

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

School of Engineering and Sciences



Segmentation of Breast Cancer Masses 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, December, 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

Dr. Santiago Conant Pablos
Tecnológico de Monterrey
Committee Member

José Tamez Peña
Tecnológico de Monterrey
Committee Member

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

Monterrey, Nuevo León, December, 2016

Declaration of Authorship

I, Erick Michael Cobos Tandazo, declare that this thesis titled, "Segmentation of Breast Cancer Masses 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, December, 2016

©2016 by Erick Michael Cobos Tandazo
All Rights Reserved

Dedication

I dedicate this work to my parents, Eladio and Aracelly. Thank you for all your confidence, support, patience and encouragement.

Acknowledgements

I would like to express my gratitude to all those who have supported me throughout the often long hours of this journey. I would like to thank my advisor, Dr. Hugo Terashima, for being my compass even when I felt I was lost and being in great part responsible for the culmination of this work. I would also like to thank my supervisors for their helpful advice, which greatly improved the quality of this work. Finally, I thank this institution for hosting me during this years, securely earning its place as my home and CONACyT for its financial support.

I am very thankful to my mom and dad, whose values and education motivate me to keep asking questions; to my siblings and family for their infinitely appreciated love and to my town which, although inanimate, keeps me anchored and offers me an example of resiliency.

**Segmentation of Breast Cancer Masses in Digital
Mammograms: A Convolutional Network**

by

Erick Michael Cobos Tandazo

Abstract

Yet to write

List of Figures

2.1	Sample ROC and PR curves	10
2.2	An artificial neural network	12
2.3	Example of Dropout	13
2.4	Illustration of a convolutional network	14
2.5	Convolutional layer applied to a volume	15
2.6	Convolutional network in action	17
2.7	Segmentation of an image	17
2.8	Example of dilated convolutions	18
2.9	Anatomy of the female breast	25
2.10	A digital mammogram	26
2.11	Signs of possible breast cancer	26
3.1	Overview of the solution	35
3.2	Preprocessing pipeline	38
3.3	Post-processing pipeline	43
4.1	FROC curves	46
A.1	Example 1	50
A.2	Predictions for example 1	50
A.3	Example 2	51
A.4	Predictions for example 2	51
A.5	Example 3	52
A.6	Predictions for example 3	52
A.7	Example 4	53
A.8	Predictions for example 4	53
A.9	Example 5	54
A.10	Predictions for example 5	54

List of Tables

2.1	Confusion matrix for a binary classifier	9
2.2	Breast cancer convolutional network architectures	32
3.1	Data set summary	36
3.2	Convolutional network architecture for Experiment 1	38
3.3	Selected convolutional network architecture	40
3.4	Convolutional network architecture for Experiment 3	41
3.5	Available hardware for experiments	42
4.1	Hyperparameter search results	45
4.2	Sensitivity at 1 FP/image for the final models	46

Contents

Abstract	v
List of Figures	vi
List of Tables	vii
1 Introduction	1
1.1 Motivation	2
1.2 Problem Definition	2
1.3 Objectives	3
1.4 Hypothesis	4
1.4.1 Research Questions	4
1.5 Methodology	5
1.6 Contributions	5
1.7 Outline of the thesis	6
1.8 Summary	6
2 Background	7
2.1 Classification	7
2.2 Artificial Neural Networks	11
2.3 Convolutional Networks	13
2.4 Image Segmentation	17
2.5 Practical Deep Learning	19
2.6 Breast Cancer	25
2.6.1 Mammograms	25
2.6.2 Mammographic databases	27
2.7 Convolutional Networks in Breast Cancer Research	27
2.7.1 Detection of breast masses	28
2.7.2 Detection of microcalcifications	29
2.7.3 Diagnosis of breast masses	30
2.7.4 Tissue segmentation	30
2.8 Summary	34
3 Solution	35
3.1 Task definition	35
3.2 Data set	36

3.2.1	Database	36
3.2.2	Data division	36
3.2.3	Image enhancement (Exp. 1.3 and Exp. 3)	36
3.2.4	Resizing	37
3.2.5	Cropping	37
3.2.6	Data augmentation	37
3.2.7	Storage	37
3.3	Experiment 1	37
3.3.1	Architecture	38
3.3.2	Regularization	39
3.3.3	Loss function	39
3.3.4	Experiments	39
3.4	Experiment 2	39
3.4.1	Architecture	40
3.4.2	Regularization	40
3.4.3	Loss function	40
3.4.4	Experiments	40
3.5	Experiment 3	41
3.5.1	Architecture	41
3.5.2	Regularization	41
3.5.3	Loss function	41
3.5.4	Experiments	41
3.6	Training	42
3.6.1	Hardware	42
3.6.2	Software	42
3.6.3	Initialization	42
3.6.4	Hyperparameter search	42
3.6.5	Optimization	42
3.7	Evaluation	43
3.7.1	Post-processing	43
3.7.2	Segmentation	43
3.7.3	Metrics	43
3.8	Summary	44
4	Results	45
4.1	Hyperparameter Search	45
4.2	Experiments	45
4.3	Discussion	47
5	Conclusions	48
5.1	Future Work	48

A Examples	49
A.1 Example 1	50
A.2 Example 2	51
A.3 Example 3	52
A.4 Example 4	53
A.5 Example 5	54
Bibliography	61

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 [47]. 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 process images 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 [20, 34], a natural extension to feedforward neural networks, are statistical learning models that use raw images as input and learn the relevant image features during training. They work well with minimally preprocessed images and

encapsulate segmentation, feature extraction and classification in a single trainable model. Despite some drawbacks, convolutional networks are the state-of-the-art technology for object recognition [54].

Researchers have used small convolutional networks to separate breast masses from normal tissue [55] and individual microcalcifications from noise in the image [37, 21]. A bigger network incorporating newer features such as rectified linear unit activations, pooling, momentum, data augmentation and dropout was trained to identify malignant masses [4]. For these experiments mammograms were enhanced, potential lesions were located and relevant regions were presented to the network for classification. Recently, Dubrovina et al. [19], segmented different breast tissue using a modern convolutional network. This work relates closely to our own. We train deep convolutional networks end-to-end to identify lesions in digital mammograms.

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% [27].

With current technology, a high quality mammogram is “the most efective way to detect breast cancer early” [44]. 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 addressed to further advance the field:

- Lack of standard preprocessing techniques. Commonly used techniques vary in performance.
- Handcrafted features. Image features are chosen beforehand, designed with help of experts and extracted using complex, problem-dependent techniques.
- Expertise needs. Traditional systems require knowledge in many fields such as radiology, oncology, image processing, computer vision and machine learning.
- Pipeline structure. Traditional systems are composed of sequential steps. At each stage, researchers choose among many techniques and adjust their parameters; as achieving an optimal combination of techniques is unlikely, this is both inconvenient and inefficient.
- Low ceiling. Algorithms are already complex and require much work to achieve incremental improvements.
- Complexity. Issues such as non-desired or unknown dependencies between subsystems, difficulty to localize errors and maintainability arise.

In this thesis, we use convolutional networks, a recent development in machine learning, (Sec. 2.3) 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 focus on improving the learning mechanism, both the model and algorithm, rather than on designing novel image features or improving specific subsystems.

1.3 Objectives

The main goal of this work is to use convolutional networks to segment breast cancer lesions in digital mammograms and measure how different architectural decisions affect performance. Particularly, there are various minor goals 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.
- Train a modern, fine-tuned convolutional network to perform lesion segmentation.
- Test the viability of convolutional networks for breast cancer detection and diagnosis.

^aSee [5] for an example.

- Use alternative convolutional network models to improve 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 [54, 62, 18]. 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 convolutional network apt for image segmentation 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 (Sec. 2.7) and plenty of software is available.

1.4.1 Research Questions

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

- Are convolutional networks sufficiently powerful to perform breast cancer lesion segmentation as an end-to-end task? Is scarce data an obstacle for learning?
- 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?
- What are the advantages of using a deep versus a shallow convolutional network?
- 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 developed tools to store, label and preprocess the images.

3. Software review

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

4. Model selection

Using insights from the current literature on convolutional networks and medical image analysis, we selected image preprocessing techniques, network architectures, training and regularization procedures, evaluation metrics and post-processing techniques for our experiments.

5. Experiments

We trained the chosen convolutional networks 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. Gathering results

We evaluated our final models on the test set and elaborated figures and tables to present the results.

7. Reporting results

We revised and wrote the final draft of this thesis.

1.6 Contributions

This thesis is the first documented use of convolutional networks for breast cancer lesion segmentation. It shows the viability of using modern machine learning techniques to simplify medical image processing pipelines and hints towards the viability of training more flexible

models as larger data sets become available. Software, from preprocessing to evaluation tools, as well as some trained models are available online and have already been used by other groups working on similar tasks. Furthermore, a copy of the mammographic database and preprocessing tools remain available for use inside the institution. Lastly, this work supported the development of several deep learning projects by contributing theoretical and technical knowledge.

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.

1.8 Summary

We are tasked with performing automatic segmentation of breast cancer masses in digital mammograms. Computerized diagnosis assist radiologists in the search for signs of cancer development. Due to its high incidence, improving breast cancer diagnosis will benefit million of women around the world. We aim to implement a modern machine learning model, convolutional neural networks, to obtain results comparable to those of conventional methods.

Chapter 2

Background

We offer an introduction to some of the essential concepts needed to understand this document. We explore classification in Section 2.1, introduce artificial neural networks and convolutional networks in Sections 2.2 and 2.3, address image segmentation in Section 2.4, offer advice to deep learning practitioners in Section 2.5, discuss breast cancer in Section 2.6 and review the use of convolutional networks in breast cancer research in Section 2.7.

2.1 Classification

Machine learning is the study of algorithms that build models of a population or function of interest estimating 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 and their true valuations; *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 and *reinforcement learning*, where feedback is received intermittently, for instance, learning to play Tetris from a data set of world states, actions and rewards received only when points are earned. 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 output 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 (Sec. 2.3), 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 one or more hidden layers 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:

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

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* estimates these parameters by initializing them at random and iteratively updating them using the gradient of the loss function. Specifically, at each iteration it performs the update:

$$\theta = \theta - \alpha \nabla L(\theta) \quad (2.2)$$

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, which 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 candidate on a subset of the data and evaluate it on a disjoint subset to estimate their performance. In the validation set approach the data set is split into a training set (usually 60-90%) 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 adjust the underlying model or learning algorithm, can be selected similarly.

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 *overfit* the data; this means that parameters are fitted too tightly 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*:

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

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 training error and regularization error. *l_1 -norm regularization* or *lasso* is defined similarly except that it shrinks the l_1 -norm of θ rather than the l_2 -norm.

We evaluate the performance of a classifier on a separate set of examples, a test set, that should have not been used for training or validation. The standard performance measure in machine learning is classification accuracy; *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 inappropriate 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 is highly accurate (it is 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. A *confusion matrix* summarizes the results of a classifier in the test set (Tab. 2.1). *True*

		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

positives is the number of positive examples correctly 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:

$$\text{Sensitivity or Recall} = \frac{TP}{TP + FN} \quad (2.4)$$

$$\text{Specificity} = \frac{TN}{FP + TN} \quad (2.5)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.6)$$

^aMedical data sets are often unbalanced as most examples belong to the negative class (no disease) than the positive class (disease)

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. Sensitivity and specificity are preferred in medical diagnosis while precision and recall are preferred in machine learning.

It is often 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.

$$F_1 \text{ score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.7)$$

$$G\text{-mean} = \sqrt{\text{Sensitivity} \times \text{Specificity}} \quad (2.8)$$

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 (Fig. 2.1). The *receiver operating characteristic curve* plots sensitivity (also called true positive rate) against 1-specificity (also called false positive rate) as the threshold varies (Fig. 2.1). 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 can also be 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 used mostly in machine learning.

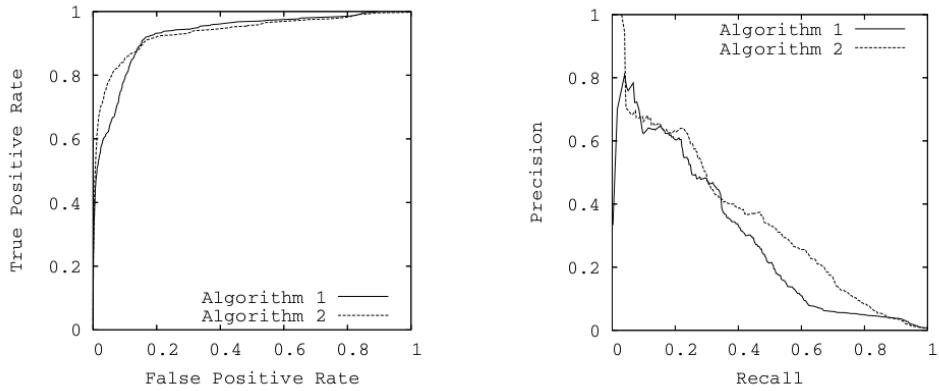


Figure 2.1: A sample receiver operating characteristic curve (left) and precision-recall curve (right). Each algorithm is evaluated on different thresholds and the points produced are used to obtain the curves. Image courtesy of [17]

For unbalanced data sets, “using the classifiers produced by standard machine learning algorithms without adjusting the output threshold may well be a critical mistake” [51]. 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 (disease) that is more interesting and harder to predict. Furthermore, PRAUC has been shown to have better properties than AUC in unbalanced data sets [17]. We introduce metrics tailored to image segmentation models in Sec. 2.4.

2.2 Artificial Neural Networks

Artificial neural networks or simply *neural networks* are one of the most popular nonlinear classifiers. Although initially inspired in the way biological neurons integrate information [40, 66, 52], they evolved to specialize in nonlinear modelling at the expense of biological adherence [53].

Multilayer feedforward neural networks are composed of L layers of *neurons*, the computation units. The first layer, called the *input layer*, has $s^{(1)} = n$ units and receives 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* (Fig. 2.2). 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 output from units in the previous layer and transmits the result to units in the next layer through their connections. Furthermore, every connection has a *weight* w that is learned in the training phase, i.e., the weights are the parameters Θ of the model. A neural network is *shallow* or *deep* according to its number of layers or *depth*: networks with one or more hidden layers are considered deep.

A unit computes a function of the form:

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

where $a_i^{(l)}$ is 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 $u = 1, \dots, L-1$ (defined below); $a_v^{(1)} = x_v$ 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 $l+1$. Equation 2.9 seems convoluted but it simply defines the activation of a unit as the weighted linear combination of the activations of units in the previous layer passed through a nonlinear function $g(\cdot)$.

Each layer (except for the output layer) includes a *bias unit* that outputs 1 regardless of its input ($a_0^{(1)} = 1$, $a_0^{(2)} = 1$, etc) allowing units in the next layer to learn a parameter Θ_{i0} to account for its own predisposition to activate. Bias units are included in the vectors $a^{(l)}$, hence, the summation in Equation 2.9 starts at 0 instead of 1.

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

$$g(z) = \max(0, z) \quad (2.10)$$

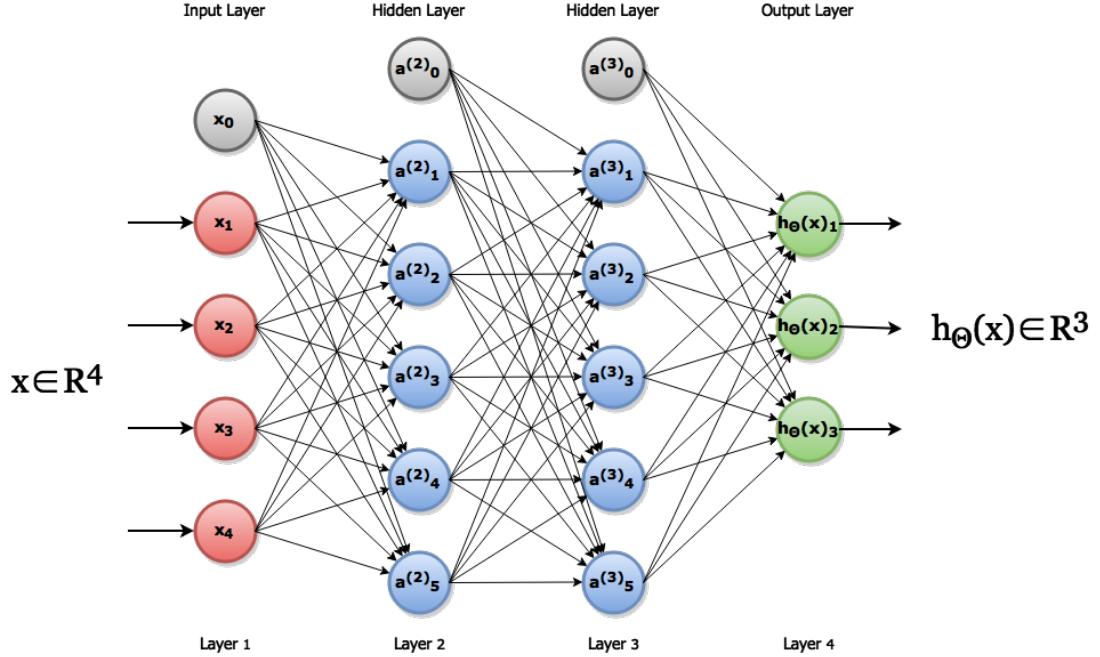


Figure 2.2: A small neural network: input layer with 4 units (red), two hidden layers of 5 units (blue) and output layer of 3 units (green). Bias units appear in gray. It approximates a function $h_\Theta(x) : \mathbb{R}^4 \rightarrow \mathbb{R}^3$, i.e., it classifies an input vector $x \in \mathbb{R}^4$ into 3 possible classes.

This nonlinear function and its derivative ($1_{z>0}$) are computed easily and, unlike sigmoid or tanh activation functions, are immune to vanishing and exploding gradients. Besides, it greatly accelerates convergence of gradient descent [33] and is currently the recommended activation function for deep neural networks [30].

The vector of activations in the output layer $a^{(L)} \in \mathbb{R}^{s^{(L)}}$, called a *score vector*, is the predicted (unnormalized log) probabilities $h_\Theta(x) \in \mathbb{R}^K$ that example x belongs to class $k \in K$. We can exponentiate each of these values and normalize them to obtain a probability distribution over the possible classes K ($p(x) \in [0..1]^K$); this improves interpretability and preserves original predictions.

Every unit produces a nonlinear activation $g(z)$ that is received by units in the next layer, linearly recombined with the activation of other units and passed again through the nonlinear function $g(z)$; these operations repeat until the processed input reaches the output layer. As a result, the network computes a function $h_\Theta(x)$ that is highly nonlinear on the original input x . This explains why neural networks are able to model complex functions and why increasing the number of layers increases its expressive power. It may be insightful to think of each unit as a feature detector: in the first hidden layer, units learn to detect simple features of the input, in the second hidden layer, units activate when a distinct combination of the simple features is found and so on. Thus, the network learns to recognize the most relevant features of the input learning more complex features as the number of units increases.

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

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

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. $L(\Theta)$ is differentiable with respect to Θ but non-convex, nonetheless, gradient descent usually converges to a good estimate of Θ [46]. *Error backpropagation* [36, 65], an algorithm to calculate the derivatives of the loss function with respect to Θ , computes error terms in the output layer and backpropagates them layer by layer using the chain rule of calculus.

Because many parameters need to be estimated, deep neural networks are susceptible to overfitting. The simplest approach to overcome this is using regularization. Regularization for neural networks is done by performing gradient descent on the regularized loss function presented in Equation 2.12

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

Dropout [59] 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 some probability p (usually 0.5-1), and gradient descent works on this sampled network (Fig. 2.3). 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 is interpreted as training many models (with shared weights) and averaging their results at test time.

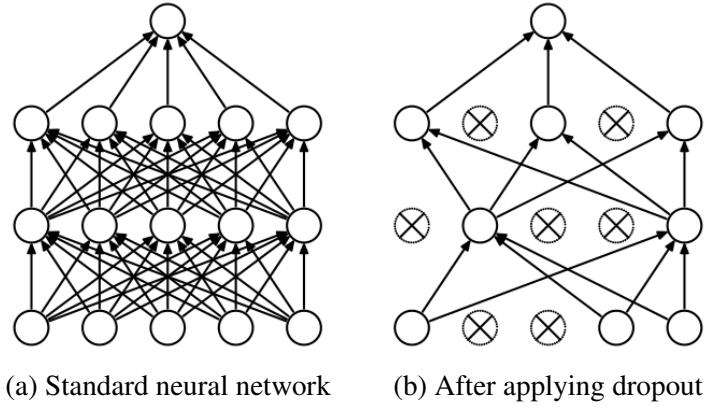


Figure 2.3: Dropout applied to a simple neural network. Crossed units are dropped. Image courtesy of [59].

2.3 Convolutional Networks

Convolutional networks are inspired by models of the visual cortex [20] but, like regular neural networks, favor practical performance over biological accuracy. LeCun et al. introduced modern convolutional networks in 1998 to successfully recognize handwritten digits from the MNIST data set [34]. Recently, Krizhevsky et al. achieved state-of-the-art performance on the ImageNet Large-Scale Visual Recognition Challenge [33], an image classification and object

localization challenge with 1000 categories [54]. Thanks to recent developments, convolutional networks have become one of the most popular methods for image classification and a driving force behind deep learning.

Due to the number of parameters, classifying images with regular neural networks is unfeasible; for instance, a small 100×100 grayscale image amounts to 10 000 units in the input layer—a 10 000-dimensional input vector—, therefore, a unit in the second layer would need to learn 10 000 parameters and a simple two-layer network with 100 units in its second layer would have 1 000 000 parameters. Besides, even if data and time requirements were unrestrictive, a regular neural network would destroy the original structure of the image hindering learning. Convolutional networks take advantage of the two-dimensional structure of images to reduce the number of parameters and facilitate learning.

Layers in a convolutional network are *sparingly connected*, i.e., a unit connects only to a small subset of the units in the previous layer, and *locally connected*, i.e., a unit connects to other units considering their position in the original image. Convolutional networks force *weight sharing* between units in the same layer, i.e., different units share the same parameters. Lastly, *pooling* subsamples the image reducing the spatial dimensions and adding invariance to local translations. All these features arise from the definition of convolutional networks, which we discuss below.

Each layer in a convolutional network is a set of *feature maps*, 2-dimensional grids of unit activations ($\mathbb{R}^{h \times w}$), arranged into a 3-dimensional *volume* ($\mathbb{R}^{h \times w \times d}$). The input layer is a volume ($\mathbb{R}^{h \times w \times c}$) that holds the input image of size $w \times h$ with c color channels. The output layer is a volume of size $R^{1 \times 1 \times K}$ where feature maps are single activations ($R^{1 \times 1}$) representing the final score for each class. The network receives an image x as an input volume that is transformed layer by layer into new volumes (whose dimensions may differ from previous ones) until it reaches the output layer of size $h_\Theta(x) = R^{1 \times 1 \times K}$ (Fig. 2.4).

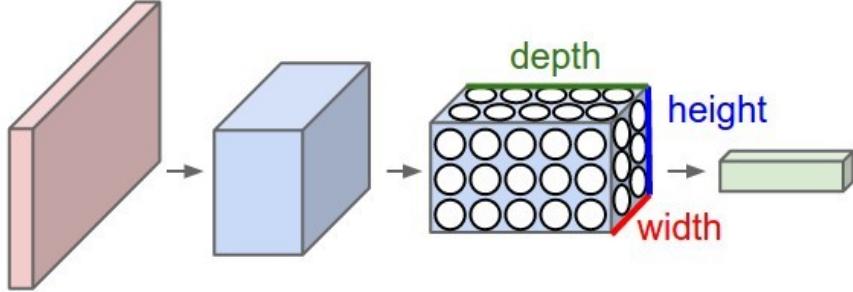


Figure 2.4: Transformations computed by a convolutional network. Input layer is shown in pink, hidden layers are shown in blue and output layer is shown in green. The third layer has 5 feature maps of size 2×3 (width is listed first by convention). Image courtesy of [30].

We build a convolutional network using four types of layers: convolutional layers, ReLU layers, pooling layers and fully connected layers; all of which compute a differentiable function on its input allowing error backpropagation.

Convolutional layer Convolutional layers are the heart of convolutional networks. They apply filters 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 applied to a volume with 10 feature maps will have 90 parameters ($\mathbb{R}^{3 \times 3 \times 10}$) (Fig. 2.5). A *feature map* is obtained by sliding a filter across the spatial dimensions of the previous volume calculating the dot product (a weighted sum) between the filter and the input at each position ^b. Many feature maps are computed separately (each with its own filter) and stacked together over the third dimension to form the output volume of the layer; these filters are the parameters that need to be learned.

We could think of each filter as looking for a specific feature on the input and the feature map as showing its likelihood at each position. If we regard each feature map as a grid of units, we notice that units connect only to a small local subset of units in the previous volume and that all units in the same feature map share the same weights (Fig. 2.5).

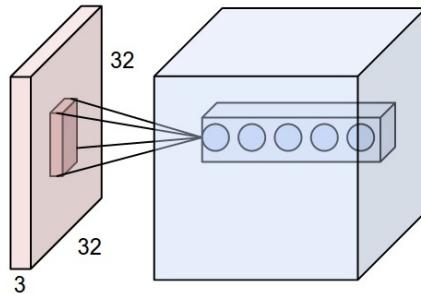


Figure 2.5: Convolutional layer applied to a volume ($\mathbb{R}^{32 \times 32 \times 3}$). The resulting volume has five feature maps as shown by the units in the blue volume ($\mathbb{R}^{32 \times 32 \times 5}$). Five small filters slide across the spatial dimensions (32×32) of the volume to produce the feature maps. Image courtesy of [30].

We choose four hyperparameters for this layer: number of filters, filter size, stride (the number of places to shift the filter at each step) and amount of padding around the volume; these define the shape of the resulting volume. The number of filters depends on the amount of distinct features we wish to learn, the filter size is usually small (3×3 to 9×9), stride is 1 and padding is usually $\lfloor (f_s - 1)/2 \rfloor$ where f_s is the filter size to preserve the spatial dimensions of the volume in the previous layer.

ReLU layer This layer performs an elementwise ReLU activation function to the volume in the previous layer, i.e., each value z in the volume is passed through the nonlinearity $\max(0, z)$. It outputs a volume with the same dimensions of the previous one and has no parameters to learn. A ReLU layer (or any other activation function) usually follows a convolutional layer so it is sometimes considered part of it; we separate them for clarity.

Pooling layer The pooling layer subsamples the volume in the previous layer reducing the size of its feature maps but keeping their number. Max pooling slides a fixed-size windows (normally 2×2) with stride 2 along each feature map and selects the maximum element on that space. This reduces each feature map dimension by half reducing the total number of activations by 75%, for instance, a 4×4 feature map gets subsampled to a 2×2 feature map

^bEach filter also adds a bias term.

where values are the maximum activation in each of the four quadrants of the original feature map. Pooling is applied to each feature map separately. A variant of max pooling uses 3×3 windows with stride 2, allowing for overlapping.

Fully connected layer A fully connected layer is a convolutional layer with $w \times h$ filters where w and h are the spatial dimensions of the volume in the previous layer and no padding, i.e., filters cover the entire volume, resulting in feature maps with size 1×1 . The output layer of a convolutional network is always fully connected with as many feature maps as possible classes.

The standard convolutional network architecture can be represented textually as:

INPUT \rightarrow [[CONV \rightarrow RELU] *N \rightarrow POOL?] *M \rightarrow [FC \rightarrow RELU] *K \rightarrow FC

where $*N$ indicates that the components are repeated N times, $?$ indicates an optional component and $N, M, K \geq 0$. We use this template to build a large range of models from a linear classifier INPUT \rightarrow FC ($N, M, K = 0$) to a regular neural network INPUT \rightarrow [FC \rightarrow RELU] + \rightarrow FC ($N, M = 0, K > 0$) to a convolutional network INPUT \rightarrow [[CONV \rightarrow RELU] + \rightarrow POOL?] + \rightarrow [FC \rightarrow RELU] * \rightarrow FC ($N, M > 0, K \geq 0$).

For example, a typical deep convolutional network could be:

INPUT \rightarrow [[CONV \rightarrow RELU] *2 \rightarrow POOL] *3 \rightarrow [FC \rightarrow RELU] *2 \rightarrow 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 not a standard way of counting the number of layers, we usually ignore ReLU and pooling layers because they lack learnable parameters. Therefore, our example network has 10 layers (21 in total), which is a good depth for big data sets. Practical advice on choosing an architecture is offered in Section 2.5.

Figure 2.6 shows a convolutional network with different kinds of layers. The image was obtained in a simulation accessible at cs231n.stanford.edu.

Lately, networks formed solely by convolutional layers have emerged [60, 24]: convolutional layers with stride greater than one perform pooling and the average of each feature map in the last convolutional layer is used for classification. This reduces the number of parameters enabling deeper architectures. *Transfer learning* is a related trend where we train a convolutional network on data from a specific domain and later reuse it to extract image features in a different domain or as an initial network to fine-tune with new data.

The loss function for a multiclass convolutional network is similar to that for a regular neural network (Eq. 2.12) except that, in this case, $h_{\Theta}(x)$ is defined as a convolutional network.

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

This function is differentiable with respect to Θ , thus, we can train the entire network via gradient descent. We calculate the gradient of the loss function with backpropagation.

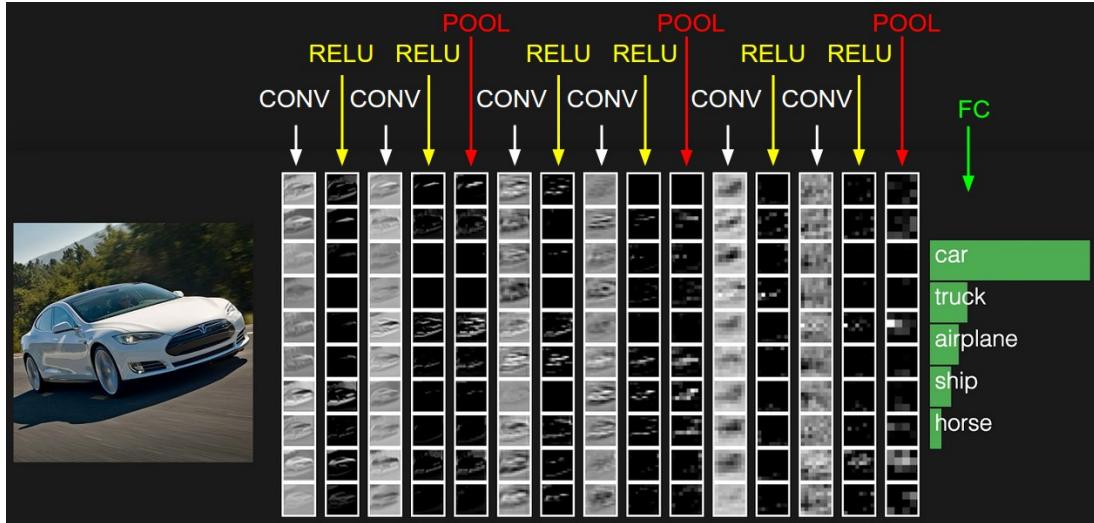
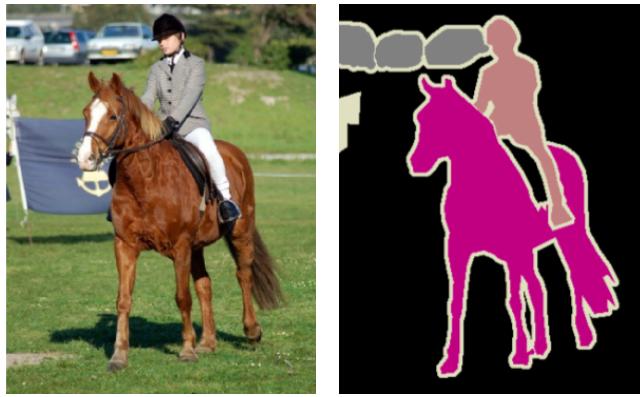


Figure 2.6: Convolutional network with architecture INPUT \rightarrow [[CONV \rightarrow RELU] * 2 \rightarrow POOL] * 3 \rightarrow FC. The input image has size 32×32 . Each hidden layer (a column) has 10 feature maps. Although the size of feature maps looks constant, each pooling layer reduces its dimensions by half (after the final pooling layer, feature maps have size 4×4). We show final scores for the five most probable classes. Image courtesy of [30].

2.4 Image Segmentation

Image segmentation is the task of labelling each pixel in an image according to the object to which it belongs (Fig 2.7). We can segment an image by training a classifier on small patches, sliding it across bigger images to obtain per-pixel predictions and assigning each pixel to its highest predicted class.



Convolutional networks are specially apt for image segmentation: we can transform

spatial dimensions ^c creating a network that is able to segment images of any size, albeit, due to subsampling in the pooling layers, it produces a coarse segmentation that needs to be upsampled to match the size of the original image. For instance, a convolutional network trained with 32×32 images and two pooling layers (that reduce the input by a factor of 4) acts as a 32×32 filter with stride 4 so for a 256×256 image it will produce a 64×64 segmentation that needs to be upsampled by a factor of 4 to recover the original dimensions. To account for the difference in size between the output of the network and the label image—the ground truth segmentation—, we downsample the label image or append an upsampling layer at the end of the network [39]; the additional layer computes a differentiable function either fixed such as bilinear interpolation or learned such as a linear mapping. Alternatively, we could remove pooling layers and use *dilated convolutions*, filters with spaces between any two values (Fig. 2.8), thus enlarging the receptive field of the filter as pooling would do but keeping the spatial dimensions of the volume and the number of parameters fixed [68].

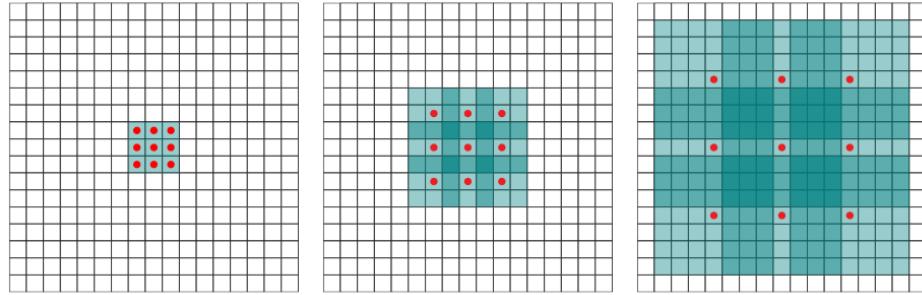


Figure 2.8: Example of a 3×3 convolutional filter at increasing dilations: zero, two and four. The spatial dimensions of the input volume are represented as a black grid, red dots signal the places where the filter is applied and the green shadows represent the effective receptive field of the filter. Courtesy of [68].

Lastly, the loss function sums the loss over all pixels so the network performs a single gradient update after one example segmentation. Convolutional networks transform image segmentation into an end-to-end, learnable task.

To evaluate the model, we evaluate each segmentation using standard metrics (Sec. 2.1) and average over all test images. *Pixel accuracy*, for instance, is the average proportion of correctly classified pixels in an image. Another popular metric, *intersection over union* or *IOU*, is calculated for a single segmentation as $TP/(TP + FP + FN)$ (Tab. 2.1). In medical diagnosis, the *free-response operating curve* or FROC curve evaluates a radiologist or CAD system at localizing lesions: for varying thresholds, the FROC curve plots the sensitivity of the system (proportion of lesions correctly localized) versus the number of false positives committed per image. We can compare between models using their sensitivity at a fixed number of false positives per image, for instance, sensitivity at 1 FP/image.

If we are interested in a particular class, e.g. object vs. background, we present a single score map showing its probability distribution across every position in the image. We can refine this map using gaussian smoothing, cluster-based thresholding and conditional random

^cPadding by $\lfloor (f_s - 1)/2 \rfloor$ where f_s is the filter size.

fields among other techniques. Finally, to produce a segmentation we threshold the post-processed map at some specific value, which can be chosen using a validation set.

2.5 Practical Deep Learning

In this section we collect guidelines for building as well as efficiently training deep convolutional networks; while they are specific to this thesis, they may prove useful in similar projects. Deep learning is a fast-changing field so these recommendations may soon outdated.

Image preprocessing Convolutional networks handle raw image data, but some level of preprocessing speeds training and improves performance.

- Crop images to contain only the relevant regions, denoise, enhance and resize them to maintain the input size fixed and manageable.
- Zero-center each image feature (the raw pixels) by subtracting its mean across all training images. Optionally, normalize each zero-centered feature to range from $[-1 \dots 1]$ by dividing them by its standard deviation [30].
- Test data should not be used to calculate any statistic used for preprocessing. Furthermore, the same statistics (calculated from training data) should be used to preprocess test data [30].

Convolutional network architecture Designing convolutional networks involves many decisions, we provide recommendations for the architecture and sensible values for related hyperparameters.

- Select a network architecture flexible enough to model the data and manage overfitting rather than a simpler architecture that may be incapable to model the data [46, 33].
- 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), practically, deeper architectures produce better results using less units overall. This holds for convolutional networks [8].
- Use at least 8 layers (not counting pooling or ReLU layers) for big data sets, use less layers or transfer learning for small data sets. “You should use as big of a neural network as your computational budget allows, and use other regularization techniques to control overfitting.” [30]
- Use 2-5 CONV \rightarrow RELU pairs before pooling (N above) [30]. Pooling is a destructive operation, placing two convolutional layers together allows them to detect more complex features.
- Use 1-8 [CONV \rightarrow RELU] $^+$ \rightarrow POOL blocks (M above). This hyperparameter regulates the representational power of the architecture. The exact number depends on the complexity of the features in the data and the computational resources available. It also defines how much the volume is subsampled.

- Use less than 3 FC \rightarrow RELU pairs before the output layer (K above) [30]. The volume that arrives to fully connected layers is already complex, adding many layers only increases the number of parameters and risks overfitting.
- The number of filters per convolutional layer controls the number of features detected at that layer—similar to the number of units per layer in a regular neural network. A common pattern is to start with a small amount of filters and increase them layer by layer [56].
- The number of filters per fully connected layer decreases layer by layer^d. For instance, for a convolutional network with ten possible classes and two fully connected layers, 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—the 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 [58]. If the input size is too big, use a bigger filter in the first convolutional layer [30].
- Use 2×2 pooling with stride 2. Overlapping max pooling may produce slightly better results but slows training and causes resizing headaches [33, 18].
- Use square input images (width = height) with spatial dimensions divisible by 2 at least as many times as the number of pooling layers in the network.
- Use number of parameters to measure the complexity of an architecture rather than number of layers or units.

Hyperparameters About setting and searching hyperparameters other than those of the network architecture.

- Use a single sufficiently large validation set (15-30% of data) rather than cross validation [8]. Use cross validation in very small data sets [46].
- Use random search rather than grid search. Random search draws each parameter from a value distribution rather than from a set of predefined values. [9]
- Search for the best combination of hyperparameters rather than each individually.
- Train each combination of hyperparameters for 1-2 epochs to narrow the search space; then, train for more epochs on these ranges [30]. Explore further when the best value for a hyperparameter is found in the limit of the range. [7].
- Partial convergence is sufficient to assess hyperparameters [30].

^dThe number of units in a convolutional layer is the number of units in a feature map times the number of feature maps.

- Hyperparameters related to the convolutional architecture, e.g., number of layers, number of filters and filter sizes are set manually (as explained above) rather than using a validation set.
- Several hyperparameters are set: initial learning rate α , learning rate decay schedule, regularization strength λ , Adam hyperparameters (β_1 , β_2 and ϵ), probability of keeping a unit active in dropout p , mini-batch size and type of image preprocessing.
- We could fit all hyperparameters using a validation set but, in practice, this is computationally unfeasible and results in overfitting 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 continuous uniform distribution) [30].
- The regularization strength λ is usually data (and loss function) dependant. It ranges from 10^{-3} to 10^4 . Search in log scale ($\lambda = 10^{unif(-3,4)}$).
- Divide the learning rate by 10 every time the validation error stops improving or every fixed number of epochs chosen by observing when it stops improving in a similar network [33, 24].
- Use $\beta_1 = 0.9$, $\beta_2 = 0.995$ and $\epsilon = 1 \times 10^{-6}$ [30].
- 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 [59]. Less dropout is used on the first layers because they have less parameters [30].
- Use mini-batch size of 64 or 32. A larger batch size requires more memory and training time. Test performance is unaffected [7].
- Choose among standard preprocessing techniques by (qualitatively) inspecting results on images from the validation set. If none seems superior, fit it along α and λ

Training Convolutional networks have millions of parameters and require careful training. We offer general advice and list some commonly used techniques.

- Shuffle training examples before each epoch to ensure that examples in every mini-batch are sampled independently [7].
- Multiply the number of examples by 25-100 to estimate the maximum number of parameters that could be learned (assuming data augmentation). Groups have learned up to 40M parameters from as little as 60K training examples [18, 58].

- Weight initialization is vital for convergence. Initialize each filter weight as a value from a normal distribution $\mathcal{N}(\mu = 0, \sigma = \sqrt{2/n_{in}})$ where n_{in} is the number of connections to the filter (90 for a 3×3 filter with depth 10) [25]. In code, `w = randn() * sqrt(2/nIn)` where `randn()` returns values drawn from a standard normal distribution. Initialize biases to zero.
- Use mini-batches rather than the entire training set to compute the gradient of the loss function. Mini-batches allows us to make more updates, more frequently resulting in faster convergence and better test results [7].
- Use (inverted) dropout [33] as a complement to l_2 -norm regularization. Dropout improves results but may slow network convergence.
- Batch normalize the output of each convolutional layer [28]. This speeds convergence (using higher learning rates and faster decay) and reduces the need for dropout.
- Use leaky rectified linear units (`max(0.1x, x)`) rather than simple rectified linear units. Leaky ReLUs slightly improve results at almost no cost [67].
- Use Adam [32] to update weights. Adam incorporates momentum and per-parameter adaptive learning rates. Nesterov's Accelerated Gradient is also a viable option [30].
- Store network parameters regularly during training. This allows us to inspect the network at different stages, retrain one or select the one with the best validation error [8].
- Stop training if the validation error has not improved since the last learning rate reduction: gradient descent may not have converged but the validation error will start to increase (overfit) [7].
- If you used test set results to refine a model, shuffle the entire data set and choose a different training and test set to avoid overfitting to the test set [46].

Sanity checks We perform some simple tests to make sure training works properly.

- After weight initialization, run a test on a small set of examples to assert that the network predicts similar scores for every class, the loss function without regularization equals $-\log(1/K)$ and adding regularization increases the loss [30].
- Run a gradient check if backpropagation seems faulty. Gradient checks compare the analytic gradient obtained via backpropagation with a numerical gradient obtained via a finite difference approximation [30].
- Train the network (without regularization) in 10-40 examples to check that the loss function becomes zero. If the network is unable to overfit a tiny subset of data, flexibilize the model. [46].

- During training, the loss function evaluated on training examples decreases monotonically ^e. If not, check for implementation errors, reduce the learning rate or augment momentum [30].
- Monitor the loss function during training to identify underfitting, characterized by high training loss and high validation loss, and overfitting, characterized by low training loss and high validation loss [46].

Image segmentation We usually require to post-process, upsample and threshold the output of a convolutional network to produce a valid segmentation. Given that the network is already trained, using the validation set to choose the best techniques is virtually free.

- For big input images, use smaller mini-batches to avoid overloading the GPU memory [30].
- Set the post-processing technique using the validation set. Conditional random fields are specially effective for refining convolutional network scores [15].
- Use bicubic (or bilinear) interpolation for upsampling [15]. If the upsampling factor is greater than 16, use a learnable upsampling layer or skip layers [39].
- Set the segmentation threshold using the validation set; this hyperparameter is equivalent to the threshold used in classification.

Data augmentation We reduce overfitting in image data by applying label-preserving transformations to the original images to generate additional examples. Transformations include rotations, translations, horizontal and vertical reflections, crops, zooms and jittering (adding noise to the colors). The network receives different views of the object and learns invariant features; for instance, if we present images of books at different rotations, we expect it to learn to identify a book on any orientation.

- Exploit invariances you expect in the data set, e.g., galaxies are rotation-invariant because in space there is no up or down [18].
- Be careful to preserve the original label of the image. Applying many transformations may cause the image to lose its meaning.
- Generate the augmented images during training to save storage [33].
- If applying many affine transformations, combine them into a single operation. This is faster and reduces information loss [18].
- Optionally, use data augmentation at test time: present the network with different versions of the image and average its predictions [33].

^eBecause of momentum and regularization it could increase slightly before decreasing.

Unbalanced data In practice, it is common to have very few examples of one class compared to the rest. Managing unbalanced data sets is still an open problem.

- For binary classification with rare positive class, use PRAUC as a performance metric. If the threshold is selected using a validation set, F_1 score is also valid [17].
- For multiclass classification, use the macro-averaged F_1 score, an average of F_1 scores per class, with validated thresholds [48]. A multiclass PRAUC exists but is not used.
- If using an appropriate metric is impractical, divide each predicted class probability by the prior probability of the class and renormalize [11]. This rescales predictions to account for the original imbalance in the class distribution.
- Avoid oversampling, repeating examples from the rare class, and undersampling, discarding examples from the dominant class, because they do not add any information to the data set.
- If rare examples are insufficient for learning, augment them more or balance each mini-batch via stratified sampling; the latter is oversampling.

Software We shortly describe four of the most popular packages for deep learning. All have similar capabilities and are available to the public (open-source).

- Tensorflow [1] is a Python/C++ library released by Google that supports automatic differentiation on data flow graphs—neural networks, included—, distributed training on clusters of CPUs and GPUs, graph visualization and easy deployment in multiple platforms. It has been quickly adopted by the deep learning community.
- Theano [10, 6]: Theano is a Python library developed in Python/CUDA at the University of Montreal that performs symbolic differentiation on computational graphs. It is tightly integrated with NumPy and uses the GPU to evaluate mathematical expressions involving multidimensional arrays.
- Torch [16]: Torch is a scientific computing framework developed in C/Lua/CUDA at the IDIAP Research Institute. It offers multidimensional arrays (tensors), automatic differentiation, a command line and Lua interface, GPU support and easy building of complex neural network architectures.
- Caffe [29]: Caffe is a deep learning framework developed in C++/CUDA by the Berkeley Vision and Learning Center (BVLC) and community contributors. It offers a Python, Matlab and command line interface; reference models and tutorials; and fast code with easy GPU activation.

We acknowledge that mammographic data is different from regular image segmentation data: labelling is imperfect, image sizes and ratio change, images are bigger, quality varies, objects of interest are small in relation to the background, data sets are smaller, texture is uniform across the image, among others. Therefore, some of the advice given above may prove counterproductive. Whenever possible, design decisions should be based on data and compiled results.

2.6 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 [43].

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 (Fig. 2.9). 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 [27].

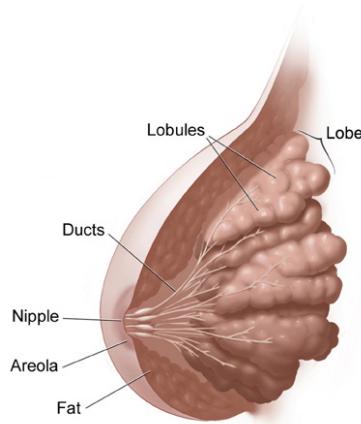


Figure 2.9: Anatomy of the female breast. Image courtesy of [43].

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.

2.6.1 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 [44]. A standard mammogram is shown in Figure 2.10.

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

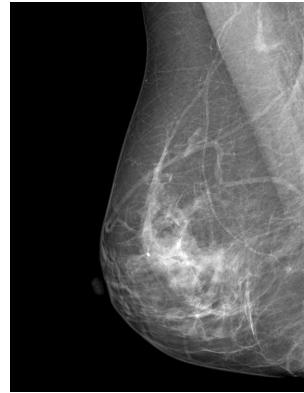


Figure 2.10: A standard mammogram.

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 [44].

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 (Fig. 2.11). *Breast masses* or breast lumps are a variety of things: fluid-filled cysts, fibrotic 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 (Fig. 2.11). 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].

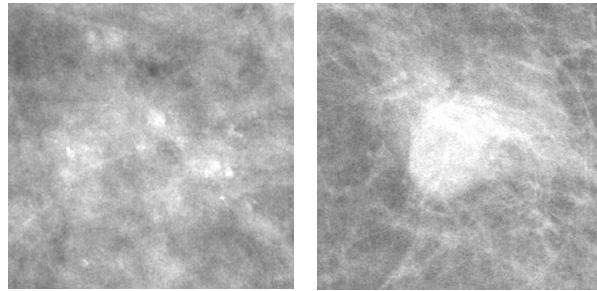


Figure 2.11: Signs of possible breast cancer in a mammogram. Left: A cluster of microcalcifications in an irregular layout. Right: A poorly defined breast mass.

Conventional mammography records mammograms in film; *digital mammography*, on the other hand, converts x-rays into electrical signals and stores images electronically. Digital mammograms offer a clearer picture of the breast and ease manipulation and sharing between health care providers. Although researchers still debate whether they offer an advantage over film mammograms [31, 50, 57], digital mammography has become standard in breast cancer screening. Figure 2.10 is, in fact, a digital mammogram. *Digital tomosynthesis* is a new technology that produces three-dimensional x-ray images of the breast and improves the efficacy of mammograms in certain scenarios [61].

Computer-aided detection (CAD) systems assist radiologists during screening signaling

suspicious areas and displaying relevant information. Detection or CADe looks for any type of lesion while diagnosis or CADx focuses on malignant lesions. Whether their use improves accuracy has been challenged [35].

2.6.2 Mammographic databases

Researchers use data from previous patients—mammograms, segmentations and clinical features—to train and evaluate their models. *Contrast resolution*, the number of gray values per pixel, and *spatial resolution*, breast area per pixel, dictate the quality of a digital mammogram: 12-bit images ($2^{12} = 4096$ gray values) with pixel size of at most 0.1mm are preferred. Many publicly available databases, including those described below, satisfy these conditions.

The Digital Database for Screening Mammography (DDSM) [26] is the most popular database used for CAD development. It is composed of around 10.5K digitized film mammograms from 2620 patients. Mammograms are either 12-bit or 16-bit images with 0.05 mm spatial resolution. Lesion segmentations are provided along with its type, assessment, subtlety and malignancy.

The BancoWeb database [45] consists of around 1.5K digitized film mammograms from 300 Brazilian patients. Mammograms are 12-bit images with 0.075 or 0.15 mm pixel size. Although few lesions are segmented, this repository may be useful to assess performance in Latin American patients. Its current state is unknown.

Lastly, the Breast Cancer Digital Repository (BCDR-DM) [42, 41] consists of around 1.2K digital mammograms from 237 patients. Mammograms are 8-bit images with 0.07mm spatial resolution. Lesion segmentations are provided as well as their assessment, hand-crafted texture and shape features and relevant clinical data.

This section was written using information from the National Cancer Institute. We recommend to visit its website (www.cancer.gov) for further details.

2.7 Convolutional Networks in Breast Cancer Research

Traditional CAD systems for breast cancer are composed of successive stages: preprocessing and image enhancement, localization of suspicious regions, feature extraction from suspect image patches, feature selection and region classification using machine learning. Tang et al. [63] review the state-of-the-art in CAD systems. We focus on convolutional networks applied to lesion detection and diagnosis in mammographic images.

Overview In 1995, Sahiner et al. [55] used simple convolutional networks to detect masses. Although results were competitive, the research community favored feature-based classifiers, such as regular neural networks, that employed expert knowledge and advanced image techniques. During that time, Lo et al. [37, 38] used convolutional networks to detect individual microcalcifications. This work was continued by Gurcan et al. [23] who selected an optimal architecture to detect individual microcalcifications. The optimized network played a small role in two CADe systems for clustered microcalcifications: one for film mammography [23] and one for digital mammography [22]. Until this point, researchers used few data to train

networks (2 to 3 layers, less than 10 thousand parameters) without modern features to classify preselected regions. Recently, Arevalo et al. [4] diagnosed breast masses with a bigger convolutional network (4 layers, 3.4 million parameters) that incorporates many deep learning advances; it outperformed hand-crafted and image-based features. Lastly, Dubrovina et al. [19] trained a convolutional network to perform segmentation of different breast tissues; despite the small network and data set, results were promising. To the best of the author knowledge, this is the only attempt to use supervised convolutional networks for mammographic segmentation^f. These last studies are the most relevant to this thesis.

In summary, convolutional networks have been used sporadically for breast cancer detection and diagnosis but they have not been used for lesion segmentation or trained with big data sets of digital mammograms.

The following sections expand on the work mentioned above. Architectural details are compiled in Table 2.2

2.7.1 Detection of breast masses

The first attempt to use convolutional networks for breast cancer is reported in "Detection of masses on mammograms using a convolution neural network" [64]. This four-page article was expanded in [55].

They used a small convolutional network (2 layers, $\sim 1K$ parameters) to detect masses. The data set consisted of 672 manually selected potential masses from 168 digitized mammograms: out of which 168 were positive examples and 504 were dense or fatty tissue. Background reduction was performed using a rather convoluted method. The original images 256×256 (equivalent to 2.56×2.56 cm) were downsampled via non-overlapping average pooling to size 16×16 ; downsampling to 32×32 pixels was also tried and produced similar results. 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). 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 from the initial 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. Authors showed that network architecture was not as important for performance as providing the network with texture information. Texture features give back some of the information lost during downsampling, which explains the improvement. Authors also acknowledge that the network architecture is far from optimal given its simplicity (one convolutional layer with three filters) and the incomplete hyperparameter search. A deeper network with bigger input size could produce better results without the need for handcrafted texture features.

^fPetersen et al. [49] segmented breast tissue using a convolutional autoencoder.

2.7.2 Detection of microcalcifications

The first use of convolutional networks to detect microcalcifications is reported in [37]. They performed various experiments on a small convolutional network (3 layers, $\sim 5.4\text{K}$ parameters). The input size (16×16), number of convolutional layers (2) and kernel size (5×5) were chosen using a validation set, although few options were explored: input sizes of 8, 16 and 32; one and two hidden layers and kernel sizes of 2, 3, 5 and 13. A high sensitivity image technique was used to obtain a set of 2104 image patches (16×16 pixels equivalent to 0.17×0.17 cm) of potential microcalcifications from 68 digitized mammograms; of these, 265 were microcalcifications and 1821 were “false subtle microcalcifications”. Prior to training, a wavelet high-pass filter 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 ^g. Other performance metrics were not explicitly reported. This article showed that deeper networks, background removal and data augmentation improved results. Together with [55], it proved that simple convolutional networks can be used to detect breast cancer lesions.

A convolutional network with a similar architecture (3 layers, $\sim 4.5\text{K}$ parameters) was presented by the same group in [38]. It detects microcalcifications from 16×16 image patches that were pre-selected and preprocessed using the same techniques. For these experiments, nonetheless, they used 38 digitized mammograms and extracted 220 microcalcifications and 1132 negative examples 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 [37]). It showed that a convolutional network outperforms a regular neural network and a DYSTAL network in detecting clustered microcalcifications when using raw pixels as input features.

Gurcan et al. [23] optimized the filter size and number of filters of each convolutional layer of a network used to detect clustered microcalcifications (3 layers, $\sim 7.6\text{K}$ parameters). The convolutional network was part of a CAD system that identifies and enhances the breast area via a bandpass filter, segments potential microcalcifications with adaptive thresholding methods, filters the suspect areas with a rule-based classifier, classifies the remaining image patches with a convolutional network and clusters individual microcalcifications to obtain the detected clustered microcalcifications. The network was trained on 1117 image patches (16×16 pixels equivalent to 0.16×0.16 cm) obtained from 108 digitized mammograms without data augmentation. The best architecture had 14 feature maps on the first convolutional layer with filter size 5×5 and 10 on the second layer with a 7×7 filter. Details about the hyperparameter search or network training are not provided. The CAD reached 84.6% sensitivity at 0.7 false cluster detections per image. The article shows that optimizing the network architecture improves CAD performance significantly; nonetheless, given the small training set and incomplete hyperparameter search results may vary.

^gThis evaluation is not clearly explained in the article or in [38] so our interpretation may be incorrect. This affects the validity of the reported results.

Ge et al. [22] tailored the CAD system developed in [23] to digital mammograms. It consists of seven stages: preprocessing via inverted logarithmic transformation, image enhancement via box-rim filter, segmentation of potential microcalcifications via thresholding, classification of individual candidates via rule-based classifier and convolutional network, regional clustering via neighborhood growing, stepwise LDA feature selection and classification via LDA. The convolutional network had the same optimized architecture and was trained on around 500 16×16 image patches obtained from 48 digital mammograms: half of which were microcalcifications. Its threshold was manually set to 0.4. The network reached 0.96 AUC for the detection of individual microcalcification.

These CAD systems were compared in [21]: digital mammograms considerably improved performance. This seems intuitive given that digital mammograms have less noise which allows the system to pick up subtler details without the need for manual enhancement.

2.7.3 Diagnosis of breast masses

Arevalo et al. [4] trained a convolutional network (4 layers, $\sim 3.4M$ parameters^h) with modern deep learning techniques to diagnose masses. The data set contained 426 benign and 310 malignant masses obtained from 736 digitized mammograms. Lesions were cropped tightly, resized to 150×150 pixels and augmented (original and flipped images were rotated at 0° , 90° , 180° and 360°). Pixels were normalized globally—subtracting the mean intensity of the 150×150 patch—and locally—subtracting the mean intensity of an 11×11 neighborhood and dividing by its standard deviation. The final architecture was chosen among 25 architectures using a validation set. The network extracts image features and forwards them to a separate linear classifier for scoring. This setup competed against linear classifiers trained on HOG features, HGD features, hand-crafted features and features from a convolutional network trained on natural images. The convolutional network trained on mammographic images achieved the best result: 0.86 AUCⁱ. Moreover, adding hand-crafted to network-produced features proved ineffective. This work showed that convolutional networks outperform image-based and hand-crafted features in breast cancer diagnosis. They also learned that normalization and additional convolutional layers facilitate learning.

2.7.4 Tissue segmentation

A convolutional network was used in [19] to segment breast tissue into four categories: pectoral muscle, fibroglandular tissue, nipple and general breast tissue. A network was trained on approximately 800 000 overlapping patches (61×61 pixels) cropped from 39 digital mammograms and tested on a single mammogram; this was repeated for every image and results were averaged. Pixels were zero-centered but not further enhanced. The network architecture (6 layers, 34K parameters) was chosen by hand; the network is deep although simple. It was trained using stochastic gradient descent with momentum, learning rate decay, weight decay, dropout and mini-batches of 256 patches. Its output was preprocessed by filling regions and deleting clusters smaller than a predefined threshold. This network reached 0.55 mean IOU.

^hThey actually learned 4.6 million parameters but many were redundant.

ⁱThis may be optimistic due to flawed experiment design: examples used to choose the model could belong to the test set.

They proved that convolutional networks can perform mammographic image segmentation even with little labelled data. Bigger networks trained with more examples could improve results.

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

Article	Goal	Architecture	Volumes	Filters	# Params
[55]	Detect masses	INPUT -> CONV -> SIGMOID -> FC -> SIGMOID	16 × 16 × 3 7 × 7 × 3 1 × 1 × 1	10 × 10 7 × 7 5 × 5	1047 5436
[37]	Detect individual microcalcifications	INPUT -> [CONV -> SIGMOID] * 2 -> FC -> SIGMOID	16 × 16 × 1 12 × 12 × 12 8 × 8 × 12 1 × 1 × 2	5 × 5 5 × 5 8 × 8	
[38]	Detect individual microcalcifications	INPUT -> GAUSSIAN -> [CONV -> SIGMOID] * 2 -> FC -> SIGMOID	16 × 16 × 1 12 × 12 × 10 8 × 8 × 10 1 × 1 × 2	5 × 5 5 × 5 8 × 8	4530
[23]	Detect individual microcalcifications	INPUT -> [CONV -> SIGMOID] * 2 -> FC -> SIGMOID	16 × 16 × 1 12 × 12 × 14 6 × 6 × 10 1 × 1 × 1	5 × 5 7 × 7 6 × 6	7570
[22]	Detect individual microcalcifications	INPUT -> [CONV -> SIGMOID] * 2 -> FC -> SIGMOID	16 × 16 × 1 12 × 12 × 14 6 × 6 × 10 1 × 1 × 1	5 × 5 7 × 7 6 × 6	7570
[4]	Diagnose masses	INPUT -> [CONV -> RELU -> POOL] * 2 -> MAXOUT -> SOFTMAX	150 × 150 × 1 140 × 140 × 64 35 × 35 × 64 32 × 32 × 64 8 × 8 × 64 1 × 1 × 400 1 × 1 × 2	11 × 11 5 × 5 4 × 4 4 × 4 8 × 8 1 × 1 3.35M	

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

Article	Goal	Architecture	Volumes	Filters	# Params
[19]	Segment tissue	INPUT -> [CONV -> RELU -> POOL] * 3 -> [FC -> RELU] * 2 -> SOFTMAX	61 × 61 × 1 55 × 55 × 16 27 × 27 × 16 23 × 23 × 16 11 × 11 × 16 6 × 6 × 16 3 × 3 × 16 1 × 1 × 128 1 × 1 × 16 1 × 1 × 4	7 × 7 3 × 3 5 × 5 3 × 3 5 × 5 3 × 3 5 × 5 3 × 3 1 × 1 1 × 1 1 × 1	34324

2.8 Summary

Machine learning studies mathematical models and the algorithms used to estimate their parameters from data. A machine learning model especially suited to images, convolutional networks, has produced state-of-the-art results in a variety of computer vision tasks, including image segmentation—assigning a class to each pixel in an image. Breast cancer detection can be framed as segmentation by assigning breast cancer lesions (breast masses and micro-calcifications) to the positive class and normal breast tissue to the negative class; convolutional networks can then be used to separate lesions from normal breast tissue. Use of convolutional networks for similar tasks has been reported in the literature but this work will be the first report of end-to-end lesion segmentation.

Chapter 3

Solution

In this chapter, we describe our experiments, justify design decisions and detail our implementation. A high-level overview of our solution is provided in Fig. 3.1.

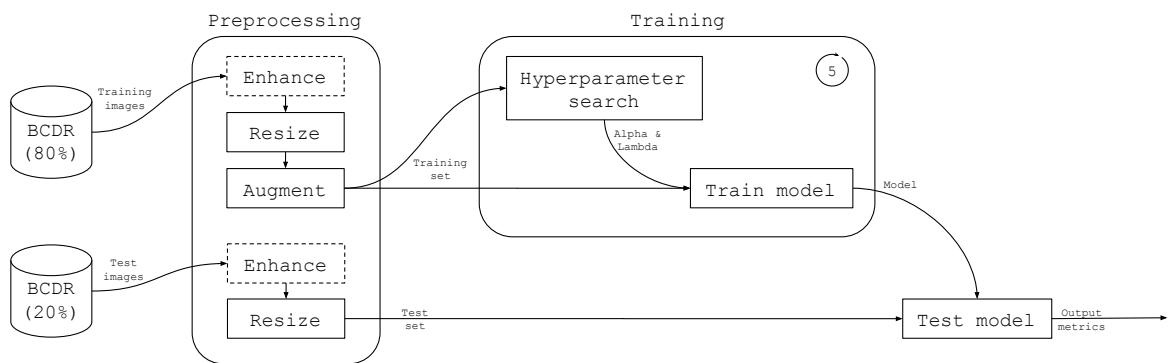


Figure 3.1: Overview of our solution. We divide the database into training and test patients, train a convolutional network with the training set (we select hyperparameters α and λ using 20% of the training set for validation) and evaluate our model in the test set. We repeat each experiment for five folds. Experiments test different preprocessing and network configurations.

3.1 Task definition

We segment digital mammograms into two separate regions: breast mass (benign or malignant) and general tissue. Breast area is previously separated from the background by simple thresholding. In particular, we train a convolutional network to estimate the probability of each pixel belonging to a mass and evaluate these predictions using FROC.

3.2 Data set

We document the retrieval, enhancement and augmentation of our data set.

3.2.1 Database

We use the Breast Cancer Digital Repository (BCDR-DM) database, specifically, the BCDR-D01 data set, which is composed of patients with at least one breast mass. We select 256 digital mammograms from 63 patients. A patient with breast implants (511) was ignored. Digital mammograms have higher image quality and lack any marks or scanning artifacts present in digitized film mammograms; this allows the network to learn sharper features easing segmentation. We overcome having few mammograms at our disposal by augmenting our data set and training on overlapping patches.

Mammograms in the BCDR-D01 data set are 8-bit grayscale images with 0.07mm spatial resolution sized 3328×4084 , 2816×3072 or 2560×3328 pixels equivalent to 23.3×28.6 , 19.7×21.5 and 17.9×23.3 centimeters. The data set provides the segmentation, type (mass, microcalcification, calcification, axillary adenopathy, architectural distortion or stroma distortion) and biopsy result (benign or malignant) of each lesion. We disregard patient data (age and breast density) and image features (intensity, texture, shape and location descriptors).

We generate our labels thresholding the mammogram to zero to separate the background and using the provided lesion outlines to separate the lesions. Masses (benign or malignant) appear as white; breast area as gray and background as black (Fig. 3.2a).

3.2.2 Data division

For each fold, we randomly assign 80% of patients to the training set and 20% to the test set (Tab. 3.1). In total, our data set counts with 63 patients, 256 mammograms and 139 lesions.

	Patients		Mammograms		Masses	
	Training	Test	Training	Test	Training	Test
Fold 1	50	13	189	67	106	33
Fold 2	50	13	209	47	112	27
Fold 3	50	13	204	52	110	29
Fold 4	51	12	209	47	110	29
Fold 5	51	12	213	43	118	21
Average	50.6	12.6	204.8	51.2	111.2	27.8

Table 3.1: Data set summary

3.2.3 Image enhancement (Exp. 1.3 and Exp. 3)

We zero any pixel below the mean pixel intensity of the image (calculated only on the breast area) and scale the rest linearly to cover the entire intensity range (0-255); this reduces all small variations in the background to black and increases the contrast of the image (Fig. 3.2b).

Background reduction plus contrast normalization highlights breast masses, which are brighter than normal breast tissue; normalizes images from patients with darker or lighter tissue and improves convergence [4]. However, it may destroy important texture information by blending it with the background or cause false positives by highlighting dense tissue.

3.2.4 Resizing

Our convolutional networks have an effective receptive field (spatial dimensions around a pixel that affect its prediction) of roughly 128×128 pixels. We resize our images to contain 2×2 cm in this area—roughly a 2.2 downsampling factor ^a (Fig. 3.2c). Considering that masses are rarely bigger than 2cm (length of the long axis) [55], the network sees a good portion of the lesion during classification.

We resize images with PILLOW, the Python Image Library, using Lanczos interpolation for mammograms and nearest neighbor interpolation for labels. Lanczos interpolation is a high quality downsampling filter recommended by PILLOW and nearest neighbor interpolation assures that the reduced label contains only valid values (white, gray and black).

3.2.5 Cropping

We calculate the bounding box of the breast area in the label image and crop the mammogram and the label to delete unnecessary black spaces (Fig. 3.2d). Because our networks downsample images to later upsample them by the same factor, we make sure the dimensions of the cropped image are multiples of this factor (cropping a slightly bigger area if needed) to recover the exact dimensions after upsampling.

3.2.6 Data augmentation

We mirror each image (mammograms and labels) and rotate the original and mirrored version at 0, 90, 180 and 270 degrees to increase our training set by a factor of 8. Images in the test set are not augmented.

This transformations are common when training convolutional networks with small data sets. In principle, the test set should not be augmented as it is a proxy for real data.

3.2.7 Storage

All mammograms and their respective labels are stored as grayscale 8-bit images preserving their original names (plus a suffix) and folder organization. The entire data set before augmentation weights approximately 120 megabytes.

3.3 Experiment 1

We use a simple architecture to have a baseline performance for convolutional networks and test the effect of using a weighted loss function and enhancing the input images.

^aWe use 96×96 for Experiment 1, whose network has a smaller receptive field.

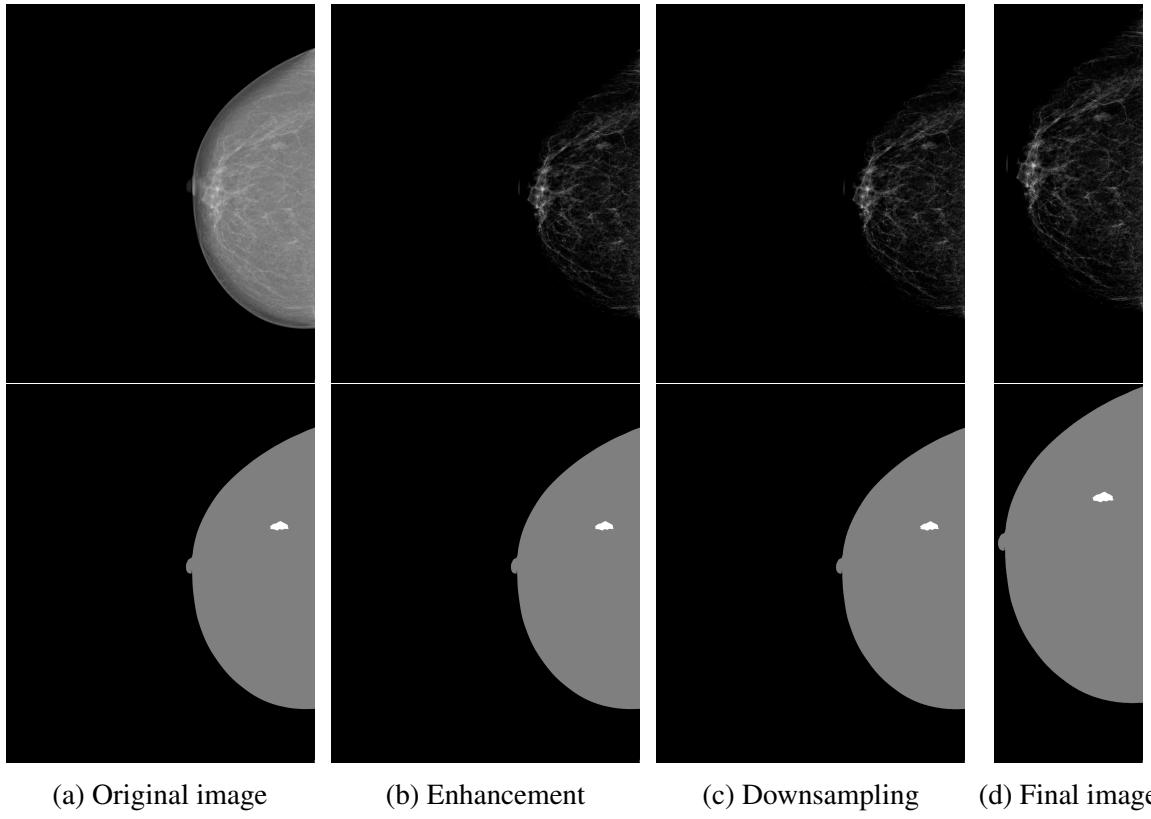


Figure 3.2: A mammogram (top) and its label (bottom) being preprocessed: (1) original images (4084×3328), (2) background reduction plus contrast normalization, (3) downsampling and (4) cropping to delete black spaces (560×1424). Augmentation is not shown.

3.3.1 Architecture

Layer	Filter	Stride	Pad	Dilation	Volume	Parameters
INPUT	-	-	-	-	$52 \times 52 \times 1$	-
CONV -> RELU	5×5	2	2	1	$26 \times 26 \times 32$	832
CONV -> RELU	3×3	1	1	1	$26 \times 26 \times 32$	9 248
CONV -> RELU	3×3	2	1	1	$13 \times 13 \times 64$	18 496
CONV -> RELU	3×3	1	1	1	$13 \times 13 \times 64$	36 928
CONV -> RELU	3×3	1	2	2	$13 \times 13 \times 96$	55 392
CONV -> RELU	3×3	1	2	2	$13 \times 13 \times 96$	83 040
CONV	5×5	1	6	3	$13 \times 13 \times 1$	2 401
BILINEAR (x4)	-	-	-	-	$52 \times 52 \times 1$	-

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

Each image is whitened (zero-mean centered and divided by its standard deviation) individually before being input. The first convolutional layer reduces the spatial dimensions

of the input from 52×52 to 26×26 to reduce the number of parameters and memory requirements while augmenting its receptive field. Subsequent convolutional layers preserve the dimensions of its input volume relegating subsampling to pooling layers. Finally, the volume is upsampled by a factor of 4 to recover the dimensions of the original image (Sec. 2.4). The network outputs a heatmap of logits with the same size as the input image.

The effective receptive field of the network is 101×101 pixels that equates to 2.1×2.1 cms. This architecture uses 206 337 parameters.

3.3.2 Regularization

We use l2-norm regularization with λ selected using a validation set as explained in Sec. 3.6.4.

3.3.3 Loss function

We compute the logistic loss function for each pixel in the produced segmentation and average the loss over pixels in the breast area—background is ignored. This amounts to using a weighted loss function where errors in the breast tissue and masses are weighted by one and those in the background are weighted by zero.

3.3.4 Experiments

We performed three experiments using this architecture:

Experiment 1.1

To obtain a performance baseline, we trained this simple network in minimally processed images, i.e., mammograms without any enhancement.

Experiment 1.2

To fight the class imbalance produced by breast mass pixels being rare in comparison to normal breast tissue pixels, we modify the loss function by weighting the losses over masses by 15, those over normal breast tissue by 1 and ignoring those over the background. Assigning a higher value to errors on breast masses forces the network to invest more resources in correctly classifying masses, thus learning better features to avoid this costly errors [51]. As in Experiment 1.1, we use input without enhancement

Experiment 1.3

In the final experiment, we combine both a weighted loss function and enhanced input images.

3.4 Experiment 2

We build a complex architecture to test whether we can benefit from the added flexibility.

3.4.1 Architecture

We model our architecture on a VGG network [56], winner of the 2014 ImageNet competition (Tab. 3.3).

Layer	Filter	Stride	Pad	Volume	Parameters
INPUT	-	-	-	$112 \times 112 \times 1$	-
CONV -> Leaky RELU	6×6	2	2	$56 \times 56 \times 56$	2 072
CONV -> Leaky RELU	3×3	1	1	$56 \times 56 \times 56$	28 280
MAXPOOL	2×2	2	0	$28 \times 28 \times 56$	-
CONV -> Leaky RELU	3×3	1	1	$28 \times 28 \times 84$	42 420
CONV -> Leaky RELU	3×3	1	1	$28 \times 28 \times 84$	63 588
MAXPOOL	2×2	2	0	$14 \times 14 \times 84$	-
CONV -> Leaky RELU	3×3	1	1	$14 \times 14 \times 112$	84 784
CONV -> Leaky RELU	3×3	1	1	$14 \times 14 \times 112$	113 008
CONV -> Leaky RELU	3×3	1	1	$14 \times 14 \times 112$	113 008
MAXPOOL	2×2	2	0	$7 \times 7 \times 112$	-
FC -> Leaky RELU	7×7	1	3	$7 \times 7 \times 448$	2 459 072
FC	1×1	1	0	$7 \times 7 \times 1$	449
BILINEAR (x16)	-	-	-	$112 \times 112 \times 1$	-

Table 3.3: Architecture of the network used for experiments. It shows the filter size, stride, padding, resulting volume and number of learnable parameters per layer.

This network has an effective receptive field of 184×184 pixels (2.9×2.9 cms) and defines 2 906 681 parameters.

3.4.2 Regularization

We use dropout after every leaky ReLU layer with probability 0.9, 0.9, 0.8, 0.8, 0.7, 0.7, 0.7 and 0.6, respectively. We also use l2-norm regularization with λ selected using a validation set.

3.4.3 Loss function

As explained in Sec 3.3.4, we use a weighted logistic loss function with errors over breast mass pixels weighted by 15, over normal breast tissue by 1 and over the background by 0.

3.4.4 Experiments

We train a single network using input images with no enhancement. Results from this experiment are directly comparable to those in Experiment 1.2 which has the same configuration.

3.5 Experiment 3

To investigate whether negative results are the product of a poorly chosen architecture we designed a different network that is deeper but has fewer parameters than the one used in Experiment 2.

3.5.1 Architecture

We model the architecture on the Residual network [24], winner of the 2015 ImageNet competition (Tab 3.4). This 10-layer architecture uses 894 017 parameters and has a receptive field

Layer	Filter	Stride	Pad	Dilation	Volume	Parameters
INPUT	-	-	-	-	$116 \times 116 \times 1$	-
CONV -> LRELU	6×6	2	2	1	$58 \times 58 \times 32$	1 184
CONV -> LRELU	3×3	1	1	1	$58 \times 58 \times 32$	9 248
CONV -> LRELU	3×3	2	1	1	$29 \times 29 \times 64$	18 496
CONV -> LRELU	3×3	1	1	1	$29 \times 29 \times 64$	36 928
CONV -> LRELU	3×3	1	2	2	$29 \times 29 \times 128$	73 856
CONV -> LRELU	3×3	1	2	2	$29 \times 29 \times 128$	147 584
CONV -> LRELU	3×3	1	2	2	$29 \times 29 \times 128$	147 584
CONV -> LRELU	3×3	1	2	2	$29 \times 29 \times 128$	147 584
CONV -> LRELU	3×3	1	4	4	$29 \times 29 \times 256$	295 168
CONV	8×8	1	14	4	$29 \times 29 \times 1$	16 385
BILINEAR (x4)	-	-	-	-	$116 \times 116 \times 1$	-

Table 3.4: Architecture of the network used for experiments. It shows the filter size, stride, dilation and padding in each layer as well as the resulting volume and number of learnable parameters per layer. LRELU stands for leaky ReLU.

size of 228×228 , equivalent to a 3.6×3.6 cm area.

3.5.2 Regularization

We use l2-norm regularization (λ chosen using a validation set) and dropout after RELU layers with probabilities 0.9, 0.9, 0.8, 0.8, 0.7, 0.7, 0.7, 0.7 and 0.6.

3.5.3 Loss function

We use a weighted logistic loss function with errors over breast masses weighted by 0.9, errors over normal breast tissue weighted by 0.1 and errors over background weighted by 0.

3.5.4 Experiments

We train a single network in preprocessed and enhanced mammograms. Results from this experiment can be compared to those in Experiment 1.3.

3.6 Training

We offer details about the learning stage of our experiments.

3.6.1 Hardware

We performed our experiments in 30 machines with the following specifications:

Location	GPU	CPU	HD	RAM
A4-401	Nvidia Quadro K620 384 cores, 2GB, compute capability 5.0	i5-4570 3.2GHz	240 GB	8 GB

Table 3.5: Available hardware for experiments

Each computer was used independently to train networks (not distributed). Although we designed our models to be small enough to fit in the GPU memory, big images surpassed this storage limit causing errors; thus, we decided to train our networks only with the CPU. Training times ranged from 1 to 1.5 hours per 1000 examples.

3.6.2 Software

Models were implemented and trained using TensorFlow (v.11) [1]. Tools for image retrieval, augmentation, evaluation and similar tasks were implemented in Python3. Code, as well as trained models, are freely accessible at: github.com/ecobost/cnn4brca.

3.6.3 Initialization

Weights for the incoming connections to a unit are drawn from a normal distribution with zero mean and $\sqrt{2/n_{in}}$ standard deviation where n_{in} is the number of connections. Biases are initialized to zero.

3.6.4 Hyperparameter search

We fit the learning rate and regularization parameter simultaneously using random sampling. For each fold independently, we train 20 networks for five epochs (6 520 examples) and select the best (α, λ) combination using 20% of the fold as a validation set. We use sensitivity at one false positive per image as performance metric.

3.6.5 Optimization

We use ADAM ($\beta_1 = 0.9$, $\beta_2 = 0.995$ and $\epsilon = 10^{-6}$) for optimization. Each parameter update uses the information from only a single training example but given that the loss is a weighted average over all pixels in the image, gradients are as rich as if we were performing mini-batch gradient descent with a batch composed of all image patches for which the network produced a prediction.

We trained the final model for all experiments for 30 epochs (49 200 examples). No early stopping was performed.

3.7 Evaluation

We describe how we evaluate our results.

3.7.1 Post-processing

We aim to evaluate convolutional networks as a single end-to-end segmentation model; thus, we choose to use the produced heatmaps without any post-processing. However, adding post-processing to our best performing architecture will certainly improve results and remains as a viable future endeavor.

3.7.2 Segmentation

We generate a segmentation by setting each pixel whose value was zero in the original mammogram to background (0), each non-background pixel whose logit is greater than a threshold to breast mass (255) and any remaining pixel to normal breast tissue (127) (Fig. 3.3).

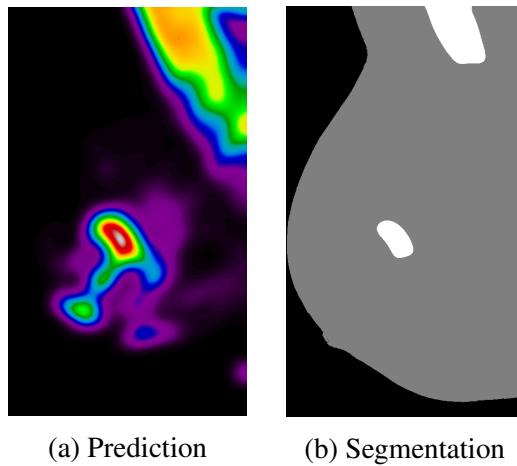


Figure 3.3: Heatmap of probabilities and segmentation produced by assigning all background to black and thresholding at probability 0.5.

3.7.3 Metrics

We use five-fold cross-validation and the free-response ROC curve to evaluate our models. We count a breast mass as a true positive if a blob in the segmentation covers at least 10% of its area. To avoid any bias, we compute the number of false positives per image only in the images without a breast mass [14]. Furthermore, we force the number of false positives in an image to be non-decreasing for laxer, i.e., lower thresholds.

3.8 Summary

Mammograms from the BCDR-D01 database were enhanced, resized and divided to obtain our data set. A simple architecture (6 layers, 206K parameters) was used for the first experiments, one of which used a weighted loss function to tackle class imbalance. Two more sophisticated architectures based on the VGG network (9 layers, 2.9M parameters) and residual networks (10 layers, 0.9M parameters) were used for the following experiments. We performed hyperparameter search to fine tune the learning rate and regularization parameter of each network; other hyperparameters were set manually. Networks were written in TensorFlow and optimized using ADAM. We use five-fold cross-validation for evaluation. We report the FROC curve for the final models.

Chapter 4

Results

We show the results of the different experiments and discuss their implications.

4.1 Hyperparameter Search

We list the selected learning rate and regularization parameter for the five folds of each experiment (Tab 4.1). Although hyperparameters were selected independently for every fold, in the same experiment they converge to similar values.

	Fold 1		Fold 2		Fold 3		Fold 4		Fold 5	
	α	λ								
Exp. 1.1	2×10^{-4}	3×10^{-2}	2×10^{-4}	4×10^{-2}	9×10^{-5}	5×10^{-3}	6×10^{-4}	8×10^{-3}	6×10^{-4}	1×10^{-2}
Exp. 1.2	4×10^{-4}	9×10^{-2}	4×10^{-4}	5×10^{-2}	4×10^{-4}	7×10^{-3}	3×10^{-4}	5×10^{-3}	6×10^{-4}	3×10^{-2}
Exp. 1.3	4×10^{-4}	3×10^{-2}	2×10^{-4}	4×10^{-2}	7×10^{-5}	6×10^{-3}	4×10^{-4}	8×10^{-2}	8×10^{-4}	3×10^{-2}
Exp. 2	1×10^{-5}	4×10^{-4}	5×10^{-5}	4×10^{-3}	5×10^{-5}	1×10^{-3}	1×10^{-5}	2×10^{-4}	5×10^{-5}	2×10^{-4}
Exp. 3	1×10^{-4}	4×10^{-4}	7×10^{-5}	5×10^{-4}	1×10^{-4}	7×10^{-4}	7×10^{-5}	2×10^{-4}	8×10^{-5}	9×10^{-5}

Table 4.1: Selected learning rate and regularization parameter configurations.

4.2 Experiments

Models perform similarly while committing less than four false positives per image with experiment 1.1 (in red), 1.3 (in green) and 2 (in blue) slightly outperforming the others (Fig. 4.1). After that mark, Experiment 1.1 and 1.3 converge faster to a sensitivity of 1 and experiment 2 follows them with a slightly slower convergence. Performance varies significantly among different folds of the same experiment as noticed by the spread of the background curves. Comparing experiment 1.1, 1.2 and 1.3, we also notice that for the same network architecture using a weighted loss function either with or without enhancing mammograms worsens the performance.

We analyze the sensitivity at one false positive per image (Tab 4.2) to have a single quantitative evaluation of the predictions made by each model. We choose a low number of

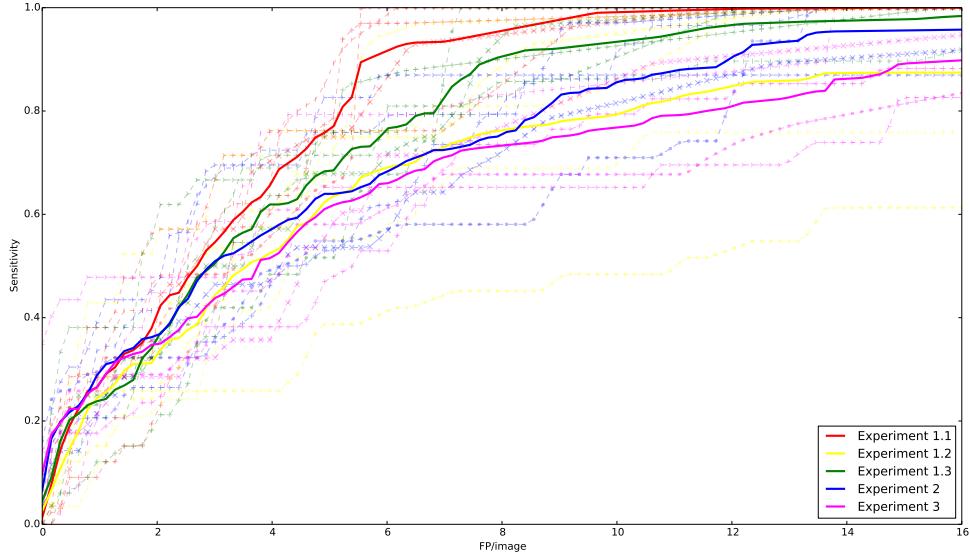


Figure 4.1: FROC curves for every experiment and fold. Each experiment appears in a different color. The solid line is the average FROC curve per experiment over all folds. FROC curves for each fold are drawn in dashed lines; each fold has a unique marker.

false positives to analyze the performance when models are committing few errors. We notice that performance is similar among different models but that experiment 2 outperforms simpler models. It is also clear that performance is also dependent on the training and test set used shown by the different average performance in each fold.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Experiment 1.1	0.09	0.29	0.28	0.29	0.38	0.26
Experiment 1.2	0.15	0.28	0.13	0.26	0.43	0.25
Experiment 1.3	0.12	0.25	0.21	0.24	0.38	0.24
Experiment 2	0.21	0.21	0.35	0.30	0.41	0.29
Experiment 3	0.18	0.21	0.22	0.26	0.48	0.27
Average	0.15	0.25	0.23	0.27	0.42	

Table 4.2: Sensitivity at 1 FP per image. Best model in each column is shown in bold.

Finally, to get a sense of the qualitative value of our results, we examined the output produced by each network in some chosen examples (Appendix A). Networks used for experiment 2 and 3 seem to outperform those from the first experiment, they perform well when the breast mass is easier to see in the mammogram and are less prone to signal chest muscle as a possible mass. The network from experiment 2 was specially impressive in signaling some of the more subtler lesions, while none of the other networks excelled at this task.

4.3 Discussion

We found that convolutional neural networks are a viable option for breast cancer lesion segmentation.

More research is needed to confirm some of the insights from this work.

Chapter 5

Conclusions

We found that...

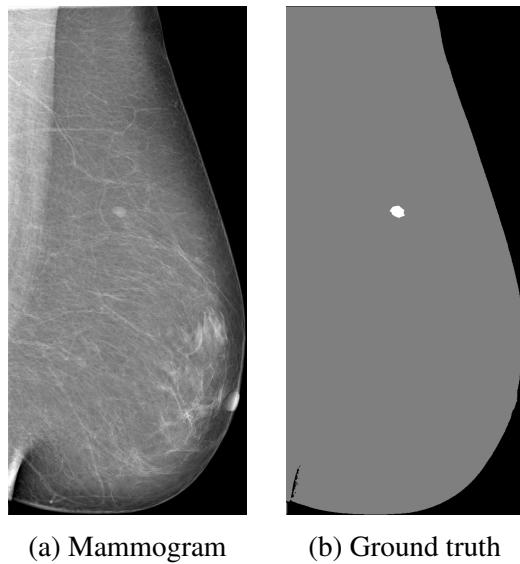
5.1 Future Work

Appendix A

Examples

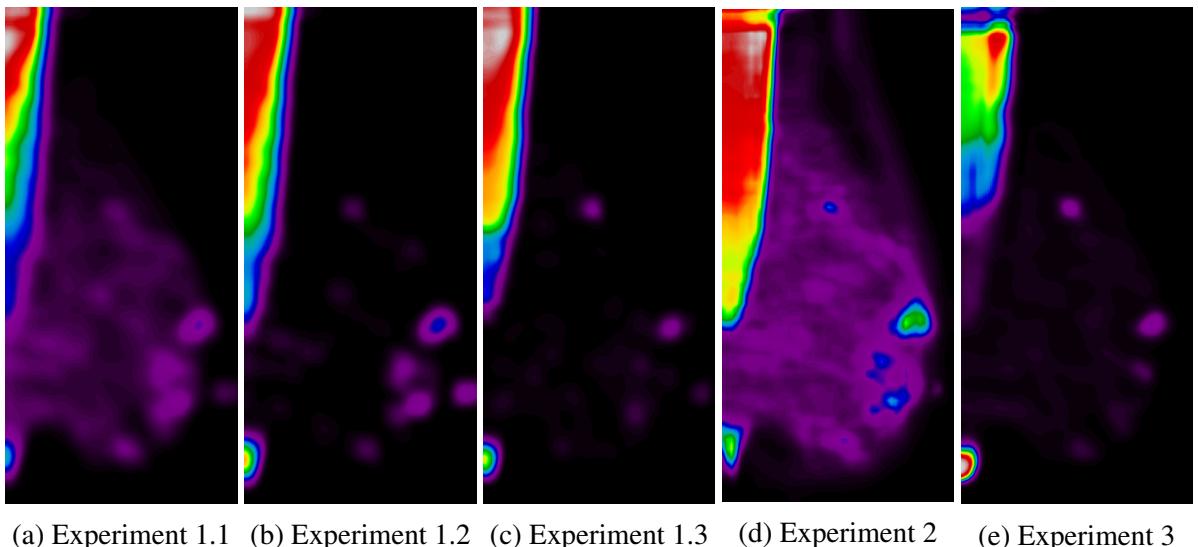
We select five different mammograms and show the predictions made by our different models (next page).

A.1 Example 1



(a) Mammogram (b) Ground truth

Figure A.1: Example 1



(a) Experiment 1.1 (b) Experiment 1.2 (c) Experiment 1.3 (d) Experiment 2 (e) Experiment 3

Figure A.2: Predicted probabilities for example 1.

A.2 Example 2

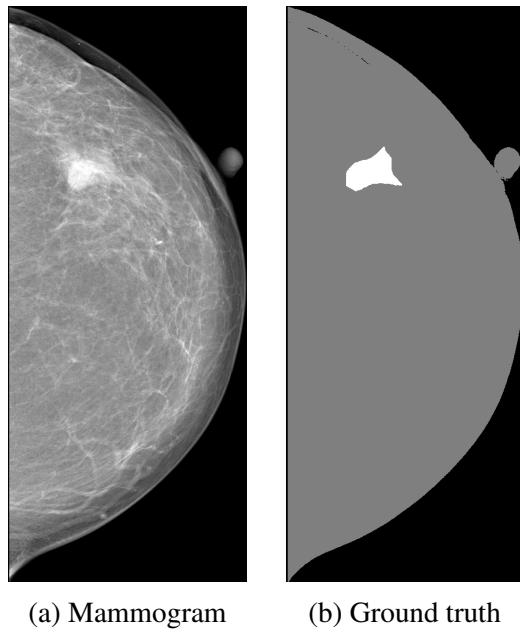


Figure A.3: Example 2

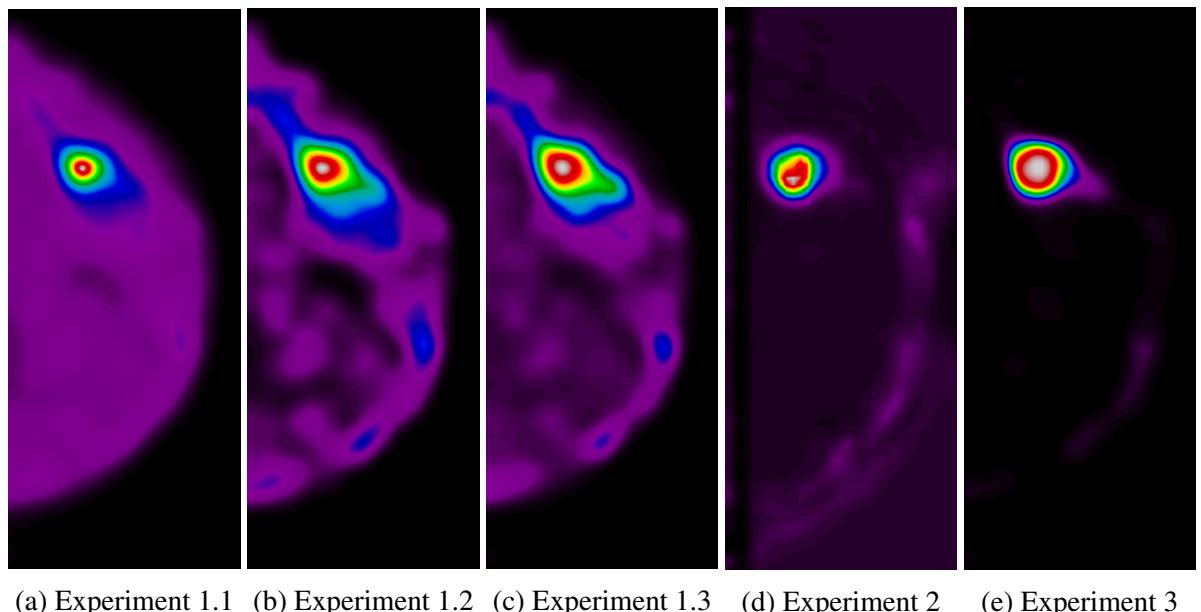


Figure A.4: Predicted probabilities for example 2.

A.3 Example 3

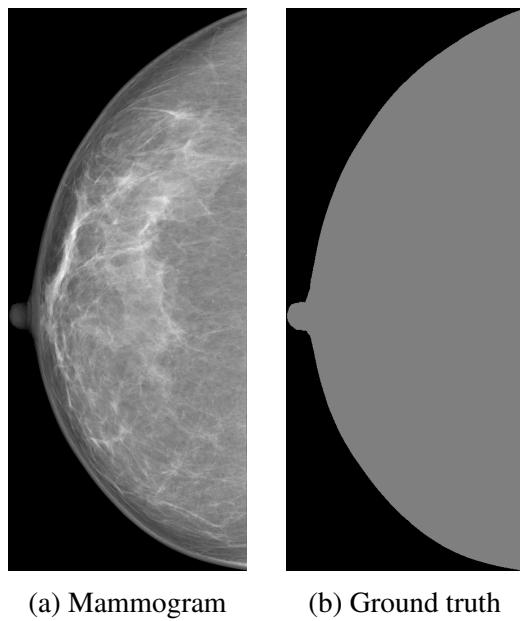


Figure A.5: Example 3

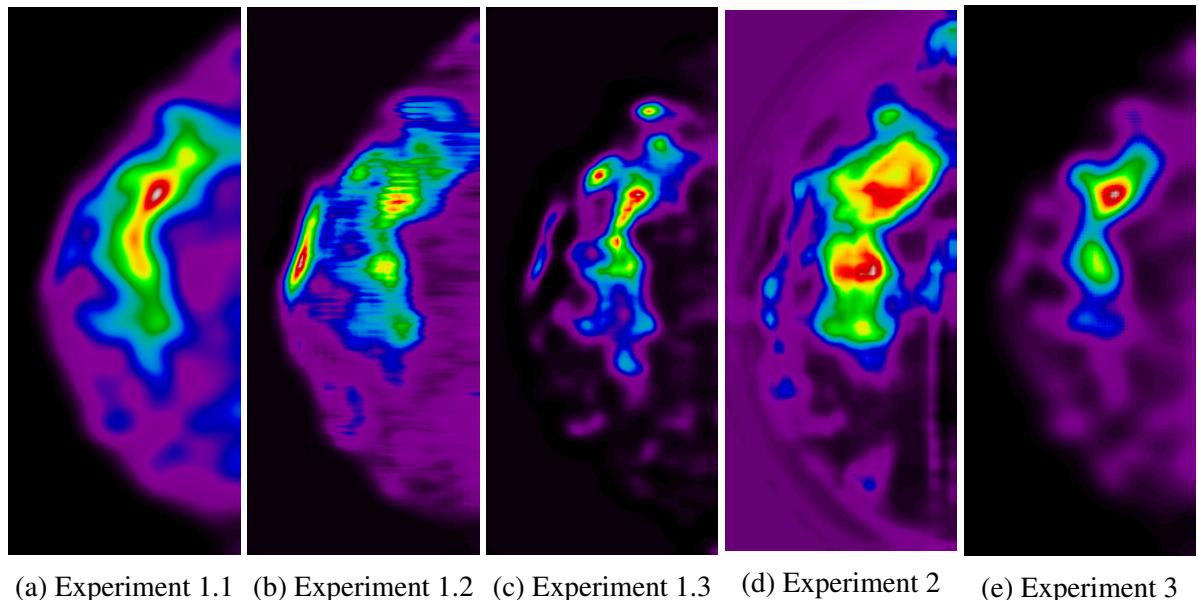


Figure A.6: Predicted probabilities for example 3.

A.4 Example 4

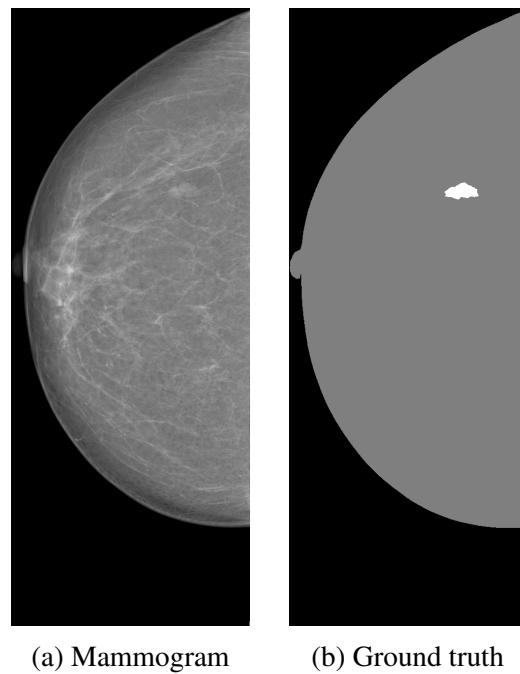


Figure A.7: Example 4

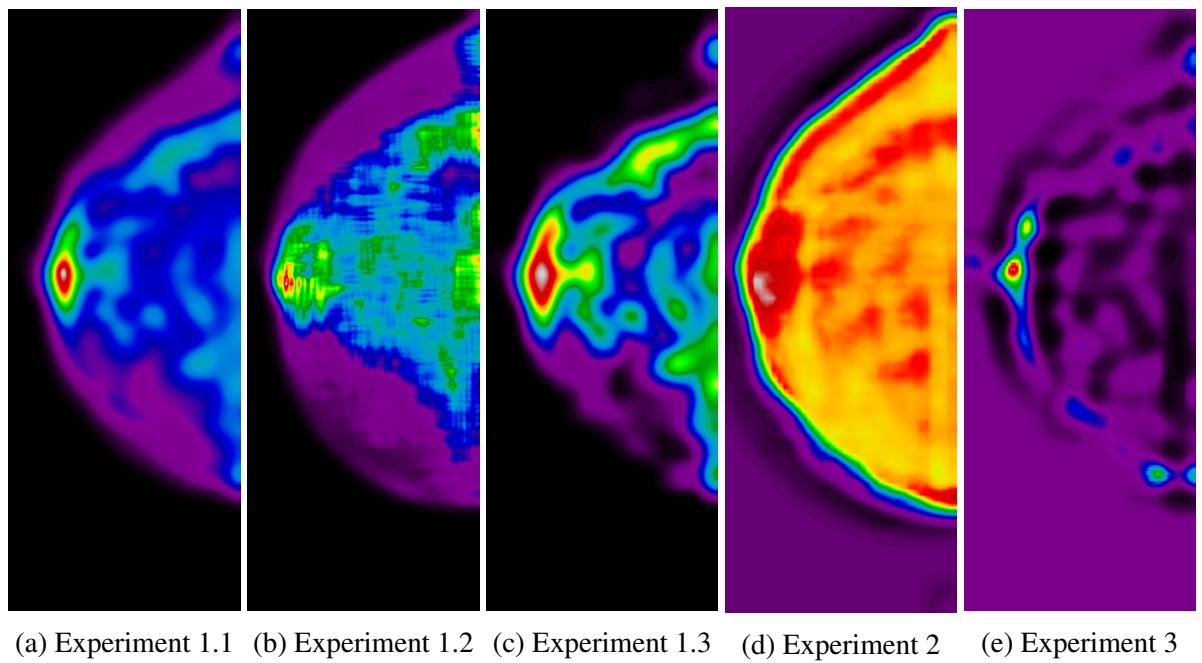


Figure A.8: Predicted probabilities for example 4.

A.5 Example 5

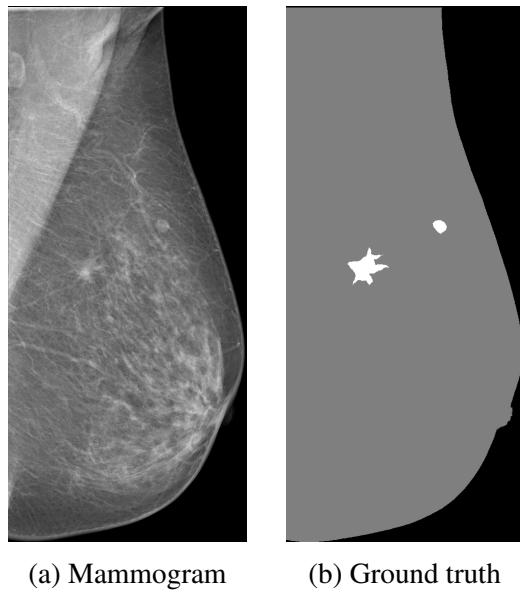


Figure A.9: Example 5

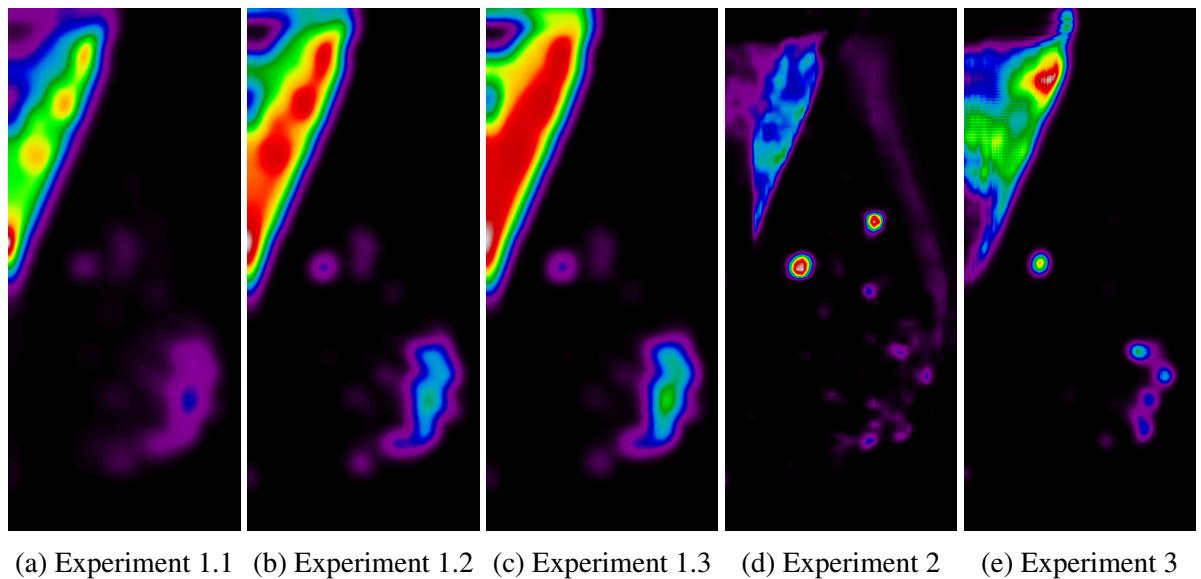


Figure A.10: Predicted probabilities for example 5.

Bibliography

- [1] ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G. S., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., GOODFELLOW, I., HARP, A., IRVING, G., ISARD, M., JIA, Y., JOZEFOWICZ, R., KAISER, L., KUDLUR, M., LEVENBERG, J., MANÉ, D., MONGA, R., MOORE, S., MURRAY, D., OLAH, C., SCHUSTER, M., SHLENS, J., STEINER, B., SUTSKEVER, I., TALWAR, K., TUCKER, P., VANHOUCKE, V., VASUDEVAN, V., VIÉGAS, F., VINYALS, O., WARDEN, P., WATTENBERG, M., WICKE, M., YU, Y., AND ZHENG, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [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 online on cancer.org/acs/groups/content/@editorial/documents/document/acspc-044552.pdf.
- [4] AREVALO, J., GONZÁLEZ, F. A., RAMOS-POLLÁN, R., OLIVEIRA, J. L., AND GUEVARA LOPEZ, M. A. Representation learning for mammography mass lesion classification with convolutional neural networks. *Computer Methods and Programs in Biomedicine* (2016).
- [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). Available online on <http://arxiv.org/abs/1206.5533>.

- [8] BENGIO, Y. Deep learning workshop. online, April 2014. Available on youtu.be/JuimBuvEWBg.
- [9] BERGSTRA, J., AND BENGIO, Y. Random search for hyper-parameter optimization. *J. Machine Learning Research* 13, 1 (February 2012), 281–305.
- [10] 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.
- [11] BISHOP, C. M. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.
- [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 online 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] CHAKRABORTY, D. P. A brief history of free-response receiver operating characteristic paradigm data analysis. *Academic Radiology* 20, 7 (2013), 915 – 919.
- [15] CHEN, L., PAPANDREOU, G., KOKKINOS, I., MURPHY, K., AND YUILLE, A. L. Semantic image segmentation with deep convolutional nets and fully connected CRFs. *CoRR abs/1412.7062* (2015). Available online on arxiv.org/abs/1412.7062.
- [16] COLLOBERT, R., KAVUKCUOGLU, K., AND FARABET, C. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop* (2011).
- [17] 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.
- [18] 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). Available online on arxiv.org/abs/1503.07077.
- [19] DUBROVINA, A., KISILEV, P., GINSBURG, B., AND HASHOUL, S. Computational mammography using deep neural networks. In *MICCAI-2015 Proc. Deep Learning in Medical Image Analisys (DLMIA)* (2015).
- [20] 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.

- [21] 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.
- [22] 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.
- [23] 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.
- [24] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. *CoRR abs/1512.03385* (2015). Available online on arxiv.org/abs/1512.03385.
- [25] 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). Available online on <http://arxiv.org/abs/1502.01852>.
- [26] 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.
- [27] 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., MARLOTTO, 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 online on seer.cancer.gov/csr/1975_2011/.
- [28] IOFFE, S., AND SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR abs/1502.03167* (2015). Available online on arxiv.org/abs/1502.03167.
- [29] 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).
- [30] KARPATY, A. Convolutional neural networks for visual recognition course. online, February 2016. Available on cs231n.github.io.
- [31] 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.

- [32] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014). Available online on arxiv.org/abs/1412.6980.
- [33] 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.
- [34] 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.
- [35] LEHMAN, C. D., WELLMAN, R. D., BUIST, D. S., KERLIKOWSKE, K., TOSTESON, A. N., AND MIGLIORETTI, D. L. Diagnostic accuracy of digital screening mammography with and without computer-aided detection. *JAMA Internal Medicine* 175, 11 (2015), 1828–1837.
- [36] 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.
- [37] 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.
- [38] 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.
- [39] LONG, J., SHELHAMER, E., AND DARRELL, T. Fully convolutional networks for semantic segmentation. *CVPR* (November 2015). Available on arxiv.org/abs/1411.4038.
- [40] McCULLOCH, W. S., AND PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5 (1943), 115–133.
- [41] MOURA, D. C., AND GUEVARA LÓPEZ, M. A. An evaluation of image descriptors combined with clinical data for breast cancer diagnosis. *International Journal of Computer Assisted Radiology and Surgery* 8, 4 (2013), 561–574.
- [42] MOURA, D. C., LÓPEZ GUEVARA, M. A., CUNHA, P., GONZÁLEZ DE POSADA, N., RAMOS-POLLAN, R., RAMOS, I., PINHEIRO LOUREIRO, J., MOREIRA, I. C., FERREIRA ARAÚJO, B. M., AND CARDOSO FERNANDES, T. Benchmarking datasets for breast cancer computer-aided diagnosis (CADx). In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 18th Iberoamerican Congress, CIARP 2013* (Berlin, Heidelberg, November 2013), J. Ruiz-Shulcloper and G. Sanniti di Baja, Eds., Springer Berlin Heidelberg, pp. 326–333.

- [43] 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.
- [44] NATIONAL CANCER INSTITUTE. Mammograms. electronic, March 2014. Available online on cancer.gov/cancertopics/types/breast/mammograms-fact-sheet.
- [45] 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.
- [46] NG, A. Machine learning course. online, December 2014. Available online on [coursera.org/course/ml](https://www.coursera.org/course/ml).
- [47] 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. *Journal of the American Medical Association* 314, 15 (2015), 1599–1614.
- [48] 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.
- [49] PETERSEN, K., NIELSEN, M., DIAO, P., KARSSEMEIJER, N., AND LILLHOLM, M. Breast tissue segmentation and mammographic risk scoring using deep learning. In *Breast Imaging: 12th International Workshop, IWDM 2014* (June 2014), H. Fujita, T. Hara, and C. Muramatsu, Eds., Springer International Publishing, pp. 88–94.
- [50] PISANO, E. D., HENDRICK, R., YAFFE, M. J., BAUM, J. K., ACHARYYA, S., COR-MACK, 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.
- [51] PROVOST, F. Machine learning from imbalanced data sets 101. *Proceedings of the AAAI-2000 Workshop on Imbalanced Data Sets* (2000).
- [52] ROSENBLATT, F. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Report (Cornell Aeronautical Laboratory). Spartan Books, 1962.
- [53] 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.

- [54] 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.
- [55] SAHINER, B., CHAN, H.-P., PETRICK, N., WEI, D., HELVIE, M. A., ADLER, D. D., AND GOODSON, 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.
- [56] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014). Available online on arxiv.org/abs/1409.1556.
- [57] 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.
- [58] SPRINGENBERG, J. T., DOSOVITSKIY, A., BROX, T., AND RIEDMILLER, M. A. Striving for simplicity: The all convolutional net. *CoRR abs/1412.6806* (2014). Available online on arxiv.org/abs/1412.6806.
- [59] 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.
- [60] SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V., AND RABINOVICH, A. Going deeper with convolutions. *CoRR abs/1409.4842* (2014). Available online on http://arxiv.org/abs/1409.4842.
- [61] TAGLIAFICO, A. S., CALABRESE, M., MARISCOTTI, G., DURANDO, M., TOSTO, S., MONETTI, F., AIRALDI, S., BIGNOTTI, B., NORI, J., BAGNI, A., SIGNORI, A., SORMANI, M. P., AND HOUSSAMI, N. Adjunct screening with tomosynthesis or ultrasound in women with mammography-negative dense breasts: Interim report of a prospective comparative trial. *Journal of Clinical Oncology* (2016).
- [62] 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).
- [63] 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.

- [64] 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.
- [65] WERBOS, P. J. *Beyond regression: new tools for prediction and analysis in the behavioral sciences.* PhD thesis, Harvard University, Boston, MA, 1974.
- [66] WIDROW, B., AND HOFF, M. E. Adaptive switching circuits. In *1960 IRE WESCON Convention Record, Part 4* (New York, 1960), IRE, pp. 96–104.
- [67] XU, B., WANG, N., CHEN, T., AND LI, M. Empirical evaluation of rectified activations in convolutional network. *CoRR abs/1505.00853* (2015). Available online on arxiv.org/abs/1505.00853.
- [68] YU, F., AND KOLTUN, V. Multi-scale context aggregation by dilated convolutions. *CoRR abs/1511.07122* (2016). Available online on arxiv.org/abs/1511.07122.

Curriculum Vitae

Erick Michael Cobos Tandazo was born in Macará, Ecuador on March 10, 1992. He earned a B.S. Computer Science and Technology 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 L^AT_EX 2_{\varepsilon}^a by Erick Michael Cobos Tandazo.

^aThe 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