, 2000). Morphological operations consist in non-linear filtering of image regions and are explained in detail in chapter 4. Other methodologies exist with good results also (Kus and Karagoz, 2012).

A harder task is to obtain the pectoral muscle region. Normally, this region in the breast has a high intensity and so its boundary can easily be mistaken with lesions. Some works use this

information to design intensity based approaches. Saltanat et al. (2010) first performs morphological operations, to reduce image detail, and then maps pixel intensity into an exponential scale. A threshold is selected using an iterative algorithm. After this, regions brighter than this parameter are considered pectoral muscle. The muscle's boundary with the breast can be approximated well by a straight line, and so, other authors, like Yam et al. (2001), base their approach on this. Through linear filtering, they compute the image's gradient. After this, a Hough transform is used. This operation maps each point with coordinates x, y to a sinusoidal curve in the new space r, θ. Sinusoids intercept at r, θ if the points that originated those curves, can be joined by a straight line with distance r to the origin and slope perpendicular to θ. Therefore we can deduce the position of straight lines, in the original image, by selecting local maximum values in the Hough space. Authors used this to obtain the pectoral muscle boundary with the breast. Similar approaches were followed by other authors as well (Kwok et al., 2004). Cardoso et al. (2010) also compute the gradient to assist in the pectoral detection. However, instead of detecting a straight line, authors start by mapping the image to polar coordinates. Then a graph is defined, where each pixel is a node connected to its neighbors by arcs. Each arc has a weight based on the gradient in that region. Segmentation is obtained by searching for the shortest path in that graph and transforming the image back into cartesian coordinates. This segmentation task has been reviewed by Ganesan et al. (2013). He concludes that no particular method seems to work perfectly for all datasets.

Contrast enhancement is also common, in the early stages of CAD systems. The idea is to increase detail in the mammogram. Histogram equalization is a well known method for this. The main idea is to map pixel intensities to a uniform distribution. As such, most frequent pixel values will be spread out into a bigger range, effectively increasing contrast in those locations. An extension of this method is CLAHE (Pizer et al., 1987), used very frequently in medical images. Instead of adjusting pixel values globally, local histograms are computed in a square grid. For a point, equalization is done similarly to the previous method, but interpolating between the four closest histograms, instead of one global pixel distribution. Additionally, to avoid increasing noise in relatively homogeneous regions, the algorithm establishes a maximum number of pixels that can have the same intensity and redistributes those uniformly, for each local histogram.

For image denoising, some authors use linear or non-linear filtering operations. For instance, Anitha et al. (2017) used a median filter of size 3×3 to remove digitalization noise from mammograms. This works by screening all image pixels, and assigning each the median value in a 3×3 neighborhood. Another very common method for contrast enhancement and denoising in image processing applications is the discrete wavelet transform. This method decomposes an input signal in two: 1) approximation, which keeps low frequency components and 2) detail, which keeps the higher frequencies. To obtain these two signals, a wavelet function, with special properties, is used. This allows following algorithms to perform operations on these components separately. The input image can be reconstructed after processing the approximation and detail. Gorgel et al. (2010) used this method to apply different enhancement algorithms to each sub image. For the approximation coefficients, authors performed homomorphic filtering to emphasize low contrast suspicious regions. For the detail, author modeled the noise statistically as a normal distribution,

while important edges were considered to be distributed according to a Laplacian probability distribution. Using Bayes theorem, authors reduce the values of the coefficients, according to their probability of being noise. After enhancement the image is reconstructed.

2.1.2 Detection and Segmentation

After image enhancement authors focus on detection and segmentation of lesions. Although, a CADe system only needs to point lesion locations to the specialist, information about the contour can be used to discard false positives or perform pre-classification. In the case of automatic diagnosis systems, limiting the region of interest is vital for a good characterization.

Region growing strategies are used by some authors for this task. The main idea is to start with a seed point that will grow to neighboring pixels, if these are sufficiently similar. When growing stops, the final region is considered a mass. In the case of Yousuf and Mohammed (2013), a specialist selects the seed in the center of a suspected lesion. Regions around this pixel are aggregated, if they have similar intensity and local contrast. Although in this case a specialist is required, some methods automatically detect these points. For instance, a difference of Gaussian filter could be used to this end (Catarious et al., 2006). As the name says, the coefficients of this operator are given by the subtraction of two Gaussian functions centered in the same point but with different standard deviations. This filter highlights bright, circular regions and so, it can be used to detect masses. One drawback is the fact that it has a fixed radius, while mammographic lesions vary in size. This can be addressed by analyzing the image at different resolutions.

Another common strategy for segmentation in medical images is active contours. For instance, Yuan et al. (2007) used this strategy to obtain lesion segmentation in mammograms. The proposed model works, by minimizing an energy function, which measures the intensity and homogeneity of inside and outside regions of an initial imperfect segmentation.

Some authors have proposed feature extraction followed by classification for lesion detection. In this case, to determine if a region belongs to a lesion or not, a numerical characterization is done, based on selected feature extractors, followed by a classifier. In some cases this methodology is more robust, eliminating false positives. In (Karssemeijer and te Brake, 1996) authors utilize the fact that malignant lesions often present spiculation patterns at their edges, to design two numerical features. These are then combined using classifiers to obtain the "suspiciousness" level of each point.

These are only some of the methods that could be used to this end. Authors usually focus on particular geometrical, intensity or edge features to design algorithms able to capture that information. After this, an heuristic algorithm is designed to translate that information to a detection and segmentation, or, in some cases, classification is done using machine learning methods.

2.1.3 Feature extraction

With segmented regions, authors have explored a wide variety of feature descriptors to characterize lesions. These consist in numerical values that quantify a certain property of an object or

region in an image. Sometimes that property is too abstract to have a direct "meaning". The most basic ones consist of area, perimeter and mean intensity. Some authors try to capture signs of malignancy, designing handcrafted features that resemble the medical description of what is considered malignant. As seen before this is the case of Karssemeijer and te Brake (1996), who tried to capture spiculation patterns. Contrast features also fall into this category, as often radiologists mention a well separated lesion, from the rest of the tissue, is a sign of malignancy (Kooi et al., 2017b). te Brake et al. (2000) used contrast and intensity features to capture this information. These can be measured in a variety of ways, but commonly mean intensities are computed for the inside and outside region. The contrast is given by subtracting these values. Geometrical and edge features are also important. The ones proposed by Peura and Iivarinen (1997), which include eccentricity and compactness are often used.

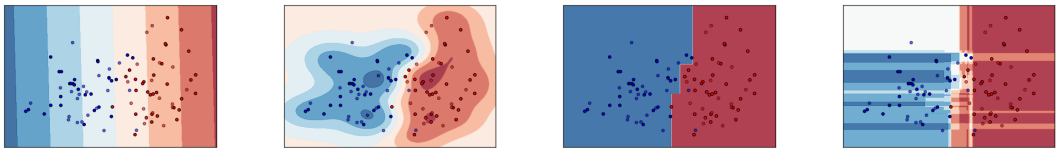
Another common approach is to use texture to capture information. This property of some images implies variation at lower scales and repetition at higher. Although texture is generally hard to describe by specialists, malignant and benign masses have different patterns. Further, these methods work well in many applications, capturing a wide range of information. One of the most used texture feature extraction methods is the gray-level co-occurrence matrix (GLCM). Each element of the matrix is obtained by measuring the number of times elements separated by an horizontal and vertical offset appear with fixed intensities. After this, features like mean, contrast, energy and variance can be computed using matrix coefficients. This is effectively a statistical approach to texture. Another common example are local binary patterns (LBP). Each of these encode a specific intensity relation of a point and its neighbors. After counting, we can build an histogram and use each bin as a numerical feature. Some works have compared multiple texture methods in this particular application. (Arevalo et al., 2015; Kim et al., 2013)

2.1.4 Classification

After obtaining a good feature description of each region, machine learning methods are often used to model what patterns in data are related to malignancy. Normally, this is done by using classification algorithms, which work by optimizing one predefined criterion based on a training algorithm and the available data. Here two of the most common will be explained, Support Vector Machines (SVM) and Random Forests (RF).

The concept of feature space is important for the following paragraphs. This consists in a multi dimentional space, where each numerical feature is an axis. As such, the characterization of a lesion can now be considered a point in this space. Classifiers work by defining boundaries, that separate samples from different classes, in this case malignant and benign. As such, after training one of these models, we obtain two separate space regions. For a new lesion, we first characterize it, using the methods above, and then verify in which region that data point lies.

The SVM (Boser et al., 1992; Cortes and Vapnik, 1995) is a very well known binary classifier. It separates the feature space trough an hyperplane. This boundary is obtained by maximizing the margin between the most difficult points of each class, the ones that lay near this border. These points are called support vectors. The hyperplane is completely defined by a vector with



(a) Linear SVM (b) RBF SVM (c) Decision Tree (d) Random Forest

Figure 2.2: Examples of classifiers and how they divide the feature space. Images were obtained using sk-learn (scikit-learn.org, 2017).

its orientation and the distance to the origin. These parameters are optimized using quadratic programming algorithms or by gradient descent based methods. One interesting property of SVMs is that they can be applied to non-linear data by using the kernel trick. This yields more interesting divisions of the feature space, increasing accuracy in non linearly separable datasets. Many works have used SVM for mass classification (Fusco et al., 2016; Kim et al., 2013; Lesniak et al., 2011).

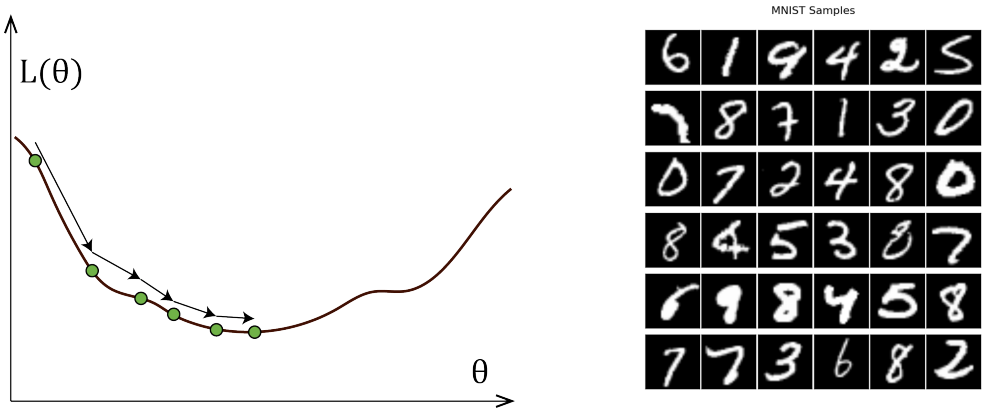
RF (Ho, 1998) is another classifier frequently used, based on simpler units called decision trees. Each of these, divides the space iteratively with binary decisions based on one feature only. To train these algorithms, binary decisions are learned one at a time by maximizing one criterion, usually the gini index or information gain. The combination and averaging of many decision trees constitutes a RF classifier, used commonly in literature (Vibha et al., 2006).

Representations of how these methods separate the feature space can be found in fig. 2.2.

2.2 Convolutional Neural Networks in Image Recognition

As previously seen, much of the progress done in automatic medical image analysis is based on the development of feature extraction procedures, either by carefully designing algorithms that resemble the way specialist diagnose, or by utilizing typical computer vision features, like texture descriptors. An alternative way to design diagnosis systems is based on representation learning, a field in machine learning which deals with raw data directly. The key idea is to learn a feature representation that is useful to the task at hand, directly from data. This approach requires significantly more training examples, but has two major advantages: solutions are not specific to an application, and the algorithms are not biased by the "human way" of diagnosing. Deep learning methods work by learning sequential representations of data, using neural networks. Essentially, at the first level, inputs are mapped into a new feature description. The second level utilizes this description to compute a new one. This process is repeated until the last level is reached. With this, a sequence of functions is generated which can extract increasingly abstract features, and maps data to output. These functions depend on parameters, and, due to this, we can find specific values for these variables, that are good solutions for a particular problem (Lecun et al., 2015b).

The versatile nature of these methods allows them to model many different types of problems. For instance, the same algorithm can be used to distinguish between malignant and benign masses



(a) Diagram of Gradient Descent Algorithm

(b) Examples of MNIST dataset

Figure 2.3

in mammography images or handwritten digits, depending on the raw data fed to it. Normally, these functions are called layers which are stacked to form a deep learning model.

The process of discovering a good set of parameters is called training and is usually done using gradient descent algorithms. Conceptually, this is a minimization problem, where we first define a loss function based on the difference between the model's output and the value we expect it to predict. Then, for the training data we compute the gradient of that loss with respect to the parameters, and update these in the opposite direction. This is done iteratively until a good solution is obtained. Fig. 2.3a shows a diagram of how the parameters (θ) and loss function ($L(\theta)$) evolve throughout this process.

One very important discovery for the development of multi-layer models was that the parameters can be optimized using gradient descent, as long as each module is a relatively smooth function of its inputs and parameters (Lecun et al., 2015b). Specifically, Rumelhart et al. (1986) showed that gradients for each layer could be computed by backpropagation, a process based on the chain rule of derivatives, later explained in chapter 3. Depending on the concrete algorithm, gradients can be computed on the whole data, on subset of examples or on a single example (Lecun et al., 1998b).

Generally, there are two types of learning, supervised and unsupervised. While, in the former, data has associated labels, which the model is trained to output, in unsupervised learning only unlabeled data is used. Broadly, while the first forces models to learn label discriminative features, the second favors complete representations of the input data. Sometimes, unsupervised learning is used to obtain a good set of initial values for parameters, then used at the start of supervised learning. This process of using a defined, instead of random, initialization of parameters is called fine-tunning.

Deep learning has achieved incredible success in many different tasks, such as, speech recognition (Mikolov et al., 2011), image classification or language translation (Jean et al., 2014). As pointed out by Lecun et al. (2015b), deep learning methods have thrived in traditionally difficult problems for the artificial intelligence community. A particular type of these methods is Convolu-

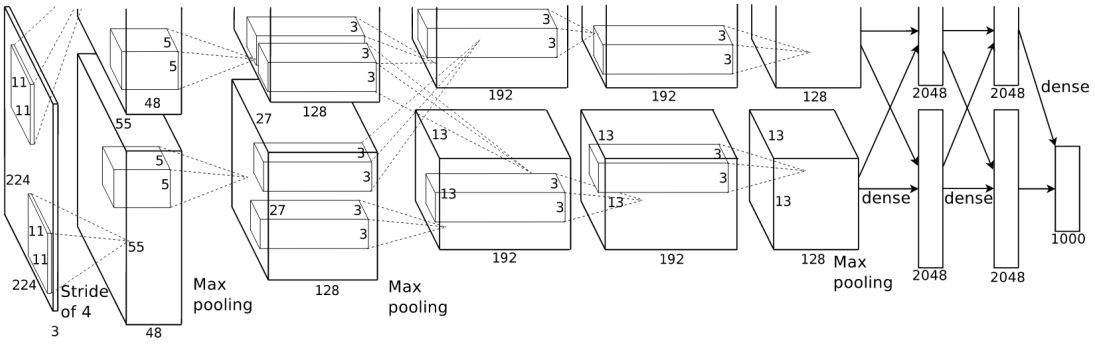


Figure 2.4: One of the first major architectures, AlexNet. Taken from Krizhevsky et al. (2012).

tional Neural Networks (ConvNets), which are commonly used for image and video recognition, and explained in detail in section 3.

Briefly, these can be defined as computational neural networks with special constraints, such as heavy parameter restrictions and organized spatial layers, which make models easier to optimize and also increase their ability to generalize to new data. Importantly, these constraints do not affect its representation ability, when learning compositional hierarchies, which are present in many natural signals (Lecun et al., 2015b). Due to the above mentioned properties, the early layers effectively compute convolutions between the input signal and a set of learned filters, and thus, the name Convolutional Neural Networks.

2.2.1 Early Works

The early works that laid ground for ConvNets were inspired in the visual nervous system model proposed by Hubel and Wiesel (1977). In particular, the neocognitron, designed by Fukushima (1980), uses many biological concepts to explain the operations performed by this network. In this work, the spatial organization properties of ConvNets were already being used. Notably, much of our understanding and discussion in this field is still based on analogies with the human visual system.

Although some authors like Zhang et al. (1994); Vaillant (1994) had already published results using these models before, the standard first reference to ConvNets is normally Lecun et al. (1998a). In this paper, the LeNet-5 model is proposed in the context of handwritten digit recognition on MNIST dataset (LeCun et al., 1998) (Fig.2.3b). Its deep learning architecture contained two convolutional layers with a number of filters which, compared to today's models, is extremely small. Many of concepts discussed in this paper are still relevant today. In particular, Lecun et al. (1998a) identified the high amount of data necessary for deep models, when compared to traditional algorithms. To address this, he introduced distortions in the training set, artificially creating more data. Although the artificial examples are not ideal, in the sense that they are correlated with the original, this data augmentation strategy proved efficient. The authors also point out the advantages of these methods regarding their non-specificity and little field knowledge required.

They obtained a final test-error of 0.7% by combining the prediction of 3 of these models, optimized on different data, using a boosting strategy. This consisted in selecting which images to use for training each model, depending on how well the previously optimized models performed on them. Almost 20 years later, the current state-of-the-art on the same dataset is 0.21% achieved with modern regularization algorithms and ConvNets with much more capacity (Lian et al., 2016).

2.2.2 Successful Approaches in the 2000s

During the first decade of the 21st century, ConvNets were still far away from the mainstream position they occupy today in the computer vision community. Still, many important results were obtained by adapting and applying these methods to specific problems. For instance, Garcia and Delakis (2002) proposed a system for face detection based on a small sized ConvNet. They collected 2146 highly variable face areas from multiple sources, on the Internet. Then, they performed data augmentation, by rotating images within the range of $[-30^{o}, +30^{o}]$, and by reducing contrast. Note that, the type of distortions introduced should resemble the variations in unseen data. Otherwise, training becomes more difficult, without increased generalization. After the learning process was completed, new images were tested, to measure how well the model performed on unseen data. To do this, they computed the output of all possible fixed size areas in an image, at multiple scales. All zones classified as positive were considered final face locations. Importantly, they do this in an efficient way, by computing the output of each layer for the whole image, instead of one location at a time. This allowed their system to screen new images at a rate of 4 frames per second.

Another example of a successful application of ConvNets was proposed by Ning et al. (2005). They employed a model with three convolutional layers, integrated in a bigger framework used for segmentation of cells and nuclei of developing embryos, in microscopic videos. In this particular problem, the development of a loss function related to segmentation accuracy, that could be easily minimized was not trivial. Instead, they used the ConvNet for initial pixel-wise supervised classification, to distinguish between five categories: cell wall, cytoplasm, nucleus membrane, nucleus and outside medium, based on the surrounding region. For a new image, the network is applied to all possible regions, resulting in a label map, where each point is classified as one of the five categories. Then, à priori knowledge of the spatial relation between cellular structures is used, to refine that classification. Finally, this result is matched to an elastic model. The model took 18h to train in two CPUs, showing how computationally expensive these methods can be. One of the important aspects of this work is the integration of deep learning algorithms with à priori knowledge about the problem to enhance results.

Similar to Ning et al. (2005), Osadchy et al. (2007) adapt the problem of typical supervised classification to their specific application. But instead of coupling it with other methods, authors change the model and loss function to address face detection and incorporate it with pose estimation. As previously seen, each layer maps an input to a new feature description. These descriptions can be thought of as a point in a multidimensional space, where each axis is a feature. Following this idea, they first define a feature space for pose, based on the pitch, yaw and row angles of a persons, face in relation to the camera. For instance, the yaw can take values in the range

$[-90^o, +90^o]$. Authors map these values into a 3 dimensional space, where valid yaws lie on a semi circle. The combination of the three angles gives a 9 dimensional space, where valid poses lie on a specific region. Then, through the use of a carefully designed loss function, they perform supervised classification, mapping images with faces to that space, in the specific point of the respective pose. Images without a face are mapped to points, away from the region of valid poses. Their results showed that the integration of pose and face information yielded better results for the two tasks than supervised classification for each individually. Again, the computationally expensive aspect of the algorithms is shown, by the 26h training process in one CPU. However, after this, the model can run relatively quickly, at 5 frames per second.

Differently, Turaga et al. (2010) used convolutional neural networks for 3D images segmentation in volumetric electron microscopy. Their model is used to compute affinity graphs, where each node has an affinity value to each of its neighbors. This is done in a supervised manner, by labelling affinity of 1 to adjacent pixels of the same region, and 0 for different regions. Similarly to the previous work, the model does not output a label for each image. Instead, it outputs a set of 3D affinity graphs, one for each dimension, x, y, z. These are then partitioned using other algorithms to obtain a good segmentation. A particularly interesting aspect of this work was that the model used is very unusual compared to the ones seen previously. First it did not contain any pooling layers. These operations are used to reduce the spatial resolution of the feature maps from one layer to next, which provide interesting properties to models. They also avoided using dense layers, so that the model kept its spatial structure. These concepts will be explained later, in chapter 3. Essentially, instead of learning only new parameters, to suit this particular problem, authors introduced drastic changes in the layers organization, often called the architecture of the model. Through these, the algorithm becomes more specific to the problem of segmentation, based on local information only.

2.2.3 Advances in Optimization

Until now, all described models had a relatively small number of layers and took hours or even days to train. This was an impediment to the application of these algorithms to new contexts, which required bigger input images or had huge amounts of raw data available. In parallel, interesting advances were being made in terms of better hardware and training algorithms. Particularly, some studies at that time were developing new ways to compute machine learning optimization methods, by taking advantage of the fact that computation could be parallelized using general purpose graphical processing units (GPUs).

Steinkraus et al. (2005) introduced this concept by speeding up the training and inference processes of a two layer neural network. Authors discuss that developing and using machine learning specific hardware can be costly. Additionally, users could lack support for these processors and they could become obsolete after a few years. They show that GPU can be a good solution when training machine learning algorithms due to its parallelization capabilities. Further, these units have become increasingly more potent and available in the market. In particular, neural networks

are very well suited for GPU implementation due to low memory access requirements. The memory access still plays a huge role in today's hardware. For ConvNets all parameters, data and intermediate layer results for one training iteration, need to be stored in the GPU for fast implementations. As such, for bigger models, higher capacity processors are needed. In many cases, authors need to make compromises in terms of model architecture to cope with hardware limitations.

At the same time, better algorithms were developed to make the training process of deep models more tractable. Hinton et al. (2006) proposed a greedy learning method for deep belief networks, another type of deep model, which learns a probabilistic representation of the inputs. Authors trained the model in an unsupervised way, one layer at a time. This was done by restricting non-trained parameters to have the same value as the ones being trained. After one layer finished trained, they froze its parameters and performed the same operation on the next layer. After completing unsupervised training, they show the model could be used for classification, by training a classifier on top, in a supervised way. They achieve an error of 1.25% on MNIST dataset. Bengio et al. (2007) proposed another unsupervised learning strategy based on stacking auto-encoders. Instead of tying up the parameters, authors train the first layer to map the input to a new feature space and reconstruct it again. For the second the same is done, but using as input the first layer representation. This is done iteratively, until the network is finished. This knowledge was extended to convolutional models by Masci et al. (2011), which obtained better results with this strategy than by training in a traditional supervised manner. On MNIST particularly, they were able to lower the test-error to 0.71%.

One of the first fast GPU implementations of convolutional neural networks was done by Cireşan et al. (2011). They trained a model in a purely supervised way, achieving the best results obtained until date in three object recognition datasets. In particular they improved on MNIST decreasing the error rate to 0.35%. This proved the high performance of deep ConvNets when compared to hand crafted features, provided that enough data and computational power is available. Further, on this dataset, results showed that a better performance could be obtained using bigger models both in the number of layers (deeper), and the number of parameters in each layer (wider). Their training time was 10 to 60 times smaller on a GPU implementation when compared to the CPU's.

The base work that enabled the fast development of the last six years was done in 2011. From this point on, the use of deep learning algorithms exploded, becoming a major topic of research with many potential applications in the real world.

2.2.4 ImageNet Large Scale Visual Recognition Challenge

Perhaps the most well known dataset for benchmarking models in image recognition tasks, ImageNet (Deng et al., 2009) played a huge role and still occupies a privileged position when it comes to object classification and detection. Currently, it contains over 14 million images labelled in more than 20 thousand classes, organized in a tree like structure. The huge labelling effort was crucial to the revolution that took place in the deep learning field. Organized once a year, the

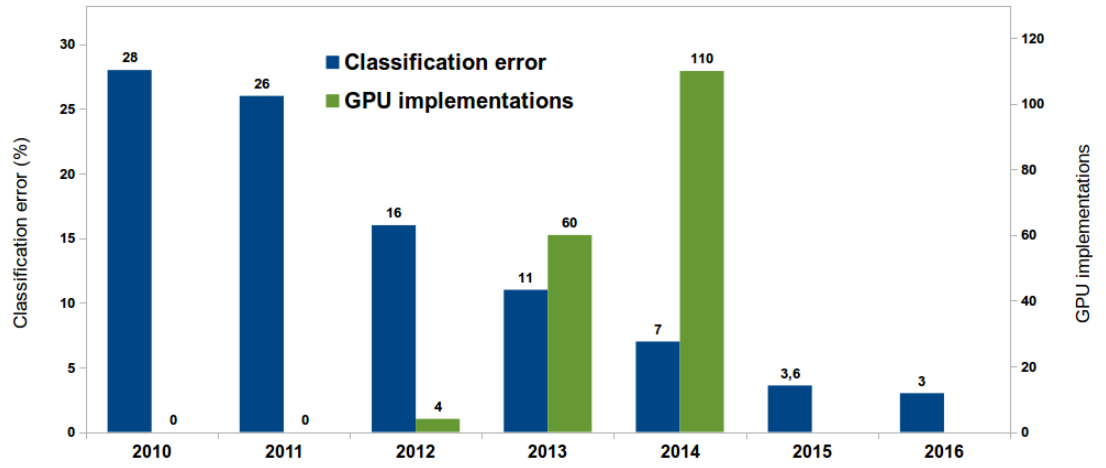


Figure 2.5: ILSVRC challenge over the years. In blue we can see the test error of the best entry of each year. At green are the number of entries using GPU implementations (no data for 2015 and 2016). (NVIDIA, 2015)

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2015) was the perfect stage for determining which architectures and training strategies performed best. In Fig. 2.5 we can see the increasingly better results in the challenge, as well as the improvement brought by the use of ConvNets. In the following paragraphs an overview of this development in the context of ILSVRC is presented. Some of the concepts discussed here will be explained in detail in chapter 3.

In 2012, Krizhevsky et al. (2012) proposed AlexNet, a deep convolutional neural network. Their performance was astonishing, winning the competition by a large margin. This work made deep learning methods a strong reference in the computer vision community, becoming one of if not the most influential paper in the field. Although the proposed model's architecture was relatively simple when compared to modern state of the art large scale models, authors utilized many important methods which are still relevant today. Specifically, they emphasize the use ReLU, a non-linear function, that, when incorporated in the model's architecture, increases training speed. They also used dropout layers, with the objective of increasing the model's generalization ability. The proposed architecture was unusual, as it had two separate streams of information, which allowed the use of a fast implementation, based on two GPUs. Importantly, and although a high number of examples is present in ImageNet, they augmented the dataset by cropping different image regions and flipping them horizontally. In the context of the competition, each model predicts five classes for each image, and if one of them is right, that classification is counted as correct. By this metric, they achieve an error of 15.3%, about 10% lower than the competition.

In the following year, Zeiler and Fergus (2013) were the winners, using a very similar model, with minor modifications. Specifically, they noticed AlexNet was discarding too much information in the first layers, due to aggressive pooling operations. As seen previously, these operations reduce the spatial resolution from one layer to the next. The factor of that reduction is given by the stride parameter, which is predefined in the model architecture. Zeiler and Fergus (2013)

reduced that value while also increasing the number of filters in each convolutional layer, making the network wider. Their implementation used one GPU only and achieved a score of 11.2%.

In 2014, the first place was achieved by Szegedy et al. (2015), which proposed the Inception model. This work introduced many new ideas in the deep learning field. The model contained near a hundred layers stacked in an unusual way. Authors propose the inception module, which divides the input in 4 parallel streams of information, each subjected to different operations, and integrates them after. These units were stacked on top of each other, to reduce the top-5 error on the competition to 6.7%.

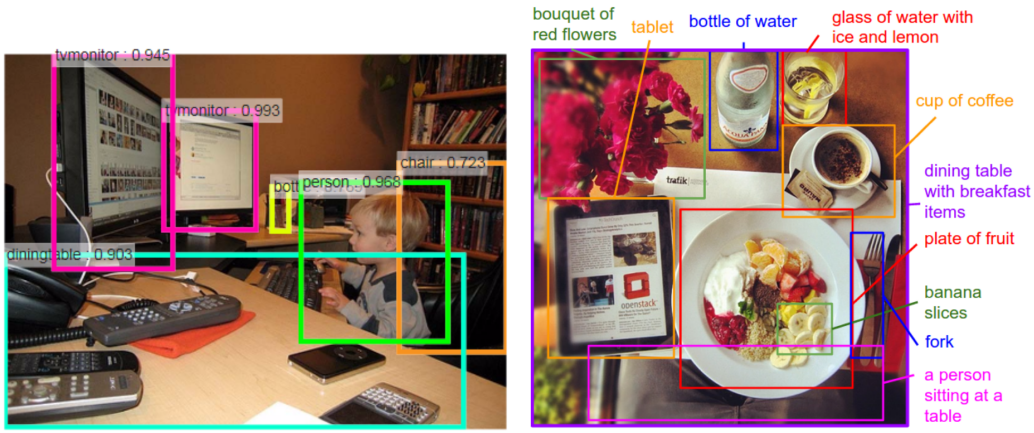
Some key ideas to the success of this architecture were the emphasis on convolutional layers, the introduction of 1×1 filters to reduce computational requirements, and the introduction of classifiers in the middle of the network to assist optimization. Although the Inception model was extremely more complex than AlexNet, and required more memory, the number of optimizable parameters was $12\times$ smaller. This, as discussed later in this document, as to do with the adequate use of convolutional layers, which have much less parameters than dense ones. Training took a week using a few high-end GPUs.

Using some ideas of the previous year, ResNet (He et al., 2015) was the best model in 2015. Similar to Szegedy et al. (2015), authors proposed a new module, the residual block, and stacked many of these to form the model. Different to conventional architectures, where a feature representation is used by the following layer, the residual block sums this representation to the input of the layer. The key difference is that layers are no longer representing the input, but introducing a small change to it. Authors hypothesized that learning these functions was easier than conventional feature representation. Each residual block is composed of two convolutional layers separated by a non-linear ReLU function. The best performing architecture had 152 layers stacked on top of each other, with no parallelization. The final test-error rate was 3.6% surpassing for the first time the human level accuracy (around 5%). In 2016 the competition winners were CUImage, which utilized a model based on ResNet. Although no publication is available, their good results come from combining multiple models and utilizing context information for classification. They obtained a final score of 3.0%.

2.2.5 Recent Influential Works

The rapid development of deep learning is normally attributed to three main factors. First, the discovery of more efficient training methods was fundamental. In the case of convolutional neural networks, ReLU layers, batch normalization and dropout increased training speed and performance. Second, the availability of more processing power allowed bigger models to be optimized efficiently, through the use of GPUs. Finally, the availability of high amounts of labeled data, played a key role in this development. The history of the ILSVRC reflects just that.

In recent years, many works increased the versatility and understanding of deep convolutional models. For example, the introduction of R-CNN (Girshick et al., 2013), Fast R-CNN (Girshick, 2015) and Faster R-CNN (Ren et al., 2015), are all based on the same idea of incorporating a region selection scheme into a convolutional neural network. In the first work, authors use selective



(a) Faster R-CNN (Ren et al., 2015) (b) Image to Text (Karpathy and Fei-Fei, 2017)

Figure 2.6: Impressive results using deep learning models

search method to propose regions, then analysed by a ConvNet that outputed a feature description of each. A SVM classifier is trained on top. The whole method had the advantage of not only classifying which objects were in the image but also to locate them. (Girshick, 2015) build again, on top of this concept, to increase the speed of the overall process. The author integrated all steps into the same ConvNet model, using two outputs, the first which learned to locate regions of interest and the second which classified those regions. This was done by adapting the loss function to have two terms, one for each task. (Ren et al., 2015) improved further, by using internal feature maps to obtain regions of interest, instead of learning them as one output. This is done with a smaller network, that will look to all points in this feature map and generate for each a set of proposed regions along with their "objectiveness" score. After this, a RoI Warping Layer (Dai et al., 2015) is used to extract that region, and feed it to a classifier. What is fascinating about these works is the ability to incorporate attention in ConvNets, essentially integrating classification and location in the same task.

The development of region proposal ConvNets allowed the innovative work of Karpathy and Fei-Fei (2017). Authors, based on weakly labeled data, were able to generate text descriptions to each region of an input image. The term weakly labeled data refers to the fact that, for training examples, information was incomplete. Authors only had access to a description of the whole image and not for each specific region. They did this by mixing ConvNets and Recurrent Neural Network, a different type of deep model, especially suited to deal with sequential inputs. Importantly, authors were able to associate image and text data in the same feature space.

Another important work in this field was the use of generative adversarial networks (Goodfellow et al., 2014), which work by training two models at the same time, the first generative and the second discriminative. For this, a training dataset is required. The objective of the generative model is to generate samples that look like original ones. The second model, learns whether a sample is original or generated by the first model. This is effectively an unsupervised learning strategy. By this interaction, we can obtain two models with different applications. For instance,

the discriminative model can then be used for supervised classification, while the generative has applications in terms of increasing image resolution or generating realistic images.

As seen, many ideas have been implemented with ConvNets and deep learning models in general. Due to the complex nature of these networks, some research has referred to them as black boxes. However, many works have shed light on how they operate. The work of (Zeiler and Fergus, 2013) did precisely that. Authors focused on attentive analysis of the features learned by the model using multiple tools. They propose many visualization techniques which greatly contributed to the intuition we have of ConvNets. One example is the use of a DeconvNet model to map feature representations back to the pixel space. This model computes an approximation of the inverse function the model learns, showing what patterns each layer captures. Another approach is to block some portions of the image during classification, to test what parts is the model utilizing for discrimination. Additionally, they visually compare the parameter values of their trained network against AlexNet, showing that smaller filter sizes and strides lead to more distinctive features. This last experiment links to the results obtained by Simonyan and Zisserman (2015) which surpassed the previous state-of-the-art models back in 2013 in ImageNet. They did this by designing very deep and simple models. Following the idea of Zeiler and Fergus (2013) their models are constituted by minimal filter sizes, increased number of layers and increasing number of filters for later layers. These two papers were of major importance during the early stages of the deep learning boom.

In summary, the high performance and potential applications of deep learning models was supported by many early works. Labelling efforts, fast hardware and the development of better algorithms were extremely important to this end. Many works achieved success by adapting these models, based on field knowledge, for each particular application. The following section talks about the use of deep learning in mammography image analysis and the strategies authors suggested to increase the performance of these methods in different conditions.

2.3 Convolutional Neural Networks in Mammography Image Analysis

In the 20th century, some authors had already used ConvNets for mammography image analysis. In the case of Sahiner et al. (1996), authors use a model with a single convolutional layer, to classify between masses and normal tissue in image patches, obtaining an AUC of 0.87. In their work, authors compare different input sizes to the model, number of filters in the convolutional layer and filter sizes. Curiously, they obtained better results by feeding the network texture images as input. These are characterized by assigning each pixel with a texture feature of the surrounding region. In the conclusion of their paper, authors identify the need to develop a fully automatic diagnosis scheme. Zhang et al. (1994) used similar models applied to microcalcification detection, obtaining an AUC of 0.91. They show the superiority of convolutional architectures for image classification against conventional neural networks. At that time, the low computational power was an impediment to the development of better models. These models were ignored for many years for automatic breast cancer diagnosis. Renewed interest came a few years after the success of AlexNet.

2.3.1 Transfer Learning

Medical images very often differ from natural images. However, as seen before, conventional computer vision feature extraction methods have seen success. The same applies to ConvNets, with some works suggesting these algorithms can improve over conventional techniques in medical image processing, including in mammography fields. Actually, some authors even use models pre-trained on natural images to perform mammography related tasks, proving the adequability of ConvNets for medical images.

This methodology, called transfer learning, is based on the idea that, to solve a particular problem one can extract information from similar tasks. It is particularly useful in scenarios where the amount of data available is not enough to train a good representative model. For instance, Huynh et al. (2016) used the ImageNet pre-trained AlexNet architecture to extract lower dimensional representations of image patches centered in breast masses. Their objective was to distinguish between malignant and benign masses, in a private dataset. They contrasted AlexNet representation with traditional approaches by using size, shape, intensity and margin features. For a fair comparison, both approaches were used to train an SVM classifier achieving the same AUC value of 0.81. Interestingly, the combination of the two methods improved results to 0.85, suggesting that the representations did not hold exactly the same information. This phenomenon has been observed by other authors (Kooi et al., 2017b).

Yosinski et al. (2014) discussed transfer learning in ConvNets, concluding that the first layers of the network are often more general, while later layers are problem specific. Due to this, we can derive multiple strategies for transfer learning, for instance, fine-tuning all layers, fine-tuning later portions of the model or use the model as is to obtain feature representations, then fed to a different classifier. In the case of Huynh et al. (2016), this last strategy was used. Authors

verified that, for early and later stages of the pre-trained model, information is not useful for the classification. In early layers, feature representations have high dimensionality and are similar to the input data. As such, non representation learning algorithms perform worse. Further, the use of high dimensional data requires more training time and labeled examples. For later layers, more compact representations can be obtained, but, in the case of pre-trained models, this can already be too specific to the original problem. As such, the best for classification was an intermediate layer.

In a similar fashion, Carneiro et al. (2015) utilized ImageNet pre-trained model (Chatfield et al., 2014) to distinguish mammograms with and without cancer. Differently to Huynh et al. (2016), instead of using the network as is, authors fine-tuned the parameters to this dataset. Carneiro et al. (2015) showed this methodology was more effective when compared to learning "from scratch". These results have been obtained by other researchers as well. However, their method utilized ground truth locations for lesions, and so, cannot be considered fully automatic diagnosis. By combining both views of each breast as well as mask images authors obtained an AUC of 0.91 on INBreast database (Moreira et al., 2012) and 0.97 on the DDSM dataset. In the context of discriminating mass lesions from normal tissue, Lévy and Jain (2016) reported much better results when initializing AlexNet with pre-trained parameters, instead of randomly. Their best result was obtained by a fine-tuned Inception model, with a validation accuracy of 0.929.

One disadvantage of fine-tuning ImageNet pre-trained models is the fact that natural images, normally in the RGB color space, are composed of three color channels. For mammography images, authors usually stack the same image three times, to meet the input requirements of such models. However, the model filters are still optimized to extract color features. Samala et al. (2016) utilized a transfer learning strategy for false positive reduction in digital breast tomosynthesis, but using mammography images for pre-training. These two imaging technologies are similar in the sense that X-rays are used. However, a 3D representation of the breast is obtained in tomosynthesis. Authors collected and used a mammography dataset to train a ConvNet, later fine-tuned to the main task. In this case, there is a high affinity between the classification tasks where knowledge is being transfered, with both images having the same number of classes, number of input channels and purpose. This is shown by the obtained validation AUC of 0.80 in tomosynthesis patches using only the pre-trained model. After fine-tuning a final AUC of 0.92 was obtained. Likewise, the work of Kooi et al. (2017c) utilized two mammography datasets. The first was used to train an ensemble of simple classifiers that will generate true and false positive regions. Then, a ConvNet is trained in these examples, in a supervised way. The model is used as a feature extraction method for the second dataset, obtaining an AUC of 0.786, when discriminating benign cysts from masses.

The performance of pre-trained models for feature representation versus fine-tunning is heavily dependent on the problem and available data. As a rule of thumb, Yosinski et al. (2014) discussed that the first should be preferred in scenarios where there is less data available or the pre-learning and final problems are very distinct. The fact that knowledge from natural scenes can be used in much different problems, like X-ray images, suggests that there should be at least a

small degree of similarity between ConvNets trained for very different problems. In fact, as noted by Yosinski et al. (2014), the first layer filters of a ConvNet, after training, tend to resemble either Gabor features or color blobs. Additionally, ImageNet trained models, used in most studies, learn to discriminate between a high number of very different classes and thus have a very rich feature representation, applicable in most scenarios.

2.3.2 Breast Lesions Classification

In opposition to transfer learning, some authors train ConvNets from "scratch" in mammography datasets. However there are several considerations that need to be accounted for. These come from three significant differences between big image recognition datasets, like ImageNet or MNIST, and medical ones: 1) normally, much less medical data is available, 2) image resolution is significantly higher and 3) much of the information required for diagnosis is in details. Due to this, a brute force approach could fail for lack of data and computational power, while simple rescaling the image to lower sizes can remove too much relevant information.

Many authors have tested the performance of ConvNets in classification related to breast cancer diagnosis. To this end, normally, only a relevant part of the image is considered. Using expert knowledge to select regions of interest, authors crop image patches centered in relevant portions of the exam. With this, detail information is kept, while significantly reducing the image size. This is the case of Lévy and Jain (2016), Arevalo et al. (2015) and Dhungel et al. (2017) where models are trained in a supervised manner, with the objective of distinguishing malignant and benign masses. In the first case a poor accuracy (0.66) was obtained, when training from scratch an AlexNet inspired model. However, an important finding was that healthy tissue around lesions is relevant for accurate diagnosis. The dataset contained 1820 mammograms, from which images from masses and normal tissue were cropped. In the second case, authors attempt the same task with 736 film mammograms of the BCDR dataset (López et al., 2012) and a custom architecture. Image contrast was normalized globally and locally. After training the network is used as a feature extraction method to train an SVM classifier. They obtained an AUC of 0.860. Additionally, the ConvNet outperformed other feature extraction methods like HOG descriptors and handcrafted features. Dhungel et al. (2017) also reported better results when using a ConvNet when compared to handcrafted methods. The model was trained in a supervised way and the last layer used to train a Random Forest classifier. Their obtained accuracy was 0.82 on the INBreast database. An original strategy was used for regularizing the ConvNet. Authors initialized the model randomly and trained it for regression, a task where the model learns to predict continuous values instead of a set of classes. With this method, the ConvNet is trained to output handcrafted features, often used in mammography image analysis. Finally, the model is fine-tuned for classification, improving considerably. In fact, regularization plays a big role, when the amount of data available is smaller. The aim of these methods is to increase generalization, which is hard when we have a small number of training examples. Dropout is probably the most common technique and was used by many works, including Geras et al. (2017), Dhungel et al. (2017) and Arevalo et al. (2015). This method

is explained in detail in chapter 3. Additionally, Arevalo et al. (2015) also uses max-norm regularization. This consists in setting an absolute upper bound to the values of parameters of each unit in the neural network, and has been shown to have a positive effect in model generalization.

Artificial data augmentation is also seen as a regularization strategy. For instance, some authors randomly crop the image around a lesion center, which can increase substantially the number of samples. Samala et al. (2016) use jittering, which consists of selecting randomly, as a central pixel of the patch, one of the points near the lesion's center, at each iteration. Similarly, Lévy and Jain (2016) use cropping to increase the dataset size by a factor of five. Again, Geras et al. (2017) use random crops, which include almost the whole breast in the same manner. Additionally, authors also use this at test time, to make predictions more stable. For this, ten inputs are fed to the model and their prediction is averaged.

Different to ImageNet, a mass can appear in any orientation. As such, additional types of distortions can be added. For instance, Sun et al. (2016) increased the number of examples by rotating them by $90^{o} \times k$, with equal probability for $k \in \{0,1,2,3\}$, and horizontally mirroring. The same strategy was used by Arevalo et al. (2015) and Samala et al. (2016). By these two operations, one can increase the dataset by a factor of 8, with a very simple implementation. Also, no additional RAM memory is necessary, as these operations can be done online, during the training phase. In the case of Lévy and Jain (2016) five rotations were taken from each mass, requiring interpolation.

One common problem to all these strategies is that the new data is highly correlated with the original one. Ideally, one would want as much "real" data as possible. Sun et al. (2016) proposed the use of unlabeled data to this end, in a semi-supervised learning strategy. They extracted, from a private dataset, 2400 examples from which 100 were labeled. They first computed 21 shape, intensity, texture and global features from each patch and applied dimensionally reduction methods. After this, they trained ten classifiers, each with different feature sets, that will classify data. An unlabeled example is assigned a class if a minimum number of classifiers agree. This way, a majority of the data will be labeled, and a few difficult examples will remain unassigned. Using the labeled examples, a ConvNet is trained in a supervised way and then used to label all data. Their final AUC is 0.8818, significantly more than a model trained on only 100 examples (0.8236). In this way they show that it is possible to rely on unlabeled examples to increase the amount of data available. Another unusual approach to the data-bottleneck was proposed by Kooi et al. (2017c). In this paper, authors propose to add patches of normal tissue to malignant masses or solitary cysts. The summing operation is done pixel-wise, in the log space, so that physically plausible patches are obtained. As seen previously, when augmenting the dataset, it is important that the variation we are introducing is already present in the distribution from where we are sampling the data. Results show a small improvement but not statistically significant.

2.3.3 Automatic Detection of Breast Cancer

Many of the works described are based on the classification of masses against either benign lesions or normal tissue. For a real world application this would require a specialist to select which regions need to be tested, limiting the potential of these methods. Some authors rely on detection systems using conventional approaches. This is the case of Kooi et al. (2017b), which use for initial mass detection an algorithm similar to (Karssemeijer and te Brake, 1996). It works by computing 5 features based on first and second order Gaussian kernels, which are correlated with malignancy patterns. The ConvNet is then applied to the top 10 suspicions regions of an image. More rarely, some works focus on the use of deep leaning methods for detection. Although not related to mammography, the work of Ciresan et al. (2013) is worth mentioning. In 2012 they were the winners of ICPR mitosis detection competition. Their dataset consisted on 50 high-power fields in H&E stained biopsy slides, which contained many nuclei, but only a few mitosis (positives). To construct a labeled dataset for training, they sampled all pixels near a mitosis center as positives, and grid sampled other regions to obtain negatives. This yielded a heavily imbalanced dataset, where the majority of samples were obvious false positives, while some examples, which contained nuclei, were more difficult. To counter this problem, authors first created an auxiliary ConvNet, trained on this dataset. Then, they selected only the incorrectly classified as positive patches, to be used as negatives in the main classifier. At test time each image was screened for all possible patches, to obtain a probability map. Additionally, the image was rotated and mirrored, and the same operation was performed. The resulting maps are aligned and averaged to obtain a final prediction. Authors argue that, because these models have large variance and low bias, this combination significantly improve the results. Additional techniques were used to improve performance which include the use of multiple models and smoothing of the probability maps, through linear filtering.

The system proposed by Dhungel et al. (2017) consists in the use of multiple models to detect, segment and classify individual mammography images, with minimal user intervention. For detection authors use a cascade of simple to complex classifiers, where the next module will only classify the previous cases assigned as positives. Due to the fact that, for each positive we have many negative examples, this framework works by, at each point, discarding the most obvious negative samples while keeping as much true positives as possible. First a sequence of three DBN models are trained, as explained in section, to classify pixels as positive (if they belong to a mass) or negative, based on a local information. The first takes as input positive and negative pixels at a coarser resolution, while the next ones only take inputs previously classified as positive at an increasingly finer resolution. The resulting positive pixels are combined by union with a Gaussian Mixture Model trained only at the finer resolution. After this, connected pixels are considered as possible lesions and pass on to the next stage. These examples are then processed by two sequential ConvNet models, where, by the same logic, the second only sees previously classified as positive examples. A third stage follows with two sequential Random Forest classifiers trained on a set of handcrafted features. At this point, their system requires a user to reject false positives.

Although complex, the framework proposed has two main advantages. First, simpler models are used to discard false positives, while more complex models are used only on the difficult cases, allowing the whole system to be more computationally efficient. Second, due to the multiple stages filtering out many negatives, while keeping a big portion of the positives, the data becomes increasingly more balanced.

As an alternative, some authors have proposed systems that feed a whole mammography exam to the model. This has the advantage of allowing the combination of both CC and MLO views. However, to keep the needed detail for diagnosis, a big model is necessary, which requires more computational examples to train and can overfit more easily if the amount of data is not enough. The system proposed by Carneiro et al. (2015), obtained good results while resizing images to 264×264. Their deep model received as input 6 images, one mammogram and two binary images with lesion segmentation per view, and integrated the information at a later layer in the network. As discussed previously, this required ground truth information about lesion location and, as such, cannot be realistically implemented in practice. The work by Kooi et al. (2017a) uses a similar approach to combine information of both breasts, requiring only mammograms for classification. Very different from most deep learning models applied on mammography, they used image sizes of 2600×2000. To do this, author collected data from almost 18000 patients during two years. Additionally, some architectural and learning strategies compromises were done to cope with the computational cost of training the model. The batch size, used for gradient descent, was reduced to 4 and early layers had big stride values, so that internal feature maps resolution would decrease quickly. They applied the model to distinguishing between BIRADS 0, 1, 2 obtaining an average AUC of 0.685 when testing one class vs all others. Experiments show that reducing image size or number of training examples lowered performance. Authors predict that, if more data was collected, results would be better.

Another important part of Geras et al. (2017) work, is the interpretability of the results. Instead of only returning a number with the probability of cancer, authors propose a visualization method to measure which pixels were important for that classification. They do this by introducing a perturbation in each input pixel and measuring the change in entropy between the three BI-RADS class probabilities. Lévy and Jain (2016) used a similar visualization technique, based on the same principle, to study what the deep model was paying attention to when making a decision. In their case, they use saliency maps (Simonyan et al., 2013), which consist of the image pixels' gradient with respect to the final class scores. Results showed the model was sensible to pixels at the edge of lesions.

2.4 Summary

In this chapter, previous work was presented both in the field of CAD systems and Convolutional Neural Networks. Conventional approaches to mammography image analysis often consist of four basic steps: 1) preprocessing; 2) detection and segmentation; 3) Characterization; 4) Classification. Authors have proposed many different approaches to each of these tasks. In terms of Convolutional Neural Networks, the review focused on an historical perspective as well. Over the last 20 years important work has been done that culminated in the design of very efficient deep models. With these, researchers have been able to solve specific tasks with incredible results. Additionally, original ideas are being incorporated into ConvNets, to increase its versatility and performance. The exploration of deep learning in mammography is still recent, but already achieved some good results. The application of these methods in CAD systems requires adaptation, to deal with unbalanced datasets, low amounts of data and importance of detail information.

Chapter 3

Convolutional Neural Networks

3.1 Notation

Before proceeding, here is presented the notation used for the rest of the document. There are some exceptions, but explained in the text.

3.1.1 n-Dimensional Arrays

First, in formulas, we use n-dimensional arrays. These are a finite collection of numbers that extend in n dimensions. We call n rank. Importantly, they have fixed sizes for each dimension.

A rank 0 array is a scalar and noted by a lower-case letter:	a
A rank 1 array is a vector and noted by a bold lower-case letter:	$\mathbf{a}$
Arrays with rank > 1 are noted with an upper-case letter:	A

To note elements that belong to the n-dimensional array A we use the same letter, lower-case with the index in subscript, e.g.: $a_{1,1}$.

3.1.2 Operations

In terms of operations there are two that need clarification:

Concatenation between arrays:	$(A_1, A_2, ..., A_n)$
2D strided convolution:	$A \circledast_s B$

The 2D strided convolution is just an extension of the normal convolution operator. The difference is that this operation is performed in two dimensions and we take steps of size S:

$$(A \circledast_s B)_{m,n} = \sum_{k_1=0}^{K_1/S} \sum_{k_2=0}^{K_2/S} (a_{k_1,k_2} b_{m-Sk_1,n-Sk_2}) \tag{3.1}$$

3.2 Artificial Neural Networks

Artificial Neural Networks (ANN) are models with applications in many research fields including machine learning. An ANN is composed of simple units, called neurons, organized in a complex system. Each neuron computes an output (activation), based on its inputs, which can be other neurons' activations or data.

The most common type of ANN is the fully-connected feed forward neural network. These networks have inputs, where data is fed, and outputs. Normally, the objective is to use this model to solve a regression or classification task, by approximating the output activation with a target value, for each input data. It is organized in sequential layers (feed forward), where a unit of layer k receives as inputs all neurons of layer $k-1$ (fully-connected), computes a linear combination of these values and passes it through a non-linear function:

$$o_{k,i} = actv(\mathbf{w}_{\mathbf{k,i}}^{\mathbf{T}} \cdot \mathbf{l}_{\mathbf{k-1}} + b_{k,i}) \tag{3.2}$$

Where $o_{k,i}$ is the ith unit of layer k and $\mathbf{l}_{\mathbf{k-1}}$ is the vector with all activations of layer $k-1$. The vector $\mathbf{w}_{\mathbf{k,i}}$ and scalar $b_{k,i}$ are parameters, often called weights, and are learned for a specific task. The non-linear activation function $actv$ can take many forms, and is later explored in section 3.3.1.3. The universal approximation theorem (Cybenko, 1989) states that, a model with a single hidden layer (all except input and output), with a finite amount of units can approximate any bounded continuous function to an arbitrary error, if the adequate parameters are given. This however, does not give us any information on the possibility of learning those weights.

3.3 Convolutional Neural Networks

Here we discuss ConvNets, a type of ANN, in the particular case of image inputs. Some concepts do not apply directly to other types of input data, but can be easily generalized.

These models can be divided in two parts, convolutional and fully-connected. The first can be thought of as a spatial feature extractor while the second is essentially a fully-connected feed forward neural network.

In the convolutional part, each layer is organized as a 3D volume, with width, height and depth. Each neuron receives as input the previous layer's neurons within a specific interval of width and height, to which we call receptive field. Further, all neurons in the same layer at the same depth have the same parameters. Due to this, the output of a neuron is independent of its horizontal and vertical position, hence the name of shift invariant networks in Zhang et al. (1994). As such, what we are effectively doing is applying a series spatial filters to the previous layer. The number of filters is the same as the depth.

Each layer has a stride parameter which controls the step size, when filtering the image. As such, if we have a stride of s, the height of this layer will be $1/s$, when compared to the previous. The same goes for width. Technically we can have different strides for each orientation, but that is practically never used. Normally, this operation, used to reduce the spatial resolution, is achieved

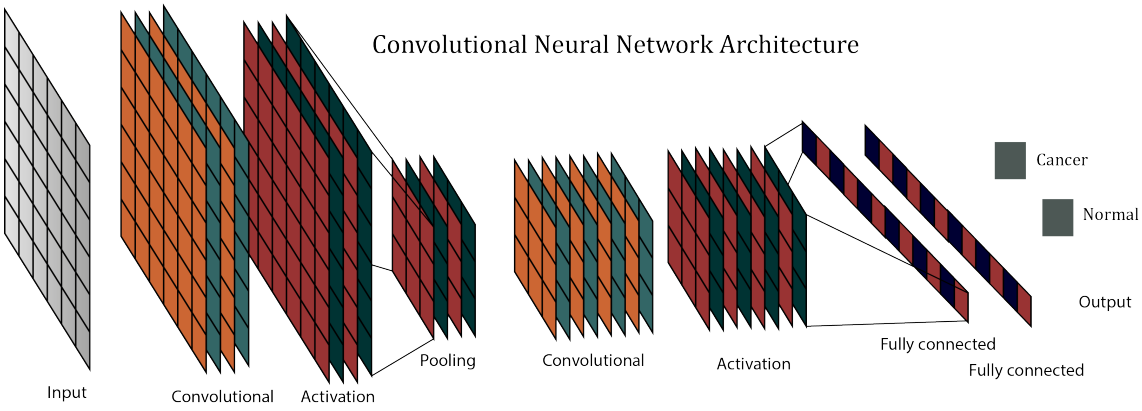


Figure 3.1: Diagram of a Convolutional Neural Network Architecture.

using pooling layers, as discussed next. This property is important, as it allows following neurons to "see" a feature representation of a bigger portion of the input.

In a ConvNet, later layers are trained on feature representations computed by earlier ones. Due to this, these models are well suited to learn from signals that can be described by a hierarchy of features (Lecun et al., 2015a), where lower level components are used to compose higher level components. Many natural signals can be well described this way.

The fully connected part of the network has the same structure of a fully-connected feed forward neural network. Ultimately, this is just a portion of the model that does not have the same spatial and parameter restrictions imposed on the convolutional part. Due to this, the fully-connected part is simply a generalization, and one can convert these layers in convolutional ones, as explained in section 3.3.1.5. Some authors have proposed fully convolutional models(Shelhamer et al., 2016).

Fig 3.1 shows a diagram of a ConvNet architecture.

3.3.1 Layers

As seen previously, ConvNets are feed-forward models composed of many layers. Most of these have parameter and spatial restrictions, as described next. However, they differ in the transformations they apply to its input. Here we describe all layers utilized in this work. Note that the same architecture can learn to solve very different problems, as long as the parameters are well optimized for each of them.

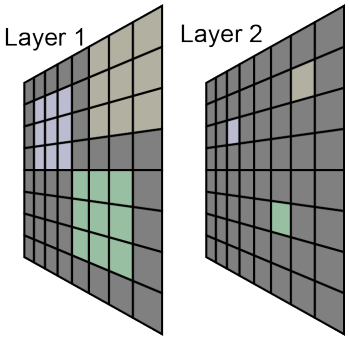
3.3.1.1 Input

The input layer is just a representation of the raw data that is fed to the model, which needs to have a fixed input shape. In the most common case, an image is converted in a 3-dimensional array, with shape $[w, h, 3]$, where w and h are the width and height. The last dimension is often 3 due to the use of RGB colored images. In this work, X-ray images only have an intensity channel. When using

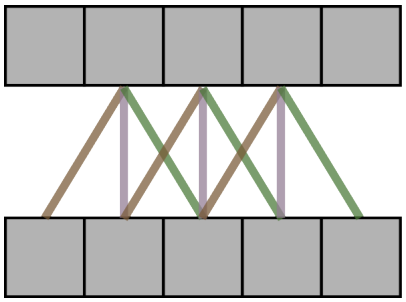
pre-trained models authors stack the same X-ray image three times, while, when training them from scratch an input of $[w,h,1]$ is used. In this work, all networks were trained from scratch.

3.3.1.2 Convolutional

These models get their names from convolutional layers. For this layer, we have $[w,h,f]$ neurons, where f is the depth. Each of these is connected to a region of the previous layer, depending on the size of the convolutional filters. For clarity, if we have a filter with size $(3,3)$, the neuron in position $[x,y,z]$ is connected to all neurons of the previous layer, with horizontal and vertical positions in the interval $[x-1:x+1],[y-1:y+1]$. We call this region the receptive field. Note that, this includes all depths of the previous layer. This is the spatial arrangement of the network, a special property of ConvNets.



(a) Receptive fields for neurons of Layer 2 shown in Layer 1 by the same color.



(b) Parameter sharing property. Same colors represent the same weight.

Figure 3.2: Illustration of the spatial properties of convolutional layers

Using the same example, the neuron in position $[x,y,z]$ will be connected to $3\times3\times f_{-1}$ inputs, where f_{-1} symbolizes the number of filters in the previous layer. Each of these connections has a parameter associated with it, to which we call weight. Additionally, this neuron has a bias term. The activation of this unit is computed by multiplying the output of each neuron in the receptive field, by the weight corresponding to their connection, and summing all these values together with the bias. The parameter restriction of the ConvNet, has to do with these weights. Neurons in the same layer, at the same depth share the weights and biases. However, due to the fact that they have different x,y positions, they will have different receptive fields, and so different activations. As such, each neuron computes a linear combination of its receptive field. This operation is shifted throughout the whole width and height of the layer. This is effectively a 2D convolution for each value of z. Formally:

$$R_z = \sum_{c=1}^{f_-} (L_c \circledast_s F_{z,c}) + b_z \tag{3.3}$$

$$R = (R_1, R_2 ..., R_f) \tag{3.4}$$

R, is the result of propagating input feature map L, with f_- filters, by a convolutional layer with f filters, weights F, and bias $\mathbf{b}$.

The result of the convolution operation shrinks the output, relative to the input. To account for this it is common to pad the image with $(m-1)/2, (m-1)/2$ zeros on each side, where m is the filter size, which is the same for horizontal and vertical directions, for convenience. To the same end, one can also use symmetric or constant padding.

In most typical architectures f grows as spatial resolution decreases. This can intuitively be explained by the fact that, while on earlier layers we have more general features, on later layers, with smaller resolutions, we have more specific, higher level features, requiring an higher dimensional space. Additionally, at early stages feature maps have bigger resolution, requiring more memory and thus penalizing high values of f.

(m,m) is normally small, e.g. $(3,3)$. Note that by stacking two $(3,3)$ layers, our effective receptive filter becomes $(5,5)$. Additionally, this has the advantage of diminishing the number of weights to optimize ($2 \times 3 \times 3 \times f$ vs $5 \times 5 \times f$). As noted by Simonyan and Zisserman (2015), this also allows us to introduce a non-linear function in between, which makes the function more discriminative. This however has the disadvantage of requiring more memory to represent intermediate feature maps.

Szegedy et al. (2015) utilized 1×1 filter sizes to reduce the depth of the previous layers, decreasing computational cost and memory consumption. This, however, is not a common practice.

One important thing to note is that the resulting feature map has f filters, independently of the number filters in the input.

As for the stride, s, the most common value is one. Some authors, utilize higher strides, mainly at the first stages, to reduce the spatial resolution quickly, due to memory limitations. This approach can lose information, but is useful to allow bigger inputs to the network.

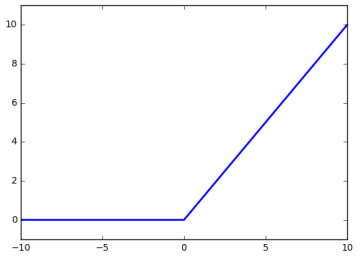
3.3.1.3 Activation

Activation functions introduce non-linearities in the model, making the model function more discriminative. This is important because, otherwise, the computation of the output would be a simple linear combination of the inputs, and so could be reduced to a single layer. They perform a simple element-wise operation in the model, conserving layer size, without needing any learned parameters.

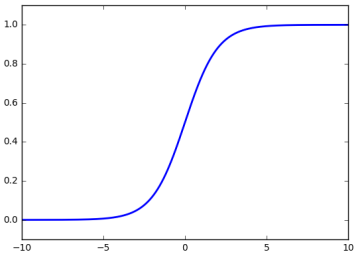
The ReLU function is the most commonly used as it facilitates training. Other examples include the sigmoid and hyperbolic tangent. However, these contain regions where the derivative is very small, and, as explained in section 3.4.1, this can impair learning.

$$\text{ReLU:} \qquad r_{m,n,c} = max\{0, l_{x,y,z}\} \tag{3.5}$$

$$\text{sigm:} \qquad r_{m,n,c} = \frac{1}{1+e^{-l_{m,n,c}}} \tag{3.6}$$



(a) ReLU activation function



(b) Sigmoid activation function

One of the problems with ReLU is that the derivative for an input $r_{k,i,j} < 0$, is 0. If, for some reason, during learning a unit always receives negative values, it will never be activated, a phenomenon called dying out. This is not a big problem, particularly if a careful initialization of the weights is done and adequate learning rates are used. A leaky ReLU has been proposed to address this issue, which returns $max(\alpha \times l_{k,i,j}, r_{k,i,j})$ with the α parameter assigned to a low value.

In this work, for the activation layers, we selected ReLU, as it is the most commonly used, and allows faster learning.

3.3.1.4 Pooling

In ConvNets architectures, it is common to periodically insert pooling layers. There are some exceptions to this (He et al., 2015). These have different functions in different architectures. Most commonly, they are used to reduce the spatial size of the network, provide some translation invariance and reduce overfitting. In the case of Szegedy et al. (2015), some pooling layers act as any other filter.

This operation is done individually to each depth, just like activation layers. The max pooling function is the most used. Each neuron outputs the maximum for a small region of the input, depending on the filter size. Due to this, for a slightly shifted input feature map, the pooling layer will output the same result, thus providing some translation invariance to the model. Additionally, when computing the maximum, we are discarding some information, depending on the filter size, which is an interesting property to avoid overfitting.

Pooling layers usually have a stride bigger than one. This is important as it reduces the computational power required, memory and number of parameters that need to be learned, in the

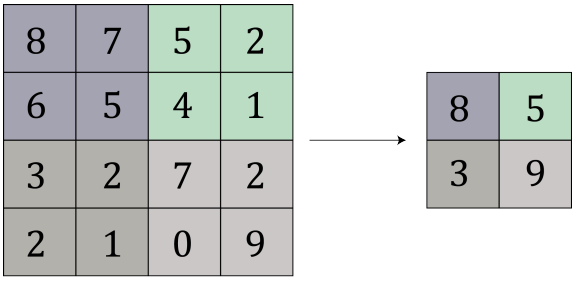


Figure 3.4: Max-pool operation on a small 2-dimensional array. In this case, $m = 2$ and $s = 2$

following layers. This spatial reduction also allows the units of later layers to be influenced by a bigger region of the original image. This is vital to ensure the property that convolutional neural networks excel at modeling compositional hierarchies.

$$r_{x,y,z} = max(l_{i,j,z}), \qquad with\ i \in [s \times x, s \times x + m[\,, j \in [s \times y, s \times y + m[\tag{3.7}$$

Some authors have also utilized average pooling, which works in the same way, but computing the average value, instead of the maximum. An important difference between these is that, while the first outputs the same value if a feature exists in the receptive field, average pooling computes the mean strength of that feature in that region. Additionally, max-pooling is a non-linear operation. An example of this operation is shown in fig. 3.4.

3.3.1.5 Dense

Dense layers work similarly to the hidden layer in the fully-connected feed forward neural network. They do not preserve spatial structure of the input. If the previous layer's output has a spatial structure, a flattening operation is performed, reshaping the rank 3 tensor with size $[w,h,d]$ to a rank 1 of size $[w \times h \times d]$, before computing the output. The number of weights in this layer is given by $n_l \times (n_{l-1}+1)$, where n_l is the number of units in layer l. The additional weight per neuron refers to the bias parameter. For consistency, because in this description of convolutional neural networks we considered convolutional and activation layers separately, for the fully-connected we assume that only the linear part of the operation is performed and that an activation layer is stacked on top.

As described, this layer performs the following operation:

$$o_i = \mathbf{w_i^T}.\mathbf{l_{k-1}} + b_{k,i} \tag{3.8}$$

$$\mathbf{o} = (o_1, o_2, ..., o_d) \tag{3.9}$$

The result is a vector of d linear combinations of all the inputs. If we have a convolutional layer with the same filter size as its input, filters will only be applied in one position, the one where they completely fit the input feature map. Due to this and to the fact that parameters are not shared for different depths, the layer is effectively computing a linear combination. No weight is used twice. As such, dense layers can be transformed into convolutional ones, just by reshaping the weights to the input size.

3.3.1.6 Output layer

The output layer is normally a linear combination of the inputs, coupled with a non-linear function, between the last fully-connected layer and the output neurons. For classification settings, the case of this work, the output of the model is a set of values, each one representing the probability of the input belonging to a specific class. For problems with a number of classes, K, greater than two,

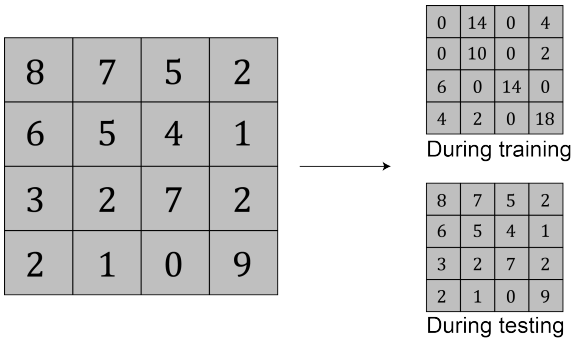


Figure 3.5: Dropout Layer, $\sigma = 0.5$

we have K output neurons, each computing the probability with the softmax function:

$$P(C_j) = \frac{e^i_j}{\sum_{k=1}^{K} e^k} \tag{3.10}$$

In the case of two class classification we can use the softmax function with two outputs or, instead, use one neuron and compute the sigmoid function. For the two classes the probability is given by:

$$P(1) = \frac{1}{1+e^{-i}} \tag{3.11}$$

$$P(0) = 1 - P(1) \tag{3.12}$$

We opted for the latter.

3.3.1.7 Dropout

Dropout is a very common technique for regularizing ANN, including deep learning models. Initially proposed by Srivastava et al. (2014), the idea is to build more robust features by preventing neurons from co-adapting. Its implementation is simple, and only requires stacking additional layers in the network, usually after activation functions. This module randomly sets some points of the input feature map to zero. Formally, each of these has an independent probability σ of being kept and, if this happens, it is scaled by $1/\sigma$. Points not kept are set to zero.

As we can see, this layer only has one parameter, σ, which will be in the interval $]0,1[$ for training, and set to 1 for testing. Intuitively, this process can be thought of as removing some neurons from the network, temporarily, along with its input and output connections. As such, in each iteration a slightly different network architecture is effectively being trained. This relates to the idea that averaging the combination of multiple models increases performance, which is generally true, but computationally expensive. At test time no unit is dropped, effectively combining the prediction of all trained neurons. The scaling operation is done so that each layer, during test is approximating the average of all trained networks, instead of the sum.

Another good intuition for why this works is to think that the dropping mechanism penalizes neurons which rely on fewer input connections. This happens because a drop in a subset of inputs will be more significant when compared to a neuron which relies on many inputs. And, in this manner, more general features are privileged.

3.3.1.8 Batch Normalization

Batch normalization is a recent but very efficient technique. While training deep models, weights are updated at every iteration. A side effect of this is that, in each layer, the input distributions will change, a phenomenon called internal covariate shift. Ioffe and Szegedy (2015) identified this slows down training, requires a more careful weight initialization and hinders optimization of models with saturating nonlinearities, such as sigmoid or hyperbolic tangents. Additionally, they propose the batch normalization method to address it.

Similar to dropout batch normalization is implemented as a layer in the network with different behaviors during training and inference. To address the problem of internal covariance shift, for each training batch (see section 3.4), this layer normalizes its input by subtracting the mean and dividing by the standard deviation, of all neurons at the same depth. The mean and standard deviation are referred to as mini-batch statistics. Additionally, a running average of these is kept to be used during inference. Otherwise, the output of the model for a new example would depend on mini-batch statistics, which are affected by other inputs running in parallel. To guaranty that the model, at that stage, can represent the exact same function with or without batch normalization two new trainable weights are added, γ and β, which scale and offset the output. The output is therefore given by:

$$I_c = \gamma \frac{I_c - mean(I_c)}{std(I_c)} + \beta \qquad , during\ training \tag{3.13}$$

$$I_c = \gamma \frac{I_c - u_c}{v_c} + \beta \qquad , during\ inference \tag{3.14}$$

Where u_c and u_v are running averages of the $mean(Ic)$ and $std(I_c)$.

It has been demonstrated that, batch normalization allows higher learning rates and makes the model converge in fewer iterations. Authors also demonstrated that this method has a regularization effect, and recommended higher σ values in dropout layers or even removing them entirely.

Although in their paper, batch normalization layers are used between linear and activation layers, some authors have argued that their benefits are more notorious when used after activation layers. Mishkin et al. (2016) achieved a significantly better accuracy this way.

3.4 Optimization

The optimization problem can be formulated as follows. Suppose we have a dataset, D, with images $I \in \mathbb{R}^2$. These images can either be a lesion or not, and so, have a label associated with it $y \in \{1,0\}$. In our case we want to build a model that, given an input image I_i, produces a probability $p(I_i)$, that approximates as close as possible the label associated with that image, y_i.

Although there are other alternatives to optimize neural networks, like genetic algorithms, the minimization of a loss function by gradient descent is by far the most common. In the case of classification, cross entropy is commonly used as this loss function. It measures how well one distribution of probabilities approximates another, given a set of events, in our case, I. We want to measure how well $p(I_i)$ can approximate the label y_i. This is given by:

$$L = -\frac{1}{|D|}\sum_{i}^{|D|}(y_i \log(p(I_i)) + (1-y_i)\log(1-p(I_i))) \tag{3.15}$$

As previously seen, assuming an architecture for our model, the probability for an input depends only on its weights, θ, and can be noted as $p(I,\theta)$. Given θ we can compute $L(\theta)$ by first running the model on the dataset and then computing the cross entropy.

3.4.1 Backpropagation

As explained next, the computation of the gradients of the loss function with respect to the weights, $\nabla_\theta L(\theta)$, is necessary during training. This is done by a process called backpropagation, in which we first propagate the input through the network, compute $L(\theta)$ and then backpropagate this loss through all the weights in the network. More specifically, the gradient with respect to the output is given by:

$$\frac{\partial L}{\partial p} = \frac{\partial(-(y\log(p)+(1-y)\log(1-p)))}{\partial p} = \tag{3.16}$$

$$-\frac{y}{p} - \frac{1-y}{1-p} = \frac{-(1-p)y + p(1-y)}{p(1-p)} = \frac{p-y}{p(1-p)} \tag{3.17}$$

The derivative of the sigmoid with respect to its input, i, is given by:

$$\frac{\partial sigm(i)}{\partial i} = sigm(i)(1-sigm(i)) \tag{3.18}$$

Thus, we can compute derivative of L with respect to i, using the chain rule of derivatives:

$$\frac{\partial L}{\partial i} = \frac{\partial L}{\partial p}\frac{\partial p}{\partial i} = (p-y) \tag{3.19}$$

Now, if we want to compute the gradients of the loss function with respect to the weights of the last fully-connected layer, we can apply the same principle again. Because this layer computes a linear combination of its inputs, $\mathbf{a}$, the derivative of i with respect to the weights, $\mathbf{w}$, is simply the

vector **a**:

$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial p}\frac{\partial p}{\partial i}\frac{\partial i}{\partial \mathbf{w}} = (p-y)\mathbf{a} \tag{3.20}$$

This though can be easily extended to the previous layers and thus, we can compute $\nabla_\theta L(\theta)$ in an efficient manner.

3.4.2 Gradient Descent

To simplify, an explanation with one variable only is presented. In this case, we have a differentiable function, $f(x)$, and we want to find x, so that $f(x)$ is a local minimum. For this, we can compute the derivative $\frac{\partial f}{\partial x}$, which tells us in which direction the function is growing. As such, we can update x, so that it approximates a local minimum, by advancing in the opposite direction of $\frac{\partial f}{\partial x}$. We can do this iteratively, until we become close enough to a solution. One thing to note is that the steps taken into this direction need to be small, to avoid jumping over the solution.

Gradients are multi-variable derivatives, and gradient descent works in a similar way. In this case we want to minimize $L(\theta)$, by iteratively changing θ in the opposite direction of the gradient. To do this, we first initialize θ_0 with random values. The subscript denotes the iteration. Then, for each iteration, we compute the gradient of L with respect to the weights, $\nabla_\theta L(\theta)$. Finally, we set the next iteration weights, $\theta_{t+1} = \theta_t - \eta\nabla_\theta L(\theta)$, with η denoting the learning rate. It is important to select an adequate learning rate, as small learning rates slow the training process, while high learning rates might make updates to θ too large, impeding convergence. Note that, by using this method we do not guaranty the best possible solution, but only local minimum. However, as pointed out by Lecun et al. (2015a), in most cases this is not a big problem, as good local minimum are quite frequent.

Several variations of this method exist. In deep learning models, the gradient is not computed for the whole dataset, but rather for a small subset to which we call batch. As such, during training, and for each iteration, we extract a batch from the D, which is used as an approximation of the gradient of the whole dataset. With bigger batches, a better estimation of the gradient is obtained and so, higher learning rates should be used. This has been confirmed by experimental data in some works (Mishkin et al., 2016). Normally, this batch approach is called mini-batch gradient descent. If the batch size is one, then we have stochastic gradient descent.

In many works, a dynamic learning rate is used, allowing it to be decreased according to some rules. This has the benefit of smaller training times in the beginning while allowing for more careful optimization at later stages, when θ is close to local minima. The most common strategies are an exponentially decaying learning rate or abruptly changing it to a smaller value every k iterations.

3.4.3 Adam

Some authors proposed variations of basic gradient descent, which have been shown to converge faster and, in some cases, achieve better final solutions. For instance, a common practice is to incorporate momentum in the update rule. In this case, we change the update rule to:

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta L(\theta) \tag{3.21}$$

$$\theta_{t+1} = \theta - v_t \tag{3.22}$$

This way, if, for a weight θ_i, frequent updates in the same direction are done, increasingly bigger steps are taken in that direction. Contrarily, weights that have been oscillating are changed more slowly. This concept is easier to visualize if we assume $L(\theta)$ is a 2D surface, where each point is a set of values for θ and the height in that point is the value of $L(\theta)$. With simple gradient descent, when we enter a valley, we will go down at a constant speed for many iterations. With momentum, we will gain speed in that direction, reaching a terminal velocity.

The Adaptive Moment Estimation, or Adam, is a method which increases convergence speed using this idea. Similar to momentum, a running average of the past gradients is kept. Additionally, it incorporates the idea, present in algorithms like Adadelta and RMSprop, of favoring the update of weights that have not been frequently updated. For this, a running average of the squared gradients is kept. Formally, for one parameter θ_i, if we consider the gradient at time t, $g_{i,t} = \nabla_\theta L(\theta_i)$ we have:

$$m_{i,t} = \beta_1 m_{i,t-1} + (1-\beta_1) g_{i,t} \tag{3.23}$$

$$v_{i,t} = \beta_2 v_{i,t-1} + (1-\beta_2) g_{i,t}^2 \tag{3.24}$$

Where $m_{i,t}$ and $v_{i,t}$ are the estimated values for the gradients and squared gradients, respectively, and β_1 and β_2 are selected parameters. Because $m_{i,1}$ and $v_{i,1}$ are set to be zero, at the beginning, authors do the following correction:

$$\overline{m_{i,t}} = \frac{m_{i,t}}{1-\beta_1} \tag{3.25}$$

$$\overline{v_{i,t}} = \frac{v_{i,t}}{1-\beta_2} \tag{3.26}$$

And the update rule for each weight is given by:

$$\theta_{i,t+1} = \theta_{i,t} - \frac{\eta}{\sqrt{\overline{v_{i,t}}} + \varepsilon} \overline{m_{i,t}} \tag{3.27}$$

This method was utilized during experiments. After the first few, it was possible to observe that Adam optimization, generally, reached better solutions, and took fewer iterations than mini-batch

gradient descent. However, each individual iteration took more time.

3.4.4 L2 Regularization

One very common method of improving the generalization ability of a model is L2 regularization. Using this method, high weights are penalized in the loss function thus forcing predictions to be based in many features instead of a small subset. To this end, we simply add a term to the loss function:

$$L'(\theta) = L(\theta) + \lambda\theta^2 \tag{3.28}$$

λ is a constant to define how aggressive the L2 regularization is. Note that, when we compute the derivative we get $2\lambda\theta$. When most implementations the 2 is ignored. Effectively what we are introducing is a term that, at each iteration is decreasing the weights by a fraction of their current value, or, in other words, we are exponentially decaying weights. Because of this, the λ constant is also called weight decay. Some authors have reported that introducing this penalization increases not only generalization but also training accuracy as well (Krizhevsky et al., 2012).

3.5 Detection as a Classification Task

By building a detector, we can work locally in the image. This has the major advantage of reducing the input size of the model, while keeping some detail.

So far, supervised learning for classification tasks was explored. Here we extend this formulation in the context of detection. For this, we simply create a dataset composed of selected regions from whole mammography images. Then, we use this dataset to train a model that is able to distinguish between regions centred in a lesion and normal regions. Finally, we apply this model to new unseen images in all possible regions. The formulation is as follows.

Suppose we have a set of mammography images D. Each image I_i has an associated binary mask, B_i, which specifies which pixels belong to a lesion. To each image, $I_i \in D$ we apply two sampling algorithms, S_p and S_n, which will return smaller regions to which we call patches. S_p returns a patch centred in the middle of every lesion, while S_n returns many patches that do not include any point p so that $B(p) = 1$. Details about the sampling algorithms are described in section 6.5.3. Following this idea we can build two datasets:

$$D_p = \{S_p(I_i, B_i) | I_i \in D\} \tag{3.29}$$

$$D_n = \{S_n(I_i, B_i) | I_i \in D\} \tag{3.30}$$

Now, for training model M, in each iteration, we select n samples from each dataset D_p and D_n. M is optimized in a supervised way, as for classification, as specified in previous sections. For inference, we build a detector model, M_D, based on the weights learned by M. M_D takes an input

image I and outputs a probability map, P, where, for each point, we have the probability of that point being a lesion center.

A common approach to building M_D is to sample a patch in each point, q, feed it to M, and assign $P(q)$ to that value. Instead of this, we followed a computationally less expensive approach with the same numerical results, explained in section 5.1.

Chapter 4

Image Analysis

Although the bulk of this work is centered in machine learning techniques, image processing methods used are briefly explained in this chapter.

4.1 Preprocessing techniques

Preprocessing techniques allow the standardization of image conditions before further processing. These are important to increase robustness of the following algorithms. In this case histogram stretching was used as a global contrast equalization technique. This technique is simply a rescaling of the pixels according to the following rule:

$$I_c = \frac{I_c - min(I_c)}{max(I_c) - min(I_c)} \tag{4.1}$$

After this, we are certain that all pixel intensities in the image are in a controlled interval. Images were resized, so that small sized patches could contain a complete view of objects like lesions. This is important due to memory limitations and overfitting concerns, in the convolutional models. For this, bilinear interpolation was used.

In terms of ground truth information, we have access to binary images where all pixels with value "True" correspond to a lesion. In this case, the resizing operation was performed using nearest neighbor interpolation.

4.2 Morphological operations

Morphological operations are a set of methods which transform an input image based on a structuring element. They are classified as non-linear filters. Usually, these are used in binary images, but they can be extended to work with pixel intensity. Here we will describe the operations used, first for binary images and then extend it to with intensity images.

If we have a binary image, I, so that the value in point p, is either 1 or 0 we can define the set $I = p|I(p) = 1$. We do the same for the structuring element S, $S = p|S(p) = 1$. Although the

same letter is attributed to the image and the set, they are simply different notations of the same information. This facilitates notation of the following operations.

4.2.1 Morphological Dilation and Erosion

A binary dilation between I and S is noted as $I \oplus S$ and its result is given by:

$$I \oplus S = \{(p+q)|p \in I, q \in S\} \tag{4.2}$$

Effectively, this defines the dilation operation as all points which can be obtained by translating a point in the image, p by a point in the structuring element q. Alternatively we can think of this as all points in which the structuring element touches, if we make it slide through all points of the image. As we can easily see, the result of the operation will at least be the same number of positive pixels, as the image.

The inverse operation is erosion, noted as $I \ominus S$:

$$I \ominus S = \{p|(p+q) \in I, for\ all\ q \in S\} \tag{4.3}$$

In this case, what happens is that we only keep a point if all possible translations of that point by all the points in the structuring element belongs to the image. Again, we can think of it as sliding the structuring element through all the points in the input image. If, for a point, the whole structuring element is covered by the image, that point is kept. Contrary to dilation, these operation keeps at most, the same number of pixels as the input image.

Generalizing for the case of intensity images, we can no longer work with sets, due to the fact that each pixel has an intensity value. As previously seen, these operations can be computed by shifting the structuring element through all points of the binary image. In this case we shift the structuring element through all the pixels in the image. If we have an image, $I(x,y)$ and the binary structuring element, S, the dilation and erosion operations are given by:

$$I \oplus S(x,y) = max_{[u,v] \in S}\{I(x-u, y-v)\} \tag{4.4}$$

$$I \ominus S(x,y) = min_{[u,v] \in S}\{I(x-u, y-v)\} \tag{4.5}$$

In some cases the structuring element have real values. Although not used in this work, the definitions seen before can be easily extended the following way:

$$I \oplus S(x,y) = max_{[u,v] \in S}\{I(x-u, y-v) + S(u,v)\} \tag{4.6}$$

$$I \ominus S(x,y) = min_{[u,v] \in S}\{I(x-u, y-v) - S(u,v)\} \tag{4.7}$$

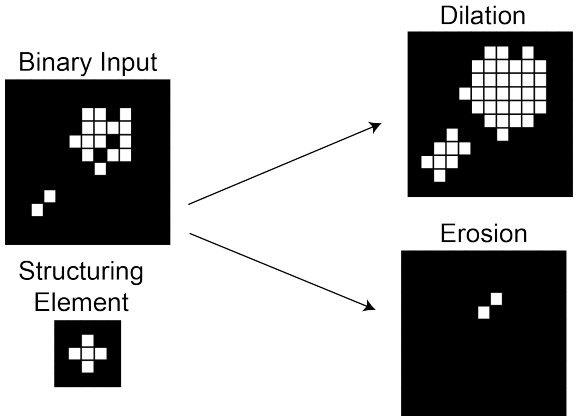


Figure 4.1: Binary Erosion and Dilation illustrated.

4.2.2 Morphological Opening, Closing and White Top Hat

The operations presented next, are obtained by composition of dilation and erosion and thus, they have the same formulation in binary and intensity images.

The opening operation is normally used to remove small white regions in binary images. In opposition, the closing operation removes small black regions. The size and shape of the structuring element influences what regions will be filtered and which will be kept. They are defined as:

$$I \circ S = (I \ominus S) \oplus S \tag{4.8}$$

$$I \bullet S = (I \oplus S) \ominus S \tag{4.9}$$

In this work, a white top hat was used as part of an algorithm for artifact removal. This operation is defined as:

$$T_w(I,S) = I - I \circ S \tag{4.10}$$

If we analyze this operation, considering white objects as elements, we can see that, first we compute the opening operation, which will remove small elements. After this, we subtract this image to the original. By doing this we are removing all elements that were not filtered by the opening operation. In theory, the final result will keep only small white objects. The size of those elements can be adjusted by defining an adequate structuring element.

Chapter 5

Contributions

In this chapter we expose the contributions of this work. These consist of architectural changes to the models and learning strategies. Each of the following methods were investigated with the aim of solving some problems that come from the application of ConvNets to medical datasets. In particular, we address the positive negative unbalance and the low amount of data.

5.1 Image Screening with Convolutional Neural Network Detector

Normally, after training the model as seen in chapter 3, to detect lesions in a new image, all possible regions are tested individually. Here we show a much more efficient way to do this, changing the architecture of the model. If we have a squared sized patch, with side length l, one convolutional layer will perform $O(l^2)$ operations, in that patch. Now, for a squared sized image of side length w, we would extract $O(w^2)$ possible patches. The total number of operations would be proportional to $O(l^2 \times w^2)$. This method is inefficient due to the fact that we are computing the convolution for the same region multiple times. This happens because patches overlap by $l-1$, when we extract all possible image regions.

We could compute the same numerical result by performing the convolution on the whole image, with the same weights. Then, the number of operations would be limited by $O(w^2)$. In theory, this approach can be significantly faster. For instance, if $l = 10$ we are performing $l \times l = 1/100$ of the operations with this method. This way we effectively avoid redundant computations.

We can apply this to all layers with spatial structure with some restrictions:

1. The layer should not pad the input. Otherwise, with the sampling approach, some zeros are used to compute the output result. These zeros cannot be inserted in the middle of the image for every possible patch to perform the convolution over the whole image. As such, the numerical result will not be the same.

2. All filters should have stride of 1. Let's consider the example of having a stride of two in one dimension for simplification. In this case for the first region the model would place the filters in positions $[1,3,5,...]$. For the second region the model would run for the positions

$[2,4,6,...]$. There is an offset and so, if we use a stride of two for the whole image, we will not have the same numerical results.

For the first restriction we settled with using layers with no padding for all models intended to screen the image. For the second, we can apply the filter without and with the offset and save both results. Then, computations continue for the next layers in the two outputs. For the case of images we generalize as follows. Suppose we have a feature map R, which is a 3-dimensional array. Before passing it through a layer with stride, s, we create a set of copies with small offsets, G. The operation we want to perform is then applied to all of these copies, individually.

$$G = \{translate(R, s_x, s_y) | s_x \in \{0,...,s\}, s_y \in \{0,...,s\}\}$$

Where $translate(R, s_x, s_y)$ denotes a copy of feature map R translated s_x pixels to the right and s_y to the bottom.

In the case of dense layers, we need to transform them so that they preserve spatial structure. As seen previously, after the convolutional part of the network all units are flattened into a vector. So, for the first dense layer, we can transform it into a convolutional layer by reshaping its weights in the opposite way. As such, the transformed version will have an input shape of $[w,h,c]$, a filter size of $w \times h$ and the same number of filters as before. For the following layers we perform 1×1 convolutions with the same number of filters, as previously seen.

5.2 Convolutional Layer with Rotated Filters

As seen previously, in mammograms, lesions do not have a particular orientation. Motivated by this, we explore a custom version of convolutional layer, using rotated filters. In convolutional layers, weights are shared only across width and height. Using this variation, the parameter sharing property is partially extended to filters with different depths. This is done by rotating each set of weights by $k \times 90$ degrees with $k \in \{0,1,2,3\}$ to obtain a different feature map.

For instance, if we had 32 filters in a convolutional layer with size 3×3 and an input map of size 16 we would have $32 \times (3 \times 3 \times 16 + 1)$ parameters to optimize. With rotated filters, we can obtain the same number of internal feature maps, reducing the number of weights by a factor of 4. Formally, using the previous convolutional layer definition:

$$R_{i,k} = \sum_{c=1}^{C} (I_c \circledast_s rot90(F_{i,c}, k)) + b_i \tag{5.1}$$

$$R_i = (R_{i,0}, R_{i,1}, R_{i,2}, R_{i,3}) \tag{5.2}$$

$$R = (R_1, R_2..., R_{d/4}) \tag{5.3}$$

This method can be seen as a regularization strategy. Effectively, we are using field knowledge to reduce the number of parameters to optimize. Note however, that rotational invariance is not achieved, but only filter symmetry. Intuitively, due to the fact that a mass can have any particular orientation, the rotated version of a discriminant feature, should also be discriminant.

During learning, it is common for all the training data to be correctly classified. In this case, the model has captured patterns, that are only present in this subset, and do not generalize to unseen data. This version of the convolutional layer, forces the model to learn filters in four directions. As such, it is unlikely that patterns specific to the training data appear in all four directions. Due to this, the rotated filters might hinder the models ability to learn non generalizable features.

Optimization is not changed. When we compute the gradients for each parameter in a convolutional filter, using backpropagation, we take in consideration the whole feature map. For this version, we do the same, taking in consideration four feature maps per parameter.

In terms of implementation, filters are rotated before performing the convolution. This leads to a small computational penalty. The time per training iteration increased by a factor always smaller than two. In principle, this technique could be generalized for more directions, but in the context of this work that was not attempted.

5.2.1 First Dense Layer with Rotated Filters

The same principle can be applied to the first dense layer as well, if we consider the transformation shown in subsection 3.3.1.5. As seen, this layer is equivalent to a convolutional one, where the filter size is the same as the previous layer's feature map size. After this conversion we apply exactly the same rotated filters modification.

Effectively, this restriction on the model's architecture forces the dense layer to extract the same features in four orientations. One interesting aspect of this approach is that the first dense layer often has many parameters when compared to the rest of the network. As such, the impact of rotated filters in this layer should be more significant than in the case of convolutional ones.

One possible disadvantage of this method is the introduction of redundant features. For instance, if one filter is capturing round objects well centered in the input, the introduction of rotated filters will quadruple the number of features encoding that information, removing space for other features and increasing the computational cost with no benefit. This happens due to the fact that, in this case, by rotating the filter we obtain the same result. In fact, this effect can happen to all layers with rotated filters.

5.3 Rank Learning

In some machine learning problems, we want to build a model that is able to tell which of two examples has a higher value for some property. For instance, for a recommendation system, we want to distinguish between a set of documents, in which one is more relevant than the others. In the context of this work, rank learning is used to artificially augment data and robustly address the

unbalance problem. Due to the much smaller number of lesion regions, compared to normal tissue, we create a model that learns to distinguish between the two classes, by learning the difference. This way instead of N_p and N_n positive and negative cases, we have a set constituted of $N_p \times N_n$ examples.

To this end, we create, during training, randomly ordered tuples of positive and negative examples. If the first case is positive, we assign it the label 1, and, otherwise, we label it as 0. Three setups were considered: 1) A ConvNet, specially designed to this strategy, is trained from scratch; 2) a ConvNet is fine-tuned after conventional supervised optimization and 3) The features extracted from a previously trained ConvNet are used to train a linear classifier using this approach.

5.3.1 Convolutional Neural Network for Rank Learning

Fig. 5.1 shows a representation of the proposed setup. Essentially, we have two identical models which share weights during training. Each computes a feature vector representation of its input. Then, the information of both is combined by subtracting these vectors at layer $\ominus$. A linear classifier computes the probability of the first input being a positive, based on the feature vectors difference. This setup is trained with gradient descent.

After training, we want to extract the probability of an input being a lesion. For this, one model is used to extract the feature vector and fed to the classifier. Because the classifier divides the space linearly, by training we are effectively discovering the direction that best separates the two classes, in the training data. Additionally, because positive and negative samples take turns in each input, the best plane that divides the feature space intercepts the origin. As such, the final model is completely defined using this training strategy.

Compared to normal supervised classification this method has the disadvantage of requiring more time and memory to train. This happens because each training sample is composed of two images, instead of one, which are propagated through the models, requiring intermediate layers results to be kept in memory for the backward pass. However, it is less probable that the model overfits training data, due to the huge amount of combinations between negative and positive samples.

5.3.2 Linear classifier

In the proposed second methodology, we use the same principle but instead of training a model, end to end, we use the representation learned from a previous experiment. The output probability for a new differences feature vector, x is given by:

$$p(x) = sigm(\mathbf{w}^T x) \tag{5.4}$$

$\mathbf{W}$ are trainable weights, optimized by mini-batch gradient descend, on the cross entropy loss function. In this case, the bias parameter is not necessary. Similar to the previous method, the

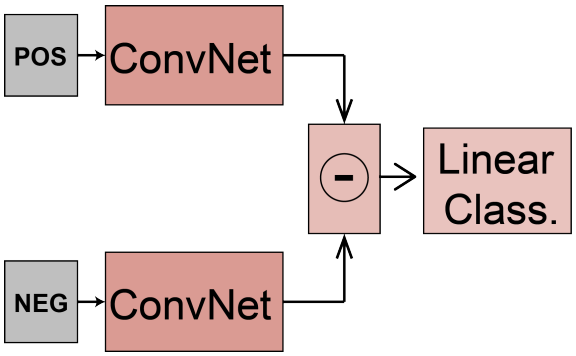


Figure 5.1: Rank learning applied to ConvNets. Models share parameters. Positive and negative samples alternate between inputs.

hyperplane that separates the classes should pass by zero, due to the fact that images can be fed to each input with the same probability.

For inference, we feed a feature representation of an image, computed with the same model used to obtain the difference feature vector, x.

This method has the advantage that it requires very small computational power for the training phase, once the initial model has been trained.

5.4 Cascade approach

As seen previously, the number of lesions is much smaller than the number of normal regions. Due to this, when we create a patch dataset to train a detector, it is heavily unbalanced. Naively training a model can yield high performances by simply attributing probability of 0 to all patches. Common approaches to deal with this problem include oversampling the positive cases or selecting only a few negative samples. As a side effect of these strategies, after optimization, the model becomes biased towards the minority class.

To solve this problem we explore a Cascade approach, by training multiple models used in succession. For this, the first model uses the initial dataset, oversampling the positive cases. This detector is then run on the training data, to create new patch dataset. All regions classified as positive are extracted. If they correspond to a lesion they are labeled as positive, otherwise they

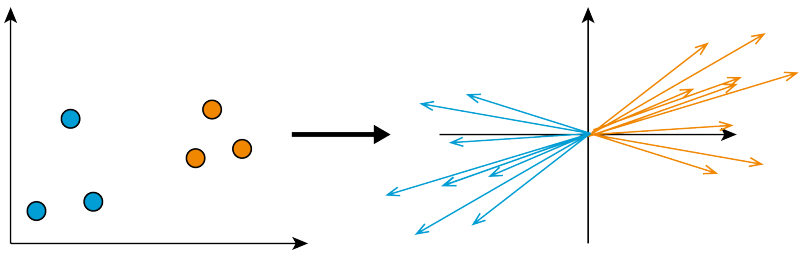
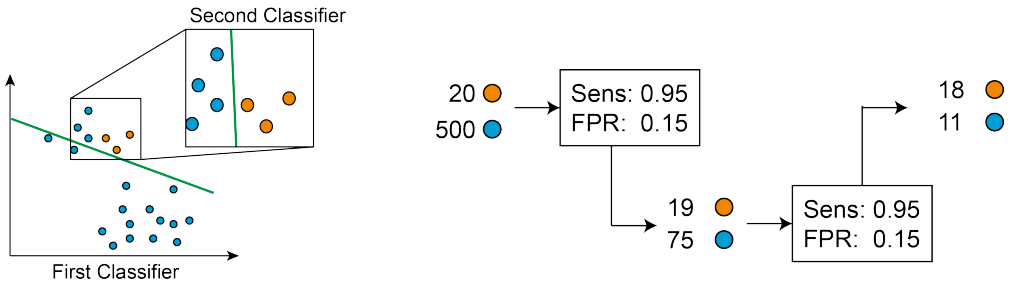


Figure 5.2: Illustration of data points in the original feature and the dataset created by taking the differences.



(a) (b)

are negative. Ideally, following this strategy, each model will reduce significantly the number of normal regions in the data, while keeping as much lesions as possible.

Conceptually, we speculate that some features, which might be optimal to discard obvious false positives, can work against the model to discriminate more difficult cases. For instance, image regions with high intensity are relatively rare and can occur due to lesions, the pectoral muscle boundary or epithelial tissue. As such, simple intensity features, which help the model to discriminate lesions from most of the normal regions, can have an undesirable effect in these few examples. The second model however, does not need to learn this difference, as it only receives data previously classified as positive. This can facilitate learning. For instance, in the same example, the second model does not need to learn intensity features to discriminate more obvious negatives and so, it might be trained to learn other patterns like shape or texture, which help discriminate between epithelial tissue and lesions. Due to this, the learning problem might be simplified by breaking down classification this way.

This simple explanation helps to understand intuitively why a cascade approach might work in this case. More formally, if we think about the distribution of unbalanced data in the feature space, harder negative examples should be close to many positive cases, while easier ones should be far. Additionally, the majority of negative examples should be easy. The cascade approach works by first, learning a boundary that separates easy negatives from positives and harder negatives, and then, learning a second boundary, to distinguish between these last two. What we are saying is that, learning the second boundary becomes easier if we do not pay attention to the easy negative samples. This process is illustrated on fig. 5.3a.

An obvious advantage from this approach is that we can train models with artificially balanced data, by oversampling positive cases, while reducing the bias towards the minority class.

Additionally, although not used in this work, this also allows us to use lower capacity models to discard obvious false positives, which are faster, and run more complex models only on difficult examples. This can significantly reduce the time required to screen new images, at test time.

A schematic view of this method is shown in fig. 5.3b.

Chapter 6

Experimental Work

In this section, the carried out experimental work is presented. First, an overview of the data used in the experiments is shown, followed by implementation details and performance assessment methods. From that point on, experimental results are shown and discussed, for each of the proposed methodologies. The final performance of the breast lesion detection system is discussed in the last section.

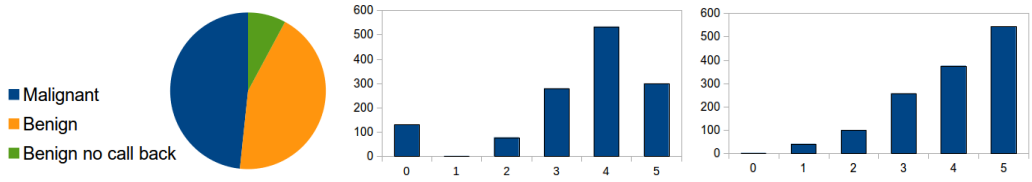
6.1 CBIS-DDSM Dataset

To develop, train and benchmark the performance of proposed methods, data in the form of mammography images must be used. As seen, much research on this field utilizes unpublished data, which impairs fair comparison of methods as well as reproducibility. To avoid this, the publicly available Curated Breast Imaging Subset of DDSM (CBIS-DDSM) (Lee et al., 2016) was used. This is an updated and standardized version of the Digital Database for Screening Mammography (DDSM) (Heath et al., 1998), created recently. DDSM contains thousands of scanned film of mammography studies. Some of these were used to build the CBIS version, which introduced some key modifications to facilitate the use of data:

1. Images with annotations of lesions that could not be seen by a trained mammographer were removed from the dataset.
2. Images, previously in lossless JPEG format, were decompressed and saved in the DICOM format, the standard in medical imaging.
3. Manual segmentation of lesions was refined.

Currently, this dataset is integrated in The Cancer Imaging Archive (TCIA) (Clark et al., 2013), a large collection of images of many cancer types and imaging techniques, aimed at facilitating research.

It is divided in two parts, one with masses and the other with microcalcifications. In this work, only the first was used. To our knowledge there are no published works on using CBIS-DDSM. A total of 691 patients, 1231 images and 1318 lesions were available and used.



(a) Pathology (b) BIRADS (c) Subtlety

Figure 6.1: Distribution of lesions in the dataset according to ground truth information. BIRADS 0 stands for incomplete assessment. The subtlety level of zero is the harder to classify, while 5 is the easiest. These levels were grouped in hard $[0-3]$, average $[4]$ and easy $[5]$.

Each lesion had an associated detailed segmentation, shape and margins descriptions and a BIRADS score, which quantifies the suspicion level of that lesion. Additionally, a subtlety level and pathology assessment were present. The distribution among these categories is shown in fig. 6.1.

6.2 Implementation

All software was implemented in the python programming language (Rossum, 1995). This is the preferred tool when it comes to deep learning development for research. Python has some important features that facilitate development, which include the easy installation and use of well documented open source packages.

For image processing methods, we used numpy (van der Walt et al., 2011) and scikit-image (van der Walt et al., 2014) packages, which are standard for these applications. Implementation of deep learning models was done in tensorflow (Abadi et al., 2015). This package, initially developed at Google for research, is a flexible tool with many built-in functions for implementing machine learning methods, including convolutional neural networks. Additionally, it provides integration with two NVIDIA libraries, CUDA (Nickolls et al., 2008) and cuDNN (Chetlur et al., 2014). These are highly efficient software libraries for parallel numerical computation, using GPUs. As seen previously, nowadays this is a necessity in this field of research.

Experiments were carried out in an Ubuntu machine with an Intel quad core i7 processor @ 4.00 GHz, and an NVIDIA GTX 750 Ti with 2Gb memory GPU.

6.3 Evaluation metrics

In the context of this experimental work, datasets only contained two classes: positives (malignant and benign lesions) and negatives (normal tissue). After running a model for new examples we get a value for each, that expresses the probability of that region being a lesion. Based on this we can create a ROC curve. Each point on this curve is defined by the sensitivity and false positive rate obtained, by assigning all examples with predictions above a threshold to positive and the others to negative. They can be computed as follows:

$$sensitivity = \frac{TP}{TP+FN} \tag{6.1}$$

$$FPR = \frac{FP}{FP+FN} \tag{6.2}$$

Where TP and FN are positive correctly and incorrectly classified, respectively, and FP are incorrectly classified negatives. A very common metric to compare algorithms is the area under the ROC curve (AUC). This is illustrated in fig. 6.2. Throughout experiments, this metric was used for comparison between methods. Additionally, in the last section, we compare results using the FROC curve. The same principle applies, but instead of the false positive rate in the horizontal axis, the average number of false positives per image is used.

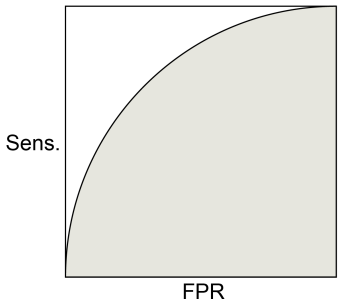


Figure 6.2: Diagram of a ROC curve. The are under the curve is shown in gray.

6.4 Preprocessing and Region of Interest Segmentation

The first module of the system is used to select a region where lesions might occur, removing the black portions around the breast. This is useful as it discards many obvious false positives. When using a learning strategy, like the one proposed, we need to avoid spending training resources on samples we could have easily discarded.

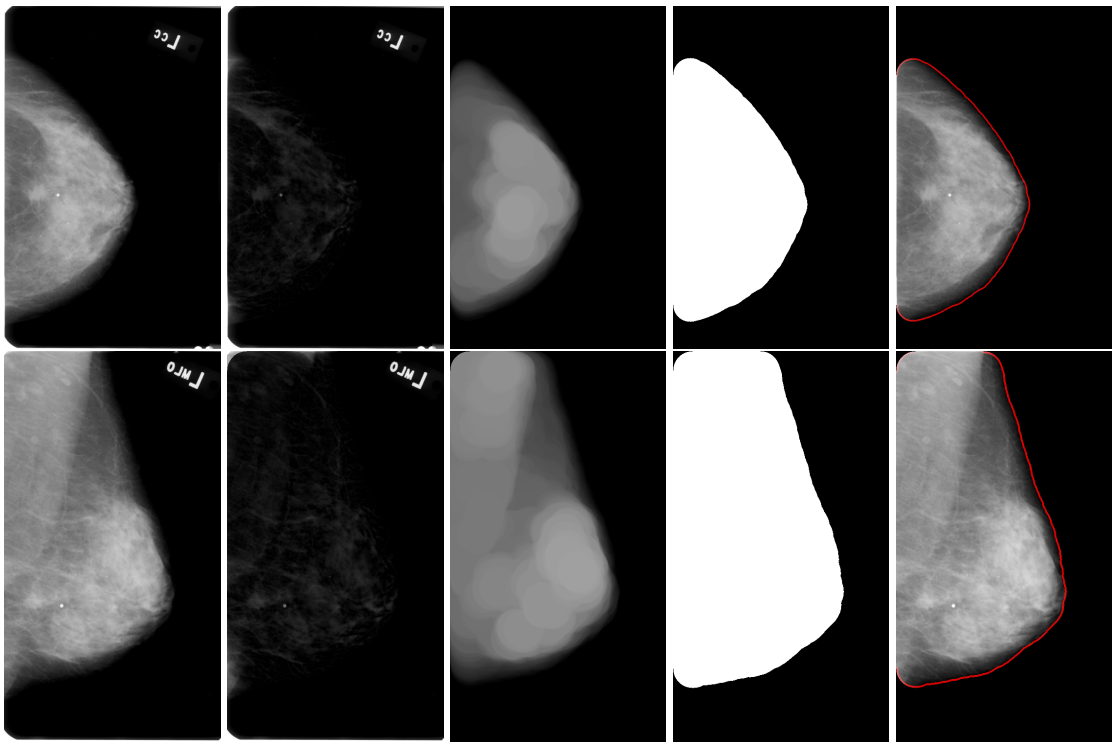
Additionally, this module also removes artifacts, characterized by high intensities values, which could be mistaken for lesions. These artifacts appear outside the breast boundaries, and so, are easy to discard with a global view of the image, but hard with just local information.

The used algorithm is similar to Pereira et al. (2014) with some minor modifications. First, for a new image, histogram stretching is performed, mapping pixel intensities to the interval $[0,255]$. After this, the image is resized and padded with zeros. Depending on the experiment, different resize factors were used. The padding operation ensures the whole image, including the edges, can be filtered by a structuring element using morphology operations.

After padding, a white top hat operation is performed, with a disk shaped structuring element. The result, containing only non important regions, is subtracted to the original image. This image is then thresholded to obtain a binary mask. If this mask contains more than one object, the biggest

is selected. The radius used for the structuring element was $240r$ pixels, with r the resize factor. This size was empirically found and provided good results for a random subset of images.

The intermediate results of these operations are shown in fig. 6.3. As we can see, the segmented region is well adjusted to the breast boundary and avoids well the artifacts on the CBIS-DDSM dataset. Segmentation boundaries in the edge of the image look round, due to the combination of morphological filtering with the zero padding operation. This could be avoided by using reflective padding, but it does not influence further methods, due to the absence of lesions in those regions.



(a) Initial image (b) Top-hat filtering (c) subtracion: a-b (d) Mask (e) Final result

Figure 6.3: Results of breast segmentation for two examples using the proposed method.

6.5 Preliminary experiments

Initially, we tested common machine learning methods applied using the classification problem formulation in chapter 3. As such, we evaluated the impact on performance of data augmentation, dropout, batch normalization and different input sizes. In the following sections we look to improve on this performance, by using the proposed methods.

6.5.1 Dataset Construction

For training, small image patches were taken from images according to two rules, one for negative examples and one for positive ones. First, patients are split evenly between 5 folds, each with

approximately 138. Then, we follow the methodology described in the previous section, with each image being scaled to $1/24$ of the original size. The same scale factor was used for all images due to the lack of information about the resolution of each exam. Negative patches were obtained by grid sampling the images and taking all patches completely inside the breast, without any pixel belonging to a lesion. Positives were obtained by taking one centered patch per lesion only. This choice was made because the ideal result would be a maximum response in the center of each lesion with all other pixels being zero.

In fig. 6.5 we can see the distribution of bounding box size for all the lesions in the dataset. Based on this we selected patch dimensions. The $(25,25)$ input contains completely the majority of lesions. The $(35,35)$ input also catches some surrounding tissue. Fig. 6.4 shows examples of positive patches with these two input sizes. Sizes, were selected odd, so that the central pixel could be the center of a lesion.

We then create five data splits, by selecting three folds for training, one for validation and one for testing. We guarantied that each fold was used the same number of times in each of these tasks. This procedure was done twice, one time for $(25,25)$ inputs and another for $(35,35)$. The final datasets contained almost 1000 negative examples per positive for the first input size and 300 for the second.

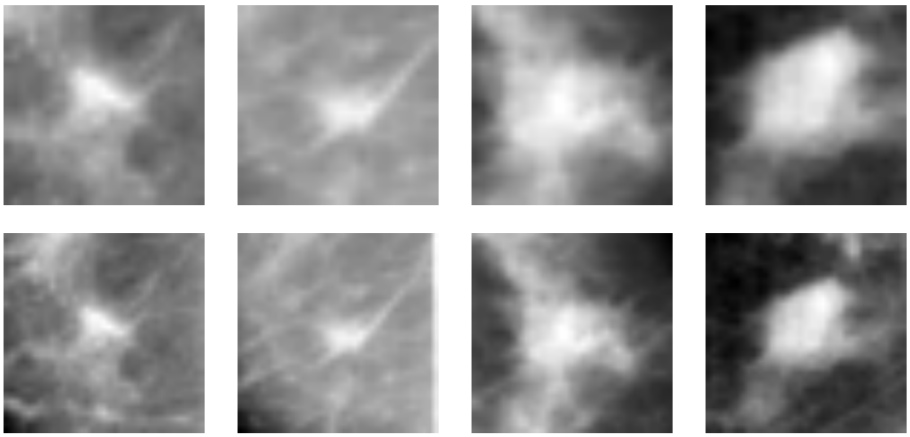


Figure 6.4: Examples of patches in the datasets. On top $(25,25)$, and on bottom $(35,35)$.

6.5.2 Model design

When it comes to designing a model, in deep learning, authors usually can follow three strategies: utilizing other authors proposed architectures, designing a custom model based on intuition of what might work and testing multiple models. Due to the computational requirements of training and benchmarking performance of multiple models, the last option was discarded. This allowed more time to perform experimentation with different methods. The strategy in this work was to make custom networks based on the proposed VGG models (Simonyan and Zisserman, 2015) and intuition. These, are simple, easy to implement and follow the conventional architecture of ConvNets.

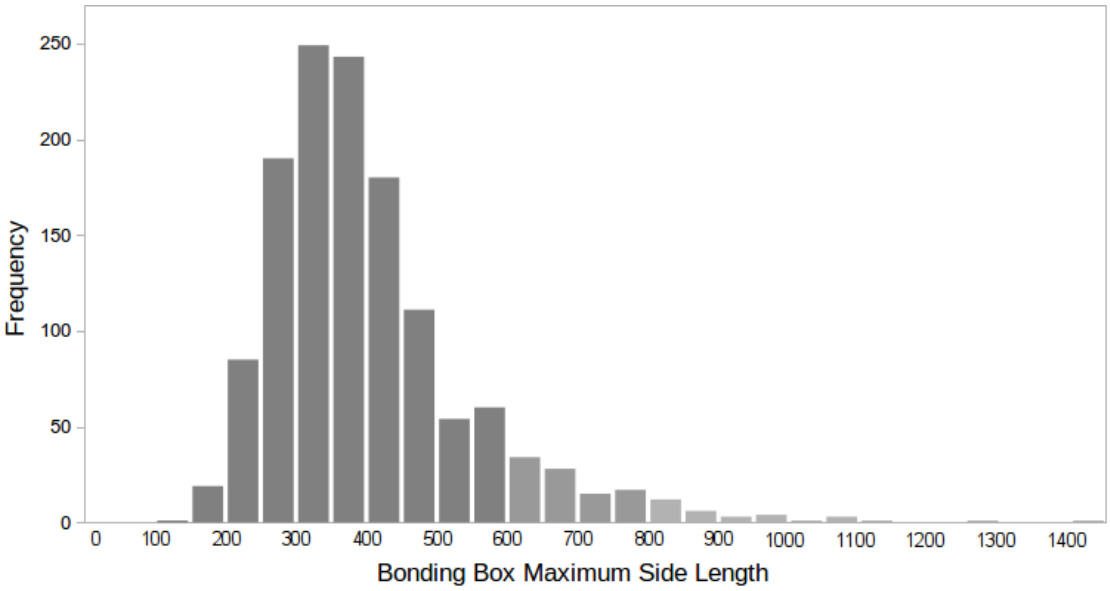


Figure 6.5: Side size of the masses bounding boxes in CBIS-DDSM

There are more recent and better performing models in the literature. However, most of them are usually tested in big natural images, very different from the data used here. Additionally, some architectures, like Inception and ResNet require input padding in convolutional layers, which would disallow testing whole images through the method described in section 5.1.

The final architecture of the model is depicted in table 6.1. As seen, all convolutional filters have size $(3,3)$, and they are stacked on top of each other to obtain $(5,5)$ receptive fields, effectively reducing the number of weights, and allowing the introduction of an activation layer in between. Due to the considerably smaller input size used in experiments, we have fewer pooling operations. Compared to the VGG model, the first convolutional layers have fewer filters. As we are dealing with single channel images, there are less possible patterns for the first layer to detect and so, only 32 filters were used. For the fully connected layers the proposed model only has 512 neurons, compared to 4096. VGG was trained on ImageNet, to distinguish between 1000 classes, probably requiring a bigger feature space to encode that information, when compared to our two class detection problem. The total amount of parameters depends on the input patch size. Two possibilities were evaluated $(25,25)$ and $(35,35)$. In these cases the total number of parameters optimized is around 2 and 4 million, respectively.

6.5.3 Training the model

For each input size, 5 versions of each model were trained to study the impact of common regularization methods used in deep learning:

1. The model as described in table 6.1.

2. Added dropout after ReLU activation in dense layers with a keeping probability of $\sigma = 0.5$.

Table 6.1: Description of the model architecture used for the first experiments. All Convolutional and Dense layers are followed by a ReLU activation, here omitted for simplicity. The first Dense layer is described as having a filter size bigger than one. This is done for consistency in parameter calculation, and also to emphasize the idea that these layers are in essence convolutional layers with output size of 1×1. The output layer has a sigmoid activation function.

Layer	Side Size	nº Filters	Filter Size	Parameters
1. Input	25 / 35	1	1	-
2. Convolutional	23 / 33	32	3	320
3. Convolutional	21 / 31	32	3	9k
4. Max-Pooling	10 / 15	32	1	-
5. Convolutional	8 / 13	256	3	74k
6. Convolutional	6 / 11	256	3	590k
7. Max-Pooling	3 / 5	256	1	-
8. Dense	1	512	3 / 5	1.2m | 3.3m
9. Dense	1	512	1	262k
10. Output	1	1	1	513

3. Added batch normalization after every convolutional and dense layer. The use of batch normalization after activation layers was tested, in an initial phase, and no significant difference was found. We stick with the recommendation in the original paper.

4. Combination of dropout and batch normalization with $\sigma = 0.5$

5. Combination of dropout and batch normalization with $\sigma = 0.8$. Ioffe and Szegedy (2015) claim that better results are obtained with batch normalization by choosing a high value of σ, or even disregarding dropout completely.

Additionally, we perform the same experiments with data augmentation. For this, every patch at training time has an equal probability of being rotated by $90k$ degrees, with $k \in \{0,1,2,3\}$, and horizontally mirrored. Also, patches were normalized by subtracting the mean and dividing by the standard deviation of the training dataset. This is a common practice in many machine learning applications, including deep learning models.

In initial experiments, it was noted that, by using Adam optimization algorithm, the model took much fewer iterations to train until it converged, when compared to standard mini-batch gradient descent. Additionally, the final solution was better. Due to this Adam was used in all experiments, with constants β_1 and β_2 set to 0.9 and 0.999, respectively. The learning rate was set to 0.001 and dropped by a factor of 10 at the midpoint of the training process. Weight decay was set to 0.0001.

Models were trained for 80 epochs, each corresponding to 200 balanced batches of 32 positive and 32 negative samples. Normally, an epoch is the number of iterations necessary for the model to see all data in the training set. In our experiments, this was only the case when using data augmentation, where 8 times more data was available. In the case of models with batch normalization, the optimization was much faster. To adjust for this, in each epoch these models only trained for $1/5$ of the iterations. When not using data augmentation each epoch corresponds to approximately 8 passages through data. After each pass, the model loss is evaluated in a validation set, with all

Table 6.2: AUC values for validation and test sets in CBIS-DDSM using 5-fold cross validation $average \pm 2 \times std$

		25 x 25		35 x 35	
Data aug	Model	Val	Test	Val	Test
False	Simple	0.958 ± 0.023	0.958 ± 0.029	0.982 ± 0.009	0.981 ± 0.012
	Dropout 0.5	0.963 ± 0.010	0.959 ± 0.016	0.977 ± 0.025	0.979 ± 0.015
	BN Simple	0.971 ± 0.017	0.966 ± 0.018	0.985 ± 0.008	0.981 ± 0.016
	BN Drop 0.8	0.973 ± 0.019	0.967 ± 0.032	0.985 ± 0.009	0.980 ± 0.012
	BN Drop 0.5	0.975 ± 0.021	0.969 ± 0.018	0.987 ± 0.011	0.984 ± 0.011
True	Simple	0.979 ± 0.008	0.975 ± 0.013	0.987 ± 0.010	0.985 ± 0.012
	Dropout 0.5	0.977 ± 0.015	0.975 ± 0.013	0.988 ± 0.008	0.984 ± 0.008
	BN Simple	0.974 ± 0.020	0.976 ± 0.012	0.991 ± 0.006	0.989 ± 0.013
	BN Drop 0.8	0.976 ± 0.015	0.977 ± 0.009	0.990 ± 0.008	0.987 ± 0.006
	BN Drop 0.5	0.977 ± 0.021	0.976 ± 0.010	0.990 ± 0.006	0.986 ± 0.023

positive and 10000 negative examples, and saved. The whole validation set was not used, to avoid spending too much time testing. After training, the model with lower validation loss is selected for further evaluation.

6.5.4 Results and Discussion

Each individual training procedure took around 20-40 minutes to complete, depending on the size of the input and number of iterations per epoch. In total these experiments run for a little more than a day. Results are shown in table 6.2.

Globally, high AUCs were obtained, in part due to the high number of "easy" negatives in this first dataset. Although only portions of the breast were considered, some of these regions have low intensity and contrast, making them more obvious.

Additionally, high standard deviations were obtained, which are in part related to the difference of difficulty between folds. Some patients are harder to diagnose due to characteristics like breast density, while others present more obvious lesions. Because each split contains 138 patients, these differences can affect results. If we look at fig. 6.9, which presents results from another experience, we see that, performance varies more across splits, than between different models.

When we compare the obtained AUCs, we can see that, for all experiments the $(35, 35)$ models perform better. This result, suggests that surrounding context of a lesion is important for its correct classification. This is consistent with results presented by Lévy and Jain (2016). Bigger input sides were briefly tested and did not seem to improve performance, while requiring higher training times.

Data augmentation, as expected, also improved consistently the performance of the models. This is common in deep learning, due to the high number of examples required to train these models. In terms of dropout no improvements were found, which is uncommon. One plausible justification for this is that because we are working with small models, the problem of overfitting might not be as relevant, and so the effect of dropout is minimal. Batch normalization had a

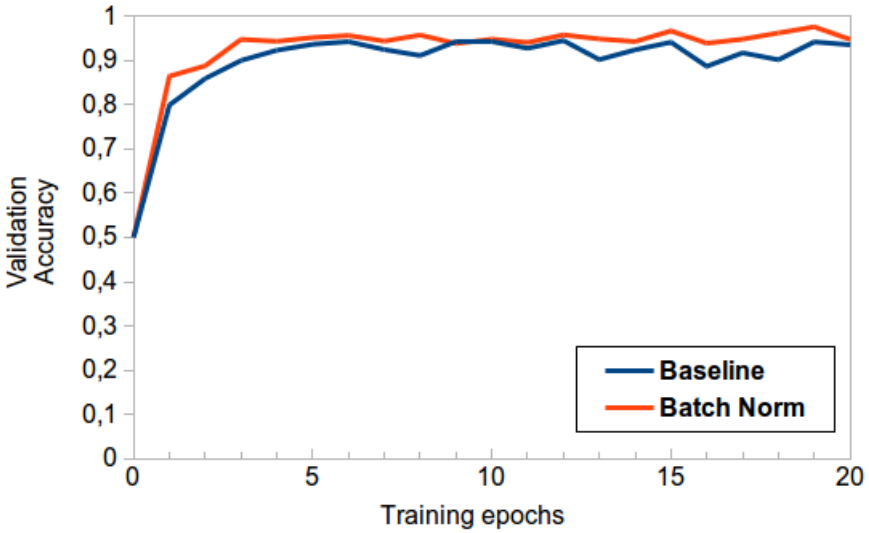


Figure 6.6: Effect of batch normalization. After one epoch a higher validation accuracy is obtained than the baseline after two epochs. Note that, for the batch normalization model each epoch had $1/5$ of the data. Additionally, maximum accuracy is obtained very soon.

positive impact in most experiments. Additionally, as seen in fig. 6.6, this technique increases training speed significantly. In future experiments, batch normalization was included whenever possible.

One interesting aspect found in these preliminary experiments is that the model only overfits the positive cases. This is shown in fig. 6.7. The training loss is decreasing for both classes, while the validation loss is only decreasing for negatives. Additionally, the training loss for positives decreases more rapidly than for negatives.

The reason for this is that models are seeing positive examples multiple times, whereas the number of negatives is enough for many epochs. This could motivate the development of strategies to regularize learning for one class only.

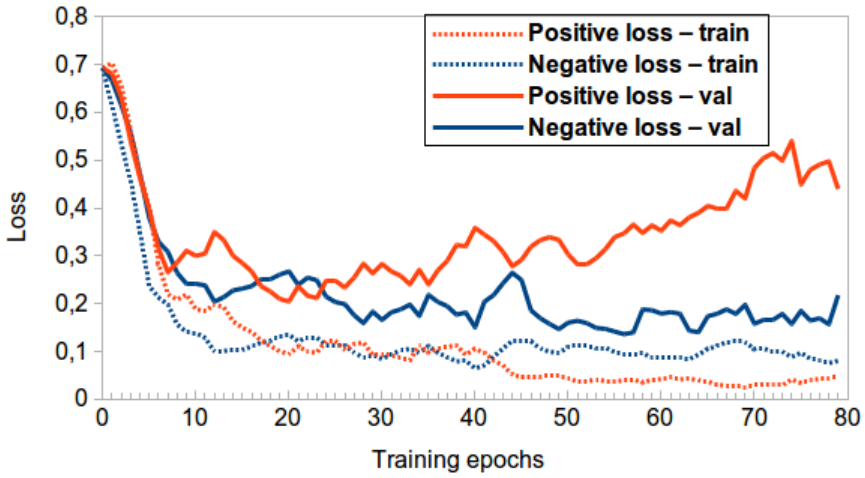


Figure 6.7: Train and validation loss for positive and negative samples. The model is overfitting to the positive data.

6.6 Convolutions with Rotated Filters

To evaluate the effect of rotated filters as a regularization method three experiments were considered. We address each individually.

For the first experiment we used the BN Drop model with $\sigma = 0.5$ as a baseline. The $(35, 35)$ input size was used. As seen in section 5.2, the rotated filters version can substitute convolutional layers and the first dense layer of a model. Due to the fact that this version has a fewer weights, it works as a regularization technique. In principle, we impose restrictions in the model to increase the universality of the features learned. This is based on the field knowledge that lesions do not have orientation, and so, can appear with any rotation in the mammograms. Due to this, individual features should also appear in any orientation.

In this experiment, four models were designed by changing conventional layers to this new version, and are presented in table 6.3.

Table 6.3: Models considered for the first Convolution with Rotated Filters experiment

Model	Layers changed	Removed parameters	% of parameters removed
Rot1	[2]	240	0.0%
Rot2	[2,3]	7k	0.2%
Rot3	[2,3,5]	62.5k	1.5%
Rot4	[2,3,5,6]	504.8k	11.9%

The same exact data and training methodology, described in section 6.5, was used for this experiment. Also, batch normalization and dropout were used.

Training and inference processes were significantly slower than the baseline model in each iteration. The reason behind this is related to implementation. In each forward pass through the

Table 6.4: AUCs obtained for models with rotated filters. Results are presented as $average \pm 2 \times std$ over 5-fold cross validation.

	Normal data		Augmented data	
Model	Val	Test	Val	Test
Baseline	0.987 ± 0.011	0.984 ± 0.011	0.990 ± 0.006	0.986 ± 0.023
1 rot layer	0.982 ± 0.019	0.983 ± 0.015	0.992 ± 0.004	0.992 ± 0.005
2 rot layer	0.985 ± 0.014	0.986 ± 0.008	0.988 ± 0.009	0.988 ± 0.012
3 rot layer	0.982 ± 0.014	0.980 ± 0.014	0.988 ± 0.006	0.989 ± 0.002
4 rot layer	0.985 ± 0.010	0.983 ± 0.008	0.988 ± 0.008	0.992 ± 0.007

model, filters were rotated, which took a small amount of time. In the worst case, for model Rot4, each training iteration took approximately 50% more time.

The validation and test AUCs are shown in table 6.4. The performance of these models increases with data augmentation. As explained before, layers with rotated filters provide feature extraction symmetry not rotation invariance, and so, this behavior is expected. The obtained AUCs are very close to the baseline model and did not provide any clear advantage. Several hypothesis can justify this:

1. The number of parameters removed is small compared to the 4 million in the base line model. At best, approximately 11.9% were removed in Rot4.

2. The amount of data available was sufficient to learn filter symmetry, and so, the effect of this regularization technique is minimal. Additionally the AUCs are very close to 1.0 making a significant increase in performance harder to obtain.

3. The imposed restrictions make the model harder to optimize. The fact that the gradient of a parameter is computed using four filters, instead of just one could decrease the ability of the model to converge to a good solution.

4. This regularization technique is ineffective. The main justification for this is that, some features extracted from the network might have no orientation. For instance, one filter could be capturing round objects and, in this case, the rotated convolutions version will replicate that feature four times, with no gain. Further, the introduction of redundant filters in the network might remove space for other relevant features.

5. L2 and dropout regularizations are already "strong" enough. This is less likely but possible. Although dropout and L2 have different effects than the proposed method they might be limiting the ability of the baseline model to overfit the training data. Batch normalization also promotes regularization.

Due to the previous results being inconclusive, two additional experiments were designed, using datasets created for the purpose of Cascade evaluation, explained in section 6.8.4. Two input sizes were considered $(72, 72)$ and $(36, 36)$.

Table 6.5: Convolutional Neural Network architecture for the second experiment of rotated filters.

Layer	Side Size	N^o Filters	Filter Size	Parameters
1. Input	72	1	1	-
2. Convolutional	68	16	5	416
3. Max-Pooling	34	16	1	-
4. Convolutional	32	64	3	9.2k
5. Max-Pooling	16	64	1	-
6. Convolutional	10	128	7	402k
7. Max-Pooling	5	128	1	-
9. Dense	1	256	5	819k
10. Dense	1	256	1	66k
11. Output	1	1	1	257

In terms of architecture, the smaller input sized model is described in subsection 6.8.2, and the larger is depicted in table 6.5. In this model, filters bigger than $(3,3)$ were used to increase the number of parameters in the convolutional layers and reduce the computational cost of the experiment. Dropout was disregarded. Differently to the previous section, the best model was chosen based on validation AUC instead of loss.

The first experiment, with $(72,72)$ inputs was performed only on one split due to the computational cost. The model was not optimized until it fitted perfectly the training data due to the time taken for each training iteration. Also, the validation AUC started decreasing for the later stages of the training process. We considered four scenarios: 1) baseline model, 2) introducing rotated filters in all convolutional layers (Conv All), 3) introducing rotated filters only on the first dense layer (Dense) and 4) combining the second and the third scenarios (Conv+Dense).

In fig. 6.8 we show the maximum AUC obtained in training and validation data. Again, the differences between results are very small. All models with rotated filters perform worse on the training data, which is to be expected, due to the restrictions imposed in the network. The Dense 1 model exhibits a typical behavior for regularization methods. Although its performance is smaller for training examples, the validation AUC is better. This could be the result of learning more general features. The same happens, at a lower scale, with the ConvAll model. The Conv+Dense model seems to perform worse in both sets.

Although the difference is really small, these results suggests models with rotated filters are harder fit to the training data. However, this increase in difficulty does not always translate to more generic features. For a better assessment on the effects of rotated filters in the performance of the network, the method should be tested in different settings that enhance the effects of regularization. For instance, one could try the application of this method on smaller datasets, with higher capacity models.

For the third experiment, rotated filters were only applied to the first dense layer. The obtained results, shown in fig. 6.9, were similar to the ones seen previously in three of the five splits. In the other two, the model performed equally well in one split and worse in the other, in terms of validation AUC. In terms of training data, the model performed always worse.

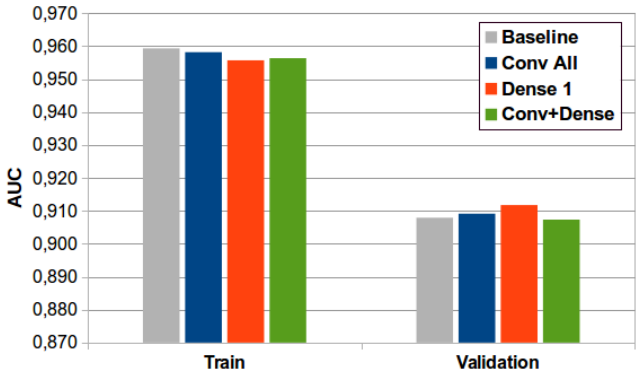


Figure 6.8: Train and Validation AUCs for $(76, 76)$ sized data for multiple models with rotated filters.

The combination of all these experiments suggest a very small gain when using rotated filters on the first dense layer. Note that, this variation reduces the number of parameters by approximately 21% for the $(36, 36)$ model and 17% for the $(72, 72)$ model. As such, the hypothesis that the reduction of weights was insignificant should be discarded. Also, dropout was removed, which reduces the generalization capacity of the baseline model.

In the end, these results suggest that, either this method is ineffective in regularizing Convolutional Neural Networks or for this particular problem the amount of data available and low capacity of the models diminish the need for regularization methods.

As argued previously, further experiments using different data and higher capacity models could be interesting to determine the effects of rotated filters in the model. From our results, when applied to the first dense layer, this method increased results only slightly.

The fact that some features learned by the network might have no orientation, can lead to the model learning redundant information. To test this hypothesis, one possible experiment would be to utilize this version on a fraction of the filters only.

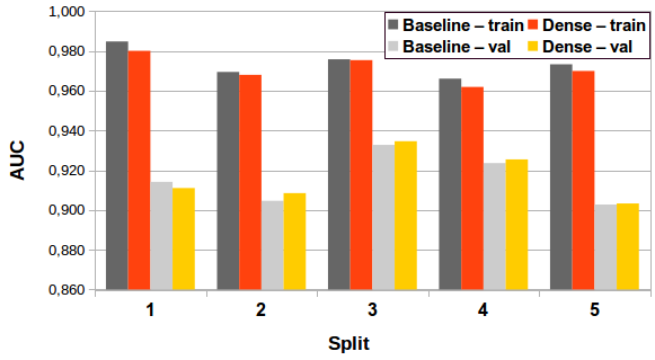


Figure 6.9: AUCs for multiple splits in $(36, 36)$ sized data using the Baseline and Dense1 models.

6.7 Rank learning

To evaluate the effect of rank learning applied to Convolutional Neural Networks, the $(36, 36)$ dataset, described in 6.8.1 was used. The baseline model is the one described in 6.8.2 without batch normalization. In early experiments we verified that this technique impeded optimization for rank learning architectures. Batch normalization works by adjusting the output's distribution of one layer for each mini-batch during optimization. Because in the case of rank learning we have two streams of information, the normalization performed is different for each. We also tried to apply the test version of batch normalization on one stream, during training, while performing normal batch normalization on the other. The same effect was observed. Based on this, we decided to remove it completely.

We compared three approaches to the baseline model. First, the rank learning architecture, described in 5.3.1 is trained from scratch. Second, the same model is used, but parameter initialization is done with the weights learned by the baseline model. Parameters after the subtraction layer are learned from scratch, as these were not present in the original architecture. Third, a linear classifier is optimized in a rank learning setting, based on the last dense layer's activations of the baseline model as described in 5.3.2.

In early experiments, it was noted that the rank learning architecture required smaller learning rates. As such, this parameter was halved, and the number of batches increased to 400. Apart from this, all other parameters were equal to section's 6.8 experiments. Data augmentation was performed. Table 6.6 shows the obtained AUCs over 5 split cross validation.

Table 6.6: AUCs obtained for rank learning experiments. Results are presented as $average \pm 2 \times std$ over 5-fold cross validation.

Model	Val	Test
Baseline	$0,982 \pm 0.010$	$0,983 \pm 0.005$
Sctratch	$0,981 \pm 0.008$	$0,980 \pm 0.009$
Initialized	$0,982 \pm 0.010$	$0,981 \pm 0.009$
Linear Class.	$0,982 \pm 0.008$	$0,982 \pm 0.008$

As seen, the followed rank learning approaches perform equally to the baseline model. For the linear classifier approach this was expected. As seen in previous architectures, the number of parameters in the output layer is only a very small fraction of the number present in the model. As such, adding a classifier on top of a feature representation and training on the same data should not increase results significantly.

For the other methodologies results show there is no significant improvement over the baseline. The hypothesis that we could increase the number of examples, by feeding tuples of data to the network, as a data augmentation scheme does not hold. Additionally, these methodologies require significantly more time to train, with the exception of the linear classifier which runs in a few seconds. The fact that, during optimization we have two equal models stored in the GPU memory doubles the amount of this resource needed.

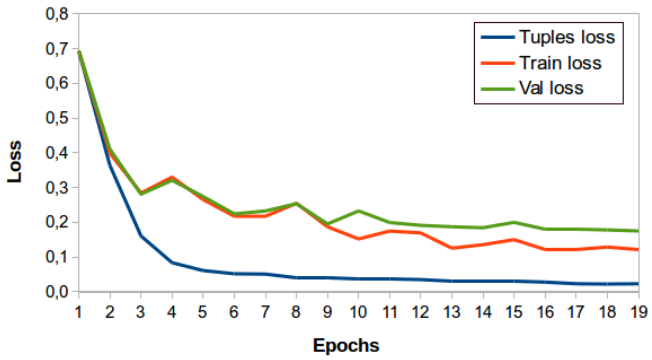


Figure 6.10: Losses for the rank learning approach trained from scratch on the first 20 iterations.

These results can be explained by the fact that the rank learning methodology was not able to increase generalization. In particular, the model is fitting better to the training data than to the validation, when running individual examples. This can be seen in fig. 6.10, which shows the loss for training tuples as well as for train and validation single inputs, for the first 20 epochs. We can see that the model is quickly minimizing the loss for training tuples. However, for single training and validation inputs, the loss decreases much slower. Additionally, we can see that after just 10 epochs, the model is already fitting training data better than validation. This happens with both the training from scratch and pre-initialized models. Even though, in each batch we are feeding different combinations of examples in the form of tuples, the network can still learn features specific to the training data.

In the end, rank learning in Convolutional Neural Networks does not seem to be an effective strategy to avoid overfitting. From these results we showed it performed equally, while taking more computational resources to optimize. It can still be an alternative strategy in cases of unbalanced data, similarly to oversampling the minority class.

6.8 Cascade learning

As previously seen, the cascade approach consists of sequential models that will eliminate the most obvious normal regions, while keeping as much true lesions as possible. In the following subsections we will describe the followed training methodology and report the obtained results for each model. In the end, results for the proposed mammography lesion detection system are presented.

6.8.1 First Dataset Construction

For the first dataset, the steps described in 6.5.1 were followed with small differences. First, patches were taken from all regions centered in the breast tissue instead of completely contained in it. Because we will be using the model to screen the image, regions close to the breast boundary can potential present lesions. Another key difference was the selected patch size of $(36, 36)$. This

was done to avoid odd numbers before max-pooling layers. Otherwise, the numerical results of screening the whole image would be different than individual patch evaluation, due to the max-pooling layers, similarly to convolutional ones, disregarding edge neurons when the filter is not completely contained in the previous feature map. The ratio between negatives and positive examples was approximately 300 for this dataset.

6.8.2 First Cascade Architecture and Training

The first model used is very similar to the one presented on table 6.5.3. The main difference is in the number of filters in layers 5 and 6, which was 128, and layers 8 and 9, which was 256. We verified no significant drop in performance by reducing these parameters. Additionally, it slightly reduced training times. Due to the fact that we are using $(36,36)$ sized inputs, the filter size of the first dense layer is 6, instead of 5.

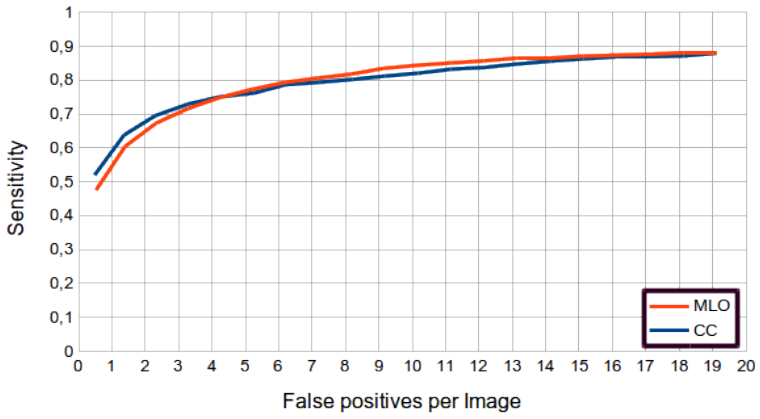
Training was performed similarly, but reducing the number of iterations to 40 and increasing the number of batches per iteration to 200. This modification reduces the time spent testing on the validation set and increases time spent training. Dropout was disregarded and batch normalization was used.

6.8.3 Results and Discussion

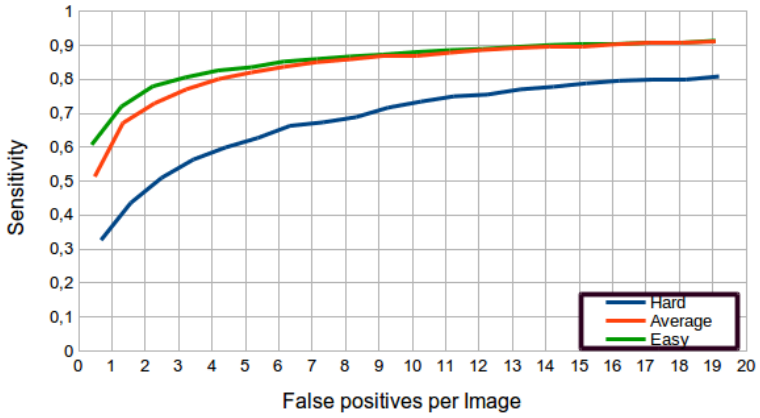
The AUC obtained for the first model in the validation data was $0,987 \pm 0,006$. This value is very similar to the ones achieved in the early experiments. After this, the model was used to screen whole images in the test set, yielding a probability map. This operation took less than on second per image. Instead of using the method explained in section 5.1, an adaptation was done, to decrease the amount of memory needed. As previously explained, the network can be transformed to run the whole image, instead of fixed sized patches. This requires more GPU memory, which was not available. To solve this problem, a region of $(100,100)$ is run each time and outputs concatenated to obtain a final result. This compromise significantly reduces the amount of redundant computations, while requiring small amounts of memory. Examples of these probability maps are shown in figure 6.12. From each map, detections are obtained iteratively, by first taking the coordinates of the maximum point, and then assigning a small region around the detection to zero.

The results obtained for whole images is depicted in fig. 6.11. As we can see, for both views, a sensitivity of 80% is only achieved with 8 false positives per image. This should be expected due to the fact that, by oversampling the positive cases during the training, we are effectively biasing the model towards this class. When we look at results for different subtleties, we can see a very significant decrease in terms of model performance for harder to detect lesions. This behavior is undesirable because these lesions are also the ones specialist can easily oversee. In terms of malignancy, this framework is able to detect 80% of positive cases with only 4 false positives per image. Although malignant masses are the most well represented in the dataset, the difference between malignant and benign masses is small. This suggests that malignant lesions are

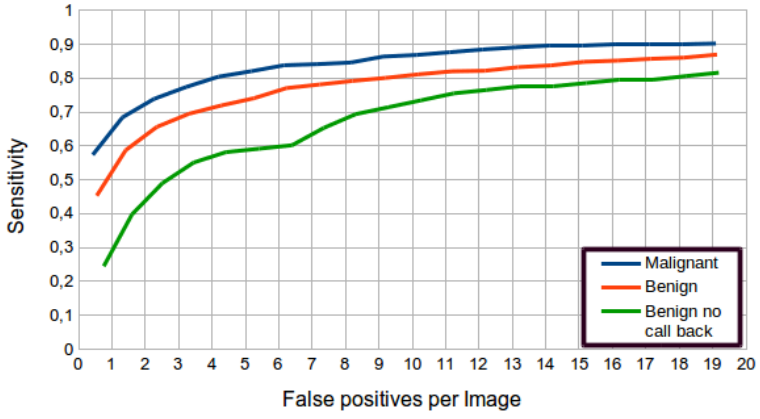
easier to spot, with Convolutional Neural Networks, than benign ones. This behavior is extremely interesting to CAD systems. A justification for this is the fact that malignant lesions are often bigger in size and contrast more with the surrounding tissue. Between MLO and CC views the performance is very similar with no view presenting a clear advantage for lesion detection using this methodology.



(a) View



(b) Subtlety



(c) Malignancy

Figure 6.11: Detection results obtained by the first cascade model.

6.8.4 Second Dataset Construction and Training

The second model is trained only on data misclassified by the first one. For this, utilizing the previously obtained detections, a new dataset is constructed. As seen before the images were scaled by a factor of $1/24$. In this experiment, a resize factor of $1/12$ was also tried. To obtain the detections for these images, the probability map was scaled by 2 by using bilinear interpolation. We took at most 100 patches, centered in the strongest detections, of sizes $(36,36)$ and $(74,74)$ depending on the resize factor. This yielded a dataset with approximately 8 negative examples per positive. Note that some detections fall on the same mass. Additionally, detections with probability smaller than 0.5 were not considered.

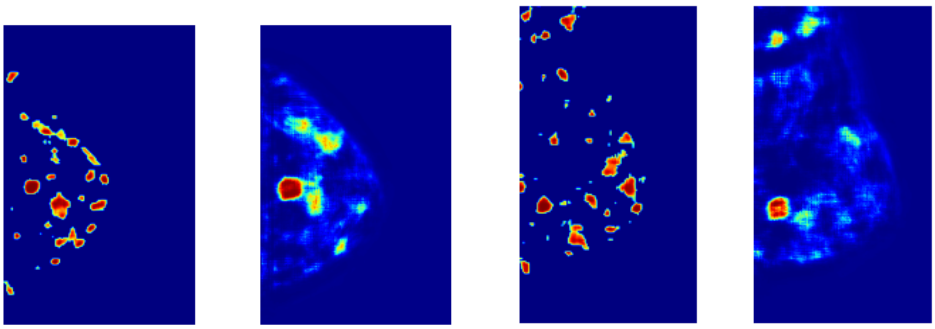
For the second cascade model two architectures were tested depending on the patch size. For $(36,36)$ inputs, training was done exactly the same as the previous model. For $(74,74)$, the architecture depicted at table 6.7 was used. Additionally, dropout was added after dense layers with $\sigma = 0.8$. Due to the model having a higher capacity to overfit the training data, more regularization should be added. For each iteration, the $(74,74)$ model trained on 1000 mini-batches, with size 32. This reduction was done to decrease the GPU memory requirements of the model. All remaining parameters were kept the same.

Table 6.7: $(76,76)$ input sized model for the Cascade 2 experiments.

Layer	Side Size	N^o Filters	Filter Size	Parameters
1. Input	74	1	1	-
2. Convolutional	72	16	3	160
3. Max-Pooling	36	16	1	-
4. Convolutional	34	64	3	9280
5. Convolutional	32	64	3	36928
6. Max-Pooling	16	64	1	-
7. Convolutional	14	128	3	73856
8. Convolutional	12	128	3	147584
9. Convolutional	10	128	3	147584
10 Max-Pooling	5	128	1	-
11. Dense	1	256	5	819456
12. Dense	1	256	1	65792
13. Output	1	1	1	257

6.8.5 Results and Discussion of Cascade Approach

The maximum validation AUC obtained during training was $0,918 \pm 0,013$ and $0,916 \pm 0,025$ for input sizes of $(36,36)$ and $(74,74)$, respectively. This is significantly lower than the one obtained in previous experiments. Considering the second dataset only uses false positives of the first model as negative samples, this classification task should in principle be more difficult. Additionally, there is significantly less data in this second dataset. Another problem with this methodology is that, when sampling the training set, we are using a model trained on that data. As such, the sampling distribution could be different from the test data. For instance, a model that overfit the



(a) Cascade 1 example 1 (b) Cascade 2 example 1 (c) Cascade 1 example 2 (d) Cascade 2 example 2

Figure 6.12: Probability maps obtained by screening example images with the each cascade models.

positive cases, can in the training data frequently predict a maximum in the center of lesions, while on the validation set presenting deviations from this position. Due to this, for the second model, training data might be slightly different from validation.

When we compare the two input sizes, AUCs are very similar. In theory, the second model receives more detail information, that should be useful for classification. Several factors can contribute to this result:

1. The amount of data available is not enough to train a model with more capacity. Due to the sampling method of building the second dataset, the amount of negative samples is much smaller than in the initial dataset. If this is the case, better regularization methods would enhance results.

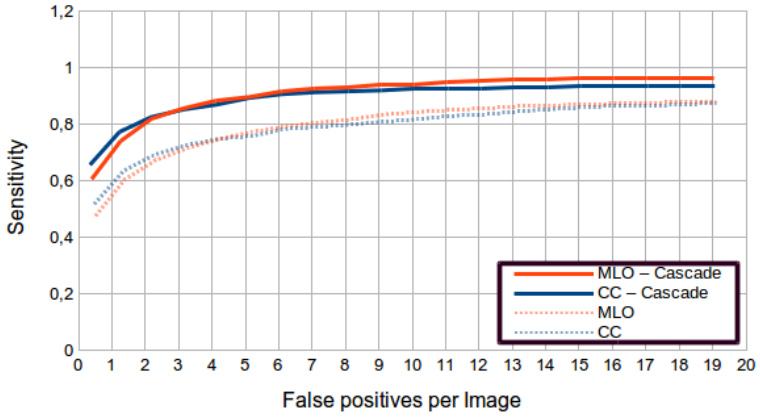
2. Additional detail information might not contribute to the task of discriminating lesions from normal tissue. Although in mammography images detail is important to distinguish between malignant and benign masses, for problem of detection this information might not be so relevant.

3. The classification task is very hard with no significant differences between some false and true positives. This is unlikely, but in a task like this an irreducible error rate should be present. If this is the case, the best way to increase performance would be to feed more information to the model, in the form of location, view or surrounding context.

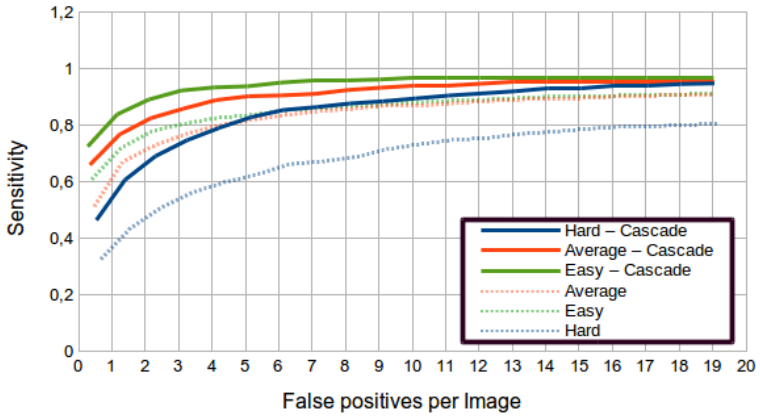
We used the $(76, 76)$ model to evaluate detection performance. For this, a new image is fed to the first classifier, yielding a probability map. This map is binarized using a fixed thresholded of 0.50. The second model is run on the whole image, at a different scale. The resulting probability map is multiplied by the previously obtained binary image. The same procedure explained in 6.8.3 is used to obtain detections. The results are depicted in fig. 6.13. As we can see, there is a significant increase in performance, when compared to using only one model. At two false positives per image the system is able to detect 80% of the lesions. This number rises to 85% if we

only consider the malignant cases. In terms of subtlety, the second model considerably increases performance on hard cases. As with the previous experiments the difference between MLO and CC for lesion detection is very small.

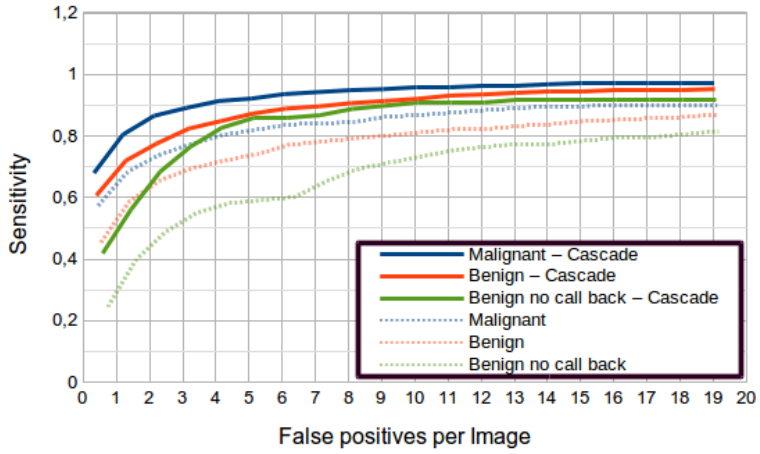
To our knowledge there is no work in the bibliography using the CBIS-DDSM dataset to compare this results to. As mentioned by Dromain et al. (2013), specialists have a false negative interpretation of $10-15\%$ for malignant cases. The proposed system is able to achieve comparable sensitivity at at 2-3 false positives per image. Additionally, in this framework, mammographic images are analyzed individually. Although not trivial the integration of both views could be used to enhance results. The same author explains that typically, the sensitivity of a CAD system to masses varies from 83% to 90% with a 0.72-1.82 false positive detections per image. The obtained detection results are slightly under mark with 80% detections at two false positives per image.



(a) View



(b) Subtlety



(c) Malignancy

Figure 6.13: Detection results obtained by the second cascade model.

Chapter 7

Final Remarks

7.1 Future Work

The proposed system for lesion detection is able to detect 80% of lesions at 2 false positives per image. Importantly, it has an increased sensitivity towards malignant cases and the time taken for a new image is only a few seconds. Future work should focus on increasing sensitivity at low false positives per image. There are several complementary strategies that could be followed to improve on this. We divide them in three groups: resources, deep learning methodologies and image processing in mammograms.

As always in the deep learning field, more data and computational power should contribute significantly to better results. Although ideal, this would require significant efforts in terms of collecting and labeling data. Additionally, the methods proposed here should be applied to other datasets, particularly in full field digital mammography images. As seen before, this modality has no digitalization artifacts and a higher signal-to-noise ratio, which could facilitate the detection process.

In parallel, the study of better regularization strategies to diminish the amount of mammograms required for the optimization of deep models should be pursued. In terms of data augmentation, in this work only rotation and mirroring transformations were applied. Perhaps the use of translation, scaling and more rotations could provide more robust models. Additionally, sophisticated approaches could be followed, for instance elastic distortions or normal tissue addition to lesions. The combination of all these methods could yield datasets up to one or two orders of magnitude superior to the ones used here. The rotated filters approach should be further explored to determine weather it is an effective regularization strategy or not. Additionally, instead of only using filter symmetry, approaches to achieve rotation invariance in the models could be explored, decreasing the number of optimizable parameters even further. The cascade approach was useful and could be extended by adding more models on top. Additionally, several techniques have been proposed in previous works to stabilize predictions, which were shown to have a positive impact in final result. These include running the model for an image at multiple rotations or filtering the output probability map to decrease noise.

As seen in the experimental work, the model is less sensitive to more subtle lesions. To address this problem preprocessing methods should be used to enhance these regions and facilitate posterior detection. Additionally, the model only sees a small portion of the image. The use of global image information to facilitate learning could be interesting to test. This could include location in the breast, pectoral muscle segmentation or epithelial tissue segmentation.

Future work should focus on integrating all these areas, to create more robust systems, able to surpass current state-of-the-art CAD systems. To increase performance deep learning should be used as a tool for the development of CAD rather than directly applied.

7.2 Conclusions

Nowadays, breast cancer is still a heavy burden in our society. The fact that early detection greatly improves patient outcome, motivates the development of new technologies to assist in this process. CAD systems have been shown to increase sensitivity in some studies. However, they still lack the desirable performance. The integration of modern deep learning methods in these systems could improve on this. Due to the nature of medical image data, the use of deep models requires some adjustments. In this work we focused on adapting Convolutional Neural Networks to the problem of breast lesion detection in mammography images.

For this, a region based approach was followed by training deep models to distinguish between normal breast tissue and lesions at a local level. These were then used to analyze whole mammograms to obtain detections. To deal with the small amount of data and the unbalance between healthy and lesion tissue three methods were studied.

First, rotated filters were incorporated in the model architecture to reduce the number of optimizable parameters. This technique produced only very small gains in certain conditions. Further research is necessary to better understand the benefits of this strategy.

Second, a rank learning approach was used to artificially increase the amount of data available. The performance was similar to conventional training, and proved inefficient in terms of computational resources. Additionally, we show that this strategy also leads to overfitting. It can still be used as an alternative to oversampling.

Third, a cascade approach was used to deal with unbalanced data. This strategy worked by training sequential biased models that, when used together, can discard many normal regions while keeping lesions.

The proposed detection system uses the cascade approach, and is able to detected 80% of lesions with only 2 false positives per image. This number rises to 85% , if we only consider malignant lesions. An important property of the proposed method is that it only takes a few seconds to process each image.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

American Cancer Society. Breast cancer survival rates, by stage, 2016. URL http://www.cancer.org/cancer/breastcancer/detailedguide/breast-cancer-survival-by-stage.

J. Anitha, J. Dinesh Peter, and S. Immanuel Alex Pandian. A dual stage adaptive thresholding (DuSAT) for automatic mass detection in mammograms. *Computer Methods and Programs in Biomedicine*, 138:93–104, 2017. ISSN 18727565. doi: 10.1016/j.cmpb.2016.10.026. URL http://dx.doi.org/10.1016/j.cmpb.2016.10.026.

John Arevalo, Fabio A. González, Raúl Ramos-Pollán, Jose L. Oliveira, and Miguel Angel Guevara Lopez. Representation learning for mammography mass lesion classification with convolutional neural networks. *Computer Methods and Programs in Biomedicine*, 127:248–257, 2015. ISSN 18727565. doi: 10.1016/j.cmpb.2015.12.014. URL http://dx.doi.org/10.1016/j.cmpb.2015.12.014.

Hakop Sarukhanyan Armen Sahakyan. Computer Vision - ECCV 2000. *International Journal of Advanced Computer Science and Applications(IJACSA)*, 1842(2):102–105, 2000. ISSN 0162-8828. doi: 10.1007/3-540-45054-8. URL http://link.springer.com/10.1007/3-540-45054-8.

Edward Azavedo, Sophia Zackrisson, Ingegerd Mejàre, and Marianne Heibert Arnlind. Is single reading with computer-aided detection (CAD) as good as double reading in mammography screening? A systematic review. *BMC medical imaging*, 12:22, 2012. ISSN 1471-2342. doi: 10.1186/1471-2342-12-22. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3464719{&}tool=pmcentrez{&}rendertype=abstract.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-Wise Training of Deep Networks. *Advances in Neural Information Processing Systems*, 19(1):153, 2007. ISSN 01628828. doi: citeulike-article-id:4640046. URL http://papers.nips.cc/paper/3048-greedy-layer-wise-training-of-deep-networks.pdf.

Helen Blumen, Kathryn Fitch, and Vincent Polkus. Comparison of Treatment Costs for Breast Cancer, by Tumor Stage and Type of Service. *American health & drug benefits*, 9(1): 23–32, 2016. ISSN 1942-2962. URL http://www.ncbi.nlm.nih.gov/pubmed/27066193{%}5Cnhttp://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4822976.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM. ISBN 0-89791-497-X. doi: 10.1145/130385.130401. URL http://doi.acm.org/10.1145/130385.130401.

Freddie Bray, Peter McCarron, and D Maxwell Parkin. The changing global patterns of female breast cancer incidence and mortality. *Breast cancer research : BCR*, 6(6):229–39, 2004. ISSN 1465-542X. doi: 10.1186/bcr932. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1064079{&}tool=pmcentrez{&}rendertype=abstract.

Breastcancer.org. Male breast cancer, 2017. URL http://www.breastcancer.org/symptoms/types/male_bc.

Meagan Brennan and Nehmat Houssami. Discussing the benefits and harms of screening mammography. *Maturitas*, 92:150–153, 2016. ISSN 18734111. doi: 10.1016/j.maturitas.2016.08.003. URL http://dx.doi.org/10.1016/j.maturitas.2016.08.003.

D. Brzakovic, X. M. Luo, and USA Brzakovic. An approach to automated detection of tumors in mammograms. *IEEE Transactions on Medical Imaging*, 9(3):233–241, 1990. ISSN 02780062. doi: 10.1109/42.57760.

Jaime S. Cardoso, Inês Domingues, Igor Amaral, Inês Moreira, Pedro Passarinho, João Santa Comba, Ricardo Correia, and Maria J. Cardoso. Pectoral muscle detection in mammograms based on polar coordinates and the shortest path. *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC'10*, pages 4781–4784, 2010. ISSN 1557-170X. doi: 10.1109/IEMBS.2010.5626634.

Gustavo Carneiro, Jacinto Nascimento, and Andrew P. Bradley. Unregistered multiview mammogram analysis with pre-trained deep learning models. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9351:652–660, 2015. ISSN 16113349. doi: 10.1007/978-3-319-24574-4_78.

David M Catarious, Alan H Baydush, and Carey E Floyd. Characterization of difference of Gaussian filters in the detection of mammographic regions. *Medical physics*, 33(2006):4104–4114, 2006. ISSN 00942405. doi: 10.1118/1.2358326.

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *CoRR*, abs/1405.3531, 2014. URL http://arxiv.org/abs/1405.3531.

Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *CoRR*, abs/1410.0759, 2014. URL http://arxiv.org/abs/1410.0759.

Dan C. Cireşan, Ueli Meier, Jonathan Masci, Luca M. Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1237–1242. AAAI Press, 2011. ISBN 978-1-57735-514-4. doi: 10.5591/978-1-57735-516-8/IJCAI11-210. URL http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-210.

D C Ciresan, A Giusti, L M Gambardella, and J Schmidhuber. Mitosis Detection in Breast Cancer Histology Images using Deep Neural Networks. *Proc Medical Image Computing Computer Assisted Intervenction (MICCAI)*, pages 411–418, 2013. ISSN 03029743. doi: 10.1007/978-3-642-40763-5_51.

Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, David Maffitt, Michael Pringle, Lawrence Tarbox, and Fred Prior. The cancer imaging archive (tcia): Maintaining and operating a public information repository. *Journal of Digital Imaging*, 26(6):1045–1057, 2013. ISSN 1618-727X. doi: 10.1007/s10278-013-9622-7. URL http://dx.doi.org/10.1007/s10278-013-9622-7.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. ISSN 1573-0565. doi: 10.1007/BF00994018. URL http://dx.doi.org/10.1007/BF00994018.

G Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989. ISSN 1435-568X. doi: 10.1007/BF02551274. URL http://dx.doi.org/10.1007/BF02551274.

Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. *CoRR*, abs/1512.04412, 2015. URL http://arxiv.org/abs/1512.04412.

DeepMind. Alphago, 2017. URL https://deepmind.com/research/alphago/.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Neeraj Dhungel, Gustavo Carneiro, and Andrew P. Bradley. A deep learning approach for the analysis of masses in mammograms with minimal user intervention. *Medical Image Analysis*, 37:114–128, 2017. ISSN 13618423. doi: 10.1016/j.media.2017.01.009. URL http://dx.doi.org/10.1016/j.media.2017.01.009.

Direção-Geral de Saúde. Norma 051/2011 - Abordagem Imagiológica da Mama Feminina. https://www.dgs.pt/directrizes-da-dgs/normas-e-circulares-normativas/norma-n-0512011-de-27122011-jpg.aspx, 2011. Online; accessed 13 January 2017.

Richard Drake, A Wayne Vogl, and Adam W M Mitchell. Gray's Anatomy for Students E-Book, 2009. URL https://www.123library.org/ebook/isbn/9781437720556/.

C. Dromain, B. Boyer, R. Ferré, S. Canale, S. Delaloge, and C. Balleyguier. Computed-aided diagnosis (CAD) in the detection of breast cancer. *European Journal of Radiology*, 82(3):417–423, 2013. ISSN 0720048X. doi: 10.1016/j.ejrad.2012.03.005. URL http://dx.doi.org/10.1016/j.ejrad.2012.03.005.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980. doi: 10.1007/bf00344251.

Roberta Fusco, Mario Sansone, Salvatore Filice, Guglielmo Carone, Daniela Maria Amato, Carlo Sansone, and Antonella Petrillo. Pattern Recognition Approaches for Breast Cancer DCE-MRI Classification: A Systematic Review. *Journal of Medical and Biological Engineering*, 36(4): 449–459, 2016. ISSN 21994757. doi: 10.1007/s40846-016-0163-7.

Karthikeyan Ganesan, U. Rajendra Acharya, Kuang Chua Chua, Lim Choo Min, and K. Thomas Abraham. Pectoral muscle segmentation: A review. *Computer Methods and Programs in Biomedicine*, 110(1):48–57, 2013. ISSN 01692607. doi: 10.1016/j.cmpb.2012.10.020. URL http://dx.doi.org/10.1016/j.cmpb.2012.10.020.

C Garcia and M Delakis. A neural architecture for fast and robust face detection. *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, 2(11):44 —- 47 vol.2, 2002. ISSN 1051-4651. doi: 10.1109/ICPR.2002.1048232.

Krzysztof J. Geras, Stacey Wolfson, S. Gene Kim, Linda Moy, and Kyunghyun Cho. High-resolution breast cancer screening with multi-view deep convolutional neural networks. *CoRR*, abs/1703.07047, 2017. URL http://arxiv.org/abs/1703.07047.

Maryellen L. Giger, Heang-Ping Chan, and John Boone. Anniversary Paper: History and status of CAD and quantitative image analysis: The role of Medical Physics and AAPM. *Medical Physics*, 35(12):5799–5820, 2008. ISSN 00942405. doi: 10.1118/1.3013555. URL http://doi.wiley.com/10.1118/1.3013555.

Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. URL http://arxiv.org/abs/1504.08083.

Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. URL http://arxiv.org/abs/1311.2524.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Pelin Gorgel, Ahmet Sertbas, and Osman N. Ucan. A wavelet-based mammographic image denoising and enhancement with homomorphic filtering. *Journal of Medical Systems*, 34(6):993–1002, 2010. ISSN 01485598. doi: 10.1007/s10916-009-9316-3.

Nelson HD, O'Meara ES, Kerlikowske K, Balch S, and Miglioretti D. Factors associated with rates of false-positive and false-negative results from digital mammography screening: An analysis of registry data. *Annals of Internal Medicine*, 164(4):226–235, 2016. doi: 10.7326/M15-0971. URL +http://dx.doi.org/10.7326/M15-0971.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

M. Heath, K. Bowyer, D. Kopans, P. Kegelmeyer, R. Moore, K. Chang, and S. Munishkumaran. *Current Status of the Digital Database for Screening Mammography*, pages 457–460. Springer Netherlands, Dordrecht, 1998. ISBN 978-94-011-5318-8. doi: 10.1007/978-94-011-5318-8_75. URL http://dx.doi.org/10.1007/978-94-011-5318-8_75.

Boulehmi Hela, Mahersia Hela, Hamrouni Kamel, Boussetta Sana, and Mnif Najla. Breast cancer detection: A review on mammograms analysis techniques. *2013 10th International Multi-Conference on Systems, Signals and Devices, SSD 2013*, pages 1–6, 2013. doi: 10.1109/SSD.2013.6563999.

GE Hinton, Simon Osindero, and YW Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.7.1527. URL http://www.mitpressjournals.org/doi/abs/10.1162/neco.2006.18.7.1527.

Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, Aug 1998. ISSN 0162-8828. doi: 10.1109/34.709601.

D. H. Hubel and T. N. Wiesel. Ferrier lecture: Functional architecture of macaque monkey visual cortex. *Proceedings of the Royal Society B: Biological Sciences*, 198(1130):1–59, 1977. doi: 10.1098/rspb.1977.0085.

Benjamin Q. Huynh, Hui Li, and Maryellen L. Giger. Digital mammographic tumor classification using transfer learning from deep convolutional neural networks. *Journal of Medical Imaging*, 3(3):034501, 2016. ISSN 2329-4302. doi: 10.1117/1.JMI.3.3.034501. URL http://medicalimaging.spiedigitallibrary.org/article.aspx?doi=10.1117/1.JMI.3.3.034501.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. *CoRR*, abs/1412.2007, 2014. URL http://arxiv.org/abs/1412.2007.

Andrej Karpathy and Li Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):664–676, 2017. ISSN 01628828. doi: 10.1109/TPAMI.2016.2598339.

N. Karssemeijer and G. M. te Brake. Detection of stellate distortions in mammograms. *IEEE Transactions on Medical Imaging*, 15(5):611–619, Oct 1996. ISSN 0278-0062. doi: 10.1109/42.538938.

Brigid K. Killelea, Jessica B. Long, Anees B. Chagpar, Xiaomei Ma, Rong Wang, Joseph S. Ross, and Cary P. Gross. Evolution of breast cancer screening in the medicare population: Clinical and economic implications. *JNCI: Journal of the National Cancer Institute*, 106(8):dju159, 2014. doi: 10.1093/jnci/dju159. URL +http://dx.doi.org/10.1093/jnci/dju159.

Dae Hoe Kim, Jae Young Choi, and Yong Man Ro. Boosting framework for mammographic mass classification with combination of cc and mlo view information. volume 8670, pages 86701V–86701V–5, 2013. doi: 10.1117/12.2007553. URL http://dx.doi.org/10.1117/12.2007553.

Thijs Kooi, Geert Litjens, Bram van Ginneken, Albert Gubern-Mérida, Clara I. Sánchez, Ritse Mann, Ard den Heeten, and Nico Karssemeijer. Large scale deep learning for computer aided detection of mammographic lesions. *Medical Image Analysis*, 35:303–312, 2017a. ISSN 13618423. doi: 10.1016/j.media.2016.07.007.

Thijs Kooi, Geert Litjens, Bram van Ginneken, Albert Gubern-Mérida, Clara I. Sánchez, Ritse Mann, Ard den Heeten, and Nico Karssemeijer. Large scale deep learning for computer aided detection of mammographic lesions. *Medical Image Analysis*, 35:303–312, 2017b. ISSN 13618423. doi: 10.1016/j.media.2016.07.007.

Thijs Kooi, Bram van Ginneken, Nico Karssemeijer, and Ard den Heeten. Discriminating solitary cysts from soft tissue lesions in mammography using a pretrained deep convolutional neural network. *Medical physics*, 44(3):1017–1027, 2017c. ISSN 24734209. doi: 10.1002/mp.12110.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F Pereira, C J C Burges, L Bottou, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Pelin Kus and Irfan Karagoz. Fully automated gradient based breast boundary detection for digitized x-ray mammograms. *Comput. Biol. Med.*, 42(1):75–82, January 2012. ISSN 0010-4825. doi: 10.1016/j.compbiomed.2011.10.011. URL http://dx.doi.org/10.1016/j.compbiomed.2011.10.011.

Sze Man Kwok, R. A. Chandrasekhar, and Y. Attikiouzel. Automatic assessment of mammographic positioning on the mediolateral oblique view. In *Image Processing, 2004. ICIP '04. 2004 International Conference on*, volume 1, pages 151–154 Vol. 1, Oct 2004. doi: 10.1109/ICIP.2004.1418712.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998a. doi: 10.1109/5.726791.

Yann Lecun, Leon Bottou, Genevieve B. Orr, and Klaus Robert Müller. Efficient backprop. *Lecture Notes in Computer Science Neural Networks: Tricks of the Trade*, page 9–50, 1998b. doi: 10.1007/3-540-49430-8_2.

Yann LeCun, Corinna Cortes, and Chris Burges. The mnist database, 1998. URL http://yann.lecun.com/exdb/mnist/.

Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(1):436–444, 2015a. ISSN 1548-7091. doi: 10.1038/nature14539. URL http://www.nature.com/nature/journal/v521/n7553/full/nature14539.html.

Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015b. doi: 10.1038/nature14539.

Rebecca Sawyer Lee, Francisco Gimenez, Assaf Hoogi, and Daniel Rubin. Curated breast imaging subset of ddsm., 2016. URL http://dx.doi.org/10.7937/K9/TCIA.2016.7O02S9CY.

Jan Lesniak, Rianne Hupse, Michiel Kallenberg, Maurice Samulski, Rémi Blanc, Nico Karssemeijer, and Gàbor Székely. Computer aided detection of breast masses in mammography using support vector machine classification. *SPIE Medical Imaging 2011: Computer-Aided Diagnosis*, pages 79631K–79631K–7, 2011. ISSN 0031-9155. doi: 10.1117/12.878140. URL http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=725203.

Daniel Lévy and Arzav Jain. Breast mass classification from mammograms using deep convolutional neural networks. *CoRR*, abs/1612.00542, 2016. URL http://arxiv.org/abs/1612.00542.

Zifeng Lian, Hai Huang, Youheng Tan, Yuanhao Cui, Xiaojun Jing, and Xiaohan Wang. Dropconnect regularization method with sparsity constraint for neural networks. *Chinese Journal of Electronics*, 25(1):152–158, Jan 2016. doi: 10.1049/cje.2016.01.023.

Miguel A. Guevara López, Naimy González de Posada, Daniel C. Moura, Raúl Ramos Pollán, José M. Franco Valiente, César Suárez Ortega, Manuel R. del Solar, Guillermo Díaz Herrero, Isabel M.A. Pereira Ramos, Joana Pinheiro Loureiro, Teresa Cardoso Fernandes, and Bruno M. Ferreira de Araújo. BCDR: A BREAST CANCER DIGITAL REPOSITORY. *ICEM15: 15th International Conference on Experimental Mechanics*, 2012.

Ramon Luengo-fernandez, Jose Leal, Alastair Gray, and Richard Sullivan. Economic burden of cancer across the European Union : a population-based cost analysis. *Lancet Oncology*, 14(12):1165–1174, 2008. ISSN 1470-2045. doi: 10.1016/S1470-2045(13)70442-X. URL http://dx.doi.org/10.1016/S1470-2045(13)70442-X.

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. *Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction*, pages 52–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-21735-7. doi: 10.1007/978-3-642-21735-7_7. URL http://dx.doi.org/10.1007/978-3-642-21735-7_7.

T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černocký. Strategies for training large scale neural network language models. In *2011 IEEE Workshop on Automatic Speech Recognition Understanding*, pages 196–201, Dec 2011. doi: 10.1109/ASRU.2011.6163930.

Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. Systematic evaluation of CNN advances on the imagenet. *CoRR*, abs/1606.02228, 2016. URL http://arxiv.org/abs/1606.02228.

Inês C Moreira, Igor Amaral, Inês Domingues, António Cardoso, Maria João Cardoso, and Jaime S Cardoso. INbreast: Toward a Full-field Digital Mammographic Database. *Academic Radiology*, 19(2):236–248, 2012. ISSN 1076-6332. doi: https://doi.org/10.1016/j.acra.2011.09.014. URL http://www.sciencedirect.com/science/article/pii/S107663321100451X.

John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, March 2008. ISSN 1542-7730. doi: 10.1145/1365490.1365500. URL http://doi.acm.org/10.1145/1365490.1365500.

Feng Ning, Damien Delhomme, Yann LeCun, Fabio Piano, Léon Bottou, and Paolo Emilio Barbano. Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, 14(9):1360–1371, 2005. ISSN 10577149. doi: 10.1109/TIP.2005.852470.

NVIDIA. Nvidia and ibm cloud support imagenet large scale visual recognition challenge, Aug 2015. URL https://devblogs.nvidia.com/parallelforall/nvidia-ibm-cloud-support-imagenet-large-scale-visual-recognition-challenge/.

Margarita Osadchy, Yann Le Cun, and Matthew L Miller. Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research*, 8:1197–1215, 2007. ISSN 1532-4435. doi: 10.1007/11957959.

Danilo Cesar Pereira, Rodrigo Pereira Ramos, and Marcelo Zanchetta do Nascimento. Segmentation and detection of breast cancer in mammograms combining wavelet analysis and genetic algorithm. *Computer Methods and Programs in Biomedicine*, 114(1):88–101, 2014. ISSN 18727565. doi: 10.1016/j.cmpb.2014.01.014. URL http://dx.doi.org/10.1016/j.cmpb.2014.01.014.

M. Peura and J. Iivarinen. Efficiency of simple shape descriptors. In *In Aspects of Visual Form*, pages 443–451. World Scientific, 1997.

Etta D. Pisano, Constantine Gatsonis, Edward Hendrick, Martin Yaffe, Janet K. Baum, Suddhasatta Acharyya, Emily F. Conant, Laurie L. Fajardo, Lawrence Bassett, Carl D'Orsi, Roberta Jong, and Murray Rebner. Diagnostic performance of digital versus film mammography for breast-cancer screening. *New England Journal of Medicine*, 353(17):1773–1783, 2005. doi: 10.1056/NEJMoa052911. URL http://dx.doi.org/10.1056/NEJMoa052911. PMID: 16169887.

Stephen M Pizer, E Philip Amburn, John D Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart Ter Haar Romeny, and John B Zimmerman. Adaptive Histogram Equalization and Its Variations. *Comput. Vision Graph. Image Process.*, 39(3):355–368, 1987. ISSN 0734-189X. doi: 10.1016/S0734-189X(87)80186-X. URL http://dx.doi.org/10.1016/S0734-189X(87)80186-X.

Radswiki. Mammography views | radiology reference article, 2016. URL https://radiopaedia.org/articles/mammography-views.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

Guido Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Sep 1986. doi: 10.1038/323533a0.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Berkman Sahiner, Heang Ping Chan, Nicholas Petrick, Datong Wei, Mark A. Helvie, Dorit D. Adler, and Mitchell M. Goodsitt. Classification of mass and normal breast tissue: A convolution neural network classifier with spatial domain and texture images. *IEEE Transactions on Medical Imaging*, 15(5):598–610, 1996. ISSN 02780062. doi: 10.1109/42.538937.

N. Saltanat, M. A. Hossain, and M. S. Alam. An efficient pixel value based mapping scheme to delineate pectoral muscle from mammograms. *Proceedings 2010 IEEE 5th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010*, pages 1510–1517, 2010. doi: 10.1109/BICTA.2010.5645272.

Ravi K. Samala, Heang-Ping Chan, Lubomir Hadjiiski, Mark A. Helvie, Jun Wei, and Kenny Cha. Mass detection in digital breast tomosynthesis: Deep convolutional neural network with transfer learning from mammography. *Medical Physics*, 43(12):6654–6666, 2016. ISSN 00942405. doi: 10.1118/1.4967345. URL http://doi.wiley.com/10.1118/1.4967345.

scikit-learn.org. Classifier comparison¶, 2017. URL http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html.

Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1605.06211, 2016. URL http://arxiv.org/abs/1605.06211.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICRL)*, pages 1–14, 2015. ISSN 09505849. doi: 10.1016/j.infsof.2008.09.005. URL http://arxiv.org/abs/1409.1556.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013. URL http://arxiv.org/abs/1312.6034.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. ISSN 15337928. doi: 10.1214/12-AOS1000.

D. Steinkraus, I. Buck, and P. Y. Simard. Using gpus for machine learning algorithms. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 1115–1120 Vol. 2, Aug 2005. doi: 10.1109/ICDAR.2005.251.

Wenqing Sun, Tzu-Liang (Bill) Tseng, Jianying Zhang, and Wei Qian. Enhancing deep convolutional neural network scheme for breast cancer diagnosis with unlabeled data. *Computerized Medical Imaging and Graphics*, 57:4–9, 2016. ISSN 08956111. doi: 10.1016/j.compmedimag.2016.07.004. URL http://linkinghub.elsevier.com/retrieve/pii/S0895611116300696.

Tony M. Svahn, Petra Macaskill, and Nehmat Houssami. Radiologists' interpretive efficiency and variability in true- and false-positive detection when screen-reading with tomosynthesis (3D-mammography) relative to standard mammography in population screening. *Breast*, 24(6):687–693, 2015. ISSN 15323080. doi: 10.1016/j.breast.2015.08.012. URL http://dx.doi.org/10.1016/j.breast.2015.08.012.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June-2015:1–9, 2015. ISSN 10636919. doi: 10.1109/CVPR.2015.7298594.

Guido M te Brake, Nico Karssemeijer, and Jan H C L Hendriks. An automatic method to discriminate malignant masses from normal tissue in digital mammograms. *Physics in Medicine and Biology*, 45(10):2843, 2000. URL http://stacks.iop.org/0031-9155/45/i=10/a=308.

Lindsey A. Torre, Freddie Bray, Rebecca L. Siegel, Jacques Ferlay, Joannie Lortet-tieulent, and Ahmedin Jemal. Global Cancer Statistics, 2012. *CA: a cancer journal of clinicians.*, 65 (2):87–108, 2015. ISSN 1542-4863 (Electronic). doi: 10.3322/caac.21262. URL http://onlinelibrary.wiley.com/doi/10.3322/caac.21262/abstract.

Srinivas C. Turaga, Joseph F. Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H. Sebastian Seung. Convolutional Networks Can Learn to Generate Affinity Graphs for Image Segmentation. *Neural Computation*, 22(2):511–538, 2010. ISSN 0899-7667. doi: 10.1162/neco.2009.10-08-881. URL http://www.mitpressjournals.org/doi/10.1162/neco.2009.10-08-881.

R. Vaillant. Original approach for the localisation of objects in images. *IEE Proceedings - Vision, Image, and Signal Processing*, 141(4):245, 1994. doi: 10.1049/ip-vis:19941301.

S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13(2):22–30, March 2011. ISSN 1521-9615. doi: 10.1109/MCSE.2011.37.

Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014. ISSN 2167-8359. doi: 10.7717/peerj.453. URL http://dx.doi.org/10.7717/peerj.453.

L. Vibha, G. M. Harshavardhan, K. Pranaw, P. D. Shenoy, K. R. Venugopal, and L. M. Patnaik. Statistical classification of mammograms using random forest classifier. In *2006 Fourth International Conference on Intelligent Sensing and Information Processing*, pages 178–183, Oct 2006. doi: 10.1109/ICISIP.2006.4286091.

Fred Winsberg, Milton Elkin, Josiah Macy Jr., Victoria Bordaz, and William Weymouth. Detection of Radiographic Abnormalities in Mammograms by Means of Optical Scanning and Computer Analysis. *Radiology*, 89(August):211–215, 1967. ISSN 0033-8419. doi: 10.1148/89.2.211.

World Health Organization. Eucan factsheets | portugal, 2012. URL http://eco.iarc.fr/eucan/Country.aspx?ISOCountryCd=620.

M. Yam, M. Brady, R. Highnam, C. Behrenbruch, R. English, and Y. Kita. Three-dimensional reconstruction of microcalcification clusters from two mammographic views. *IEEE Transactions on Medical Imaging*, 20(6):479–489, 2001. doi: 10.1109/42.929614.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014. URL http://arxiv.org/abs/1411.1792.

S. E. K. Yousuf and S. H. A. A. Mohammed. A parallel computer system for the detection and classification of breast masses. In *2013 INTERNATIONAL CONFERENCE ON COMPUTING, ELECTRICAL AND ELECTRONIC ENGINEERING (ICCEEE)*, pages 202–207, Aug 2013. doi: 10.1109/ICCEEE.2013.6633933.

Yading Yuan, Maryellen L. Giger, Hui Li, Kenji Suzuki, and Charlene Sennett. A dual-stage method for lesion segmentation on digital mammograms. *Medical Physics*, 34(11):4180–4193, 2007. ISSN 00942405. doi: 10.1118/1.2790837. URL http://doi.wiley.com/10.1118/1.2790837.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. URL http://arxiv.org/abs/1311.2901.

Wei Zhang, Kunio Doi, Maryellen L. Giger, Yuzheng Wu, Robert M. Nishikawa, and Robert A. Schmidt. Computerized detection of clustered microcalcifications in digital mammograms using a shift-invariant artificial neural network. *Medical Physics*, 21(4):517–524, 1994. doi: 10.1118/1.597177.