
Pre-classification of Hairy Cells Using Supervised Contrastive Learning

Can contrastive learning improve cell image classifiers?

Alexia Han & Rebecca Schömer Ericsson

Examiner:

Niels Christian Overgaard

Supervisors at LTH:

Kalle Åström

Ida Arvidsson

Jennie Karlsson

Supervisor at CellaVision:

Kent Stråhlén



2022/06/18

Master's Thesis

Degree Project in Mathematics for Engineers

Faculty of Engineering

Lund University

Lund, Sweden

Abstract

Differential of the peripheral blood is an important tool when assessing blood-related diseases. Lymphocytes are part of the immune system and morphological changes thereof should raise attention. CellaVision's current systems automatically detect and pre-classify lymphocytes that exhibit atypical morphologies but do not further distinguish those into subclasses. By adding the subclasses into the pre-classification, the overall accuracy would improve and the additional information would also alleviate the work load of healthcare professionals. The objective of this thesis is to discriminate hairy cells from lymphocytes and other cell classes. The data consists of expert-labeled images of white blood cells belonging to 20 classes, and is split into a training and a test set, with 76 612 and 15 374 images each. We compare a traditional transfer learning network against a supervised contrastive learning network and propose a methodology based on a three-step training sequence combining the two. The networks that were trained using supervised contrastive learning outperformed the traditional transfer learning networks. The best test accuracy for a contrastive learning network was 90.24%, while the best transfer learning network only obtained a test accuracy of 88.21%. With our findings, the supervised contrastive loss-aided methodology has proven to have great potential for pre-classifying hairy cells, as well as being superior in overall automatic classification of white blood cells.

Acknowledgments

The work with this master's thesis took place at CellaVision, Lund, during the spring of 2022. The thesis is the final part of our study program Engineering Physics (M.Sc.Eng.) at the Faculty of Engineering, Lund University.

The journey of a thousand miles begins with a single step. We are immensely grateful to Kent Stråhlén, our supervisor at CellaVision, for introducing us to CellaVision three years ago, as well as his never-ending support and humorous encouragements all through the project, and for tirelessly solving all the hiccoughs with us. It is almost unfathomable to look back and realize how much we have learned during this work. For this, we would like to thank our supervisors at LTH, Kalle Åström, Ida Arvidsson and Jennie Karlsson, for generously sharing their expertise knowledge with us, and providing us with invaluable ideas that propelled us to explore further into the topic which added much more depth to the thesis. Without adequate data, a machine learning model cannot be built. Therefore, we thank Steven Mariounneaux for sharing his knowledge and enthusiasm with lymphocytes and for the time spent on labeling our images.

To Hans-Inge Bengtsson for inspiring talks in the process of finding a topic for the thesis.

To Åse Sykfont Snygg and Annelie McCorkindale for locating and letting us borrow the peripheral blood smears that contained hairy cells.

To all our friends and family for being there for us and enduring the "elevator pitches" even though they do not work in machine-learning nor hematology related fields.

Finally, we would like to extend a huge thank you to CellaVision and all our colleagues for fika and for being, as always, helpful and welcoming.

Alexia Han & Rebecca Schömer Ericsson
Lund, 2022/06/18

Contents

Abstract	iii
Acknowledgments	v
List of Figures	x
List of Tables	xi
Nomenclature	xii
1 Introduction	1
1.1 Motivation	1
1.2 Aim	2
1.3 Scope	2
1.4 Prior Work	2
2 Background	5
2.1 Peripheral Blood	5
2.2 CellaVision	6
2.3 Lymphocytes	6
2.3.1 Hairy Cells	8
3 Theory	9
3.1 Artificial Neural Networks	9
3.1.1 Perceptron	9
3.1.2 Multilayer Perceptron	10
3.1.3 Loss Functions	11
3.1.4 Optimization	12
3.1.5 Performance Measures	13
3.1.6 Generalization	15
3.1.7 Augmentation	15
3.1.8 Dropout	16
3.2 Convolutional Neural Networks	17
3.2.1 Introduction	17
3.3 The Inception Architecture	18
3.3.1 Introduction	18

3.3.2	Naïve Implementation	18
3.3.3	The Inception Module	19
3.3.4	Xception	21
3.4	Transfer Learning	21
3.5	Contrastive Learning	22
3.5.1	Supervised Contrastive Loss	23
3.5.2	Pair-wise Augmentation	24
3.5.3	Architecture	24
4	Methodology	25
4.1	Collection of Cell Images	25
4.1.1	Training, Validation, and Test Split	26
4.1.2	Expert Classification	27
4.2	Transfer Learning	27
4.2.1	Model Creation	27
4.2.2	Metrics	28
4.2.3	Training	29
4.2.4	Model Selection	29
4.2.5	Data Augmentation	29
4.2.6	No Transfer Learning: Model with Randomly Initialized Weights	30
4.3	Contrastive Learning	31
4.3.1	Model Creation	31
4.3.2	Metrics	32
4.3.3	Training	33
4.3.4	Model Selection	34
4.3.5	Data Augmentation	34
4.4	Sanity Test	35
5	Results	37
5.1	Transfer Learning with Xception	37
5.1.1	Randomly Initialized Model	39
5.2	Contrastive Learning	41
5.3	Sanity Test	45
6	Discussion	49
6.1	Random initialization of Xception	49
6.2	Transfer Learning with Xception	50
6.3	Contrastive Learning	51
6.4	Sanity Test	53
6.5	Comparison of the Learning Methods	54
6.6	General Discussion	55
7	Conclusion	57
	References	59
A	Miscellaneous	63
B	Architectures	65
B.1	Architecture for HAL-T and HAL-R	65
B.2	Architecture for the Encoder of HAL-C	65

B.3	Architecture for the Projection Network of HAL-C with 16 Projection Units	66
B.4	Architecture for the Projection Network of HAL-C with 128 Projection Units	66
B.5	Architecture for the Classifier Network of HAL-C	67
C	Hyperparameters	69
D	Popular Science Summary	71

List of Figures

2.1	CellaVision DC-1 Analyzer	6
2.2	Small and large lymphocytes	7
2.3	Reactive and abnormal lymphocytes	7
2.4	Hairy cells	8
3.1	Perceptron	10
3.2	Classification problems	11
3.3	Confusion matrix	14
3.4	Three views of Arnold's cat	18
3.5	Naïve Inception module	19
3.6	1x1 convolution	20
3.7	Improved Inception module	20
3.8	Schematic drawing of Xception	21
3.9	Illustration of contrastive learning	23
4.1	Three-step training procedure	33
4.2	Abnormal lymphocytes with morphologies like hairy cells	36
5.1	Confusion matrix for HAL-T3002	38
5.2	Lymphocyte confusion matrix for HAL-T3002	38
5.3	Encoded space of HAL-T3002	39
5.4	Confusion matrix for HAL-R	40
5.5	Encoded space of HAL-R	40
5.6	Confusion matrix for HAL-C3001	42
5.7	Confusion matrix for HAL-C3003	42
5.8	Lymphocyte confusion matrix for HAL-C3001	43
5.9	Lymphocyte confusion matrix for HAL-C3003	43
5.10	Encoded space of HAL-C3001	44
5.11	Encoded space of HAL-C3003	44
5.12	Confusion matrix for HAL-T3003 trained on A and B, tested on C . .	46
5.13	Confusion matrix for HAL-T3003 trained on A, B and C, tested on C	46
5.14	Confusion matrix for HAL-T3003 trained on A and B, tested on D . .	47
5.15	Confusion matrix for HAL-T3003 trained on A, B and C, tested on D	47

List of Tables

3.1	Table with activation functions	12
4.1	The number of cell images used for the final training	26
4.2	The number of cell images used for testing	27
4.3	Metrics used in transfer learning	28
4.4	Selected hyperparameters in transfer learning	30
4.5	Selected augmentation settings in transfer learning	31
4.6	Selected hyperparameters in the randomized initialization training . .	31
4.7	Metrics used in contrastive learning.	32
4.8	Selected hyperparameters for the projection network in contrastive learning	34
4.9	Selected hyperparameters for the classifier network in contrastive learning	34
4.10	Selected augmentation settings in contrastive learning	35
4.11	A summary of the datasets used in the sanity test	36
5.1	Results for HAL-T	37
5.2	Results for HAL-R	39
5.3	Results for HAL-C	41
5.4	Sanity test results part 1 - tested on C	45
5.5	Sanity test results part 2 - tested on D	45
6.1	Examples of false negative hairy cells	51
6.2	Examples of false positive hairy cells	53
C.1	Selected hyperparameters for projection-head-16 in contrastive train- ing with validation split	69

Nomenclature

- ANN** Artificial neural networks
- CNN** Convolutional neural networks
- Cytoplasm** The cell material surrounding the cell nucleus confined within the cell membrane
- HAL** Hairy abnormal lymphocyte network
- HAL-C** Supervised contrastive learning network
- HAL-R** Randomly initialized network
- HAL-T** Cross-entropy transfer learning network
- HCL** Hairy cell leukemia
- ICSH** International Council for Standardization in Haematology
- Lineage** The cell type of earlier stages from which a cell has developed
- Neoplastic cells** Cells with an uncontrolled cell division, i.e. tumor cells
- Nucleus** The part of the cell that contains the genetic information
- Slide** A small, thin sheet of glass with a smeared out drop of blood on top, which is used for examination of the peripheral blood under a microscope
- Stain** A chemical solution to make it easier to detect cells and to distinguish their morphologies in microscopy
- t-SNE** t-distributed stochastic neighbor embedding, a non-linear method for dimensionality reduction

The i.i.d. assumption Assumption that random variables are independent and identically distributed

WBC White blood cells

Introduction

What we neglect we become [1].

1.1 Motivation

White blood cells (WBCs) can be found in peripheral blood. Based on their morphology, the WBCs can be separated into several cell classes. Lymphocytes are one such cell class, which in case of leukemia or lymphoma can exhibit atypical morphologies. In some cases, these abnormal lymphocytes can in turn be divided into neoplastic cell classes, such as Sézary cells, mantle cells and hairy cells, etc. The authors of [2] state that, when possible, abnormal lymphocytes should be classified to particular neoplastic cells, i.e., a hairy cell that can be classified as a hairy cell should be classified as a hairy cell and not just an abnormal lymphocyte.

Hairy cells can appear in the blood stream in case of hairy cell leukemia (HCL) [3]. According to [3] the main diagnostic tools for establishing if a patient has HCL are examinations of peripheral blood and bone marrow, where the former will be the focus of this project. Definitive diagnosis is normally conducted with flow cytometry using immunophenotypic markers, but since flow cytometry is more expensive, the networks developed in this project can be used to produce an initial indication to the necessity of further investigations using flow cytometry [4].

This project will provide insight into the feasibility of using deep learning to pre-classify hairy cells in peripheral blood smears, and we will suggest a novel learning approach to pre-classify hairy cells and simultaneously improve the classification performance for other WBC classes. The current method of manual assessment is tedious, time consuming, requires a high level of competence and can be subjective. An automated method could help diminish these limitations. A more reliable automated classification of abnormal lymphocytes could accelerate diagnosis and result in an improvement in the quality of patient care.

1.2 Aim

The aim of the thesis is to pre-classify WBCs, and in particular distinguish hairy cells from other types of lymphocytes in a peripheral blood smear. The classification problem will be solved using deep artificial neural networks (ANNs). We will look at two different methods for learning. The first method is *transfer learning*, and the second method is *supervised contrastive learning*. Special attention will be given to the comparison between the models, in regard to their accuracy and generalization abilities on the task at hand. To summarize, our goals are to:

- distinguish hairy cells,
- pre-classify white blood cells into 20 classes, and
- compare traditional transfer learning to supervised contrastive learning.

1.3 Scope

One limitation is due to the rareness of HCL. The hairy cell data are collected from 11 slides originating from three hospitals. The cell images of the other cell classes come from a wide range of hospitals to ensure generalization of the trained classifier. Different hospitals use different staining methods, and unless the training data is representative of the different techniques, unwanted bias can be introduced towards the staining method that is most prevalent in the dataset.

The performance of a deep learning network is strongly influenced by the network architecture and the selection of hyperparameters. We will limit ourselves to using Xception as base architecture, and one of the challenges is therefore to find an optimal set of hyperparameters for the model and each learning method.

1.4 Prior Work

The traditional machine learning techniques involving segmentation, feature extraction and analysis, as well as classification using statistical methods have been utilized to identify subtypes of lymphocytes in several earlier studies.

The usage of segmentation and feature extraction, together with fuzzy C-means to classify chronic lymphocytic leukemia cells, HCL cells, and normal lymphocytes, is described in [5]. The authors of [6] use feature analysis and support vector machine to identify reactive lymphoid cells (RLC), lymphoid blast cells, and myeloid blast cells.

In recent years, the development of deep learning using convolutional neural networks (CNNs) has further advanced automatic classification.

One usage of a CNN was carried out in [7], where the authors constructed a transfer learning model using pre-trained AlexNet together with fine-tuned top-layers and reached a high level of accuracy for the detection of acute lymphocytic leukemia subtypes. AlexNet is a network presented in [8].

An interesting approach was defined in [9], where two deep learning networks are joined in a tandem fashion to first separate abnormal cells from normal cells, then distinguish the lineage of the cells. This knowledge helps derive the diagnosis on acute leukemia. Several pre-trained models, such as ResNet, are evaluated.

The earlier studies have shown that it is possible to use machine learning techniques to identify cell types that share similar characteristics. There are both studies that have had success with identifying hairy cells using traditional machine learning techniques and studies that have used deep learning network on other abnormal lymphocytes. This makes us believe that we should be able to find a network architecture that can distinguish hairy cells from other abnormal lymphocytes using deep learning techniques.

Contrastive loss-aided self-supervised learning has been proven to achieve higher accuracy score on the ImageNet dataset than cross-entropy based losses [10]. In [11], it was shown that contrastive loss could be modified to accommodate fully-labeled data and thus be used for supervised training, and still perform better than cross-entropy.

Two previous works have conducted experiments on classifying WBCs with contrastive loss. One used unsupervised learning and achieved better results than other unsupervised methods [12]. The other one presented a hierarchical network that combined supervised, unsupervised, and semi-supervised contrastive losses to allow the usage on datasets with all degrees of labeling [13]. The dataset used in [13] only included 15 WBC classes. Furthermore, it was only trained on partially-labeled datasets.

We propose the use of supervised contrastive loss combined with transfer learning for the pre-classification of hairy cells along with 19 WBC classes, as suggested by CellaVision, and compare it to supervised cross-entropy transfer learning models.

If you can understand it, you can understand it [1].

2.1 Peripheral Blood

The blood that circulates in the human body and transports nutrients to organs and tissues is called the peripheral blood. A laboratory analysis of the peripheral blood of the patient is often one of the initial evaluations the physician orders when suspicions on blood-related illnesses arise [14].

After the blood sample is collected from the patient, it is analyzed through a workflow consisting of systems with different purposes. The cell counter measures the concentrations of cells in the blood. When an anomaly is detected by the cell counter, a blood smear is made either automatically or manually, and sent to a cell image analyzer that performs blood differential. During a blood differential, the cells are categorized into different classes based on their morphology, to assess the WBC distribution in the blood. The differential can then be used as an aid for the physician in diagnosis and care-decision making.

Traditionally, the examination of blood smears is conducted by trained laboratory personnel using conventional microscopes. Not only is this process bottle-necked by the availability of experts, but it also requires manually adjusting the focus and shifting the field of view, while also discerning the morphology of the cells examined. The opinions of experts often differ based on their training and preferences. Since the automated differential method is trained on the ground truth produced by many experts, the output is more universal. Therefore, it has the advantage of producing consistent results across all labs.

2.2 CellaVision

Founded in 1994 with the HQ located in Lund, Sweden, CellaVision is a global company that aims to alleviate the pressure on healthcare providers caused by the combination of shortage on hematology experts and increased volume of samples that need to be analyzed [15]. The product line consists of digital microscopy systems that automate hematology lab work. The main customers are large laboratories.

The analyzers collect high-resolution images of the blood cells, and the software makes pre-classifications of each of the cells based on their morphology, aided by ANNs. Lab workers can then examine the pre-classification results and assign new labels to cells if needed. Currently, analyses of WBCs, red blood cells (RBCs), and blood platelets are available. Only WBCs will be considered in this thesis. A CellaVision DC-1 analyzer is shown in Figure 2.1.



Figure 2.1: *To the left in the figure is the CellaVision DC-1 Analyzer, and to the right is the user interface of the accompanying software. Image from [16].*

2.3 Lymphocytes

One class of WBCs that CellaVision pre-classifies to is lymphocytes. They can be subdivided into normal, reactive and abnormal lymphocytes according to the standard set by ICSH in [2]. Examples of the different types of lymphocytes can be seen in Figure 2.2 and Figure 2.3. Reactive lymphocytes are lymphocytes that are responding to some sort of intrusion of the body, e.g., a viral infection, while abnormal lymphocytes are cells that have morphological characteristics which cannot be regarded as normal. It can e.g., be atypical color, shape or nucleus. Some abnormal lymphocytes can be subclassed into neoplastic cell classes, whose presence in the blood can be directly linked to different forms of leukemia and lymphoma [2].

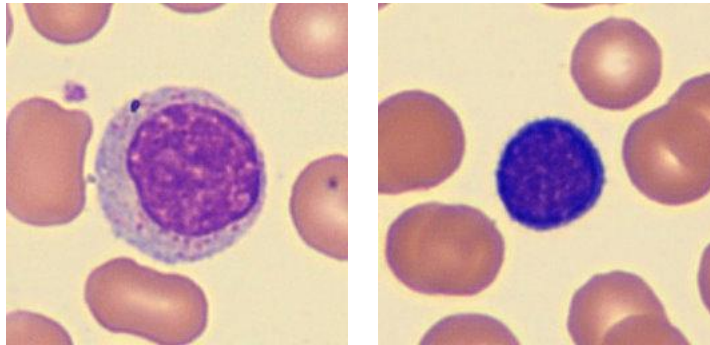
(a) *Large lymphocyte*(b) *Small lymphocyte*

Figure 2.2: *Examples of lymphocytes of varying sizes. There are visual differences between the cells despite them belonging to the same cell class. For instance, the cytoplasm of the small lymphocyte is noticeably smaller in area.*

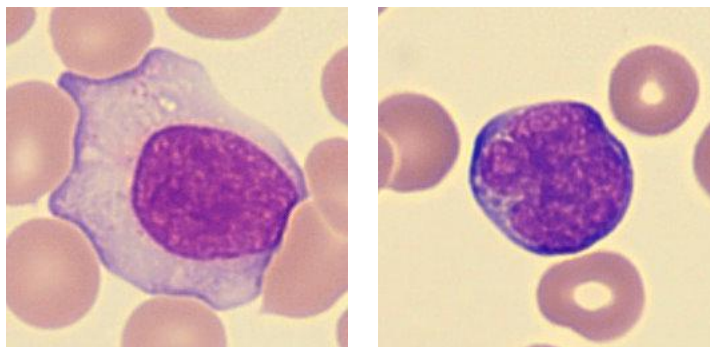
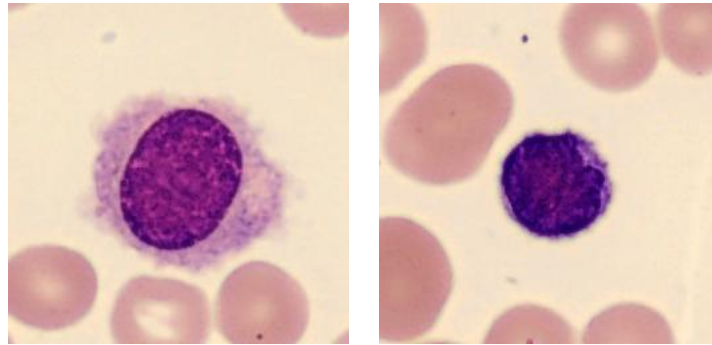
(a) *Reactive lymphocyte*(b) *Abnormal lymphocyte*

Figure 2.3: *Reactive lymphocytes are often characterized by abundant cytoplasm, while abnormal lymphocytes can take on many different shapes depending on the variant. The abnormal lymphocyte shown here has distinct "notches" in the nucleus that are not seen on a normal lymphocyte.*

2.3.1 Hairy Cells

Hairy cells are a subclass of abnormal lymphocytes that can be found in the peripheral blood of a patient with HCL, as described in [3]. They are recognizable mainly by their hairy projections, i.e., the thin, fuzzy strains on the perimeter of the cytoplasm, as seen in Figure 2.4. Other distinctive features are the color of the cytoplasm, and the appearance of the nucleus, the shape of which can vary and have a spongy appearing chromatin. The position of the nucleus can deviate from the center of the cell.



(a) *Large hairy cell*

(b) *Small hairy cell*

Figure 2.4: *Just as the lymphocytes, hairy cells also come in different sizes. For both cells, however, the hairy projections are clearly visible.*

We believe that these distinctive features make hairy cells a good candidate for a classification problem. It is worth noting that there are other cells that can have a similar appearance as hairy cells, without being actual hairy cells. Such cells can be found in splenic marginal zone lymphoma, which often exhibit polar villous projections, T-cell prolymphocytic leukemia, or as artefacts due to bad storage conditions [4]. As the network can only decide based on the appearance of the cells, the network will only give a suggestion of classification, while the final diagnosis will be left to medical professionals, who will have a more holistic understanding of the situation of a patient.

I am deep, I am deep, I am deep [1].

3.1 Artificial Neural Networks

3.1.1 Perceptron

Artificial neural networks (ANNs) are loosely inspired by the neurons in the human brain and how they work together to solve complex problems by propagating information between each other [17]. The descriptions of ANNs and its components are derived from [18], unless stated otherwise.

Each neuron in an ANN is commonly referred to as a node and can be seen as a scalar multiplication of the input vector \mathbf{x} and a weight vector $\boldsymbol{\omega}$. Frank Rosenblatt developed the simplest example of an ANN, the *perceptron*, in 1958 [19]. The objective of the perceptron is to simply sum the output from a number of nodes, add a bias weight b to the sum, and calculate the result with an output activation function. A perceptron is shown in Figure 3.1. The bias serves to shift the sum numerically, and the usage of the activation function will be explained later on.

The mathematical formulation for the perceptron is

$$y(\mathbf{x}, \boldsymbol{\omega}, b) = \phi(\boldsymbol{\omega}^T \mathbf{x} + b) = \phi\left(\sum_{k=1}^K \omega_k x_k + b\right) \quad (1)$$

where y is the output of the perceptron, ϕ the activation function, and K the length of the input vector.

For a classification problem each input \mathbf{x} , commonly referred to as a pattern, has a corresponding ground truth, called the target d . The core idea of the ANN is to take the input and then, by approximating some function f , output y such that y

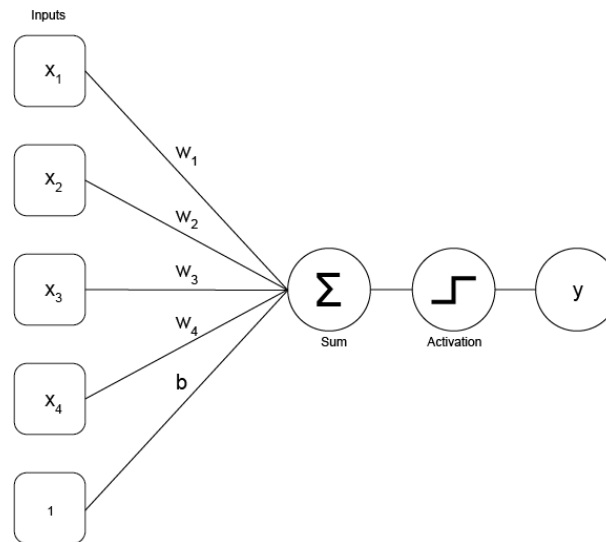


Figure 3.1: A perceptron with four inputs. The inputs $x_1 \dots x_4$ are multiplied with the weights $w_1 \dots w_4$. The products are then summed and added with the bias. An activation function is applied on the sum to produce the output y .

is the same as the target [20], i.e.,

$$y \approx f(\mathbf{x}) = d. \quad (2)$$

To understand the perceptron as a linear classifier, we study the expression without the activation function. By expanding the part inside the parentheses in Equation (1) and setting it equal to a constant C , we get

$$\omega_1 x_1 + \omega_2 x_2 + \dots + b = C. \quad (3)$$

Restricting the problem to be on the two-dimensional plane and subtracting C from both sides, this becomes the equation of a line,

$$\omega_1 x_1 + \omega_2 x_2 + b - C = 0. \quad (4)$$

The line is called the decision boundary. If we train a perceptron and use the found weight values to plot a line, it will fit between the samples of the two classes.

With this interpretation, it becomes apparent that the perceptron cannot solve a nonlinear problem. Examples depicting one linear and one nonlinear problem are illustrated in Figure 3.2.

3.1.2 Multilayer Perceptron

Adding one or several hidden layers between the input layer and the output layer, the network becomes a multilayer perceptron (MLP). The intermediate layers are called hidden since they, unlike the input and output layers, are not seen from outside the network. Each node in one layer is connected to every node in the next layer. For

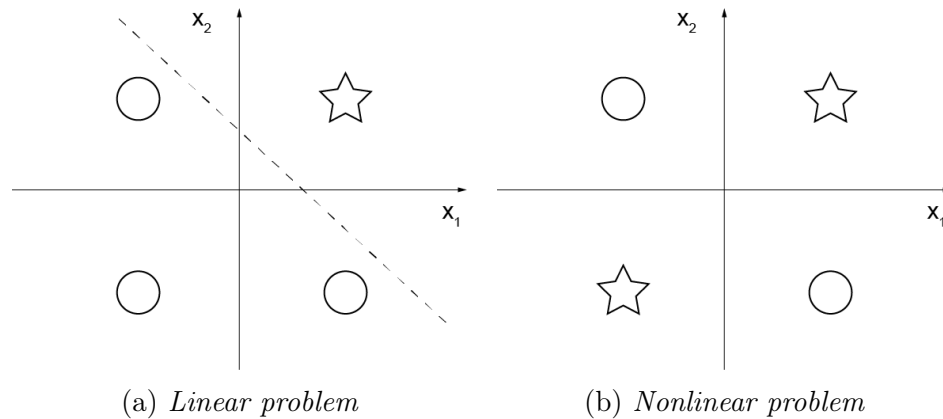


Figure 3.2: Two illustrative examples of classification problems. The shapes represent class labels, and the dotted line is a decision boundary. A linear decision boundary can be found for the left problem while the right problem can only be solved using a nonlinear decision boundary.

an MLP with one hidden layer, the value at each hidden node is given by

$$h_j = \phi_h(\mathbf{u}_j^T \mathbf{x} + b_h), \quad (5)$$

where ϕ_h is called the hidden activation function, and \mathbf{u} is the hidden weights and b_h the bias. The values at the hidden nodes are then used to calculate the output,

$$y = \phi_o(\boldsymbol{\omega}^T \mathbf{h} + b_o), \quad (6)$$

where ϕ_o is the output activation function, and $\boldsymbol{\omega}$ and b_o are the output weights and bias respectively. The hidden activation should be some nonlinear function, otherwise the function f will be reduced to a single matrix multiplication, and the effect of the hidden layer or layers is lost. One commonly used hidden activation function is the *rectified linear unit* (ReLU).

The output activation is normally a sigmoid function for binary classification, or softmax for multiclass problems. The activation function makes it possible to interpret the output as the probability of belonging to a certain class. Some examples of activation functions used for hidden layers and the output layer are shown in Table 3.1

The MLP has one major drawback, which is that the number of weights needed grows with the number of layers and nodes beyond what is reasonable for a normal computer to handle, which makes an MLP with several layers unrealistic to train and use for more complex problems. Thus, a different approach is needed. But first, let us take a look at how neural networks are trained.

3.1.3 Loss Functions

A loss function is needed to measure the difference between the output and the target. One requirement of this function is differentiability. To suit different tasks,


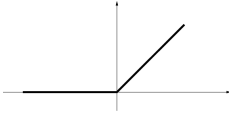
Name	Function	Plot
Logistic	$\frac{1}{1 + e^{-x}}$	
ReLU	$\max(0, x)$	

Table 3.1: *Examples of activation functions [18]. Both their expressions and appearances can be seen in the table. The logistic function is usually used in an output layer, while the ReLU is more often seen in a hidden layer.*

many loss functions have been developed. For classification, the most commonly used loss function is the cross-entropy [18],

$$E_n(\omega) = -d_n \ln(y_n) - (1 - d_n) \ln(1 - y_n) \quad (7)$$

for each pattern n , where the targets d_n are either 0 or 1. The implication is that, if pattern n has target value 0 and prediction $y_n = 0$, or target value 1 with prediction 1, the term E_n will be 0. Contrarily, if the prediction disagree with the target value, the term will be registered and this pattern is "punished" by the loss function. The size of the term is larger the farther the prediction is from the target. The function shown is the binary cross-entropy loss. The categorical cross-entropy loss works similarly, but is instead used when the number of target classes exceeds 2. Another example is the supervised contrastive loss, which is used in clustering tasks. Explanation of the supervised contrastive loss can be found in 3.5.1.

3.1.4 Optimization

To find the optimal solution of the network, the loss function needs to be minimized with respect to the weights. The standard approach of minimization is the gradient descent, in which steps $\Delta\omega$ are taken in the opposite direction of the gradient,

$$\Delta\omega = -\eta \frac{\partial E}{\partial \omega}, \quad (8)$$

where η is the adjustable learning rate.

For a perceptron, the partial differential can simply be calculated on the loss function with respect to the weight vector, using the chain rule. But in an MLP, the loss function of the output layer is dependent on not only the output weights, but also the weights to the hidden layer as shown in Equation (5) and Equation (6). To handle the updates of weights through such a network, the learning algorithm *backpropagation* is used. The principle is that the gradients are computed at the output layers, and

propagated backwards in the network to be used to update the weights in the earlier layers.

The gradient descent method was appreciated for its simplicity, but has some drawbacks. It can get stuck in plateaus or local minima, and if the dataset is large, the computation gets time-consuming.

3.1.4.1 Stochastic Gradient Descent

When the gradient is calculated on all samples, it can only have one direction at each training step and the method is deterministic. If the weights get stuck in a local minimum, there is no way for it to rebound and escape for further improvement. A way to introduce randomness is to instead calculate the gradient of a single sample and make mini-updates to the weights. This is called online updating. When all samples have been used once for updating the weights, we say that an epoch has elapsed. An intermediate method between the fully deterministic and the fully stochastic approach is to instead divide the set of training samples into mini-batches. When training on such mini-batches, the method is known as the stochastic gradient descent algorithm. The batch size is a tunable hyperparameter that affects the model performance. [18]

3.1.4.2 Adam

There have been many modifications on the gradient descent algorithm resulting in new algorithms that store information from previous updates to adjust the magnitude and direction of the learning step. Adaptive moment estimation (Adam) is widely used [18], and it keeps track of both the average of the previous gradients, as well as the average of the square of the magnitudes of the gradients. The previous magnitudes allow the method to increase the learning step if the algorithm is stuck in a plateau. When the algorithm is near a minimum, however, the gradient rapidly switches sign, and Adam uses this information to shorten the learning step so that it does not miss the minimum.

3.1.5 Performance Measures

The value of the loss function can be used to monitor the training improvement of a model and to validate its final performance. However, it is difficult to interpret, and more often than not, additional evaluation metrics are needed which are more directly connected to the predicted outcome.

For the classification task, a confusion matrix is a straight-forward visual representation of how well a model has succeeded in predicting the patterns to the correct classes. It is illustrated in Figure 3.3.

		Predicted	
		Positive	Negative
Groundtruth	Positive	TP	FN
	Negative	FP	TN

Figure 3.3: Schematic drawing of a confusion matrix for a binary classification problem, where the true predictions can be found on the diagonal. The confusion matrix can also be produced for a general classification problem with n classes. In that case, it will turn into an $n \times n$ matrix with true positives on the diagonal.

The matrix is a table that consists of all patterns divided into the four following categories:

- TP: True positives, the number of patterns that are correctly predicted to belong to the positive class, and should indeed be positive.
- FP: False positives, the number of patterns that are predicted to be positive, but the targets are negative.
- FN: False negatives, the number of patterns that are predicted to be negative, but the targets are positive.
- TN: True negatives, the number of patterns that are predicted to be negative, and should indeed be negative.

From the confusion matrix, additional information can be computed. Three measures that are widely used are [21]:

- *Accuracy*, which measures the overall correctness,

$$\frac{TP + TN}{TP + FP + FN + TN}. \quad (9)$$

- *Recall*, which measures the ratio between the correctly predicted positive samples and all samples that should have been positive,

$$\frac{TP}{TP + FN}. \quad (10)$$

- *Precision*, which measures the correctly predicted positive samples out of all samples that are predicted as positive,

$$\frac{TP}{TP + FP}. \quad (11)$$

3.1.6 Generalization

The usefulness of a model can be determined by the generalization ability of the model, as described in [20]. Generalization measures how well the model performance on new data matches the training performance, where new data is data that the model has not seen during training. Such a dataset can be referred to as a test set, while the dataset used for training is usually referred to as a training set. For the training and test sets to be modelled with the same model, some assumptions are needed. It is assumed that both datasets are sampled from the same underlying distribution, and that the samples are all independent, i.e., that they fulfill the i.i.d. assumption. If the performance of the model is lower on the test set than it is on the training set, then the model is said to be overfitted. If the model severely underperforms even for the training data, then the model is said to be underfitted.

Usually the training is repeated several times using different hyperparameters. At this stage, the training set is split into two sets, where the smaller set, which is called the validation set, is held out during the training [18]. After each epoch the model is evaluated on the validation set. Based on the evaluation result on the validation set, the best set of hyperparameters are selected. This is called model selection. After the model selection step, the best performing models are retrained on the entire training-validation set, and tested on the test set.

To avoid under- and overtraining, the capacity of the model can be changed [20]. An example is the number of nodes in an MLP. With too few nodes, the model cannot approximate the assumed underlying function well enough, resulting in a too low capacity. Adding more nodes will make the capacity of the model increase, but on the other hand it comes with the risk of overfitting. A rule of thumb is to select the model which is the least complex and at the same time performs as good as the set of best models.

An alternative to varying the capacity of the model, if one aims to improve generalization, is to use regularization during training [20]. Two regularization methods will be described in Section 3.1.7 and Section 3.1.8.

3.1.7 Augmentation

The aim of augmentation is to produce more training data by applying randomized transformations on the input [20]. By artificially introducing a larger variation to the training set, a better representation of the underlying data distribution can be achieved, and therefore, the generalization ability of the model is improved. Since

hairy cells are quite rare, the data collected for this project was limited in terms of patient and hospital variations. Therefore, data augmentation was used.

For the task of classification, the model should be transformation-invariant, since an image of a cell will still be a valid cell image after e.g., rotation or translation of the image. For a transformation to be valid, it should be label preserving, i.e., the transformed image should retain the same class belonging as before the transformation.

When comparing how well different models perform, it is important to use identical augmentation settings during training, to make the comparison independent of the augmentation itself.

3.1.8 Dropout

Dropout is a method of regularization, where many subnetworks are created during training, all based on the main network [20]. For each time a pattern is presented during training, the set of nodes to be used are drawn based on the probability p , and the other nodes are "dropped", thereby creating a subnetwork. The hyperparameter p can be different for each layer and node in the network and is set before training. When the trained model is tested on new data, all nodes of the main network are used.

The reason Dropout will regularize training is that, since the nodes are dropped at random, one node can never rely on the presence of another node [20], which means that the nodes will not be able to cooperate to make a jagged decision boundary. A jagged decision boundary is a result of overfitting of the model to the noise in the training data [18].

Dropout is especially useful for models with many nodes, since dropping nodes during training will lower the capacity of the model [20]. Too few nodes in the main model will make underfitting more likely for the submodels. Another advantage of the slimmed submodels is that they will be less computationally expensive to train.

3.2 Convolutional Neural Networks

3.2.1 Introduction

A convolutional neural network (CNN) is a type of ANN, which uses sparse connections and shared weights to extract features from data [20]. Instead of a fully-connected MLP with weights connecting each input node to each output node, a convolutional layer has a kernel, containing only a limited number of weights. The kernel is used for feature extraction and is moved systematically over the spatial dimensions of the image. At each position, the kernel calculates a feature, and all these features are combined into the output feature map.

A convolution, in its discrete form, is defined in [20] as

$$S(i, j) = (K * I)(i, j) = \sum_k \sum_l I(i - k, j - l)K(k, l), \quad (12)$$

where S is the output image, K the convolution kernel, I the input image, and k , l , i and j are indices. The height m_w and width n_w of the kernel are decided in advance, and the number of weights of the kernel is $m_w \times n_w \times d + 1$, where d is the number of channels of the input and the 1 represents the bias [18]. However, each convolutional layer can have several kernels acting on the same input. Thus, the total number of weights of a convolutional layer is $m_w \times n_w \times d \times c + c$, where c is the number of kernels, which also is equal to the number of channels of the output. Since m_w and n_w normally is selected as significantly smaller than the height and width, m and n , of the input image, the total number of weights will be reduced compared to the MLP [20]. After the convolution, a non-linear activation function is used.

The activation is often followed by a pooling step [20]. A pooling step uses a non-trainable kernel to look at limited areas of the input, and outputs one value for each area based on some criterion. The kernel is moved over the input systematically just like the kernel for the convolution. A popular choice is max pooling, which outputs the largest value of each small area which the kernel passes. One main reason to use a pooling after a convolution is that it helps the network to be invariant to minor translations of the detected features. E.g., if an edge is found at one location or found a few pixels to the right might not be important to the larger classification problem.

When moving any kind of kernel over an image, the step size needs to be considered. This is called the stride. If the stride is larger than one, this will reduce the spatial dimension of the output, and this is common to do when using pooling [20]. The dimensionality reduction is a way to condense the detected information before the final classification.

In a CNN there are usually several convolutional layers and pooling steps stacked after each other, after which the output of the last convolution or pooling layer is flattened into a large vector and sent into a small MLP [18]. The output of the MLP

is finally activated by some function to transform the output into probabilities, as described in Section 3.1.2.

It is clear that CNNs are an important machine learning architecture, which solves the main issue of the MLPs, i.e., that an increasingly deepened network leads to an explosively growing number of weights. However, rather than just simple stacking of layers, could there be more advanced kinds of architectures?

3.3 The Inception Architecture

3.3.1 Introduction

The approach of simply adding more convolutional layers has two disadvantages.

With the model getting larger, the number of parameters also increases. This makes the model more prone to overfitting and thus requires a large and detailedly labeled dataset. The size of the available dataset is often a strict limitation in medical imaging tasks, and even if one can acquire the amount of image data needed, it is still laborious work for the expert tasked with labeling the data. Increasing the number of parameters leads to longer computation time, this not only presents a hardware limitation but also bottlenecks throughput.

The Inception architecture was first published in [22], 2014. It is inspired by sparse computation and the Hebbian principle, which states that “neurons who fire together, wire together” [23]. The main idea is to replace fully dense operations inside the convolutions with sparse components that imitate them. In doing so, the authors of the 2014 paper believed that both the performance and the robustness of the network would be improved.

3.3.2 Naïve Implementation

When facing a classification task, the network makes decisions based on features found in the image. Even for a group of images with the same label, the features that mostly define an image as belonging to the correct class can have varying sizes, see Figure 3.4.

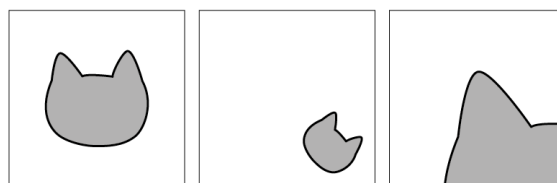


Figure 3.4: To identify Arnold’s cat [24] from the three images, the network would need to detect features of varying sizes.

Traditionally, the convolutional layers of different sizes and pooling layers are stacked upon each other, and the usual sizes are 3×3 , 5×5 , 7×7 . This makes it possible for the network to capture features of different scales. The novel approach in the Inception module is to instead let a series of convolutional kernels in varying sizes run parallelly and concatenate the results into one single feature map that is propagated into the next computational layer [22]. A schematic drawing of the architecture is shown in Figure 3.5. This utilizes parallel computation while keeping the number of parameters down. By allowing the network to choose between different filter sizes, the network adapts itself to detect features of different sizes. During training, the kernels with scales that match the salient features will be activated while the other scales “die out”.

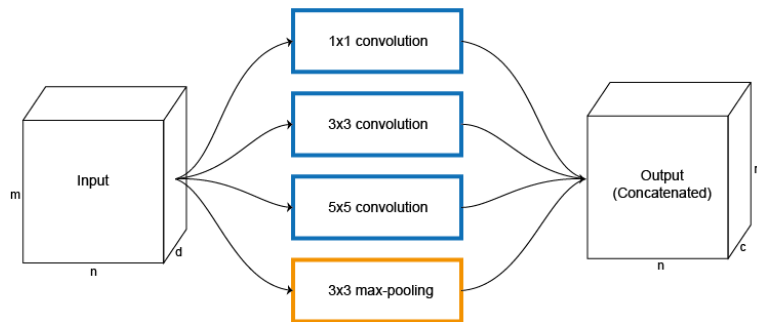


Figure 3.5: *The naïve Inception module architecture.*

3.3.3 The Inception Module

Simply concatenating convolutions of varying sizes results in the naïve implementation. The drawback of this is that each convolution operation requires a large number of multiplications. Let us illustrate this problem with the first Inception module used in the GoogLeNet architecture, an input block that is $28 \times 28 \times 192$ (height \times width \times depth) [22]. The convolutional operation will be done with 32 filters of size 5×5 with "same" padding. The number of multiplication operations in total are

$$(28 \times 28 \times 32) \times (5 \times 5) \times (192) = 120\,422\,400$$

The problem can be remedied by introducing 1×1 convolutions before the spatial convolutions. These serve to capture the depth-wise connections in the input while simultaneously introduce even more sparsity by separating the convolution into two dimensions. An ordinary convolutional operation can be viewed as two operations carried out simultaneously on the input data in two dimensions, the spatial dimension and the depth dimension. By dividing the convolution into two dimensions, the number of parameters can be greatly reduced.

Figure 3.6 shows a 1×1 convolution. It essentially multiplies the information from all the channels for each pixel. The output has the same spatial dimensions as the input, with a depth that equals the number of convolutional kernels. In other words, a 1×1 convolution teaches the network information about cross-channel correlations while reducing the dimension depth-wise.

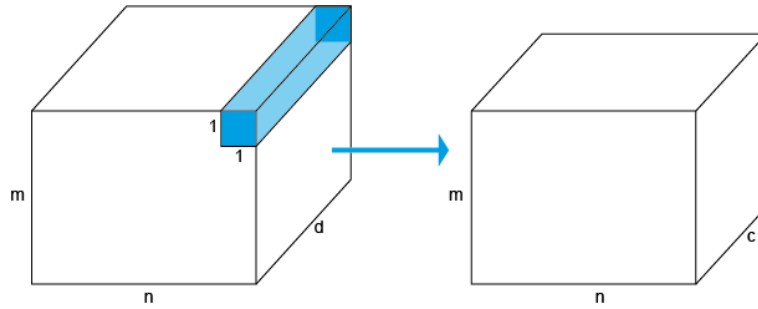


Figure 3.6: Illustration of a 1×1 convolution where the input has height m , width n , and d channels. The output retains the same width and height, but has instead c channels since c convolutional filters are used.

Continuing on the same module, let 16 filters of 1×1 convolutions reduce the dimension of the input block, this results in

$$(28 \times 28 \times 16) \times (1 \times 1) \times 192 = 2\,408\,448$$

multiplication operations.

After the 1×1 convolutions, the spatial convolutions are appended to produce the output with depth 32. For the 5×5 kernel, the number of multiplication operations during this step is

$$(28 \times 28 \times 32) \times (5 \times 5) \times 16 = 10\,035\,200$$

The total number of operations is

$$2\,408\,448 + 10\,035\,200 = 12\,443\,648$$

which is roughly ten times smaller than the number we got when the full convolution was carried out in a single step.

The modified Inception module can be seen in Figure 3.7.

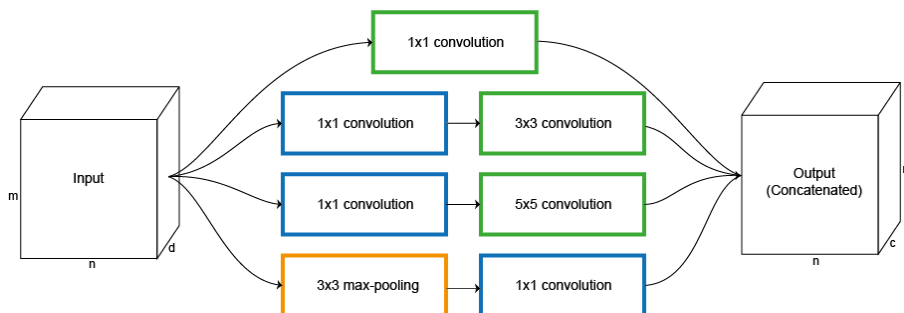


Figure 3.7: The Inception module architecture with the convolution filters and how they are concatenated to produce the output.

3.3.4 Xception

Based on an even stronger hypothesis on the separability of the convolution operation, the Xception module was designed [25]. Such decoupling was already used in Inception where the 1×1 point-wise convolutions were done before the depth-wise convolutions. In Xception, the point-wise and depth-wise convolutions are *extremely* separated, hence the name. The number of such chains of convolutions equals the number of desired output channels. Instead of using several kernels of different sizes, only the 3×3 convolution is used in the Xception architecture. The Xception network slightly outperforms Inception [25], and the architecture of an Xception module can be seen in Figure 3.8.

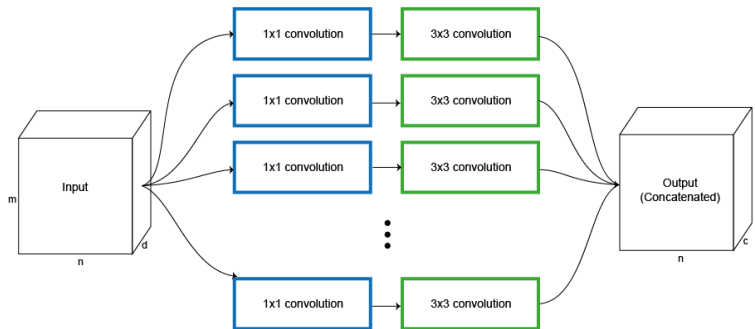


Figure 3.8: Schematic drawing of the Xception module. The chains of convolutions are "extremely" many compared to the Inception module. There are as many such chains as the number of output channels.

3.4 Transfer Learning

Even with the efficient architecture provided by the Xception network, the amount of available data and time resource is still a limiting factor for training. Transfer learning is a method that takes advantage of a network that is pre-trained on a larger, general dataset and applies the knowledge to a different dataset. If the dataset used for the pre-trained network is large enough with many different classes of images, the resulting feature maps serve as general knowledge of image classification and can be applied to other classes than those present in the pre-training dataset.

To adapt the network to the specific task, the top fully connected layers that are customized for the generic task need to be switched out for ones with properties that are better suited for the current task. The size of the output layer and the type of loss function should correspond to the number of classes in the new task. Now, the new top layers are blank slates without any trained weights. If the whole network is trained, the already trained weights of the base model will overpower the ones in the top layers, resulting in trivial weights and thereby sub-optimal training results. For this reason, the base layer weights are set to be frozen during the initial training step and only the top layer weights are trained. This first step allows the network to have a rough knowledge of where the minimum could be located, and fine-searching

is taken care of in the second step. The learning rate is often set to be high in the first step, to ensure that the network converges to the approximate location.

Since the base layer is only trained on the pre-training dataset, their weights need to be adjusted in the second step in order to find a more precise minimum and improve the model performance. Here is how we can think about the procedure intuitively: The differences between the visual representations of the images in the current task and the data used in pre-training would probably have generated different feature maps through the network. By fine-tuning the base network, the feature maps will be adapted to the new task. Due to the large number of parameters present in the base model, this second training step is usually done with very small training step size.

3.5 Contrastive Learning

What if, before applying a classifier, the data is first processed so that samples of the same class are represented with similar coordinates in a latent space?

Contrastive learning uses information on the similarities and differences between images within a dataset to cluster images [26]. First, a copy is made of the input image, and both the original image and the copy are augmented. This random augmentation generates a variation that allows the model to learn the essential information that describes the similarity between the two images. The pair of images are then compared against other images, called the negative samples. The goal of training is to find the representational space in which the positive samples are close together while the negative samples are pushed away.

There are several different approaches developed through the years. Most of the earlier implementations are used for self-supervised learning. The earliest usage of contrastive loss was presented in [27], where the authors used the loss as a dimensionality reduction method prior to a classifier. In semi-supervised tasks, such as the one presented in [10], the model is trained to find the underlying structure of a dataset using the contrastive loss and assigns pseudo-labels to the found clusters. If we have some labeled data, we can train the network to find similarities between the unlabeled samples and the labeled samples and find which classes the unlabeled ones belong to. This is done by contrasting the positive sample in a batch against all other samples in the same batch while maximizing similarity between the positive sample and its augmented counterpart. A classifier is then applied to find the real classes to which the samples belong.

It is later found that contrastive learning could also be applied in a fully labeled scenario, as described in [11]. The label information can be used to further leverage accuracy. In this case, all samples belonging to the same class as the positive sample in a batch are used to attract each other while repelling the negative samples in the same batch. The "clustering" approach is what makes the contrastive learning also appealing for supervised learning. Imagine that the dataset can be described in a N -dimensional space. Each of the classes then forms a data cloud. If we could push

samples belonging to the same class closer together, while simultaneously pushing away the other classes, it would become much easier to draw the decision boundaries. An illustration of how a supervised contrastive model works is shown in Figure 3.9.

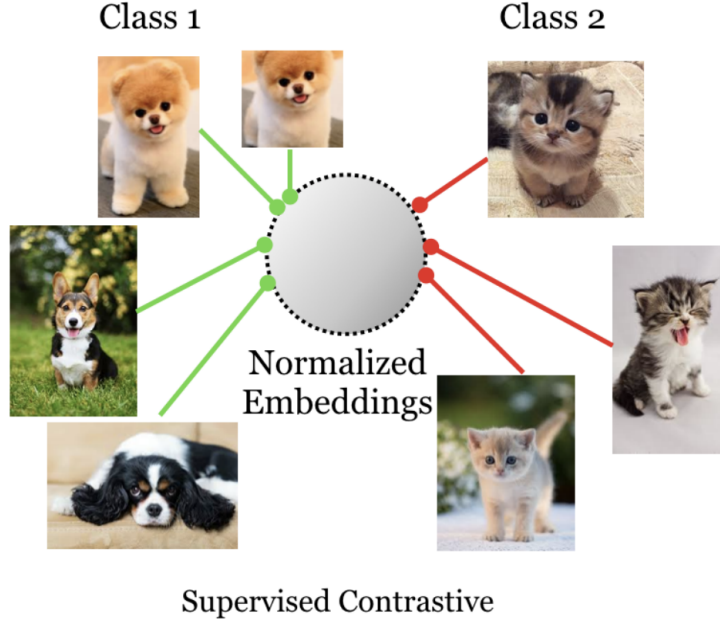


Figure 3.9: The augmented pair of puppy images (the upper left images) and all the other puppies belong to class 1. The objective of the network is to gather the puppies of all breeds while pushing the kittens away. Image used with permission from P. Khosla, author of [11].

In addition to all the building blocks used for transfer learning, two additional ingredients are needed to train a contrastive learning network, namely contrastive loss and pair-wise augmentation.

3.5.1 Supervised Contrastive Loss

The objective of the contrastive loss is to measure the similarity between the representations of two images. In other words, it is a distance measurement between vectors consisting of image features in a batch of images. The supervised contrastive loss function that will be used in this thesis was introduced in [11] as

$$L_{out}^{sup} = \sum_{i=1}^{2M} L_{out,i}^{sup} = \sum_{i=1}^{2M} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a / \tau)}. \quad (13)$$

The sample i is known as the anchor, and z_i is the corresponding vector in the projection space. For each training batch consisting of M images, the loss is calculated on $2M$ pair-wise augmented samples. The set P denotes all positive samples apart from the sample i that the loss is being calculated for. The set A includes all samples in the batch except for the anchor. It can be seen from the equation that if the latent space pushes the positive samples in a batch together, while pushing

away the negative samples, the loss will be small, otherwise the loss will be large. τ is called the *temperature* and is a tunable hyperparameter.

3.5.2 Pair-wise Augmentation

To teach the model the general representation of a given class, it is ideal for each sample from a class to have different views. This can be achieved by applying generous augmentations on the image pairs. In other contrastive applications (such as ones that discern objects in traffic, or dog breeds from one another), views often have the literal meaning of the object seen from different viewing angles. Since blood cells are only viewed from top, the definition of view needs to be broadened. It can then include rotation, translation, zooming, cropping, and tuning of contrast and tone.

3.5.3 Architecture

The following is a description of the architecture and training set-up used in [11]. An encoder is needed to reduce the dimensions in the input image to a representational space. This space still retains many dimensions, and 2048 is the number proposed in the original paper. The calculation of the loss function would be very time-consuming if conducted on such large vectors. Therefore, a projection network is needed for further dimensionality reduction. This network is called the projection head and is attached to the end of the encoder network. It is an MLP with either one hidden layer or none, and an output layer with 128 nodes. The concatenated network is trained with the contrastive loss.

After adequate training, the projection network is discarded, and a classifier is appended to the encoder instead. To keep the training of the weights to the classifier network only, the weights of the encoder are set to frozen. After already finding a good representational space during the first training step, the task of the new network is to assign the correct labels to samples. Therefore, it is trained with categorical cross-entropy loss instead of the supervised contrastive loss. The size of the output from the encoder network as well as the output layer from the projection head could largely depend on the dataset and the number of classes studied and will be investigated in this project.

4

Methodology

If you ignore technology, produce demons [1].

4.1 Collection of Cell Images

The slides and datasets were provided by CellaVision. We were able to collect slides that have hairy cell occurrence observed in laboratory. These slides were then run on the CellaVision DC-1 system and the cell images were saved on a database called "hairy_db". In total, 11 slides containing hairy cells originating from three hospitals were acquired.

For each run of a slide the system was set to count 1000 WBCs and stop the scanning once the threshold was reached. For two of the slides the system was unable to find enough WBCs and the analysis was automatically stopped. The system was set to pre-classify 19 standard WBC classes, i.e., not including hairy cells. All scanned images were stored in a database along with the pre-classification labels.

The cell images should be stored in both .bmp and .jpg formats, where the .bmp format contains more information while the .jpg format is compressed and normalized. Due to a corruption in the database, we lost many of the .bmp images, and had to resort to using the .jpg images instead.

It is vital to ensure that all cells, including the hairy cells, receive the correct ground truth labeling. Therefore Steven Marionneaux, expert in hematology, was consulted to examine all the cells in the database and assign them with the correct labels.

To complement the collected dataset with cells from other cell classes, additional cell images were extracted from earlier runs on other slides. These are stored in the "WBCTrain" database at CellaVision. We were able to use an internal tool to select which cells that should be used. For each of the cell classes, up to 5000 cells were collected, depending on the availability of cell images. Some of the classes, e.g., promyelocytes and plasma cells were very rare, hence all the cells belonging to

these classes across the whole database were collected.

For the classes lymphocytes, abnormal lymphocytes, and reactive lymphocytes, the criteria for labeling at collection time does not necessarily align with our goal which is that hairy cells should belong to their own class. Manual examination of images of these three classes were thus conducted to remove possible occurrences of hairy cells.

In the model selection phase only the hairy cells were used from "hairy_db", and not the cells from the other 19 standard classes. However, after suspicions of the model learning to recognize system and software specific information of the images from this database, and not only the morphology of the hairy cells, images of all cell classes from "hairy_db" were added to the training set for the final retraining of the model on all training and validation data.

The total numbers of cell images used for training and validation are displayed in Table 4.1.

Table 4.1: *The table contains the number of cell images used for the final training on all available data. Note that these numbers include cells from both the databases "WBCTrain" and "hairy_db", which is why some cell classes exceed the 5000 number limit.*

Number of cells in each class for the final training			
Segmented neutrophil	6542	Plasma cell	116
Eosinophil	5063	Hairy cell	4279
Basophil	2512	Smudge cell	6762
Lymphocyte	5983	Erythroblast (NRBC)	4667
Monocyte	5094	Artefact	4515
Band neutrophil	3530	Giant thrombocyte	994
Promyelocyte	365	Thrombocyte aggregation	1660
Myelocyte	3561	Reactive lymphocyte	4697
Metamyelocyte	2070	Abnormal lymphocyte	4201
Blast (no lineage spec)	5001	Large thrombocyte	5000
Total:			76612

4.1.1 Training, Validation, and Test Split

Since we did not have access to more hairy cells than the collected 11 slides, two of the slides were put aside into a holdout test set.

For the other 19 cell classes, a database called "WBCTest" that is used for internal validation of neural network performance at CellaVision is used. These images are collected from different slides than the ones in "WBCTrain", hence the usage of

them for testing will ensure that the generalization ability of the model is evaluated. In Table 4.2 are the number of cell images used for testing.

Table 4.2: *The number of cell images used for testing can be seen in the table.*

Number of cells in each class for testing			
Segmented neutrophil	1135	Plasma cell	185
Eosinophil	1113	Hairy cell	1453
Basophil	707	Smudge cell	1171
Lymphocyte	1137	Erythroblast (NRBC)	409
Monocyte	1094	Artefact	1047
Band neutrophil	524	Giant thrombocyte	121
Promyelocyte	265	Thrombocyte aggregation	221
Myelocyte	425	Reactive lymphocyte	1277
Metamyelocyte	195	Abnormal lymphocyte	610
Blast (no lineage spec)	1431	Large thrombocyte	854
Total:			15374

The validation split is done during training using the built-in *split* argument from the Tensorflow data.dataset module [28]. To prevent the data from being shuffled differently during each run, a seed is set, and the split ratio between the training and validation sets was constant at 0.8:0.2.

4.1.2 Expert Classification

"WBCTrain" and "WBCTest" have previously been labeled by several experts, although all slides have not been labeled by all experts. For "WBCTrain" we found that using images labeled by one expert or more, where all experts are in agreement, resulted in a dataset with acceptable ground truth. Yet, when using the same configuration on "WBCTest", there were several severe expert misclassifications, such as segmented neutrophils being labeled as lymphocytes. We therefore decided to be even more critical of the expert labels in the test set, and to only use images with at least two or more experts, of whom all agree.

4.2 Transfer Learning

4.2.1 Model Creation

The model used for evaluating transfer learning is implemented with adaptations from [29], and the architecture can be found in Appendix B.1.

First, an input layer receives the images and passes them on to the rest of the network. The second layer receives the input and applies online-augmentation.

Between the input and the base model, a normalization layer is added which scales the input to be between -1 and 1. This is because the pre-trained weights in Xception only accept inputs in this range.

After augmentation, the images are ready to be processed by the base model. As mentioned in section 3.4, the model should be built upon a pre-trained base model (with its top layers removed) that already contains information on feature maps for image classification. Since this work focuses on the comparison between the transfer and contrastive learning methods, we limited ourselves to using one base model for all experiments, and selected the Xception model for its performance. We let the base model initialize with weights obtained from training on the ImageNet dataset [30].

After the base model, a global average pooling layer is needed to extract a feature map for each of the classes. The global average pooling layer acts like a fully connected layer but adapts better to CNNs. A dropout layer with 0.2 dropout strength is added. The outputs are then passed on to the top layers.

Our task has 20 classes while the original ImageNet task has 1000 classes [25]. Therefore the top layers from the original Xception architecture were removed and a new top layer with size 20 was appended. It is worth noting that it is possible to add hidden layers to the top architecture. However, after experimenting with different architectures, we arrived at the conclusion that a single dense layer produced the best results.

4.2.2 Metrics

The metrics that are monitored are shown in Table 4.3.

Table 4.3: *Metrics used in transfer learning.*

Metrics	Training	Validation	Test
Categorical cross-entropy loss	•	•	•
Accuracy	•	•	•
Recall		•	•
Precision		•	•
Confusion matrix		•	•
Scatter plot		•	•

A 20×20 confusion matrix is plotted to assess the performance of the model for each cell class. Since one part of the objective is to discern hairy cells from the other lymphocytes (normal, abnormal, and reactive), special consideration is taken

to evaluate the performance of the model on these classes. During the test phase, another simplified confusion matrix is plotted, in addition to the full matrix. In the simplified version, the four lymphocyte classes are presented, while all other cell classes are assigned to the category “Other”. This smaller confusion matrix helps in understanding whether the hairy cells and the other lymphocyte classes get mixed up with each other or with non-lymphocyte classes.

To view the effect of the contrastive learning, a scatter plot of the output of the global average pooling, i.e., the part that corresponds to the encoder in the contrastive model, is created. The output is l_2 -normalized and a dimensionality reduction is performed to be able to plot the space in 2D.

4.2.3 Training

During the first training step, the initialized base model weights are set to be frozen. This way the only trainable weights are the ones in the top layers. The loss function is monitored during training, and the number of epochs are selected to let the model train until the validation performance stagnates.

Since the weights of the base model have not been trained on the images used specifically for this task, some fine tuning is needed for it to find better features. The weights are unfrozen, and the whole model is trained.

Both training steps are monitored using the cross-entropy loss and accuracy.

4.2.4 Model Selection

The hyperparameters are selected based on the metrics and the loss/accuracy graph obtained during training, as well as the confusion matrices. For simplicity, all models including the ones described in Section 4.3, will be named HAL-XXXXX (Hairy abnormal lymphocyte network), where the first letter after the hyphen denotes whether it is a transfer learning (T), contrastive learning (C), or a randomly initialized (R) model, while the last four digits act as a distinguisher. The hyperparameters for the three best performing transfer models can be seen in Table 4.4.

4.2.5 Data Augmentation

To achieve better generalization performance, augmentation layers are added to each model. For transfer learning, this is the only augmentation step. The augmentation is done online which means that each sample will be randomly augmented within the given parameters. The morphology of blood cells depends largely on the size, so zooming was limited to a small amount only. Furthermore, it is important to take care when using color transformations, since the stains used have very distinct color palettes. Thus, doing hefty color transformations might lead to the training data

Table 4.4: *Selected hyperparameters in transfer learning, excluding augmentation settings.*

Hyperparameter	HAL-T3001	HAL-T3002	HAL-T3003
Top layer epochs	10	6	15
Fine tuning epochs	4	5	5
Top layer learning rate	10^{-3}	10^{-3}	10^{-3}
Fine tuning learning rate	10^{-4}	10^{-4}	10^{-4}
Dropout	0.2	0.2	0.2
Batch size	20	25	25

not being representative of future data [31].

The augmentation is performed using `tf.keras.layers` [32], and the different settings are:

- Random contrast: The contrast is calculated as $(x - \mu)c + \mu$, where x is the input pixel contrast, μ the image pixel mean, and c is a factor drawn from the interval $[1.0 - C, 1.0 + C]$, where C is the contrast factor chosen by the user.
- Fill mode reflect: When an image is moved in any way such that parts of image no longer contain any pixel information, these parts will be filled by reflecting the content of the nearest pixels.
- Random rotation: The rotation is drawn from the interval $[-2\pi r, 2\pi r]$, where r is the rotation factor.
- Random translation: The translation factor, t , is the bound of the percentage of the input height or width with which the output image is shifted. The shifts can be either upwards, downwards, to the left or to the right depending on the sign that is assigned to the randomized values. We select the translation bounds to be the same in both axes so only one factor will be present.
- Random zoom: The zooming factor z is the bound of the percentage of the input image size with which the output image will be enlarged or shrunken, i.e., the image is zoomed in the range $[-z, z]$ where a negative factor means shrinking.

The augmentation setting used for transfer learning can be seen in Table 4.5.

4.2.6 No Transfer Learning: Model with Randomly Initialized Weights

As a comparison, a model was trained with identical architecture as HAL-T but with randomly initialized weights instead of the ones obtained from ImageNet training.

Table 4.5: *Selected augmentation settings in transfer learning.*

Augmentation	Classifier
Contrast C	0.1
Fill mode	reflect
Rotation r	0.5
Translation t	0.05
Zoom z	0.05

This model will be referred to as HAL-R. When setting up the model, no weights are loaded and an "empty" model is compiled. By doing so, the entire model is trained from scratch, and both top layers and the base model are trained simultaneously. Except for the weight initialization, most hyperparameters are kept to be the same as the ones used for the model with best performance, as can be seen in Table 4.6. Since there is no separate top layer training, nor fine tuning, there is only one learning rate and one number of epochs. To ensure that the model will converge or stop improving in a reasonable amount of time, early-stopping is used. If the loss has not improved in 3 epochs, the training will be stopped.

Table 4.6: *Selected hyperparameters in the randomized initialization training, except augmentation settings. The augmentation settings were the same as for transfer learning, Table 4.5.*

Hyperparameter	HAL-R
Epochs	100
Learning rate	10^{-3}
Dropout	0.2
Batch size	25

4.3 Contrastive Learning

4.3.1 Model Creation

The implementation of the contrastive training and loss is based on two previous works. From [33], a first iteration of the loss function was created. Some minor tweaks to the loss function were added, inspired by [11]. The structure of the architecture and the training pipeline are also influenced by [11] with adaptations to integrate with the available software and hardware used for this project. The model, HAL-C, consists of an encoder, a projection head, and a classifier, and all architectures can be found in Appendix B.2 - Appendix B.5.

The encoder network is very similar to the transfer learning architecture from the

input to the global average pooling layer, except that it does not include the augmentation layer. The base model for the encoder is still the Xception module pre-trained on ImageNet. The encoded space will thus be the output of the global average pooling layer, and will be a 2048-dimensional space.

The projection network consists of an input layer, an online augmentation layer, the encoder, and the projection head. As recommended by [11] both the encoded features, before being sent into the projection head, and the output of the projection head are l_2 -normalized. The projection head is chosen to be an MLP with a dropout layer and a single dense output layer. The output is an N -dimensional space, where N is a hyperparameter. The projection model had to be implemented as a subclass of Tensorflow's `keras.Model` for it to be able to handle the contrastive training step, and this implementation is loosely based on [34]. The reason is that the contrastive training step includes a doubling of the data before data augmentation in each batch to create the two views of the same image, which is not part of any standard training packages.

The classifier network has a similar architecture as the projection network, but with some differences. The most important being that the classifier network does not use l_2 -normalization on the output layer, and that the number of output nodes corresponds to the number of classes. Other differences are that the augmentation will have slightly milder settings, and that there of course was no doubling of data in the training steps.

4.3.2 Metrics

For the contrastive training almost the same metrics were studied as for the transfer learning. The metrics can be seen in Table 4.7.

Table 4.7: *Metrics used in contrastive learning.*

Metrics	Training	Validation	Test
Supervised contrastive loss	•	•	•
Categorical cross-entropy loss	•	•	•
Accuracy	•	•	•
Recall		•	•
Precision		•	•
Confusion matrix		•	•
Scatter plot		•	•

4.3.3 Training

Previous work in [10] and [11] have pressed upon the importance of larger batch sizes during training of contrastive networks. However, due to the large size of Xception, and the limited GPU memory, the batch size was limited to 16, and could not be explored further.

Since pre-trained Xception weights are used in this project, the training can be seen as a combination of transfer and contrastive learning. In the initial training experiments, firstly the projection network was trained, followed by training of the classifier network. However, the initial training of the projection network did not work as expected. The input was projected down to practically a single point in the projection space, and even sometimes to the zero vector, which naturally made training of the classifier impossible, since it had no means of distinguishing between the classes. The issue was solved by this three-step training procedure:

1. Train the projection network with the projection head weights unfrozen, but with the encoder weights frozen.
2. Train the projection network with both the projection head and encoder weights unfrozen.
3. Train the classifier network with the classifier weights unfrozen, but with the encoder weights frozen.

The three-step training is illustrated in Figure 4.1.

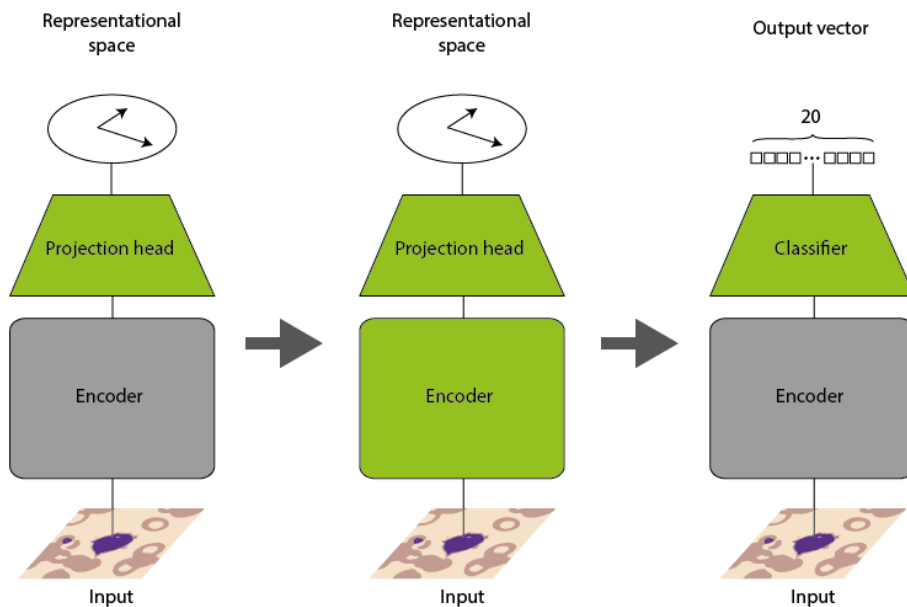


Figure 4.1: *Three-step training procedure. The green color means that the weights are trainable, while the grey color means that the weights are frozen.*

4.3.4 Model Selection

When the training procedure was established, a hyperparameter search was initiated. The resulting hyperparameters can be seen in Table 4.8 and Table 4.9 for the projection networks and the classifier networks respectively. The projection networks and the encoders will be referred to as projection-head-X and encoder-X, where X corresponds to the number of projection head units used in the training.

Table 4.8: *Selected hyperparameters for the projection networks in contrastive learning. Note that the settings for projection-head-16 was changed slightly from the training with validation-split to the final training using all training and validation data. This means that the validation and the test results are produced with somewhat different hyperparameters. The hyperparameters used for the validation results can be found in Table C.1 in Appendix C, whereas the hyperparameters shown here were used in the final training. The reason for the change is that it was believed already in the model selection that these hyperparameters would be better, but due to the long training time no retraining was performed.*

Hyperparameter	projection-head-16	projection-head-128
Projection head epochs	7	7
Encoder + projection head epochs	10	10
Projection head learning rate	10^{-3}	10^{-3}
Encoder + projection head learning rate	10^{-5}	10^{-5}
Dropout	0.2	0.2
Batch size	16	16
Temperature	0.1	0.1
Projection head units	16	128
Encoder base	encoder-16	encoder-128

Table 4.9: *Selected hyperparameters for the classifier networks in contrastive learning.*

Hyperparameter	HAL-C3001	HAL-C3002	HAL-C3003
Classifier epochs	20	20	20
Classifier learning rate	10^{-3}	10^{-3}	10^{-3}
Dropout	0.3	0.2	0.3
Batch size	16	16	16
Encoder base	encoder-16	encoder-128	encoder-128

4.3.5 Data Augmentation

The same sorts of augmentation as in transfer learning were used for contrastive learning, but with other hyperparameters. The augmentation settings for the pro-

jection network and the classifier network can be seen in Table 4.10.

Table 4.10: *Selected augmentation settings for contrastive learning. The table contains the settings both for the projection networks and the classifier networks.*

Augmentation	Encoder/Projection Head	Classifier
Contrast C	0.5	0.1
Fill mode	reflect	reflect
Rotation r	0.5	0.5
Translation t	0.1	0.05
Zoom z	0.1	0.05

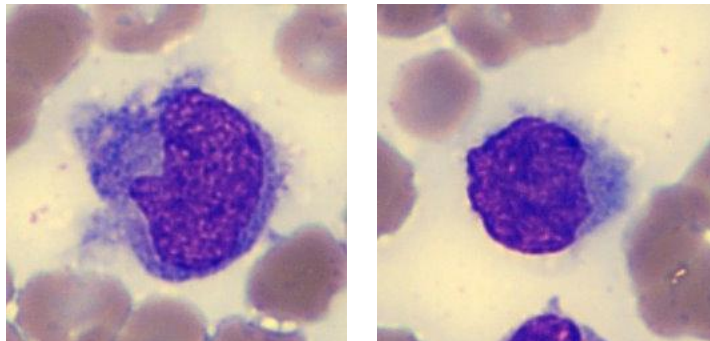
4.4 Sanity Test

In the initial training, hairy cell was the only class used from the database "hairy_db". Let us call this set A. The other 19 cell classes were taken from the database "WBCTrain". This set will be referred to as B. This led to models with 1.0 recall and 1.0 precision for hairy cells in the test set, and arose suspicions of the models only learning to recognize the difference in appearance between the databases, and not the actual morphology of hairy cells. Therefore a sanity test consisting of two parts was conducted.

A new test set, C, was formed of the 19 other cell classes, but this time only from "hairy_db". In part 1, the models trained on the original training set, which consists of data from both A and B, were tested on a C-test set. The models were then retrained on a new training set with data from A, B and C. These models were again tested on the C-test set.

In part 2, a fourth set was formed called D. D was composed of the possible hairy cells that had been removed from "WBCTrain" and "WBCTest" in the data collection phase. These images have not been confirmed as hairy cells, but their morphology show clear resemblances to that of a hairy cell. Examples of such images can be seen in Figure 4.2. Again, the models were first trained on A and B and tested on D, and then retrained on A, B and C and tested on D.

Note that all other test results presented in this report are produced using a training set with data from A, B and C, and tested on a test set with data from A and B. The validation results are produced by doing a validation split on data from A and B. Only this section uses the data in D. A summary of the four datasets can be seen in Table 4.11.



(a) *Large hairy lymphocyte* (b) *Small hairy lymphocyte*

Figure 4.2: *Abnormal lymphocytes with morphologies similar to those of hairy cells. Images belong to dataset D.*

Table 4.11: *A summary of the datasets used in the sanity test. Note that A, B, C and D all have separate training and test subsets.*

Dataset	Description
A	Hairy cells from "hairy_db".
B	Cells from the other 19 classes from "WBCTrain" or "WBCTest", depending on if it is a training set or a test set.
C	Cells from the other 19 classes from "hairy_db".
D	Cells with morphologies similar to hairy cells from "WBCTrain" or "WBCTest".

It's ultimately about ambitions [1].

5.1 Transfer Learning with Xception

In Table 5.1, the results of the training of HAL using transfer learning, HAL-T, is presented. Three different hyperparameter combinations were selected for the final retraining using all training data, HAL-T3001, HAL-T3002, and HAL-T3003. Both the validation and test accuracy and loss can be seen in the table.

Table 5.1: *Results for the three best transfer learning models.*

Model	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
HAL-T3001	0.9087	0.2811	0.8766	0.4598
HAL-T3002	0.9057	0.2921	0.8821	0.4200
HAL-T3003	0.9056	0.3021	0.8817	0.4297

For the best model, HAL-T3002, Figure 5.1 shows the full confusion matrix with all classes. The lymphocyte-specific confusion matrix can be found in Figure 5.2. Figure 5.3 shows the dimensionality reduction of the encoded space for HAL-T3002. The dimensionality reduction was performed using t-SNE.

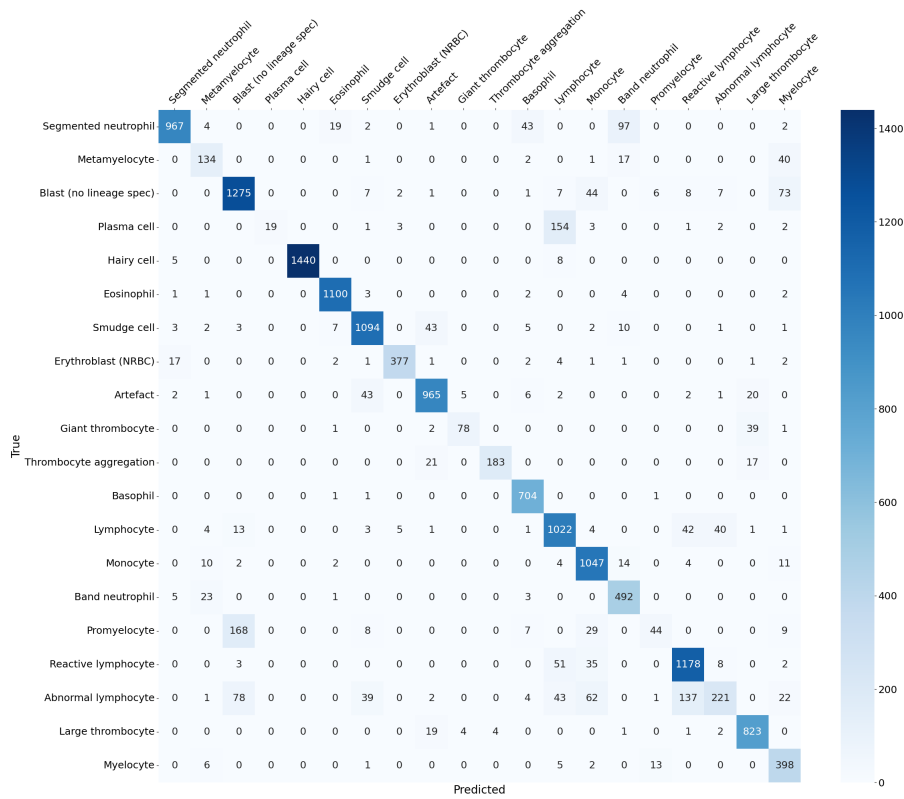


Figure 5.1: Confusion matrix for HAL-T3002, which is the best performing transfer learning network.

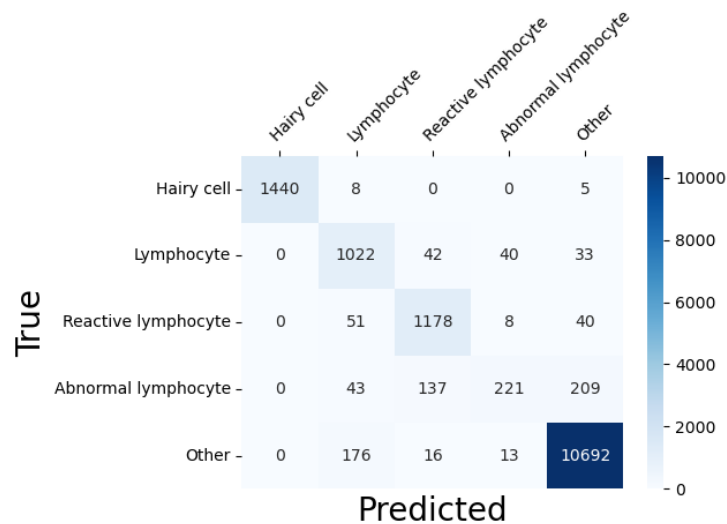


Figure 5.2: Confusion matrix with only the lymphocyte classes for the best transfer learning network, HAL-T3002. The non-lymphocyte classes have been forwarded to the label "Other".

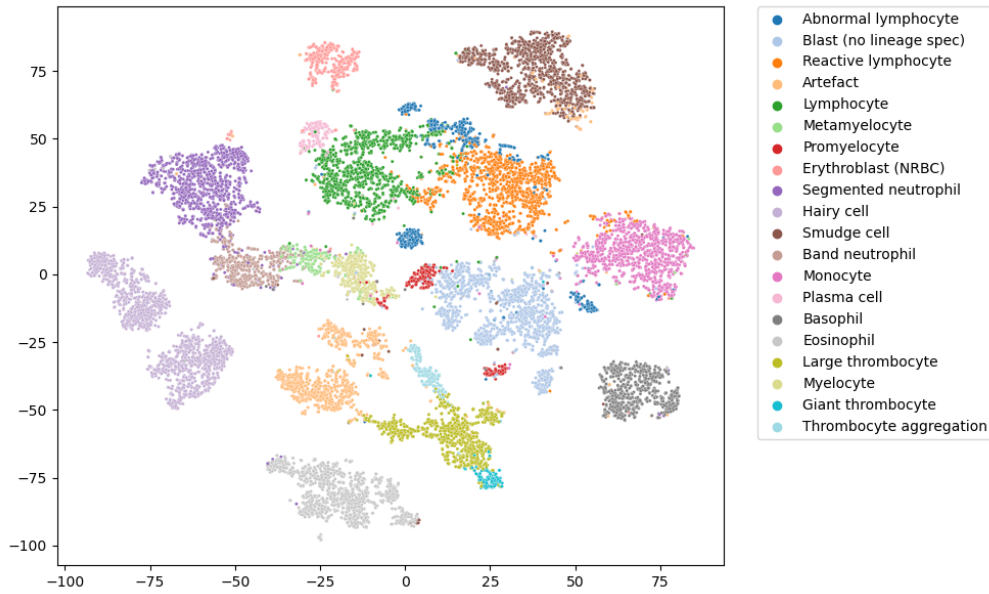


Figure 5.3: *t-SNE* dimensionality reduced scatter plot of the encoded space for *HAL-T3002* used on the test set.

5.1.1 Randomly Initialized Model

The model that was initialized with random weights, instead of ImageNet pre-trained weights, reached the early-stopping criterion and stopped training after 11 epochs. The accuracy and loss for both training and test of the HAL-architecture with random initialization, HAL-R, can be seen in Table 5.2. The confusion matrix with all 20 classes can be found in Figure 5.4. In Figure 5.5 the encoder-space is visualized using a scatter plot and t-SNE.

Table 5.2: Results for *HAL-R*, which is the network initialized with random weights.

Model	Training Accuracy	Training Loss	Test Accuracy	Test Loss
HAL-R	0.08778	2.837	0.07617	2.889

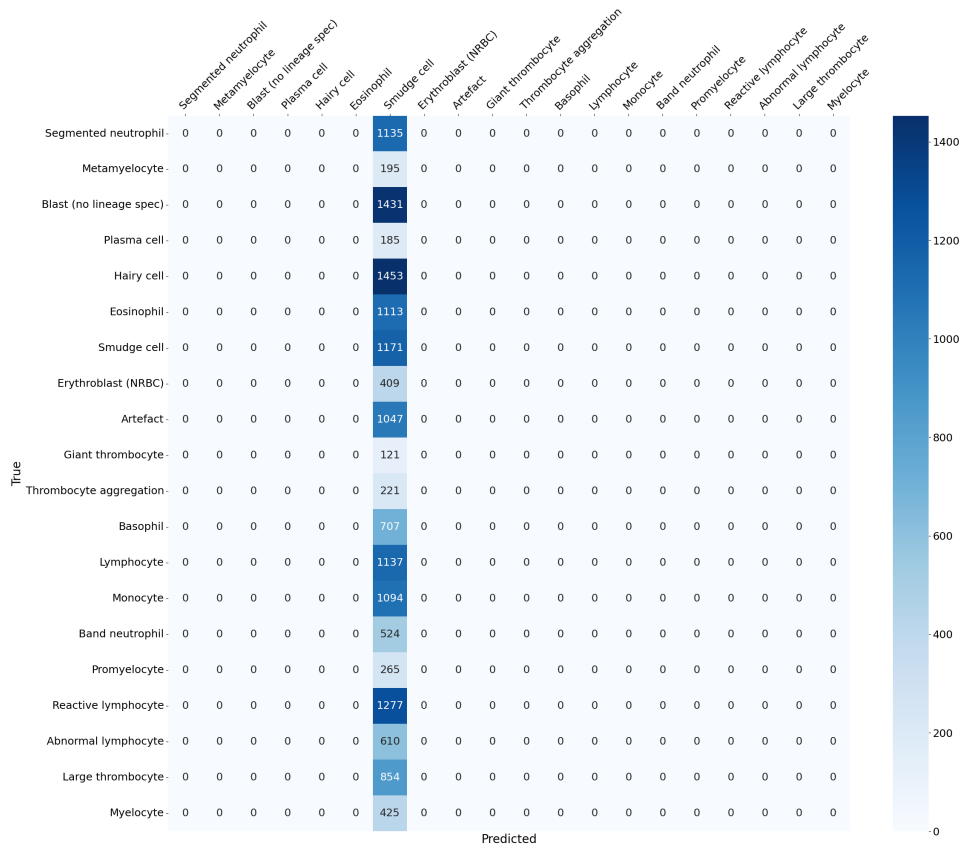


Figure 5.4: Confusion matrix for the model with randomized weight initialization, HAL-R.

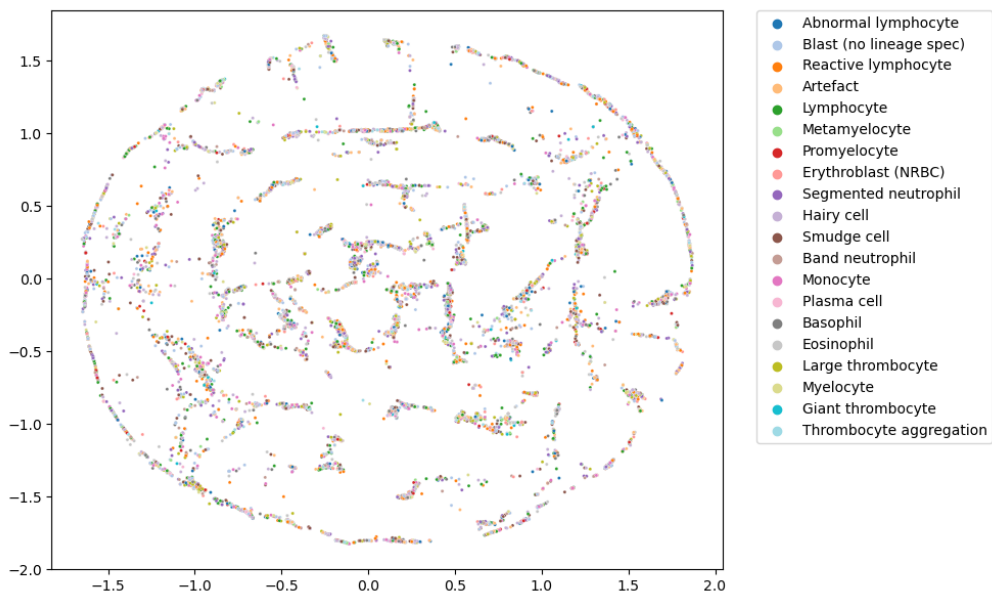


Figure 5.5: Dimensionality reduction using t-SNE on the encoded space of the model with randomized weight initialization, HAL-R. There are no visible clusters of the same color. All cell classes are rather mixed up.

5.2 Contrastive Learning

The accuracy and loss on validation and test for the contrastive models, HAL-C, can be seen in Table 5.3. Three models are presented, where HAL-C3001 has $N = 16$ projection head units, and the other two, HAL-C3002 and HAL-C3003, have $N = 128$. Of the three models, HAL-C3003 has the best performance.

Table 5.3: *Results for the HAL-C models. Note again that the validation and test results for HAL-C3001 are obtained with slightly different hyperparameters as explained in Table 4.8.*

Model	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
HAL-C3001	0.9129	0.2660	0.9004	0.3550
HAL-C3002	0.9186	0.2546	0.9020	0.3518
HAL-C3003	0.9184	0.2551	0.9024	0.3462

The full confusion matrices with all cell classes for HAL-C3001 and HAL-C3003 can be found in Figure 5.6 and Figure 5.7, and the condensed, lymphocyte-only confusion matrices can be seen in Figure 5.8 and Figure 5.9. The dimensionality reductions of the encoded space for HAL-C3001 and HAL-C3003 can be seen in Figure 5.10 and Figure 5.11.

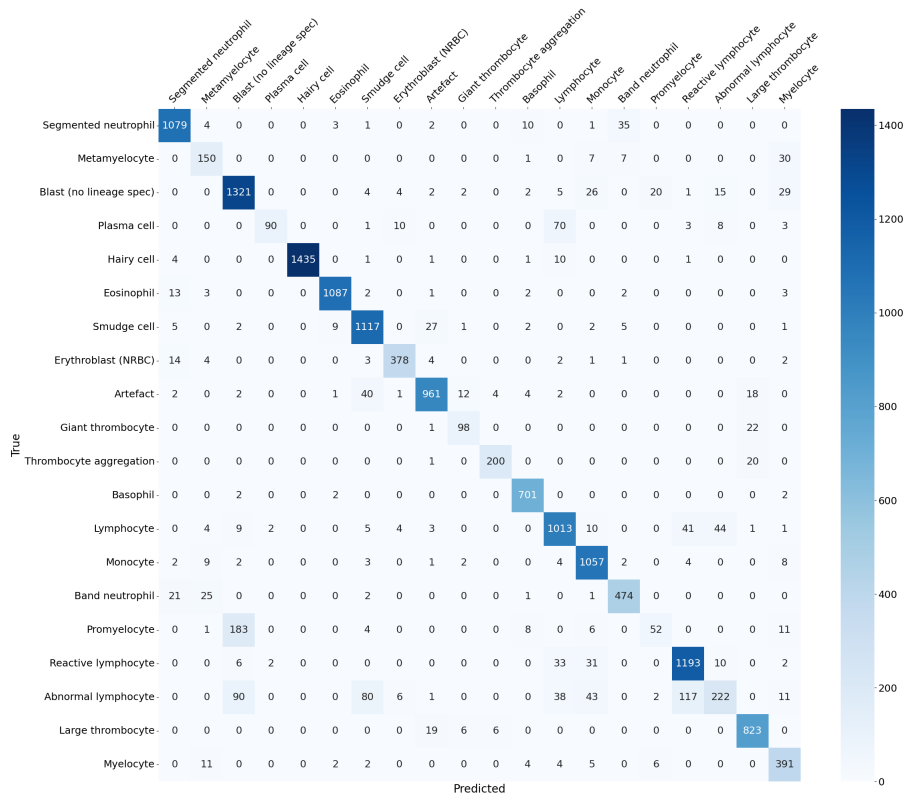


Figure 5.6: Confusion matrix for the contrastive learning model with 16 projection head units, HAL-C3001.

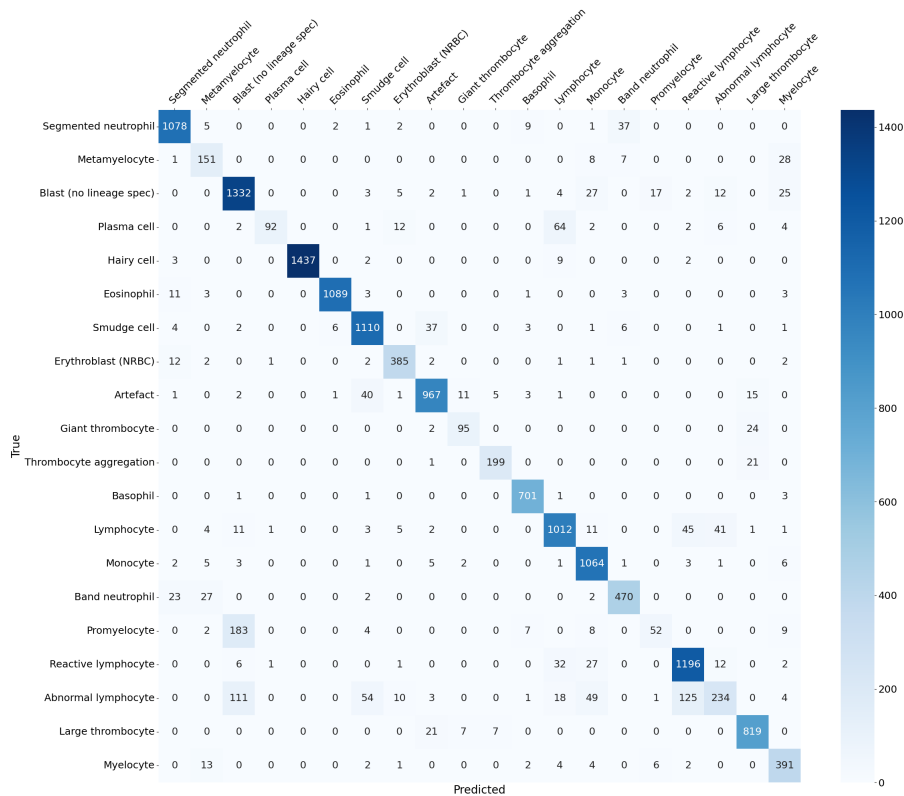


Figure 5.7: Confusion matrix for the best contrastive learning model with 128 projection head units, HAL-C3003.

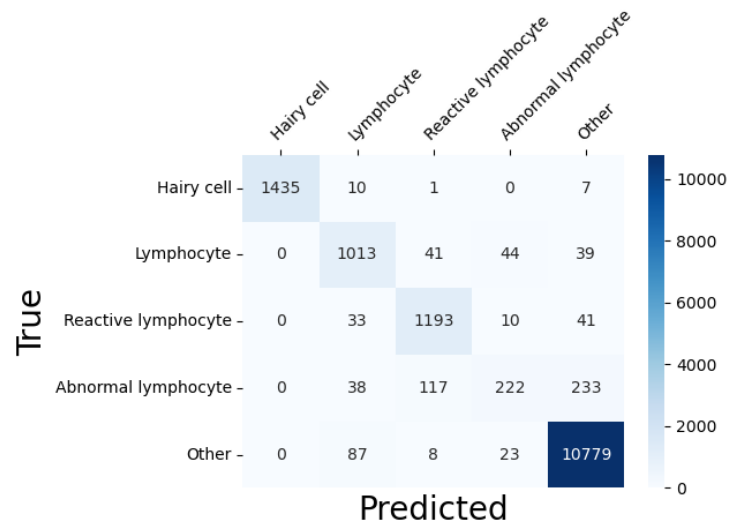


Figure 5.8: Confusion matrix with the lymphocyte classes for HAL-C3001.

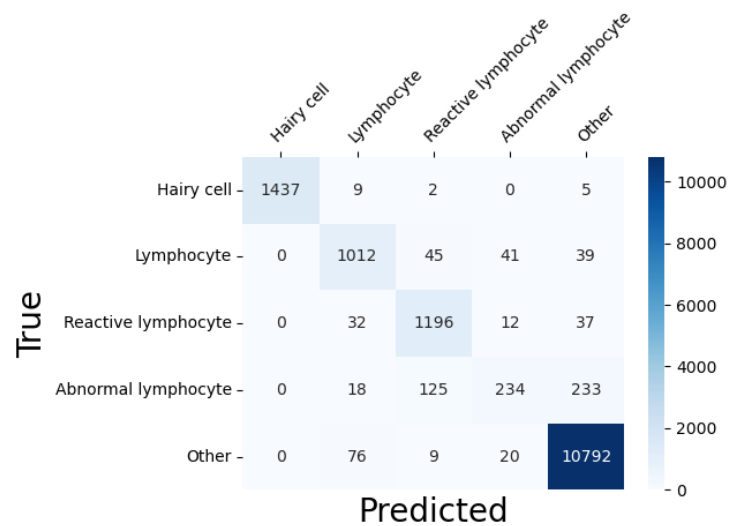


Figure 5.9: Confusion matrix with the lymphocyte classes for HAL-C3003.

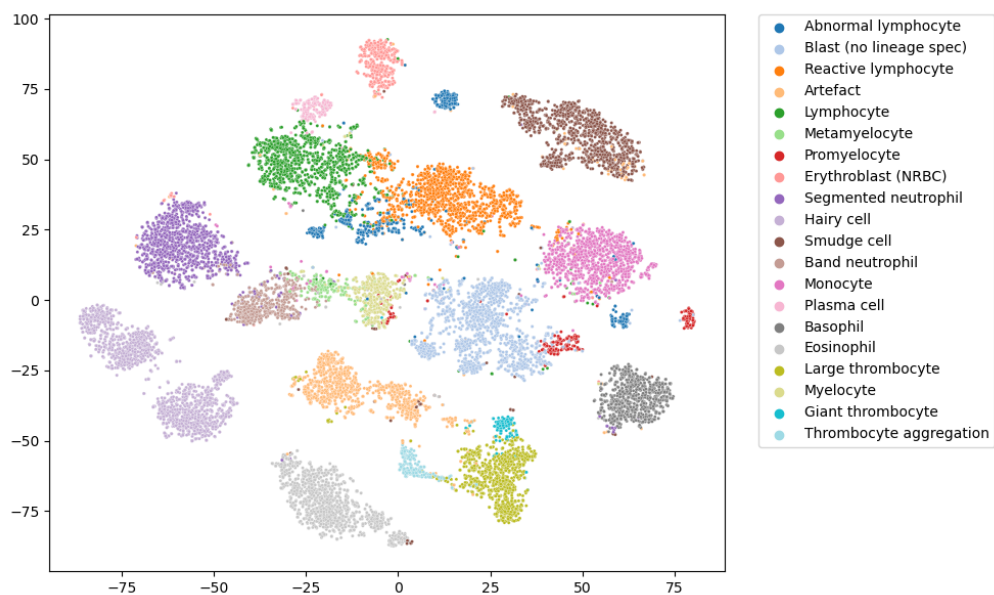


Figure 5.10: Dimensionality reduced scatter plot of the encoded space of HAL-C3001. Produced with *t-SNE*.

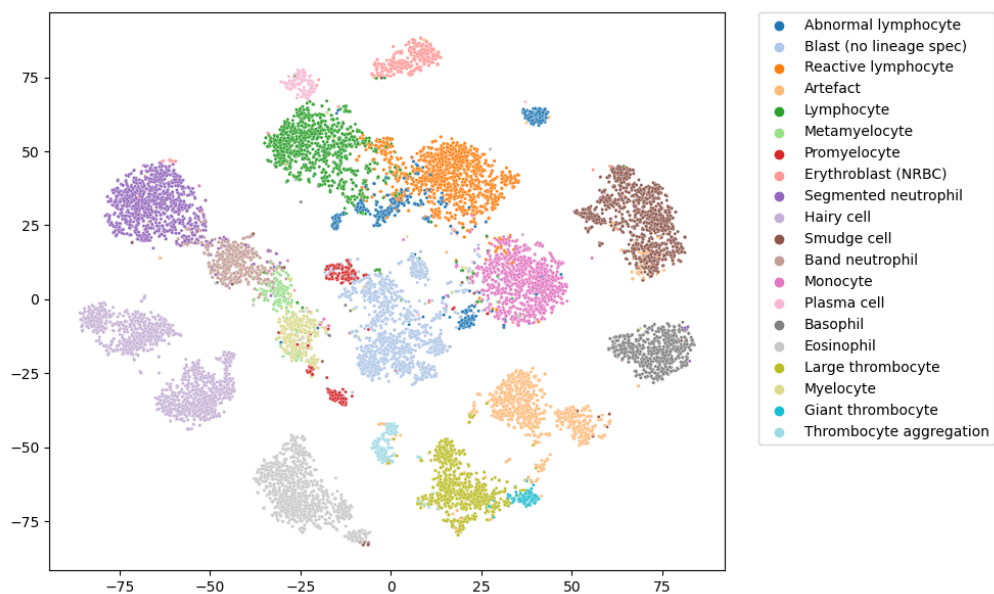


Figure 5.11: Dimensionality reduced scatter plot of the encoded space of HAL-C3003. Produced with *t-SNE*.

5.3 Sanity Test

The part 1 results on the C-test set can be seen in Table 5.4, and the results for part 2 on the D-test set can be seen in Table 5.5. For both tables the test accuracy when training on A and B can be seen in the first column and the test accuracy when training on A, B and C can be seen in the second column. All networks in the sanity test are trained using transfer learning.

Table 5.4: Results for part 1 of the sanity test. The accuracy is shown for a test set with data from C. For the first accuracy the networks are trained on data from A and B, and for the second accuracy the networks are trained on data from A, B and C. The networks are trained using transfer learning.

Model	Accuracy tested on C	
	Trained on A & B	Trained on A, B & C
HAL-T3001	0.6138	0.9116
HAL-T3002	0.4836	0.8825
HAL-T3003	0.2974	0.9212

Table 5.5: Results for part 2 of the sanity test. The accuracy is shown for a test set with data from D.

Model	Accuracy tested on D	
	Trained on A & B	Trained on A, B & C
HAL-T3001	0.07407	0.1852
HAL-T3002	0.07407	0.1481
HAL-T3003	0.03704	0.1852

For model HAL-T3003 four confusion matrices were produced for the four different training-testing configurations. The matrices for part 1 of the sanity test can be seen in Figure 5.12 and Figure 5.13, and the matrices for part 2 can be seen in Figure 5.14 and Figure 5.15.

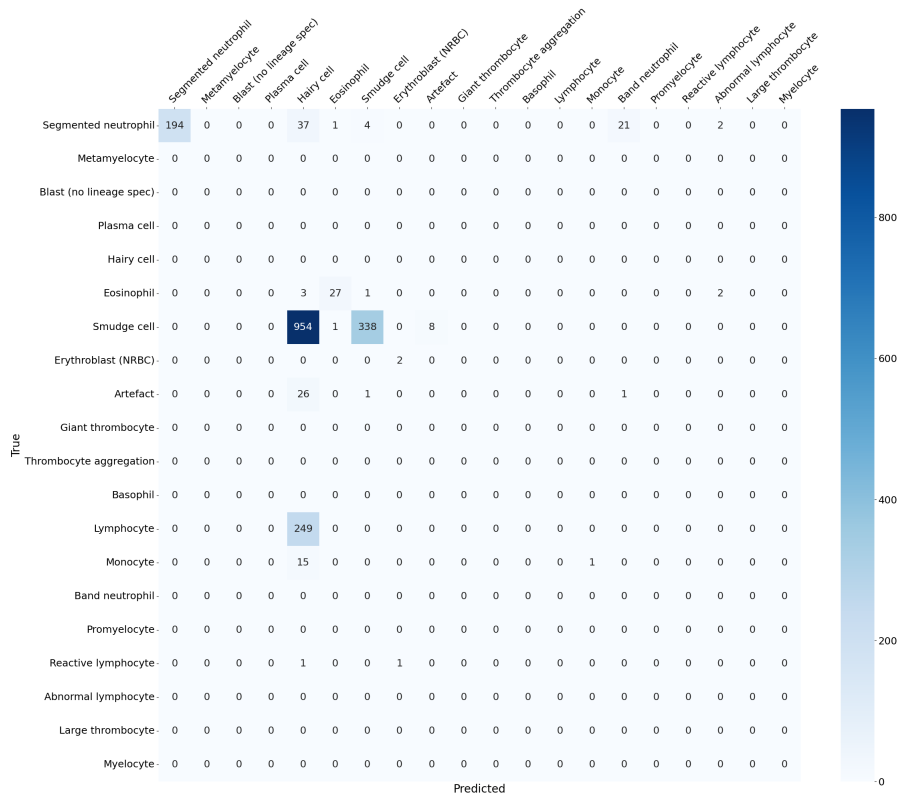


Figure 5.12: Confusion matrix for the model HAL-T3003 trained on sets A and B, and tested on set C.

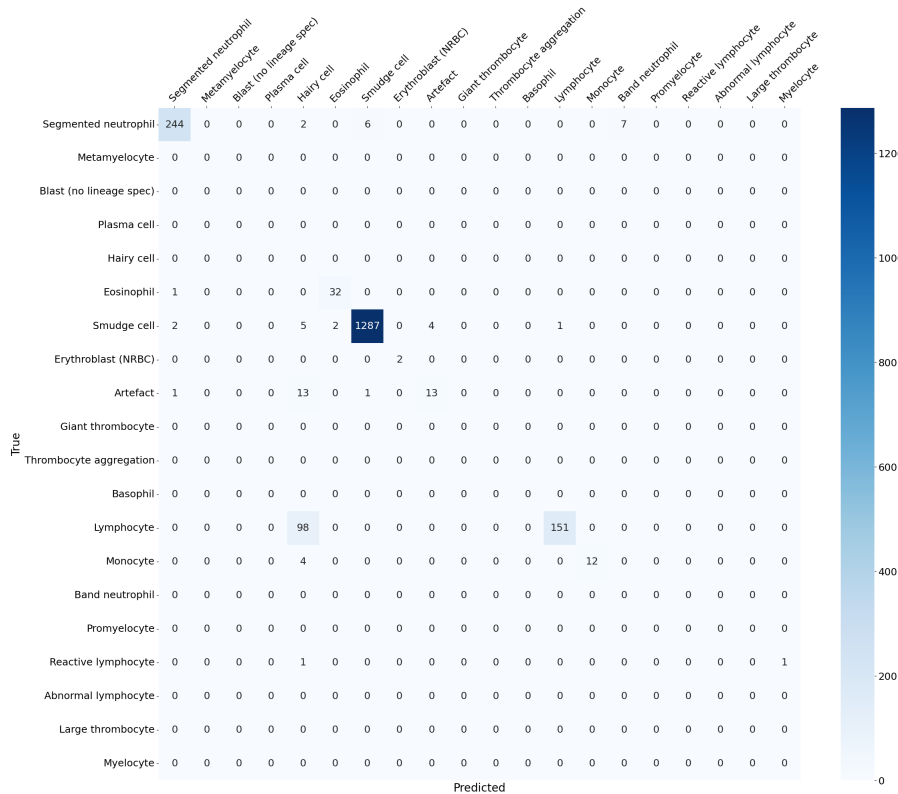


Figure 5.13: Confusion matrix for the model HAL-T3003 trained on sets A, B, and C, and tested on set C.

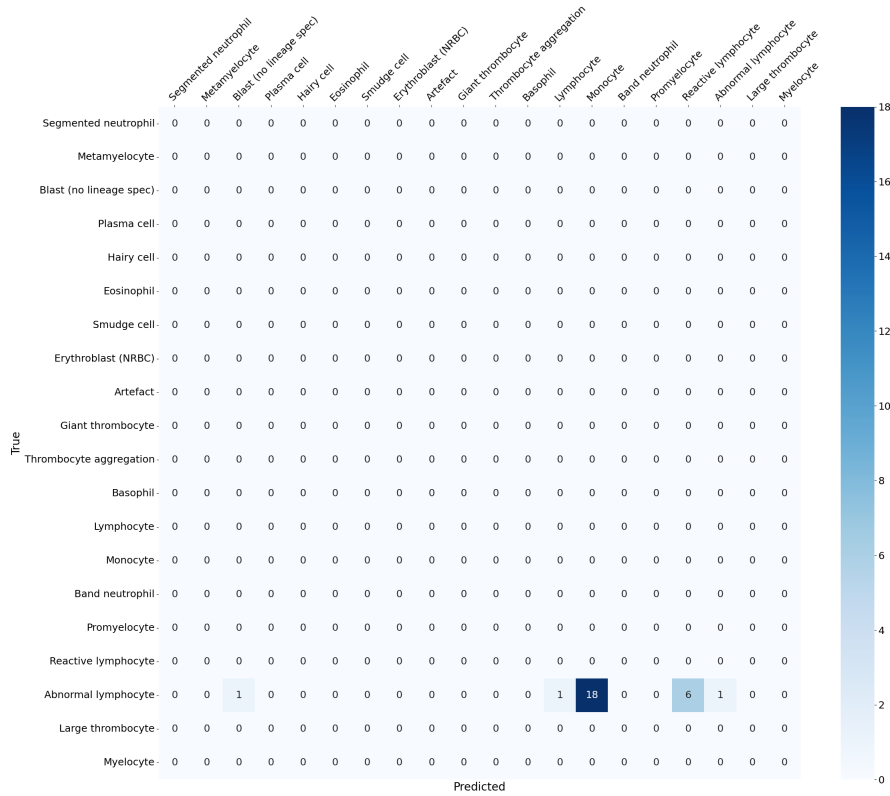


Figure 5.14: Confusion matrix for the model HAL-T3003 trained on sets A and B, and tested on set D.

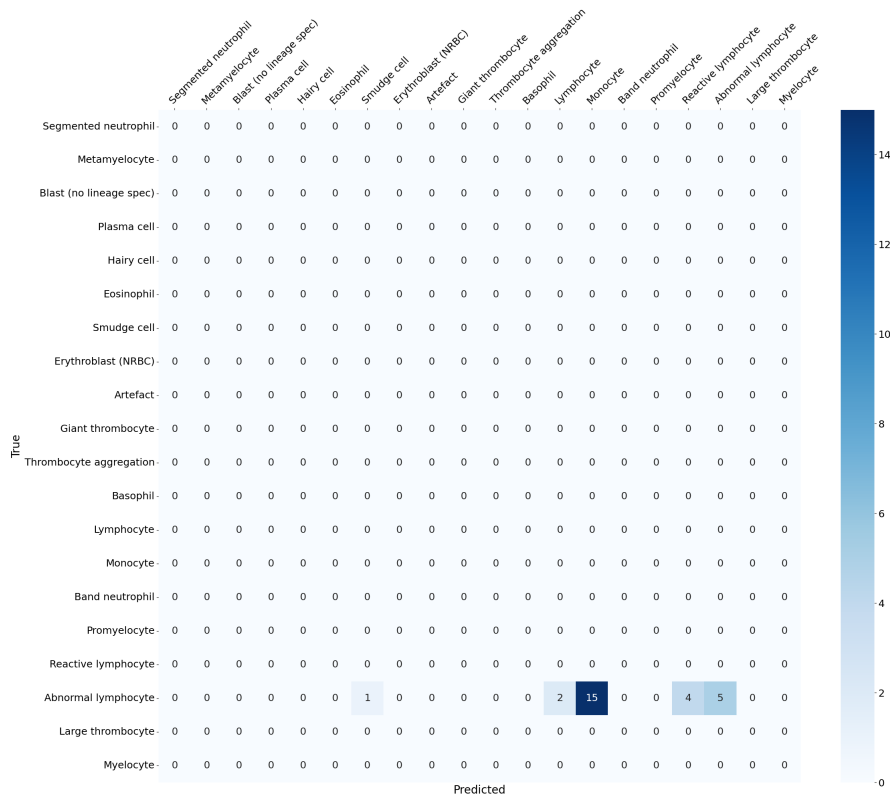


Figure 5.15: Confusion matrix for the model HAL-T3003 trained on sets A, B, and C, and tested on set D.

Believing is like being in an experiment on mind control [1].

6.1 Random initialization of Xception

We will start the discussions by examining the randomly initialized model, as this will better illustrate the benefits gained by using transfer learning. As expected, using random initialization for the network did not yield any reasonable results. All images were classified to the same class, namely smudge cell. This implies that the model has found a very poor local minimum and was unable to detect any useful features. Since the selected architecture has 20 847 932 trainable weights, the weight space is very vast. Therefore, the likelihood of starting close to any decent local minimum is extremely low. Additionally, if the optimization algorithm had been able to find a minimum, the risk of this minimum being overfitted to the training data, which is relatively small compared to the number of weights, would be very high. This means that such a minimum might not be a general one.

Another factor that makes it hard to train large networks from random weights is that there are many hyperparameters to tune. A hyperparameter search would take a long time, since for each hyperparameter setting a new training is needed and when all weights are trainable, the time for an epoch is quite long. We chose to base the settings for this training on the hyperparameters in the transfer learning section, but with a few changes. Firstly, we used a learning rate of 10^{-3} , since we believed that an untrained model would need to move around more than a transfer learning model. Secondly, we set the total number of epochs to 100, to give it a chance to converge, though it stopped early already after 11 epochs. A more in-depth hyperparameter search was not performed, since it falls outside of the scope of this project.

Looking at the encoded space it is clear that the model has not learned to differentiate between different classes. There is no clustering of similar classes of any sort.

6.2 Transfer Learning with Xception

With the best performing model, HAL-T3002, achieving a 88.21% test accuracy, the results of the transfer learning model far exceed the training of the randomly initialized model. The idea is that the transferred weights already have learned some general image features, and all the top layers have to do is to find a decision boundary well suited for the cell classification task. This local minimum will potentially be close to an even better minimum, which can be found after unfreezing the base model.

There is a drop of 2.6% in accuracy between the validation and test sets. This could be connected to the training and validation split of a common dataset. The split will not take into account that some images come from the same slide. This means that some cells in the training and validation set might share some similarities since they come from the same patient, are stained the same way and that the images are collected using the same system. The test set on the other hand will exclusively come from another set of slides. The cell images might therefore be different when it comes to these previously mentioned factors, and the training data can be said to not completely represent the underlying distribution.

Looking more closely on the confusion matrix in Figure 5.2 on page 38, it is worth noting that there are no false positive hairy cells, i.e., no images are falsely classified as such. The most common class among the false negative hairy cells, are lymphocytes. This is not completely unexpected, due to their strong likeness. If the "hairs" are very thin or small, both large and small hairy cells can look very similar to lymphocytes. The other lymphocyte classes suffered more from being misclassified to each other. There are especially an overclassification from abnormal lymphocytes to reactive lymphocytes. Generally the model seems to have had difficulties to learn to recognize abnormal lymphocytes, which is expected due to its large intra-class variation.

The scatter plot of the encoded space, Figure 5.3 on page 39, is far more interesting than the one created for the random initialization. Each class has now one or several more or less clearly defined clusters, where each class is marked with its own color. Note that the hairy cell images have formed two semi-close clusters at the left side of the figure. One hypothesis is that the two clusters represent the small and large hairy cells, since these have slightly different morphologies, e.g., when it comes to the size of the cells, but also the appearance of the cytoplasm.

Most of the reactive lymphocytes, the lymphocytes and some of the abnormal lymphocytes can be found in clusters close together in the upper part of the image. The abnormal lymphocytes have been split into a few different clusters that are somewhat spread out over the figure. This likely reflects the inhomogeneous morphologies of the class. The abnormal lymphocytes can come in many sizes, shapes and colors, making it a difficult class to cluster. This is why it from a machine learning perspective might be a really good idea to label the subclasses, such as hairy cells. This gives the network the possibility to separate the abnormal lymphocytes into these morphologically dissimilar subclasses instead of trying, and failing, to classify

them into one large, very diverse class.

6.3 Contrastive Learning

HAL-C3003 has the best result of the contrastive models and achieves an accuracy of 90.24%, though the others are very close in performance. Nonetheless, all the contrastive models distinctly outperform all the transfer models, meaning that HAL-C3003 is the best of all models in this project. HAL-C3003 has a 1.7% lower accuracy on the test set than on the validation set.

To compare the effect of using 16 or 128 projection units, confusion matrices and dimensionality reduction scatter plots for both HAL-C3001 and HAL-C3003 were included in the results. The confusion matrices, Figure 5.6 and Figure 5.7 on page 42, are very similar. HAL-C3003 has a better accuracy for hairy cells, reactive lymphocytes and abnormal lymphocytes, and has slightly fewer false negative hairy cells, while HAL-C3001 has one more true positive lymphocyte than HAL-C3003. However, the differences are small and could be due to the randomness in learning. Just like the best transfer learning model, none of the contrastive models has any false positive hairy cells. This indicates that the models generally are unwilling to classify images from "WBCTest" as hairy cells.

In Table 6.1 some false negative hairy cells can be seen. The first image contains two cells, which unsurprisingly confuses the classifier. Looking at the morphology of the left cell in the image, this is most likely a segmented neutrophil, which is the predicted class of the classifier. The rest of the images all have features which could make the classifier put them in the predicted class. The second image lack visible hairy strains, similarly with the fourth and sixth, the third looks quite smudged, and the fifth has a nucleus with many indentations which could lead the classifier to think that it is a segmented neutrophil.

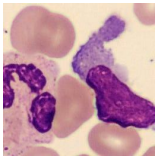
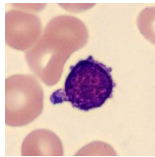
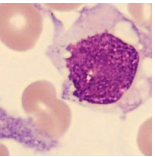
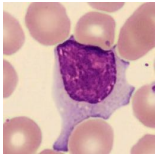
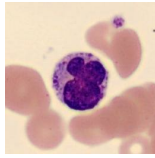
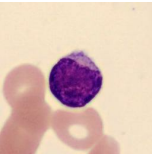
Image			
Predicted label	Segm. Neutr.	Lymphocyte	Smudge
Image			
Predicted label	Lymphocyte	Segm. Neutr.	Lymphocyte

Table 6.1: *Examples of images that should be classified as hairy cells but were predicted as other classes (false negatives). The images come from the test set and the predictions are done with HAL-C3003. The class labels denote the predictions.*

The scatter plots, Figure 5.10 and Figure 5.11 on page 44, are also quite similar,

though some clusters are moved and rotated. Some clusters seem to be a bit more tight for HAL-C3003, than for HAL-C3001. This can e.g., be seen for the lymphocyte and reactive lymphocyte clusters.

The use of more projection units gives the model more features to use for maximizing the similarities in the supervised contrastive loss, but this could increase the risk of overfitting to the training data. For the presented models, no such clear difference can be seen, as all models have a slight tendency of overfitting, i.e., that they have a lower accuracy on the test data.

The three step-training method for the contrastive models is superior compared to training both the encoder and the projection head at the same time in the first step. We believe that the advantage can be attributed to that, by first only training the projection head, the projection head is given the chance to find a good starting point at a reasonable local minimum. If the first step is left out, the network could not find any other contrastive loss minimum than the one where all images are projected to the same point in the encoded space and the projection head space. One can think of it as if the very small projection head is overshadowed by the enormous encoder. Since the weights in the projection head is only random, while the encoder is pre-trained on ImageNet, the pure force of the encoder will drive the training, leaving the weights of the projection head forgotten, and the result will be meaningless. The first training step will clearly not affect the encoded space, since these weights are frozen, but it can be seen as an advanced form of weight initialization of the projection head.

When training the projection networks, the second training step was very time-consuming. Each epoch took around 45 minutes, while for the contrastive classifier networks each epoch only took about 5-6 minutes. Due to this slow training we were limited in the number of experimental networks we could train. Therefore we only tried a temperature of 0.1, as suggested by previous articles, e.g., [11], however another temperature could be more suited for this task. Likewise we could not perform a huge search for the other hyperparameters for the projection network. For this reason the results on the validation set for the projection-head-16 was run with the hyperparameters in Table C.1, while for the final model it was believed that the model would gain from some hyperparameter alterations, such as more epochs in the training of the encoder, and larger learning rate when training only the projection head weights. These alterations were thus used in the final training, and could be seen in Table 4.8.

Since the difference between the model using 16 projection units and the models using 128 projection units is small, and the fact that projection-head-16 will have slightly less weights, meaning it is a smaller network to train, it could be preferable to use the smaller model. On the other hand HAL-C3003 did achieve the best accuracy of all models. The projection head weights used for training the projection network are discarded in the later training stages, which means that the deployment speed of the network is not affected by the difference in the number of weights. With available computing power, it is therefore worth to choose 128 projection units over 16 projection units.

6.4 Sanity Test

For part 1 of the sanity test, where the models are tested on C, the AB result is very poor compared to the ABC result, see Section 5.3. It confirms the hypothesis that the models initially might have learned to recognize the difference between the databases rather than the differences between the cells. The network HAL-T3003 went from an accuracy of 29.74% to 92.12%, which really shows the effect of using the other cell classes from "hairy_db". "hairy_db" is a very recently created database, whereas "WBCTrain" and "WBCTest" are a bit older, meaning that there could be some software or hardware discrepancies. Even with the naked eye it is possible to tell images from the two types of databases apart, based on other image characteristics than the morphologies of the cells. The three databases do not seem to completely represent the same underlying distribution.

The performance improvement can also be seen in the confusion matrices for part 1, Figure 5.12 and Figure 5.13. Between the first and second matrix there is a clear move of classifications to the diagonal, which represents the correctly classified cells. This is especially true for segmented neutrophils, smudge cells, artefacts, lymphocytes, and monocytes. For lymphocytes there are no true positives at all in the first figure, while there are 151 correct classifications in the second figure. When training with only AB, there was obviously an overclassification to hairy cells.

Some examples of false positive hairy cells can be seen in Table 6.2. The first two images are labeled as smudge cells, but they both contain more than one cell in the image. Again, this makes it terribly hard for the classifier to know where to put them. The other images are all lymphocytes, which could be because of the similarities they share with hairy cells.

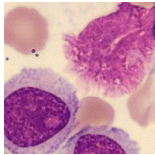
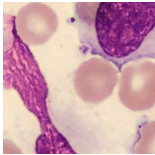
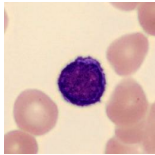
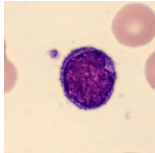
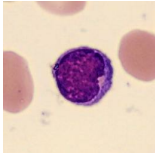
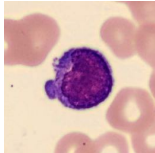
Image			
True label	Smudge	Smudge	Lymphocyte
Image			
True label	Lymphocyte	Lymphocyte	Lymphocyte

Table 6.2: *Examples of images that belong to other classes but were classified by the network as hairy cells (false positives). The images come from set C and the prediction is done with HAL-C3003. The class labels denote the ground truth.*

The result for part 2 is not quite as hypothesis confirming as the result for part 1. We believe that these cells could be hairy cells, but they were labeled as abnormal lymphocytes, and since we lacked a confirmation of the origin of the images, we chose not to use them in the training. This means that the accuracy gives an account for how many of these suspicious images were classified as abnormal lymphocytes.

There is a small increase of the already really low accuracy, but note that none of these images were actually classified as hairy cells. Neither when training on AB, nor when training on ABC. In short, our model does not know what to do with these images. This is likely because the model has not seen anything similar before, i.e., the images from "WBCTrain"/"WBCTest" that look like hairy cells are not represented in the training distribution. Thus, further image collection work is needed to better represent the underlying distribution. As mentioned in the theory, the i.i.d assumption needs to be fulfilled for the training and testing data. In the case of ABC and D the assumption seems not to hold, especially the identically distributed part, thus making it impossible to get an accurate result on D.

Some of these problems described in this section could, if not completely, at least partially be removed if the data was collected once again to obtain the .bmp images, instead of only the .jpg. Since the data loss was discovered only after the images had already been labeled, this was unfortunately not realistic to achieve in the limited time of this project.

6.5 Comparison of the Learning Methods

The best contrastive learning model, HAL-C3003, boosted the accuracy on the test set by 2.3%, compared to the best transfer learning model, HAL-T3002. This difference can partially be explained graphically by studying Figure 5.3 on page 39 and Figure 5.11 on page 44. The scatter plot for the contrastive learning model has an overall denser appearance, most noticeably for classes such as plasma cells and reactive lymphocytes. The contrastive model has also succeeded in separating the thrombocyte classes, which are more intertwined for the transfer learning model.

Apart from the loss, training set up and hyperparameters, the same base model and top architecture were used for both learning methods. Therefore, there is a reason to believe that this difference largely depends on the contribution of the contrastive loss.

It is interesting to note that the predictions for hairy cells did not improve. From the confusion matrix we see that the transfer learning network actually succeeded better with 1440 correctly classed hairy cells, compared to the best contrastive learning network with 1437. One of the reasons for the inter-class difference could be attributed to the data set. For the non-hairy cell classes, the images are gathered from a large number of patients and hospitals, which cover many different instruments and smear making techniques. The hairy cell images for training, on the other hand, are limited to nine slides from three hospitals. The contrastive loss is designed to learn the similarities between images depicting the same object but in different settings. When training on a somewhat homogeneous data set, the network is not exposed to samples of varying appearances and the strength of the contrastive loss is diminished. This is possibly why we could observe a performance gain in some of the non-hairy classes and a small loss in accuracy for the hairy cells.

Another observation is that many of the smaller cell classes like metamyelocytes,

plasma cells, and giant thrombocytes received a greater benefit from the contrastive loss. Recalling the underlying mechanism in contrastive learning, we suspect that this is due to the batch size. An interpretation could be that, for the smaller classes, a larger chunk of data is examined at a time by the network, and it is easier for the training to drive the weights toward a more suitable representational space.

The findings of other studies on contrastive learning also indicate that larger batch size yields better results. Although we were unable to experiment on different batch sizes due to hardware limitation, the results were still satisfactory. And we believe that with larger batch size, the supervised contrastive loss could improve classification performance in all cell classes.

The contrastive model is more robust than the transfer learning model, with a 1.7% validation-test accuracy drop compared to 2.6% for the transfer learning model. Recalling the formulations of the losses, one could argue that this difference is general rather than coincidental. The learning objective for the cross-entropy loss is "what does a cell class look like?" and trains the network to learn features for each of the classes. On the other hand, the contrastive loss instead tries to find out "how are the cell classes best separated from each other?", and creates the representational space such that each cluster is as concentrated within each other and as separated from the others as possible. With this interpretation and the awareness that cell images can differ in non-morphological attributes, it is plausible to suggest that a learning method which focuses on the differences between classes is better at knowledge generalization than a method that relies on a specific appearance of each cell class.

6.6 General Discussion

The hairy cells used in this thesis are labeled by only one expert, which might have introduced an expert bias, i.e., labeling of the cells can be quite subjective. In future studies it would therefore be interesting to use several experts to remove some of the subjectivity.

Another difficulty is that the network is only able to judge the cell images based on appearance, which might lead to misclassifications of the group of cells resembling hairy cells mentioned in section 2.3.1. This problem is seemingly quite hard to combat with deep neural networks, and is left to future studies.

There are a few images in the training and test sets that contain two or more cells per image. To improve the models further, these images should have been removed prior to training and testing. The images are labeled as one class, however the network prediction will be random, since there is no way for the network to know which cells is the main cell. A cleanup of the data would therefore be suggested by us. In addition we suggest recollecting the images as .bmp to reduce the software bias.

The final models, for which a test accuracy is presented, were trained on more data than the models, for which the validation accuracy is presented. Not only are these

final models trained on both training and validation data, but also all training data from "hairy_db". Clearly, the use of more data helped the generalization ability.

Due to the limit in time and computing resources some things were not explored in this project. Future work could look into more advanced classifiers and projection heads, or other hyperparameter combinations. Especially hyperparameters for the projection network could be investigated further. We experimented to some extent with hidden layers in the classifier. This did however only reduce the performance, and the idea was thus abandoned.

In this thesis we only focused on hairy cells, but for the medical field it would be of great interest to also include additional abnormal lymphocyte classes, such as mantel cells and Sézary cells. For this to be possible, a great effort must be made on obtaining cases of the corresponding leukemias and lymphomas. The key to a well generalizing network is the data. The several clusters within the abnormal lymphocyte class could be representing these subclasses and an interesting starting point for future investigations.



Conclusion

Keep robotizing [1].

Based on the results, we conclude that both models work well for pre-classifying hairy cells along with 19 other WBC classes. Furthermore, supervised contrastive learning outperforms traditional transfer learning in both accuracy and robustness for WBCs in general. The cell class-specific comparisons indicate that the method works well on classifying cells with large diversities and could possibly even distinguish other abnormal lymphocyte cell classes than hairy cells. We believe that these findings contribute to a step forward in digital hematology.

With a combination of using datasets with larger in-class variations, and training using larger batch sizes, the performance of the contrastive learning network can be improved even further. The hematology expert shortage that the healthcare sector is experiencing at the time means that much of the available cell image data is unlabeled. This working example of supervised contrastive loss applied on cell images paves the way for further work to explore whether contrastive loss could also be used on partially labeled data for classifying abnormal lymphocytes and other WBCs.

References

- [1] *Inspirobot*. [Online]. Available: <https://inspirobot.me/> (visited on 2022/06/02).
- [2] L. Palmer, C. Briggs, S. McFadden, *et al.*, "Icsh recommendations for the standardization of nomenclature and grading of peripheral blood cell morphological features", *International Journal of Laboratory Hematology*, vol. 37, no. 3, pp. 287–303, 2015. DOI: <https://doi.org/10.1111/ijlh.12327>.
- [3] H. M. Golomb and J. W. Vardiman, "Diagnosis of hairy cell leukemia", in *Holland-Frei Cancer Medicine, 6th edition*, D. W. Kufe, R. E. Pollock, R. R. Weichselbaum, *et al.*, Eds., 6th ed., Hamilton (ON): BC Decker, 2003. [Online]. Available: <https://www.ncbi.nlm.nih.gov/books/NBK13250/>.
- [4] S. Marionneaux, *Personal communication*, 2022.
- [5] S. Alférez, A. Merino, L. Mujica, M. Ruiz, L. Bigorra, and J. Rodellar, "Automatic classification of atypical lymphoid b cells using digital blood image processing", *International Journal of Laboratory Hematology*, vol. 36, no. 4, pp. 472–480, 2013. DOI: [10.1111/ijlh.12175](https://doi.org/10.1111/ijlh.12175).
- [6] L. Bigorra, A. Merino, S. Alférez, and J. Rodellar, "Feature analysis and automatic identification of leukemic lineage blast cells and reactive lymphoid cells from peripheral blood cell images", *Journal of Clinical Laboratory Analysis*, vol. 31, no. 2, 2016. DOI: [10.1002/jcla.22024](https://doi.org/10.1002/jcla.22024).
- [7] S. Shafique and S. Tehsin, "Acute lymphoblastic leukemia detection and classification of its subtypes using pretrained deep convolutional neural networks", *Technology in Cancer Research & Treatment*, vol. 17, 2018. DOI: [10.1177/1533033818802789](https://doi.org/10.1177/1533033818802789).
- [8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017, ISSN: 0001-0782. DOI: [10.1145/3065386](https://doi.org/10.1145/3065386).
- [9] L. Boldú, A. Merino, A. Acevedo, A. Molina, and J. Rodellar, "A deep learning model (alnet) for the diagnosis of acute leukaemia lineage using peripheral blood cell images", *Computer Methods and Programs in Biomedicine*, vol. 202,

- 2021, ISSN: 0001-0782. DOI: <https://doi.org/10.1016/j.cmpb.2021.105999>.
- [10] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, *A simple framework for contrastive learning of visual representations*, 2020. arXiv: 2002.05709. [Online]. Available: <https://arxiv.org/abs/2002.05709>.
- [11] P. Khosla, P. Teterwak, C. Wang, *et al.*, *Supervised contrastive learning*, 2020. arXiv: 2004.11362. [Online]. Available: <https://arxiv.org/abs/2004.11362>.
- [12] Y. Zhong, M. Huang, H. Fan, R. Hu, and Z. Li, "An improved unsupervised white blood cell classification via contrastive learning", in *Data Mining and Big Data*, Y. Tan, Y. Shi, A. Zomaya, H. Yan, and J. Cai, Eds., Singapore: Springer Singapore, 2021, pp. 100–109, ISBN: 978-981-16-7476-1.
- [13] M. Tran, S. J. Wagner, M. Boxberg, and T. Peng, *S5cl: Unifying fully-supervised, self-supervised, and semi-supervised learning through hierarchical contrastive learning*, 2022. arXiv: 2203.07307. [Online]. Available: <https://arxiv.org/abs/2203.07307>.
- [14] E. C. Lynch, "Peripheral blood smear", in *Clinical Methods: The History, Physical, and Laboratory Examinations. 3rd edition*. H. J. Walker HK Hall WD, Ed., Boston: Butterworths, 1990.
- [15] *Cellavision - why automate the manual differential*. [Online]. Available: <https://www.cellavision.com/en/why-automate-the-manual-differential> (visited on 2022/04/28).
- [16] *Cellavision - press resources*. [Online]. Available: <https://www.cellavision.com/en/press-resources> (visited on 2022/05/17).
- [17] T. Hastie, R. Tibshirani, and J. Friedman., "The elements of statistical learning - data mining, inference, and prediction", in 2nd ed. New York, NY: Springer, 2009, p. 394, ISBN: 978-0-387-84857-0.
- [18] M. Ohlsson and P. Edén., *Introduction to Artificial Neural Networks and Deep Learning*. Lund: Computational Biology, Biological Physics, Department of Astronomy, and Theoretical Physics, Lund University, 2021.
- [19] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain", *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958. DOI: <https://doi.org/10.1037/h0042519>.
- [20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [21] D. L. Olson and D. Delen, "Advanced data mining techniques", in 1st ed. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2008, ch. 9, p. 138, ISBN: 978-3-540-76916-3.
- [22] C. Szegedy, W. Liu, Y. Jia, *et al.*, *Going deeper with convolutions*, 2014. arXiv: 1409.4842. [Online]. Available: <https://arxiv.org/abs/1409.4842>.

- [23] S. Löwel and W. Singer, "Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity", *Science*, vol. 255, 1992, pp. 209-212.
- [24] Wikipedia contributors, *Arnold's cat map* — *Wikipedia, the free encyclopedia*, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Arnold%27s_cat_map&oldid=1050670712 (visited on 2022/05/19).
- [25] F. Chollet, *Xception: Deep learning with depthwise separable convolutions*, 2016. arXiv: 1610.02357. [Online]. Available: <https://arxiv.org/abs/1610.02357>.
- [26] E. Tiu, *Understanding contrastive learning - learn how to learn without labels using self-supervised learning*, 2021. [Online]. Available: <https://towardsdatascience.com/understanding-contrastive-learning-d5b19fd96607> (visited on 2022/05/19).
- [27] S. Chopra, R. Hadsell, and Y. LeCun, *Learning a similarity metric discriminatively, with application to face verification*. [Online]. Available: <http://yann.lecun.com/exdb/publis/pdf/chopra-05.pdf>.
- [28] TensorFlow, *Tf.data.dataset | tensorflow core v2.9.0*. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/data/Dataset (visited on 2022/05/24).
- [29] F. Chollet, *Transfer learning & fine-tuning*, 2020. [Online]. Available: https://keras.io/guides/transfer_learning/ (visited on 2022/04/29).
- [30] *About imagenet*. [Online]. Available: <https://www.image-net.org/about.php> (visited on 2022/04/29).
- [31] G.-O. Carlsson, *Personal communication*, 2022.
- [32] TensorFlow, *Tf.keras.layers.layer | tensorflow core v2.9.0*. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Layer (visited on 2022/05/24).
- [33] Z. Wang, *Contrastive loss for supervised classification*, 2020. [Online]. Available: <https://towardsdatascience.com/contrastive-loss-for-supervised-classification-224ae35692e7> (visited on 2022/05/02).
- [34] A. Béres, *Semi-supervised image classification using contrastive pretraining with simclr*, 2021. [Online]. Available: https://keras.io/examples/vision/semisupervised_simclr/#selfsupervised-model-for-contrastive-pretraining (visited on 2022/05/02).

A

Miscellaneous

All code for the thesis was written in Python 3.8/3.9. The models were created and trained using Keras and tensorflow-gpu 2.7 on an NVIDIA RTX 2080 8GB and an NVIDIA RTX 2060 Super 8GB. The contrastive network with all data required too much memory for our computers and was instead trained on an NVIDIA RTX 1080 Ti 11GB. The Xception architecture and weights were obtained from the Keras library.

All scatter plots and confusion matrices were produced using the Seaborn package. All illustrations, except the ones that have origin references, are produced by the authors in Adobe Illustrator. The cell images were extracted from the CellaVision databases.



Architectures

B.1 Architecture for HAL-T and HAL-R

Layer (type)	Output Shape	Parameters
Input	(None, 256, 256, 3)	0
Augmentation	(None, 256, 256, 3)	0
Rescaling	(None, 256, 256, 3)	0
Xception	(None, 8, 8, 2048)	20861480
GlobalAveragePooling2D	(None, 2048)	0
Dropout	(None, 2048)	0
Dense	(None, 20)	40980
Total parameters		20902460

B.2 Architecture for the Encoder of HAL-C

Layer (type)	Output Shape	Parameters
Input	(None, 256, 256, 3)	0
Rescaling	(None, 256, 256, 3)	0
Xception base model	(None, 8, 8, 2048)	20861480
GlobalAveragePooling2d	(None, 2048)	0
Total parameters		20861480

B.3 Architecture for the Projection Network of HAL-C with 16 Projection Units


Layer (type)	Output Shape	Parameters
Input layer	(None, 256, 256, 3)	0
Augmentation	(None, 256, 256, 3)	0
Encoder	(None, 2048)	20861480
Normalization	(None, 2048)	0
Dropout	(None, 2048)	0
Dense	(None, 16)	32784
Normalization	(None, 16)	0
Total parameters		20894264

B.4 Architecture for the Projection Network of HAL-C with 128 Projection Units

Layer (type)	Output Shape	Parameters
Input layer	(None, 256, 256, 3)	0
Augmentation	(None, 256, 256, 3)	0
Encoder	(None, 2048)	20861480
Normalization	(None, 2048)	0
Dropout	(None, 2048)	0
Dense	(None, 128)	262272
Normalization	(None, 128)	0
Total parameters		21123752

B.5 Architecture for the Classifier Network of HAL-C

Layer (type)	Output Shape	Parameters
Input	(None, 256, 256, 3)	0
Augmentation	(None, 256, 256, 3)	0
Encoder	(None, 2048)	20861480
Normalization	(None, 2048)	0
Dropout	(None, 2048)	0
Dense	(None, 20)	40980
Total parameters		20902460



Hyperparameters

In Table C.1 the hyperparameters used in the training of projection-head-16 in the model selection with validation split can be seen.

Table C.1: *Selected hyperparameters for the projection network in contrastive learning in the training with validation split for projection-head-16.*

Hyperparameter	projection-head-16
Projection head epochs	7
Encoder and projection head epochs	7
Projection head learning rate	10^{-5}
Encoder and projection head learning rate	10^{-5}
Dropout	0.2
Batch size	16
Temperature	0.1
Projection head units	16
Encoder base	encoder-16



Popular Science Summary

Truffles and Champignons: How to Find Cancer Cells with Machine Learning?

The white blood cells in our blood stream tell stories about how our bodies are doing. Hairy cells are cancerous cells indicating leukemia. How do we find such cells using contrastive machine learning?

You've heard about self-driving cars that can "see" the traffic. Did you know that the same kind of technology can be used to detect blood cancer cells, such as hairy cells, and help doctors diagnose and treat patients? This tool is called machine learning. To be specific, we compared a method called transfer learning and another called contrastive learning and got 88 % and 90 % correctly predicted cells, respectively. Now, machine learning might sound scary, but remember, we want to train machines to listen to us and to make our lives happier. So in a way, machines are like puppies (but much less fluffy and bark in 1's and 0's). For the time being, let's replace "machine learning" with "puppy learning".

Harry is a very cute puppy and you're eager to show off to your friends that Harry can find a truffle (the cells we want to find) among a bunch of champignons (other cells). How do you teach him? Puppies aren't inherently crazy about mushrooms, but they do love treats. You let Harry smell a truffle and a champignon. When Harry puts his little snoot on a truffle, you yell "Good boy!" and reward him with a yummy treat! The same thing happens inside a machine learning model. Every time it's correct, the behavior is rewarded, and if it makes the wrong decision, we'll guide it towards making a better decision next time. You continue with the next pair of fungi. As the training session progresses, Harry is more and more eager to boop the truffle.

Next day, you wonder, would it be easier for Harry if he got to see many fungi at once? You present Harry with 5 champignons and 5 truffle at once. As the goodest boy in the world, Harry doesn't disappoint you and quickly separates all the truffles from the set. This is the essence of contrastive learning. The similar smells of the

truffles make them more recognizable as a group.

Harry gets better at the game each day, but during one session, he picked the wrong mushrooms. No treat was served and Harry looks at you with pleading puppy eyes. Is there anything wrong with the mushrooms? Suddenly, it struck you that when packing the mushrooms in a hurry yesterday, you put both kinds in the same plastic bag. Their scents have gotten mixed up! Dogs don't see as good as humans do and Harry has been relying on his nose to tell them apart. Of course he can't find the right ones now!

What happened here is similar to using wrongly labeled data for training. Harry's nose can't tell a truffle from a champignon if they smell the same. Likewise, a machine learning network also cannot make the right choices if it has been fed poor quality data.

After discovering the mistake you store the mushrooms apart and, to your relief, Harry excels at the fungi game the next day.

Now your training can continue. Hopefully you and Harry can go and win truffle searching competitions together! Oh, and for us? We'll continue finding new ways to make data and algorithms work for us and help doctors discover illnesses.