FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# New Learning Strategies in Deep Models for Breast Cancer Screening

**Ricardo Dantas Cerqueira**

DISSERTATION

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Jaime S. Cardoso, Phd

Second Supervisor: José Costa Pereira, Phd

June, 2018

# New Learning Strategies in Deep Models for Breast Cancer Screening

**Ricardo Dantas Cerqueira**

Mestrado Integrado em Engenharia Informática e Computação

June, 2018

# Abstract

Breast cancer is one of the leading causes of death in women, responsible for over 508,000 deaths in 2011 alone. Early identification of the disease drastically increases odds of survival and screening tests through mammography are a crucial part of this process. Computer-aided detection and diagnosis (CAD) systems can help medical professionals analyze these screenings, increasing their accuracy and speed, allowing for more cases of the disease to be caught early.

Recent efforts in applying deep learning techniques to lesion classification and detection in mammographic images have been extremely successful, obtaining state-of-the-art results in this problem, often without the need for the domain-specific feature engineering of more traditional approaches.

However, these algorithms could be improved with larger amounts of labeled data which, for the case of mammograms, are not publicly available. In addition, public datasets are of varying quality and often captured with methods and equipment, making the transfer of machine learning models between them difficult.

The other big challenge in this task is its inherent imbalance; both in the distribution of positive and negative cases and in the disproportionately large effect of a few pixels in the overall classification. Positive examples are rare and can only be discriminated by the presence of small abnormalities in very high-resolution images. Models must then be both efficient enough to quickly process large images and sensitive enough to pick up on the minute details that completely inform the desired classification.

In addition, an exam can be composed of multiple images of the patient's breasts, which are deformed in unique and complex ways by the mammogram capture process. Experts' diagnoses of these exams take into account all of the views, often relating matching areas in the tissue in terms of symmetries or asymmetries. Computerized methods, on the other hand, have historically struggled with fusing the various views to produce unified predictions, particularly in the task of lesion detection.

We propose a one-shot, multi-task, multi-input unified model for detection of lesions and mammogram malignancy classification in one or all four views of a typical mammogram exam and validate it using the INbreast dataset. We propose two methods of relating images from the same exam: through explicit mammogram alignment, and through the merging of features in the model's architecture. Finally, we introduce a novel loss function for use in cluster detection and employ it to detecting clusters of microcalcifications.

# Contents

**1 Introduction** **1**

**2 State of the art** **5**

2.1 Introduction . . . 5
2.2 Deep Learning . . . 5
2.3 Deep Learning in Mammography . . . 12
2.4 Conclusion . . . 14

**3 Convolutional Neural Networks** **15**

3.1 Linear Models . . . 15
3.1.1 Regression . . . 15
3.1.2 Classification . . . 17
3.2 Multi Layer Perceptron . . . 20
3.2.1 Backpropagation Algorithm . . . 21
3.2.2 Activation Functions . . . 22
3.2.3 Optimization . . . 23
3.2.4 Initialization . . . 25
3.3 Convolutional Neural Networks . . . 25

**4 Methodology** **31**

4.1 Data . . . 31
4.2 Architecture . . . 31
4.2.1 Single Image Architecture . . . 31
4.2.2 Multiple Image Architecture . . . 34
4.3 Cluster Loss . . . 36
4.4 Preprocessing and Data Augmentation . . . 39
4.5 Implementation . . . 40
4.6 Metrics . . . 42
4.6.1 Training and Results . . . 44

**5 Conclusions and Future Work** **45**

**References** **47**

# List of Figures

1.1 Mammography capture process [3] . . . . . . . . . . 2
1.2 Views of a typical mammography exam . . . . . . . . . . 3
1.3 Examples of masses for each of the Birads classifications. . . . . . . . . . 3

2.1 LeNet architecture [73] . . . . . . . . . . 6
2.2 AlexNet architecture [71] . . . . . . . . . . 7
2.3 Inception Module, Szegedy et al. [117] . . . . . . . . . . 8
2.4 Residual block He et al. [43] . . . . . . . . . . 9
2.5 Examples of object detection [97] . . . . . . . . . . 10
2.6 R-CNN methodology [37] . . . . . . . . . . 10
2.7 Generative Adversarial Networks [35] . . . . . . . . . . 11
2.8 Domain-Adversarial Networks, Ganin et al. [34] . . . . . . . . . . 12

3.1 Line fitting . . . . . . . . . . 16
3.2 Gradient descent along a non-convex function surface, Huang et al. [51]. . . . . . 17
3.3 Logistic regression . . . . . . . . . . 20
3.4 Convolutional Neural Network, LeCun et al. [73]. . . . . . . . . . . 25
3.5 Comparison of independent weights in an MLP (left) and weight sharing in a 1D CNN with a filter size of $3 \times 3$ and stride $1 \times 1$ (right) . . . . . . . . . . 26
3.6 Convolutional layer with a $4 \times 4$ input, $3 \times 3$ filter size and stride of 1, Dumoulin and Visin [27] . . . . . . . . . . 27
3.7 Example of application of a max pooling layer with filter size and stride of $2 \times 2$. 28
3.8 Example of application of a global max pooling layer. . . . . . . . . . . 28
3.9 Transposed convolutional layer with $2 \times 2$ input, $3 \times 3$ filter size and stride of 1, Dumoulin and Visin [27] . . . . . . . . . . 29
3.10 Max pooling layer followed by its associated unpooling layer. . . . . . . . . . . 29
3.11 SegNet architecture, a fully convolutional network for semantic segmentation, Dumoulin and Visin [27] . . . . . . . . . . 29
3.12 Anchors in a one-shot detection architecture, Liu et al. [80]. . . . . . . . . . . 30
3.13 RetinaNet architecture, Lin et al. [79]. . . . . . . . . . . 30

4.1 The full object detection model proposed by Lin et al. [78], showing inputs (blue), bottom-up feature pyramid (red) and the top-down feature pyramid (green) . . . . 32
4.3 Feature pyramid of the model. . . . . . . . . . . 32
4.2 Backbone of the model. . . . . . . . . . . 33
4.4 Detection outputs of the model. . . . . . . . . . . 33
4.5 Segmentation and regression outputs of the model. . . . . . . . . . . 33
4.7 Proposed extensions to the RetinaNet architecture . . . . . . . . . . 34
4.6 Classification head of the model. . . . . . . . . . . 34

4.8 Generalization of the proposed architecture to multiple images . . . . . . . . . . . 35
4.9 Generalization of the classification submodel to multiple images . . . . . . . . . 36
4.10 Mammogram preprocessing pipeline . . . . . . . . . . . . . . . . . . . . . . . . 41
4.11 Precision-Recall curve (purple), for sequence of correct (green) and incorrect (red) detections. Corrected curve in red. Henderson and Ferrari [46]. . . . . . . . . . . 43
4.12 ROC curve . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43

# List of Tables

1.1 BI-RADS Mammographic Assessment Categories [74] . . . . . . . . . . . . . . . 1
1.2 BI-RADS breast density categories [74] . . . . . . . . . . . . . . . . . . . . . . 2

3.1 Common activation functions . . . . . . . . . . . . . . . . . . . . . . . . . . . 23

4.1 View fusion experiment results. . . . . . . . . . . . . . . . . . . . . . . . . . . 44

LIST OF TABLES

# Abbreviations

| | |
|---|---|
| BP | Backpropagation |
| NN | Neural Network |
| CNN | Convolutional Neural Network |
| MLP | Multi Layer Perceptron |
| GPU | Graphics Processing Unit |
| CADe | Computer-Aided Detection |
| CADx | Computer-Aided Diagnosis |
| MC | Micro-calcification |
| CC | Craniocaudal |
| MLO | Mediolateral |
| ROC | Receiver Operating Characteristic |
| FROC | Free Receiver Operating Characteristic |
| AUROC | Area Under the Receiver Operating Characteristic |

# Chapter 1

# Introduction

Breast cancer is one of the leading causes of death among women. The American Cancer Society estimated that, in the US, 30% of all new cancer cases in women are of breast cancer, and that 14% of all cancer-related deaths were caused by the disease [110]. These accounted for 252,710 cases and 40,610 deaths in 2017 alone. Early detection is essential in increasing the odds of survival of patients [59]. Often, this means identifying the presence of the disease before it displays any symptoms in a process known as screening. One of the main tools available to medical professionals is the mammography (MG) screening. Mammography consists in capturing X-ray images of the patient's breasts, as depicted in Figure 1.1, so as to identify any abnormalities indicative of breast cancer, manifesting as bright regions in the image. These abnormalities include microcalcifications, masses, bilateral asymmetries and architectural distortions and can be classified as either malignant or benign. It is estimated that these tests have a sensitivity of between 85 and 90% [10], but performance can be improved by increasing the number of readers, up to a total of 10 [11, 65].

| Class. | Description |
|---:|---|
| 0 | Incomplete |
| 1 | Negative |
| 2 | Benign findings |
| 3 | Probably benign |
| 4 | Suspicious abnormality |
| 5 | Highly suspicious of malignancy |
| 6 | Known biopsy with proven malignancy |

Table 1.1: BI-RADS Mammographic Assessment Categories [74]

Mammographic images can be taken from multiple views, but the most common are the craniocaudal (CC) and mediolateral oblique (MLO) views. Radiologists make use of both of these views, as well as comparisons between breasts and even with previous mammograms, if available, to make a decision. The result of the screening process is a breast-level Breast Imaging Reporting

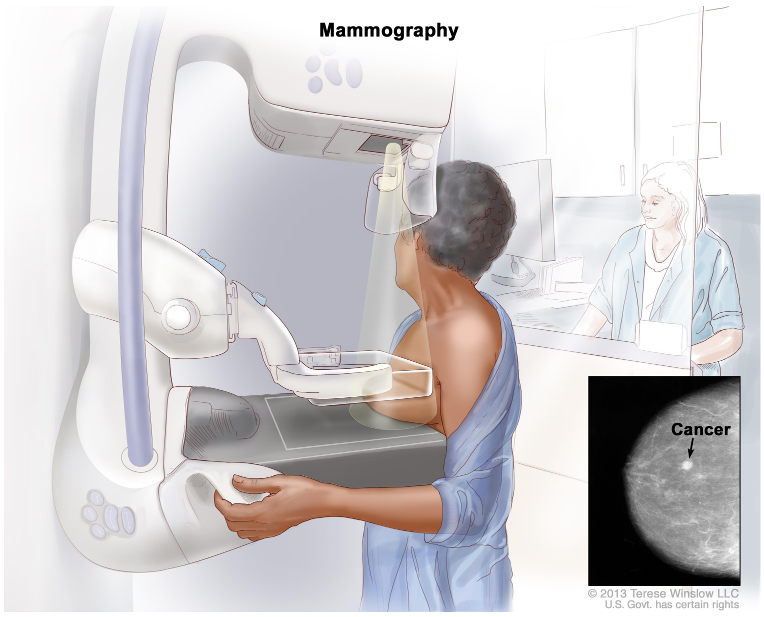


Figure 1.1: Mammography capture process [3]

| Class. | Description |
|---|---|
| 1 | Breast is almost entirely fat |
| 2 | Breast has scattered areas of fibroglandular density |
| 3 | Breast tissue is heterogeneously dense |
| 4 | Breast tissue is extremely dense |

Table 1.2: BI-RADS breast density categories [74]

and Data System (BI-RADS) classification, which indicates the level of confidence of the radiologist of the presence of benign or malignant lesions. Suspicious findings require the use of more conclusive, albeit invasive, tests like biopsies. Table 1.1 describes the meaning of each level of this classification system. Another useful factor is the density of the breast, for which BI-RADS defines 4 levels, as described in table 1.2.

The process of manually analyzing these images, however, is long, tedious and error-prone, especially when radiologists face large workloads. To alleviate this problem, various Computer Assisted Diagnosis (CADx) systems have been developed. They aim to assist professionals with an automatic analysis of the patient's data to form a second opinion. In breast cancer screening, CAD has been found to improve the performance of individual radiologists to that of double screenings [19]. By decreasing the number of false-positives, more patients can be spared the associated psychological and emotional distress, and by decreasing false-negatives the likelihood of the disease spreading unnoticed decreases as well. There is, therefore, a significant effort in both academia and the industry to develop ever more accurate CAD systems.

In recent years, the advent of Deep Learning techniques has revolutionized various fields in Computer Vision, obtaining state-of-the-art results in various difficult tasks such as object recognition, detection, and segmentation by replacing the manual feature engineering common in tra-

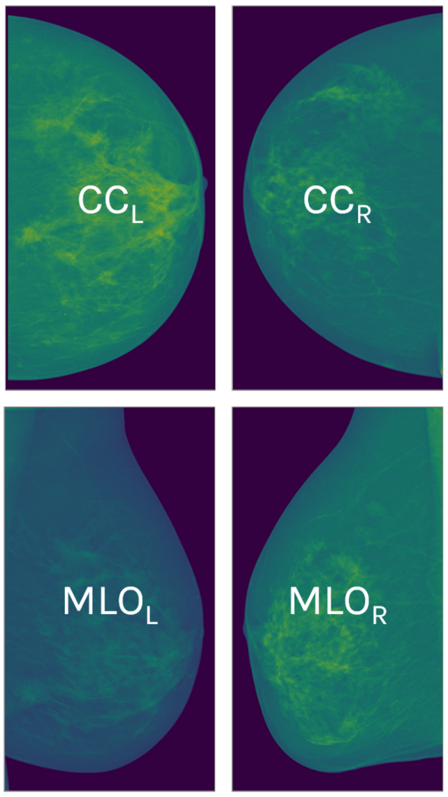


Figure 1.2: Views of a typical mammography exam

ditional machine learning methods with learned hierarchical representations. This often translates to better results with less domain knowledge built-in compared to other methods.

The interest in applying these techniques in CAD applications has grown accordingly, hoping to achieve the same level of success. Medical images, however, present a series of different challenges compared to natural ones. The most salient are: class imbalance, lack of public datasets and a disproportionate size of images compared to the size of the regions of interest. In mammography in, particular, the fact that a single image may not be enough to detect asymmetries and to monitor the evolution of abnormalities has been a significant obstacle that CAD systems have only recently shown some modest success.

The goal of this work is to develop and apply techniques that allow these models to output clinically relevant information, such as locations and types of lesions, as well as overall classifica-

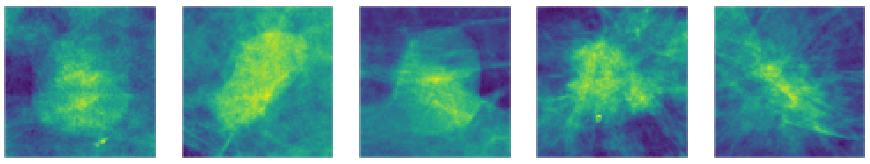


Figure 1.3: Examples of masses for each of the Birads classifications.

tions of each of the patient’s breasts by taking into account and relating together all the views in a typical mammographic exam.

# Chapter 2

# State of the art

## 2.1 Introduction

This chapter describes the most influential works in Deep Learning and its applications to mammography.

## 2.2 Deep Learning

The history of Deep Learning begins in 1943 with the introduction of the McCulloch Pitts Neuron by McCulloch and Pitts [84]. This model restricted inputs and outputs to be binary and the weights given to the inputs to be positive integers. This reduced expressive power, combined with its inability to learn, limited its applicability.

In 1958, Rosenblatt [99] proposed the Perceptron, capable of automatically learning a mapping from real-valued inputs to a binary output. However, due to its single layer, the Perceptron could only learn affine functions, which famously excludes the XOR operation, a non-linear function. In their book *Perceptrons*, Minsky and Papert [86] proved this property and speculated that networks composed of Perceptrons would be similarly limited. These criticisms are often considered to have contributed to the period of reduced funding and interest in Artificial Intelligence research that followed, and the move from connectionist to symbolic approaches.

Perhaps the most influential work in Deep Learning came in 1988 when Rumelhart et al. [101] showed how to train Multi-Layer Perceptrons (MLP) via the Backpropagation (BP) algorithm. This algorithm describes how to compute the derivatives of a loss function with respect to each of the weights of the network. These derivatives are then used to minimize a loss function by iteratively updating the values of the weights. This method is general enough that all modern Deep Learning applications still use it as a learning mechanism, despite featuring sometimes widely different architectures from the original paper.

Inspired by the visual cortex of mammals [52], Fukushima [33] proposed the Neocognitron in 1980, the first Convolutional Neural Network (CNN), a specialization of neural networks to spatial data such as images. This specialization manifested in the form of weight sharing, as the same small set of weights is applied to the image by a convolution operation, and through the use of downsampling layers, that reduce the size of activations. Unlike modern CNNs however, the Neocognitron was not trained with BP.

In 1989, LeCun et al. [72] used BP to train CNNs for the first time. Applied to the problem of Optical Character Recognition (OCR) of handwritten digits, the authors reported significantly increased performance compared to both hand-crafted weights and fully connected networks.

LeCun et al. [73] introduced the MNIST dataset, still widely used in Machine Learning research today. This dataset is split into a training and a test set of 60,000 and 10,000 images of isolated handwritten digits respectively. The authors showed that CNNs outperformed other pattern recognition models known at the time and that they were commercially viable, using the architecture depicted in Figure 2.1, now known as LeNet.

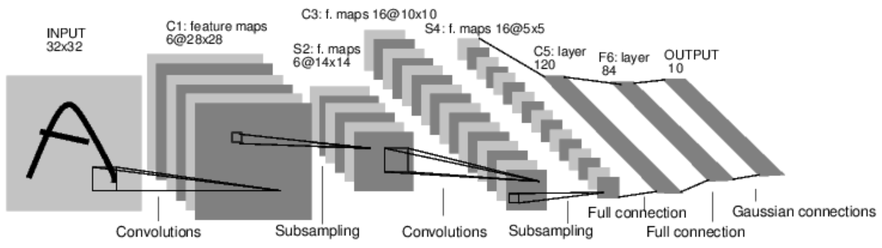


Figure 2.1: LeNet architecture [73]

Despite this, it was clear that Neural Network research was severely limited by the extremely long time models took to train. Their parallel nature meant that researchers could take advantage of the massively parallel capabilities of GPUs, which were entering the mass market in the early to mid-2000's. Steinkraus et al. [113] were the first to implement arbitrary two layer fully-connected networks on GPUs in 2005. In 2006, Chellapilla et al. [15] implemented CNNs on GPUs for the first time by unrolling convolution operations into matrix products and applied them to the problem of document processing. Cireşan et al. [20] used a GPU implementation to break records for the NORB, MNIST and CIFAR10 datasets for the first time. All authors reported significant speedups, up to 60 times.

Another limitation to research was the insufficient amount of available data. This motivated Jia Deng et al. [61] to introduce ImageNet, perhaps the most popular dataset in computer vision literature. It currently contains over 15 million images, each classified into one of thousands of classes. From 2010 to 2017, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) evaluated methods and algorithms for object detection and classification tasks submitted by competing teams on a subset of 1.2 million images of ImageNet split into 1000 classes. The massive amount of quality labeled data in this dataset was instrumental in many of the recent successes achieved in Deep Learning since it was made available. In fact, many models used in computer vision applications today are first trained with this dataset and then fine-tuned to their specific task. Many architectures were either created with the intent of winning this competition or were inspired by the ones that did.

The first influential work using ImageNet came in 2012, when Krizhevsky et al. [71] won ILSVRC-2012 with a CNN architecture now known as AlexNet (Figure 2.2), trained on two GPUs using the CUDA parallel computing platform [90]. They achieved a top-5 error rate of 15.3%, down from the 26% of the winning entry of the previous year, a 38.5% reduction in error rate and more than 10.8% ahead of the second place model [102]. The use of the ReLU activation function and the multi-GPU configuration allowed them to train the network faster, whereas dropout and data augmentation increased the ability of the network to generalize to new examples. Dropout is the process of randomly setting values in the activation of a layer to zero, thereby reducing neuron co-adaptation in the same layer [112]. Data augmentation refers to random transformations applied to images at training time. These transformations include random flipping, rotations, translations and scaling and have the effect of regularizing the networks by increasing the number of training examples [120]. Both these methods are a crucial part of the training process in modern models. More specialized forms of data augmentation are still an active area of research [76]. This record-breaking victory showed the effectiveness of CNNs in large-scale visual recognition tasks and is widely credited as a major contribution to the recent growth in interest and funding in Deep Learning research and applications.

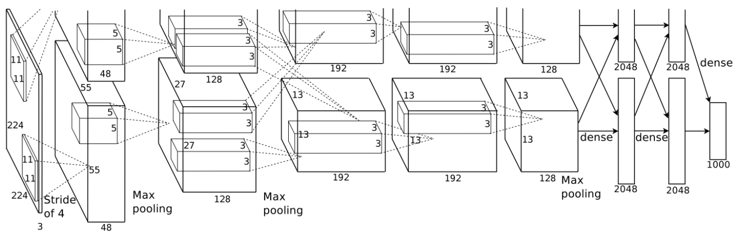


Figure 2.2: AlexNet architecture [71]

The following year, the winner of ILSVRC-2013, with a top 5 error rate of 14.8%, was only a slight modification of the AlexNet architecture known as ZFNet by Zeiler and Fergus [127]. In their paper, however, the authors proposed novel methods of visualizing the network's activations and provided key insights into its learned representations. These insights influenced the design of ZFNet, as well as many of the architectures that followed.

ILSVRC-2014 was marked by a significant increase in the depth of competing entries. VG-GNet, by Shin et al. [109], was remarkably simple compared to other existing architectures, as it was composed exclusively of $3 \times 3$ convolutions and $2 \times 2$ max pooling with a stride of 2. It was also much deeper than previous architectures, up to 19 layers against the previous winner's 8, which showed the importance of the number of layers to classification performance. Despite not winning the prestigious competition, it is still widely used, due to its simplicity and capacity, particularly in the task of artistic style transfer [62], and is implemented in most Deep Learning frameworks.

The winner in 2014 was GoogLeNet, by Szegedy et al. [117] with a top 5 error rate of 6.67%. In contrast with VGGNet, GoogLeNet had a much more complicated architecture. The most distinguishing feature was the use of the Inception module (Figure 2.3), which had the goal of processing feature maps at multiple scales efficiently. Besides its record-breaking performance, it was also the first successful model to feature parallel paths as well as multiple classification heads, design principles that would greatly influence future architectures. This architecture would go on to see various improvements in the form of the Inception family of networks (Ioffe and Szegedy [57], Szegedy et al. [119, 118], Chollet [17]).

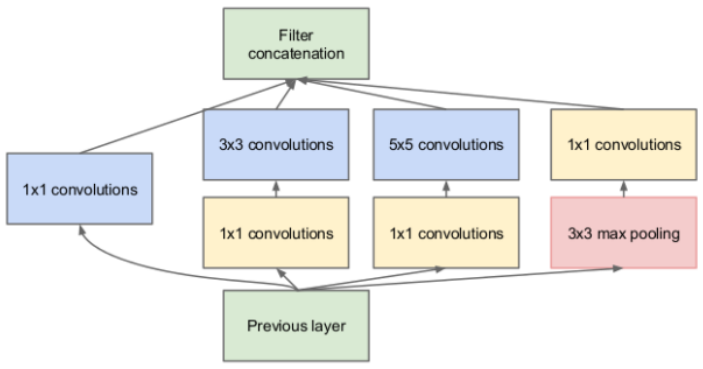


Figure 2.3: Inception Module, Szegedy et al. [117]

By this time, one of the main barriers to the use of more powerful models was the problem of internal covariate shift [108]. This is the problem of shifting distributions of the inputs to layers throughout training. Weight initialization schemes were known to decrease its effects by making layer inputs unlikely to saturate neurons while also not exclusively falling into the linear component of the activation function, throughout all the layers of the network [38, 42]. However, the distributions of these layer inputs could still shift during training. To solve this, Ioffe and Szegedy [57] proposed the Batch Normalization method. It consists of subtracting the featurewise mean and dividing by the featurewise standard deviation of the mini-batch, at the input of each layer, thereby normalizing it. Then two more learnable parameters, $\gamma$ and $\beta$ are multiplied and added to the result, allowing the network to learn to reconstruct the original input if that is more advantageous. This process allows the network to more easily control the distribution of layer inputs and therefore train faster. The performance benefits of Batch Normalization are so ubiquitous that it is now a standard part of most architectures.

Another limiting factor associated with training very deep models is the vanishing or exploding gradient problem [48, 47]. It refers to the fact gradient values tend to either increase or decrease exponentially in deeper layers of networks. Although techniques had already been developed for its mitigation, such as the use of multiple heads in GoogLeNet [117], the most successful one was the residual block (Figure 2.4), proposed by He et al. [43]. It aims to reduce the effects of the phenomenon by providing parallel paths, allowing the network to "skip" entire groups of layers. The input of a residual block is added to the output of its last layer, so the layers in between only have to learn the residual of the function represented by the whole block, that is, the difference between the output and the input values. Should the identity function be more advantageous than a group of convolutions, the block can easily learn to map its inputs to zeros. The authors proposed the ResNet family of architectures, composed of multiple such blocks, the deepest of which, ResNet-152, won the ILSVRC-2015 classification challenge. It had a top-5 error of 3.57%, and it was the first time a computer model beat human performance in this task (around 5.1% top 5 error rate [64]). Thanks to the use of these skip connections as well as the Batch Normalization technique, they were able to train a network with 152 layers, significantly more than the winner of the previous year, GoogLeNet, which only had 22. Residual Blocks have since then been used to improve several different visual recognition models [118, 17], but the applicability of the idea goes beyond image data [111].

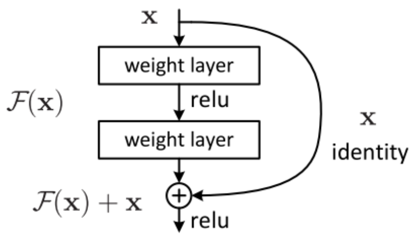

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.4: Residual block He et al. [43]

After an uneventful ILSVRC-2016, the winner of the final edition of the challenge, ILSVRC-2017 was the SENet family of architectures by Hu et al. [50], with a top 5 error rate of 2.251%. They proposed the "Squeeze-and-Excitation" blocks, which take into account channel dependencies by reweighing feature channels based on global feature information. They use global average pooling to compute feature-wise statistics. These are fed into an MLP ending with a sigmoid activation function, which outputs a vector of weights that are multiplied with each channel of features, allowing the network to dynamically focus on certain features.

The applicability of these trained models is not restricted to classifying natural images. The features learned in this problem can often be reused for related problems, in a process known as transfer learning [126]. This has a regularizing effect on the network, reducing the effective number of samples needed for a good model. Intuitively, this makes sense as many tasks, especially in vision, require solving similar subtasks, such as recognizing colors, edges, shapes, and textures.

Besides improvements in accuracy; speed and memory footprints are also important factors for many Deep Learning applications, such as in robotics and mobile computing settings. Some architectures that seek to optimize these metrics are SqueezeNet [56], MobileNet [49] and ShuffleNet [120].

The architectures mentioned so far have focused on the problem of classification. A related problem and no less challenging is that of object detection (Figure 2.5). It is an extension of classification, in that there may be any number of objects in the image and the goal is to classify and localize each one, usually in terms of bounding boxes. The closeness of these two problems is evidenced by the fact that many of the methods used for object detection use architectures originally developed for classification and often even parameters learned from classification datasets.

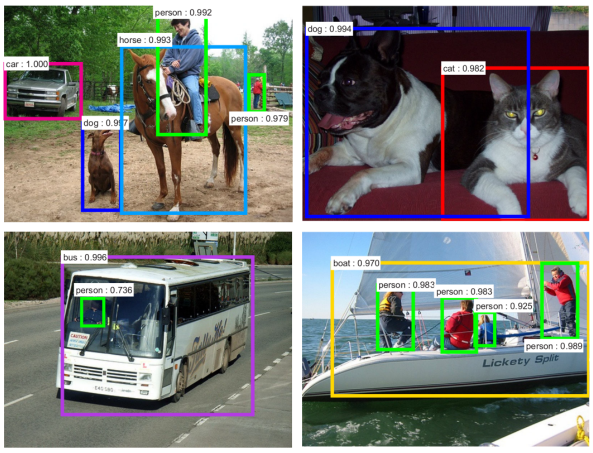


Figure 2.5: Examples of object detection [97]

The first successful application of CNNs to object detection, Overfeat, was proposed by Sermanet et al. [106]. Their method involved applying the CNN in a sliding window over the image, at various scales. At each application, the CNN is tasked with predicting both the class of any objects in the window, as well as the offset between their bounding box and the window. The final predictions are made by accumulating the intermediate ones.

In R-CNN [37] (Figure 2.6), object candidates are first extracted using a region proposal method, such as Selective Search [122], then features are computed using a CNN and finally an SVM is trained to classify the objects. This method has two major disadvantages: (1) it is hard to train, as it is composed of several different components, each of which must be tuned separately, and (2) it is slow, due to the multiple applications of the CNN structure, one for each region.

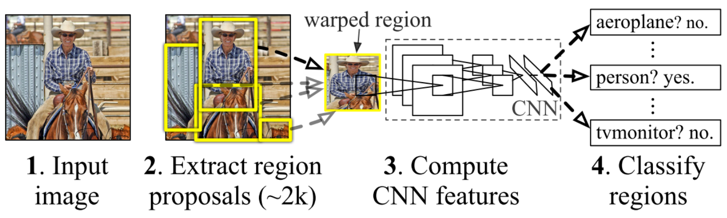


Figure 2.6: R-CNN methodology [37]

Its second iteration, Fast R-CNN [36], applied the CNN to the whole image only once to obtain a single feature map. Features for each region proposed by Selective Search are then extracted by a novel Region of Interest Pooling layer. Finally, the regions are classified with fully connected layers, rather then SVM. Despite significant speedups and an increased performance over its predecessor, Selective Search was still a limiting factor.

Faster R-CNN [97] again improved on its previous version by replacing selective search with a region proposal network that shares features with the detection network. As its name implies, it was faster than both its previous iterations and achieved state of the art results in object detection.

Its final form, Mask-RCNN [44], extended Faster R-CNN by adding instance segmentation to the output of the network. It achieved state of the art results in various tasks, including object detection.

Despite their success, Overfeat, R-CNN, and its variants were unsuitable for real-time object detection applications, reaching a maximum of 5 frames per second. Approaches based on one-stage detection try to accelerate the process by predicting both class and bounding boxes in one pass through a CNN. One-stage detectors include YOLO [96], YOLO 9000 [95], SSD [80], DSSD [32], RetinaNet [79]. The latter represents the current state-of-the-art of object detection, both in speed and accuracy, mostly due to its novel Focal loss, which aims to give more importance to difficult examples. YOLO 9000 is also noteworthy for its use of hierarchical information for bridging the gap between multiple datasets, as some include narrower classes than others. In doing so they are able to detect more than 9000 classes.

In 2014, Goodfellow et al. [39] proposed Generative Adversarial Networks (GANs), shown in Figure 2.7. These were composed of two separate networks: a generator, and a discriminator, that learn in a competitive setting. The generator is tasked with generating realistic images, whereas the discriminator must distinguish between the generated images and real ones. Through training, each network gets progressively better at their respective functions. The generator network can then be used to sample images similar to the original dataset. GANs are now a popular research topic and numerous improvements and variations have since been proposed (e.g. [9, 40]).

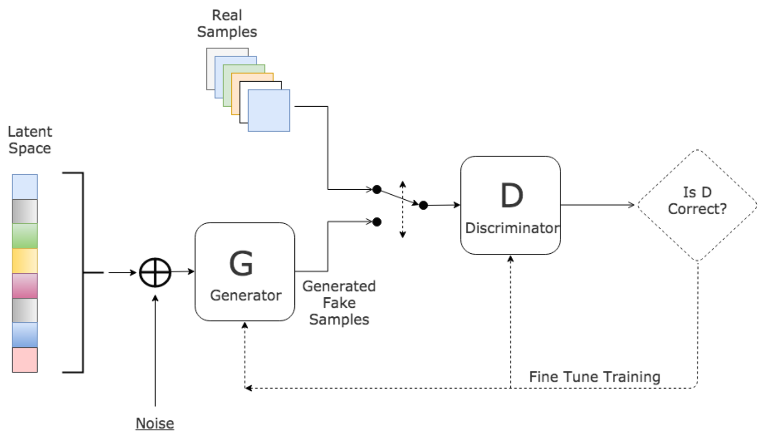


Figure 2.7: Generative Adversarial Networks [35]

Inspired by this work, Ganin et al. [34] proposed Domain-Adversarial Training (Figure 2.8). This technique replaces the competition between generator and discriminator networks with competition between a feature extractor network and a label predictor network, which cooperate to successfully classify samples, while competing against a domain classifier network, which tries to identify the domain of the sample from the features produced by the feature extractor. This forces the feature extractor to learn features that are invariant to the source domain.

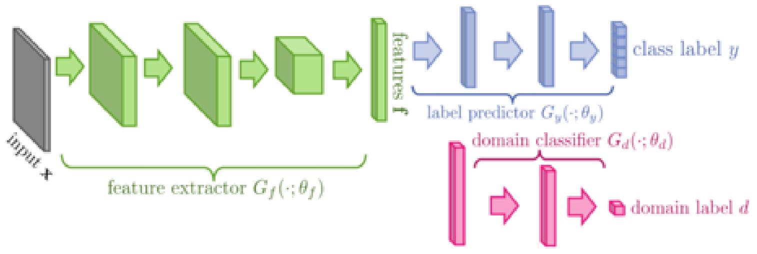

input x
feature extractor $G_f(\cdot;\theta_f)$
features f
label predictor $G_y(\cdot;\theta_y)$
class label $y$
domain classifier $G_d(\cdot;\theta_d)$
domain label $d$

Figure 2.8: Domain-Adversarial Networks, Ganin et al. [34]

## 2.3 Deep Learning in Mammography

Like all tasks in Computer Vision, research in machine learning applied to mammography images is only made possible with the existence of extensive datasets that accurately reflect the setting in which they will be applied. Despite the prevalence of private datasets in the medical imaging literature in general, there are a few publicly available mammography datasets.

The Mammographic Image Analysis Society Digital Mammogram Database (MIAS) [114] was the first public mammographic dataset. It contains 322 MLO images, from 161 cases with BI-RADS and density classifications. Abnormalities are classified with respect to their type and level of malignancy. However, their position is only indicated by the coordinates of its center and its approximate radius in pixels. In addition, the images have been resized to 1024 $\times$ 1024 pixels, reducing the amount of detail they contain.

The Digital Database for Screening Mammography (DDSM) [45] is the most commonly used dataset in the literature and it consists of 2620 cases with a total of 10,480 images. Abnormalities are annotated with their type and malignancy, BI-RADS and subtlety classifications, as well as a boundary delineation at the pixel-level. DDSM-BCRP is a subset of DDSM with 89 cases for training and 90 for testing, all of them with at least one mass or one microcalcification. The Curated Breast Imaging Subset of DDSM (CBIS-DDSM) [75] is an updated version of DDSM, with ambiguous or personally identifiable cases removed, updated annotations, suggested training and test splits, and converted to DICOM and CSV formats.

The INbreast dataset [88] consists of 410 Full-Field Digital Mammogram (FFDM) images from 115 cases obtained from the Breast Centre in Centro Hospitalar São João. Abnormalities are annotated with their respective type and pixel-level boundaries in the image. Each image is also categorized according to its ACR and BI-RADS classifications.

The Breast Cancer Digital Repository (BCDR) [81] includes 3703 CC and MLO images from 1010 cases with both pixel-level annotations of images and image-level BI-RADS classifications.

Numerous techniques applying Deep Learning to mammography images have been proposed. The first application of CNNs to mammographic images was proposed by Sahiner et al. [103] with the goal of classifying patches of images according to the presence of masses, a very common task in the literature. Huynh et al. [54] and Lévy and Jain [77] fine-tuned CNNs pre-trained on ImageNet to improve performance in this task, whereas Kooi et al. [70] used a CNN pre-trained on a similar mammography task as a feature extractor.

The inclusion of hand-crafted features is also reported to yield better results. Arevalo et al. [8], [7] proposed adding these features to the input of the network, whereas Dhungel et al. [25] proposed first training a CNN on a regression task of learning to estimate hand-crafted features, before fine-tuning on the full classification of patches into one of benign or malignant. The authors tested the system on the INbreast dataset and obtained an AUC of 0.91 using a Random Forest trained on the CNN's features.

Kooi et al. [69] showed that a CNN could obtain similar performance to professional radiologists at patch level classification. Yi et al. [125] used CNNs to classify masses and used techniques similar to DeepDream to visualize the characteristics in the image that triggered its prediction. They observed that the resulting images amplified well-known features of the corresponding lesions, such as spiculation in malignant masses.

Patch level training is also commonly used as a first step towards an image-level model. Lotter et al. [82] and Shen [107] first train a CNN on patch classification before applying it in a sliding window fashion to classify whole images from DDSM and INbreast.

Castro [14] proposed using rotated filters to reduce the number of parameters of the network and rank learning, by pairing samples and learning the differences between them, which increases the effective number of training samples on the CBIS-DDSM dataset. Qiu et al. [94] attempt to predict the risk of future cancer development in negative examples.

One of the more challenging aspects of applying deep models to this problem is taking into account multiple views and the patients' history of exams. Carneiro et al. [13] proposed an end-to-end method to classify the patient's risk of developing cancer, based on two different views of the same breast and the associated segmentation maps using a model pre-trained on ImageNet. They used the DDSM and INbreast datasets. Kooi and Karssemeijer [68] took into account symmetrical and temporal differences between different views and different exams. Salehinejad et al. [104] also included information from radiologists' reports in natural language format.

Another obstacle is that often, not all the data is labeled or only labeled at the image level. Sun et al. [115] proposed a semi-supervised learning algorithm to take advantage of additional, unlabeled data and trained it on a private dataset. Hwang and Kim [55] and Zhu et al. [128] use weakly supervised learning to detect masses training only on image level annotations. In detection, the most common approaches are cascade based, where possible regions are progressively rejected by discriminator models, [23, 83, 87], or R-CNN variants Kisilev et al. [67], Ribli et al. [98], Akselrod-Ballin et al. [6]

Another important aspect for diagnosis is the segmentation of the lesions, as their shape is highly discriminative. Cardoso et al. [12] compared the performance of graph-based models using hand-crafted features and deep learning models [24] for the task of mass segmentation when tested on a different dataset from the one it was trained on. They showed that, in addition to better performance when training and test data came from the same dataset, the deep models suffered less performance degradation when they originated from different datasets. Zhu et al. [129] proposed using adversarial networks as a regularization method for mass segmentation.

Despite their prevalence in the literature of mammogram analysis, there are other tasks of interest besides lesion detection and classification, CNNs have been used to classify the density of the breast [31, 30, 63, 5], for whole breast segmentation [26], and even to detect cardiovascular disease [124].

## 2.4 Conclusion

It is clear that the history of deep learning is one of revisiting old ideas with new modifications. The architectures used today in many successful applications are, on the surface, very similar to the ones that emerged in the 80's. Numerous algorithmic and technological improvements, combined with vast datasets allowed for deep learning models to shine in a wide variety of tasks and types of data, with relatively few alterations or domain-specific knowledge.

In mammogram analysis, deep learning provides the possibility of replacing the complex multi-stage models that represented the previous state-of-the-art with simpler ones that can be trained in an end-to-end fashion and, in many cases, with better performance. The high prevalence of private datasets in the literature makes it difficult to compare the various models that have been proposed directly. Some recent efforts in creating and curating high-quality public datasets [88, 75] may change this in the future. In addition, although some work has been done in merging data from multiple images from the same patient, fully fusing all views of an exam remains an open problem, especially in detection tasks.

# Chapter 3

# Convolutional Neural Networks

In an increasingly connected world with ever cheaper storage devices, there is demand for computational methods capable of efficiently analyzing the vast amounts of data that can be collected. In machine learning, insight is drawn from previously gathered data, known as training data, that are then generalized to new unseen circumstances. By automating knowledge extraction, new problems can be solved with less domain specific knowledge and their solutions deployed on a larger scale than what would otherwise be possible.

## 3.1 Linear Models

### 3.1.1 Regression

Perhaps the simplest example of a machine learning technique is line fitting or regression. Given $N$ labeled observations $\{x_i, y_i\}_{i=1}^{N}, \quad x_i \in \mathbb{R}^{d_1}, y_i \in \mathbb{R}^{d_2}$, the task is to find the line that best fits the set of points. We assume the one-dimensional case, $d_1 = d_2 = 1$, for illustrative purposes. Finding this line involves finding the parameters in the expression that define the line, $\hat{y} = a \cdot x + b$, that optimize some criteria. We assume that the observations have an error term, $\varepsilon$ inherent to their measurement that is independent between the observations and Gaussian in its distribution, $\varepsilon \sim N(0, \sigma^2)$. The line can then be written in the form $\hat{y} = a \cdot x + \varepsilon$ and has a distribution given by $p(y|a,b) = N(a \cdot x + b, \sigma^2)$. Because all points are independent from each other, their joint distribution can be expressed as the product of the marginal distributions,

$$p(\boldsymbol{y}|a,b) = p(y_1, y_2, ..., y_n|a,b) = \prod_{i=1}^{n} p(y_i|a,b) = \prod_{i=1}^{n} N(a \cdot x_i + b, \sigma^2)$$

This function is known as the likelihood function of $a$ and $b$, denoted as $L(a,b)$. By maximizing the likelihood function we can obtain a point estimate, that is, a single value, $a_{ML}$ and $b_{ML}$, that best fit the observations under the assumptions we have made. These are known as Maximum

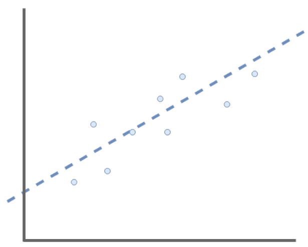


Figure 3.1: Line fitting

Likelihood Estimates (MLE). We instead minimize the negative of the log of this function as it preserves extrema and simplifies the calculations in this case:

$$
\begin{aligned}
\arg\max_{a,b}\ L(a,b) &= \arg\min_{a,b}\ -log\, L(a,b) \\
&= \arg\min_{a,b}\ -log\left(\prod_{i=1}^{n} N(\hat{y}_i,\ \sigma^2)\right) \\
&= \arg\min_{a,b}\ \sum_{i=1}^{n} -log\left(N(\hat{y}_i,\ \sigma^2)\right) \\
&= \arg\min_{a,b}\ \sum_{i=1}^{n} -log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-\hat{y}_i)^2}{2\sigma^2}}\right) \\
&= \arg\min_{a,b}\ \sum_{i=1}^{n} -log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \sum_{i=1}^{n} -log\left(e^{-\frac{(y_i-\hat{y}_i)^2}{2\sigma^2}}\right) \\
&= \arg\min_{a,b}\ \sum_{i=1}^{n} \frac{(y_i-\hat{y}_i)^2}{2\sigma^2} \\
&= \arg\min_{a,b}\ \frac{1}{2n} \sum_{i=1}^{n} (y_i-\hat{y}_i)^2
\end{aligned} \tag{3.1}
$$

The function that is to be minimized is called the loss or cost function. This particular loss function is called the Mean Squared Error (MSE) loss function.

Although we can solve this particular expression in closed form, we can also opt for numeric optimization using gradient descent. In gradient descent (3.2), an initial point in the parameter space is iteratively improved using gradient information at that point. In its simplest form, the parameters $\theta$, are updated by adding the gradients of the function, multiplied by a hyper-parameter, the learning rate, $\eta$.

$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \nabla L(\theta^{(t)}) \tag{3.2}$$

Given that the loss function is convex, it is guaranteed that this process will converge to the global optimum, the parameters defining the line that best fits the data. This method can also be extended to arbitrarily many inputs, outputs, and their associated parameters.

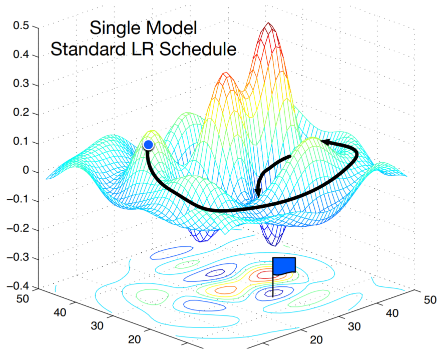


Figure 3.2: Gradient descent along a non-convex function surface, Huang et al. [51].

### 3.1.2 Classification

Another common task in machine learning is classification or logistic regression. In binary classification the goal is to classify a data point into one of 2 classes, positive or negative, given $N$ labeled observations $\{x_i, y_i\}_{i=1}^{N}, x_i \in \mathbb{R}^d, y_i \in \{0,1\}$. The posterior probability for the positive class can be written as:

$$\begin{aligned}
p(y=1|x) &= \frac{p(x|y=1)p(y=1)}{p(x|y=1)p(y=1)+p(x|y=0)p(y=0)} \\
&= \frac{\frac{p(x|y=1)p(y=1)}{p(x|y=1)p(y=1)}}{\frac{p(x|y=1)p(y=1)+p(x|y=0)p(y=0)}{p(x|y=1)p(y=1)}} \\
&= \frac{1}{1+\frac{p(x|y=0)p(y=0)}{p(x|y=1)p(y=1)}} \\
&= \frac{1}{1+e^{-log\frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)}}} \\
&= \frac{1}{1+e^{-a}} \\
&\triangleq \sigma(a)
\end{aligned} \tag{3.3}$$

Where:

$$a = log\frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)} \tag{3.4}$$

And $\sigma(a)$ is the logistic sigmoid function. Its properties are further detailed in section 3.2.2. We then assume that the data points, within their own class, are distributed normally, that is,

$P(X|Y=1) \sim N(\mu_1, \Sigma)$ and $P(X|Y=0) \sim N(\mu_0, \Sigma)$.

$$
\begin{aligned}
a &= log\frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)} \\
&= log\ p(x|y=1) - log\ p(x|y=0) + log\frac{p(y=1)}{p(y=0)} \\
&= -\frac{1}{2}\left(log|\Sigma| + klog(2\pi) + (x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right) \\
&\quad + \frac{1}{2}\left(log|\Sigma| + klog(2\pi) + (x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\right) + log\frac{p(y=1)}{p(y=0)} \\
&= -\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) \\
&\quad + \frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0) + log\frac{p(y=1)}{p(y=0)} \\
&= -\frac{1}{2}(x^T\Sigma^{-1}x - \mu_1^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu_1 + \mu_1^T\Sigma^{-1}\mu_1) \\
&\quad + \frac{1}{2}(x^T\Sigma^{-1}x - \mu_0^T\Sigma^{-1}x - x^T\Sigma^{-1}\mu_0 + \mu_0^T\Sigma^{-1}\mu_0) + log\frac{p(y=1)}{p(y=0)} \\
&= \frac{1}{2}\mu_1^T\Sigma^{-1}x + \frac{1}{2}x^T\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 \\
&\quad - \frac{1}{2}\mu_0^T\Sigma^{-1}x - \frac{1}{2}x^T\Sigma^{-1}\mu_0 + \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 + log\frac{p(y=1)}{p(y=0)} \\
&= \frac{1}{2}\mu_1^T\Sigma^{-1}x + \frac{1}{2}\left(x^T\Sigma^{-1}\mu_1\right)^T - \frac{1}{2}\mu_0^T\Sigma^{-1}x - \frac{1}{2}\left(x^T\Sigma^{-1}\mu_0\right)^T \\
&\quad + \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + log\frac{p(y=1)}{p(y=0)} \\
&= \frac{1}{2}\mu_1^T\Sigma^{-1}x + \frac{1}{2}\mu_1^T\Sigma^{-1}x - \frac{1}{2}\mu_0^T\Sigma^{-1}x - \frac{1}{2}\mu_0^T\Sigma^{-1}x \\
&\quad + \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + log\frac{p(y=1)}{p(y=0)} \\
&= \left(\mu_1^T\Sigma^{-1} - \mu_0^T\Sigma^{-1}\right)x \\
&\quad + \left(\frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + log\frac{p(y=1)}{p(y=0)}\right) \\
&= \left(\Sigma^{-1}(\mu_1-\mu_0)\right)^T x \\
&\quad + \left(\frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + log\frac{p(y=1)}{p(y=0)}\right) \\
&= w^Tx + w_0
\end{aligned}
\tag{3.5}
$$

Where $w = \Sigma^{-1}(\mu_1 - \mu_0)$ and $w_0 = \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + log\frac{p(y=1)}{p(y=0)}$. It follows that $p(y=1|x) = \sigma(w^Tx + w_0)$ and that $p(y=0|x) = 1 - \sigma(w^Tx + w_0)$.

$y$ is assumed to follow a Bernoulli distribution. Let $\theta$ denote the probability that $y_i$ is positive. The probability density function of $y$ can be written as:

$$p(y_i|\theta) = \begin{cases} \theta, & \text{if } y_i = 1 \\ 1-\theta, & \text{if } y_i = 0 \end{cases} \tag{3.6}$$

Or, alternatively:

$$p(y_i|\theta) = p(y_i = 1)^{y_i} p(y_i = 0)^{1-y_i} = \theta^{y_i}(1-\theta)^{1-y_i} \tag{3.7}$$

Because the points are independent and identically distributed (i.i.d.), their joint probability function can expressed as the product of the marginals:

$$p(\boldsymbol{y}|\theta) = \prod_{i=0}^{N} p(y_i|\theta) = \prod_{i=0}^{N} \theta^{y_i}(1-\theta)^{1-y_i} \tag{3.8}$$

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=0}^{N} p(y_i|x_i) = \prod_{i=0}^{N} p(y_i = 1|x_i)^{y_i} p(y_i = 0|x_i)^{1-y_i} \tag{3.9}$$

Again, we wish to maximize the likelihood of the labels, given the data points.

$$\begin{aligned} \max p(\boldsymbol{y}|\boldsymbol{x}) &= \min L(w, w_0) \\ &= \min -logp(\boldsymbol{y}|\boldsymbol{x}) \\ &= \min -log \prod_{i=0}^{N} p(y_i = 1|x_i)^{y_i} p(y_i = 0|x_i)^{1-y_i} \\ &= \min -\sum_{i=0}^{N} y_i \; logp(y_i = 1|x_i) + (1-y_i) logp(y_i = 0|x_i) \\ &= \min -\sum_{i=0}^{N} y_i \; log \; \sigma(w^T x_i + w_0) + (1-y_i) \; log \; \sigma(w^T x_i + w_0) \end{aligned} \tag{3.10}$$

This loss function is known as binary cross entropy. By finding the parameters $w_{ML}$ and $w_{0ML}$ that minimize it, we can model the distribution $p(y|x)$ as $\sigma(w^T x + w_0)$. This allows for the probability of a data point belonging to each of the classes to be computed. Unlike regression, however, the problem is not in general convex, in cases where points are not linearly separable, making a closed form solution impossible.

This technique can easily be extend to multiple classes. The logistic sigmoid function is replaced by the softmax function:

$$softmax(a)_j = \frac{e^{a_j}}{\sum_{j'}^{C} e^{j'}} \tag{3.11}$$

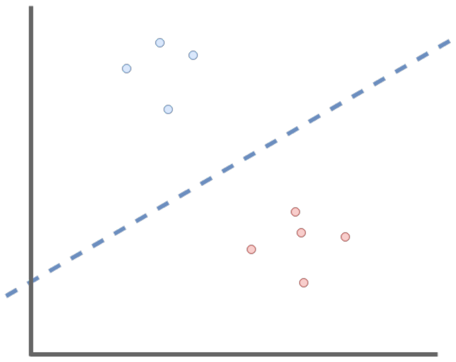


Figure 3.3: Logistic regression

And the loss function becomes the cross entropy function:

$$L(w, w_0) = -\sum_{i=0}^{N}\sum_{j=0}^{C} \mathbb{1}(y_i = j)\ log\left(softmax(w^T x_i + w_0)_j\right) \tag{3.12}$$

Where $\mathbb{1}(x = y)$, the indicator function, is defined as:

$$\mathbb{1}(x = y) = \begin{cases} 1, & \text{if } x = y \\ 0, & otherwise \end{cases} \tag{3.13}$$

## 3.2 Multi Layer Perceptron

More complex relationships can be expressed by stacking multiple linear functions as layers, interspersed with non-linear functions, known as activation functions. The output from each layer is fed into the input of the next, after which the activation function is applied. This type of model is called a Multi Layer Perceptron (MLP) and its layers are called dense or fully connected layers. For a model with $L$ layers, the output is given by:

$$\begin{aligned} a_i^{(0)} &= x_i \\ z_i^{(l)} &= W^{(l)} a_i^{(l)} + b^{(l)} \\ a_i^{(l+1)} &= f(z_i^{(l)}) \\ \hat{y}_i &= a_i^{(L)} \end{aligned} \tag{3.14}$$

Where the matrix of weights $W^{(l)}$, and the vector of biases $b^{(l)}$ extend the first and zero order coefficients in the line equation to multiple dimensions and the $a_i^l$, are the activation values at layer $l$ for point $i$. The dimensions of $a^l$ can vary from layer to layer and are referred to as the number of neurons of that layer. In real implementations, the output of a layer can be computed for an entire batch in parallel by making use of matrix multiplication.

### 3.2.1 Backpropagation Algorithm

Unlike the linear model, optimization of an MLP cannot be solved in closed form, necessitating the use of gradient information. In addition, its arbitrary number of layers also precludes a closed form expression for computing gradients in the general case due to its complexity. Its cascade nature, however, allows for an efficient recursive algorithm that computes gradients from later layers to earlier ones, known as the Backpropagation Algorithm (BP).

In BP, gradients are propagated backwards through successive applications of the chain rule:

$$[f(g(x))]' = f'(g(x))g'(x) \tag{3.15}$$

The goal is to compute the gradient of the loss with respect to each of the parameters, $W^{(l)}$ and $b^{(l)}$:

$$\begin{aligned}
\frac{\partial L}{\partial W^{(l)}} &= \sum_i^n \frac{\partial L}{\partial z_i^{(l)}} \frac{\partial z_i^{(l)}}{\partial W^{(l)}} \\
&= \sum_i^n \frac{\partial L}{\partial z_i^{(l)}} \frac{\partial}{\partial W^{(l)}} \left( W^{(l)} a_i^{(l)} + b^{(l)} \right) \\
&= \sum_i^n \frac{\partial L}{\partial z_i^{(l)}} a_i^{(l)T}
\end{aligned} \tag{3.16}$$

$$\begin{aligned}
\frac{\partial L}{\partial b^{(l)}} &= \sum_i^n \frac{\partial L}{\partial z_i^{(l)}} \frac{\partial z_i^{(l)}}{\partial b^{(l)}} \\
&= \sum_i^n \frac{\partial L}{\partial z_i^{(l)}} \frac{\partial}{\partial b^{(l)}} \left( W^{(l)} a_i^{(l)} + b^{(l)} \right) \\
&= \sum_i^n \frac{\partial L}{\partial z_i^{(l)}}
\end{aligned} \tag{3.17}$$

Where $\frac{\partial L}{\partial z_i^{(l)}}$ can be computed as follows:

$$\begin{aligned}
\frac{\partial L}{\partial z_i^{(l)}} &= \frac{\partial L}{\partial a_i^{(l+1)}} \frac{\partial a_i^{(l+1)}}{\partial z_i^{(l)}} \\
&= \frac{\partial L}{\partial a_i^{(l+1)}} \frac{\partial}{\partial z_i^{(l)}} f(z_i^{(l)}) \\
&= \frac{\partial L}{\partial a_i^{(l+1)}} f'(z_i^{(l)})
\end{aligned} \tag{3.18}$$

Where $\frac{\partial L}{\partial a_i^{(l)}}$ can be computed recursively as follows:

$$\begin{aligned}
\frac{\partial L}{\partial a_i^{(l)}} &= \frac{\partial L}{\partial a_i^{(l+1)}} \frac{\partial a_i^{(l+1)}}{\partial a_i^{(l)}} \\
&= \frac{\partial L}{\partial a_i^{(l+1)}} \frac{\partial a_i^{(l+1)}}{\partial z_i^{(l)}} \frac{\partial z_i^{(l)}}{\partial a_i^{(l)}} \\
&= \frac{\partial L}{\partial a_i^{(l+1)}} \frac{\partial}{\partial z_i^{(l)}} f(z_i^{(l)}) \frac{\partial}{\partial a_i^{(l)}} \left( W^{(l)} a_i^{(l)} + b^{(l)} \right) \\
&= \frac{\partial L}{\partial a_i^{(l+1)}} f'(z_i^{(l)}) W^{(l)T}
\end{aligned} \tag{3.19}$$

Starting with the gradients of the last layer:

$$\frac{\partial L}{\partial a_i^{(L)}} = \frac{\partial L}{\partial \hat{y}_i} = L'(\hat{y}_i) \tag{3.20}$$

The backpropagation algorithm can be summed up as follows: an input is first fed through the network, from the first layers to the later ones using equations 3.14, then the error function is computed taking into account the predicted value, the output of the last layer, and the expected value or ground truth. Finally, derivatives of the loss function are computed, using with equations 3.16, 3.17, 3.18, 3.19, from the later layers to the initial ones. Like the feedforward step, gradients of layers can be computed efficiently for an entire batch of inputs by multiplying the transpose of the weight matrix. To apply this algorithm, both the activation functions and the loss function must be differentiable with respect to their inputs and the expressions of the derivatives known.

### 3.2.2 Activation Functions

The exact form of the activation function can vary from model to model, and even within the same model. Historically, the most used activation functions in early models were the sigmoid and tanh functions. Although mathematically similar, the tanh is typically better for learning due to its output being centered at 0, but both suffer from a saturation problem, as the gradients go to zero with a high absolute value, stopping the learning process altogether.

More recently, many activation functions have been proposed to deal with these issues. The simplest and most popular of these is the Rectified Linear Unit (ReLU), which sets the output to zero if it is negative and keeps it otherwise and thus has the advantages of both being faster to compute and not saturating at high values. Its gradients can still go to zero if all its inputs are negative. ReLUs that always receive negative inputs and so always output 0 are called dead ReLUs. Leaky ReLUs solve this problem by instead multiplying negative values by a small constant. Other variants replace the constant with learnable parameters or randomize its value. Though less common in feed forward models, the sigmoid and tanh functions are still the standard in recurrent models due to their finite ranges. These functions are defined in table 3.2.2.

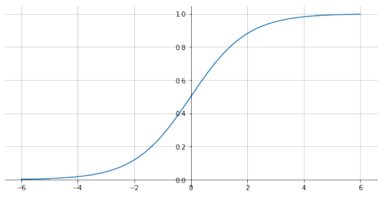
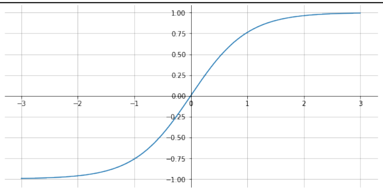
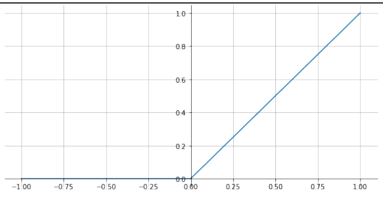
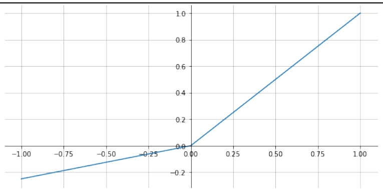
| | | |
|---|---|---|
| Sigmoid | $f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ |  |
| Tanh | $f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ |  |
| ReLU | $f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$ |  |
| Leaky ReLU | $f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha \cdot x, & \text{otherwise} \end{cases}$ |  |

Table 3.1: Common activation functions

### 3.2.3 Optimization

Although gradient descent performs well with simple loss surfaces and small datasets, it struggles when the loss function is non-convex as it can converge to a local minimum. In addition, computing the derivatives of the loss function for each data point at each iteration is slow and inefficient. In stochastic gradient descent (3.21), each update is instead calculated based on the gradient of a single data point. The noisier gradients can help to avoid local minima at the cost of more frequent updates to the weights and more variance in the performance of the model. Mini-batch gradient descent (3.22) finds a middle ground between the two approaches by using only a small subset of the dataset, called the mini-batch, in each update. Using mini-batches, the gradients for all the examples in the batch can be computed in parallel and then be used to update the weights, resulting in less noisy gradient signals and more efficient processing.

$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \nabla L(\theta^{(t)}; \hat{y}_i, y_i) \tag{3.21}$$

$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \frac{1}{n} \sum_{k=i}^{i+n} \nabla L(\theta^{(t)}; \hat{y}_k, y_k) \tag{3.22}$$

Gradient descent with momentum seeks to improve convergence by adding an additional hyper-parameter, momentum, $\alpha$. By reducing the effects of repeated oscillations in the directions of the gradient while accelerating consistent ones, it functions analogously to the momentum of a moving object. The model can then learn faster while also having a higher chance of escaping local minima. The update rules are the following:

$$\begin{aligned} v^{(t+1)} &= \alpha \cdot v^{(t)} - \eta \nabla L(\theta^{(t)}) \\ \theta^{(t+1)} &= \theta^{(t)} + v^{(t+1)} \end{aligned} \tag{3.23}$$

With Nesterov momentum [116], gradients are instead computed in a lookahead fashion using the current "velocity" values. This correction allows for the optimization process to better adapt to the topology of the error function. It uses the following update rules:

$$\begin{aligned} v^{(t+1)} &= \alpha \cdot v^{(t)} - \eta \nabla L(\theta^{(t)} + \alpha \cdot v^{(t)}) \\ \theta^{(t+1)} &= \theta^{(t)} + v^{(t+1)} \end{aligned} \tag{3.24}$$

One of the most popular optimizer is Adam [66], which stands for adaptive moments. Adam includes both first and second moment information to adapt the learning rate of each parameter. It does so by keeping an exponential moving average of these moments. Because it starts with estimates of these moments initialized at zero, they are biased towards this value, especially in the initial steps. To counteract this, Adam includes bias correcting updates for these estimates. The update rules are the following:

$$\begin{aligned} g^{(t+1)} &= \nabla L(\theta^{(t)}) \\ m^{(t+1)} &= \beta_1 m_t + (1-\beta_1) g^{(t+1)} \\ v^{(t+1)} &= \beta_2 v_t + (1-\beta_2) {g^{(t+1)}}^2 \\ \hat{m}^{(t+1)} &= \frac{m^{(t+1)}}{1-\beta_1} \\ \hat{v}^{(t+1)} &= \frac{v^{(t+1)}}{1-\beta_2} \\ \theta^{(t+1)} &= \theta^{(t)} - \eta \frac{\hat{m}^{(t+1)}}{\sqrt{\hat{v}^{(t+1)}} + \varepsilon} \end{aligned} \tag{3.25}$$

Where $m_t$ and $v_t$ are the first and second moments and $\hat{m}_t$ and $\hat{v}_t$ their bias corrected counterparts. $\varepsilon$ is a small constant added for numerical stability. $\beta_1$ and $\beta_2$ are the exponential decay rates for the first and second-moment estimates, with a typical value of 0.9 and 0.999 respectively.

Other methods for numerical optimization that rely on higher order derivatives, such as Newton's method, Levenberg–Marquardt and Broyden–Fletcher–Goldfarb–Shanno algorithms, despite often converging in fewer iterations in many conditions, are not typically used in Deep Learning contexts, due to the increasing complexity of computing second these derivatives.

### 3.2.4 Initialization

Another important aspect in the optimization process is weight initialization. Different properties of the derivatives of the activation functions at various values of its inputs can affect the ability of the model to learn. Tanh and sigmoid functions, for example, saturate at very high or low values in its inputs, setting its gradients to zero. In addition, weights with very high or very low absolute values may increase or decrease the values of the activations from one layer to the next, which can lead to vanishing or exploding gradients. As such, it is critical that weights are initialized in such a way as to keep inputs to layers in a range that allows for the model to converge. Initialization schemes aim to keep the values of the inputs to the activation function close to their optimal range, typically around zero, throughout all the layers of the network. Most do this by manipulating the distribution from which the weights are generated. The two most common initialization schemes are Xavier initialization [38] and He initialization [42]. Xavier initialization uses a normal distribution with mean zero and variance:

$$Var(W) = \frac{1}{n_{in}} \tag{3.26}$$

Where $n_{in}$ is the dimensionality of the input of the layer. This maintains the same approximate variance in the input and outputs of the layer both in the forward and backward passes, improving gradient flow. Whereas Xavier initialization is applicable for linear layers, or layers with activation functions which closely approximate linear ones, such as the sigmoid or tanh near zero, He initialization applies to layers with ReLU activation functions. The variance is instead computed with the expression:

$$Var(W) = \frac{2}{n_{in}} \tag{3.27}$$

## 3.3 Convolutional Neural Networks

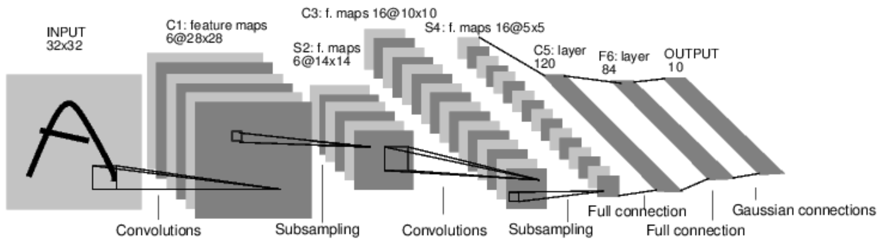


Figure 3.4: Convolutional Neural Network, LeCun et al. [73].

Certain types of data, such as images, have an inherent grid-like structure which an MLP cannot exploit, as they assign an entirely separate set of weights to each pixel. Convolutional layers, unlike fully connected layers, exploit the spatial characteristics of their inputs by applying filters of weights to them, in a sliding window fashion, such that a set of weights is always multiplied in the same relative position everywhere in the input. Models containing these layers are called Convolutional Neural Networks (CNNs). In reality, this operation is usually implemented as a matrix multiplication, which is highly efficient on modern massively parallel hardware such as GPUs. Despite their name, they are not computed with the convolution operator used in signal processing, but with the closely related cross-correlation operator, defined by:

$$F \star I(x,y) = \sum_{j=-F_H}^{F_H} \sum_{i=-F_W}^{F_W} F(i,j) \cdot I(x+i, y+j) \tag{3.28}$$

Where $F$ is the weight filter, $I$ is the input, and $F_H$ and $F_W$ are the height and width of the filter.

The weight sharing inherent to the convolution allows the layer to reuse knowledge extracted from any position in the input to any other position, which regularizes the network, greatly reducing the number of training examples needed and increasing its ability to generalize. It also gives it a degree of translation equivariance. This means that the translated output of a layer is roughly the same as the output of the layer when its input is translated by the same amount.

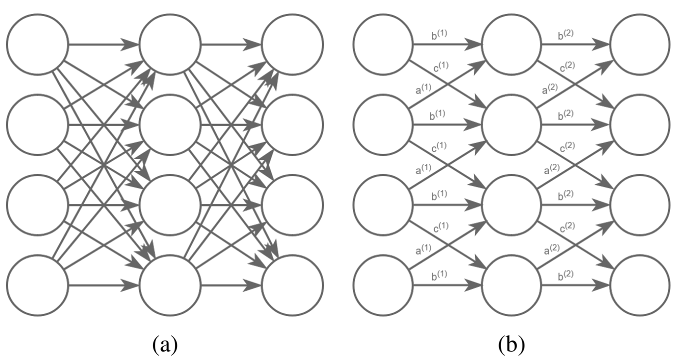


Figure 3.5: Comparison of independent weights in an MLP (left) and weight sharing in a 1D CNN with a filter size of $3 \times 3$ and stride $1 \times 1$ (right)

Convolutional layers can be constructed to operate on data with any number of dimensions, from 1D layers used in sound processing, 2D layers used in image analysis and 3D layers use in voxel-based tasks. We focus on 2D convolutional layers due to their effectiveness in image processing.

At each layer, the input to a 2D convolutional layer is a tensor of shape $H_{in} \times W_{in} \times D_{in}$, its height, width and depth. Depth translate to the input's number of channels. In traditional images, the channels refer to the red, green and blue channels. In deeper layers, the number of channels is usually much higher and reflects the number of shape or semantic features the layer that generated it was able to capture. Each layer is composed of $D_{out}$ filters of weights, each of size $F_H \times F_W \times D_{in}$ and a vector of $D_{out}$ biases. To apply the layer to the input, each weight filter is multiplied element-wise with an equally sized section of the input, the associated bias is added and then the activation function is applied. This process is repeated over all parts of the input in a sliding window fashion, skipping $S$ positions between applications, the stride number. Each filter produces a 2D feature map representing the activations of the filters at each position in the input. These maps are concatenated along the channel axis to form a tensor of shape $H_{out} \times W_{out} \times D_{out}$:

$$\begin{aligned} W_{out} &= \frac{W_{in} - F_W + 2P}{S} + 1 \\ H_{out} &= \frac{H_{in} - F_H + 2P}{S} + 1 \end{aligned} \tag{3.29}$$

Where $P$ is the padding amount, the number of pixels added to each dimension before applying the filters. Padding is often used to maintain the spatial dimensions equal throughout multiple layers. Even with the same spatial size, the number of activations that affect other later activations grows further downstream they are, when the filter size is higher than 1. The number of upstream neurons that a neuron is affected by is called its receptive field. A higher receptive field means that a neuron has access to a greater amount of surrounding context, which is essential for high-level semantic features.

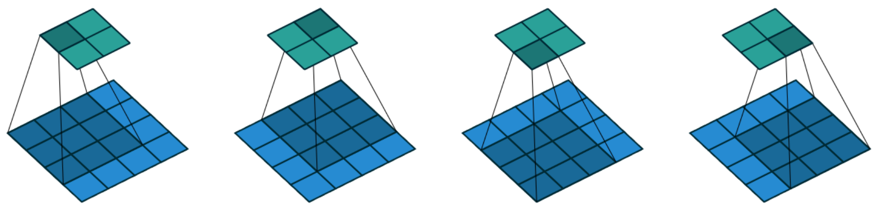


Figure 3.6: Convolutional layer with a $4 \times 4$ input, $3 \times 3$ filter size and stride of 1, Dumoulin and Visin [27]

Another typical operation of CNNs is pooling. In pooling, the goal is to reduce the spatial dimensions of an activation map by replacing groups of values with the result of a reduce operation, such as max or average. Pooling layers are defined by size, stride and, padding, but lack a depth parameter. The output of a pooling layer has size $H_{out} \times W_{out} \times D_{in}$. The most common type of pooling layer is the max pooling layer, depicted in Figure 3.7. By returning only the maximum of the activations in each $F_H \times F_W$, the max pooling layer gives a CNN a degree of translation invariance, as small translations in the input may fall in the same area of the max pooling layer, yielding the same output. This is particularly useful in classification, as the highest activations of features are usually the most distinctive and therefore discriminatory. Pooling layers also increase the receptive fields of neurons downstream, without the need for larger filters in those layers.

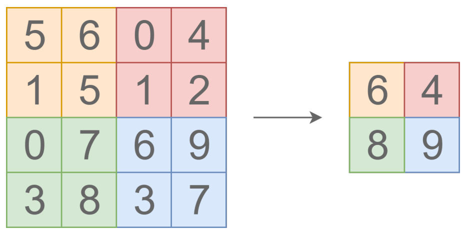


Figure 3.7: Example of application of a max pooling layer with filter size and stride of $2 \times 2$.

In global pooling, the reduce function is instead applied to the entire input, for each channel. It is equivalent to using a pooling layer with a filter size equal to the input size. Global pooling layers have the advantage of producing outputs with constant spatial dimensions, of size $1 \times 1 \times D_{in}$. A global max pooling layer is depicted in Figure 3.8.

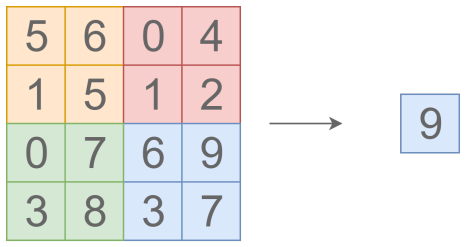


Figure 3.8: Example of application of a global max pooling layer.

Typical CNNs, such as the ones used for image classification, consist of sequences of multiple convolutional layers followed by pooling layers, until the last pooling layer, which is connected to an MLP that produces the final classification. By using a global pooling layer as the last spatial layer, these CNNs can use MLPs to classify variable size images.

In semantic segmentation, the model is tasked with classifying each individual pixel. These typically use fully convolutional networks that transform input images into lower resolution semantic spaces before predicting the higher resolution classification mask. To perform this upsampling in a learnable way, these architectures make use of transposed convolutional layers, so called because they can be implemented using the transpose of the matrix that would be used in a normal convolutional layer. Instead of multiplying the weight filters with same sized sections of the input, transposed convolutions instead multiply a single value of the input with the weights and add that to an accumulator matrix.

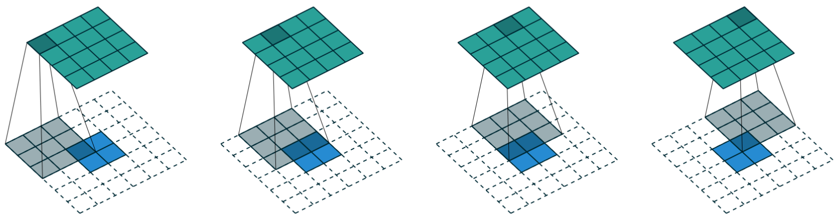


Figure 3.9: Transposed convolutional layer with $2 \times 2$ input, $3 \times 3$ filter size and stride of 1, Dumoulin and Visin [27]

Some architectures instead upsample the feature map using a conventional image resizing method, such as nearest neighbor or linear interpolation, before feeding the upsampled map to a conventional layer, which is tasked with performing the learned part of the operation. This can avoid some of the artifacts that can originate from adding multiple activations to the same positions in some configurations of transposed convolutional layers.

A third option for upsampling feature maps is unpooling. Unpooling aims to reverse the operation performed by pooling layers. However, because this operation is not reversible, the result is always lossy. Max unpooling layers use the indices of the maxima captured in their corresponding max pooling layer to project the feature map to a higher size, maintaining the locations of the maxima while setting all other values to zero. This operation is depicted in Figure 3.10. One example of its application is in the SegNet architecture 3.11.

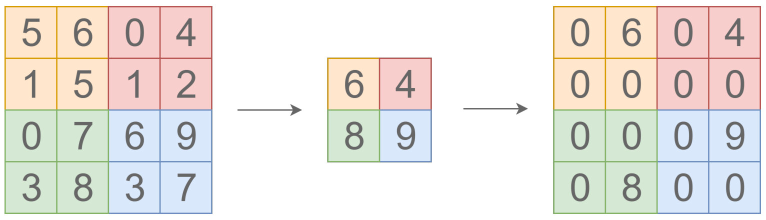


Figure 3.10: Max pooling layer followed by its associated unpooling layer.

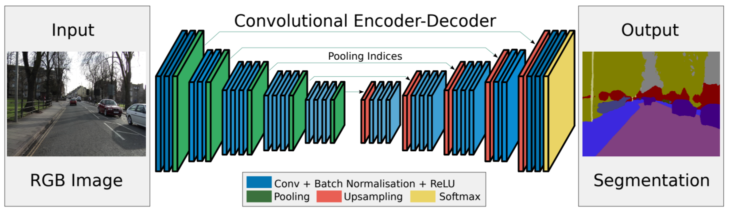


Figure 3.11: SegNet architecture, a fully convolutional network for semantic segmentation, Dumoulin and Visin [27]

.

In object detection, both regression and classification are combined into a single task. Object detection models are trained to predict bounding boxes for objects, as well as their associated class.

In one-shot detection architectures, images are split into a $n \times m$ grid. For each section in the grid, anchors are created at $s$ different scales and $r$ different ratios for a total of $n \times m \times s \times r$ anchors. The ground truth annotations and anchors with the highest overlap are matched and their offsets computed. Offsets are the differences in center coordinates, width, and height. These anchors are depicted in Figure 3.12. A fully convolutional network is tasked with predicting offsets and classification scores for each of the anchors, for a total of $n \times m \times s \times r \times (4+c)$ outputs, where $c$ is the number of classes. A regression loss is used for the offsets and a classification loss for the classes. In architectures such as the Feature Pyramid Network [78], or RetinaNet Lin et al. [79], the grid is created at multiple scales, allowing the network to detect objects with a wider range of sizes.

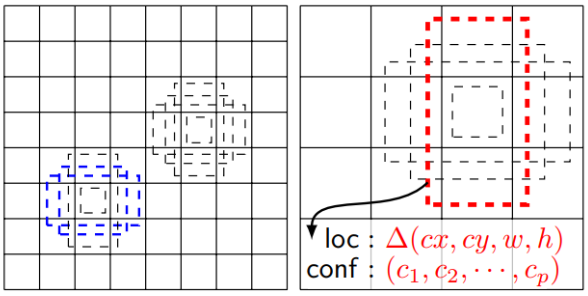

loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \cdots, c_p)$

Figure 3.12: Anchors in a one-shot detection architecture, Liu et al. [80].

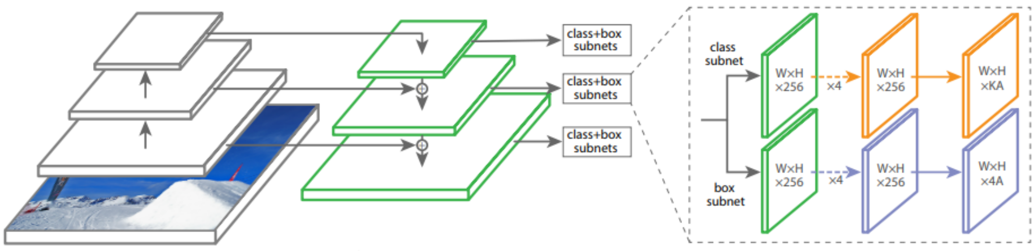


Figure 3.13: RetinaNet architecture, Lin et al. [79].

# Chapter 4

# Methodology

This chapter details the methodology employed to build and train a model capable of detecting and diagnosing anomalies in mammograms.

## 4.1 Data

The dataset chosen was INbreast [88]. It consists of 410 Full-Field Digital Mammogram FFDM images of both craniocaudal (CC) and mediolateral oblique (MLO) views, from 115 cases obtained from the Breast Centre in Centro Hospitalar São João. Regions of interest (ROIs), such as microcalcifications, masses, spiculated regions, distortions, and asymmetries are annotated with pixel-level boundaries. There are also image-level annotations of ACR and BI-RADS classifications. The exams were split into training and validation set at the patient level, with 80% of patients used for training and 20% reserved for validation.

## 4.2 Architecture

The model used in this work builds on the RetinaNet architecture [79] and extends it by adding additional outputs relevant to mammogram analysis and by taking into account multiple inputs, consisting of each of the 4 images that are part of an exam. The model is made fully modular and so each of these components can be added or removed independently. By predicting all outputs in a single pass, the entire model can be trained in a fully end-to-end process, simplifying the learning process.

### 4.2.1 Single Image Architecture

As in RetinaNet, the backbone of the model is a ResNet [43] submodel, which receives the input image. As in Lin et al. [78], we extract the features output by the last block of each stage in the network (Figure 4.2). Each stage is defined by the blocks producing feature maps with the same

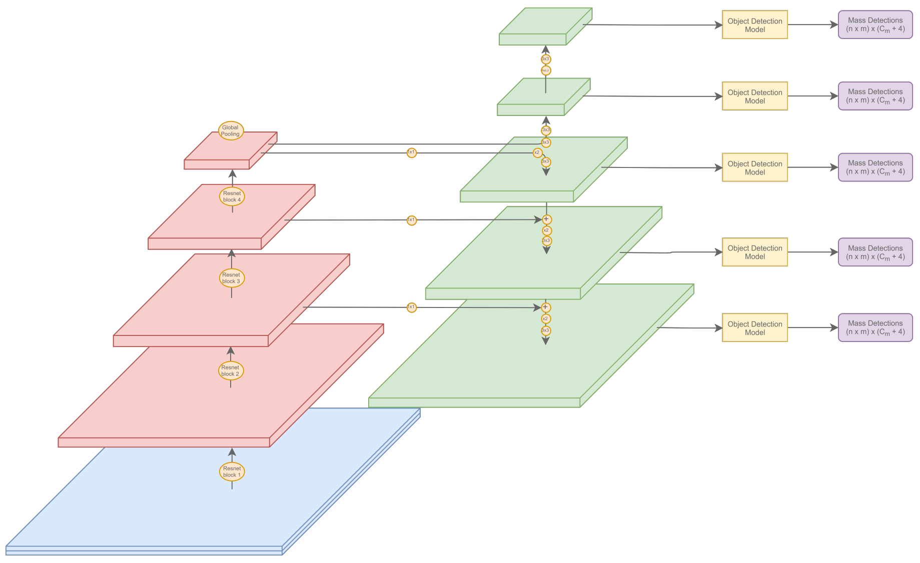


Figure 4.1: The full object detection model proposed by Lin et al. [78], showing inputs (blue), bottom-up feature pyramid (red) and the top-down feature pyramid (green)

spatial dimensions. These activations form the feature pyramid for the bottom-up pathway. These are then used to build a second feature pyramid in a top-down pathway as depicted in Figure 4.3.

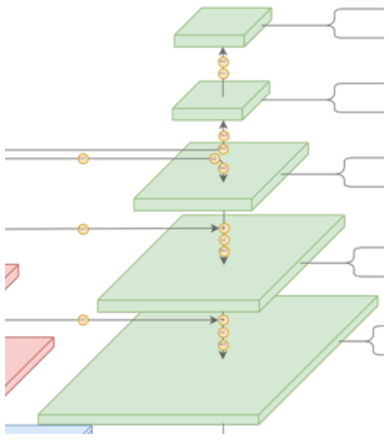


Figure 4.3: Feature pyramid of the model.

As in Lin et al. [78], a fully convolutional object detection submodel is fed the features of each level of the top-down pyramid and is tasked with predicting bounding boxes, through regression, and associated classes of objects contained in the image at the corresponding image scale. By predicting at multiple feature scales, the model can efficiently detect objects of different sizes.

In mammogram analysis, both large masses and small microcalcifications are of clinical interest. However, their large differences in relative size would require a model based on the RetinaNet architecture to predict significantly higher resolution outputs, with a corresponding increase in training and inference times. We instead add a second detection model tasked with predicting bounding boxes for clusters of microcalcifications at the same resolution. Due to the ambiguous

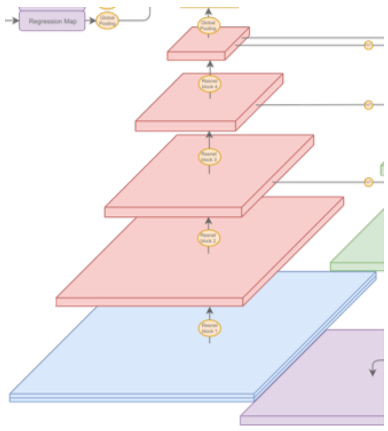


Figure 4.2: Backbone of the model.

nature of microcalcification clusters, we introduce a novel loss function that is tailored to be less sensitive to the particular grouping the detection model chooses. This is detailed in 4.3. The detection outputs of the model are depicted in Figure 4.4.

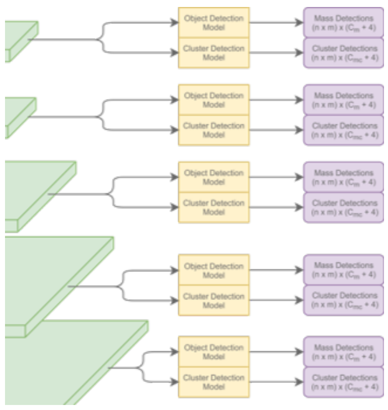

Object Detection Model
Cluster Detection Model
Mass Detections (n x m) x ($C_m$ + 4)
Cluster Detections (n x m) x ($C_{mc}$ + 4)

Figure 4.4: Detection outputs of the model.

Two alternative methods of microcalcification detection are explored: segmentation and regression. In segmentation, the network is tasked with predicting which pixels belong to either a single microcalcification or a cluster of microcalcifications. In regression, the network is tasked with predicting a continuous map roughly corresponding to the proximity to nearby microcalcifications. These outputs are depicted in Figure 4.5.

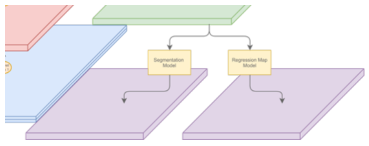


Figure 4.5: Segmentation and regression outputs of the model.

A classification submodel (Figure 4.6) is tasked with classifying the mammogram into one of negative, benign or malignant based on the last feature map of the bottom-up feature pyramid and

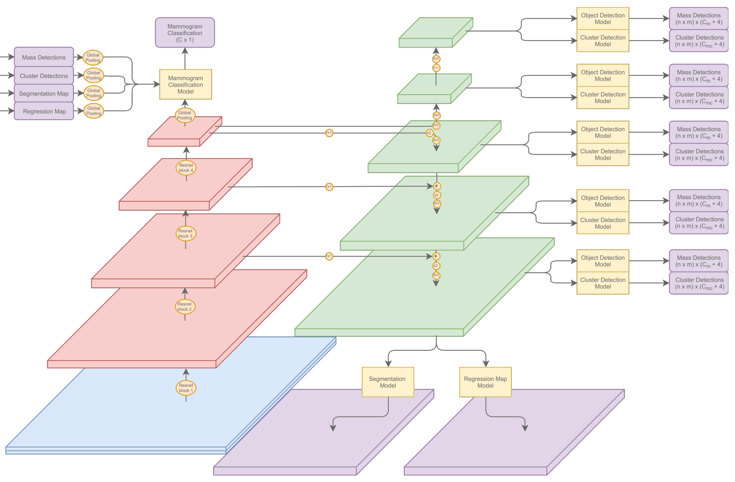


Figure 4.7: Proposed extensions to the RetinaNet architecture

each of the outputs from the other models. By using a learnable submodel, rather than a fixed function based on the other outputs, the network can still learn from mammograms that only have associated ground-truth classifications and no anomaly annotations. These extensions are depicted in Figure 4.7.

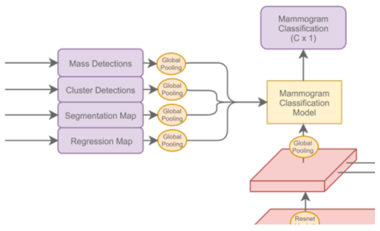


Figure 4.6: Classification head of the model.

Using a multi-task model allows not only for the network to share features between the various tasks but also to learn in a fully supervised way from datasets that contain only subsets of the data necessary for all the tasks. Some datasets in mammography, for example, only contain image-level annotations of the patient's cancer classification. In a multi-task setting, the network can be trained with these images by updating the weights only from the gradients of the classification loss and the features learned in this mode may also improve the performance of the network in its other tasks, as they are highly related in this problem.

### 4.2.2 Multiple Image Architecture

A single image, however, is often insufficient to accurately detect anomalies in the tissue. During the decision process, radiologists make use of multiple views, usually MLO and CC, of both breasts simultaneously. In addition to potentially revealing anomalies not visible in one view, asymmetries between the breasts can also be indicative of malignancy, whereas as an apparent

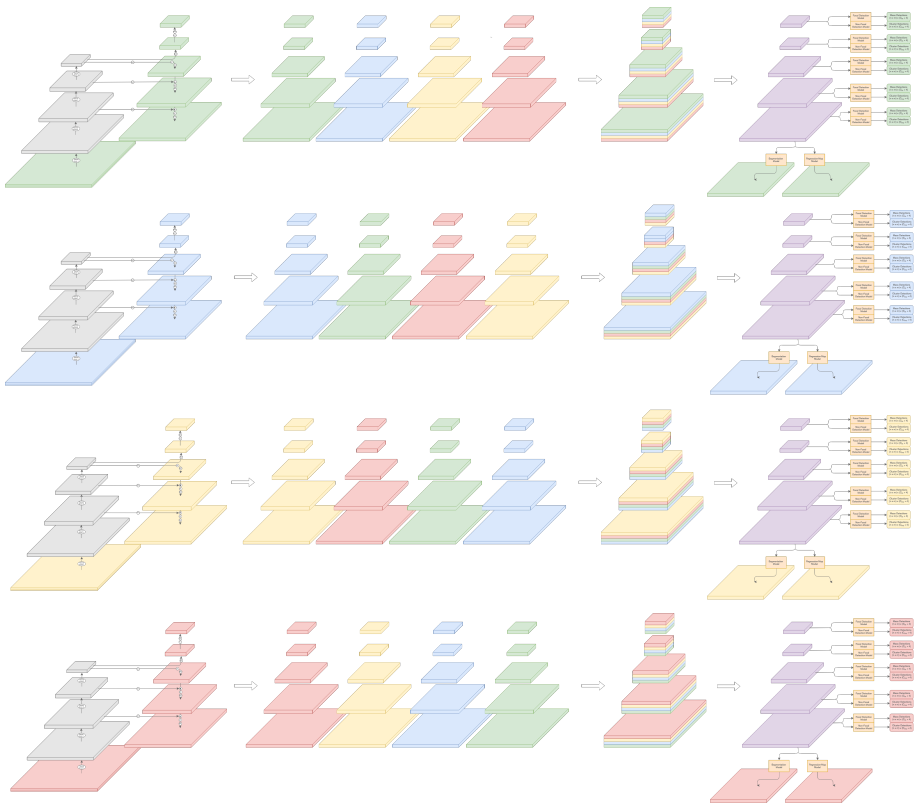


Figure 4.8: Generalization of the proposed architecture to multiple images

anomaly that is present in both breasts is less likely to be malignant. CAD systems have historically had difficulty taking into account this link between images due to complex deformations breast tissue undergoes in the mammogram capture process.

We propose fusing information from multiple images both explicitly and implicitly. Right and left images of the same view are fused explicitly through image registration or alignment, a process detailed in Section 4.4. The two aligned images are stacked along the channel axis, resulting in a spatially aligned image with one channel corresponding to the right image and one corresponding to the left image, which is then fed to the backbone network. This allows the network to directly relate co-located features from one breast to the other.

The views are fused implicitly by merging high-level features extracted from each of the images. First, bottom-up and top-down feature pyramids are generated from each of the input images. Then, for each of the images, a new feature pyramid is obtained by stacking, at each level of the pyramid, the features corresponding to each image. These stacked feature pyramids will, at each level, contain first the features corresponding to that image, then the features corresponding to the image from the other view, followed by the features corresponding to the image corresponding to the same view and opposite laterality, and ending with the features from the image corresponding to the other view and opposite laterality. These feature pyramids are then reduced feature-wise using a feature-reducing submodel. Output submodels are then fed the reduced feature maps to produce final outputs for each of the 4 images. This process is depicted in Figure 4.8.

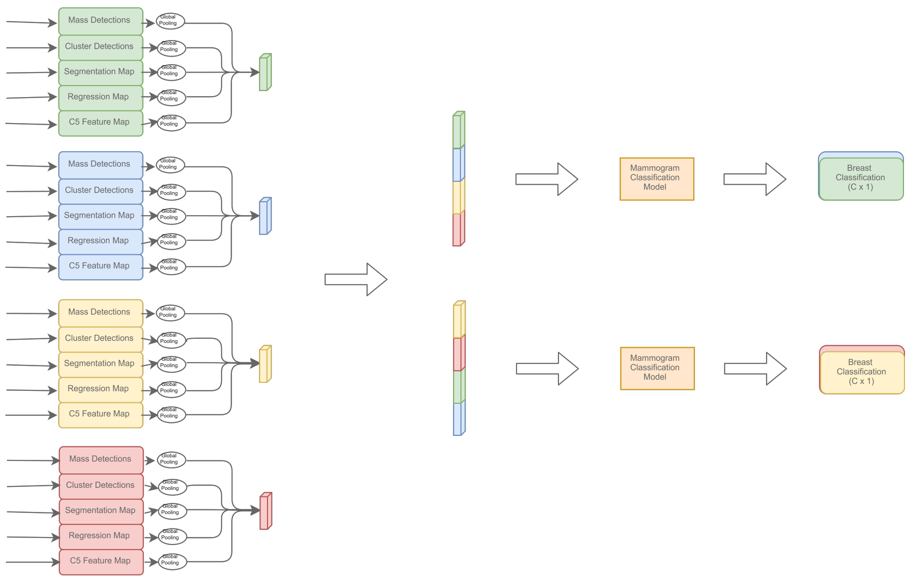


Figure 4.9: Generalization of the classification submodel to multiple images

Because the set of images consists of two pairs corresponding to the two views of the patient's breasts, only two classification outputs are required. The last feature maps from the bottom-up feature pyramid of each image are fused and fed into a global pooling layer, along with the outputs corresponding to the same image to form that image's feature vector. Feature vectors from all images are then appended. The vectors corresponding to the first breast are placed at the start of the resulting vector. This vector is fed to MLP which is tasked with predicting the classification for that breast. Finally, the process is repeated for the second breast. This submodule is depicted in Figure 4.9.

## 4.3 Cluster Loss

Traditional losses used in object detection tasks seek for the network to output a specific prediction, that is identical to the ground truth. Clusters of objects, however, are inherently ambiguous in their composition; multiple assignments of a set of points to clusters may be equally valid. To capture this ambiguity, we propose a new loss function that does not penalize predictions that differ from the ground truth as long as the prediction covers the same areas as the ground truth annotations by computing a continuous approximating of an Intersection over Union (IOU) function. A set of bounding boxes $B$ with elements $B_i = (x_i^{(1)}, y_i^{(1)}, x_i^{(2)}, y_i^{(2)})$ is interpreted as a mixture of continuous two-dimensional uniform distributions, represented by the probability density function $p_B(x,y)$.

$$p_B(x,y) = \sum_i^N \pi_i \cdot U(a_i, b_i), \quad \sum_i^N \pi_i = 1 \tag{4.1}$$

Where:

$$a_i = (x_i^{(1)}, y_i^{(1)}), \quad b_i = (x_i^{(2)}, y_i^{(2)}) \tag{4.2}$$

This mixture is then approximated by a mixture of two-dimensional Gaussians, where each Gaussian approximates one of the uniform distributions by minimizing the KL divergence. The values for $\mu$ and $\sigma$ in the one-dimensional case can be derived as follows:

$$\begin{aligned}
\min_{\mu,\sigma} KL(U(a,b) \,||\, N(\mu,\sigma)) &= \min_{\mu,\sigma} \int_{-\infty}^{\infty} U(a,b) \cdot log\left(\frac{U(a,b)}{N(\mu,\sigma)}\right) dx \\
&= \min_{\mu,\sigma} \int_a^b \frac{1}{b-a} \cdot log\left(\frac{\sqrt{2\pi\sigma^2}}{(b-a)e^{\frac{(x-\mu)^2}{2\sigma^2}}}\right) dx \\
&= \min_{\mu,\sigma} \int_a^b \frac{1}{b-a} \cdot \left(\frac{1}{2}log(2\pi\sigma^2) - log(b-a) - \frac{(x-\mu)^2}{2\sigma^2}\right) dx \\
&= \min_{\mu,\sigma} \frac{1}{2}log(2\pi\sigma^2) - log(b-a) - \frac{a^2+b^2+ab-3a\mu-3b\mu+3\mu^2}{6\sigma^2} \\
&= \min_{\mu,\sigma} \frac{1}{2}log(2\pi\sigma^2) - \frac{a^2+b^2+ab-3a\mu-3b\mu+3\mu^2}{6\sigma^2}
\end{aligned} \tag{4.3}$$

$$\begin{aligned}
&\frac{\partial}{\partial\mu} KL(U(a,b) \,||\, N(\mu,\sigma)) = 0 \\
&\Rightarrow \frac{3a+3b+6\mu}{6\sigma^2} = 0 \\
&\Rightarrow \mu = \frac{a+b}{2}
\end{aligned} \tag{4.4}$$

$$\begin{aligned}
&\frac{\partial}{\partial\sigma} KL(U(a,b) \,||\, N(\mu,\sigma)) = 0 \\
&\Rightarrow \frac{a^2+b^2+a(b-3\mu)-3b\mu+3(\mu^2+\sigma^2)}{3\sigma^3} = 0 \\
&\Rightarrow \sigma^2 = \frac{(b-a)^2}{12}
\end{aligned} \tag{4.5}$$

The two-dimensional case follows simply from these equations.

The loss function is then defined as the Jensen–Shannon divergence between the two Gaussian mixture models, where:

$$D_{JS}(P,Q) = \frac{KL(P||Q) + KL(Q||P)}{2} \tag{4.6}$$

Because computing the KL divergence between arbitrary Gaussian mixture models is intractable, a variational upper bound is computed instead. We use the method proposed by Durrieu et al. [28]. Let $f$ and $g$ denote the probability density functions for each of the mixture models and $f_a$, $\pi_a$, $g_b$ and $\omega_b$ their $a$-th and $b$-th Gaussians and their associated weights:

$$f = \sum_a \pi_a f_a$$
$$g = \sum_b \omega_b g_b$$

They then introduce variational parameters $\phi_{b|a}$ and $\psi_{a|b}$ such that:

$$\begin{aligned}
\forall b,a : \phi_{b|a} > 0, \quad & \sum_b \phi_{b|a} = \pi_a \implies \sum_{a,b} \phi_{b|a}\, f_a = f \\
\forall a,b : \psi_{a|b} > 0, \quad & \sum_a \psi_{a|b} = \omega_b \implies \sum_{a,b} \psi_{a|b}\, g_b = g
\end{aligned}$$

$$\begin{aligned}
KL(f||g) &= \int f\, log\left(\frac{f}{g}\right) dx \\
&= -\int f\, log\left(\frac{g}{f}\right) dx \\
&= -\int f\, log\left(\frac{\sum_{a,b} \psi_{a|b}\, g_b}{f}\right) dx \\
&= -\int f\, log\left(\sum_{a,b} \frac{\psi_{a|b}\, g_b}{\phi_{b|a}\, f_a} \frac{\phi_{b|a}\, f_a}{f}\right) dx \\
&\leq -\int f \sum_{a,b} \frac{\phi_{b|a}\, f_a}{f}\, log\left(\frac{\psi_{a|b}\, g_b}{\phi_{b|a}\, f_a}\right) dx \\
&= -\int \sum_{a,b} \phi_{b|a}\, f_a \left( log\left(\frac{\psi_{a|b}}{\phi_{b|a}}\right) + log\left(\frac{g_b}{f_a}\right)\right) dx \\
&= -\int \sum_{a,b} \phi_{b|a}\, f_a\, log\left(\frac{\psi_{a|b}}{\phi_{b|a}}\right) dx - \sum_{a,b} \phi_{b|a} \int f_a\, log\left(\frac{g_b}{f_a}\right) dx \\
&= \int \sum_{a,b} \phi_{b|a}\, f_a\, log\left(\frac{\phi_{b|a}}{\psi_{a|b}}\right) dx + \sum_{a,b} \phi_{b|a} \int f_a\, log\left(\frac{f_a}{g_b}\right) dx \\
&= KL(\phi||\psi) + \sum_{a,b} \phi_{b|a}\, KL(f_a||g_b) \\
&\overset{\text{def}}{=} KL_{\phi,\psi}(f||g)
\end{aligned} \tag{4.7}$$

Where $KL_{\phi,\psi}(f||g)$ is the variational upper bound to the target value, $KL(f||g)$. $KL(f_a||g_b)$ is the KL divergence between two Gaussians, which can be written in closed form with the following expression:

$$KL(N(\mu_1,\Sigma_1)||N(\mu_2,\Sigma_2)) = \frac{1}{2}\left(log\frac{|\Sigma_2|}{|\Sigma_2|} - d + tr(\Sigma_2^{-1}\Sigma_1) + (\mu_2-\mu_1)^T\Sigma_2^{-1}(\mu_2-\mu_1)\right) \tag{4.8}$$

The values $\hat{\phi}$ and $\hat{\psi}$ that minimize the upper bound $KL_{\phi,\psi}(f||g)$ are then computed. Starting with initial values $\phi_{b|a} = \psi_{a|b} = \pi_a\omega_b$, the values are optimized using the following update rules, derived from equation 4.7:

$$\psi_{a|b} = \frac{\omega_b\phi_{b|a}}{\sum_{a'}\phi_{b|a'}} \tag{4.9}$$

$$\phi_{b|a} = \frac{\pi_a\psi_{a|b}\ e^{KL(f_a||g_b)}}{\sum_{b'}\psi_{a|b'}\ e^{KL(f_a||g_{b'})}} \tag{4.10}$$

By minimizing the upper bound given by $KL_{\phi,\psi}(f||g)$, the target value, $KL(f||g)$ is also minimized. The variational upper bound $KL_{\phi,\psi}(f||g)$ is fully differentiable with respect to its inputs, the parameters of both Gaussian mixture models, which are fully differentiable with respect to the parameters of the uniform mixture models, which are, in turn, fully differentiable with respect to the parameters of the bounding boxes output by the model, this function can, therefore, be used as a loss function to train a model.

At inference time the parameters of the uniform distributions - and therefore of the bounding boxes - can then be retrieved from the Gaussian mixture model as follows:

$$a_i = \mu_i - \sqrt{3}\sigma, \quad b_i = \mu_i + \sqrt{3}\sigma \tag{4.11}$$

## 4.4 Preprocessing and Data Augmentation

To help the convolutional model analyze matching regions in the breast, we align the two images corresponding to the same view of the two breasts, right and left. To align the mammograms we first find the contour of the breast. After thresholding the image, the breast is assumed to be the largest contiguous area. This simple approach to segmenting the breast works well in the INbreast dataset as its images have a relatively high contrast between background and foreground. In less ideal datasets, such as DDSM, a more robust method would be required. The curve farthest away from the center is considered to be the contour of the breast.

Most images contain extra tissue above or below the breast which may not be present in the other corresponding image and so can be removed. To do this, the magnitudes of the second derivatives of the curve of the contour of the breast are computed. If this value exceeds a preset threshold, the sections of the curve above or below the point are removed.

We then predict the position of the nipple, the primary reference point to compare the images. In this work, the nipple is assumed to be the point along the contour of the breast that is farthest away from the base of the breast, the edge of the image where the torso would be located.

We then choose $n$ equidistant points along the base edge and $n$ points along the contour, where half the points are located above the nipple in the contour and half are located below the nipple, with the points in each set being equidistant. Then the positions of each pair of points from the contour and base sets are linearly interpolated to form $m-2$ new points for a total of $n \times m$ points. These points are assumed to correspond to similar positions in their respective images and so function as control points in piece-wise linear interpolation between the two images.

The area containing the breast is cropped from the image to maximize the relative area of tissue in the image. This process is depicted in Figure 4.10.

The main limitations of this method lie in the detection and processing of the nipple. Some image pairs contain nipples in only one of the images. In these cases, the nipple is matched with very structurally different tissue, resulting in noticeable visual artifacts. Even in cases where both nipples are matched successfully, slight differences in the exact points of the nipple which are matched still sometimes result in some distortion.

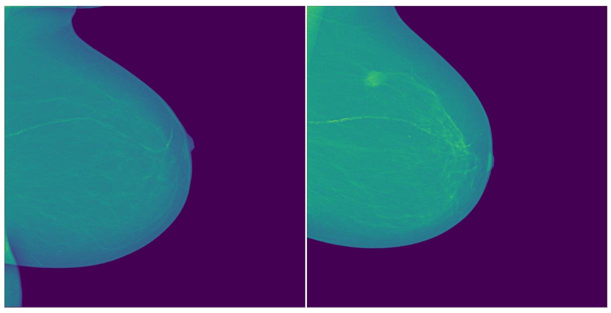
Lesion annotations from the dataset are converted to both bounding boxes, by taking the maxima and minima of the points of each lesion, and masks. For very small lesions, like microcalcifications, bounding boxes are instead computed after a clustering step. Clustering is performed using the DBSCAN algorithm [22] with hand-picked values. The bounding box ground truth annotations used in training, therefore, represent both large masses and clusters of small microcalcifications. Multiple masks are computed for each image, unchanged masks, dilated or blurred masks and cluster masks, reflecting different types of annotation that might be fed to the models. Unchanged masks reflect precise annotations for both masses and microcalcifications. Dilated or blurred masks allow for the masks to be downscaled more aggressively while being more lenient with precise segmentations. Cluster masks reflect the clusters of microcalcifications computed with DBSCAN. Clusters are drawn using an ellipse that approximates its dimensions. These masks also aim to not excessively punish inexact segmentations.

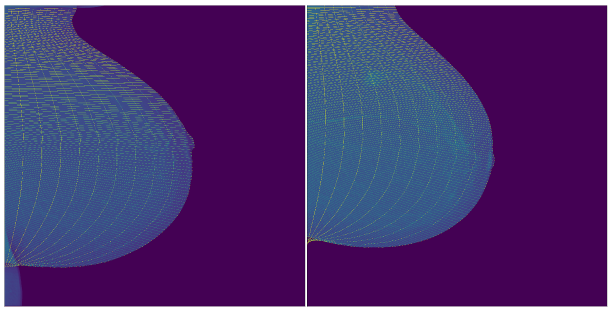
To help regularize the network, images are augmented with random rotations, translations, scalings, shearing, and flipping. An affine matrix is constructed with random values for these operations. Then the matrix is applied to the image, as well as to its bounding boxes and masks. In multiple image mode, the same transformation is applied to each image for consistency.

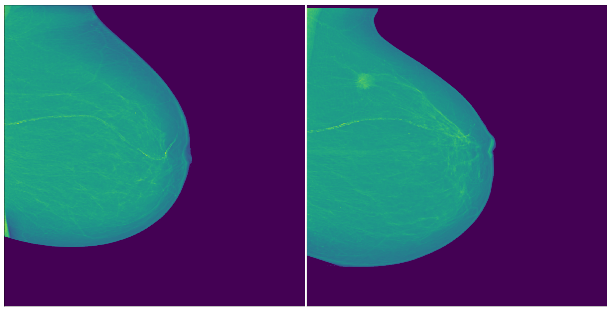
## 4.5 Implementation

Deep Learning frameworks significantly speed up research and applications by providing commonly used components, such as automatic differentiation. The main frameworks in use today are:

- Tensorflow [4]


(a) Left mammogram (b) Right mammogram


(c) Grid for left mammogram (d) Grid for right mammogram


(e) Aligned left mammogram (f) Aligned right mammogram

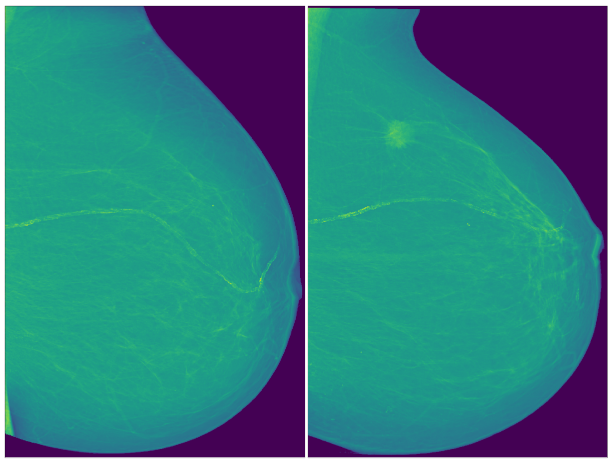

(g) Cropped aligned left mamogram (h) Cropped aligned right mamogram

Figure 4.10: Mammogram preprocessing pipeline

- PyTorch [91]
- Chainer [121]
- Caffe [60]
- CNTK [105]
- MXNet [16]
- Torch [21]
- Deeplearning4j [1]
- DyNet [89]

In this work, we chose to use the Python programming language [100] for its variety of packages useful for Deep Learning and data analysis with Keras [18] for its high level APIs allowing for rapid development and integration with other frameworks. OpenCV [58, 2] was used for image processing, Pandas [85] for reading .csv and .xls files, Matplotlib [53] for plotting, Scikit-learn [92] for validation splitting and various metrics, Numpy [123] for numerical computation and IPython [93] as a development environment. The base detection architecture was based on the open-source implementation by Github user *fizyr* [29].

## 4.6 Metrics

The primary metric used in detection is the Average Precision (AP) which corresponds to the area under the corrected curve formed by plotting the Precision and the Recall at each possible detection threshold value, the confidence beyond which a detection is considered to be valid [41]. Detections are considered correct if their Intersection over Union value with a ground truth annotation is above the minimum threshold. An example of such the Precision-Recall curve is depicted in Figure 4.12.

$$Precision = \frac{\#TruePositives}{\#TruePositives + \#FalsePositives}$$

$$Recall = \frac{\#TruePositives}{\#TruePositives + \#FalseNegatives}$$

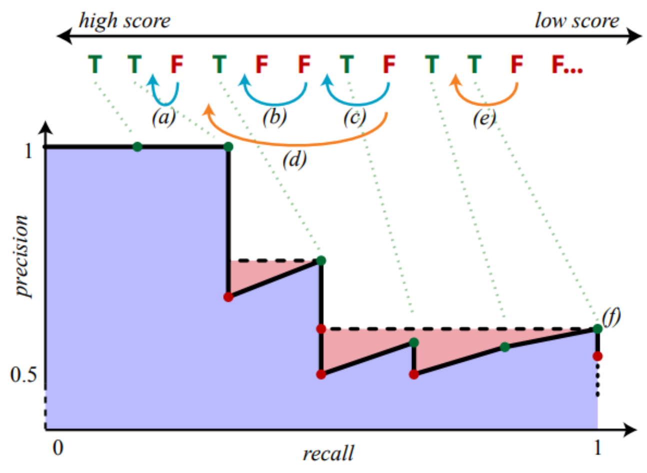


Figure 4.11: Precision-Recall curve (purple), for sequence of correct (green) and incorrect (red) detections. Corrected curve in red. Henderson and Ferrari [46].

Another common metric in breast cancer detection is the Free Receiver Operating Characteristic (FROC) curve, which plots the number of False Positives per Image (FPI) against the True Positive Rate (TPR).

In classification, the Area Under the Receiver Operating Characteristic (AUROC) is used to measure the performance of models. It corresponds to the area under the Receiver Operating Characteristic (ROC) curve, formed by plotting the True Positive Rate (TPR), or Recall, and the False Positive Rate (FPR) at each threshold value. An example of such a curve is depicted in Figure 4.12.

$$FalsePositiveRate = \frac{\#FalsePositives}{\#FalsePositives + \#TrueNegatives}$$

$$TruePositiveRate = Recall = \frac{\#TruePositives}{\#TruePositives + \#FalseNegatives}$$

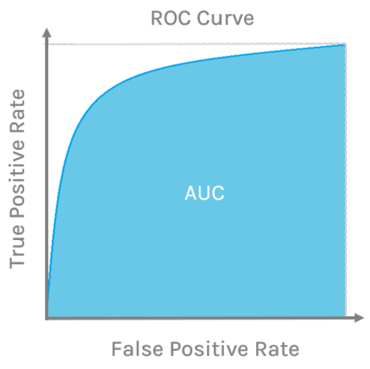


Figure 4.12: ROC curve

The quality of a segmentation is often measured using the Dice coefficient, calculated with the following expression:

$$Dice = \frac{2\#TruePositives}{2\#TruePositives + \#FalsePositives + \#FalseNegatives}$$

### 4.6.1 Training and Results

The networks were trained with the Adam optimizer with a learning rate of $1 \times 10^{-4}$ and a dropout rate of 0.35. The learning rate was halved when the validation loss failed to decrease for 50 epochs. The best result obtained was an AP of 0.62 at an IOU threshold of 0.5 and score threshold of 0.05, in single class lesion detection in single task, single image mode, with a training time of $\tilde{1}3$ hours on a Nvidia Tesla K80 GPU. The cluster loss proved too computationally expensive to train a model.

To validate the alignment and multi-image architectures, an experiment was carried out to compare their performance in single task 1 class mass detection. Here, the images were downscaled further to $300 \times 400$ to speed up training time, at the cost of performance. The results are presented in Table 4.1.

| Model | Average Precision |
|---|---|
| Non-aligned single image | 0.45 |
| Aligned single image | 0.51 |
| Non-aligned multi-image | 0.35 |
| Aligned multi-image | 0.35 |

Table 4.1: View fusion experiment results.

As expected, the results show that the additional context of the aligned mammograms helped the network to more accurately detect masses in the single image case. In the multiple image case, we speculate that this additional context is redundant due as the network already has access to all the images. We speculate that the reduced performance of the multi-image model compared to the single-image model is due to the additional complexity of merging and reducing the feature pyramids. This issue could be improved with alternate merging methods, such as not reducing the feature pyramids, longer training times, or pre-training the non-reducing layers on the single-image task.

# Chapter 5

# Conclusions and Future Work

This work described some of the first steps towards creating a system capable of detecting and classifying abnormalities in mammographic images, as well as their overall malignancy classification. We described a novel architecture based on existing object detection models to meet some of the many challenges inherent to mammographic images. To the task of detecting masses, we add the tasks of detecting clusters of microcalcifications and the classification of the mammogram into one of negative, benign or malignant. With the goal of detecting clusters while taking into account their inherent ambiguity, we introduce a novel loss function that approximates the IOU of predictions and ground truth annotations by casting them as mixtures of Gaussians and computing their KL divergence. We then extend this methodology by integrating all four views of an exam in a single, end-to-end differential model. The feature pyramid extraction component of the single-image model is fed each image and the resulting features are appended in cyclical order, level by level, before passing through a feature reducing model. This results in a feature pyramid for each input image, each containing information from the other views as well. The predictor part of the single image model is then applied to each pyramid. By sharing weights between models applied to each of the images, the network is forced to reuse features from all views, which are visually similar. Explicit view fusion is also performed by aligning collateral mammograms, with the goal of improving detection performance. Experiments show the promise of these methods, but further tests are required to fully validate their effectiveness and compare them to the state of the art.

Although our techniques take into account all four views of a typical mammographic exam, some exams or datasets may contain more or fewer images. In some cases, the patient has had a mastectomy or one or more images are unavailable or were uncaptured. In others, extra images may have been captured in the same or different views. Frequently, multiple exams of the same patient are recorded over time, which radiologists compare in their decision-making process. Future work might include extending the proposed architecture by applying convolutional recurrent networks to construct feature pyramids from a variable number of inputs.

Another challenge in deploying a deep learning based solution on a large scale would be dealing with differences in images that originate from the variety of equipment that might be used to capture them. In addition to allowing the models to have better performance when trained on multiple different datasets, it could also significantly increase its ability to generalize to new, unseen conditions or equipment, for which less training data would then be required. The proposed architecture might be extended by adding Domain-Adversarial Networks, which would encourage the model to produce features which are indistinguishable between data sources.

# References

[1] Deeplearning4j: Open-source, distributed deep learning for the JVM. URL http://deeplearning4j.org/.

[2] The OpenCV Reference Manual, apr 2014.

[3] Breast Cancer Treatment and Pregnancy, 2017. URL https://www.cancer.gov/types/breast/patient/pregnancy-breast-treatment-pdq.

[4] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, and Google Brain. TensorFlow: A System for Large-Scale Machine Learning TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, pages 265–284, 2016. ISBN 978-1-931971-33-1. doi: 10.1038/nn.3331. URL https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.

[5] Chul Kyun Ahn, Changyong Heo, Heongmin Jin, and Jong Hyo Kim. A novel deep learning-based approach to high accuracy breast density estimation in digital mammography. In Samuel G. Armato and Nicholas A. Petrick, editors, *SPIE Medical Imaging*, volume 10134, page 101342O, mar 2017. doi: 10.1117/12.2254264. URL http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2254264.

[6] Ayelet Akselrod-Ballin, Leonid Karlinsky, Sharon Alpert, Sharbell Hasoul, Rami Ben-Ari, and Ella Barkan. A Region Based Convolutional Network for Tumor Detection and Classification in Breast Mammography. In *MICCAI 2016 DL workshop*, volume 10008, pages 197–205. 2016. ISBN 978-3-319-46975-1. doi: 10.1007/978-3-319-46976-8_21. URL http://link.springer.com/10.1007/978-3-319-46976-8{_}21http://arxiv.org/abs/1704.06040{%}0Ahttp://link.springer.com/10.1007/978-3-319-46976-8{%}5Cnhttp://link.springer.com/10.1007/978-3-319-46976-8{_}21.

[7] Natalia Antropova, Benjamin Q. Huynh, and Maryellen L. Giger. A deep feature fusion methodology for breast cancer diagnosis demonstrated on three imaging modality datasets. *Medical Physics*, 44(10):5162–5171, oct 2017. ISSN 00942405. doi: 10.1002/mp.12453. URL http://doi.wiley.com/10.1002/mp.12453.

[8] John Arevalo, Fabio A González, Raúl Ramos-Pollán, Jose L Oliveira, and Miguel Angel Guevara Lopez. Representation learning for mammography mass lesion classification with

convolutional neural networks. *Computer Methods and Programs in Biomedicine*, 127: 248–257, apr 2016. ISSN 01692607. doi: 10.1016/j.cmpb.2015.12.014. URL https://ac.els-cdn.com/S0169260715300110/1-s2.0-S0169260715300110-main.pdf?{_}tid=8312aae0-009a-11e8-954d-00000aacb35e{&}acdnat=1516752606{_}ec9b87701da9bb0dc50169f1c8dfeadehttp://linkinghub.elsevier.com/retrieve/pii/S0169260715300110.

[9] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. 2017. ISSN 1701.07875. URL https://arxiv.org/pdf/1701.07875.pdfhttp://arxiv.org/abs/1701.07875.

[10] R E Bird, T W Wallace, and B C Yankaskas. Analysis of cancers missed at screening mammography. *Radiology*, 184(3):613–617, sep 1992. ISSN 0033-8419. doi: 10.1148/radiology.184.3.1509041. URL http://pubs.rsna.org/doi/10.1148/radiology.184.3.1509041.

[11] R G Blanks, M G Wallis, and S M Moss. A comparison of cancer detection rates achieved by breast cancer screening programmes by number of readers, for one and two view mammography: results from the UK National Health Service breast screening programme. *J Med Screen*, 5:195–201, 1998. URL http://journals.sagepub.com/doi/pdf/10.1136/jms.5.4.195.

[12] Jaime S Cardoso, Nuno Marques, Neeraj Dhungel, Gustavo Carneiro, and Andrew Bradley. Mass Segmentation in Mammograms: a Cross-Sensor comparison of deep and tailored features. *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 1737–1741, 2017. URL http://www.inescporto.pt/{~}jsc/publications/conferences/2017JaimeICIP.pdf.

[13] Gustavo Carneiro, Jacinto Nascimento, and Andrew P Bradley. Unregistered Multiview Mammogram Analysis with Pre-trained Deep Learning Models. In Nassir Navab, Joachim Hornegger, William M Wells, and Alejandro F Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 652–660, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

[14] Eduardo Meca Castro. *Rotated Filters and Learning Strategies in Convolutional Neural Networks for Mammographic Lesions Detection*. PhD thesis, Universidade do Porto, 2017.

[15] Kumar Chellapilla, Sidd Puri, and Patrice Simard. High Performance Convolutional Neural Networks for Document Processing. In Guy Lorette, editor, *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule (France), oct 2006. Université de Rennes 1, Suvisoft. URL https://hal.inria.fr/inria-00112631.

[16] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. 2015. ISSN 03600300. doi: 10.1145/2532637. URL https://arxiv.org/pdf/1512.01274.pdfhttp://arxiv.org/abs/1512.01274.

[17] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. 2016. ISSN 1063-6919. doi: 10.1109/CVPR.2017.195. URL https://arxiv.org/pdf/1610.02357.pdfhttp://arxiv.org/abs/1610.02357.

[18] François Chollet and Others. Keras. https://github.com/keras-team/keras, 2015.

[19] Stefano Ciatto, Marco Rosselli Del Turco, Gabriella Risso, Sandra Catarzi, Rita Bonardi, Valeria Viterbo, Pierangela Gnutti, Barbara Guglielmoni, Lelio Pinelli, Anna Pandiscia, Francesco Navarra, Adele Lauria, Rosa Palmiero, and Pietro Luigi Indovina. Comparison of standard reading and computer aided detection (CAD) on a national proficiency test of screening mammography. *European Journal of Radiology*, 45(2):135–138, feb 2003. ISSN 0720048X. doi: 10.1016/S0720-048X(02)00011-6. URL http://linkinghub.elsevier.com/retrieve/pii/S0720048X02000116.

[20] Dan C. Cireşan, Ueli Meier, Jonathan Masci, Luca M Gambardella, and Jürgen Schmidhuber. High-Performance Neural Networks for Visual Object Classification. *Advances In Neural Information Processing Systems*, page 12, 2011. ISSN 10495258. doi: http://arxiv.org/abs/1102.0183. URL https://arxiv.org/pdf/1102.0183.pdfhttp://arxiv.org/abs/1102.0183.

[21] Ronan Collobert, Samy Bengio, and Johnny Marithoz. Torch: A Modular Machine Learning Software Library, 2002.

[22] M. Daszykowski and B. Walczak. Density-Based Clustering Methods. In *Comprehensive Chemometrics*, volume 2, pages 635–654. 2010. ISBN 9780444527011. doi: 10.1016/B978-044452701-1.00067-3. URL https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf.

[23] Neeraj Dhungel, Gustavo Carneiro, and Andrew P. Bradley. Automated Mass Detection in Mammograms Using Cascaded Deep Learning and Random Forests. In *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8. IEEE, nov 2015. ISBN 978-1-4673-6795-0. doi: 10.1109/DICTA.2015.7371234. URL http://ieeexplore.ieee.org/document/7371234/.

[24] Neeraj Dhungel, Gustavo Carneiro, and Andrew P. Bradley. Deep Learning and Structured Prediction for the Segmentation of Mass in Mammograms. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9349, pages 605–612. 2015. ISBN 978-3-319-24552-2. doi: 10.1007/978-3-319-24553-9_74. URL http://link.springer.com/10.1007/978-3-319-24553-9{_}74.

[25] Neeraj Dhungel, Gustavo Carneiro, and Andrew P. Bradley. The Automated Learning of Deep Features for Breast Mass Classification from Mammograms. In Gustavo Carneiro, Diana Mateus, Loïc Peter, Andrew Bradley, João Manuel R S Tavares, Vasileios Belagiannis, João Paulo Papa, Jacinto C Nascimento, Marco Loog, Zhi Lu, Jaime S Cardoso, and Julien Cornebise, editors, *Deep Learning and Data Labeling for Medical Applications*, pages 106–114. Springer International Publishing, Cham, 2016. ISBN 978-3-319-46976-8. doi: 10.1007/978-3-319-46723-8_13. URL http://link.springer.com/10.1007/978-3-319-46723-8{_}13.

[26] Anastasia Dubrovina, Pavel Kisilev, Boris Ginsburg, Sharbell Hashoul, and Ron Kimmel. Computational Mammography using Deep Neural Networks. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 1163(October):1–5, 2016. ISSN 2168-1163. doi: http://dx.doi.org/10.1080/21681163.2015.1131197Downloaded. URL http://web.

stanford.edu/{~}adkarni/publications/DKGHK{_}DLMIA2015.pdfhttp://dx.doi.org/10.1080/21681163.2015.1131197.

[27] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. mar 2016. URL https://arxiv.org/pdf/1603.07285.pdfhttp://arxiv.org/abs/1603.07285.

[28] J. L. Durrieu, J. Ph Thiran, and F Kelly. Lower and upper bounds for approximation of the Kullback-Leibler divergence between Gaussian mixture models. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 4833–4836, 2012. ISBN 9781467300469. doi: 10.1109/ICASSP.2012.6289001. URL https://infoscience.epfl.ch/record/174055/files/durrieuThiranKelly{_}kldiv{_}icassp2012{_}R1.pdf.

[29] Fizyr. fizyr/keras-retinanet. URL https://github.com/fizyr/keras-retinanet/.

[30] Pablo Fonseca, Julio Mendoza, Jacques Wainer, Jose Ferrer, Joseph Pinto, Jorge Guerrero, and Benjamin Castaneda. Automatic breast density classification using a convolutional neural network architecture search procedure. volume 9414, page 941428, mar 2015. ISBN 9781628415049. doi: 10.1117/12.2081576. URL http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2081576.

[31] Pablo Fonseca, Benjamin Castañeda, Ricardo Valenzuela, and Jacques Wainer. Breast Density Classification with Convolutional Neural Networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10125 LNCS, pages 101–108. 2017. ISBN 9783319522760. doi: 10.1007/978-3-319-52277-7_13. URL https://arxiv.org/pdf/1711.03674.pdfhttp://link.springer.com/10.1007/978-3-319-52277-7{_}13.

[32] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C. Berg. DSSD : Deconvolutional Single Shot Detector. jan 2017. URL http://arxiv.org/abs/1701.06659.

[33] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, apr 1980. ISSN 0340-1200. doi: 10.1007/BF00344251. URL https://doi.org/10.1007/BF00344251http://link.springer.com/10.1007/BF00344251.

[34] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Victor Lempitsky, Urun Dogan, Marius Kloft, Francesco Orabona, and Tatiana Tommasi. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17:1–35, 2016. ISSN 1475-7516. doi: 10.1088/1475-7516/2015/08/013. URL https://arxiv.org/pdf/1505.07818.pdf.

[35] Al Gharakhanian. Generative Adversarial Networks – Hot Topic in Machine Learning. URL https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html.

[36] Ross Girshick. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, volume 2015 Inter, pages 1440–1448. IEEE, dec 2015. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.169. URL http://ieeexplore.ieee.org/document/7410526/.

[37] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587. IEEE, jun 2014. ISBN 978-1-4799-5118-5. doi: 10.1109/CVPR.2014.81. URL http://ieeexplore.ieee.org/document/6909475/.

[38] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *PMLR*, 9:249–256, 2010. ISSN 15324435. doi: 10.1.1.207.2059. URL http://machinelearning.wustl.edu/mlpapers/paper{_}files/AISTATS2010{_}GlorotB10.pdfhttp://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf.

[39] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. pages 1–9, jun 2014. ISSN 10495258. doi: 10.1001/jamainternmed.2016.8245. URL http://arxiv.org/abs/1406.2661.

[40] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved Training of Wasserstein GANs. mar 2017. URL https://arxiv.org/pdf/1704.00028.pdfhttp://arxiv.org/abs/1704.00028.

[41] Karimollah Hajian-Tilaki. Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation. *Caspian J Intern Med*, 4(2):627–635, 2013. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3755824/pdf/cjim-4-627.pdf.

[42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, volume 2015 Inter, pages 1026–1034. IEEE, dec 2015. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.123. URL https://www.cv-foundation.org/openaccess/content{_}iccv{_}2015/papers/He{_}Delving{_}Deep{_}into{_}ICCV{_}2015{_}paper.pdfhttp://ieeexplore.ieee.org/document/7410480/.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, jun 2016. ISBN 978-1-4673-8851-1. doi: 10.1109/CVPR.2016.90. URL https://www.cv-foundation.org/openaccess/content{_}cvpr{_}2016/papers/He{_}Deep{_}Residual{_}Learning{_}CVPR{_}2016{_}paper.pdfhttp://ieeexplore.ieee.org/document/7780459/.

[44] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B Girshick. Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. URL http://arxiv.org/abs/1703.06870.

[45] M Heath, K Bowyer, D Kopans, R Moore, and P Kegelmeyer. The digital database for screening mammography. *Proceedings of the Fifth International Workshop on Digital Mammography*, pages 212–218, 2001. ISSN 1381-6446. doi: 10.1007/978-94-011-5318-8_75. URL http://www.nd.edu/{~}kwb/HeathEtAlIWDM{_}2000.pdf{%}5Cnpapers2://publication/uuid/37C3C3F3-5994-4145-AB4B-5B1CC5381282http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.442.8879{&}rep=rep1{&}type=pdf.

[46] Paul Henderson and Vittorio Ferrari. End-to-end training of object class detectors for mean average precision. URL https://arxiv.org/pdf/1607.03476.pdf.

[47] Josef Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. *Master's thesis, Institut fur Informatik, Technische Universitat, Munchen*, pages 1–71, 1991. URL http://people.idsia.ch/{~}juergen/SeppHochreiter1991ThesisAdvisorSchmidhuber.pdfhttp://www.bioinf.jku.at/publications/older/3804{_}2.pdf{%}0Ahttp://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Untersuchungen+zu+dynamischen+neuronalen+Netzen{#}0.

[48] Sepp Hochreiter, Yoshua Bengio, and Paolo Frasconi. Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies. In *A Field Guide to Dynamical Recurrent Networks*. IEEE press, 2009. ISBN 978-0-7803-5369-5. doi: 10.1109/9780470544037.ch14. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.7321{&}rep=rep1{&}type=pdfhttp://ieeexplore.ieee.org/search/srchabstract.jsp?arnumber=5264952.

[49] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. page 9, apr 2017. doi: arXiv:1704.04861. URL https://arxiv.org/pdf/1704.04861.pdfhttp://arxiv.org/abs/1704.04861.

[50] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-Excitation Networks. sep 2017. URL https://arxiv.org/pdf/1709.01507.pdfhttp://arxiv.org/abs/1709.01507.

[51] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot Ensembles: Train 1, get M for free. mar 2017. URL http://arxiv.org/abs/1704.00109.

[52] D H HUBEL and T N WIESEL. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(38):106–54, jan 1962. ISSN 0022-3751. doi: 10.1523/JNEUROSCI.1991-09.2009. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1359523/pdf/jphysiol01247-0121.pdfhttp://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.1991-09.2009http://www.ncbi.nlm.nih.gov/pubmed/14449617http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC135952.

[53] John D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. ISSN 1521-9615. doi: 10.1109/MCSE.2007.55. URL http://ieeexplore.ieee.org/document/4160265/.

[54] Benjamin Q. Huynh, Hui Li, and Maryellen L. Giger. Digital mammographic tumor classification using transfer learning from deep convolutional neural networks. *Journal of Medical Imaging*, 3(3):034501, aug 2016. ISSN 2329-4302. doi: 10.1117/1.JMI.3.3.034501. URL http://medicalimaging.spiedigitallibrary.org/article.aspx?doi=10.1117/1.JMI.3.3.034501{%}5Cnhttp://www.ncbi.nlm.nih.gov/pubmed/27610399{%}5Cnhttp://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4992049https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4992.

[55] Sangheum Hwang and Hyo-Eun Kim. Self-Transfer Learning for Weakly Supervised Lesion Localization. pages 239–246. 2016. ISBN 9783319467221. doi: 10.1007/978-3-319-46723-8_28. URL http://arxiv.org/abs/1602.01625https://arxiv.org/pdf/1602.01625.pdfhttp://link.springer.com/10.1007/978-3-319-46723-8{_}28.

[56] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. 9349, feb 2016. ISSN 0302-9743. doi: 10.1007/978-3-319-24553-9. URL https://arxiv.org/pdf/1602.07360.pdfhttp://arxiv.org/abs/1602.07360http://link.springer.com/10.1007/978-3-319-24553-9.

[57] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. feb 2015. ISSN 0717-6163. doi: 10.1007/s13398-014-0173-7.2. URL http://arxiv.org/abs/1502.03167.

[58] Itseez. Open Source Computer Vision Library. \url{https://github.com/itseez/opencv}, 2015.

[59] Ahmedin Jemal, Elizabeth M. Ward, Christopher J. Johnson, Kathleen A. Cronin, Jiemin Ma, A. Blythe Ryerson, Angela Mariotto, Andrew J. Lake, Reda Wilson, Recinda L. Sherman, Robert N. Anderson, S. Jane Henley, Betsy A. Kohler, Lynne Penberthy, Eric J. Feuer, and Hannah K. Weir. Annual Report to the Nation on the Status of Cancer, 1975–2014, Featuring Survival. *JNCI: Journal of the National Cancer Institute*, 109(9), sep 2017. ISSN 0027-8874. doi: 10.1093/jnci/djx030. URL http://academic.oup.com/jnci/article/doi/10.1093/jnci/djx030/3092246/Annual-Report-to-the-Nation-on-the-Status-of.

[60] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 675–678, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3063-3. doi: 10.1145/2647868.2654889. URL http://doi.acm.org/10.1145/2647868.2654889.

[61] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. ISBN 978-1-4244-3992-8. doi: 10.1109/CVPRW.2009.5206848. URL http://www.image-net.org/papers/imagenet{_}cvpr09.pdfhttp://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5206848.

[62] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, and Mingli Song. Neural Style Transfer: A Review. may 2017. URL https://arxiv.org/pdf/1705.04058.pdfhttp://arxiv.org/abs/1705.04058.

[63] Michiel Kallenberg, Kersten Petersen, Mads Nielsen, Andrew Y. Ng, Pengfei Diao, Christian Igel, Celine M. Vachon, Katharina Holland, Rikke Rass Winkel, Nico Karssemeijer, and Martin Lillholm. Unsupervised Deep Learning Applied to Breast Density Segmentation and Mammographic Risk Scoring. *IEEE Transactions on Medical Imaging*, 35 (5):1322–1331, may 2016. ISSN 0278-0062. doi: 10.1109/TMI.2016.2532122. URL http://ieeexplore.ieee.org/document/7412749/.

[64] Andrej Karpathy. What I learned from competing against a ConvNet on ImageNet, sep . URL http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/.

[65] Nico Karssemeijer, Johannes D. Otten, Antonius A. J. Roelofs, Sander van Woudenberg, and Jan H. C. L. Hendriks. Effect of independent multiple reading of mammograms on detection performance. page 82, may 2004. doi: 10.1117/12.535225. URL http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.535225.

[66] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. pages 1–15, dec 2014. ISSN 09252312. doi: http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503. URL http://arxiv.org/abs/1412.6980.

[67] Pavel Kisilev, Eli Sason, Ella Barkan, and Sharbell Hashoul. Medical image description using multi-task-loss CNN. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10008 LNCS, pages 121–129. 2016. ISBN 9783319469751. doi: 10.1007/978-3-319-46976-8_13. URL http://arxiv.org/abs/1704.06040http://link.springer.com/10.1007/978-3-319-46976-8{_}13.

[68] Thijs Kooi and Nico Karssemeijer. Classifying Symmetrical Differences and Temporal Change in Mammography Using Deep Neural Networks. mar 2017. ISSN 2329-4302. doi: 10.1117/1.JMI.4.4.044501. URL https://arxiv.org/pdf/1703.07715.pdfhttp://arxiv.org/abs/1703.07715.

[69] Thijs Kooi, Geert Litjens, Bram van Ginneken, Albert Gubern-Mérida, Clara I. Sánchez, Ritse Mann, Ard den Heeten, and Nico Karssemeijer. Large scale deep learning for computer aided detection of mammographic lesions. *Medical Image Analysis*, 35:303–312, jan 2017. ISSN 13618415. doi: 10.1016/j.media.2016.07.007. URL https://ac.els-cdn.com/S1361841516301244/1-s2.0-S1361841516301244-main.pdf?{_}tid=3081100c-0084-11e8-9b74-00000aacb361{&}acdnat=1516743019{_}207fa5941e6ca7a45d4ca6417ff6f2f1http://linkinghub.elsevier.com/retrieve/pii/S1361841516301244.

[70] Thijs Kooi, Bram van Ginneken, Nico Karssemeijer, and Ard den Heeten. Discriminating solitary cysts from soft tissue lesions in mammography using a pretrained deep convolutional neural network. *Medical Physics*, 44(3):1017–1027, mar 2017. ISSN 00942405. doi: 10.1002/mp.12110. URL http://doi.wiley.com/10.1002/mp.12110.

[71] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012. ISSN 10495258. doi: http://dx.doi.org/10.1016/j.protcy.2014.09.007.

[72] Y LeCun, B Boser, J S Denker, D Henderson, R E Howard, W Hubbard, and L D Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, dec 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541. URL http://dx.doi.org/10.1162/neco.1989.1.4.541http://www.mitpressjournals.org/doi/10.1162/neco.1989.1.4.541.

[73] Yann LeCun, L Bottou, Yoshua Bengio, and P Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. ISSN 00189219. doi: 10.1109/5.726791. URL http://vision.stanford.edu/cs598{_}spring07/papers/Lecun98.pdfhttp://ieeexplore.ieee.org/document/726791/.

[74] Jiyon Lee. Practical and illustrated summary of updated BI-RADS for ultrasonography. *Ultrasonography*, 36(1):71–81, jan 2017. ISSN 2288-5919. doi: 10.14366/usg.16034. URL http://e-ultrasonography.org/journal/view.php?doi=10.14366/usg.16034.

[75] Rebecca Sawyer Lee, Francisco Gimenez, Assaf Hoogi, Kanae Kawai Miyake, Mia Gorovoy, and Daniel L Rubin. A curated mammography data set for use in computer-aided detection and diagnosis research. *Scientific Data*, 4:170177, dec 2017. ISSN 2052-4463. doi: 10.1038/sdata.2017.177. URL https://www.nature.com/articles/sdata2017177.pdfhttp://www.nature.com/articles/sdata2017177.

[76] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart Augmentation Learning an Optimal Data Augmentation Strategy. *IEEE Access*, 5:5858–5869, 2017. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2696121. URL http://ieeexplore.ieee.org/document/7906545/.

[77] Daniel Lévy and Arzav Jain. Breast Mass Classification from Mammograms using Deep Convolutional Neural Networks. (Nips), dec 2016. URL http://arxiv.org/abs/1612.00542.

[78] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. dec 2016. URL https://arxiv.org/pdf/1612.03144.pdfhttp://arxiv.org/abs/1612.03144.

[79] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. 2017. ISSN 00392499. doi: 10.1016/j.ajodo.2005.02.022. URL https://arxiv.org/pdf/1708.02002.pdfhttp://arxiv.org/abs/1708.02002.

[80] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single Shot MultiBox Detector. pages 21–37. 2016. doi: 10.1007/978-3-319-46448-0_2. URL https://arxiv.org/pdf/1512.02325.pdfhttp://link.springer.com/10.1007/978-3-319-46448-0{_}2.

[81] Miguel A Guevara Lopez, Naimy Gonzalez de Posada, Daniel C Moura, Raul Ramos Pollan, Jose M Franco Valiente, Cesar Suarez Ortega, Manuel R del Solar, Guillermo Diaz-Herrero, Isabel M A Pereira Ramos, Joana Pinheiro Loureiro, Teresa Cardoso Fernandes, and Bruno M Ferreira de Araujo. BCDR: a breast cancer digital repository. *15th International Conference on Experimental Mechanics ({ICEM15})*, 1:1–5, 2012. URL http://paginas.fe.up.pt/clme/icem15/ICEM15{_}CD/data/papers/3004.pdf.

[82] William Lotter, Greg Sorensen, and David Cox. A Multi-scale CNN and Curriculum Learning Strategy for Mammogram Classification. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10553 LNCS, pages 169–177. jul 2017. ISBN 9783319675572. doi: 10.1007/978-3-319-67558-9_20. URL http://link.springer.com/10.1007/978-3-319-67558-9{_}20.

[83] Zhi Lu, Gustavo Carneiro, Neeraj Dhungel, and Andrew P. Bradley. Automated Detection of Individual Micro-calcifications from Mammograms using a Multi-stage Cascade Approach. 2016. URL http://arxiv.org/abs/1610.02251.

[84] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, dec 1943. ISSN 0007-4985. doi: 10.1007/BF02478259. URL https://pdfs.semanticscholar.org/891e/61c55b49dc55e95c4ed1803cd0801df02d00.pdfhttp://link.springer.com/10.1007/BF02478259.

[85] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51–56, 2010.

[86] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

[87] Jan-Jurre Mordang, Tim Janssen, Alessandro Bria, Thijs Kooi, Albert Gubern-Mérida, and Nico Karssemeijer. *Automatic Microcalcification Detection in Multi-vendor Mammography Using Convolutional Neural Networks*, volume 9699 of *Lecture Notes in Computer Science*. Springer International Publishing, Cham, 2016. ISBN 978-3-319-41545-1. doi: 10.1007/978-3-319-41546-8_5. URL http://link.springer.com/10.1007/978-3-319-41546-8http://link.springer.com/10.1007/978-3-319-41546-8{_}5.

[88] Inês C. Moreira, Igor Amaral, Inês Domingues, António Cardoso, Maria João Cardoso, and Jaime S. Cardoso. INbreast. *Academic Radiology*, 19(2):236–248, feb 2012. ISSN 10766332. doi: 10.1016/j.acra.2011.09.014. URL http://linkinghub.elsevier.com/retrieve/pii/S107663321100451X.

[89] Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. DyNet: The Dynamic Neural Network Toolkit. 2017. URL http://github.com/clab/dynet.http://arxiv.org/abs/1701.03980.

[90] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable Parallel Programming with CUDA. *Queue*, 6(2):40–53, mar 2008. ISSN 1542-7730. doi: 10.1145/1365490.1365500. URL http://doi.acm.org/10.1145/1365490.1365500.

[91] Adam Paszke, Gregory Chanan, Zeming Lin, Sam Gross, Edward Yang, Luca Antiga, and Zachary Devito. Automatic differentiation in PyTorch. (Nips):1–4, 2017.

[92] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011. ISSN 1532-4435. doi: 10.1007/s13398-014-0173-7.2. URL http://delivery.acm.org/10.1145/2080000/2078195/p2825-pedregosa.pdf?ip=138.246.2.134{&}id=2078195{&}acc=OPEN{&}key=4D4702B0C3E38B35.4D4702B0C3E38B35.4D4702B0C3E38B35.6D218144511F3437{&}{_}{_}acm{_}{_}=1516554979{_}f668ee1cec0d12bc92028094bf2cc565http://dl.acm.org/citation.

[93] F. Pérez and B.E. Granger. IPython: A System for Interactive Scientific Computing Python: An Open and General- Purpose Environment. *Computing in Science and Engineering*, 9(3): 21–29, 2007. ISSN 15219615. doi: doi:10.1109/MCSE.2007.53. URL http://fperez.org/papers/ipython07{_}pe-gr{_}cise.pdfhttp://ipython.org.

[94] Yuchen Qiu, Yunzhi Wang, Shiju Yan, Maxine Tan, Samuel Cheng, Hong Liu, and Bin Zheng. An initial investigation on developing a new method to predict short-term breast cancer risk based on deep learning technology. volume 9785, page 978521, mar 2016. ISBN 9781510600201. doi: 10.1117/12.2216275. URL http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2216275.

[95] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525. IEEE, jul 2017. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.690. URL http://arxiv.org/abs/1612.08242http://ieeexplore.ieee.org/document/8100173/.

[96] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788. IEEE, jun 2016. ISBN 978-1-4673-8851-1. doi: 10.1109/CVPR.2016.91. URL http://arxiv.org/abs/1506.02640http://ieeexplore.ieee.org/document/7780460/.

[97] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, jun 2017. ISSN 0162-8828. doi: 10.1109/TPAMI.2016.2577031. URL http://ieeexplore.ieee.org/document/7485869/.

[98] Dezső Ribli, Anna Horváth, Zsuzsa Unger, Péter Pollner, and István Csabai. Detecting and classifying lesions in mammograms with Deep Learning. pages 1–13, 2017. URL http://arxiv.org/abs/1707.08401.

[99] F Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. ISSN 1939-1471. doi: 10.1037/h0042519. URL http://psycnet.apa.org/journals/rev/65/6/386.pdf{%}5Cnpapers://c53d1644-cd41-40df-912d-ee195b4a4c2b/Paper/p15420http://doi.apa.org/getdoi.cfm?doi=10.1037/h0042519.

[100] Guido Rossum. Python Reference Manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

[101] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning Internal Representations by Error Propagation. *Readings in Cognitive Science*, 1:399–421, 1988. ISSN 1-55860-013-2. doi: 10.1016/B978-1-4832-1446-7.50035-2. URL http://linkinghub.elsevier.com/retrieve/pii/B9781483214467500352.

[102] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, dec 2015. ISSN 0920-5691. doi: 10.1007/s11263-015-0816-y. URL https://link.springer.com/content/pdf/10.1007{%}2Fs11263-015-0816-y.pdfhttp://link.springer.com/10.1007/s11263-015-0816-y.

[103] Berkman Sahiner, Heang-Ping Chan, Nicholas Petrick, Datong Wei, M.A. Helvie, D.D. Adler, and M.M. Goodsitt. Classification of mass and normal breast tissue: a convolution neural network classifier with spatial domain and texture images. *IEEE Transactions on Medical Imaging*, 15(5):598–610, 1996. ISSN 02780062. doi: 10.1109/42.538937. URL http://ieeexplore.ieee.org/document/538937/.

[104] Hojjat Salehinejad, Shahrokh Valaee, Aren Mnatzakanian, Tim Dowdell, Joseph Barfett, and Errol Colak. Interpretation of Mammogram and Chest X-Ray Reports Using Deep Neural Networks - Preliminary Results. aug 2017. URL http://arxiv.org/abs/1708.09254.

[105] Frank Seide and Amit Agarwal. CNTK: Microsoft's Open-Source Deep-Learning Toolkit. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 2135, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2945397. URL http://doi.acm.org/10.1145/2939672.2945397.

[106] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv preprint arXiv*, page 1312.6229, dec 2013. ISSN 10636919. doi: 10.1109/CVPR.2015.7299176. URL https://arxiv.org/pdf/1312.6229.pdfhttp://arxiv.org/abs/1312.6229.

[107] Li Shen. End-to-end Training for Whole Image Breast Cancer Diagnosis using An All Convolutional Design. 3000:1–12, 2017. URL http://arxiv.org/abs/1708.09427.

[108] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000. ISSN 03783758. doi: 10.1016/S0378-3758(00)00115-4.

URL www.elsevier.com/locate/jspihttp://linkinghub.elsevier.com/retrieve/pii/S0378375800001154.

[109] Hoo-Chang Shin, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M. Summers. Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning. *IEEE Transactions on Medical Imaging*, 35(5):1285–1298, may 2016. ISSN 0278-0062. doi: 10.1109/TMI.2016.2528162. URL http://ieeexplore.ieee.org/document/7404017/.

[110] Rebecca L. Siegel, Kimberly D. Miller, and Ahmedin Jemal. Cancer statistics, 2017. *CA: A Cancer Journal for Clinicians*, 67(1):7–30, jan 2017. ISSN 00079235. doi: 10.3322/caac.21387. URL http://doi.wiley.com/10.3322/CA.2007.0010http://doi.wiley.com/10.3322/caac.21387.

[111] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550:354, oct 2017. URL http://dx.doi.org/10.1038/nature24270http://10.0.4.14/nature24270https://www.nature.com/articles/nature24270{#}supplementary-information.

[112] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf.

[113] Dave Steinkraus, Ian Buck, and P.Y. Simard. Using GPUs for machine learning algorithms. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 1115–1120 Vol. 2. IEEE, 2005. ISBN 0-7695-2420-6. doi: 10.1109/ICDAR.2005.251. URL http://ieeexplore.ieee.org/document/1575717/.

[114] Parker J Dance D Astley S Hutt I Boggis C Ricketts I Suckling J. Mammographic Image Analysis Society (MIAS) database v1.21, 2015. URL https://www.repository.cam.ac.uk/handle/1810/250394.

[115] Wenqing Sun, Tzu-Liang (Bill) Tseng, Jianying Zhang, and Wei Qian. Enhancing deep convolutional neural network scheme for breast cancer diagnosis with unlabeled data. *Computerized Medical Imaging and Graphics*, 57:4–9, apr 2017. ISSN 08956111. doi: 10.1016/j.compmedimag.2016.07.004. URL https://ac.els-cdn.com/S0895611116300696/1-s2.0-S0895611116300696-main.pdf?{_}tid=4858092c-009a-11e8-b135-00000aab0f26{&}acdnat=1516752516{_}92aced2eb096aeedfa76b458a87392a6http://linkinghub.elsevier.com/retrieve/pii/S0895611116300696.

[116] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, (2010):8609–8613, 2013. ISSN 15206149. doi: 10.1109/ICASSP.2013.6639346. URL http://www.cs.toronto.edu/{~}fritz/absps/momentum.pdf.

[117] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. 2014. URL https://www.cv-foundation.org/openaccess/content{_}cvpr{_}2015/papers/Szegedy{_}Going{_}Deeper{_}With{_}2015{_}CVPR{_}paper.pdf.

[118] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. 2015. ISSN 08866236. doi: 10.1109/CVPR.2016.308. URL https://arxiv.org/pdf/1512.00567.pdfhttp://arxiv.org/abs/1512.00567.

[119] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. 2016. ISSN 01678655. doi: 10.1016/j.patrec.2014.01.008. URL https://arxiv.org/pdf/1602.07261.pdfhttp://arxiv.org/abs/1602.07261.

[120] Luke Taylor and Geoff Nitschke. Improving Deep Learning using Generic Data Augmentation. *arXiv preprint arXiv:1708.06020*, aug 2017. URL https://arxiv.org/pdf/1708.06020.pdfhttp://arxiv.org/abs/1708.06020.

[121] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a Next-Generation Open Source Framework for Deep Learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015. URL http://learningsys.org/papers/LearningSys{_}2015{_}paper{_}33.pdf.

[122] J R R Uijlings, K. E. A. van de Sande, T Gevers, and A W M Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2): 154–171, sep 2013. ISSN 0920-5691. doi: 10.1007/s11263-013-0620-5. URL http://www.huppelen.nl/publications/selectiveSearchDraft.pdfhttps://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013/UijlingsIJCV2013.pdfhttp://link.springer.com/10.1007/s11263-013-0620-5.

[123] S van der Walt, S C Colbert, and G Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science Engineering*, 13(2):22–30, mar 2011. ISSN 1521-9615. doi: 10.1109/MCSE.2011.37.

[124] Juan Wang, Huanjun Ding, Fatemeh Azamian Bidgoli, Brian Zhou, Carlos Iribarren, Sabee Molloi, and Pierre Baldi. Detecting Cardiovascular Disease from Mammograms With Deep Learning. *IEEE Transactions on Medical Imaging*, 36(5):1172–1181, may 2017. ISSN 0278-0062. doi: 10.1109/TMI.2017.2655486. URL http://ieeexplore.ieee.org/document/7827150/.

[125] Darvin Yi, Rebecca Lynn Sawyer, David Cohn, Jared Dunnmon, Carson Lam, Xuerong Xiao, and Daniel Rubin. Optimizing and Visualizing Deep Learning for Benign/Malignant Classification in Breast Tumors. (Nips), 2017. URL http://arxiv.org/abs/1705.06362.

[126] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? 2014. ISSN 10495258. URL https://arxiv.org/pdf/1411.1792.pdfhttp://arxiv.org/abs/1411.1792.

[127] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8689 LNCS, pages 818–833. oct 2014. ISBN 978-3-319-10589-5. doi: 10.1007/978-3-319-10590-1_53. URL http://arxiv.org/abs/1610.02251http://link.springer.com/10.1007/978-3-319-10590-1{_}53https://arxiv.org/pdf/1311.2901v3.pdfhttps://arxiv.org/abs/1311.2901v3.

[128] Wentao Zhu, Qi Lou, Yeeleng Scott Vang, and Xiaohui Xie. Deep Multi-instance Networks with Sparse Label Assignment for Whole Mammogram Classification. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10435 LNCS, pages 603–611. 2017. ISBN 9783319661780. doi: 10.1007/978-3-319-66179-7_69. URL http://link.springer.com/10.1007/978-3-319-66179-7{_}69.

[129] Wentao Zhu, Xiang Xiang, Trac D. Tran, Gregory D. Hager, and Xiaohui Xie. Adversarial Deep Structural Networks for Mammographic Mass Segmentation. (2):1–8, 2017. doi: 10.1101/095786. URL http://arxiv.org/abs/1612.05970http://arxiv.org/abs/1710.09288.