Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

National School of Engineering and Sciences

Graduate Program

Master of Science in Intelligent Systems

Thesis Proposal

Early Detection and Diagnosis of Breast Cancer Lesions in Digital Mammograms using Deep Convolutional Networks

by

Erick Michael Cobos Tandazo
1184587

Monterrey, N.L., May 14, 2015

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

National School of Engineering and Sciences

Graduate Program

The committee members, hereby, recommend that the master's thesis proposal presented by Erick Michael Cobos Tandazo be accepted to develop the thesis project as a partial requirement for the degree of **Master of Science**, with a major in:

Intelligent Systems

Thesis Committee:

Dr. Hugo Terashima Marín
Principal Advisor

Por definir
Committee Member

Por definir
Committee Member

Dr. Ramón Brena Pinero
Director of the Master's Program in
Intelligent Systems

May 14, 2015

Contents

Abstract

Breast cancer is one of the most common and deadliest cancer in woman around the world. The best tools used today for early breast cancer diagnosis are screening mammograms; mammograms are x-ray pictures of the breast used by radiologists to identify microcalcifications and breast masses, signs of early breast cancer development. Traditional computer systems use handmade features and complex image techniques to detect these lesions in mammographic images. In this work, we plan to use convolutional networks, a recent development in computer vision, which can automatically learn the relevant features for the classification task given enough training data. Convolutional networks have been used in some studies for breast cancer detection but we hope to introduce newer features and carefully tune the architecture to produce improved results. Additionally, this will be the first approximation to use deep learning techniques as part of an ongoing project in the institution which aims to develop a computer-aided diagnosis system for breast cancer. This thesis proposal is presented for approval to obtain the degree of Master of Science in Intelligent Systems.

1 Introduction

Automatic breast cancer diagnosis is a very difficult task for current computational systems. In this thesis, we apply deep learning techniques to digital mammographic images in order to improve the performance of such systems. We layout here the hypotheses, experiments and goals of our future research.

Breast cancer is a disease caused by abnormal breast cells which grow out of control forming tumors and invading surrounding tissue. It has the highest incidence rate of any cancer in the United States, an estimated 14.1% of all new cancer cases in 2015 will be breast cancer, and the third highest mortality of any cancer accounting for 6.9% of cancer related deaths. Among women it is by large the most commonly diagnosed cancer (28.6% of all cancers) and has the highest death rate (14.5%) besides lung cancer [American Cancer Society, 2015].

The recommended method for early breast cancer detection in aging women is to have regular screening mammograms. Mammograms are x-ray images of the breast used by specialists to look for signs of possible tumor formation. There are different lesions that can be found on a mammogram, we focus on two: clustered microcalcifications, tiny deposits of calcium which could appear around cancerous tissue, and breast masses, more direct signs of the existence of a tumor although they are very often benign. Most breast cancers can be detected with a mammogram.

In this work, we center on using mammograms to automatically detect microcalcifications and breast masses and predict the probability of breast cancer on the patient. Although manual examination of mammograms has a high sensitivity rate, automatic analysis could be used on places where expert radiologists are not available or it could be used by doctors as a second informed opinion or to help them decide to which regions of the image dedicate more time. With this motivation, a project that intends to design a computer aided diagnosis (CAD) tool for breast cancer has existed in this institution since 2007. This thesis falls under the scope of this project and will be its first approximation to use deep learning for breast cancer diagnosis.

Traditional CAD systems for breast cancer diagnosis work as a pipeline where each stage uses different computer vision and machine learning techniques. An standard pipeline will, for instance, preprocess the image, identify and segment the relevant parts of the picture, extract features from the segmented parts and train a classifier on the extracted features. Although

some successful systems are built in this manner, they have a few disadvantages: each stage is a separate component and hence work is needed on each of them to notably improve overall results, it is composed of dependent stages so that changes on one component can affect the performance of other parts of the system, it uses complex image vision techniques to segment the images and extract features which are difficult to handcraft and select, it requires expert knowledge to be properly tuned, among others.

We plan to investigate the potential of convolutional networks to replace some if not all of the stages of traditional image processing systems. Convolutional networks [Fukushima, 1980, LeCun et al., 1998], a natural extension to feedforward neural networks, are a statistical learning classifier which uses raw images as input and learns the important features for the classification task as it is trained. Convolutional networks are designed to work with minimally preprocessed images, can be trained to be rotational and translational invariant and perform segmentation, feature extraction and classification in one step. In our case, convolutional networks simplify the process of classification potentially reducing it to one component which is trained from labelled data and can be improved and properly tuned to obtain better results. Although there are some drawbacks with convolutional networks, they are the state-of-the-art technology for object recognition [Russakovsky et al., 2014] and we believe it is worthy to experiment with them.

We will start our experiments training a simple convolutional network in images preprocessed with different techniques including unpreprocessed images, later we will train a convolutional network using whole mammogram images, pretrain a convolutional network with a different image database and fine-tune it using our database and finally we will use the gathered knowledge to build an optimal convolutional network. We intend to learn whether convolutional networks can automatically preprocess mammographic images or else which is the best preprocessing for mammographic images, what is the best segmentation strategy we can use and whether we can achieve results similar to those of more traditional systems.

This document starts by offering an insight into the problem with traditional methods for image analysis in Section 2. It exposes the particular objectives and hypotheses of the thesis in Sections 3 and 4. Section 5 presents a comprehensive background of the scientific concepts used throughout the document and lastly a detailed methodology and work plan are shown in Sections 6 and 7.

2 Problem Statement and 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 from breast cancer; detection in its earlier stage (*in situ*) increases the survival rate to virtually 100% [Howlader et al., 2014].

With current technology, a high quality mammogram is "the most efective way to detect breast cancer early" [National Cancer Institute, 2014]. 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 [Breast Cancer Surveillance Consortium, 2013]. This high sensitivity is the product of the careful examination of the mammograms by experienced radiologists. A computer-aided diagnosis tool (CAD) could automatically detect and diagnose these abnormalities saving the

time and training needed by expert 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 diagnostic.

CAD systems are based on image and classification techniques coming from Artificial Intelligence and Machine Learning. Traditional CAD tools for breast cancer diagnosis are composed of three steps: feature extraction, feature selection and classification. In the feature extraction phase, the system uses filters and image transformations to preprocess the mammogram and find geometric patterns which are used to produce a set of features for the image; expert knowledge is sometimes used in this phase. Feature selection or regularization is used to focus only on the important features for the classification task. Once a vector of features is obtained for each image, an standard binary classifier can be used to perform the final detection or diagnosis. These techniques have been used for many years and are standard in the industry [1].

Despite its widespread use and efficiency, systems based on traditional computer vision techniques have various limitations that should be addressed to further improve its performance:

- There is no standard way of preprocessing mammograms. Some filters are commonly used but their performances can vary.
- It uses 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.
- It normally uses a small patch of the mammogram and makes a prediction on that patch but it does not consider the entire mammogram neither to make a prediction on the patient or to account for correlation between patches.
- To produce good results it requires knowledge in various fields such as radiology, oncology, image processing, computer vision, machine learning, etc.
- It is composed of many sequential steps. At each stage, there are many techniques from which the researcher can choose and many parameters which have to be estimated. This represents a cost in time and results as it is improbable that the optimal selection of techniques and parameters is achieved.
- As it is a complex system with different subsystems involved many other issues can arise such as non desired or unknown dependencies between subsystems, difficulty to localize errors, maintainability, etc.
- The techniques currently used are complex but the improvements achieved are not substantial. Much work is needed to make only incremental improvements and it is hard to know to which part of the system dedicate more resources.

This project will center around using Convolutional Networks, a recent development in computer vision, (see Section 5.4) to tackle some of these limitations, especifically automate preprocessing and feature extraction, use entire mammogram images and simplify the system pipeline by using a convolutional network as a replacement for many steps traditionally performed in succesion.

[1] See [Hernandez, 2014] for an example of a CAD system developed in this institution.

3 Objectives

The main goal of this work is to succesfully apply convolutional networks in digital mammograms to detect and diagnose breast cancer lessions, microcalcifications and breast masses, and to compare our results to those obtained by other groups working in convolutional networks for breast cancer diagnosis.

Particularly, there are various subgoals which we expect to achieve as the project advances:

- Develop a working pipeline for processing the mammographic images from our database and training a convolutional network. Essentially, this tool could also be used for other image classification tasks.
- Use a simple convolutional network to perform detection and diagnosis and study these initial results to guide further research.
- Show the viability of convolutional networks for breast cancer diagnosis.
- Use convolutional networks on an entire mammogram instead of only on small patches.
- Analyze the performance of convolutional networks reported on the literature.
- Use the improved convolutional network in the IRMA database.
- Generate results that could produce a conference or journal article.
- Propose new ideas and methods for future research in the topic.

Initial exploratory research has not yet been performed and some of these particular objectives may be modified as the project progresses. Furthermore, some new research avenues could be taken if they seem promising, for instance, using convolutional networks with digital tomosynthesis images (3-dimensional x-ray images of the breast).

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 and diagnosing breast cancer. Convolutional networks are widely used for object recognition tasks and have shown very good results [Russakovsky et al., 2014, Taigman et al., 2014, Dieleman et al., 2015]. They have a big research community and have become one of the preferred methods to perform image classification tasks.

Due to the exploratory nature of this work we are not truly certain of the results that will be obtained. Nevertheless, we have a well established idea of what we expect to obtain. Our hypothesis is that applying convolutional networks to mammographic images will produce similar or better results than those obtained using more traditional computer vision techniques. Additionally, we do not expect that a simple convolutional network will suffice to obtain competitive results; we will need a more refined convolutional network with well fitted parameters. Furthermore, we believe that implementing convolutional networks for breast cancer diagnosis will not be very difficult as it has already been done by other groups (see Section 5.5) and there is plenty of software for it.

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 simple 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 diagnosis? Can preprocessing be replaced by more layers on the same convolutional network? Could we use an entire mammogram for diagnosis instead of only small patches or could we automatically join results for small patches to generate results on the entire mammogram?
- 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?
- 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?

5 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 5.1, we explore some basic concepts about classification and evaluation metrics in Section 5.2, in Sections 5.3 and 5.4 we give a short introduction into Artificial Neural Networks and Convolutional Neural Networks, we present an overview of how convolutional networks have been used for breast cancer diagnosis in Section 5.5 and finally we offer some practical advice for deep learning in Section 5.6.

5.1 Breast Cancer

Cancer is an umbrella term to refer to 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 [National Cancer Institute, 2012].

Breast cancer is the cancer that forms in tissues of the breast. The two most common types of breast cancer are *ductal carcinoma* and *lobular carcinoma*; these cancers begin in the breast ducts and lobules, respectively (see Fig. 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 [Howlader et al., 2014].

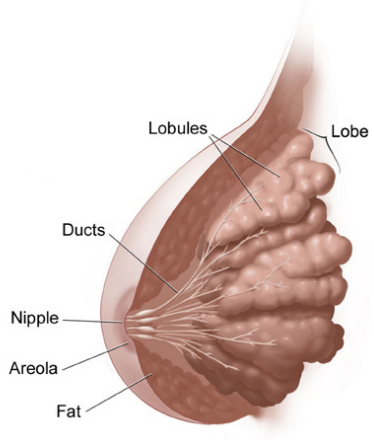

Figure 1: Anatomy of the female breast. Image courtesy of NCI.

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 breast cancer at stage IV 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 that has invaded normal surrounding breast tissue. Stage IV is used to describe metastatic cancer, i.e., breast cancer has spread beyond nearby tissue to other organs of the body.

5.1.1 Mammograms

A *mammogram* is an x-ray image of the breast. *Screening mammograms* (normally composed of two mammograms of each breast) are used to check for breast cancer signs on women who have not shown symptoms of the disease. If an abnormality is found, a *diagnostic mammogram* is ordered, these are detailed x-ray pictures of the suspicious region [National Cancer Institute, 2014]. A standard mammogram is shown in Fig. 2.

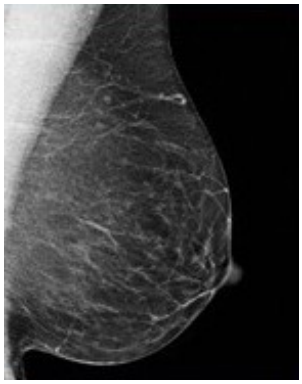

Figure 2: A standard mammogram.

Having a screening mammogram in a regular basis is the most effective method for detecting early breast cancer; around 85% of breast cancers can be detected in a screening mammogram [Breast Cancer Surveillance Consortium, 2013]. 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 [National Cancer Institute, 2014].

Mammograms are read by expert radiologists. The radiologist looks primarily for microcalcifications and breast masses. *Microcalcifications* are tiny deposits of calcium in the breast tissue which can be a sign of early breast cancer if found in clusters with irregular layout and shapes. *Breast masses* or breast lumps are possibly 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 poorly defined shape and margins. See Fig. 3 for an example of possible 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 and it also difficults the interpretation of the mammogram [American Cancer Society, 2014].

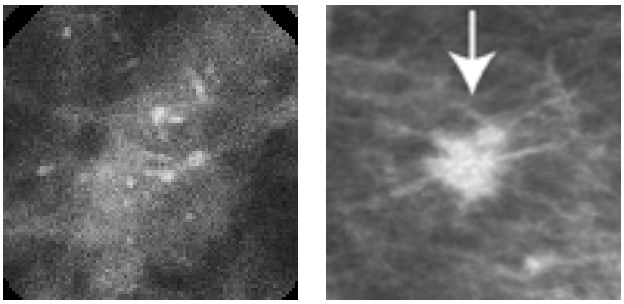

Figure 3: Signs of possible breast cancer in a mammogram. Left: A cluster of microcalcifications in an irregular layout. Right: A poorly defined breast mass.

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 electric 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. Its effectiveness to identify breast cancer over film mammograms, however, is still debated [Kerlikowske et al., 2011, Pisano et al., 2008, Skaane et al., 2007]. Digital mammography is steadily becoming the standard for breast cancer screening, Fig. 2 is, in fact, a digital mammogram.

Digital tomosynthesis, also called three-dimensional mammography, is a new technology that essentially 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, though [National Cancer Institute, 2014].

In this thesis we will center on using mammograms, either digital or manually digitized from film, to detect microcalcifications and masses and predict the likelihood of breast cancer on the patient.

Much of this section was written using information from the National Cancer Institute. We recommend to visit its website (`www.cancer.gov`) for more information.

5.2 Classification

Machine learning is the study of algorithms which build models of a population or a 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 can be divided into three categories depending on the data used to train the model: *supervised learning*, where we learn a function $f(x)$ using a set of examples which are labelled with the correct output, for instance, learning a function that estimates the price of a house given its size and number of bedrooms from a dataset of houses labelled with their real value; *unsupervised learning*, where we look for relationships and structure in unlabelled data, for instance, given a dataset of potential customers find those who are likely to buy a car and *reinforcement learning*, where the only feedback received are rewards, for example, learning to play tetris from a dataset of world states and actions and where rewards are received sparsely every time points are earned (when lines dissapear). Supervised learning can be further divided 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 will focus on classification.

A *classifier* takes as input a vector of *features* $x \in \mathbb{R}^n$ from a problem instance and produces an *output* $h(x)$ predicting the class y that instance belongs to, i.e., it concretely 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 5.4, output a *score vector* $h(x) \in \mathbb{R}^K$ where $h(x)_k$ is a measure of the probability that x belongs to class k. Every classifier partitions the *feature space*, the n-dimensional space where features exist, into separate *decision regions*, regions of the space which are assigned the same predicted 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* $J(\theta)$ of a classifier measures the amount of error the classifier incurs in for a particular choice of parameters θ. There are various ways to formulate this function. A *least-squares loss function* for a binary classifier (such as logistic regression) is presented in Equation 1

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

where m is the number of training examples, $y \in \{0, 1\}$ is the real class of the 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 the parameters that minimize $J(\theta)$: 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:

$$\theta = \theta - \alpha \nabla J(\theta) \tag{2}$$

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 each model is trained on a subset of the data set and later evaluated 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 and evaluated on the validation set and the model which shows the best performance 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 which modify the underlying model or learning algorithm, are selected in the same way.

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 compared to the size of the training set, the classifier will probably *overfit* the data, this means that the parameters are fitted way too closely to the data and will pick up every small fluctuation and noise in the training set causing the trained classifier to produce almost perfect results on the training set but perform poorly on previously unseen examples. The opposite is also true, when $h(x)$ is very simple the classifier lacks the power to model the function of interest and we say that it *underfits* the data. This problem is sometimes referred as the *bias-variance tradeoff*. A high variance classifier is prone to overfitting, while a high bias classifier is prone to underfitting.

A popular way to avoid overfitting (and underfitting) is to use a flexible model trained with regularization. *Regularization* modifies the loss function to include a penalty to the complexity of the model, thus forcing the learning stage to choose parameters that minimize both the training error of the classifier and the complexity of the model. Equation 3 shows the least-squares loss function with l_2*-norm regularization*:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} ||\theta||_2 \tag{3}$$

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, thus 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) which 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 which are correctly classified. Its compliment, *error rate*, measures the proportion of test set examples which are incorrectly classified. Accuracy, nonetheless, is not a good evaluation metric for unbalanced data sets, data sets which have many more examples

of one class than the other e.g., cancer data sets are often unbalanced as most examples belong to the negative class (no cancer) than the positive class (cancer). For instance, a classifier which always predicts no cancer regardless of the input will show a high accuracy (equivalently a low error rate) even though it is not a good model for the problem.

A different set of metrics based on the confusion matrix of the classifier are used to evaluate its quality in unbalanced data sets. A *confusion matrix* is a matrix which summarizes the results of a classifier in the test set (see Table 1). *True positives* is the number of positive

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

Table 1: Confusion matrix for a binary classifier

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

$$Sensitivity \text{ or } Recall = \frac{TP}{TP + FN} \tag{4}$$

$$Specificity = \frac{TN}{FP + TN} \tag{5}$$

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

Sensitivity and specificity are usually 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* is a measure of the proportion of examples predicted as positive which 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 7 and 8.

$$F_1 \ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{7}$$

$$G\text{-}mean = \sqrt{Sensitivity \times Specificity} \tag{8}$$

In this thesis, we will generally present results for all this metrics (precision, recall or sensitivity, specificity, F_1 score and G-mean). The metric used when selecting a model influences its characteristics and behaviour, hence, one should put some consideration into choosing it. We favor F_1 score over G-mean because it concentrates on prediction in the positive class (cancer) which is harder to predict and the class we are more interested in. Furthermore, it represents a more balanced tradeoff (an small change in precision is corresponded with a small change in recall) than G-mean where an small change in specificity can be corresponded with a big change in sensitivity.

This section is meant to be a compendium of basic concepts in machine learning, practical machine learning involves many subtleties and implementation details not mentioned here. Notation and content in this section is mostly based on materials from Stanford's Machine Learning course[Ng, 2014].

5.3 Artificial Neural Networks

Artificial neural networks or simply *neural networks* are one of the most popular nonlinear classifiers used today. They were initially inspired by the way biological neurons process information coming from its dendrites and relaying it through its axon to neighboring neurons [McCulloch and Pitts, 1943, Widrow and Hoff, 1960, Rosenblatt, 1962] but evolved to become more practical for nonlinear modelling albeit less biologically accurate [Rumelhart et al., 1986]. We discuss here multilayer feedforward neural networks, the name should become obvious after a few paragraphs.

Multilayer feedforward neural networks are composed of L layers of *neurons*, units of computation, each of which 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 the feature vector $x \in \mathbb{R}^n$ meanwhile 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. 4 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 which 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*.[2]

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

$$a_i^{(l)} = g\left(\sum_{j=0}^{s^{(l-1)}} \Theta_{ij}^{(l-1)} a_j^{(l-1)}\right) \text{ for } l = 2, \ldots, L-1 \text{ and } i = 1, \ldots, s^{(l)} \tag{9}$$

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 $u = 1, \ldots, L-1$ (bias units); $a_v^{(1)} = x_v$ for all $v = 1, \ldots, n$ 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 $i = 1, \ldots, s^{(L)}$ i.e, $g(\cdot)$ is not applied 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 which always emits activation 1 ($a_0^{(1)} = 1$, $a_0^{(2)} = 1$, etc), these are called *bias units* [3]. The bias units are assumed to be included into each vector $a^{(l)}$, hence the sumation in Equation 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

[2] There is no consensus on when a neural network becomes a deep neural network[Schmidhuber, 2015]. We consider networks with over 2 hidden layers to be deep.

[3] 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.

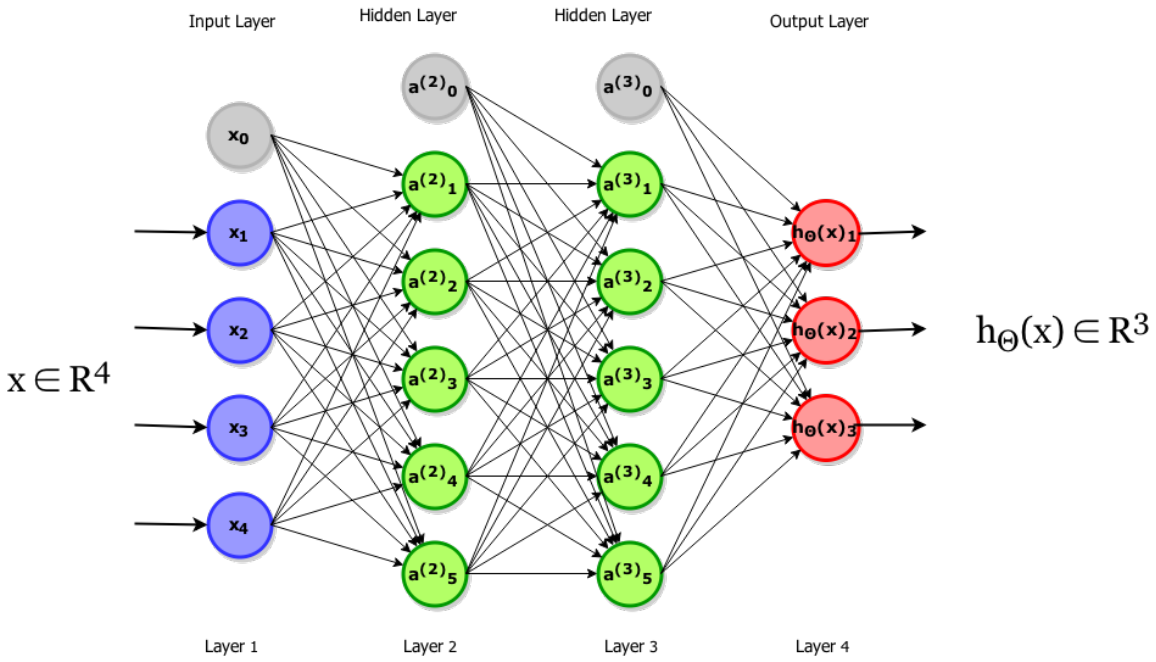

Figure 4: Small neural network example. Input layer with 4 units (blue), two hidden layers of 5 units (green) and output layer of 3 units(red). Bias units appear in gray. It approximates a function $h_\Theta(x) : \mathbb{R}^4 \to \mathbb{R}^3$, i.e., it classifies an input vector $x \in \mathbb{R}^4$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

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 the possible classes K ($p(x) \in [0..1]^K$). This improves interpretability but does not change the predicted classes.

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

$$g(z) = \max(0, z) \tag{10}$$

This is a nonlinear activation function which is very easy to implement in numerical computations, its derivative ($1_{z>0}$) can be calculated quickly and does not suffer from vanishing or exploding gradients as do the sigmoid or tanh activation functions. Furthermore, it has been shown to greatly accelerate the convergence of gradient descent [Krizhevsky et al., 2012]. For these reasons it is currently the recommended activation function for deep neural networks [Karpathy, 2015].

Each unit in a neural network outputs a nonlinear activation $g(z)$ which 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 which 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 (as long as there is enough training data).

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

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\log\left(\frac{e^{h_\Theta(x^{(i)})_{y^{(i)}}}}{\sum_{j=1}^{K} e^{h_\Theta(x^{(i)})_j}}\right) \tag{11}$$

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 [Ng, 2014]. *Error backpropagation* [Linnainmaa, 1970, Werbos, 1974], 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. Given the big number of parameters which need to be estimated, neural networks are susceptible to overfitting. The simplest approach to overcome this problenm is to use regularization. Regularization for neural networks is done by performing gradient descent on the regularized loss function presented in Equation 12

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\log\left(\frac{e^{h_\Theta(x^{(i)})_{y^{(i)}}}}{\sum_{j=1}^{K} e^{h_\Theta(x^{(i)})_j}}\right) + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s^{(l)}}\sum_{j=1}^{s^{(l+1)}}\left(\Theta_{ij}^{(l)}\right)^2 \tag{12}$$

5.4 Convolutional Networks

Convolutional networks, also *ConvNets* or *CNNs*, were first inspired on the way the human visual cortex proccesses information [Fukushima, 1980] but, as regular neural networks, they 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 [LeCun et al., 1989, LeCun et al., 1998], the first successful application of modern convolutional networks. Recently, they have been used to achieve state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge [Krizhevsky et al., 2012], an image classification and object localization challenge with 1000 categories [Russakovsky et al., 2014]. Since then, thanks to various advances (maxpooling , ReLU activations, weight initialization, GPU training, efficient backpropagation, etc.) they have become one of the most popular methods for image classification tasks and (along with recursive neural networks for generative models) an emblem for deep learning.

In this section we show the standard features and training of current convolutional networks, Section 5.6 gives some practical advice for choosing hyperparameters and training deep architectures. For an in depth review of convolutional networks, see [Karpathy, 2015]. For a complete overview of the history and state of deep learning, see [Schmidhuber, 2015].

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 easily use a regular neural network (presented in Section 5.3) to do this classification 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 time to train but because the loss function has very many local minima and thus it is harder to find a good local minimum.

Convolutional networks are specially designed to handle images reducing the number of connections between layers and the number of parameters to learn. Instead of fully connected layers such as regular neural networks 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 layers, i.e., different connections 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 this features reduce computation and improve the classification performance of convolutional networks; they are a product of 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 ($\mathbb{R}^{h\times w\times d}$) where the third dimension is used to put together all feature maps. One could think of each layer as having all unit activations in a single column as in regular neural networks but seeing them as a 3-dimensional volume makes the definitions easier. The input layer could be considered as a 3-dimensional matrix ($\mathbb{R}^{h\times w\times c}$) holding the image of size $w \times h$ with c color channels (usually one for grayscale images or three for RGB). The output layer could also be thought of as 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 convolutional network then receives an input image x, transforms it into the first layer of feature maps (which does not need to have the same dimensions as the previous layer) and keeps transforming it until we have an output layer of size $h(x) = R^{1\times 1\times K}$. See Fig. 5 for an illustration. We describe the possible transformations next.

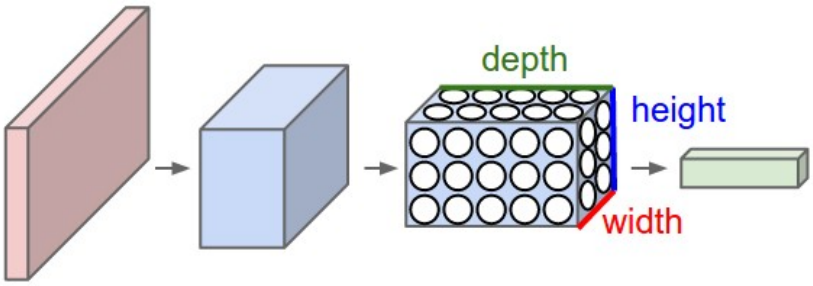

Figure 5: A simple representation of the transformations of the input that a convolutional network computes. Input layer is shown in pink, hidden layers are shown in blue and output layer is shown in green. The third layer has 5 feature maps of size 2×3. Notice that the width is listed first by convention. Image courtesy of [Karpathy, 2015].

There are four types of layers: 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 composed of a set of learnable filters which will be applied to the volume in the previous

layer. A *filter* is a matrix of weights which 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 6 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 producing a 2-dimensional array of values [4]. Notice that 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 of the input and each feature map collecting the probabilities of the feature being present in different positions of the original image.

We need to define 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 defined in a way that it preserves the spatial size of the previous volume, the third dimension is solely dependent on the number of feature maps desired.

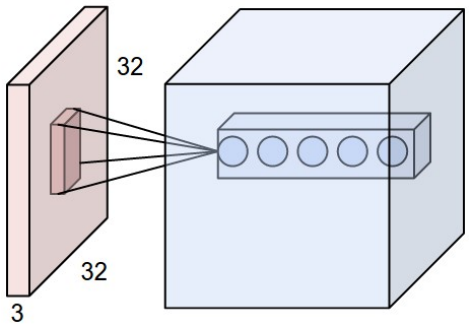

Figure 6: Example of a filter applied to a volume ($\mathbb{R}^{32\times32\times3}$) to obtain the values shown in the blue volume. The filter comprises all 3 feature maps of the input volume. We compute 5 feature maps as shown by the 5 units in the blue volume. For a complete convolution this filter will have to slide across the input volume. Notice 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 [Karpathy, 2015].

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 does not change the dimensions of the volume and has no learnable parameters, although the activation function itself could be considered as a hyperparameter. Usually a convolutional layer is always followed by a ReLU layer (or any other activation function), for this reason they are sometimes considered part of the convolutional layer, we leave them separate for clarity.

[4] Each filter has also a bias term which is added to the product.

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 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. Notice that the subsampling is applied to each feature map separately contrary to the convolution. A popular variant of max pooling uses 3×3 windows with stride 2, allowing for overlapping in the pooling.

Fully connected layer One or more fully connected layers are used 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 network with as many feature maps as classes. The interpretation of the scores of the output layer is similar to that 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 dimensions of the feature maps in the previous layer, i.e, filters that comprise the entire previous volume.

Convolutional layers (plus ReLUs) compute features on the input while pooling layers shrinken the volume before passing to the fully connected layers which act as a 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 which reports the final classification scores. Although there is no standard way of counting the number of layers of a convolutional network usually the ReLU or pooling layers are not counted as they have no learnable parameters, therefore our example architecture has 10 layers (21 in total) which is a good depth for big data sets. Practical recommendations on building convolutional network architectures is offered in the Section 5.6.

Figure 7 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`.

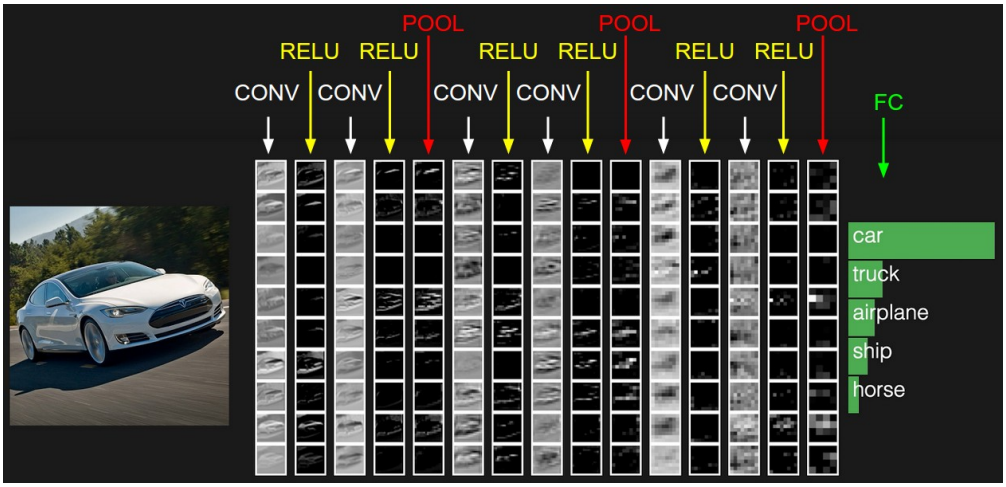

Figure 7: Example of a convolutional network with architecture `INPUT -> [[CONV -> RELU]*2 -> POOL]*3 -> FC`. The input image has size 32×32. Each hidden layer uses 10 feature maps (shown as columns). Notice that although the size of the feature maps looks constant in fact each pooling layer reduces each dimension by half (the feature maps of the final pooling layer have size 4×4). The final scores are shown only for the 5 most probable classes. Image courtesy of [Karpathy, 2015].

Recently there has been a push towards simpler convolutional network architectures. The All Convolutional Net [Springenberg et al., 2014] is a network formed solely by convolutional layers: pooling layers are replaced by convolutional layers with larger strides and fully connected layers are replaced as explained above. Notice that this greatly increases the number of parameters to be learn, therefore it may not be suitable 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 a bigger image is used 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). We can then 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 which 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 were applying 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 a convolutional network is trained on images from a specific domain and later used as a feature extractor for images on a different domain or 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 12) except that the vector score $h_\Theta(x)$ is now defined by the architecture of the convolutional network.

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\log\left(\frac{e^{h_\Theta(x^{(i)})_{y^{(i)}}}}{\sum_{j=1}^{K} e^{h_\Theta(x^{(i)})_j}}\right) + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s^{(l)}}\sum_{j=1}^{s^{(l+1)}}\left(\Theta_{ij}^{(l)}\right)^2 \tag{13}$$

Furthermore, this loss function is still differentiable with respect to Θ and thus the entire network can be trained via gradient descent. Gradients of the loss function are also calculated using backpropagation.

5.5 Convolutional Networks applied to Breast Cancer

Yet to write. Redacted version of section Related Work.

5.5.1 Related Work

In this section we offer a summary of some of the first work on using convolutional networks for breast cancer diagnosis as well as some of the articles that have influenced this thesis.

[Lo et al., 1995] were the first group to use convolutional networks for breast cancer detection. They used a CNN with two hidden layers to detect microcalcifications. A high sensitivity image processing technique was used to obtain a set of 2104 patches (16 by 16 pixels) of all potential disease areas from 68 digital mammograms; of these, 265 were true microcalcifications and 1821 were "false subtle microcalcifications". Prior to training the CNN, a wavelet high-pass filtering technique was used to remove the background of these images. Each image was flipped over (left-right) and 4 rotations for each the original and flipped images were used for training (0°, 90°, 180° and 270°). The CNN was composed of one input unit (16×16), 12 units in the first hidden layer (12×12), 12 units in the second hidden layer(8×8) and two output nodes(one for YES and one for NOT). The input size (16), number of hidden layers (2) and kernel size (5×5) was obtained via cross validation, altough not many other 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. The CNN reached 0.87 average AUC when identifying individual microcalcifications and 0.97 AUC for clustered microcalcifications. Only a minimum of three calcifications was considered a detection. Sensitivity and specificity test results were not reported. This article proved that simple convolutional networks can be efficiently used for medical image pattern recognition.

5.6 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 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 [Karpathy, 2015].
- 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 later be used when normalizing the test data [Karpathy, 2015].

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 [Ng, 2014, Krizhevsky et al., 2012].
- 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 [Bengio, 2014].
- 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." [Karpathy, 2015]
- 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) [Karpathy, 2015]. 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) [Karpathy, 2015]. 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 [Simonyan and Zisserman, 2014].

- 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 [Springenberg et al., 2014]. When training on big images, the first convolutional layer uses bigger filters [Karpathy, 2015].

- Use 2×2 pooling with stride 2. Both this pooling and the overlapping version presented in Section 5.4 produce similar results. This pooling divides the spatial dimensions of the volume by half.

- 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 [Bengio, 2014]. For small data sets, cross validation can give better estimates and is preferred [Ng, 2014].

- Use random search rather than grid search. Random search draws each parameter from a value distribution rather than a set of predefined values. [Bergstra and Bengio, 2012]

- 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 [Karpathy, 2015].

- 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, dropout probability 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 [Cawley and Talbot, 2010].

- Set α, λ and optionally the type of preprocessing using a validation set. Other hyperparameters are 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" [Bengio, 2012]. It ranges from 10^{-6} to 10. Use a log scale to draw new values ($\alpha = 10^{unif(-6,1)}$ where $unif(a, b)$ is the continous uniform distribution) [Karpathy, 2015].

- 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. [Bengio, 2012].

- 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 [Krizhevsky et al., 2012].

- Use $\mu = 0.9$. When using a validation set try values in {0.5, 0.9, 0.95, 0.99} [Karpathy, 2015].

- Use dropout probability $p = 0.5$ [Karpathy, 2015]. Lower values mean less dropout.

- Use mini-batch size of 32 or 64. A larger batch size requires more training time. It affects training time more than test performance [Bengio, 2012].

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 [Bengio, 2012].

- 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 this 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 [He et al., 2015].

- 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 result in faster convergence and better test results [Bengio, 2012].

- Use Nesterov's Accelerated Gradient (NAG) to update the weights. It is a modified version of Stochastic Gradient Descent with Momentum (SGD+Momentum) which has shown to work slightly better for recurrent networks. SGD+Momentum is also a viable option [Karpathy, 2015].
- 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 [Bengio, 2014].
- 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 [Bengio, 2012].
- Use the validation or test error to select the best parameters for the network from those stored [Bengio, 2014].
- 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 [Ng, 2014].

Sanity checks Some simple checks we can run to make sure everything is alright.

- Make sure that with random initializations it gives yuloss at chance performance. Loss should go up-increase with more regularization. No regulariaiton at first.
- If you implemented back propagation manually or have reasons to believe that it is not working properly, doing a numerical gradient check is a very easy way to confirm your intuitions [Karpathy, 2015].
- Overfit a tiny subset of data(cost=0). If i can't is not worthede going to a bigger dataset. If it cannot, then the modelis too simple.
- Training Loss should always decrease or only slightly increase. Wehn it foundas a plateau, reducing the learning rate as explained above could make it decrease more.

6 Methodology

In order to achieve the proposed objectives and test our hypotheses we will need to carry out various tasks. We list them here in the order in which we plan to execute them:

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 should be obtained and written. This will also help refine the scope of the project and the experiments to be conducted.

2. **Software review**

   Once a clear idea of what are the possible experiments to be executed, we will need to find appropiate software to perform them. Software for database managing, preprocessing and implementation of different neural networks should be either located or developed.

3. **Database preprocessing**

   We will ready the database images for the experiments; these implies joining different databases, obtaining the required features, preprocessing the images, assigning labels, etc.

4. **Assesing image preprocessing**

   We will train a standard convolutional network with fixed parameters on mammograms with three different preprocessings: no preprocessing, image enhancement using median or gaussian filters and wavelet filtered images. Furthermore, we will train a deeper convolutional network on nonpreprocessed images. We want to answer three research questions: which is the best preprocessing for convolutional networks, is using the best filter significantly better than using nonpreprocessed images and can a convolutional network automatically preprocess the images?

5. **Exploratory experiments**

   We will train standard convolutional networks in two different inputs: small image patches obtained from mammograms and whole mammogram images. We will also train a linear classifier, probably rectified linear units, on the features obtained from a convolutional network trained on the ImageNet database, i.e., we will use an already trained convolutional network instead of one trained specifically in mammograms. Here we will use the image preprocessing technique that showed better results in the previous step. We want to answer two research questions: Can a convolutional network trained on whole mammograms perform as well as one trained on small patches and can we use an already trained convolutional network to classify mammograms?

6. **Model selection**

   Using the insights from previous sections and the current literature on convolutional networks, we will select a network architecture along with novel features, preprocessing, training and regularization procedures. We aspire to find the best convolutional network configuration for mammogram classification.

7. **Further experiments**

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

8. **Gathering results**

   Produce results on the test set and elaborate figures and tables. This could be obtained directly from software output or from further program executions.

9. **Reporting results**

   Write the thesis and any article or technical guide which may result from this work. Both this and the previous step will be performed along the execution of the project, hopefully benefiting from the supervisors' feedback.

Finally, we would like to note that this is an idealized workflow and some changes may occur due to time limitations or lack of resources. In the unlikely case that the work is finished before the project deadline, we will either reiterate on model selection, experiments and results gathering and reporting or look into digital tomosynthesis, network ensembles or evolving convolutional networks.

7 Work Plan

We present here the expected work plan for this master's thesis. A description of the activities can be found in Section 6

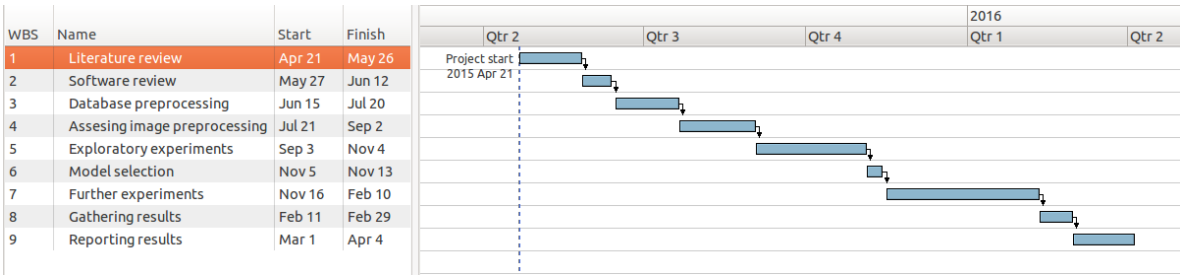

Figure 8: Thesis work plan.

Literature and software review and the preprocessing of the image database are expected to be done before the end of the summer term (late July). The first experiments about preprocessing and using convolutional networks on entire mammograms are expected for the fall semester (late October to early November). At this point, these results should be reported in the thesis. Later, the final architecture and methods will be selected and the final experiments run. The thesis document should be delivered by the start of March. During this month, if the results are valuable, we expect to write a conference or journal article to share our results with the community.

References

[American Cancer Society, 2014] American Cancer Society (2014). Mammograms and other breast imaging tests. electronic, Atlanta, GA. Available online on `cancer.org/acs/groups/cid/documents/webcontent/003178-pdf.pdf`.

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

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

[Bengio, 2014] Bengio, Y. (2014). Deep learning workshop. online. Available on youtu.be/JuimBuvEWBg.

[Bergstra and Bengio, 2012] Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *J. Machine Learning Research*, 13(1):281–305.

[Breast Cancer Surveillance Consortium, 2013] Breast Cancer Surveillance Consortium (2013). Performance measures for 1,838,372 screening mammography examinations from 2004 to 2008 by age–based on BCSC data through 2009. electronic. Available on breastscreening.cancer.gov/statistics/performance/screening/2009/perf_age.html.

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

[Dieleman et al., 2015] Dieleman, S., Willett, K. W., and Dambre, J. (2015). Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*.

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

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852.

[Hernandez, 2014] Hernandez, J. L. (2014). Selección de características y clasificacíon de masas por medio de redes neuronales, máquinas de vector de soporte, análisis discriminante y regresíon logística en mamografías digitales. Master's thesis, Tecnológico de Monterrey, Monterrey, Mexico.

[Howlader et al., 2014] 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. (2014). SEER cancer statistics review, 1975-2011. review, National Cancer Institute, Bethesda, MD. Available on seer.cancer.gov/csr/1975_2011/.

[Karpathy, 2015] Karpathy, A. (2015). Convolutional neural networks for visual recognition course. online. Available on cs231n.github.io.

[Kerlikowske et al., 2011] Kerlikowske, K., Hubbard, R. A., Miglioretti, D. L., Geller, B. M., Yankaskas, B. C., Lehman, C. D., Taplin, S. H., and Sickles, E. A. (2011). Comparative effectiveness of digital versus film-screen mammography in community practice in the united states: A cohort study. *Annals of Internal Medicine*, 155(8):493–502.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Linnainmaa, 1970] Linnainmaa, S. (1970). 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.

[Lo et al., 1995] Lo, S.-C. B., Chan, H.-P., Lin, J.-S., Li, H., Freedman, M. T., and Mun, S. K. (1995). Artificial convolution neural network for medical image pattern recognition. *Neural Networks*, 8(7–8):1201 – 1214. Automatic Target Recognition.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

[National Cancer Institute, 2012] National Cancer Institute (2012). What you need to know about breast cancer. electronic, Bethesda, MD. Available online on `cancer.gov/publications/patient-education/WYNTK_breast.pdf`.

[National Cancer Institute, 2014] National Cancer Institute (2014). Mammograms. electronic. Available on `cancer.gov/cancertopics/types/breast/mammograms-fact-sheet`.

[Ng, 2014] Ng, A. (2014). Machine learning course. online. Available on `coursera.org/course/ml`.

[Pisano et al., 2008] Pisano, E. D., Hendrick, R. E., 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. A., and Gatsonis, C. A. (2008). Diagnostic accuracy of digital versus film mammography: Exploratory analysis of selected population subgroups in DMIST. *Radiology*, 246(2):376–383. PMID: 18227537.

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

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., McClelland, J. L., and PDP Research Group, C., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 318–362. MIT Press, Cambridge, MA.

[Russakovsky et al., 2014] 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. (2014). ImageNet Large Scale Visual Recognition Challenge. *CoRR*, abs/1409.0575. Available online on `arxiv.org/abs/1409.0575`.

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

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

[Skaane et al., 2007] Skaane, P., Hofvind, S., and Skjennald, A. (2007). 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):708–717. PMID: 17709826.

[Springenberg et al., 2014] Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. A. (2014). Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806.

[Taigman et al., 2014] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). DeepFace: Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR)*, Columbus, Ohio.

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

[Widrow and Hoff, 1960] Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. In *1960 IRE WESCON Convention Record, Part 4*, pages 96–104, New York. IRE.