Feature Extraction of Mammographic Images using Convolutional Autoencoders

by

Brynjar Sigurðsson

Thesis of 30 ECTS credits submitted to the School of Computer Science
at Reykjavík University in partial fulfillment
of the requirements for the degree of
Master of Science (M.Sc.) in Computer Science

June 2019

Examining Committee:

Yngvi Björnsson, Supervisor
Professor, Reykjavík University, Iceland

Bjarni Halldórsson, Co-Supervisor
Associate Professor, Reykjavík University, Iceland

Magnús Örn Úlfarsson,
Professor, University of Iceland, Iceland

Torfi Þórhallson,
Assistant Professor, Reykjavík University, Iceland

Feature Extraction of Mammographic Images using Convolutional Autoencoders

Brynjar Sigurðsson

June 2019

Abstract

Breast cancer screening is usually done via digital mammography imaging. These screenings are creating large data sets that open up the possibility for further study in medical image analysis. Data sets of images may appear high dimensional when every pixel is considered as a dimension. However, for natural images or images belonging to a given category the signal can usually be represented with much fewer dimensions. In recent years, the advances in low dimensional representation of images can to a large extent be attributed to deep learning based methods. In this study we apply two unsupervised methods for feature extraction on a data set of 4648 mammograms. First we apply stochastic singular value decomposition (SSVD) to extract 200 features. Second, we train a convolutional autoencoder (CAE) with a bottleneck of size 200. With the two representations of the images we do a feature-age correlation study. The CAE model features correlate significantly more with age than those of the SSVD model. Finally, we run the features through a correlation study with roughly 20,000 phenotypes. 190 breast related phenotypes were found to significantly correlate with features from the CAE model, while only 2 from the SSVD model. Our study suggests that CAE models are a good starting point for creating biologically meaningful phenotypes from mammograms.

Acknowledgements

First, I would like to thank deCODE genetics for funding the project, providing me the facility to work at and the tools for doing the research.

Next, I would like to thank my supervisors Dr. Yngvi Björnsson, at Reykjavík University, and Dr. Bjarni Halldórsson who oversaw the project at deCODE. Their feedback and supervision was crucial for teaching me how to do proper scientific research.

Finally, I would like to thank Dr. Guðmundur Einarsson and Dr. Hafsteinn Einarsson for providing invaluable feedback. Their professional comments and friendship aided me greatly in the writing of this thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
BCE	Binary Cross-Entropy
CAE	Convolutional Autoencoder
CNN	Convolutional Neural Network
FIX	Feature Index
GWAS	Genome-Wide Association Study
MSc	Masters of Science
MSE	Mean Square Error
PCA	Principal Component Analysis
PN	Personal Number
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition
SSVD	Stochastic Singular Value Decomposition

Chapter 1

Introduction

This study was done at deCODE genetics, Inc. using a digital mammography image data set obtained through cancer screening from The Icelandic Cancer Society (ICS).

Mammogram screenings are medical X-ray imaging of (usually female) breasts and are generally used for cancer detection. Traditionally during a screening, a radiologist examines the obtained mammograms and determines whether any further actions are required. Routine screenings include four images in total, two of each breast (L and R, for left and right) for two views of each. Bilateral craniocaudal (CC) is a head-to-foot view of the breast and mediolateral oblique (MLO) is an angled side view.

In Iceland, all women aged 40-69 are invited to a yearly breast cancer screening. Data sets from these screenings can be challenging for computers to analyze as useful features need to be extracted from pixel data. There are also important privacy issues to consider; not every kind of research can be conducted on the data, so care must be taken as to how these data sets are used.

In recent years deep neural networks (DNN) have been applied to supervised computer vision tasks with impressive results. This is supported by the ever-growing size of data sets required for deep learning. Because medical screenings are generally applied in search of some specific disease they are often labeled which is required for supervised learning. Dhungel et al. used cascaded deep learning and random forests for automated mass detection in mammograms [1]. Kooi et al. used deep learning for computer aided detection of mammographic lesions [2]. Charan et al. used deep learning for breast cancer detection [3]. Ribli et al. used deep learning for lesion detection and classification [4]. Most of this research has focused on supervised learning, while unsupervised deep learning remains largely an unresolved, but active research topic.

Most medical images are used to screen for a very narrow set of diseases. However, one would expect that there is a wealth of information embedded in these images, unrelated to the medical diagnosis itself. Feature extraction is the process of extracting useful information that describes a higher dimensional data set in much fewer dimensions while retaining as much variance of the original data as possible. For instance, one would expect some features to be heritable while others correlate with age and/or BMI. For mammograms, one feature could be the density of some type of tissue localized to some part of the breast. Other features could be imaging artifacts that show up because of multiple X-ray machines used across the data set. As with most images, medical images are generally very high dimensional data points. They have the property that the data points of the set lie all close to a manifold of much lower dimensionality.

By decreasing the dimensionality of a medical data set, some interesting analysis methods can be applied to it. For instance, a simple linear age correlation can be done. Other more complicated analysis like phenotype correlation or genome-wide association study (GWAS) can be applied for further study. One would expect some features to be more heritable than others for instance, and as such more likely to be genetic and biological and less likely to be artifacts.

The purpose of this project was to do feature extraction of a data set of roughly 4,500 mammograms of different individuals. A (semi) arbitrary number of 200 features was chosen and two different methods applied. First, as a baseline, singular value decomposition (SVD) was used. SVD is a method for performing a linear decomposition of a matrix. The second method used was the training of a convolutional autoencoder, an artificial neural network (ANN) architecture. The autoencoder architecture has an hourglass shape of progressively narrower layers and is composed of two parts, the encoder and decoder. The encoder learns to reduce the dimensionality of the input to create a bottleneck feature vector (number of features is a hyperparameter), while the decoder takes as input this feature vector and learns to reconstruct the original input of the encoder.

The rest of the thesis is organized as follows. Next comes the background chapter, where we present the basis for the two unsupervised dimensionality reduction methods used in this study. We also provide a short description of phenotypes. Following that is the methods chapter, where we describe first the preprocessing of the images, then, the models used for feature extraction, and finally, we describe how the feature study was conducted. The results chapter comes next, where we present the raw uninterpreted results from the methods that were applied. Closing this dissertation is the discussion chapter, where we interpret, summarize and conclude based on the results. We also describe briefly what future work can be done.

Chapter 2

Background

This chapter first describes briefly in high-level terms how PCA and SVD are done, while the rest of the chapter describes the background of autoencoders and the various related neural network parts in the model used in the methods chapter. A brief introduction on phenotypes concludes this chapter.

2.1 PCA and SVD

Principal Component Analysis (PCA) is an orthogonal transformation procedure that reduces a set of possibly correlated variables into a set of linearly uncorrelated ones, called the *principal components*. This is done by finding the vectors whose direction explains the greatest variance in the data. When projecting the data points of the data set onto these vectors we get the value for the respective variable, labeled the *first principal component*, *second principal component* and so on. PCA can be done by eigenvalue decomposition of the (normalized) $n \times m$ matrix of n observations, each of m variables. The eigenvector whose corresponding eigenvalue is the greatest is the first principal component. The number of different principal components is

$$c = min(n - 1, m). \tag{2.1}$$

A common method for doing PCA is by *Singular Value Decomposition* (SVD). SVD is a linear factorization of an $n \times m$ matrix M such that

$$M = U\Sigma V^T \tag{2.2}$$

where U is an $n \times n$ orthogonal matrix of left singular vectors, Σ is an $n \times m$ diagonal matrix of singular values, and V is an $m \times m$ orthogonal matrix of right singular vectors. SVD can be used itself to compute a low-rank approximation of M by truncating the decomposition, that is a matrix M_r with rank $r < m$ such that $||M - M_r||$ is minimized. The factorization then becomes

$$M \approx U_r \Sigma_r V_r^T \tag{2.3}$$

where Σ_r is an $r \times r$ diagonal matrix of the r greatest singular values, and U_r and V_r^T are the $n \times r$ and $r \times m$ orthogonal matrices of corresponding left and right singular vectors, respectively. This factorization can be used to do PCA, with the product $U_r\Sigma_r$ being the first r principal components, or we can simply use the matrix U_r as a lower rank representation of the original data.

The time complexity for PCA by either eigenvalue decomposition or SVD is $O(m^2n + m^3)$ and for truncated SVD is $O(nmr)$ [5]. This can quickly become inhibiting with high dimensional data such as images. *Randomized* or *Stochastic SVD* (SSVD) is a probabilistic implementation of SVD for computing an approximation of the truncated decomposition of a matrix [6] [7]. The time complexity for SSVD is $O(nm\log(r))$.

2.2 Convolutional Autoencoder

Traditional fully connected ANNs have been successfully used for small scale computer vision tasks but due to their densely connected nodes suffer from the curse of dimensionality and do not scale well to high dimensional data. Convolutional neural networks (CNN) [8] have proven to be remarkably efficient for computer vision and have become the de facto method in this field. CNNs use a linear operation called *convolutions* instead of normal matrix multiplication in at least one of the layers which are generally followed by *pooling layers* in addition to the normal activation and batch normalization layers (all described below).

The motivation behind convolutional networks is that we will learn simple (progressively more complex as we forward propagate through the network) but meaningful features such as edges with kernels. Kernels are tensors applied to the input and slide over it searching for a specific feature.

A 2-D convolution $I*K$ (where asterisk is used to denote the convolution operation) consists of an $i_1 \times i_2$ input tensor I and a $k_1 \times k_2$ kernel tensor K (generally $k_1 = k_2 = k$ and we will assume such) which produces an output tensor O of dimensions $o_1 \times o_2$ where

$$o_j = \frac{i_j - k + 2p}{s} + 1, \quad j \in \{1, 2\} \tag{2.4}$$

where s is the *stride* of the convolution (default stride is 1) which specifies by how much the kernel slides or skips over the input. The value p denotes *zero padding* size (default padding is 0; this is the value, not to be confused with the name zero padding) which specifies how zeros are padded to the output tensor, generally used to keep the dimensions of the output the same as the input as convolutions shrink the input tensor size. The convolution operation generally implemented in most neural network libraries (such as `Tensorflow`) is the **cross-correlation** function defined element-wise on the output tensor as

$$O(i, j) = (I * K)(i, j) = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} I(i + m, j + n)K(m, n). \tag{2.5}$$

Intuitively, the kernel slides over the input like a window and outputs a single scalar value calculated by Equation 2.5.

Each convolutional layer generally applies multiple such kernels, also known as *filters*, on the input and outputs equally many *feature maps*, one for each filter. The number of filters in each layer is a design decision and a hyperparameter of the network. In most CNN architectures the number of filters increases with decreasing input sizes.

It has been shown that applying *batch normalization* after each convolutional layer considerably speeds up training and also acts as a regularization method thus decreasing the need of dropout to avoid overfitting [9]. Batch normalization normalizes the output of a layer by subtracting the batch mean and dividing by the batch standard

deviation but scales and shifts the normalized values of a mini batch $\mathcal{B} = \{x_1, ..., x_m\}$ by *learnable parameters* γ and β

$$\hat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \tag{2.6}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \tag{2.7}$$

where $\mu_\mathcal{B}$ and $\sigma_\mathcal{B}^2$ are the batch mean and variance, respectively, and ϵ is a small constant added for numerical stability. The normalized outputs have to be scaled and shifted because stochastic gradient descent (SGD) effectively undoes the normalization if it is a way for decreasing the loss function [10, p. 309-312].

Because convolutions and dense layers' weight matrix multiplication are linear operations, a network employing only those operations cannot model a nonlinear function. Therefore, after each convolutional or dense layer a nonlinear *activation function* is applied to the output. The function **Rectified Linear Unit**, abbreviated as ReLU, has recently been used in favor of the **sigmoid** and **tanh** functions to solve the *vanishing gradient problem* [11]. ReLU is defined as the function $R : \mathbb{R} \to [0, +\infty)$ where

$$R(z) = max(0, z) \tag{2.8}$$

All internal layers generally use this activation function. There is however one exception which is the output layers. For the output of a CNN the sigmoid function $S : \mathbb{R} \to (0, 1)$ is often used, defined as

$$S(z) = \frac{1}{1 + e^{-z}} \tag{2.9}$$

Pooling layers are commonly used after convolutional layers. *Max pooling* [12] operation outputs the maximum value in a neighbourhood of given pool size for an input tensor. A common size for max pooling on a 2-D input is 2×2, resulting in an output tensor whose size is 4 times smaller than the input (because we divide the input tensor into tiles of size 2×2 and reduce their size to 1×1). Pooling helps to make representations approximately invariant to small translations in the input [10]. This means that the network will better detect whether a feature is present rather than exactly where it is. The commonly named AlexNet, created by Krizhevsky, Sutskever and Hinton for the ImageNet ILSVRC challenge in 2012, showed that better performance can be achieved by stacking convolutional layers (followed by batch normalization and activation) followed by a single pooling layer [13]. The size of an output of a pooling layer for input size i, kernel size k and stride s is

$$o = \left\lfloor \frac{i - k}{s} \right\rfloor + 1 \tag{2.10}$$

Convolutional networks reduce the size of the input as it propagates forward through convolutional and pooling layers. If the purpose of the network is to construct an output of the same size or larger than the input (e.g. upscaling low-resolution image to higher resolution) we must use *upsampling layers*. Upsampling can be done by interpolation (nearest neighbourhood, bi-linear, etc.) but we can have the network learn how to upscale optimally as part of its training by using *transposed convolutions* with learnable kernels like for regular convolutions. A convolution with kernel size k on an

input size $i \times i$ with stride s and padding $p = 0$ has an associated transposed convolution [14] described by kernel size $k' = k$, stride $s' = s$ and padding $p' = 0$ where the output size is

$$o' = s(i - 1) + k. \tag{2.11}$$

By using a kernel size of $k = 2$ and stride $s = 2$ we get the output size

$$o' = s(i - 1) + k = 2(i - 1) + 2 = 2i \tag{2.12}$$

thus doubling the height and the width of input. The transposed convolution operation itself is described by the cross-correlation Equation 2.5.

An *autoencoder* is an unsupervised nerual network architecture that is trained to approximate the identity function [15]. It is composed of two parts, and *encoder* function $f(\mathbf{x}) = \mathbf{h}$ that maps an input vector $\mathbf{x}$ to an encoding $\mathbf{h}$, generally of much lower dimensionality, and a *decoder* which produces a reconstruction $\mathbf{r} = g(\mathbf{h})$. Approximation is a key word here, since an autoencoder that learns to map $g(f(\mathbf{x})) = \mathbf{x}$ everywhere is not useful. Autoencoders are generally restricted to learn a smaller representation than the input so as to be unable to learn the identity mapping perfectly. Since the model is made to prioritize which parts of the input it copies it often learns the most important features.

An autoencoder whose decoder function is linear and loss function is the mean squared error will learn to span the same subspace as PCA. With a nonlinear encoder and decoder, we can learn a more powerful nonlinear generalization of PCA [10].

Convolutional autoencoders (CAE) are simply autoencoders with some layers using the convolution operation instead of matrix multiplication.

2.3 Gradient-Based Optimization and Loss Function

A neural network is a multivariate differentiable function described (in part) by the weights of the network. We can define a real-valued loss function (also known as objective, cost or error function) $f : \mathbb{R}^n \to \mathbb{R}$ that measures how well the network predicts an output compared to the expected output, imposing a greater loss the greater the difference. The loss function chosen is generally differentiable everywhere. The task of minimizing f can be achieved by calculating the gradient of $f(\mathbf{x})$ and moving in the direction of steepest descent

$$\mathbf{x}' = \mathbf{x} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x}) \tag{2.13}$$

where ϵ is the *learning rate*, a hyperparameter to the network. The weights are updated via the *backpropagation* algorithm, changing them in the direction of (local) minimal loss.

A natural loss function for image regression tasks is the typical *mean squared error* (MSE) function. However, another commonly used loss function is the *binary cross-entropy* (BCE) loss, used for two classes; when the data distribution is in $[0, 1]$ as is the case with scaled and normalized pixel values, the pixel intensity can be interpreted as the probability that a certain pixel is "on" [16]. The BCE loss L is the following

$$L = -\sum_{1 \leq i \leq n} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \tag{2.14}$$

where $y = (y_1, y_2, ..., y_n)$ and $\hat{y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_n)$ are two flattened images with values in $[0, 1]$, the predicted output and the expected output.

2.4 Optimization Algorithms

Stochastic Gradient Descent (SGD) is the base optimization algorithm, that uses random batches of training data for updating the network iteratively in a given epoch. Variants and extensions of SGD include *RMSProp* and *Adam*. RMSProp (Root Mean Squared Propagation), introduced by Hinton in an online Coursera course [17], is an SGD variant in which the learning rate is adapted for each of the parameters. Adam (Adaptive Moment Estimation), introduced by Kingma and Lei Ba in 2015 [18], is an update to RMSProp using both averages of the gradients and the second moments of the gradients. Empirical results show that Adam compares favorably to the other stochastic gradient methods.

2.5 Phenotypes

Phenotype is defined as follows by Encyclopædia Britannica

> *"Phenotype, all the observable characteristics of an organism that result from the interaction of its genotype (total genetic inheritance) with the environment. Examples of observable characteristics include behaviour, biochemical properties, colour, shape, and size."* [19]

Phenotypes result from an organism's expression of its genes, as well as effects from its environment. Example phenotypes include height and blood type. The former is a *quantitative* phenotype and the latter a *qualitative* one.

In genetic research, the interaction between phenotypes and genotypes is the main subject of study. We can define new phenotypes and conduct a genome-wide association study (GWAS) to find any associations between this phenotype and the (generally human) genome. Most studies focus on finding the location of Single-Nucleotide Polymorphisms (SNPs) that are most associated with a disease, that is, locations of variations in the genome that increase the risk of a person developing a disease.

Phenotypes are generally given descriptive names followed by the date of the data freeze, e.g. High_Risk_Breast_Cancer_08042019.

Chapter 3

Methods

In here we describe our data set, necessary preprocessing steps, as well as the details of SVD and our autoencoder architecture. All code was written in Python version 3.6.3, R version 3.5.2 or Bash. The major Python libraries used are Pydicom, NumPy, Keras, TensorFlow, SciPy and scikit-learn (`sklearn`).

3.1 Data Preprocessing

The data set consisted of 18,592 `DICOM` files (Digital Imaging and Communications in Medicine, `.dcm` extension). All the data is anonymized with social security numbers (kennitala) replaced by a unique Personal Number (PN) for cross-referencing all data within deCODE. There are four files per PN, both CC and MLO views for each breast, for a total of 4,648 individuals. For this study, the CC view of the right breast was chosen.

For reading the `DICOM` files the Python package `pydicom` was used. The package offers functions for loading data from a given `DICOM` file into a dictionary. The only data and metadata used were the images themselves which are simply `numpy` arrays, the PNs and the dates of study (DOS). The images are all greyscaled and of the same ratio but not all of the same dimensions. In general height and width are at least 2,800 and 2,000 pixels, respectively. All images have the text `R-CC` embedded in the image.

To start with the text embedded in the images was removed by replacing a part of the image with a zero array. It is not strictly necessary to remove this text if it is entirely invariant among the images but since they are not all of the same sizes this was done nevertheless. The images were then automatically cropped such that the height to width ratio would be 1.25 and then rescaled down to 640-by-512 pixels.

For using the BCE loss the pixel values must all be in the range $[0, 1]$. This can be ensured by dividing each pixel value of a given image with 2^n where n is the number of bits used to store the values. The pixel arrays likely store `uint16` type values, but this was not taken for granted. For each image, the n number of bits used to store each value was found and the pixels were divided by 2^n after converting to type `float64`.

Finally the data set of images was normalized using the function `normalize` from `sklearn.preprocessing`.

3.2 Baseline Linear Dimensionality Reduction

Since the time complexity of PCA is $O(m^2n+m^3)$ where in this study's case $n = 4{,}648$ and $m = 640 \cdot 512 = 327{,}680$ a non-parallelized implementation is not applicable in practice. Instead SSVD was done using `sklearn.utils.extmath.randomized_svd` and choosing $r = 200$ to reduce each image from 327,680 dimensions down to 200. On a single CPU core this completed in roughly 30 minutes for the entire data set.

3.3 Convolutional Autoencoder

The convolutional autoencoder was written in Python with the `Keras` library using `Tensorflow-gpu v1.9.0` for compatibility with `Cuda v9.0` and `cuDNN v7.0.5`, which are the versions installed of Cuda and cuDNN on the GPU machine we used for training.

The autoencoder (see subsection 3.3.1 for model description) was trained using a batch size of 16 and the Adam optimizer with a learning rate of $\epsilon = 10^{-4}$. The loss function used is BCE. To start with the data was split into 95% training and 5% validation sets. An early stopping callback function was set up to stop training once validation loss stopped decreasing. No test set was used as there is no intention of applying the autoencoder to extract features in future data sets.

The model was also trained without using any validation. It was observed that training loss converged very quickly after only 44 epochs resulting in early stopping, with the reconstructed image being very blurry. It is questionable whether the model is really learning features capturing adequate variance in the images with this reconstruction. If the model was however trained further without regarding validation loss, the training loss *does not decrease but the observed reconstruction still gets better*. This is counter-intuitive but may be related to the loss function. I was unable to find an explanation and decided to train the model without early stopping for a various number of additional epochs: 100, 250, 500, 750 and 1000 epochs. It is to be expected that at some point the model starts to overfit considerably, resulting in features that do not capture the variance between data points adequately.

3.3.1 Model

The model consists of two major parts, the encoder (see Figure 3.1) which takes as input a 640×512 image (`numpy` array with values in $[0, 1]$), and outputs a 200×1 tensor, and the decoder (see figure 3.2) which takes as input a 200×1 tensor and outputs a 640×512 image. The model is trained as an autoencoder and the encoder part used to extract 200 numerical features thereafter.

Most layers are convolutional except for the bottleneck which consists of two fully connected layers. As the input propagates forward through the encoder, the number of feature maps increases while the image gets smaller, down to 512 feature maps of 10×8 images, just before the bottleneck. The convolution operation `Conv2D` is always applied in pairs (in addition to batch normalization and ReLU activation), followed by max pooling. Propagating forward through the decoder, the number of feature maps is decreased while the image gets larger. The `Conv2DTranspose` operation (with a 2×2 kernel and 2×2 stride) is followed by pairs of the `Conv2D` operation. The model concludes with a Sigmoid activation, for outputting values in the open interval $(0, 1)$.

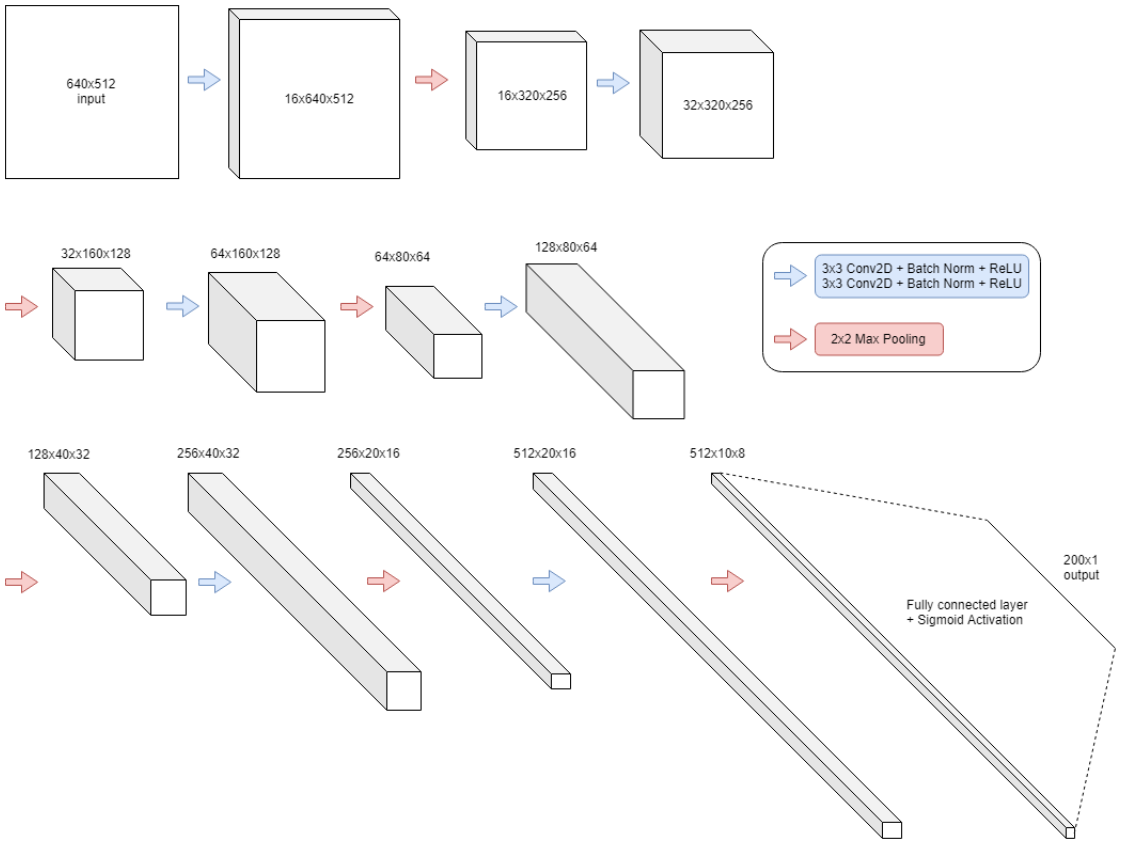

Figure 3.1: Encoder architecture

The model consists of 29,191,577 trainable parameters, with roughly half due to the two fully connected layers.

3.4 Feature Study

Once the model has been trained we extract the encoder part and use it to predict 200 features on the entire data set. With the 200 features per image obtained from SSVD, we can do some comparative empirical studies.

3.4.1 Age Correlation

We construct a vector of ages $age = [\textit{year of study}] - [\textit{year of birth}]$ while preserving the order of PNs in relation to the data set. There were 44 PNs which had no information on year of birth and the corresponding data was removed before calculating any correlation. For each feature column of the $4{,}604 \times 200$ array, the Pearson correlation coefficient between the feature vector and the age vector was calculated using the function `scipy.stats.pearsonr`. The three features with the lowest p-value were recorded and compared.

3.4.2 Feature Tuning

What a particular feature might be measuring is difficult to verify. One possible method for gleaning some information is to take a feature that we suspect measures

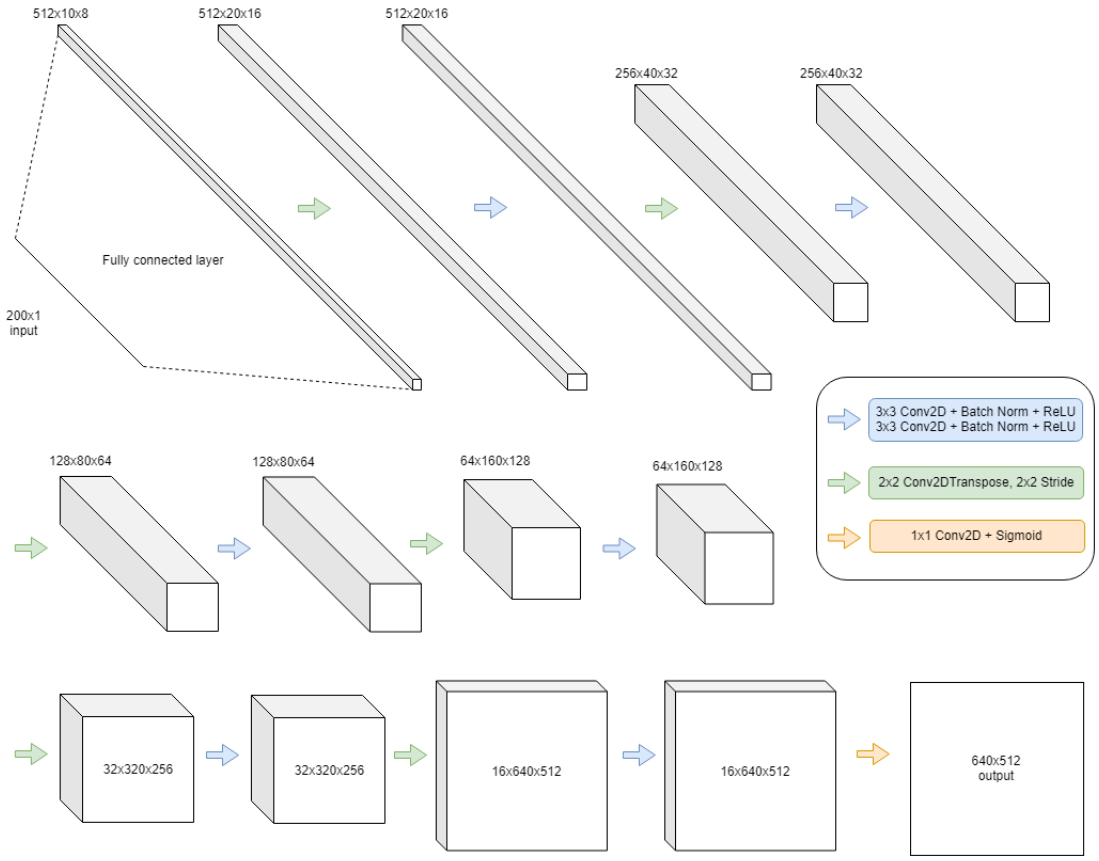

Figure 3.2: Decoder architecture

something biological or heritable and decrease or increase its value. We can then use the decoder to construct an image, and by looking at multiple images possibly argue for what that feature measures.

We took the features that correlate the most with age for a given model (from Section 3.4.1) and adjusted them by a factor of 0.95 and 1.05, respectively, 20 times (while keeping the max value still 1). The decoder was then used to construct the corresponding image.

3.4.3 Features Adjusted for Age

Before doing Phenotype Correlation (Section 3.4.4) we adjust the features for age. First we add minor Gaussian noise to the each column vector $\mathbf{v}$ with mean

$$\mu = 10^{-20} \cdot \min_{1 \leq i \leq 4604} (v_i > 0) \tag{3.1}$$

and standard deviation

$$\sigma = 0.1\mu. \tag{3.2}$$

This is necessary in edge cases if many of the features turn out to be the same value for the phenotype correlation script to run properly. The mean and the standard deviation were chosen arbitrarily. The feature vectors were then adjusted for age and squared age using the `gam` function from the `mgvc` library in R, and the columns then rank transformed using quantile normalization (`qnorm`). This is done twice. This ensures

that the feature vectors are normally distributed and the p-values meaningful for the phenotype correlations.

3.4.4 Phenotype Correlation

With the age-adjusted features, we run a phenotype correlation script on breast related phenotypes and using other phenotypes as controls, for a total of roughly 20,000 phenotypes. We apply a filter on $\alpha \neq 0$, where α is the y-interception of the slope from the regression

$$Y = \alpha + \beta X + \epsilon \tag{3.3}$$

where X is a feature value and Y is the predicted phenotype value. We also filter on the number of PNs from the feature data set intersected with PNs from a given phenotype being greater than 100, and the p-value being less than 10^{-6}. We then order the results by increasing p-values.

A strong correlation with non breast related phenotypes can ensure us that the model is behaving well and extracting something meaningful. For instance, it is expected that weight related phenotypes will strongly correlate with some features. Since this study focuses primarily on breast related phenotypes, after the initial filtering we apply some word filters on the phenotype names, specifically we filter on the word "breast" (case insensitive) and filter out phenotypes containing the word "hera" (case insensitive; HERA is a research center).

Chapter 4

Results

In here we present the results of our study, which will then be discussed further in the next chapter.

4.1 Autoencoder Training

4.1.1 Training and Validation Loss

When the autoencoder was trained using 5% validation for early stopping, the validation loss converges after about 44 epochs to $L = 0.0071$ with the training loss at $L = 0.0070$. See figure 4.1.

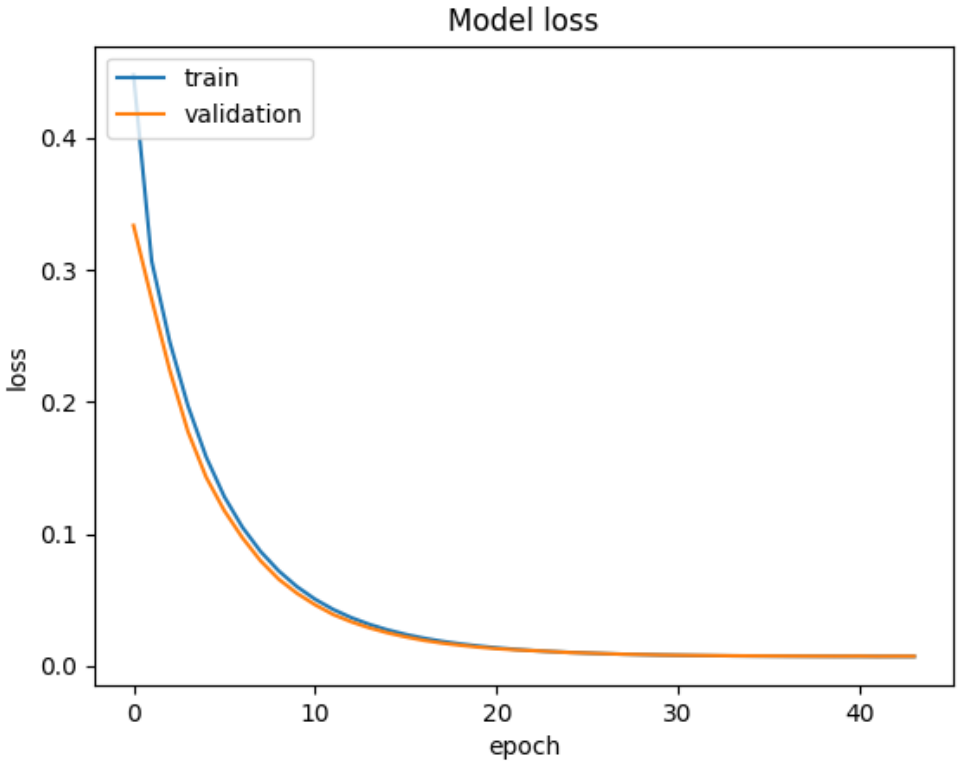

Figure 4.1: Early stopping loss
Traning and validation loss for the model trained with early stopping.

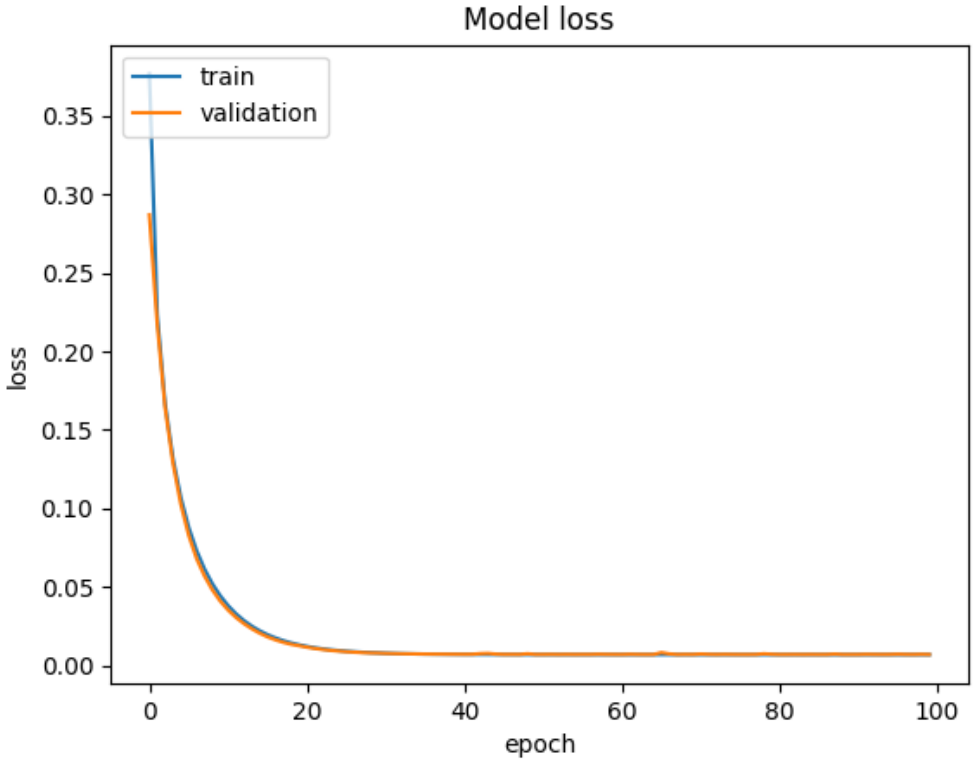

Figure 4.2: Training loss for 100 epochs without early stopping.

If the model is trained for 100 epochs without early stopping, the training loss does not continue decreasing but remains at $L = 0.0070$. The validation loss does not change significantly either, showing only a few very small perturbations. See Figure 4.2.

4.1.2 Reconstruction Error

A sample mammogram and three reconstructions are shown in Figure 4.3.

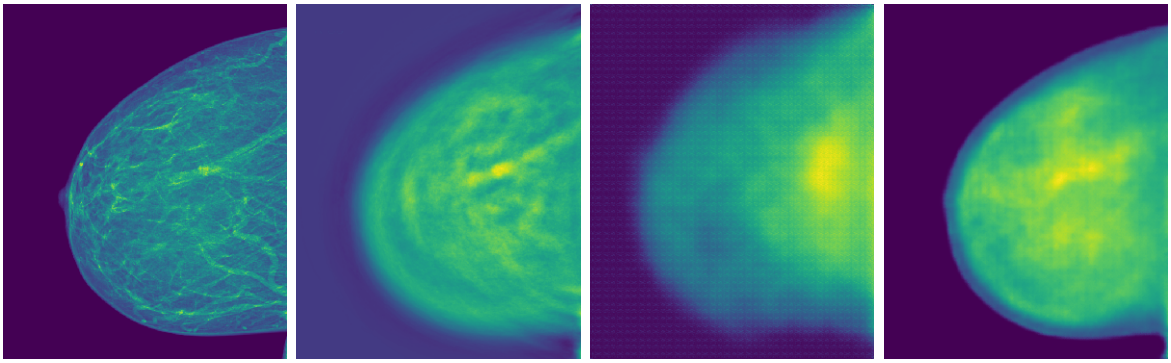

Figure 4.3: Sample mammogram reconstructions.
(Left) Preprocessed and normalized image. (Middle left) Reconstruction from SSVD features. (Middle right) Reconstruction from CAE model trained with early stopping, 44 epochs. (Right) Reconstruction from CAE model, 100 epochs.

Table 4.1: Average reconstruction error of various models as measured with MSE.

Model	Avg. Error
SSVD	7.6×10^{-8}
CAE44	8.4×10^{-7}
CAE100	9.7×10^{-7}
CAE250	9.9×10^{-7}

As a metric for how similar a reconstructed image is to the original image, the mean squared error is calculated by using `sklearn.metrics.mean_squared_error` for each reconstruction. The averages are presented in Table 4.1.

4.2 Feature Study

4.2.1 Age Correlation

The Pearson correlation coefficient (PCC) r and p-values for the three features (feature index, FIX) with the lowest p-values for different models is presented in Table 4.2.

Table 4.2: Feature-age correlation.

SSVD			CAE44			CAE100		
FIX	r	p-value	FIX	r	p-value	FIX	r	p-value
4	-0.148	7.5×10^{-24}	86	-0.244	1.4×10^{-63}	149	0.250	1.2×10^{-66}
1	-0.145	4.8×10^{-23}	192	-0.238	4.6×10^{-60}	182	-0.217	4.3×10^{-50}
5	-0.144	1.0×10^{-22}	47	-0.237	1.3×10^{-59}	88	0.188	6.4×10^{-38}

4.2.2 Feature Tuning

The resulting reconstructed images by tuning a feature that correlated well with age is shown in Figure 4.4. These differences are difficult to interpret with no conclusive results.

4.2.3 Phenotype Correlation

Presented in Table 4.3 is a summary of the results of the phenotype correlation. "Filtered" means results filtered on the word "breast" and excluding phenotypes containing the word "hera". For the unfiltered results, the phenotypes with the lowest p-values (10^{-99}) are all BMI related.

Presented in Tables 4.4, 4.5 and 4.6 are the five (or two in the case of SSVD) phenotypes after word filtering with the lowest p-values. The phenotype `"High_Risk_Breast_Cancer_females_only_EKM_20022018"` shows up for all models.

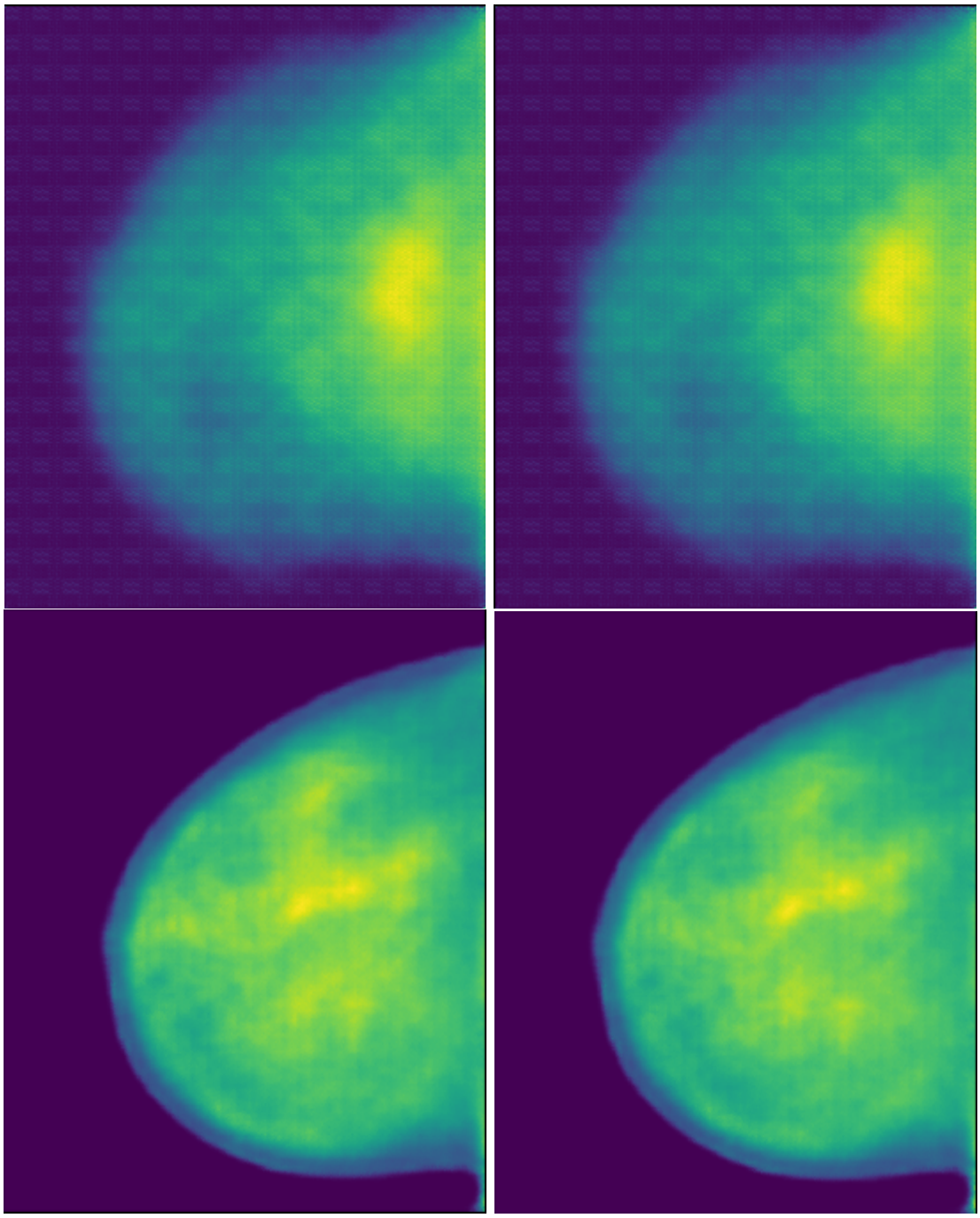

Figure 4.4: Feature tuning.
Reconstructed mammograms after tuning features from CAE models. (Upper left) Tuning CAE44 feature index 86 by a factor of 0.36 (0.95^{20}). (Upper right) Tuning CAE44 feature index 86 by a factor of 2.65 (1.05^{20}). (Lower left) Tuning CAE100 feature index 126 by a factor of 0.36 (0.95^{20}). (Lower right) Tuning CAE100 feature index 126 by a factor of 2.65 (1.05^{20}).

Table 4.3: Phenotype correlations. n corr. is the number of correlating phenotypes.

	SSVD		CAE44		CAE100	
	n corr.	lowest p-value	n corr.	lowest p-value	n corr.	lowest p-value
Unfiltered	732	1.9×10^{-99}	72,563	1.1×10^{-99}	29,259	2.9×10^{-99}
Filtered	2	7.7×10^{-7}	190	2.5×10^{-9}	37	7.1×10^{-9}

Table 4.4: All filtered phenotypes correlating with features from SSVD.

SSVD filtered phenotypes		
FIX	Phenotype	p-value
2	High_Risk_Breast_Cancer_08042019	7.7×10^{-7}
2	High_Risk_Breast_Cancer_females_only_EKM_20022018	9.1×10^{-6}

Table 4.5: Five filtered phenotypes correlating with features from CAE44.

CAE44 filtered phenotypes		
FIX	Phenotype	p-value
29	Benign_breast_lesion_all_noBC_04032011	2.5×10^{-9}
29	Benign_breast_lesion_lowrisk_noBC_04032011	3.0×10^{-9}
40	High_Risk_Breast_Cancer_females_only_EKM_20022018	6.8×10^{-9}
29	Benign_breast_lesion_all_04032011	7.9×10^{-9}
17	Benign_breast_lesion_all_noBC_04032011	8.9×10^{-9}

Table 4.6: Five filtered phenotypes correlating with features from CAE100.

CAE100 filtered phenotypes		
FIX	Phenotype	p-value
111	Benign_breast_lesion_lowrisk_noBC_04032011	7.1×10^{-9}
9	High_Risk_Breast_Cancer_08042019	1.8×10^{-8}
111	Benign_breast_lesion_lowrisk_04032011	4.2×10^{-8}
111	Benign_breast_lesion_all_noBC_04032011	5.1×10^{-8}
59	High_Risk_Breast_Cancer_females_only_EKM_20022018	9.9×10^{-8}

Chapter 5

Discussion

The goal of the project was to capture important biological features in the pixel-wise image data and create quantitative and understandable phenotypes based on those features. Although we did not achieve this, the results nonetheless suggest that the features computed might be capturing biologically meaningful information.

Early stopping is used to stop training early when validation loss stops decreasing and starts increasing instead. At this time the model is starting to overfit and is worse at generalizing. When overfitting the autoencoder trained in this study, we expect the weights of the model itself to begin storing information in some sense, instead of mapping a proper encoding from the latent space to the original space. It is therefore natural to use early stopping. An odd thing was observed when training the model, however. The training loss of the autoencoder converged after about 44 epochs to the value $L = 0.0070$ before the validation loss started increasing again. If the model is trained further, the *observed reconstructed images* are much closer to the original inputs, see Figure 5.1. This is even more so evident when looking at the reconstructed images from a model trained for up to 1,000 epochs (not presented in results as that model is likely overfitting severely). Even for the model trained for 1,000 epochs, the loss remains $L = 0.0070$ which is counter-intuitive as the optimization function is minimizing the loss function and clearly something is happening even though the loss stays the same. This may be related to the loss function chosen (BCE).

The average reconstruction error measured with MSE and presented in Table 4.1 does not seem to reflect our intuition, as it was lowest for the SSVD reconstructed

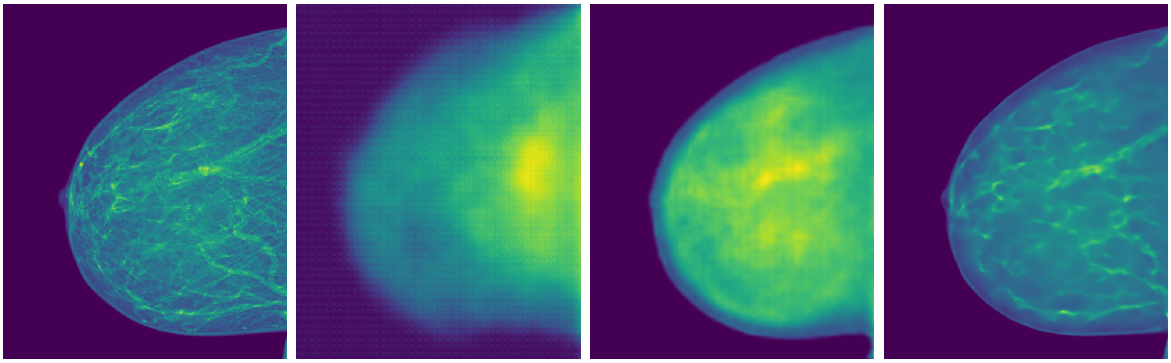

Figure 5.1: Reconstructed mammogram.
(Left) Original image. (Middle left) Reconstructed image from model trained for 44 epochs. (Middle right) Reconstructed image from model trained for 100 epochs. (Right) Reconstructed image from model trained for 1,000 epochs.

images and increasing for the CAE models the longer they were trained. This is perhaps not unusual as the MSE is not the loss function used for training and thus may be somewhat or even entirely unreliable as a metric. Interesting to note is the fact that the average MSE error for the CAE44 model is slightly lower than for other CAE models, even though the reconstructed images are very blurry and seem to be worse than other reconstructions.

Another thing worth mentioning is from preprocessing the images. When extracted raw from the `DICOM` files, the pixel values are stored as type `uint16` which can store integers in the range 0 to $2^{16} - 1 = 65{,}535$. To ensure the pixels were in the range $[0, 1]$ they were divided by 2^{16} (after casting to `float64`). Non-systematically observing the pixel values of a few raw images, we noticed they were never above 4,000. If this is the case for all images, then no pixels would end up with values close to 1. This might have some unknown effects on the loss function BCE in which we interpret pixel values as probabilities of belonging to one of two classes, "on" and "off".

In the age correlation study, we can see that for SSVD the three features with the strongest correlation to age have all low index values. This is expected since the first feature amounts to the greatest variance in the data, the second feature the second most and so on. For the CAE models, the features have no inherent order and could indeed be permuted if the same network was trained twice without changing any of the parameters. Compared to the best SSVD feature baseline correlation of -0.148, the CAE44 model achieves 65% more correlation of -0.244 and a much lower p-value (10^{-24} vs. 10^{-63}). This is some evidence that our architecture is better able to capture the latent space representation of the images than a linear dimensionality reduction method like SSVD. Even though the CAE100 model has a slightly better (considering absolute values) correlation of 0.250 and p-value with a factor of 10^{-66} the other features fall off faster than for CAE44, i.e. CAE44 features have a more consistent correlation. We can see that by just looking at the third features; for CAE44 we have $r = -0.237$ and p-value with a factor of 10^{-59} while for CAE100 we have $r = 0.188$ and p-value with a factor of 10^{-38}. It is likely that CAE100 is starting to overfit which can explain this. There is also the fact that both SSVD and CAE44 features give a negative correlation while CAE100 gives mostly positive correlation.

Feature tuning provided no conclusive results as it was very difficult to see what effect these tunings had on the reconstruction. When observing multiple reconstructions, it seemed that most changes affected the localized thickness of some bright tissue, but not enough to argue for what exactly it was. This is not surprising since one would expect multiple features to govern some aspect of the reconstruction and tuning one feature at a time while holding others fixed would not result in meaningful and easily interpretable results. Perhaps the method was also lacking; we tuned a feature that correlated (relatively) well with age in steps of 0.95 and 1.05, to a minimum of $0.95^{20} = 0.36$ and maximum $1.05^{20} = 2.65$ (while keeping the value of the feature less than or equal to 1).

Unfiltered phenotypes with the strongest correlation (p-value 10^{-99}) to some features for any given model all included "weight", "bmi", "obese", "triglyceride" or similar words. Indeed, more than half of the phenotypes are related to these keywords. Just over 25% of the phenotypes for the CAE44 features include the keyword "ECG", an abbreviation for "electrocardiography". ECG is also related to BMI. This means that over 75% of the phenotypes that correlated well with the features were somehow BMI related. This is unsurprising but gives certain credence to our models.

The number of (filtered or not) correlations is greatest for the CAE44 model, suggesting that more features are being picked up for correlation. For the CAE100 model, the number of correlating phenotypes has gone down again. This does not necessarily mean in and of itself that the CAE44 model is somehow better than the CAE100 model, but it is suggestive. The p-values for the filtered phenotypes are not particularly low although the lowest p-value for the CAE44 model is lower than for SSVD. It is noteworthy to point out that none of the features that showed a relatively high correlation with age show up in the top five filtered phenotype correlations for any model, likely because of the age adjustment. This suggests those features were likely capturing less biological variances since all features were adjusted for age. The only correlating feature for SSVD is the second one, again unsurprisingly low index as explained before. There is however some consistency in feature correlation, as for the CAE44 model, for instance, feature number 29 occurs three out of five times in the correlations with the lowest p-values.

5.1 Summary

Feature extraction was done using both Stochastic Singular Value Decomposition (SSVD) and by training a Convolutional Autoencoder (CAE). The CAE model was trained with early stopping which stopped at 44 epochs. The model was also forced to train for 100 epochs. The reason for this was that the training loss converged before validation loss started increasing again and training further, the loss stayed the same while the observed reconstructed image got more accurate, which is counter-intuitive.

The CAE44 features outperformed the baseline SSVD features considerably when calculating correlation with age. The best correlation found was $r = -0.244$ with a p-value of about 10^{-63} for the CAE44 features, compared to $r = -0.148$ with a p-value of about 10^{-24} for the SSVD features. This is a significant difference and suggests the CAE44 model is arguably finding a better lower dimensional representation of the original images than SSVD.

Feature tuning was inconclusive, but that may be explained by the method used.

The phenotype-feature correlation study found considerable correlations with phenotypes related to body mass index (BMI), with p-values as low as 10^{-99}. More features from the CAE44 model were found to correlate with phenotypes than SSVD. When filtered on the word "breast" 190 feature-phenotype correlations were found for the CAE44 model while only 2 were found for SSVD, in addition to lower p-values for CAE44 than SSVD.

5.2 Conclusion

We did find some correlation between the features extracted and age or phenotypes. These results suggest that unsupervised learning may be useful for creating biologically meaningful phenotypes from mammograms.

For both age and phenotype correlation, the features extracted with the Convolutional Autoencoder correlated better than those features extracted with Stochastic Singular Value Decomposition. This was especially the case for the age correlation.

The conclusion of this study is that a CAE model is appropriate for extracting useful features from a mammogram data set, or at least a good starting point.

5.3 Future Work

The data set has recently grown from about 4,500 images to just over 75,000. Deep learning models in computer vision are frequently trained on hundreds of thousands of samples to more than a million. A new network could be trained using this new data set. Data augmentation methods could be done by training on slides created from the original high definition data set, or by shifting or rotating the images slightly. New methods must be devised since the current implementation assumes the data can be loaded into CPU memory all at once.

Since the SVD method used was a probabilistic one that calculated an approximate decomposition, it would of interest to also to do a full SVD of the original data set for a better comparison.

Better preprocessing should be applied to ensure a more even distribution of pixel values with in the $[0, 1]$ range, since, as discussed before, no pixel values were likely close to 1 after preprocessing that might affect the backpropagation. It is possible to simply rescale the pixel values such that min and max values are closer to 0 and 1, respectively.

The autoencoder training and feature study could be repeated with a different loss function, such as MSE. It would also be interesting to apply some regularization terms to the loss function for a more sparse representation.

More investigation should be done to understand why the training loss converges so quickly, despite the reconstructions seeming to improve when forced to train for longer.

Variational Autoencoders (VAE) have recently emerged as powerful deep generative models [20] [21]. They can be used to find a disentangled representation of a data set. This means that with VAEs, we have more control over latent representations and can try to weed out factors that are of less interest. This could be useful for searching for biological and heritable features in the data set.

We intended to run the features through GWAS to look for heritability but as it turned out, there were not enough mother-daughter pairs to get meaningful results. This will likely not be the case with the new data set. Ultimately, we hope to create understandable phenotypes from the pixel data that can be used for further studies.

Bibliography

[1] N. Dhungel, G. Carneiro, and A. Bradley, "Automated mass detection from mammograms using deep learning and random forest", Oct. 2015. doi: 10.1109/DICTA.2015.7371234.

[2] T. Kooi, G. Litjens, B. van Ginneken, A. Gubern-Mérida, C. I. Sánchez, R. Mann, G. Heeten, and N. Karssemeijer, "Large scale deep learning for computer aided detection of mammographic lesions", *Medical Image Analysis*, vol. 35, Aug. 2016. doi: 10.1016/j.media.2016.07.007.

[3] S. Charan, M. Khan, and K. Khurshid, "Breast cancer detection in mammograms using convolutional neural network", Mar. 2018. doi: 10.1109/ICOMET.2018.8346384.

[4] D. Ribli, A. Horváth, Z. Unger, P. Pollner, and I. Csabai, "Detecting and classifying lesions in mammograms with deep learning", *Scientific Reports*, vol. 8, Jul. 2017. doi: 10.1038/s41598-018-22437-z.

[5] T. Elgamal and M. Hefeeda, "Analysis of pca algorithms in distributed environments", *ArXiv preprint arXiv:1503.05214*, 2015.

[6] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions", *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[7] P.-G. Martinsson, V. Rokhlin, and M. Tygert, "A randomized algorithm for the decomposition of matrices", *Applied and Computational Harmonic Analysis*, vol. 30, pp. 47–68, Jan. 2011. doi: 10.1016/j.acha.2010.02.003.

[8] 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 Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989, issn: 0899-7667. doi: 10.1162/neco.1989.1.4.541. [Online]. Available: http://dx.doi.org/10.1162/neco.1989.1.4.541.

[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, Lille, France: JMLR.org, 2015, pp. 448–456. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045167.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, MA, USA: The MIT Press, 2016, isbn: 0262035618, 9780262035613.

[11] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks", in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, G. Gordon, D. Dunson, and M. Dudík, Eds., ser. Proceedings of Machine Learning Research, vol. 15, Fort Lauderdale, FL, USA: PMLR, Nov. 2011, pp. 315–323. [Online]. Available: http://proceedings.mlr.press/v15/glorot11a.html.

[12] Y. Zhou and R. Chellappa, "Computation of optical flow using a neural network", Aug. 1988, 71–78 vol.2. doi: 10.1109/ICNN.1988.23914.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[14] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning", *CoRR*, vol. abs/1603.07285, 2016.

[15] D. H. Ballard, "Modular learning in neural networks", in *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*, ser. AAAI'87, Seattle, Washington: AAAI Press, 1987, pp. 279–284, isbn: 0-934613-42-7. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863696.1863746.

[16] A. Creswell, K. Arulkumaran, and A. A. Bharath, "On denoising autoencoders trained to minimise binary cross-entropy", *CoRR*, vol. abs/1708.08487, 2017. arXiv: 1708.08487. [Online]. Available: http://arxiv.org/abs/1708.08487.

[17] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude", *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012. [Online]. Available: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

[18] D. Kingma and L. Ba, "Adam: A method for stochastic optimization", in *3rd International Conference on Learning Representations (ICLR)*, (May 5–7, 2015), San Diego, 2015.

[19] A. Augustyn, P. Bauer, B. Duignan, A. Eldridge, E. Gregersen, J. Luebering, A. McKenna, M. Petruzzello, J. P. Rafferty, M. Ray, K. Rogers, A. Tikkanen, J. Wallenfeldt, A. Zeidan, and A. Zelazko, "Phenotype", in *Encyclopædia Britannica*, Encyclopædia Britannica, Inc., September 05, 2016. [Online]. Available: https://www.britannica.com/science/phenotype.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes", *ArXiv preprint arXiv:1312.6114*, 2013.

[21] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models", *ArXiv preprint arXiv:1401.4082*, 2014.