Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

School of Engineering and Sciences

Segmentation of Breast Cancer Lesions in Digital Mammograms: A Convolutional Network

A thesis presented by

Erick Michael Cobos Tandazo

Submitted to the
School of Engineering and Sciences
in partial fulfillment of the requirements for the degree of

Master of Science

in

Intelligent Systems

Monterrey, Nuevo León, May, 2016

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

School of Engineering and Sciences

The committee members, hereby, certify that have read the thesis presented by Erick Michael Cobos Tandazo and that it is fully adequate in scope and quality as a partial requirement for the degree of Master of Science in Intelligent Systems.

Dr. Hugo Terashima Marín
Tecnológico de Monterrey
Principal Advisor

Committee member A's name
Committee member A's institution
Committee Member

Committee member B's name
Committee member B's institution
Committee Member

Graduate Program Director's name
Associate Dean of Graduate Studies
School of Engineering and Sciences

Monterrey, Nuevo León, May, 2016

Declaration of Authorship

I, Erick Michael Cobos Tandazo, declare that this thesis titled, ”Segmentation of Breast Cancer Lesions in Digital Mammograms: A Convolutional Network” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Erick Michael Cobos Tandazo
Monterrey, Nuevo León, May, 2016

Dedication

Yet to write

Acknowledgements

Yet to write

Segmentation of Breast Cancer Lesions in Digital Mammograms: A Convolutional Network

by

Erick Michael Cobos Tandazo

Abstract

Yet to write

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Discriminating between cancerous and normal tissue in radiographic images of the breast is a complex problem in medical image analysis; in this thesis, we use convolutional networks to tackle it.

Cancer is caused by abnormal cells dividing uncontrollably, forming tumors and eventually invading surrounding tissue. It receives different names based on the part of the body where it originates. Breast cancer, among all cancers, has the highest incidence rate in the United States, an estimated 14.1% of cancer diagnoses in 2015, and the third highest mortality accounting for 6.9% of all cancer-related deaths. Among women, it is the most commonly diagnosed—29% of all cancer cases—and, besides lung cancer, the deadliest—killing 15% of all diagnosed cases [3]. The American Cancer Society recommends women aged 45 or older to get mammograms, images of the breast that show signs of tumor formation, annually or biennially [41]. We consider two types of diagnostic lesions detected on mammograms: clustered microcalcifications, tiny deposits of calcium that could appear around cancerous tissue; and breast masses, more direct signs of the existence of a tumor, although often benign.

Although radiologists are able to identify these lesions with high accuracy, computerized examination may be used to direct their attention to relevant regions, as a second informed opinion or when doctors are unavailable. This motivated the research group to design a computer-aided diagnosis system (CAD) for breast cancer; the present thesis falls under the scope of this project as its first attempt to use deep learning for lesion segmentation.

Traditional CAD systems are pipelines that process the image sequentially using different computer vision techniques; for instance, an standard layout will preprocess the image, identify regions of interest, extract features from the relevant parts and train a classifier on the extracted features. Although many successful systems are built following this pattern, it presents two disadvantages: (1) stages use intricate algorithms and handcrafted features creating an overly complex system that requires many experts to be modified or properly tuned and (2) stages are dependent and relations between them are often obscure: changes in one component affect the performance of others, every component needs to perform well for the system to perform well and every component needs to be improved to improve overall performance.

We use convolutional networks to replace most, if not all, of the stages of traditional image processing. Convolutional networks [18, 31], a natural extension to feedforward neural networks, are statistical learning models that use raw images as input and learn the image features relevant to the classification during training. They work well with minimally preprocessed images and encapsulate segmentation, feature extraction and classification in a single trainable model. Notwithstanding some drawbacks, convolutional networks are the state-of-the-art technology for object recognition [47].

Researchers have used small convolutional networks to detect breast masses from normal tissue [48] and individual microcalcifications from noise in the image [33, 19]. Recently, a bigger network incorporating newer features such as rectified linear unit activations, max-pooling, data augmentation and dropout was trained to identify malignant masses [4]. In these experiments mammograms were preprocessed, enhanced and potential masses and microcalcifications were segmented and presented to the network for classification. These work, specially Arevalo et al.'s, relates closely to our own. However, we use modern convolutional networks to identify lesions in the entire mammogram rather than to classify already segmented lesions.

We aim to learn whether convolutional networks could automatically segment mammographic images, how advantageous it is to use a bigger architecture, more data and tuned hyperparameters and whether we can achieve results similar to those of traditional systems.

In this introductory chapter, we emphasize the importance of the problem in Section 1.1, expand into the problem with traditional image analysis methods in Section 1.2, expose the particular objectives and hypotheses of the thesis in Sections 1.3 and 1.4, offer a brief summary of our methodology in Section 1.5, highlight the particular contributions of this work in Section 1.6 and, lastly, offer an outline of the thesis in Section 1.7.

1.1 Motivation

Breast cancer is the most commonly diagnosed cancer in woman and its death rates are among the highest of any cancer. It is estimated that about 1 in 8 U.S. women will be diagnosed with breast cancer at some point in their lifetime. Early detection is key in reducing the number of deaths; detection in its earlier stage (*in situ*) increases the survival rate to virtually 100% [24].

With current technology, a high quality mammogram is "the most efective way to detect breast cancer early" [38]. Mammograms are used by radiologists to search for early signs of cancer such as tumors or microcalcifications. About 85% of breast cancers can be detected with a screening mammogram [12]; this high sensitivity is the product of the careful examination of experienced radiologists. A computer-aided diagnosis tool (CAD) could automatically detect these abnormalities saving the time and training needed by radiologists and avoiding any human error. Computer based approaches could also be used by radiologists as a help during the screening proccess or as a second informed opinion on a diagnosis.

1.2 Problem Definition

Image segmentation partitions an image into multiple regions, essentially assigning a class to every pixel in the image; for instance, classifying each pixel in a street image as road, building, sky, tree, car, pedestrian, bycicle or background. Lesion segmentation is tasked with separating lesions from normal tissue in medical images. When searching for abnormal findings, radiologists perform lesion segmentation—although implicitly. Traditional CAD systems for lesion segmentation are based on computer vision methods that are often convoluted and hard-to-adapt [a].

Despite their widespread use and relative success, various limitations should be address to further advance the field:

- Lack of standard preprocessing techniques. Some techniques are commonly used but their performance can vary.
- Handcrafted features. The features extracted from the image are chosen beforehand (maybe designed with the help of experts) and special filters and image techniques are used to extract them.
- Expertise needs. They require knowledge in various fields such as radiology, oncology, image processing, computer vision, machine learning, etc.
- Pipeline structure. Systems are composed of many sequential steps. At each stage, the researcher chooses among many techniques and estimates many parameters representing a cost in time and performance as it is improbable to achieve an optimal combination.
- Low ceiling. Techniques are already complex and require much work to achieve only incremental improvements.
- Complexity. Issues such as non-desired or unknown dependencies between subsystems, difficulty to localize errors and maintainability could arise.

In this thesis, we use convolutional networks, a recent development in machine learning, (see Section 2.4) to remedy some of these problems. In particular, we simplify the system pipeline by using a single end-to-end trainable model that learns the relevant preprocessing and image features from raw data. We can also focus on improving the learning mechanism, both the model and algorithm, rather than working on designing novel image features or improving specific subsystems.

1.3 Objectives

The main goal of this work is to succesfully apply convolutional networks to segment breast cancer lesions in digital mammograms and to compare our results with those obtained by other groups doing similar work. Particularly, there are various subgoals which we expect to achieve as the project advances:

- Obtain and process the mammographic database to make it available for future research on campus.
- Develop software tools to handle the database and train new deep learning models.
- Analyze the performance of convolutional networks reported on the literature.

[a]See [5] for an example.

- Train a modern, fine-tuned convolutional network to perform lesion segmentation.
- Show the viability of convolutional networks for breast cancer detection and diagnosis.
- Generate results that could produce a conference or journal article.
- Use an alternative convolutional network model to improve our results.
- Propose new ideas for future research in the topic.

1.4 Hypothesis

Although a considerable amount of work on breast cancer detection and diagnosis has been done in the institution, this project will be the first approximation to using convolutional networks for efficiently detecting breast cancer. Convolutional networks are widely used for object recognition tasks and have shown very good results [47, 55, 17]. They have a big research community and have become one of the preferred methods for image classification tasks.

Due to the exploratory nature of this work we are uncertain of the results that will be obtained. Nevertheless, we have a well established idea of what to expect. Our hypothesis is that applying convolutional networks to mammographic images will produce similar results to those obtained using more traditional computer vision techniques with less hassle. Additionally, we expect that a simple convolutional network will fail to obtain competitive results; we will need a more refined convolutional network with well fitted hyperparameters. Furthermore, we believe that implementing convolutional networks for this domain will be moderately easy as other groups have already done it (see Section 2.6) and plenty of software is available.

1.4.1 Research Questions

Some of the questions which will be answered in this work are:

- Can we improve the results reported by other groups using convolutional networks? Is training a convolutional network on mammographic images better than computing numeric features from the mammograms and training a simpler classifier?
- Is deep learning feasible with the resources we have? Is our data and computational power sufficient? Is there any advantage to use GPU acceleration?
- Can we simplify the pipeline for breast cancer detection? Can preprocessing be replaced by more layers on the same convolutional network? Could we use the networks trained for image segmentation to perform detection or diagnosis?
- What are the best parameters for our convolutional networks (number of layers, number of units, kernel sizes, regularization, activation functions, etc)? Is there a big improvement on refining the network and tuning parameters?

- Should we train a convolutional network for each type of breast lesion or could we use a single one with multiple outputs?
- What are the advantages of using a deep versus a shallow convolutional network?
- Could we use a convolutional network trained on a different database (such as the ImageNet database) to obtain features for mammographic images and use these features for classification?
- Are convolutional networks a good option for future research?

1.5 Methodology

We carried out various tasks to achieve the proposed objectives and test our hypotheses. We list them here in order of execution:

1. **Literature review**

   A thorough review of the published work using the databases and resources available in the institution. By the end of this task, a complete theoretical background was obtained and reported. It also helped identify gaps in the literature and refine the scope of the project.

2. **Database processing**

   We looked for a mammographic database adept to our research, asked permission and developped tools to store, preprocess and label the images.

3. **Software review**

   Once we had a clear idea of what experiments will be executed, we found and learned-to-use appropiate software.

4. **Model selection**

   We performed simple experiments to determine the best parameters for the final model. Using these insights and the current literature on convolutional networks, we selected image preprocessing techniques, a network architecture and its features, training and regularization procedures, and evaluation metrics.

5. **Experiments**

   We trained the chosen convolutional network on our mammographic database. We performed crossvalidation to adjust the most important learning parameters and use regularization to avoid possible overfitting. We answered two research questions: is the performance of the convolutional network considerably improved by parameter tuning and, more importantly, is this a good performance?.

6. YET TO WRITE

1.6 Contributions

Yet to write

1.7 Outline of the thesis

This thesis is structured as follows: Chapter 1 introduced the problem and objectives of the thesis, Chapter 2 concisely presents concepts used and gives a thorough literature review, Chapter 3 lists design and implementation details for the final model, Chapter 4 reports experiments and discusses results and Chapter 5 concludes the thesis.

Chapter 2

Background

We offer an introduction to some of the essential concepts needed to understand the rest of this document. We start by discussing breast cancer and mammograms in Section 2.1, we explore some basic concepts about classification and evaluation metrics in Section 2.2, in Sections 2.3 and 2.4 we give a short introduction into artificial neural networks and convolutional networks, we offer some practical advice for deep learning in Section 2.5 and, finally, we present an overview of how convolutional networks have been used for breast cancer research in Section 2.6.

2.1 Breast Cancer

Cancer is an umbrella term for a group of diseases caused by abnormal cell growth in different parts of the body. The accumulation of extra cells usually forms a mass of tissue called a *tumor*. Tumors can be benign or malignant: *benign tumors* are noncancerous, lack the ability to invade surrounding tissue and will not regrow if removed from the body; malignant or *cancerous tumors* are harmful, can invade nearby organs and tissues (*invasive cancer*), can spread to other parts of the body (*metastasis*) and will sometimes regrow when removed [37].

Breast cancer forms in tissues of the breast. The two most common types of breast cancer are *ductal carcinoma* and *lobular carcinoma*, which start in the breast ducts and lobules, respectively (see Fig. 2.1). Breast cancer *incidence rate*, the number of new cases in a specified population during a year, is the highest of any cancer among American women. Its *mortality rate*, the number of deaths during a year, is also one of the highest of any cancer [24].

The *cancer stage* depends on the size of the tumor and whether the cancer cells have spread to neighboring tissue or other parts of the body. It is expressed as a Roman numeral ranging from 0 through IV; stage I cancer is considered *early-stage breast cancer* and stage IV cancer is considered *advanced*. Stage 0 describes non-invasive breast cancers, also known as *carcinoma in situ*. Stage I, II and III describe invasive breast cancer, i.e., cancer has invaded normal, surrounding breast tissue. Stage IV is used to describe metastatic cancer, i.e., it has spread beyond nearby tissue to other organs of the body.

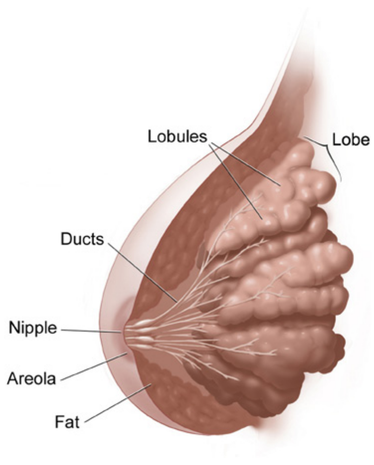

Figure 2.1: Anatomy of the female breast. Image courtesy of NCI.

Mammograms

A *mammogram* is an x-ray image of the breast. Radiologists use *screening mammograms* (normally composed of two mammograms of each breast) to check for breast cancer signs on women who lack symptoms of the disease. If an abnormality is found, a *diagnostic mammogram* is ordered, these are detailed x-ray pictures of the suspicious region [38]. A standard mammogram is shown in Fig. 2.2.

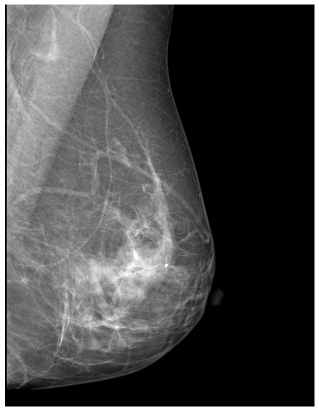

Figure 2.2: A standard mammogram.

Having a screening mammogram in a regular basis is the most effective method for detecting breast cancer early; around 85% of breast cancers can be detected in a screening mammogram [12]. Nevertheless, screening mammograms have many limitations: a high false positive rate, overtreatment in Stage 0 cancer, false negative results for women with high breast density, radiation exposure and physical and psychological discomfort [38].

Radiologists look primarily for microcalcifications and breast masses. *Microcalcifications* are tiny deposits of calcium in the breast tissue that can be a sign of early breast cancer if found in clusters with irregular layout and shapes. *Breast masses* or breast lumps are a variety of things: fluid-filled cysts, fatty tissues, fibric tissues, noncancerous or cancerous tumors, among others. A mass can be a sign of breast cancer if it has an irregular shape and poorly defined margins. See Fig. 2.3 for an example of signs of breast cancer. Radiologists will also consider the breast density of the patient when reading a mammogram given that high breast density is linked to a higher risk of breast cancer [2].

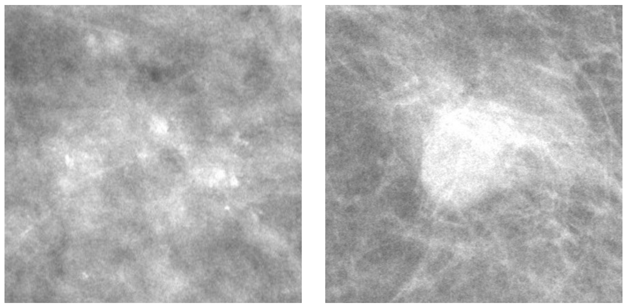

Figure 2.3: Signs of possible breast cancer in a mammogram. Left: A cluster of microcalcifications in an irregular layout. Right: A poorly defined breast mass.

Conventional mammography uses film to record x-ray images of the breast. *Digital mammography*, on the other hand, uses digital receptors to convert the x-rays into electrical signals and stores the image electronically. Digital mammograms offer a clearer picture of the breast and can be digitally manipulated and shared between health care providers. However, researchers still debate whether they offer an advantage over film mammograms [27, 43, 52]. Digital mammography is steadily becoming the standard for breast cancer screening. Fig. 2.2 is, in fact, a digital mammogram.

Digital tomosynthesis, also called three-dimensional mammography, is a new technology that produces 3-dimensional x-ray images of the breast and is expected to improve the efficacy of regular 2-d mammograms. Studies comparing the two techniques have not yet been published [38].

We use digital mammograms to detect breast masses and produce a valid image segmentation.

We wrote this section using information from the National Cancer Institute. We recommend to visit its website (`www.cancer.gov`) for further details.

2.2 Classification

Machine learning is the study of algorithms that build models of a population or function of interest and estimate their parameters from data in order to make predictions or inferences. A machine learning expert knows how to choose the right model for the problem in hand (*model selection*), how to efficiently estimate its parameters from the available data (*learning* or *training phase*) and how to evaluate the trained model (*testing phase*).

Machine learning problems divide into three categories depending on the data used to train the model: *supervised learning*, where we learn a function $f(x)$ using examples labelled with their correct output, for instance, learning to estimate the price of a house given its size and number of bedrooms from a data set of houses with their actual values; *unsupervised learning*, where we look for relationships and structure in unlabelled data, for instance, given a data set of potential customers finding those who are likely to buy a car and *reinforcement* *learning*, where feedback is received as rewards, for example, learning to play Tetris from a data set of world states, actions and rewards received every time points are earned (when lines dissapear). Supervised learning further divides in regression and classification. If the expected output is numerical, e.g., the price of a house, it is called *regression*, if the expected ouput is categorical, e.g., spam or no spam, it is called *classification*. We focus on classification.

A *classifier* takes as input a vector of *features* $x \in \mathbb{R}^n$ representing a problem instance and produces an *output* $h(x)$ predicting the class y to which that instance belongs, i.e., it models the underlying function $f(x)$ as $h(x)$ (h stands for hypothesis). *Binary classification*, when y can only take two values e.g., cancer/no cancer, is the most common kind of classification and *multiclass classification*, when y can take $K > 2$ different values, can be performed by using K binary classifiers. Some classifiers, such as convolutional networks (defined in Section 2.4, output a *score vector* $h(x) \in \mathbb{R}^K$ where $h(x)_k$ measures the likelihood of x belonging to class k. Every classifier partitions the *feature space*, the n-dimensional space where features exist, into separate *decision regions*, regions of the space that are assigned the same outcome; a *decision boundary* is the hypersurface that partitions the feature space. Classifiers are sometimes classified as *linear* or *nonlinear* according to the nature of the decision boundary they impose on the feature space. Logistic regression, for instance, is a linear classifier while an artificial neural network (with at least one hidden layer) is nonlinear.

The *loss function* $L(\theta)$ of a classifier measures the amount of error the classifier incurs in for a particular choice of parameters θ. This function could be formulated in many ways. A *least-squares loss function* for a binary classifier (such as logistic regression) is presented in Equation 2.1:

$$L(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{2.1}$$

where m is the number of training examples, $y \in \{0, 1\}$ is the real class of example x and $h_\theta(x) \in \mathbb{R}$ is the output of the classifier for input x with parameters θ, this represents the probability that x belongs to the positive class 1. We introduce another (rather more complex) loss function in the next section.

A classifier is trained by choosing the parameters θ that minimize its loss function, hence, minimizing the expected error of the classifier on the training set. *Gradient descent* is a method used to estimate these parameters: at the start, it initializes parameters at random and iteratively updates each parameter using the gradient of the loss function until it converges to a minimum. Specifically, at each iteration it performs the update:

$$\theta = \theta - \alpha \nabla L(\theta) \tag{2.2}$$

where α, called the *learning rate*, defines the step size. Gradient descent is guaranteed to converge to a global minimum if the loss function is convex; convexity of the loss function depends on the model $h(x)$.

To select the best model $h(x)$ for a particular problem, or equivalently, to select the best classifier for the problem, we train each model on a subset of the data and evaluate it on a disjoint subset. In the validation set approach the data set is split into a training set (usually 70-90%) and a validation set, each model is trained using the training set, evaluated on the validation set and the best-performing model is selected. *k-fold cross validation*, on the other hand, divides the data set in k disjoint subsets (usually 5 or 10) and uses $k - 1$ subsets to train the model and the remaining subset for evaluation, this process is repeated k times for each model leaving out a different subset each time and the k performance measures are averaged to obtain a final measure for the model. *Model hyperparameters*, settings that modify the underlying model or learning algorithm, are selected in a similar manner.

The model representation $h(x)$ needs to be chosen carefully. If we have an overly *flexible* model, i.e, $h(x)$ is a complex function with many parameters to be learned relative to the size of the training set, the classifier will probably *overfit* the data; this means that parameters are fitted too tight to the training set and pick up small fluctuations and noise causing the classifier to produce almost-perfect results on the training set but perform poorly on unseen examples. The opposite is also true, when $h(x)$ is very simple the classifier lacks the power to model the underlying function of interest and we say that it *underfits* the data.

A popular way to avoid overfitting (and underfitting) is to use a flexible model trained with regularization. *Regularization* modifies the loss function to penalize the complexity of the model, forcing the learning stage to choose parameters that minimize both the training error of the classifier and the complexity of the model. Equation 2.3 shows the least-squares loss function with l_2*-norm regularization*:

$$L(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}||\theta||_2 \tag{2.3}$$

where $||\cdot||_2$ is the euclidean norm of a vector. In addition to reducing training error, minimizing the regularized loss function will shrinken the parameters θ hopefully setting some of them to zero and simplifying $h(x)$. The *regularization strength* λ regulates the tradeoff between less training error and less regularization error. l_1*-norm regularization* or *lasso* is similar to l_2-norm regularization except that it shrinks the l_1-norm of θ instead of the l_2-norm.

To evaluate the performance of a classifier we use a separate set of examples (a test set) that should have not been used for training or validation. Classification accuracy is the standard performance measure in machine learning. *Accuracy* measures the proportion of test set examples correctly classified. Its compliment, *error rate*, measures the proportion of test set examples incorrectly classified. Accuracy, nonetheless, is inappropiate for *unbalanced data sets*, data sets that have many more examples of one class than the other [a]. A classifier that always predicts the predominant class regardless of the input will have high accuracy, it will be right most of the time, even though it is a bad model for the problem.

In unbalanced data sets, we use metrics based on the confusion matrix of the classifier to evaluate its quality. A *confusion matrix* is a matrix that summarizes the results of a classifier in the test set (see Table 2.1). *True positives* is the number of positive examples correctly

		Actual class	
		Positive	Negative
Predicted class	Positive	True Positives (TP)	False Positives (FP)
	Negative	False Negatives (FN)	True Negatives (TN)

Table 2.1: Confusion matrix for a binary classifier

[a]Cancer data sets are often unbalanced as most examples belong to the negative class (no cancer) than the positive class (cancer)

predicted as positive. *False positives* is the number of negative examples incorrectly predicted as positive. True negatives and false negatives are defined similarly. Based on the confusion matrix we can compute some commonly used metrics:

$$Sensitivity \text{ or } Recall = \frac{TP}{TP + FN} \tag{2.4}$$

$$Specificity = \frac{TN}{FP + TN} \tag{2.5}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

Sensitivity and specificity are preferred to present results in medical diagnosis meanwhile precision and recall are preferred in machine learning. *Sensitivity* measures the proportion of positive examples predicted as positive and *specificity* measures the proportion of negative examples predicted as negative. *Precision* measures the proportion of examples predicted as positive that are actually positive. A good classifier will have both high sensitivity and high specificity or similarly, high precision and high recall. It is always useful to have a single metric to evaluate classifiers, for example, to choose between two models; we show two commonly used metrics in Equation 2.7 and 2.8.

$$F_1 \; score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.7}$$

$$G\text{-}mean = \sqrt{Sensitivity \times Specificity} \tag{2.8}$$

The *threshold* of a classifier is the probability at and over which an example is classified as positive. It regulates the trade-off between sensitivity and specificity (or similarly precision and recall): a classifier with a low threshold is prone to classify examples as positive but will potentially produce many false positives thus having high sensitivity but low specificity and viceversa for high thresholds. The *precision-recall curve* of a classifier is a plot of its precision (on the y axis) against its recall (on the x axis) as the threshold varies (see Fig. 2.4). The *receiver operating characteristic curve* plots sensitivity (also called true positive rate) against 1-specificity (also called false positive rate) as the threshold varies (see Fig. 2.4). The *area under the precision-recall curve* PRAUC and the *area under the receiver operating characteristic curve* AUC summarize the performance of the classifier over all possible thresholds and are used for model selection; they range from 0 to 1 with higher being better. As with previous metrics, AUC is preferred for medical diagnosis while PRAUC is mostly used in machine learning.

For unbalanced data sets, "using the classifiers produced by standard machine learning algorithms without adjusting the output threshold may well be a critical mistake" [44]. It is preferable to use metrics that consider all possible thresholds (AUC or PRAUC) or simpler metrics (F_1 score or G-mean) with a threshold obtained via a validation set. The metric used for model selection influences its characteristics and behaviour, hence, it should be chosen carefully: we favor the use of PRAUC over AUC as well as F_1 score over G-mean because they concentrate in the positive class (cancer) which is harder to predict and more interesting.

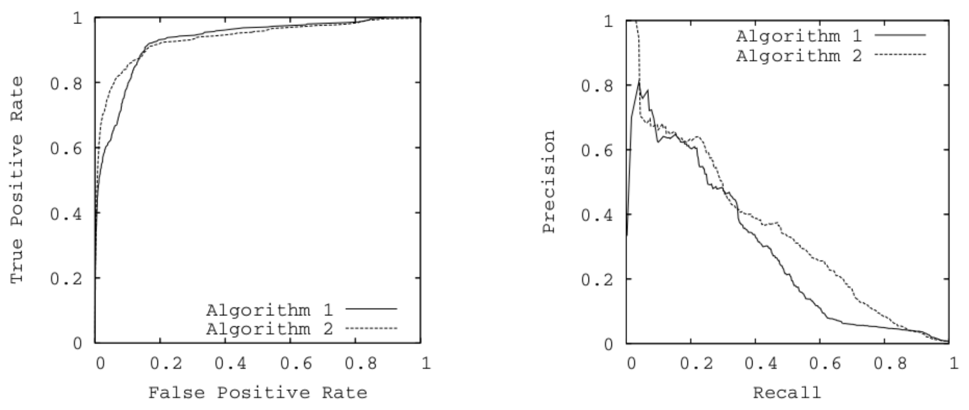

Figure 2.4: A sample receiver operating characteristic curve (left) and precision-recall curve (right). Each algorithm is evaluated on different thresholds and the points produced are used to obtain the curves. Image courtesy of [16]

Furthermore, PRAUC has been shown to have better properties than AUC in unbalanced data sets [16]. In general, we present results for all these metrics.

Finally, we point out that this section is a compendium of basic concepts in machine learning and lefts aside many subtleties of practical machine learning. Notation and content is mostly based on materials from Stanford's Machine Learning course [40].

2.3 Artificial Neural Networks

Artificial neural networks or simply *neural networks* are one of the most popular nonlinear classifiers used today. They were inspired by the way biological neurons integrate information from its dendrites and relay it to neighboring neurons [35, 59, 45] but evolved to specialize in nonlinear modelling at the expense of biological accuracy [46].

Multilayer feedforward neural networks are composed of L layers of *neurons*, units of computation. Each layer is fully connected to the next and previous layer (except for the first and last layer). The first layer, called the *input layer*, has $s^{(1)} = n$ units and receives/relays/taes as input the feature vector $x \in \mathbb{R}^n$ while the last layer or *output layer* has $s^{(L)} = K$ units corresponding to the K possible classes. Every other layer is called a *hidden layer* (see Fig. 2.5 for an example). The neural network receives an input $x \in \mathbb{R}^n$, processes it layer by layer and outputs a vector $h_\Theta(x) \in \mathbb{R}^K$, where $h_\Theta(x)_k$ is the predicted (unnormalized log) probability that x belongs to class k. Each unit performs a computation on the input from the units in the previous layer and transmits the result to the units in the next layer through its connections. Furthermore, each connection has a *weight* w that is to be learned in the training phase, i.e, the weights are the parameters Θ of the model. A neural network is said to be *shallow* or *deep* according to its number of layers or *depth*.[b]

[b]Consensus on when a neural network becomes deep has not been reached [49]. We consider networks with two or more hidden layers to be deep.

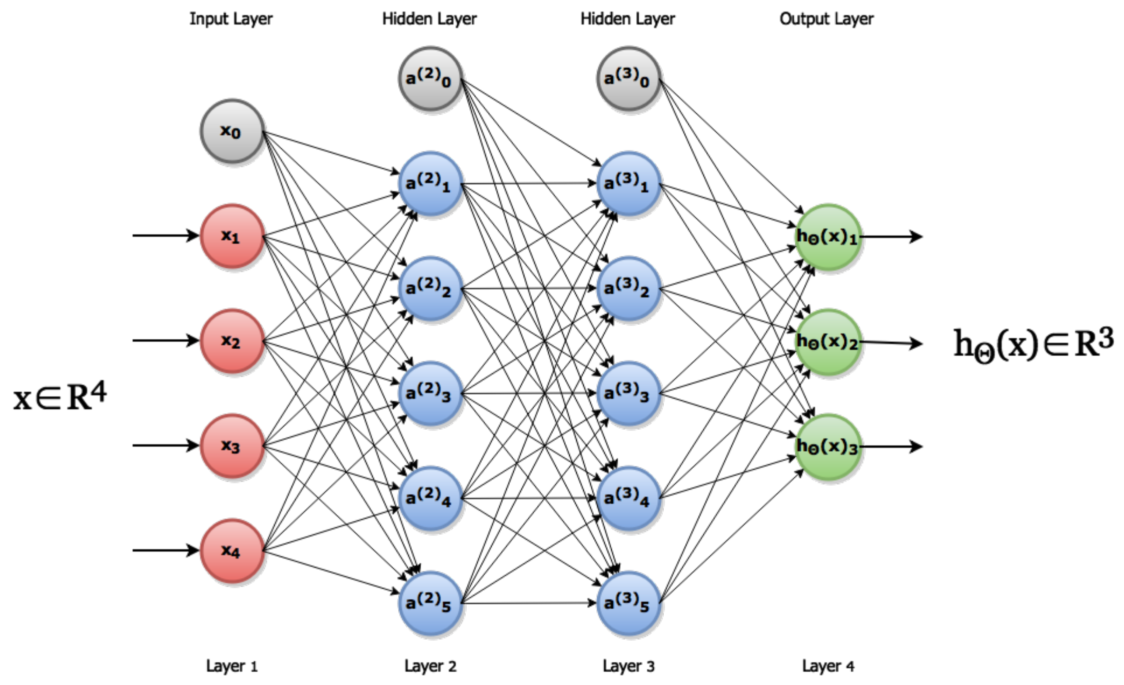

Figure 2.5: Small neural network example. Input layer with 4 units (red), two hidden layers of 5 units (blue) and output layer of 3 units (green). Bias units appear in gray. It approximates a function $h_\Theta(x) : \mathbb{R}^4 \to \mathbb{R}^3$, i.e., it classifies an input vector $x \in \mathbb{R}^4$ into 3 possible classes.

A unit i in layer l computes a function of the form:

$$a_i^{(l)} = g\left(\sum_{j=0}^{s^{(l-1)}} \Theta_{ij}^{(l-1)} a_j^{(l-1)}\right) \quad \text{for } l = 2, \dots, L-1 \text{ and } i = 1, \dots, s^{(l)} \tag{2.9}$$

where $a_i^{(l)}$ is called the *activation* or output of unit i in layer l; $g(\cdot)$ is an *activation function* (defined below); $s^{(l)}$ is the number of units in layer l; $a_0^{(u)} = 1$, for all $u = 1, \dots, L-1$ (bias units); $a_v^{(1)} = x_v$ for all $v = 1, \dots, n$ i.e, the activation of the input layer is the input x; $a_i^{(L)} = \sum_{j=0}^{s(L-1)} \Theta_{ij}^{(L-1)} a_j^{(L-1)}$ for all $i = 1, \dots, s^{(L)}$ i.e, $g(\cdot)$ is omitted in the output layer and $\Theta^{(l)} \in \mathbb{R}^{s^{l+1} \times s^l}$ is the matrix of weights connecting layer l to $l+1$. At each layer (except the output layer) we include a unit that always emits activation 1 ($a_0^{(1)} = 1$, $a_0^{(2)} = 1$, etc), these are called *bias units* [c]. The bias units are assumed to be included into each vector $a^{(l)}$, hence the sumation in Equation 2.9 starts at 0 and not 1. It may seem like a convoluted definition but it simply defines the activation of a given unit as the weighted linear combination of activations of the units in the previous layer passed through a nonlinear function $g(\cdot)$. Lastly, notice that $a^{(L)} \in \mathbb{R}^{s^{(L)}}$, the vector of activations in the last layer of the network, is equal to the predicted (unnormalized log) probabilities $h_\Theta(x) \in \mathbb{R}^K$, also called a *score vector*. The class with the highest score in $h_\Theta(x)$ is the predicted class for the example x. We could also exponentiate each of these probabilities and normalize them to obtain a distribution of probabilities over

[c]They are included for a technical detail: so that the activation function $g(w^T x + w_0)$ can shift in the x-axis changing its threshold to $-w_0$, which is learned by the neural network.

the possible classes K ($p(x) \in [0..1]^K$). This improves interpretability and preserves the predicted classes.

The activation function $g(\cdot)$ is usually a *rectified linear unit* or *ReLU*:

$$g(z) = \max(0, z) \tag{2.10}$$

This is a nonlinear activation function very easy to implement in numerical computations, its derivative ($1_{z>0}$) can be calculated quickly and, unlike the sigmoid or tanh activation functions, does not suffer from vanishing and exploding gradients. Furthermore, it has been shown to greatly accelerate the convergence of gradient descent [29]. For these reasons it is currently the recommended activation function for deep neural networks [26].

Each unit in a neural network outputs a nonlinear activation $g(z)$ that in turn is received by units in the next layer, linearly recombined with the activation of other units and passed again through a nonlinear function; these operations are repeated until the input reaches the output layer. As a result, the function calculated by units in the output layer $h_\Theta(x)$ will be highly nonlinear on the original input x. This is the reason why neural networks can model functions that are highly nonlinear and why increasing the number of layers in a neural network increases the predictive power of the model. By the same token, it may be insightful to think of each unit in a neural network as a feature detector: units in the first hidden layer are trained to activate when simple features are found on the input, units on the second hidden layer activate when a combination of those simple features is present on the input and so on. Thus, the network will learn to detect the most relevant features for the classification task and as the number of units increases, it learns ever more complex features (if enough training data is provided).

The *softmax* loss function for a multiclass neural network classifier is defined as:

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m} \log\left(\frac{e^{h_\Theta(x^{(i)})_{y^{(i)}}}}{\sum_{j=1}^{K} e^{h_\Theta(x^{(i)})_j}}\right) \tag{2.11}$$

where m is the number of examples in the training set, $h_\Theta(x)$ is the score vector, K is the number of classes and $(x^{(i)}, y^{(i)})$ is the i^{th} example. $J(\Theta)$ is differentiable with respect to Θ but non-convex, nonetheless, gradient descent usually converges to a good estimate of the network weights [40]. *Error backpropagation* [32, 58], an algorithm to calculate the derivatives with respect to Θ needed for gradient descent, computes the error terms in the output layer and backpropagates them layer by layer using the chain rule of derivatives. Deep neural networks are often susceptible to overfitting given the amount of parameters that need to be estimated. The simplest approach to overcome this problem is to use regularization. Regularization for neural networks is done by performing gradient descent on the regularized loss function presented in Equation 2.12

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m} \log\left(\frac{e^{h_\Theta(x^{(i)})_{y^{(i)}}}}{\sum_{j=1}^{K} e^{h_\Theta(x^{(i)})_j}}\right) + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s^{(l)}}\sum_{j=1}^{s^{(l+1)}}\left(\Theta_{ij}^{(l)}\right)^2 \tag{2.12}$$

Dropout [54] is another popular method to prevent overfitting. Each training iteration, dropout samples a different network architecture from the original network and updates only a subset of the values in Θ; a unit (and its connections) is retained with probability p, usually 0.5, and gradient descent works on this sampled network. See Fig. 2.6 for an example. During testing all units are active but their activations are scaled by p to match their expected output $(pa_i^{(l)} + (1-p)0)$. This can be interpreted as training many models (with shared weights) and averaging their results at test time.

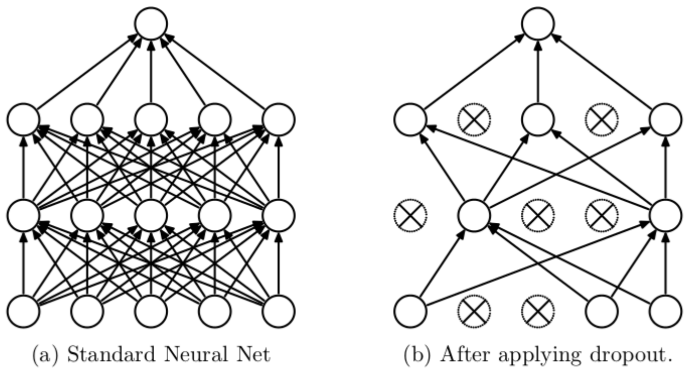

Figure 2.6: Dropout applied to a simple neural network. Crossed units were dropped. Image courtesy of [54].

2.4 Convolutional Networks

Convolutional networks, also *ConvNets* or *CNNs*, were first inspired by the way the human visual cortex proccesses information [18] but, as regular neural networks, have evolved to favor practical performance over biological accuracy. LeCun et al. used a convolutional network to achieve good classification performance on the MNIST data set of handwritten digits [30, 31], the first successful application of modern convolutional networks. Recently, Krizhevsky et al. used them to achieve state-of-the-art performance on the ImageNet Large-Scale Visual Recognition Challenge [29], an image classification and object localization challenge with 1000 categories [47]. Thanks to various advances (maxpooling, ReLU activations, weight initialization, GPU training, efficient backpropagation, etc.), convolutional networks have become one of the most popular methods for image classification tasks and (along with recursive neural networks) an emblem of deep learning.

In this section we show the standard features and training of current convolutional networks. Section **??** gives some practical advice for choosing hyperparameters and training deep architectures. For an in depth review of convolutional networks, see [26]. For a complete overview of the history and state of deep learning, see [49].

Convolutional networks map raw image pixels to a score vector $h_\Theta(x) \in \mathbb{R}^K$ representing the distribution of (unnormalized log) probabilities over the K classes. We could use a regular neural network (presented in Section **??**) to do this but the amount of learnable parameters (the weights) becomes very big. For instance, a small color image of size 100×100 with 3 color channels (RGB) will require $30\,000$ units in the input layer and each unit in the second layer will therefore have $30\,000$ weights to learn. This is impractical not only because it will require a lot of data and training time but because the loss function has very many local minima and thus it is harder to optimize.

Convolutional networks reduce the number of connections between layers and the number of parameters to learn. Convolutional layers are *sparsely connected*, i.e., a unit is only connected to a small subset of the units in the previous layer. Furthermore, they are *locally connected*, i.e., units are connected considering their position on the original image. The architecture of a convolutional network also imposes *weight sharing* between units in the same layer, i.e., different units are forced to share the same weights (this determines filters and feature maps, defined below). *Pooling* is a subsampling mechanism that reduces the spatial scale and makes the computations invariant to local translation. All these features reduce computation and improve the classification performance of convolutional networks; they are produced by the way convolutional networks are defined, which we explain below.

Each layer is composed of a set of *feature maps*, 2-dimensional grids of unit activations ($\mathbb{R}^{h \times w}$), arranged into a 3-dimensional matrix ($\mathbb{R}^{h \times w \times d}$). We use the third dimension to put together all feature maps. One could unfold a 3-dimensional volume to form a single column of unit activations as in regular neural networks but seeing them as a volume makes the definitions easier. The input layer is a 3-dimensional matrix ($\mathbb{R}^{h \times w \times c}$) that holds the input image of size $w \times h$ with c color channels ($c = 1$ for grayscale images or $c = 3$ for RGB). The output layer is a volume of size $R^{1 \times 1 \times K}$ where each feature map is just one activation ($R^{1 \times 1}$) representing the final score. The network receives an image x as an input volume that is transformed layer by layer into new volumes (whose dimensions could be different from the previous one) until it reaches the output layer of size $h(x) = R^{1 \times 1 \times K}$. See Fig. 2.7 for an illustration. We describe the transformations next.

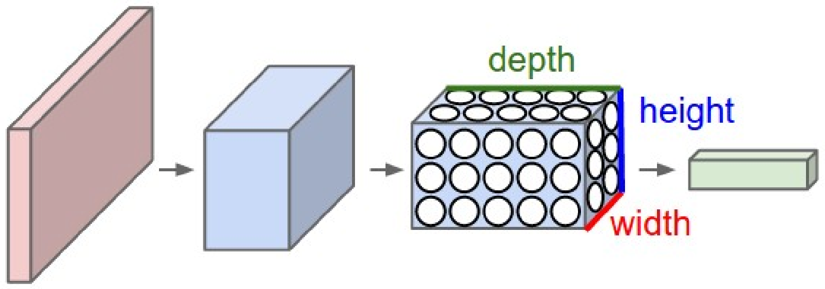

Figure 2.7: A simple representation of the transformations of the input computed by a convolutional network. Input layer is shown in pink, hidden layers are shown in blue and output layer is shown in green. The third layer has 5 feature maps of size 2×3. Notice that the width is listed first by convention. Image courtesy of [26].

Four types of layers are commonly used: convolutional layer, ReLU layer, pooling layer and fully connected layer; all of which compute a differentiable function on its input and combine to form a convolutional network architecture.

Convolutional layer Convolutional layers are the heart of convolutional networks. They are build by learnable filters that are applied to the volume in the previous layer. A *filter* is a matrix of weights that has a small spatial size (width and height) but goes across all feature maps of the volume (the third dimension). For instance, a 3×3 filter to be applied in a volume with 10 feature maps will have 90 parameters ($\mathbb{R}^{3\times3\times10}$). See Figure 2.8 for an example. Each feature map in this layer is obtained by sliding a filter across the spatial dimensions (width and height) of the previous volume computing the dot product (a weighted sum) between the filter and the input [d]. All values in a single feature map are computed using the same filter. If we think of the feature map as a grid of units we can see that every unit is connected with only a small local subset of the units in the previous layer and that all units in the map share the same weights.

At each convolutional layer, many feature maps are computed (each with its own filter) and stacked together to form the volume in the layer. We can think of each filter as looking for an specific feature on the input and each feature map as the probabilities of that feature in each position.

We choose various hyperparameters for this layer: the filter size, the stride (the number of places to shift the filter at each step), the amount of zero padding around the image and the number of feature maps. These define the shape of the resulting volume; the first three are usually chosen to preserve the spatial size of the previous volume, the number of feature maps depends on the amount of features we want to learn.

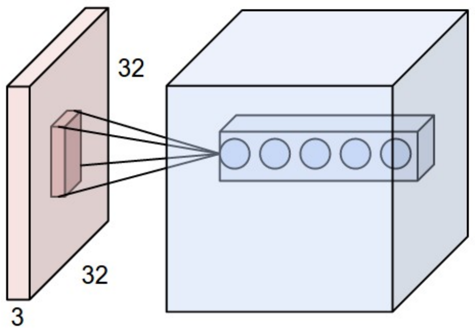

Figure 2.8: Example of a filter applied to a volume ($\mathbb{R}^{32\times32\times3}$) to obtain the values shown in the blue volume. The filter comprises all 3 feature maps of the input volume. We compute 5 feature maps as shown by the 5 units in the blue volume. For a complete convolution this filter will have to slide across the input volume. All units in the same feature map share the same filter but units in different feature maps do not, even though they can be connected to the same local region of the input. Image courtesy of [26].

ReLU layer This layer receives an input volume and performs an elementwise ReLU activation function to it, i.e, each value z in the volume is passed through the nonlinearity $\max(0, z)$. It preserves the dimensions of the volume and has no learnable parameters, although the activation function may be considered a hyperparameter. Usually, a ReLU layer (or any other activation function) follows a convolutional layer, for this reason it is sometimes considered part of the convolutional layer. We separate them for clarity.

Pooling layer The pooling layer subsamples the volume on the spatial dimensions reducing the size of the feature maps but keeping the number fixed. Standard max pooling slides a fixed

[d]Each filter has also a bias term added to the sum.

size windows (normally 2×2) along each feature map with stride 2 (it is, without overlapping) and selects the maximum element on that space. This will reduce each dimension of the feature map by half, thus reducing the total number of activations by 75%, e.g., a 4×4 feature map gets subsampled to size 2×2 where each value is the maximum activation on each of the four quadrants of the original feature map. Contrary to convolution, subsampling is applied to each feature map separately. A popular variant of max pooling uses 3×3 windows with stride 2, allowing for some overlapping.

Fully connected layer We use one or more fully connected layers at the end of the network to compute the final score vector. Feature maps in this layer have size 1×1 resulting in a row volume or alternatively a row vector of values. Each feature map in this layer is fully connected to all units in the previous volume and outputs a dot product between the input and the connection weights, which are the parameters to be learned during training. The output layer of a convolutional network is always a fully connected layer with as many feature maps as classes. We interpret the scores of the output layer similar to those of regular neural networks as the (unnormalized log) probability of x belonging to class k. Lastly, notice that a fully connected layer can be simulated by a convolutional layer with the same number of feature maps and filter size $w \times h$ where w and h are the spatial dimensions of the previos volume, i.e., filters that comprise the entire volume.

Convolutional layers (plus ReLUs) compute features on the input while pooling layers shrinken the volume before passing to the fully connected layers that act as a regular neural network classifier on the obtained features. The standard convolutional network architecture can be represented textually as:

```
INPUT -> [[CONV -> RELU]*N -> POOL?]*M -> [FC -> RELU]*K -> FC
```

where `*N` indicates that the layers are repeated `N` times, `?` indicates that the layer is optional and `N,M,K >= 0`. We can use this template to construct ever more flexible models from a linear classifier `INPUT -> FC` (`N,M,K = 0`) to a regular neural network `INPUT -> [FC -> RELU]+ -> FC` (`N,M = 0, K > 0`) to a convolutional network `INPUT -> [[CONV -> RELU]+ -> POOL?]+ -> [FC -> RELU]* -> FC`(`N,M > 0,K >= 0`). For instance, a typical deep convolutional network could be:

```
INPUT -> [[CONV -> RELU]*2 -> POOL]*3 -> [FC -> RELU]*2 -> FC
```

This network receives an input volume (the image) computes two sets of convolution plus ReLUs before pooling and repeats this pattern three times followed by fully connected layers plus ReLUs which are repeated twice and the output layer that reports the final classification scores. Although there is no standard way of counting the number of layers, we usually ignore the ReLU and pooling layers as they have no learnable parameters. Therefore, our example has 10 layers (21 in total), which is a good depth for big data sets. Practical recommendations on building architectures is offered in the Section **??**.

Figure 2.9 shows an example of a convolutional network with its different kind of layers. The image is taken from a simulation accesible at `cs231n.stanford.edu`.

Recently, simpler convolutional network architectures have emerged. The All Convolutional Net [53] is a network formed solely by convolutional layers: we replace pooling layers

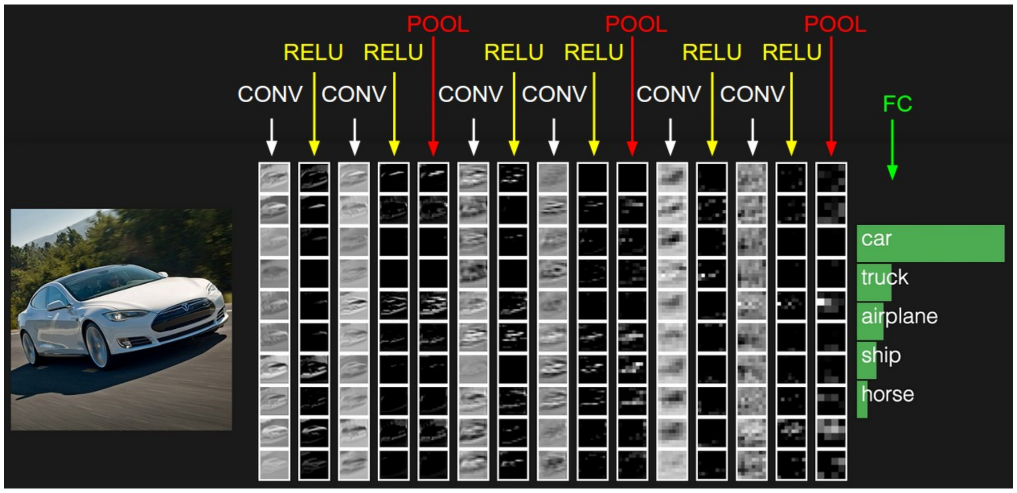

Figure 2.9: Example of a convolutional network with architecture `INPUT -> [[CONV -> RELU]*2 -> POOL]*3 -> FC`. The input image has size 32×32. Each hidden layer uses 10 feature maps (shown as columns). Although the spatial size of the feature maps looks constant in the image, each pooling layer reduces its dimensions by half (the feature maps of the final pooling layer have size 4×4). We show the final scores only for the five most probable classes. Image courtesy of [26].

by convolutional layers with larger strides and fully connected layers as explained above. This greatly increases the number of parameters to be learn, therefore it is unsuitable for small data sets.

Converting the fully connected layers to convolutional layers has another advantage: we can use a convolutional network trained on small images to classify bigger images. By the way convolutional layers are defined when we use a bigger image as input the entire convolutional network will slide across the image and be applied to different portions of the image generating a score vector for each of them. Therefore, instead of having a single score for each class we will have an entire matrix of scores (for each position where the convolutional network was applied). Then, we can average over all scores per class to obtain a single score vector for the bigger image. Furthermore, we can control the stride of the convolution to choose how the convolutional network is slided across the big image. For instance, if we train a convolutional network with images of size 32×32 that via pooling get reduced to feature maps of size 4×4 in turn passed to the (converted) fully connected layers to obtain a score vector, then when using a 96×96 image as input to the same convolutional network it will get reduced to feature maps of size 12×12 and the fully connected layers will output a matrix of scores of size 9×9 (for each class), i.e, it slides the 4×4 fully connected layers across the 12×12 feature maps. Averaging each score matrix we obtain the final scores for the big image. We could have also set a stride of 4 in the first (converted) fully connected layer to get score matrices of size 3×3 for each 9 non-overlapping 32×32 partitions of the original image. It works exactly as if we applied the convolutional network to the original image at a stride of 32 but does all computations in just one pass. This way we can reuse a pretrained network to classify images of bigger size.

Transfer learning is a related method where we train a convolutional network on images from a specific domain and later use it to extract features on images from a different domain. It could also be used as a initialized network which is fine tuned with examples of the new domain.

The loss function for a multiclass convolutional neural network is similar to that for a regular neural network (Equation 2.12) except that now the convolutional network defines the vector score $h_\Theta(x)$.

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\log\left(\frac{e^{h_\Theta(x^{(i)})_{y^{(i)}}}}{\sum_{j=1}^{K} e^{h_\Theta(x^{(i)})_j}}\right) + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s^{(l)}}\sum_{j=1}^{s^{(l+1)}}\left(\Theta_{ij}^{(l)}\right)^2 \tag{2.13}$$

Furthermore, this loss function is still differentiable with respect to Θ and thus we can train the entire network via gradient descent. We use backpropagation to calculate the gradients of the loss function.

2.5 Practical Deep Learning

In this section we collect some recommendations for constructing convolutional networks as well as efficiently training deep neural networks. These are intended to be specific to this project but most will also be useful in similar projects.

Image preprocessing Some standard processing for images.

- Images are cropped to contain only the relevant parts of the image, denoised, enhanced and optionally downsampled to maintain the input size fixed and manageable.
- Each image feature (the raw pixels) is zero centered by substracting its mean across all training images. Normalization scales the already zero-centered features to range from $[-1 \dots 1]$ by dividing them by its standard deviations. Feature normalization is not striclty neccesary but still customary [26].
- The test data should not be used to calculate any statistic used to preprocess the training data. Furthermore, these same statistics (calculated from the training data) should be used when normalizing the test data [26].

Convolutional network architecture We offer some guidelines for designing convolutional network architectures and some standard values for various hyperparameters.

- It is always better to select a complex network architecture which is flexible enough to model the data and manage overfitting with regularization rather than an architecture which is not powerful enough to model the data [40, 29].
- Although, theoretically, neural networks with a single hidden layer are universal approximators provided they have enough units ($\mathcal{O}(2^n)$ where n is the size of the input), in practice, deeper architectures produce better results using less units overall. This insight holds for convolutional networks [8].

- As a rule of thumb for big data sets, use 8-20 layers (not counting pooling or ReLU layers). For small data sets, use less layers or transfer learning. “You should use as big of a neural network as your computational budget allows, and use other regularization techniques to control overfitting.” [26]

- Use the number of parameters rather than the number of layers or units as a measure of the architecture’s complexity.

- Use 2-3 `CONV -> RELU` pairs before pooling (N above) [26]. Pooling is a destructive operation and having two convolutional layers together allows them to pick up more complex features.

- Use 1-5 `[CONV -> RELU]+ -> POOL` blocks (M above). This number depends on the complexity of the features expected in the data and the computational resources available. In a way, this regulates how much representational power will the architecture have. It also decides how much the volume is subsampled.

- Use less than 3 `FC -> RELU` pairs before the output layer (K above) [26]. When the volume arrives to the fully connected layers it has shrinken enough and using more fully connected layers risks overfitting.

- The number of feature maps per convolutional layer is set according to the expected number of features. This is similar to the number of units in a regular neural network. A common pattern is to start with a small amount of feature maps and increase them layer by layer [51].

- The number of feature maps per fully connected layer, or equivalently the number of units per fully connected layer, decreases from the number of units in the last convolutional layer (the number of units in each feature map times the number of feature maps) to the number of classes. For instance, having a convolutional network with two fully connected layers and 10 possible classes if the last convolutional layer produces a volume of size $8 \times 8 \times 512$ (8192 units), the first fully connected layer could have size $1 \times 1 \times 2048$ and the second (output) layer $1 \times 1 \times 10$.

- Use 3×3 filters with stride 1 and zero-padding 1 or 5×5 filters with stride 1 and zero-padding 2. This preserves the spatial dimensions of the volume and works better in practice [53]. When training on big images, the first convolutional layer uses bigger filters [26].

- Use 2×2 pooling with stride 2. Both this pooling and the overlapping version presented in Section **??** produce similar results. Pooling divides the spatial dimensions of the volume in half [29].

- Use square input images (width = height) with dimensions divisible by 2. The dimensions should be divisible by 2 at least as many times as the number of pooling layers in the network.

- Convert fully connected layers into convolutional layers.

Hyperparameter search We deal here with choosing hyperparameters other than those of the network architecture.

- Use a single sufficiently large validation set (20-30$\%$) rather than cross validation [8]. For small data sets, cross validation can give better estimates and is preferred [40].
- Use random search rather than grid search. Random search draws each parameter from a value distribution rather than a set of predefined values. [10]
- Train each parameter combination for 1-2 epochs to narrow the search space. Later, train for more epochs on the refined ranges. Full convergence is not needed to make a decision on the hyperparameters [26].
- Hyperparameters related to the convolutional architecture, e.g., number of layers, number of feature maps, filter sizes, etc., are set manually (as explained above) rather than using a validation set.
- There are several hyperparameters to set: initial learning rate α, learning rate decay schedule, regularization strength λ, momentum μ, probability of keeping a unit active in dropout p, mini-batch size and type of image preprocessing.
- Theoretically we could fit all the hyperparameters using a validation set but in practice it is computationally unfeasible and could result in overfitting the hyperparameters to the validation data [13].
- Set α, λ and optionally the type of preprocessing using a validation set. Other hyperparameters can be set to a sensible default.
- The learning rate α is “the single most important hyperparameter and one should always make sure that it has been tuned” [7]. It ranges from 10^{-6} to 10^{0}. Use a log scale to draw new values ($\alpha = 10^{unif(-6,0)}$ where $unif(a, b)$ is the continous uniform distribution) [26].
- The regularization strength λ is usually data (and loss function) dependant. It ranges from 10^{-3} to 10^{4}. Search in log scale ($\lambda = 10^{unif(-3,4)}$).
- If the best values for a hyperparameter are found in the limit of the range, explore further. [7].
- Use standard image enhancements. If there are no standard methods, use the validation set to choose from the options.
- Halve the learning rate every time the validation error stops improving. To obtain a fixed number of epochs, train the network (with the obtained hyperparameters) and observe when the validation error stops decreasing [29].
- Use $\mu = 0.9$. When using a validation set try values in $\{0.5, 0.9, 0.95, 0.99\}$ [26].

- Use 0.9-1 probability p of retaining a unit in the input layer, 0.65-0.85 in the first 2-4 convolutional layers and 0.5 in the last convolutional layers and all fully connected layers [54]. Less dropout is used on the first layers because they have less parameters [26].
- Use mini-batch size of 64 or 32. A larger batch size requires more training time. It affects training time more than test performance [7].

Training Some general tips for efficiently training convolutional networks with million of parameters and very big data sets. Using these algorithms for small networks may be somewhat excessive but it will not hurt the performance.

- Randomize the order of the trainig examples before training. As we are using an stochastic estimator of the gradient this ensures the examples in each batch are sampled independently. Shuffling the examples after each epoch could also speed convergence [7].
- To estimate the number of examples needed to train a convolutional network divide the total number of learnable parameters by 25-100 (assuming some data augmentation). Some groups have been able to learn up to 40M parameters from as little as 60K training examples [17, 53].
- Weight initialization is very important for a proper convergence of the network. The current recommendation for ReLU units is to initialize each weight as a value drawn from a gaussian distribution $\mathcal{N}(\mu = 0, \sigma = \sqrt{2/n_{in}})$ where n_{in} is the fan-in of the unit, i.e., the number of inputs to the unit. Specifically, each filter weight could be initialized as `w = randn()*sqrt(2/nIn)` where `randn()` returns a value drawn from a standard normal distribution and `nIn` is the number of connections to this filter (9 for a 3×3 filter, for example). Weights for units in the fully connected layer follow the same formula. Biases can be initialized likewise or to zero [22].
- Use mini-batches to compute the gradient. Using the entire training set to compute the gradient of the loss function takes a big amount of computation and points to the steepest descent direction locally but may not be the right direction if the update step is large. Using mini-batches allows us to make more updates, more frequently which results in faster convergence and better test results [7].
- Use Nesterov's Accelerated Gradient (NAG) to update the weights. It is a modified version of gradient descent which has shown to work slightly better for certain architectures [9]. Stochastic Gradient Descent with Momentum (SGD+Momentum) is also a viable option [26].
- Use dropout as a complement to l_2-norm regularization. Dropout usually improves results but it may slow network convergence [29].
- Store the network parameters regularly during training. Once per epoch should be enough but it depends on the number of parameters and size of the data. This allows you to come back to different versions of the network and select the one with the best overall validation/test error or one with some special characteristics [8].

- Stop the training process when the validation or test error has not improved since the last learning rate reduction. At this point gradient descent may not have converged but the validation error has and will start to increase (overfit) [7].
- Use the validation or test error to select the best parameters for the network from those stored [8].
- If you use the test set to refine a model, shuffle the entire data set and choose a diferent training and test set for the new model. Otherwise, you run the risk of overfitting to the test set [40].

Sanity checks Some simple checks to make sure the training is working properly.

- After weight initialization, the network should predict similar scores for each class (uniform probability) and have a loss function (without regularization) equal to $-\log(1/K)$. You can check this by running a test on a small set of examples. Adding regularization should increase the loss [26].
- If you implement back propagation manually or believe it may not be working properly you can run a gradient check. Gradient checks compare the analytic gradient produced by backpropagation with a numerical gradient produced by a finite difference approximation [26].
- Train the network with a very small subset of data (20 examples, for instance) and make sure it produces zero loss (without regularization). If it cannot overfit a tiny subset of examples the model is too simple [40].
- During training, the training loss should always decrease or only slightly increase. Otherwise, gradient descent may not be working properly either because of an implementation error or poorly tuned hyperparamters (high learning rate, low momentum) [26].
- Monitor the training and validation loss during training to identify overfitting and underfitting. Underfitting is characterized for a high training loss, overfitting is characterized for a big gap between training and validation (or test) loss [40].

Data augmentation One of the easiest ways to reduce overfitting in image data is to generate additional examples from the original data by applying some simple label-preserving transformations. Data augmentation allows the network to see different views of the same object thus enabling it to identify features that do not depend on the invariance introduced by the transformations. For instance, if we present it with images of a book on different rotations, we expect it to learn to identify a book no matter its position.

- There are many transformations one can apply: rotations, translations, horizontal and vertical reflections, crops (sample patches of the original image), zooms, etc. For color images, adding some noise to (jittering) the colors is also a valid transformation.

- Exploit the invariances you expect in the data set. For instance, galaxies are rotation invariant given that in space there is no up or down [17] but trees are not as it is rare to see an upside down tree.
- When combining different transformations in the same image be careful to preserve the original label. An overly modified image may lose its meaning.
- Most transformations are affine in the geometric plane and can be combined into a single one. If you plan to apply various transformations to the same image, applying a single affine transformation is faster and reduces information loss [17].
- Generate the augmented images during training. This saves storage and can be performed alongside the training [29].
- Data augmentation can also be used at test time by presenting the network with various versions of the same image and averaging its predictions [29].

Unbalanced data Having very few examples of one class compared to the rest is common in practice. We offer here some advice to deal with unbalanced classes using standard convolutional networks. We note that there is no accepted way to manage this problem.

- For a binary classifier, if the positive class is the rare class use PRAUC as a performance metric. If you are reporting the F_1 score, you should select an appropiate threshold using a validation set [16].
- If using PRAUC or selecting the threshold with a validation set is impractical, a simple adjustment is to divide the predicted probabilities by their corresponding class priors and to renormalize the values.
- For multiclass classification, use the macro-averaged F_1 score, an average of F_1 scores per class, with validated thresholds [42]. A multiclass PRAUC also exists but it is not as easy to interpret.
- As the threshold setting does not affect training it can be fitted independently of other hyperparameters once the network has already been trained. If fitting more than one, consider each one as a separate hyperparameter and use random search to find the best combination.
- One of the preferred methods to learn with unbalanced data sets is to use a modified loss function which gives a higher weight to errors in the rare class so that during training errors in the rare class will produce higher learning in the network parameters. Specific knowledge of the domain is required to estimate the cost of each class of error.
- Oversampling and undersampling, repeating the examples of the rare class or discarding some examples from the dominant class, are discouraged because they either not add information or throw away some of it.

- Replicating rare examples (oversampling) is useful when the examples are very scarce and the classifier simply does not have enough data to learn. This could be achieved by balancing the classes on each mini-batch via stratified sampling or by augmenting the rare class more than the dominant class during data augmentation.
- Data augmentation differs from data replication in that it only tries to enrich the data set with invariant images but actually leaves the proportion of classes unchanged.

All convolutional networks The research community has been moving towards discarding the pooling layers and using all convolutional networks. This can be interpreted as letting the network learn the pooling operation. We offer a couple of guidelines for implementing all convolutional networks.

- Replace each pooling layer with a convolutional layer with as many feature maps as in the previous layer and filter size 2×2 with stride 2 for normal pooling or 3×3 with stride 2 for overlapping pooling [53].
- For small data sets pooling layers also work as a regularizer because they reduce the number of learnable parameters and replacing them with convolutional layers may not be convenient [26].

Transfer learning When we have a small data set we could use a pretrained convolutional network either as a feature extractor for the new examples or to provide initializations for the new convolutional network, this is called transfer learning. We offer some tips for using a pretrained model specifically for mammographic images.

- Using a convolutional network pretrained in natural images, such as the ImageNet database, CIFAR-10, CIFAR-100, etc., may not work for mammographic images because features useful for one kind of classification are not very useful for the other. Nonetheless, given that features become more specific at higher layers, we could discard the higher layers of the network and use only the cropped network [26].
- Depending on the amount of data that we have we could: (1) add some fully connected layers on top of the pretrained network and train only these new layers, (2) add some convoutional and fully connected layers and train these new layers or (3) add convolutional and/or fully connected layers and train the entire network [26].
- When training on a pretrained model or fine-tuning use an smaller learning rate than when training a network from scratch. Using a small learning rate assures that we do not disturb very much the already good network parameters. [26].

Software A short description of four of the most popular packages for deep learning. They are pretty similar in capabilities and availability (open-source).

- Caffe [25]: Caffe is an already mature deep learning framework developed in C++/CUDA by the Berkeley Vision and Learning Center (BVLC) and community contributors. It offers a command line, Python and Matlab interface, reference models and tutorials and very fast code with easy GPU activation.

- Theano [11, 6]: Theano is a Python library developed in Python/CUDA at the University of Montreal. It is tightly integrated with NumPy, performs symbolic automatic differentiation and uses the GPU to efficiently evaluate mathematical expressions involving multi dimensional arrays.
- Torch7 [15]: Torch is a scientific computing framework developed in C/Lua/CUDA at the IDIAP Research Institute. It offers n-dimensional arrays (tensors), automatic differentiation, a command line and Lua interface, GPU support and easy building of complex neural network architectures.

Lastly, we acknowledge that mammographic data is fairly different to that used in common object recognition tasks, for instance, labelling may not be as sharp or correct, images come in different sizes and ratios, image quality varies, the object to recognize may be very small in relation to the image, object localization may be a requirement, etc. and therefore some of the advice given above may prove counterproductive. When possible, design decisions should be based on the data and results obtained until that point.

2.6 Convolutional Networks in Breast Cancer Research

Traditional CAD systems for breast cancer detection (CADe) and diagnosis (CADx) are normally composed of various succesive stages: preprocessing and image enhancement, segmentation of suspicious regions, feature extraction from the segemented image patches, optionally feature selection and region classification using machine learning methods. [56] offers a good review of the current state of traditional CAD systems. Here we focus on the previous application of convolutional networks for breast cancer detection and diagnosis in mammographic images.

As explained in Section 2.1 there are two main signs of breast cancer: breast masses and clustered microcalcifications, however, their presence does not necessarily imply cancer because they could be benign lesions. We refer as detection to the task of classifying a lesion as present or absent in the image no matter its malignancy, for instance, classifying an image patch as either clustered microcalcifications or normal tissue, while we talk of diagnosis when classifying a lesion as either benign or malignant [e].

Overview The only work using convolutional networks to detect masses was presented in [48]. Although the network produced competitive results, research focused more on feature-based classifiers, such as regular neural networks, that produced better results using expert knowledge and advanced image techniques. Around the same time, [33] presented the first use of convolutional networks to detect individual microcalcifications. This work was followed up by [21] who looked to select an optimal convolutional network architecture for individual microcalcification detection. This optimized network was used in two CADe systems for clustered microcalcifications: one for film mammography [21] and one for digital mammography [20]. Both systems have a similar layout consisting of preprocessing, image

[e] Images without lesions are considered benign.

enhancement, segmentation of potential microcalcifications, false positive reduction, regional clustering and false positive reduction of clusters. The convolutional network plays a relatively small role as an individual microcalcification classifier during the first false positive reduction stage. Lastly, recently an unpublished report [1] uses a convolutional network for the diagnosis of breast lesions obtaining average results.

In summary, convolutional networks have been sporadically used for breast cancer detection and diagnosis but have only played a minor role. Other than the most recent application, researchers have used very small networks (2 or 3 layers, $<$10K parameters) withouth any recent advancement (ReLUs, pooling layers, regularization, GPU computation) trained on very little data to classify image patches that were preselected using advanced image techniques. For instance, convolutional networks have only been used to detect individual microcalcifications but not clusters of microcalcifications and although they have been used to detect breast masses, using a larger network could probably improve those results. They have not been used for any type of breast cancer diagnosis either.

In the following sections we expand on the work mentioned above. We present a thorough summary of the most important articles organized by topic.

Detection of masses

To the best of the author knowledge, the first attempt to use convolutional networks for breast cancer is reported in ”Detection of masses on mammograms using a convolution neural network” [57]. This 4-page article was later expanded in [48].

They used a small convolutional network (2 layers, $\sim$1K parameters) for the detection of masses [f]. Details of the best performing architecture can be found on Table **??**. The data set consisted of 672 manually selected possible tumors from 168 digitized mammograms: out of which 168 were real tumors and 504 were not. Background reduction was performed on each image (using a rather convoluted method). The original images (size 256×256 pixels equivalent to a 2.56 cm^2 area) were downsampled via non-overlapping average pooling (filter size 16×16) to size 16×16; downsampling to 32×32 pixels via an 8×8 average pooling was also tried but gave similar results. Furthermore, the data was augmented by using 4 rotations (0°, 90°, 180° and 270°) on each original image and on each horizontally flipped image (8 in total per each training image) [g]. The network was trained via batch gradient descent plus momentum and per parameter adaptive learning rate. Two sets of experiments were performed: first, the 16×16 image patches (and their 8 rotations) were used for training producing 0.83 AUC on the best architecture, later these image patches were complemented with 2 16×16 “texture-images” calculated using image techniques on the initial mass image (a $16 \times 16 \times 3$ input volume) producing 0.87 AUC, 0.9 sensitivity and 0.69 specificity with the best network architecture. The authors showed that the network architecture was not as important for performance as providing the network with texture information. The texture features give back some of the information lost during the downsampling, which explains the improvement observed. The authors also acknowledge that the network architecture is far

[f]They use the term mass to refer to tumors (either cancerous or non-cancerous) but not other kind of breast masses (cysts, fibroadenomas, fatty tissue, etc.). Thus, it actually detects tumors.

[g]The original article does not mention any data augmentation but it was probably performed given that they obtained the same results.

from optimal given its simplicity (one convolutional layer with three feature maps) and the incomplete hyperparameter tuning. A deeper network with more learnable parameters and a bigger input size could produce similar or better results without a need to include handcrafted texture features, which in theory could be learned by the network.

Detection of microcalcifications

The first use of convolutional networks to detect microcalcifications is reported in [33]. They performed various experiments on a small convolutional network (3 layers, ~5.4K parameters), details of the architecture are offered on Table **??**. The input size (16×16), number of convolutional layers (2) and kernel size (5×5) was obtained using a validation set, although only few options were explored: they tried input sizes of 8, 16 or 32, one or two hidden layers and kernel sizes of 2, 3, 5 or 13. A high sensitivity image technique was used to obtain a set of 2104 image patches (16×16 pixels equivalent to an area of 1.7 mm^2) of potential microcalcifications from 68 digitized mammograms; of these, 265 were true microcalcifications and 1821 were "false subtle microcalcifications". Prior to training, a wavelet high-pass filtering technique was used to remove the background. Each image was flipped horizontally and 4 rotations for each the original and flipped image were used for training (0°, 90°, 180° and 270°). The network reached 0.89 AUC when identifying individual microcalcifications and 0.97 AUC for clustered microcalcifications; results obtained with a 30 fold cross validation. More than two individual microcalcifications detected on a 1 cm^2 area is considered a cluster detection, the predicted probability for the cluster is the average of the probabilities of all suspect patches inside the 1 cm^2 area [h] Other performance metrics were not explicitly reported. This article showed that deeper networks, background removal and data augmentation improved results. Together with the previous article, it also proved that simple convolutional networks can be used for breast cancer lesion detection.

A convolutional network with a similar architecture (3 layers, ~4.5K parameters) was presented by the same group in [34]. It detects microcalcifications from 16×16 image patches that were pre-selected and preprocessed using the same image techniques as above. For these experiments, nonetheless, they used 38 digitized mammograms and extracted 220 true microcalcifications and 1132 false ones that were randomly divided into a training and test set of roughly equal sizes. The network obtained a 0.9 AUC for individual microcalcifications and 0.97 AUC for clustered microcalcifications (also evaluated as in [33]). It showed that a convolutional network outperforms a regular neural network and a DYSTAL network in the clustered microcalcifications detection task when using raw pixels as input features.

[21] used simulated annealing (AUC as the objective function) to find the optimal number of feature maps and filter sizes for each of the two layers of a convolutional network used to detect clustered microcalcifications. The convolutional network was part of a bigger CAD system that first identifies the breast area and enhances it via a bandpass filter, later potential microcalcifications are segmented using adaptive thresholding methods, these suspected areas are then filtered by a rule-based classifier that uses the contrast, size and signal to noise ratio (SNR) of the microcalcifications, the remaining image patches are fed to the convolutional network for final classification and lastly the individual microcalcifications are clustered to

[h] This method is not clearly explained either in this article or [34] so this interpretation may be incorrect. The way this evaluation is performed greatly affects the validity of the reported results.

obtain the detected clustered microcalcifications. The convolutional network was trained on 1117 image patches (16×16 pixels equivalent to 1.6 mm^2) obtained as explained above from 108 digitized mammograms without data augmentation. The best network found had 14 feature maps on the first convolutional layer with filter size 5×5 and 10 on the second layer with filter 7×7 ($\sim$7.6K parameters in total). The search ranges of each of these hyperparameters is not provided nor the specifics of the simulated annealing or the network training. The CAD with the optimized network reached 84.6% sensitivity at 0.7 false clusters detected per image. Other performance metrics were not provided. The authors show that optimizing the architecture of the convolutional network significatively improves the CAD performance. Nonetheless, given the small training set and incomplete hyperparameter search these result may be invalid.

[20] developed a complete CAD system for the detection of clustered microcalcifications in digital mammograms. This system is an evolution of that presented in [21] which was tailored to film mammograms. It consists of six stages: preprocessing, image enhancement via a box-rim filter, segmentation of potential microcalcifications using thresholding methods, classification of individual candidates using rule-based classifiers and a convolutional network, regional clustering via neighborhood growing and final classification of clusters using stepwise LDA feature selection and classification. We will focus on the convolutional network. It had the same architecture as before (shown in Table **??**) and was trained on around 500 16×16 pixels image patches obtained from 48 digital mammograms: half of which were true microcalcifications while the other half were false microcalcifications. Its threshold was manually set to 0.4. The convolutional network reached 0.96 AUC for the detection of individual microcalcifications.

[19] compared the performance of both CAD systems in film mammograms and digital mammograms. They found that the performance in film mammograms is significatively better than in film mammograms which is intuitive given that digital mammograms have less noise and thus segmentation and feature extraction are sharper. Moreover, the convolutional network will be able to pick up more subtle details from the image without a need for further image enhancement.

Diagnosis of lesions

A recent unpublished report [1] uses a big convolutional network (8 layers, $\sim$3.6M parameters) with recent features to diagnose lesions as benign or malignant. Clustered microcalcifications and masses are segmented from 8752 mammograms obtained from a public database (DDSM). Two to three square images (64×64) are randomly sampled from each lesion and each of these is rotated 8 times (at 45° steps) producing 50K image lesions in total. No further preprocessing is performed. The convolutional network obtains 69.8% accuracy on the task. Other performance metrics are not provided. This is a project report for a course in Convolutional Networks [26] and may be incomplete or incorrect. We acknowledge it here for completeness.

Article	Goal	Architecture	Volumes	Filters	# Params
[48]	Detect masses	`INPUT -> CONV -> SIGMOID -> FC -> SIGMOID`	$16\times16\times3$ $7\times7\times3$ $1\times1\times1$	10×10 7×7	1047
[33]	Detect individual microcalcifications	`INPUT -> [CONV -> SIGMOID]*2 -> FC -> SIGMOID`	$16\times16\times1$ $12\times12\times12$ $8\times8\times12$ $1\times1\times2$	5×5 5×5 8×8	5436
[34]	Detect individual microcalcifications	`INPUT -> GAUSSIAN -> [CONV -> SIGMOID]*2 -> FC -> SIGMOID`	$16\times16\times1$ $12\times12\times10$ $8\times8\times10$ $1\times1\times2$	5×5 5×5 8×8	4530
[21]	Detect individual microcalcifications	`INPUT -> [CONV -> SIGMOID]*2 -> FC -> SIGMOID`	$16\times16\times1$ $12\times12\times14$ $6\times6\times10$ $1\times1\times1$	5×5 7×7 6×6	7570
[20]	Detect individual microcalcifications	`INPUT -> [CONV -> SIGMOID]*2 -> FC -> SIGMOID`	$16\times16\times1$ $12\times12\times14$ $6\times6\times10$ $1\times1\times1$	5×5 7×7 6×6	7570
[1]	Diagnose lesions	`INPUT -> [CONV -> RELU]*2 -> POOL -> [CONV -> RELU -> POOL]*3 -> [FC -> RELU]*3 -> SOFTMAX`	$64\times64\times1$ $64\times64\times32$ $32\times32\times64$ $16\times16\times128$ $8\times8\times256$ $4\times4\times512$ $1\times1\times256$ $1\times1\times64$ $1\times1\times2$	3×3 3×3 3×3 3×3 3×3 4×4 1×1 1×1	3.68M

Table 2.2: Architectures of the different convolutional networks used for breast cancer detection and diagnosis.

Chapter 3

Solution Model

Design decisions, its rationale and implemetation notes. this in ther and that in there

3.1 Operationalization

We document here how we tranform/reduce the breast cancer detection and diagnosis task into a machine learning task able to be taken by convolutional networks, i.e., how we produce a data set with m inputs $x^{(i)} \in \mathbb{R}^n$ and m corresponding labels $y^{(i)}$. We use this notation throughout this section.

3.1.1 Database

There are many publicly available mammography databases and many more which are private. Given the size of the expected network architecture and the data thirst of convolutional networks we focus only on the bigger databases. We also need pixel-level labels, i.e., lesions to be marked on each mammogram; this is generally made by expert radiologists drawing the boundaries of the lessions in the mammograms. Furthermore, we prefer to have good contrast resolution, the number of gray colors represented per pixel, and good spatial resolution, the area represented per pixel: at least 12-bit images ($10^12 = 4048$ gray values per pixel) with 0.1-0.15 mm maximum pixel size. Greater constrast resolution means that more brightness values are captured per pixel while greater spatial resolution means that hopefully more detail is included in the image. Most mammography databases including all described below satisfy these conditions.

The Digital Database for Screening Mammography (DDSM) [23] is arguably the most popular database used for CAD development. It is composed of around 10.5K digitised film mammograms from 2620 patients. Mammograms are either 12-bit or 16-bit images with 0.05 mm spatial resolution. Age and breast density of each patient is provided. Each lesion boundary is specified along with its information: type, assesment, subtlety and malignancy.

The BancoWeb database [39] consists of around 1.5K digitised film mammograms from 300 patients although they claim other 5K images stored internally "are being progressively transferred to the online database" [a]. Mammograms are 12-bit images with 0.075 or 0.15 mm

[a]This claim was made back in 2011 so we expect it to be done by now.

pixel size. At the time of publishing (2011) only very few lesions had been marked in the mammograms and it was impossible to review the current state of the database given that its webpage was not accesible online, which could be a sign of permanent closure. The only advantage of this database and the reason we include it here is that it was collected in Brazil and may be useful to test our CAD in Latin American patients.

The Breast Cancer Digital Repository (BCDR-DM) consists of 3.6K digital mammograms from 724 patients; this number was obtained from its website (`bcdr.eu/information/about`) which also states the database is still in construction and it is expected to have mammograms from 2000 to 3000 patients. Mammograms are 14-bit images with good spatial resolution [b]. Each lesion outline is marked in the image along with its assesment and other relevant clinical data. They also have a fairly big repository of digitised film mammograms called BCDR-FM (3.7K) but at lower resolutions.

Another small digital mammogram repository is called INbreast [36]. It consists of 410 digital mammograms from 115 patients. Each mammogram is a 14-bit image with 0.7 mm spatial resolution. Lesion boundaries are accurately marked and its information is also included. This could be used in conjunction with the BCDR-DM repository.

Finally, [60] used a private repository of around 6.5K digital mammograms obtained from 1120 patients. Specifics of contrast and spatial resolution are not provided but they are most probably good enough. Lesions are marked (with a circle) on the mammograms and lesion and patient information is provided. Even though this is a private repository of the University of Pittsburgh, if needed, we could ask them for access to it. This may not be plausible given the complications of sharing personal (granted anonymized) information and the size of the database.

Decision We have decided to use digital mammograms over film mammograms. Digital mammograms are sharper and do not have marks, stamps or other artifacts present in digitised film mammograms. On the downside, because the technology is newer it may be harder to obtain big databases and given its higher resolution they are normally heavier in terms of disk space. We believe the network will be able to pick up better features from the higher quality images and that the size on disk will not be a trouble given the storage availability on current computers. A small number of examples in the database is a big problem and some alternatives are offered below in case the network is not able to learn with the available data. For these reasons, we have decided to use initially the BCDR-DM and INbreast databases.

Alternatives We hope to obtain enough examples after cropping the mammograms into smaller image patches and applying some data augmentation to them (rotation and horizontal flipping). In case this is still not enough we could try various things: (1) obtain more labelled data from other sources, (2) reduce the complexity of the architecture to have less parameters to learn, (3) be more aggresive with the data augmentation, (4) pretrain the network with unlabelled digital mammograms which may be easier to get, (5) use film mammograms to pretrain the network and fine tune it on digital mammograms and (6) use an already pretrained network in other similar domains and fine tune it with digital mammograms.

[b]The webpage does not explicitly states the image's spatial resolution but judging by the size of the entire images it is good enough.

Another option is to use only digitised film mammograms for the entire project but this will produce networks which expectedly produce bad results in digital mammograms [60] and seems like a step in the wrong direction given the clear trend of hospitals replacing film mammography by digital mammography. A final option is to join film and digital mammograms into a single data set, this may or may not work given the difference between them but will most probably decrease the quality of results on digital mammograms when compared to a network trained only on digital mammograms.

3.1.2 BCDR-DM

Yet to write

3.1.3 Image retrieval

We document here the decisions taken to obtain the small image patches x and its respective labels y from the chosen databases.

Image dimensions We use square image patches because they are common in practice and simplify data augmentation. To define the size we have to consider two aspects: keeping a manageable input size for the network (in pixels) and capturing the entire lesion in the image patch (in mm).

The smallest microcalcification worth considering could be as small as 0.16 mm [34], thus the spatial resolution should be at most 0.16 mm. The standard definition of a cluster of microcalcifications is of 5 or more inside a 1 cm^2 area [50], thus the entire image patch should cover at least a 1 cm^2 area. Using an image patch of 64×64 pixels we cover an image area of 1 cm^2 with a pixel size of 0.156 mm. Mass sizes (length of the long axis) vary from 5 mm to 20 mm [48] [c] There is not really any restriction on spatial resolution other than it being good enough to capture texture information. Again, using an image size of 64×64 pixels we can cover a 4 cm^2 area (2 cm per side) with a spatial resolution of 0.313 mm.

The low spatial resolution (big pixel sizes), however, reduces the quality of the input images. An alternative could be to use 127×127 image patches with 0.079 mm and 0.157 mm pixel sizes for microcalcifications and masses, respectively, aplying a bigger filter on the first convolutional layer, for instance, a 5×5 filter with stride 2 and padding 2. This allows us to have bigger input images with a negligible increase in the number of parameters. Whether the results improve or not is not clear at this point.

Although we use the same input size (either 64×64 or 127×127) for microcalcifications and masses they do not cover the same area in the mammogram. We need to use two different sizes because if we preserve the spatial resolution of microcalcifications, the 1 cm^2 area would not be able to contain the entire mass meanwhile if we use a resolution closer to the one used for masses, small microcalcifications will dissapear and the 4 cm^2 area would have way too much noise compared to the size of the cluster of microcalcifications.

[c] Bigger masses are easily detectable by touch and thus less important for our purposes.

Cropping To obtain the image patches from the entire mammogram we slide a square window across the mammogram similar to the way a convolutional filter moves accross an image and store the image patch directly beneath it. This generates a big number of small patches from each mammogram and takes advantage of the translational invariance of our data, i.e., a breast mass will continue to be a breast mass no matter its position in the image patch.

An alternative is to sample the desired number of image patches at random positions from the mammograms.

Stride We chose a stride of 3 mm. This is midway between a very small stride, say 0.05 mm, which produces many image patches with maximum overlapping and a big stride, say 1 cm, which produces fewer patches with little to no overlapping. We use a rather small stride to have a lesion appear in various image patches (although in a slightly different place in each one) and to produce a good number of patches from the original image (around 1K per mammogram). We use no padding.

When sliding the window starting from the upper left corner of the original image it is possible that due to a dimension mismatch pixels in the rightmost and bottom strips do not appear in any image patch, this is not a problem given that the lost strips are very thin (< 3 mm) and they are normally part of the black background.

Background Mammograms capture images of the breast against a black background which covers a good part of the mammogram. We delete any image patch which is 25% or more black. No important information is lost in this process because the same part of the breast which appears in a deleted patch also appears on other image patches with less black background. A remaining question is how will the trained convolutional network react when presented with an all black input, for example, when slided across the background of a test mammogram. In practice, however, this is not relevant because it is clear that no lesion could occur outside the breast.

An alternative is to preserve all images and let the network learn that black images are negative examples but this seems rather wasteful.

Assigning labels Once each image patch is obtained we need to assign labels to it. All image patches are initially labelled as negative (or no lesion) and only those where a lesion is present are labelled as positive. There are many ways to define the presence of a lesion in an image: (1) if a percentage of the lesion, say 70% or more, appears in the image, (2) if a percentage of the image is covered by the lesion and (3) if a part of the lesion appears in the middle of the image.

We have chosen the last option to define the presence of a lesion because of three reasons: it is simple to implement, it somehow includes the other methods given that when a lesion appears on the middle of the image patch the rest of it will probably also appear on that patch and finally it encourages the convolutional network to output true only when the lesion appears in the center of the patch but not anywhere else which may give us more granular results when using it on the entire mammogram. The downside is that when a lesion is found on the outside of the image patch (in a corner, for example) it will be labelled as a negative example in the training set and may difficult the learning because even though the lesion is there we are training the network to answer negatively; if this effect actually occurs is not clear. [14] uses this method to label its image patches.

Using the first or second option is a viable alternative although they come with their own caveats, for instance, it could be hard to calculate the area of irregular objects or the lesion could be so big that even covering the the entire image patch area it would still not account for 70%.

Label information We will use the type of lession (mass, clustered microcalcifications or normal) and the malignancy (bening, malignant or nothing) to train our networks.

Databases normally offer adittional information such as age of the patient, breast density, family clinical history, assesment of the subtlety and malignancy of the lesion, etc. This information could be used as complementary features before classification or as labels for the network. In this stage we use only the mammograms with the labels stated above (binary classification).

Image enhancement In theory we want to perform classification on the raw images (without any preprocessing) so we store the raw image patches and their labels as the base training set and perform any image enhancements during training whenever possible. There is a set of simple contrast adjustments that could be applied: normalization assigns 0 to the minimum gray value in the image and 255 (or the maximum available value) to the maximum gray value in the image and stretches the rest of the values linearly, background reduction plus normalization is similar except that all values below a given threshold (the mean of all pixel values in the image) are mapped to zero and the rest is normalized effectively reducing all small variations in the background to black and histogram equalization which tries to distribute the gray values evenly on the histogram of the image. An example of each method is shown in Fig. 3.1.

We use background reduction plus normalization because it increases the signal to noise ratio, i.e., highlights lesions over normal breast tissue. On the downside, it also highlights dense structures (which could increase false positives) and may destroy important texture information by blending it with the background; using normalization only could produce better results. Unpreprocessed images seem too noisy and histogram equalization is too destructive for our purposes.

Data augmentation We augment each enhanced image by using 4 rotations (at 0°, 90°, 180° and 270°) of both the original image and a horizontally flipped version of it, thus we increase our data set by a factor of 8. Both rotations and reflections preserve the original label. In principle it is not neccesary to store the augmented images because they can be easily generated during training but if the disk space is not prohibitive explicitly storing them simplifies training.

Resizing We have to resize the images to achieve the desired patch size. There is a couple of decisions to take in this part: the type of interpolation to use and whether image enhancement should be performed before or after resizing. After some experiments neither decisions

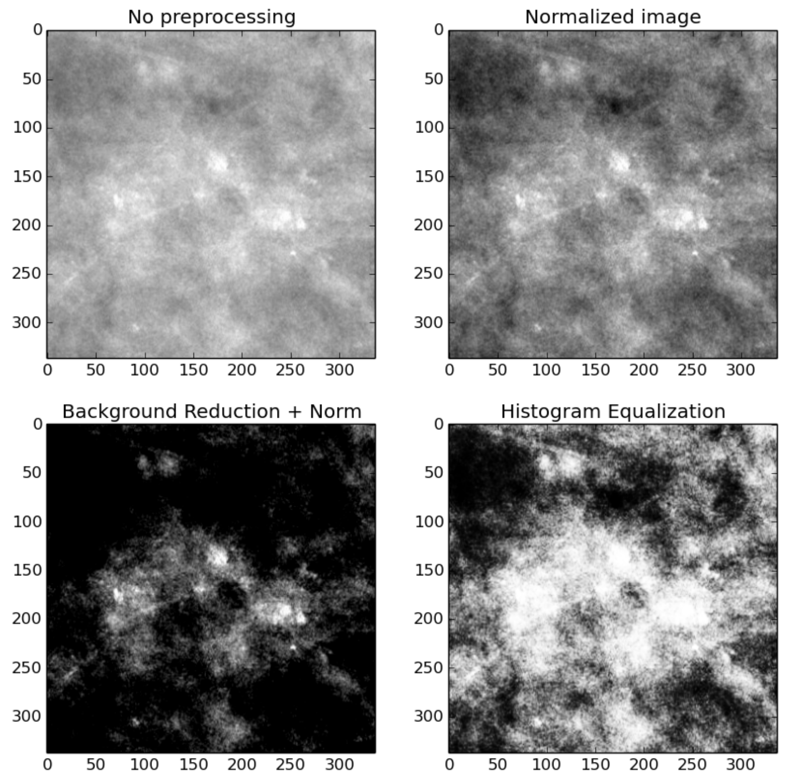

Figure 3.1: Example of simple contrast adjustment techniques applied to an image with a cluster of microcalcifications. Normalization takes the range of gray values and stretchs it up to linearly cover the entire range available. Background reduction assigns 0 to every pixel below the mean of the pixel values and applies normalization. Histogram equalization distributes the gray values more evenly in the histogram of the picture.

proved to be very important for the resulting image patches. We choose the Lanczos interpolation recommended for downsizing in the PILLOW Python Image Library. Enhancements are executed on each particular patch before being scaled to their final size. Figure 3.2 shows the effect of using the Bicubic or Lanczos interpolation scheme for resizing both before and after enhancement. Results are similar under all configurations.

Storage For each mammogram we will generate a 3-dimensional matrix ($x \times 64 \times 64$) containing all x preprocessed patches obtained from the mammogram and a 2-dimensional matrix ($x \times 4$) containing its respective labels. This matrices will be stored with the same name (plus a suffix) and on the same directory as the original image.

3.2 Training

3.2.1 Data set

Yet to write

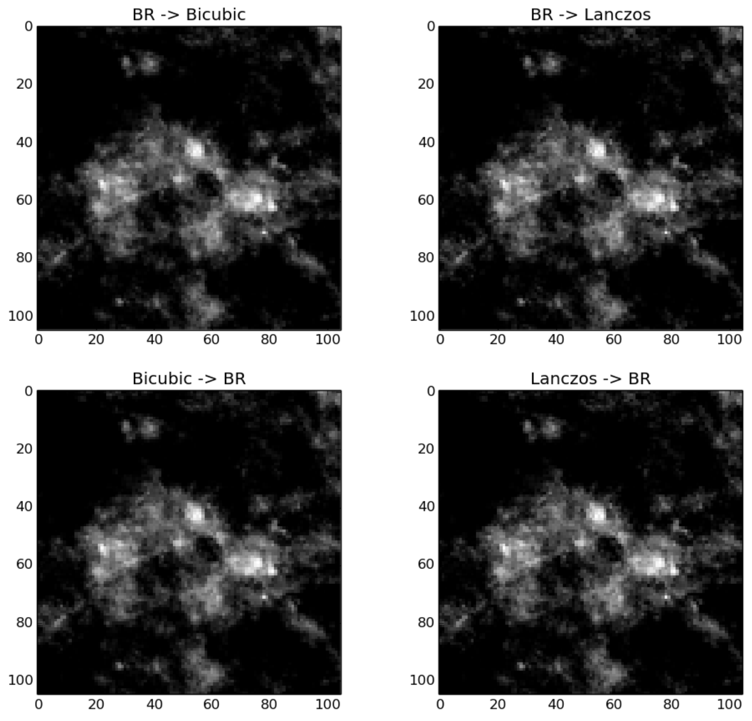

Figure 3.2: Example of different resizing schemes applied to an image with a cluster of microcalcifications. Bicubic and Lanczos inteprolation is used for resizing an original 317 pixel image to 105 pixels before and after enhancement (background reduction plus normalization) as shown by the arrows in the subtitles. Results are virtually undistinguishable.

3.2.2 Hardware

Training deep neural networks is computationally intensive and requires equipment with powerful GPUs. We enlist here the resources available for this thesis.

PC	GPU	HD	CPU	RAM	#
Personal	Nvidia NVS 5400M 96 cores, 1GB, 2.1 compatibility, 29 GB/s	57 GB (36 free)	i5-3210M 2.5GHz	4 GB	1
A4-401	Nvidia Quadro K620 384 cores, 2GB, 5.0 compatibility, 29 GB/s	240 GB (230 free)	i5-4570 3.2GHz	8 GB	27

Table 3.1: Available hardware for experiments

3.2.3 Architecture

Using the advice in Section **??** we decided to use a simple network with six convolutional layers and two fully connected layers with the following architecture:

Layer	Filter	Stride	Pad	Volume	Params
`INPUT`	-	-	-	$127 \times 127 \times 1$	-
`CONV -> RELU`	5×5	2	2	$64 \times 64 \times 64$	1 625
`CONV -> RELU`	3×3	1	1	$64 \times 64 \times 64$	36 928
`MAXPOOL`	2×2	2	0	$32 \times 32 \times 64$	-
`CONV -> RELU`	3×3	1	1	$32 \times 32 \times 96$	55 392
`CONV -> RELU`	3×3	1	1	$32 \times 32 \times 96$	83 040
`MAXPOOL`	2×2	2	0	$16 \times 16 \times 96$	-
`CONV -> RELU`	3×3	1	1	$16 \times 16 \times 128$	110 720
`CONV -> RELU`	3×3	1	1	$16 \times 16 \times 128$	147 584
`MAXPOOL`	2×2	2	0	$8 \times 8 \times 128$	-
`FC -> RELU`	8×8	-	-	$1 \times 1 \times 512$	4 194 816
`FC -> SIGMOID`	1×1	-	-	$1 \times 1 \times 1$	513

Table 3.2: Architecture of the network used for experiments. It shows the filter, stride and padding used in each layer as well as the resulting volume and the number of learnable parameters per layer.

The first convolutional layer uses a 5×5 filter with stride 2 (padding 2) to reduce the input spatial size from 127×127 to 64×64. After that all filters are 3×3 with stride 1 (padding 1), which preserves the spatial size and the pooling is 2×2 stride 2 (padding 0) which reduces the spatial size by a half. This architecture has 4.63 million learnable parameters.

In case the input was size 64×64 pixels we could replace use a 3×3 filter with stride 1 in the first convolutional layer and leave everything else unchanged. For an all convolutional architecture we could replace all pooling layers by a 5×5 filter with stride 2 and use input images of size of 113×113 or 129×129.

3.2.4 Evaluation

When dividing the data set we make sure *all* image patches obtained from the same patient are assigned to either the training set or test set (not distributed) to avoid any possible overfit to the test set. Given that our data is unbalanced, with far more negative than positive examples, we use PRAUC (see Section **??**) to choose between models for hyperparameter selection and as an overall performance metric. Other metrics are also reported for completeness.

We could also evaluate the network on all augmentations of an image and output the average prediction; in theory, this would give us better results. For simplicity, we do not apply it for model selection.

For detection of lesions on entire mammograms we slide the trained convolutional network across the mammogram computing a per-pixel prediction. The generated heatmap preserves the size of the original mammogram (with some zero-padding) and can be presented side to side to the original mammogram as a CAD system. In case this heatmap is noisy (predictions changes abruptly from pixel to pixel) we could use a median or gaussian filter to smooth it out.

3.2.5 Software

We use Caffe [25] to train the networks and Python to develop any other tools (image retrieval and augmentation, model evaluation, figure generation, etc.).

3.3 Implementation

3.4 Implementation details

What library I used

3.5 Evaluation metrics

3.6 Presentation of results

Probabilities vs UNcorrected threshold vs Threshold and cluster extent correction vs threshold-free cluster enhancement vs, show different results at different thresholds data is smooth,

uncorrected threshold Present the probabilities of tumor on each pixel. anmd threshold it on a given prob nd delete any clusters less than x. Threshold free cluster enhancement (see module 29 in introduction to fmri) Choose a threshold and delete any clusters less than x...

we could also present a heatmap with the different probabilities in each pixel of the image, for instance on mammograms we could present a grayscale image of whether a lesion is present.

Chapter 4

(Experiment's title)Detection vs diagnosis masses or microcalcifications.

4.1 Experiment

Task selected. Architecture selected. Hyperparameters selected. results and discussion.

4.2 Results

4.3 Discussion

Chapter 5

Conclusions

5.1 Future Work

Bibliography

[1] AGARWAL, V., AND CARSON, C. Using deep convolutional neural networks to predict semantic features of lesions in mammograms. Project report, 2015.

[2] AMERICAN CANCER SOCIETY. Mammograms and other breast imaging tests. electronic, Atlanta, GA, 2014. Available online on `cancer.org/acs/groups/cid/documents/webcontent/003178-pdf.pdf`.

[3] AMERICAN CANCER SOCIETY. *Cancer Facts & Figures 2015*. American Cancer Society, Atlanta, GA, 2015. Available on `cancer.org/acs/groups/content/@editorial/documents/document/acspc-044552.pdf`.

[4] AREVALO, J., GONZALEZ, F. A., RAMOS-POLLAN, R., OLIVEIRA, J. L., AND GUEVARA LOPEZ, M. A. Convolutional neural networks for mammography mass lesion classification. In *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE* (Aug 2015), pp. 797–800.

[5] ASHRAF, A. B., GAVENONIS, S. C., DAYE, D., MIES, C., ROSEN, M. A., AND KONTOS, D. A multichannel markov random field framework for tumor segmentation with an application to classification of gene expression-based breast cancer recurrence risk. *Medical Imaging, IEEE Transactions on 32*, 4 (April 2013), 637–648.

[6] BASTIEN, F., LAMBLIN, P., PASCANU, R., BERGSTRA, J., GOODFELLOW, I. J., BERGERON, A., BOUCHARD, N., AND BENGIO, Y. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[7] BENGIO, Y. Practical recommendations for gradient-based training of deep architectures. *CoRR abs/1206.5533* (2012).

[8] BENGIO, Y. Deep learning workshop. online, April 2014. Available on `youtu.be/JuimBuvEWBg`.

[9] BENGIO, Y., BOULANGER-LEWANDOWSKI, N., AND PASCANU, R. Advances in optimizing recurrent networks. *CoRR abs/1212.0901* (2012).

[10] BERGSTRA, J., AND BENGIO, Y. Random search for hyper-parameter optimization. *J. Machine Learning Research 13*, 1 (February 2012), 281–305.

[11] BERGSTRA, J., BREULEUX, O., BASTIEN, F., LAMBLIN, P., PASCANU, R., DESJARDINS, G., TURIAN, J., WARDE-FARLEY, D., AND BENGIO, Y. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)* (June 2010). Oral Presentation.

[12] BREAST CANCER SURVEILLANCE CONSORTIUM. Performance measures for 1,838,372 screening mammography examinations from 2004 to 2008 by age–based on BCSC data through 2009. electronic, September 2013. Available on `breastscreening.cancer.gov/statistics/performance/screening/2009/perf_age.html`.

[13] CAWLEY, G. C., AND TALBOT, N. L. On over-fitting in model selection and subsequent selection bias in performance evaluation. *J. Machine Learning Research 11* (August 2010), 2079–2107.

[14] CIREŞAN, D. C., GIUSTI, A., GAMBARDELLA, L. M., AND SCHMIDHUBER, J. Mitosis detection in breast cancer histology images with deep neural networks. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013*, K. Mori, I. Sakuma, Y. Sato, C. Barillot, and N. Navab, Eds., vol. 8150 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013, pp. 411–418.

[15] COLLOBERT, R., KAVUKCUOGLU, K., AND FARABET, C. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop* (2011).

[16] DAVIS, J., AND GOADRICH, M. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning* (New York, NY, USA, 2006), ICML '06, ACM, pp. 233–240.

[17] DIELEMAN, S., WILLETT, K. W., AND DAMBRE, J. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society* (2015).

[18] FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics 36* (1980), 193–202. PMID: 7370364.

[19] GE, J., HADJIISKI, L. M., SAHINER, B., WEI, J., HELVIE, M. A., ZHOU, C., AND CHAN, H.-P. Computer-aided detection system for clustered microcalcifications: comparison of performance on full-field digital mammograms and digitized screen-film mammograms. *Physics in Medicine and Biology 52*, 4 (2007), 981.

[20] GE, J., SAHINER, B., HADJIISKI, L. M., CHAN, H.-P., WEI, J., HELVIE, M. A., AND ZHOU, C. Computer aided detection of clusters of microcalcifications on full field digital mammograms. *Medical Physics 33*, 8 (2006), 2975–2988.

[21] GURCAN, M. N., CHAN, H.-P., SAHINER, B., HADJIISKI, L., PETRICK, N., AND HELVIE, M. A. Optimal neural network architecture selection: Improvement in computerized detection of microcalcifications. *Academic Radiology 9*, 4 (2002), 420 – 429.

[22] HE, K., ZHANG, X., REN, S., AND SUN, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR abs/1502.01852* (2015).

[23] HEATH, M., BOWYER, K., KOPANS, D., MOORE, R., AND KEGELMEYER, W. P. The digital database for screening mammography. In *Proceedings of the Fifth International Workshop on Digital Mammography* (2001), M. Yaffe, Ed., Medical Physics Publishing, pp. 212–218.

[24] HOWLADER, N., NOONE, A. M., KRAPCHO, M. F., GARSHELL, J., MILLER, D. A., ALTEKRUSE, S. F., KOSARY, C. L., YU, M., RUHL, J., TATALOVICH, Z., MARIOTTO, A. B., LEWIS, D. R., CHEN, H. S., FEUER, E. J., AND CRONIN, K. A. SEER cancer statistics review, 1975-2011. review, National Cancer Institute, Bethesda, MD, April 2014. Available on `seer.cancer.gov/csr/1975_2011/`.

[25] JIA, Y., SHELHAMER, E., DONAHUE, J., KARAYEV, S., LONG, J., GIRSHICK, R. B., GUADARRAMA, S., AND DARRELL, T. Caffe: Convolutional architecture for fast feature embedding. *CoRR abs/1408.5093* (2014).

[26] KARPATHY, A. Convolutional neural networks for visual recognition course. online, May 2015. Available on `cs231n.github.io`.

[27] KERLIKOWSKE, K., HUBBARD, R. A., MIGLIORETTI, D. L., GELLER, B. M., YANKASKAS, B. C., LEHMAN, C. D., TAPLIN, S. H., AND SICKLES, E. A. Comparative effectiveness of digital versus film-screen mammography in community practice in the united states: A cohort study. *Annals of Internal Medicine 155*, 8 (2011), 493–502.

[28] KRIZHEVSKY, A. cuda-convnet2: Fast convolutional neural networks in c++/cuda. online, 2014. Available online on `github.com/akrizhevsky/cuda-convnet2`.

[29] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[30] LECUN, Y., BOSER, B., DENKER, J. S., HENDERSON, D., HOWARD, R. E., HUBBARD, W., AND JACKEL, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation 1*, 4 (Winter 1989), 541–551.

[31] LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (November 1998), 2278–2324.

[32] LINNAINMAA, S. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. Master's thesis, University of Helsinki, Helsinki, Finland, 1970.

[33] LO, S.-C. B., CHAN, H.-P., LIN, J.-S., LI, H., FREEDMAN, M. T., AND MUN, S. K. Artificial convolution neural network for medical image pattern recognition. *Neural Networks 8*, 7–8 (1995), 1201 – 1214. Automatic Target Recognition.

[34] LO, S.-C. B., LIN, J.-S. J., FREEDMAN, M. T., AND MUN, S. K. Application of artificial neural networks to medical image pattern recognition: Detection of clustered microcalcifications on mammograms and lung cancer on chest radiographs. *Journal of VLSI signal processing systems for signal, image and video technology 18*, 3 (1998), 263–274.

[35] MCCULLOCH, W. S., AND PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics 5* (1943), 115–133.

[36] MOREIRA, I. C., AMARAL, I., DOMINGUES, I., CARDOSO, A., CARDOSO, M. J., AND CARDOSO, J. S. Inbreast: Toward a full-field digital mammographic database. *Academic Radiology 19*, 2 (2012), 236–248.

[37] NATIONAL CANCER INSTITUTE. What you need to know about breast cancer. electronic, Bethesda, MD, August 2012. Available online on `cancer.gov/publications/patient-education/WYNTK_breast.pdf`.

[38] NATIONAL CANCER INSTITUTE. Mammograms. electronic, March 2014. Available on `cancer.gov/cancertopics/types/breast/mammograms-fact-sheet`.

[39] NEPOMUCENO MATHEUS, B. R., AND SCHIABEL, H. Online mammographic images database for development and comparison of cad schemes. *Journal of Digital Imaging 24*, 3 (2011), 500–506.

[40] NG, A. Machine learning course. online, December 2014. Available on `coursera.org/course/ml`.

[41] OEFFINGER, K. C., FONTHAM, E. T., ETZIONI, R., AND ET AL. Breast cancer screening for women at average risk: 2015 guideline update from the american cancer society. *JAMA 314*, 15 (2015), 1599–1614.

[42] ÖZGÜR, A., ÖZGÜR, L., AND GÜNGÖR, T. Text categorization with class-based and corpus-based keyword selection. In *Proceedings of the 20th International Conference on Computer and Information Sciences* (Berlin, Heidelberg, 2005), ISCIS'05, Springer-Verlag, pp. 606–615.

[43] PISANO, E. D., HENDRICK, R., YAFFE, M. J., BAUM, J. K., ACHARYYA, S., CORMACK, J. B., HANNA, L. A., CONANT, E. F., FAJARDO, L. L., BASSETT, L. W., D'ORSI, C. J., JONG, R. A., REBNER, M., TOSTESON, A. N., AND GATSONIS, C. A. Diagnostic accuracy of digital versus film mammography: Exploratory analysis of selected population subgroups in DMIST. *Radiology 246*, 2 (2008), 376–383. PMID: 18227537.

[44] PROVOST, F. Machine learning from imbalanced data sets 101. *Proceedings of the AAAI-2000 Workshop on Imbalanced Data Sets* (2000).

[45] ROSENBLATT, F. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Report (Cornell Aeronautical Laboratory). Spartan Books, 1962.

[46] RUMELHART, D. E., HINTON, G. E., AND WILLIAMS, R. J. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds. MIT Press, Cambridge, MA, 1986, pp. 318–362.

[47] RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATHY, A., KHOSLA, A., BERNSTEIN, M., BERG, A. C., AND FEI-FEI, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV) 115*, 3 (2015), 211–252. Available online on `arxiv.org/abs/1409.0575`.

[48] SAHINER, B., CHAN, H.-P., PETRICK, N., WEI, D., HELVIE, M. A., ADLER, D. D., AND GOODSITT, M. M. 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 (October 1996), 598–610.

[49] SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural Networks 61*, 0 (2015), 85–117.

[50] SICKLES, E., D'ORSI, C., AND BASSETT, L. *ACR BI-RADS Atlas, Breast Imaging Reporting and Data System*, 5th ed. American College of Radiology, Reston, VA, 2013, ch. ACR BI-RADS Mammography.

[51] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014).

[52] SKAANE, P., HOFVIND, S., AND SKJENNALD, A. Randomized trial of screen-film versus full-field digital mammography with soft-copy reading in population-based screening program: Follow-up and final results of oslo II study. *Radiology 244*, 3 (2007), 708–717. PMID: 17709826.

[53] SPRINGENBERG, J. T., DOSOVITSKIY, A., BROX, T., AND RIEDMILLER, M. A. Striving for simplicity: The all convolutional net. *CoRR abs/1412.6806* (2014).

[54] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I., AND SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15* (2014), 1929–1958.

[55] TAIGMAN, Y., YANG, M., RANZATO, M., AND WOLF, L. DeepFace: Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR)* (Columbus, Ohio, June 2014).

[56] TANG, J., RANGAYYAN, R., XU, J., EL NAQA, I., AND YANG, Y. Computer-aided detection and diagnosis of breast cancer with mammography: Recent advances. *Information Technology in Biomedicine, IEEE Transactions on 13*, 2 (March 2009), 236–251.

[57] WEI, D., SAHINER, B., CHAN, H.-P., AND PETRICK, N. Detection of masses on mammograms using a convolution neural network. In *International Conference on Acoustics, Speech, and Signal Processing, 1995. ICASSP-95.* (May 1995), vol. 5, pp. 3483–3486.

[58] WERBOS, P. J. *Beyond regression: new tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, Boston, MA, 1974.

[59] WIDROW, B., AND HOFF, M. E. Adaptive switching circuits. In *1960 IRE WESCON Convention Record, Part 4* (New York, 1960), IRE, pp. 96–104.

[60] ZHENG, B., SUMKIN, J. H., ZULEY, M. L., LEDERMAN, D., WANG, X., AND GUR, D. Computer-aided detection of breast masses depicted on full-field digital mammograms: a performance assessment. *The British Journal of Radiology 85*, 1014 (2012), e153–e161. PMID: 21343322.

Curriculum Vitae

Erick Michael Cobos Tandazo was born in Macará, Ecuador on March 10, 1992. He earned the Computer Science and Technology B.S. degree from the Instituto Tecnológico y de Estudios Superiores de Monterrey, Monterrey Campus in December 2014. He was accepted in the graduate program in Intelligent Systems in January 2015.

This document was typed in using $\LaTeX\,2_{\varepsilon}$[a] by Erick Michael Cobos Tandazo.

[a]The style file `thesisFormat.sty` used to set up this thesis was prepared by the Center of Intelligent Systems of the Instituto Tecnológico y de Estudios Superiores de Monterrey, Monterrey Campus