Neural Network Cascades to Incorporate Domain Knowledge for Hematopoietic Cell Classification

Možnosti využití kaskád neuronových sítí
pro klasifikaci krvetvorných buněk

Master Thesis

Diplomová práce

presented by

Jonas Nienhaus

Supervisor:
Ing. Jan Havlík, Ph.D.

Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

in cooperation with

Institute of Imaging & Computer Vision
Supervisor: Philipp Gräbel, M.Sc.
Prof. Dr.-Ing. Dorit Merhof
RWTH Aachen University

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Nienhaus Jonas Daniel** Personal ID number: **487317**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Medical Electronics and Bioinformatics**

Specialisation: **Bioinformatics**

II. Master's thesis details

Master's thesis title in English:

Neural Network Cascades to Incorporate Domain Knowledge for Hematopoietic Cell Classification

Master's thesis title in Czech:

Možnosti využití kaskád neuronových sítí pro klasifikaci krvetvorných buněk

Guidelines:

Description:
A fully automated classification workflow for the evaluation of bone marrow samples could be very helpful for medical professionals by potentially speeding up diagnosis and assessment of several diseases of the blood building system, such as leukemias. This work is part of a greater hematology project at RWTH Aachen university, with the aim to develop such a workflow, involving detection, segmentation and classification of cells.
The focus of this thesis is the classification sub-task. In the past, it was shown that Convolutional Neural Networks are the most promising approach compared to other image classification algorithms [1]. However, while state-of-the-art neural network architectures show promising results, they do not yet incorporate any domain knowledge,
such as the biological cell lineages or certain morphological properties, shared by some cell types. The aim of this work is to develop a cascade of neural networks to make use of this knowledge, and compare the results to standard, flat classification architectures.
Tasks and challenges:
The task includes literature research and the development of a stage-wise classification architecture together with an appropriate learning strategy to represent the cell hierarchy, or other properties that are shared by certain cell classes.
A challenging aspect is the relatively small dataset of annotated microscopic cell images with strong class imbalances.
In the end, meaningful evaluation criteria for the final classification results should be employed to compare them to popular, flat classification architectures.

Bibliography / sources:

[1] Gräbel et al.: Evaluating Out-of-the-box Methods for the Classification of Hematopoietic Cells in Images of Stained Bone Marrow, 2018
[2] Munker et al.: Modern Hematology (second edition), Humana Press Inc., 2007
[3] Löffler et al.: Atlas of Clinical Hematology (sixth edition), Springer, 2005
[4] Goodfellow et al.: Deep Learning, MIT Press, 2016
[5] Sandro Skanski: Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence, Springer, 2018
[6] Silla, C.N., Freitas, A.A. A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery 22, 31–72, 2011

Name and workplace of master's thesis supervisor:

Ing. Jan Havlík, Ph.D., Department of Circuit Theory, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **07.09.2020** Deadline for master's thesis submission: **10.11.2020**

Assignment valid until: **19.02.2022**

Ing. Jan Havlík, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Declaration of Authenticity

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodological instructions for observing the ethical principles in the preparation of university theses.

Aachen, 31.10.2020

Abstract

In this work, classification of haematopoietic cells is performed hierarchically, employing cascades of deep neural networks. Two strategies are defined to obtain combined predictions from the cascades, a probabilistic approach and a greedy, deterministic method. Although different in theory, both strategies lead to very similar outcomes in practice. Hierarchical cascades can be trained either separately for each network or end-to-end as a whole. It is shown that the individual training is preferable both in terms of achieved performance as well as flexibility. Different hierarchies are defined to guide the classification process, either oriented on the biological cell lineages, on certain features and characteristics, or in order to compensate for class imbalance in the dataset. Several methods and optimisation techniques are evaluated, including the possibility to incorporate regressors instead of classifiers for certain sub-tasks where the classes have an ordinal character. In general, it is shown that, for appropriate hierarchies, similar classification performances as with single networks can be achieved. Especially pretraining of the individual networks on the target dataset yields improvements. The main advantage of the cascades is an increased modularity as well as additional information compared to single networks. With feature forwarding, a method introducing embedding vectors into the cascades and passing them to subsequent networks, the learnt feature space can be visualised not only for the entire model, but also at intermediate levels.

Contents

List of Figures

List of Tables

Nomenclature

This nomenclature defines important notation used throughout this thesis. Some variables which are not used repeatedly, may be not listed, but are declared appropriately within the corresponding section.

General Mathematical Notation

$\mathbb{N}_0$	natural numbers including zero
$\mathbb{R}$	real numbers
a	scalar value
$\boldsymbol{a}$	vector
n_a	length of vector $\boldsymbol{a}$
∇	gradient
$\boldsymbol{A}$	matrix (or tensor)
$\boldsymbol{A}^T$	transposition of matrix $\boldsymbol{A}$
λ	eigenvalue of a matrix
$\boldsymbol{\mu}$	element-wise mean
$\boldsymbol{\sigma}$	element-wise standard deviation
std	unbiased sample standard deviation

Neural Networks and Machine Learning

$\boldsymbol{W}$	weight matrix, including the bias
n	number of outputs, number of classes
N	total number of samples in the set (e.g. training set)
$\boldsymbol{x}$	arbitrary input
$\boldsymbol{t}$	target vector
$\boldsymbol{y}$	output vector
$\mathcal{L}$	loss
$\mathcal{L}_{\text{CE}}$	cross-entropy loss
$\mathcal{L}_{\text{MSE}}$	mean squared error loss
$\mathcal{L}_{\text{tree}}$	hierarchical tree loss
$\mathcal{L}_{\text{WCE},\gamma}$	weighted cross-entropy loss with weights γ
α	learning rate
S	softmax output
$\boldsymbol{K}$	kernel
$\boldsymbol{I}$	image
$\boldsymbol{Y}$	output tensor
$\boldsymbol{F}$	tensor of feature maps

w, h	width, height of an image
γ_i	loss weight factor
δ	exponent in loss weight
ν	upsampling (or downsampling) exponent
$\boldsymbol{S}$	similarity matrix
$\boldsymbol{L}$	Laplacian matrix

Hierarchical Classification and Cascades

C	set of all classes/nodes in a hierarchy
c_i	class and node label of the i-th node
c'	predicted class and node label
c_0	root class/node
C_i	true multi-class/multi-label of node i
C'_i	predicted multi-label
$\hat{C}_i$	true multi-label excluding the root: $\hat{C}_i = C_i \backslash \{c_0\}$
$\hat{C}'_i$	predicted multi-label excluding the root
$P(x)$	probability (confidence) of x
n_{emb}	embedding length of a single network
n_{cat}	length of concatenated embedding vectors
T	tree
leaves(T)	set of all leaves in a tree T
parent(i)	direct ancestor of the i-th node

Abbreviations

Adam	Adaptive Moment Estimation
ALL	Acute Lymphocytic Leukaemia
AML	Acute Myelogenous Leukaemia
baso.	Basophilic
CE	Cross-Entropy (Loss)
CLL	Chronic Lymphocytic Leukaemia
CML	Chronic Myelogenous Leukaemia
CNN	Convolutional Neural Network
DAG	Directed Acyclic Graph
DN	DenseNet
DNA	Deoxyribonucleic Acid
eosino.	Eosinophilic Granulocyte
ery.	Erythroblast
FF	Feature Forwarding
FLOPs	Floating Point Operations
FN	False Negative
FP	False Positive
GE-index	Ratio of Granulopoietic Cells to Erythropoietic Cells
GPU	Graphics Processing Unit
gran.	Granulocyte
ILSVRC	ImageNet Large Scale Visual Recognition Competition
immat.	Immature
megakaryop.	Megakaryopoiesis
MSE	Mean Squared Error
N/C	Nucleus-to-Cytoplasm Ratio
NNR	Non-Neighbour-Rate
orthochr.	Orthochromatic
polychr.	Polychromatic
pre.	Pretrained
px	Pixels
ref.	Reference
ReLU	Rectified Linear Unit
RGB	Red Green Blue (Additive Colour Model)
RN	ResNet
RNA	Ribonucleic Acid
segm.	Segmented
TN	True Negative
TP	True Positive
UMAP	Uniform Manifold Approximation and Projection
WCE	Weighted Cross-Entropy Loss

1 Introduction

In the human circulatory system, a variety of blood cells is found, each performing specific tasks. Perturbations of their development, for example from leukaemia, can have severe consequences, including death [1].

The different types of blood cells originate in the bone marrow, ranging from oxygen carrying red blood cells over white blood cells of the immune system up to platelets. In clinical haematology, bone marrow samples, acquired from the pelvic bone, are therefore of key importance for diagnosis and assessment of different diseases. Most prominently, the various types of leukaemia, for example chronic myelogenous leukaemia (CML), lead to different abnormalities. CML is characterised by atypical cell distributions, while other diseases can for example cause the occurrence of certain irregular cell forms. Therefore, bone marrow samples are analysed both for abnormal cell distributions as well as for pathological cell forms. In today's clinical practice, this analysis involves manual examination of multiple stained, microscopic bone marrow smear images by specially trained medical experts. This process is not only time consuming, but also prone to errors.

A joint project of the *Institute of Imaging and Computer Vision* of the RWTH Aachen University and the *Department of Haematology, Oncology, Haemostaseology and Stem Cell Transplantation* of the RWTH Aachen University Hospital aims to support medical doctors in this analysis and obtain more objective and reproducible statistics. The goal of the project is the development and implementation of a mostly automated analysis pipeline. After acquisition and digitalisation of whole slide bone marrow smear images, important steps involve the detection of cells, followed by classification of the cell types to determine the distributions of the individual cell types. Major difficulties include very heterogeneous samples – depending on the individual stain of different bone marrow samples – a very limited number of annotated samples with significant class imbalance, and a large number of different classes, including multiple types of artefacts.

The thesis at hand focuses on the classification sub-task of this process. Previous investigations have shown that deep neural networks are particularly powerful tools for image classification tasks [2, 3]. In the haematological domain, the different cell types originate from common ancestors, which differentiate into several distinguishable lineages. Therefore, some cell types are more similar than others. Single neural networks, classifying all different classes at once, are not designed to explicitly account for this hierarchical domain knowledge or morphological similarities between different cell types. In this work, the classification process is therefore split hierarchically into several sub-tasks, using deep neural networks at intermediate levels

to assign samples to different branches of the taxonomy.

Such hierarchical cascades could not only potentially improve the classification performance, but might also provide deeper and more accessible insights into the classification process. Additionally, they could make the whole classification process more flexible – up to the inclusion of different network types such as regressors to quantify, for example, a cell's maturity. Major tasks and problems covered in this thesis are the development of such hierarchical cascades, including different approaches, not only following the developmental hierarchy, but also splitting the classification by different characteristics of the cell types.

This work is structured as follows. Chapter 2 provides a detailed introduction to basic concepts required for this thesis. First, important medical terminology is established, followed by detailed descriptions of the different lineages of haematopoiesis, including important morphological characteristics. Afterwards, fundamentals from the field of machine learning are introduced, focused particularly on convolutional and deep neural networks, the underlying principles, and their training.

State of the art methods from fields such as deep learning and hierarchical classification are mentioned in Chapter 3. This includes descriptions of successful deep learning architectures and related methods to improve their performance or reduce training time. Different approaches which have been used in hierarchical classification tasks are introduced and formally defined. Additionally, the implemented methods used in the aforementioned haematology project, which form the basis for this work, are described. This includes not only details about the methods used in the other steps in the analysis pipeline, but also previous findings which are relevant for the classification task.

In Chapter 4, the methods developed in this work are introduced and described in detail. These methods involve hierarchical cascading of classification task, focused on the application in the haematological domain. Different methods to compose such cascades from deep networks – both in form of classifiers and regressors – are described. Two alternative algorithms to obtain final predictions are specified. Cascade training procedures are defined, either executed separately for each individual network, or for the entire cascade as a whole. Subsequently, various different hierarchies, incorporating different aspects of the haematological domain knowledge, for guiding the classification process are introduced and justified. Furthermore, several optimisation techniques are described, either to create and utilise relationships between different networks in the hierarchical classification process, to enable visualisation of the learnt feature spaces, or to partly compensate for class imbalance.

Chapter 5 describes several experiments to evaluate these developed methods. These experiments are described individually, followed by presentation and analysis of the respective results.

The implications of the experimental findings and interesting aspects for the future are summarised and discussed in Chapter 6, followed by a brief summary and conclusion.

2 Background

The purpose of this chapter is to introduce some basic concepts about the medical domain as well as machine learning, and in particular deep learning. Section 2.1 provides some background knowledge about haematopoiesis, the relevant cell types and their maturation. In Section 2.2, important foundations of machine learning and deep neural networks are described. Concrete state-of-the-art methods and implementations of these principles are introduced in Chapter 3.

2.1 Medical Background

Haematopoiesis is the formation of blood cells [4]. Except in the early embryotic and foetal stages, it arises in the bone marrow. In adults, haematopoiesis normally takes place in the marrow of the pelvis, the vertebral column, parts of the femur, the skull, ribs, and the sternum. Consequently, blood cells of different development stages, starting from haematopoietic stem cells, are present in the bone marrow, which additionally contains areas of fat. After development, the cells are released into the marrow sinuses, the marrow circulation and finally the systemic circulation. Consequently, in the peripheral blood of healthy subjects predominantly mature blood cells are found. Figure 2.1 provides an overview of the different cell lineages of haematopoiesis. The analysis of bone marrow samples is of major importance in diagnostic and assessment of diseases, most prominently different types of leukaemia. [4–6]

In Section 2.1.1, some general terminology and properties of cells are introduced. The subsequent sections provide basic knowledge about the cells present in the bone marrow as well as their role in haematopoiesis.

2.1.1 Basic Terminology

Biological cells usually consist of one or even multiple nuclei and the cytoplasm. The former contain the DNA and are necessary for cell divisions as well as for the regulation of biochemical processes in the cell. The cytoplasm is defined as the space between the cell membrane and the nucleus, and is the location of many biochemical reactions. An important characteristic of cells that contain nuclei is the nucleus-to-cytoplasm ratio (N/C), which is the fraction of the nucleus size relative to the entire cell volume. It depends on the cell type as well as the cell maturation and differentiation. A nucleus-to-cytoplasm ratio of 100% would mean that the nucleus fills the entire cell. Usually, a high N/C, indicating a relatively large nucleus compared to the overall cell size, can be found in young, but also in degenerated cells.

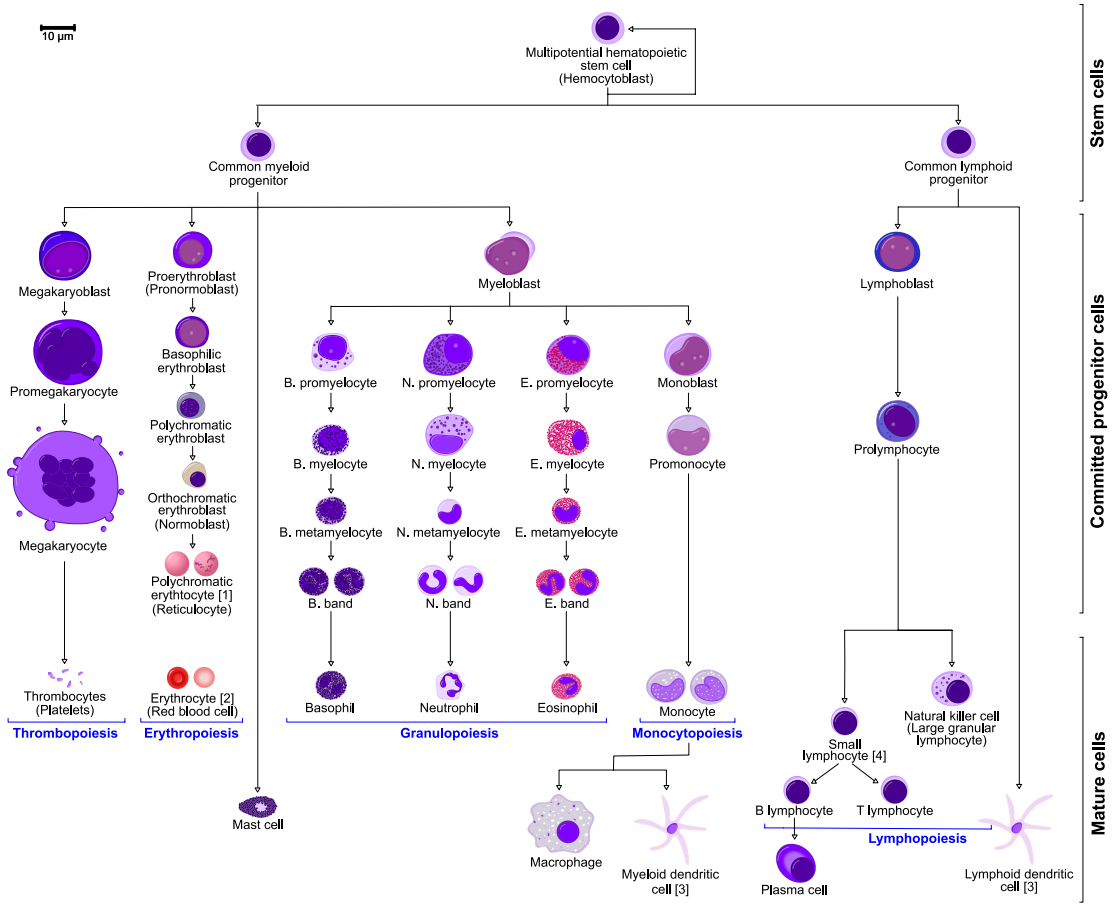


Figure 2.1: Developmental hierarchy of the cells of Haematopoiesis, as found in the bone marrow[1].

A low N/C on the other hand often indicates a mature, differentiated cell, which requires a large cytoplasm to allow for its specialised function. The appearances of both the nucleus and the cytoplasm are important distinctions between different cell types. Across different cell lineages there are some common naming schemes. For example, *blasts* are cells in very early stages, while the suffix *-cyte* is used for more mature cells. [7]

Pappenheim Stain

Haematologic samples require staining to make the cell structures visible. The stain predominantly used in haematological practice [5] is Pappenheim-stain. It is a panoptic stain, meaning that all cellular components are stained with a good contrast. As a combination of the Giemsa and the May-Grünwald stain, the Pappenheim stain is composed of several components which are specifically attracted by certain cellular components. Many cellular compounds in the cell may be either basophilic, for ex-

[1]Source: https://commons.wikimedia.org/wiki/File:Hematopoiesis_(human)_diagram_en.svg, by A. Rad and Mikael Häggström, CC-BY-SA 3.0 licence (https://creativecommons.org/licenses/by-sa/3.0/deed.en), slightly cropped.

ample the DNA, or basic, as for example some amino groups in proteins. Basophilic components attract the blue basic stain components Azure B and methylene blue, while basic (eosinophilic) components attract Eosin, which leads to a red colouring. Additionally, different stain components can form dimers, leading to alterations of light diffraction. These mechanisms lead to a range of colours and intensities that occur in haematologic samples and are specific to certain cell types, which are introduced in the following sections. [7]

Diagnostic Criteria Regarding the Nucleus

The main diagnostic criteria related to the nuclei concern the size, shape, location, number of nuclei, colour, nucleoli and the chromatin. Chromatin summarises specifically coloured components of the nucleus, mainly the DNA and basic chromosomal proteins. The chromatin structure and colour depend on the chromatin density within the nucleus, which itself depends on cell type and development stage. Dense chromatin, as it often occurs especially in highly differentiated cells, appears dark, while nuclei with a fine chromatin structure appear brighter. The shape of the nucleus is usually round for immature cells, while for some highly differentiated cells it can be indented. Nucleoli are small, round structures within nuclei and appear blue in Pappenheim stained slices, but can be obscured by chromatin. Their number, shape and size depend on the cell type and activity. [7, 8]

Diagnostic Criteria Regarding the Cytoplasm

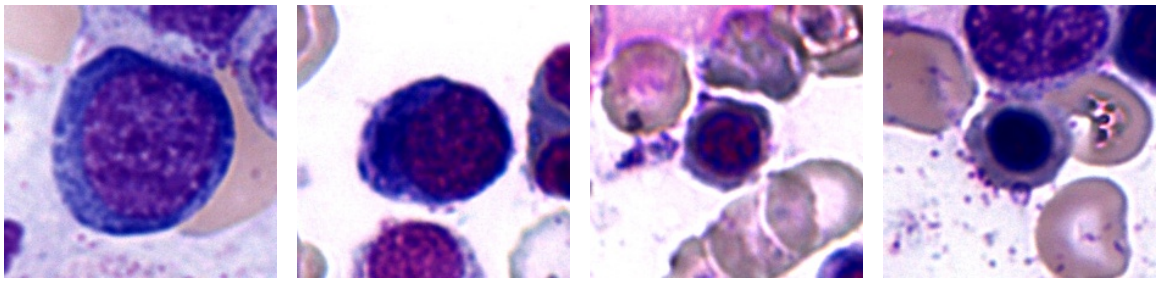
The cytoplasm contains several structures, called organelles, which can be made visible using special staining methods. Often, there are granules visible, which appear as small dots inside the cytoplasm and can have a different stain compared to the cytoplasm itself. The rough endoplasmatic reticulum contains ribosomes and is involved in protein biosynthesis. Because it is basophilic, it has blue colour in Pappenheim stain. The Golgi-apparatus, responsible for targeting of enzymes, is located close to the nucleus. In some, but not all cells it is visible as a lighter area close to the nucleus. [7, 8]

2.1.2 Erythropoiesis

The development of erythrocytes, or red blood cells, is called erythropoiesis [4, 6].

Erythrocytes originate from multipotent stem cells and are specialised to carry oxygen. For this purpose, mature erythrocytes contain haemoglobin molecules, which are responsible for the red colour. Mature red blood cells lack a nucleus, and therefore the capability to synthesise proteins. Normal erythrocytes have diameters around 8 µm and form biconcave, flexible discs. [4]

In contrast to mature erythrocytes, their precursors do contain nuclei, which decrease in size and eventually disappear during the four stages of erythropoiesis [5]. In all stages of erythropoiesis, potential granules in the cytoplasm are never red, and the nucleus is always round, if present [7].



(a) Proerythroblast **(b)** Basophilic erythroblast **(c)** Polychromatic erythroblast **(d)** Orthochromatic erythroblast

Figure 2.2: The four distinguishable cells of erythropoiesis in our dataset, showing progressive condensation of the nucleus while the cytoplasm loses its basophilic granulation.

The earliest precursor within erythropoiesis are proerythroblasts (see Figure 2.2(a) for an example), with sizes ranging from 14 to 18 µm [7]. Proerythroblasts at first do not yet contain haemoglobin, which however begins to appear around the nucleus already in this stage. They do not yet have a disc shape which is characteristic for mature erythrocytes. The nucleus appears pale blue with a dense, honeycombed chromatin structure, while the cytoplasm often shows a darkly basophilic, shadowy colour, with a perinuclear clear zone due to the Golgi-apparatus. The nucleus-to-cytoplasm ratio of 70 to 80 % [7] is still relatively large. [5]

The next stage are basophilic erythroblasts (Figure 2.2(b)). With diameters between 8 and 15 µm they tend to be smaller than proerythroblasts and have a lower N/C, which ranges from 50 to 70 % [7]. The nucleus appears rougher and more smudgy, with partially clumped chromatin, while the cytoplasm is blue. Although haemoglobin is seen in larger volumes, basophilic erythroblasts still contain basophilic material, hence the name. [5, 7]

After further maturation, the cells reach the stage of polychromatic erythroblasts (Figure 2.2(c)). As indicated by their name, their cytoplasm has mixed colours, ranging from brown to purple due to the presence of both basophilic RNA and red haemoglobin. With N/C between 30 and 50 %, the nucleus size is further decreased. [7]

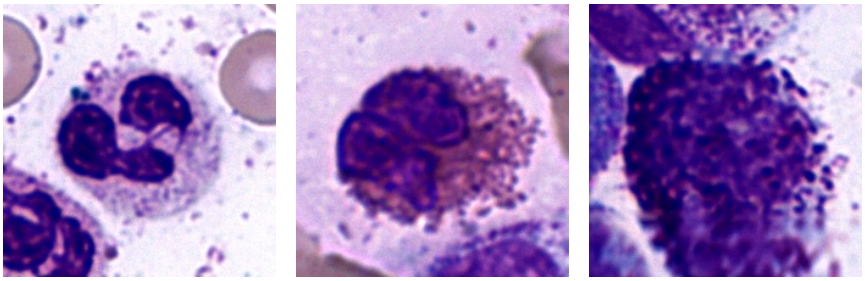
At the final stage before maturation, the stage of orthochromatic erythroblasts (Figure 2.2(d)), the cell size has decreased to diameters between 7 and 10 µm. The haemoglobin becomes more and more dominant, causing the colour of the cytoplasm to be predominantly light red, similar to the mature erythrocytes. Simultaneously, the nucleus-to-cytoplasm ratio is further decreased to values between 20 and 30 %. The nucleus appears small, round and very dense without nucleoli, and has a brownish to blackish colour. At this stage, the erythroblasts have lost their ability for cell division. [5, 7]

After passing all stages of erythropoiesis after four cell divisions and approximately four days, in total 16 erythrocytes originate from a single proerythroblast [7].

2.1.3 Myelopoiesis

Myelopoiesis is the development of myeloid effector cells. These are granulocytes, which develop in a process called granulopoiesis, and monocytes and macrophages, that evolve during monopoiesis [6].

A myeloid progenitor cell differentiates into a myeloblast, before it enters either granulopoiesis or monopoiesis. These myeloblasts have diameters of 14 to 16 µm, which makes them relatively large, and have a large nucleus with fine chromatin and many nucleoli (see Figure 2.4(a)). In the basophilic cytoplasm, neither granules nor a visibly pale area due to a Golgi-apparatus are present [7].



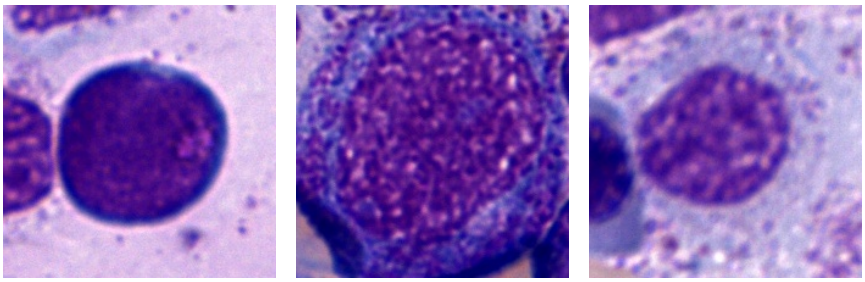
(a) Neutrophil **(b)** Eosinophil **(c)** Basophil

Figure 2.3: Segmented granulocytes from all three lineages. They all have a segmented nucleus with at least two segments, but differ in cytoplasm colour and granulation and, due to the latter, nucleus visibility.

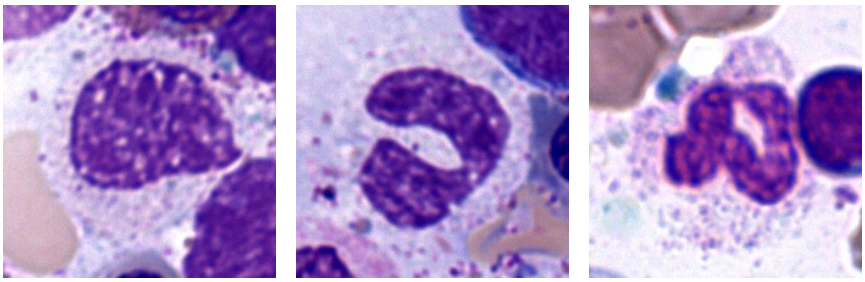
Granulopoiesis

If they enter granulopoiesis, myeloblasts differentiate into neutrophilic, eosinophilic and basophilic lineages (see Figure 2.3), which differ in granulation colour [9]. Neutrophilic granulocytes, or short neutrophils, are relevant for the defence against mainly bacterial infections, and constitute the dominant population of granulopoietic cells in the bone marrow [9]. They have a segmented nucleus, with up to five segments, and pink-blue or grey-blue granules are visible in the cytoplasm (Figure 2.3(a)) [4]. Eosinophilic granulocytes, or eosinophils, on the other hand have a role in allergies and the defence against parasitic diseases. Their nucleus is segmented into two segments, and the cytoplasm is eosinophilic, with red granules filling the entire cytoplasm (figure 2.3(b)) [7]. Lastly, basophilic granulocytes, short basophils, are also involved in the response to parasitic diseases and allergies. Their nucleus is segmented into two parts as well, while the cytoplasm shows a blue or purple, coarse granulation across the cell [7]. These dark, characteristic granules overlap the nucleus (Figure 2.3(c)) [4], which therefore is barely visible, but slightly larger than the nucleus of a neutrophilic or eosinophilic granulocyte [7].

All three lineages of granulopoiesis undergo the following stages of development, as shown in Figure 2.4 for neutrophilic granulocytes.



(a) Myeloblast **(b)** Promyelocyte **(c)** Myelolocyte



(d) Metamyelocytes **(e)** Band granulocyte **(f)** Segmented granulocyte

Figure 2.4: Examples of neutrophilic granulocytes of different maturity from our dataset. The sizes of cells and nuclei tend to decrease, and the initially round nucleus becomes progressively indented and finally segmented.

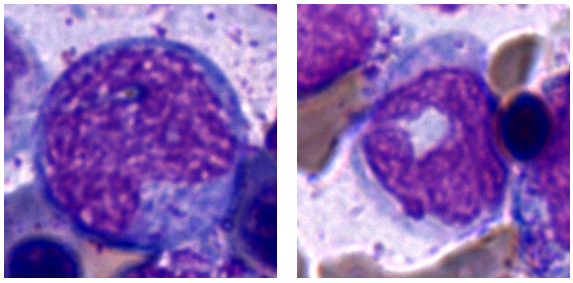
Promyelocytes (Figure 2.4(b)) result from cell division of myeloblasts, so a relatively sharp distinction is possible. In contrast to myeloblasts, they have a brighter perinuclear region due to their Golgi-apparatus, and rough, red granules in the basophilic cytoplasm. The oval nucleus often shows nucleoli, with N/C ranging from 50 to 70 %. The next stage, entered after another cell division, are myelocytes (Figure 2.4(c)). These have smaller, reddish granules in a pink cytoplasm, while the oval nucleus shows more condensed chromatin and usually no nucleoli. N/C is usually between 40 and 50 %. [7]

After the next cell division, the cells are called metamyelocytes (Figure 2.4(d)), which no longer divide and are smaller than myelocytes, with similar cytoplasm. The nucleus is bean- or kidney-shaped, and N/C is approximately 30 %. Metamyelocytes have many fine, but no coarse granules in their cytoplasm. They are further classified into juvenile – often just called metamyelocytes – band and segmented forms. Band granulocytes show a strongly indented, but not yet segmented nucleus (Figure 2.4(e)). The nucleus is defined as segmented (Figure 2.4(f)), if the most narrow connection between two wider areas is smaller than a third of the largest nucleic diameter, otherwise it is accounted as a band form. For the band form, N/C is approximately 20-30%, while it is around 20% for the segmented form, which has

a cell size of approximately 14 µm. [4, 7]

Monopoiesis

Monocytes and macrophages develop during monopoiesis. These cells can perform phagocytosis – which means they can ingest other bodies such as pathogens – present antigens on their surface, and secrete cytokines. Monocytes develop from myeloblasts after undergoing stages called monoblast and promonocyte. With a large, round nucleus and a large cytoplasm, monoblasts have a similar appearance as myeloblasts, so even though they tend to be slightly larger, they can easily be confused with their precursor stage [7]. From a cell division, they emerge as promonocytes, which have an indented nucleus with a horseshoe or kidney-like shape and light blue cytoplasm with few, reddish granules (see Figure 2.5(a) for an example) [7]. The next stage are monocytes, which leave the bone marrow relatively quickly [7]. They have diameters between 15 and 20 µm, and their cytoplasm shows blue or grey (Figure 2.5(b)) stain. It contains many fine vacuoles and often fine, reddish granules, while the nucleus is large and indented and contains clumped chromatin. The nucleus-to-cytoplasm ratio is around 30 % [7]. Monocytes can differentiate into macrophages; however this does not happen in the bone marrow, but in other tissues. Macrophages are larger than monocytes, have a blue cytoplasm, an oval nucleus with well-visible nucleoli, and special vesicles to phagocytose pathogens. [4]



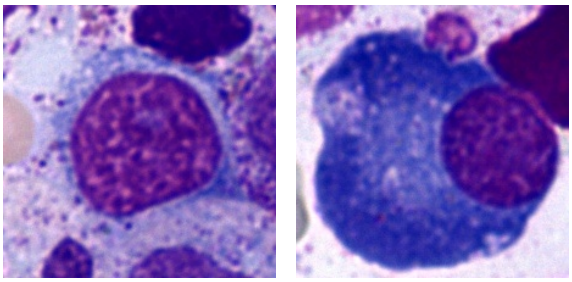
(a) Promonocyte **(b)** Monocyte

Figure 2.5: Examples of cells of monopoiesis from our dataset. The indented nucleus is well visible.

2.1.4 Lymphopoiesis

The development of lymphocytes is called lymphopoiesis [6]. Lymphoid stem cells originate from the haematopoietic stem cells located in the bone marrow [4], immature lymphocytes are also called lymphoblasts [5].

Lymphocytes are divided into B- and T-lymphocytes, which differ in their task as well as the location of their maturation. First stages of the development of B-lymphocytes, whose task is the production of antibodies, take place in the bone marrow, where the lymphoid stem cells differentiate into pro-B- and pre-B-cells. The task of T-lymphocytes on the other hand is to recognise specific antigens on cells.



(a) Lymphocyte **(b)** Plasma cell

Figure 2.6: Examples for a lymphocyte and a plasma cell from our dataset. The nucleus of the lymphocyte is large and round, while the Plasma Cell has a relatively smaller and eccentric nucleus.

In contrast to B-lymphocytes, they migrate to the thymus in an early development stage, from where they are released after maturation. [4]

Both types of lymphocytes have diameters between 8 and 10 µm [7], a single very large, centrally located round nucleus with condensed chromatin with visible gaps, and a pale basophilic cytoplasm (see Figure 2.6(a)) [6,7]. N/C is very large, it ranges between 80 and 90 % [7].

B-lymphocytes can differentiate into plasma cells, whose task is the secretion of antibodies [4]. They usually have diameters between 10 and 14 µm, and a dark blue, basophilic cytoplasm (Figure 2.6(b)). Immature plasma cells have a centric nucleus with 1 to 3 nucleoli, while the nucleus of mature plasma cells is eccentric with dense chromatin with irregular brighter areas, and no visible nucleoli. [7]

2.1.5 Megakaryopoiesis

Megakaryocytes are the largest cells found in bone marrow, their development process is termed megakaryopoiesis [4,5]. They produce thrombocytes (platelets), which are small cell fragments of 3 to 4 µm in a process called thrombopoiesis [6]. Originating from haematopoietic stem cells [4], the first and smallest precursor of megakaryocytes is called megakaryoblast, which looks similar to myeloblasts [5], but is larger with sizes ranging from 25 up to 50 µm. Megakaryoblasts have a very large, centric nucleus with N/C beyond 50 % and non-condensed chromatin often containing nucleoli, while the cytoplasm is basophilic and has no granules. With diameters ranging from 50 up to 150 µm, mature megakaryocytes are very large cells. The nucleus is often strongly segmented and the cytoplasm is basophilic with red-purple granules, which are the forming thrombocytes. [7]

2.1.6 Other Cells Found in the Bone Marrow

The previous sections gave an overview of physiological haematopoietic cells found in the bone marrow. However, also some other cells outside of these haematopoietic lineages, and some pathological forms of haematopoietic cells, can be found.

Pseudo-Pelger forms of granulocytes occur in chronic myelogenous leukaemia (CML) [5] or acute myelogenous leukaemia (AML) [4]. These granulocytes show different sizes, and changes in segmentation [5,6], specifically being bi-segmented [7]. With reticulum cells, a whole group of heterogeneous cells is comprised. Of these, many are members of the macrophage system or derived from blood monocytes. [5] Mast cells are cells that produce histamine and heparin and can sometimes be found as isolated cells in the bone marrow [5]. Presence of masts cells can indicate pathologies [7]. Osteoblasts are the stretched cells that synthesise the bone, and can also sometimes occur in the bone marrow, mainly in children [5, 7].

2.1.7 Normal and Abnormal Frequencies of Cell Types in the Bone Marrow

The relative frequencies of the individual cell types in bone marrow samples from healthy individuals are shown in Table 2.1. The ratio of granulopoietic cells to erythropoietic cells (GE-index) is normally 2.5-3:1 [7].

Cell type	Mean relative frequency [%]	95% confidence interval [%]
Neutrophils	53.6	33.6 - 73.6
Myeloblasts	0.9	0.1 - 1.7
Promyelocytes	3.3	1.9 - 4.7
Myelolocytes	12.7	8.5 - 16.9
Metamyelocytes	15.9	7.4 - 24.7
Band forms	12.4	9.4 - 15.4
Segmented	7.4	3.8 - 11.0
Eosinophils	3.1	1.1 - 5.2
Basophils and mast cells	0.1	
Erythropoietic cells	25.6	15.0 - 36.2
Proerythroblasts	0.6	0.1 - 1.1
Basophilic erythroblasts	1.4	0.4 - 2.4
Polychromatic erythroblasts	21.6	13.1 - 30.1
Orthochromatic erythroblasts	2.0	0.3 - 3.7
Lymphocytes	16.2	8.6 - 23.8
Plasma cells	1.3	0.0 - 3.5
Monocytes	0.3	0.0 - 0.6
Megakaryocytes	0.1	
Reticulum cells	0.3	0.0 - 0.8

Table 2.1: Relative cell frequencies in normal bone marrow samples [6].

Pathologic Deviations from Normal Cell Counts

Pathologies can cause deviations from the normal relative and absolute frequencies of cell types in bone marrow samples. One very important group of such diseases are leukaemias, which are subdivided into acute and chronic forms. The main types are the acute myelogenous leukaemia (AML), the chronic myelogenous leukaemia

(CML), the acute lymphocytic leukaemia (ALL) and the chronic lymphocytic leukaemia (CLL) [5, 7]. Acute leukaemias are characterised by an abnormal count of immature blasts [4], in contrast to chronic leukaemias, where the cells do mature, but in abnormal numbers or forms [7]. Clinical symptoms of acute leukaemias may initially be anemia, infections due to the weakened immune defence, and haematoma or bleeding due to reduced blood clotting, and, depending on the type, infiltration of organs. Chronic leukaemias on the other hand often lack strong specific symptoms. Among the first symptoms can be loss of appetite or weight, physical weakness and often an increased size of the spleen. [9]

AML is characterised by an excessive number of myeloid blasts beyond 20 % of all cells with a nucleus. These blasts show characteristic Auer rods, which are needle-shaped, red inclusions in the cytoplasm. In ALL on the other hand, the B- or T-lymphoblasts lack differentiation, mainly characterised by missing cytoplasmic granulation and a red cytoplasm colour, and exceed 20 % of nucleated cells. [7] CLL is characterised by an excessive number of mature, but defective B- or T-lymphocytes (Lymphocytosis) in the peripheral blood, which is caused by defective apoptosis, programmed cell death, and not by an increased cell division [6]. The lymphocytes appear small [7]. In the bone marrow, CLL leads to an infiltration of mature lymphocytes beyond 30 %, and later a suppression of normal haematopoiesis [6].

Chronic Myelogenous Leukaemia

CML makes up to 15 % of newly diagnosed cases of leukaemia in adults, with an incidence between 0.6 and 2 new cases per 100 000 inhabitants and year in Europe and the United States. Adults are more often affected than children – the median age at the time of a diagnosis is 64 – and men 1.4 (Europe) to 1.7 (USA) times as often as women. CML is fatal if untreated. It has three phases, an initial and mostly asymptomatic chronic phase, an acceleration phase and a blast phase, where the disease has transformed into an acute leukaemia, characterised by a loss of differentiation of the myeloid precursor cells. For diagnosis of CML in the chronic phase, even though blood smear samples are better suited, there are abnormalities in bone marrow samples as well. Predominantly, increased frequencies of cells of granulopoiesis, apart from neutrophilic also including basophilic and sometimes also eosinophilic granulocytes, can be found, with relative frequencies of myeloblasts below 5 % of all cells with a nucleus – in contrast to beyond 20 % in the acute blast phase. The relative frequencies of immature forms are increased, however, in the chronic stage the cells do maturate. Meanwhile, the number and morphology of erythropoietic cells is normal. Consequently, the GE-index is shifted towards granulopoietic cells to values beyond 10:1. Often an increased number of megakaryocytes can be observed, with dysplastic forms called micromegakaryocytes, which have only a size similar to promyelocytes and a round nucleus. Other dysplastic cells are not present. [1]

2.2 Introduction to Deep Learning

In recent years, the concept of artificial intelligence and machine learning has become more and more popular. Two main tasks solved by machine learning are classification, where samples of any kind are assigned to one or multiple given classes, and regression, where, instead of discrete classes, continuous numbers are predicted [10]. Over the years, many different machine learning algorithms and methods have been developed. In image processing, artificial neural networks, in form of so called convolutional neural networks, have been very successful [2, 3, 11–13].

2.2.1 From a Single Neuron to Neural Networks

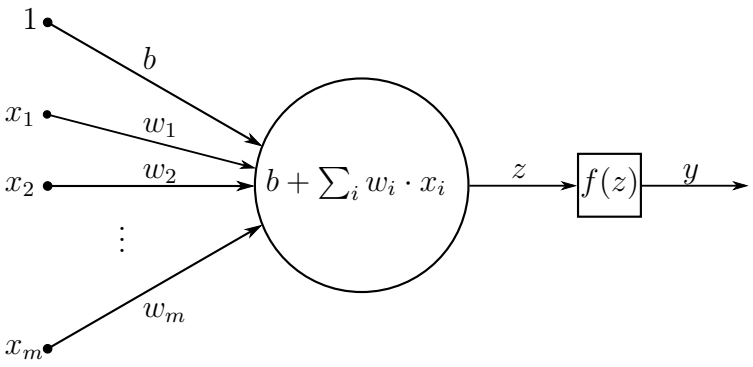


Figure 2.7: A single neuron with $m+1$ weighted inputs, includin a bias. The neuron applies an activation function f to the weighted sum z of its inputs.

In analogy to biological neural networks [14, 15], artificial neural networks consist of a weighted interconnection of basic units, so called neurons [10]. If a network contains no feedback, it is also called a feedforward neural network in contrast to recurrent neural networks [16]. A neuron, in its most basic form, applies a certain activation function to the weighted sum of all input signals [10]. Typically, this activation function introduces non-linearity into the networks [16]. Neurons can have an additional input in form of a bias. Figure 2.7 shows an illustration of such a neuron. Formally, the bias is the weight for a constant additional input of 1. The values of all weights, including the bias, need to be learned for a particular task. Training is described in more detail in Section 2.2.3.

Multiple combined neurons comprise a layer. The most basic type of such a layer is a fully connected layer, where each neuron is connected to every in- and output. The operation conducted by a fully connected layer with m neurons can be written as a matrix operation,

$$\boldsymbol{z} = \boldsymbol{W} \cdot \boldsymbol{x} \tag{2.1}$$

$$\begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix} = \begin{pmatrix} b_1 & w_{11} & \dots & w_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ b_n & w_{n1} & \dots & w_{nm} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_m \end{pmatrix} \tag{2.2}$$

where $\boldsymbol{x}$ is an input vector with $m+1$ entries, including a one for the bias, $\boldsymbol{W}$ is a weight matrix of the format $n \times (m+1)$ including the bias weight, and $\boldsymbol{z}$ is the layer's output with n entries. Finally, an activation function is applied to $\boldsymbol{z}$:

$$\boldsymbol{y} = \sigma(\boldsymbol{z}) \tag{2.3}$$

An commonly used activation function for multi-class classification is the softmax activation, which is defined as

$$S(y_j) = \frac{e^{y_j}}{\sum_i^n e^{y_i}} \in (0,1),\ j \in \{1, ..., n\}, \tag{2.4}$$

where n is the number of classes. The output is a vector, and, since

$$\sum_{j=1}^{n} S(y_j) = 1, \tag{2.5}$$

the individual components can mathematically be interpreted as the estimated class probabilities. [16]

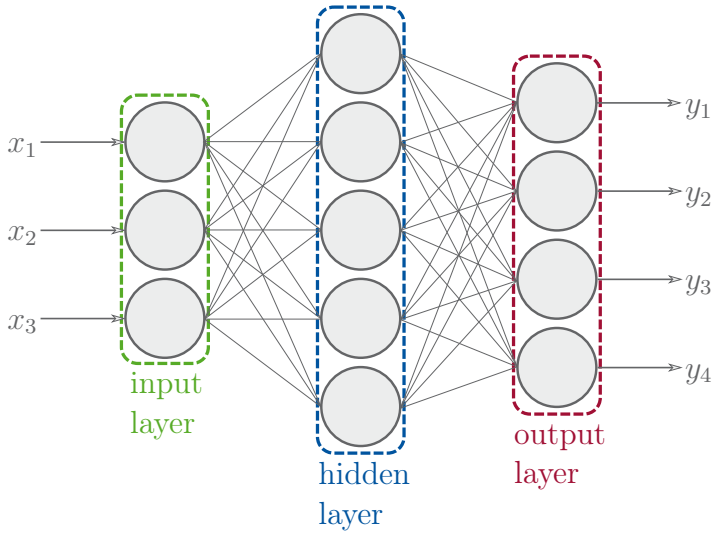


Figure 2.8: A simple feedforward network.

Figure 2.8 shows a simple feedforward neural network. All layers other than the input and output layers are called *hidden layers*. A neural network consisting of more than two or three layers, hence having at least one or two hidden layers, is called deep. [16]

2.2.2 Convolutional Neural Networks

An important special type of feedforward neural networks are convolutional neural networks (CNNs) [17], which are commonly used when the data have a grid-like topology, for example images. They employ a special convolution operation that is performed by a moving window, which is called receptive field. This receptive field of a specified size – usually much smaller than the input – scans the input step by step, moving with an adjustable step size, called the stride. Mathematically, the convolution operation in the image domain is defined as

$$\boldsymbol{Y}(i,j) = (\boldsymbol{K} * \boldsymbol{I})(i,j) = \sum_l \sum_m \boldsymbol{I}(i-l, j-m) \cdot \boldsymbol{K}(l,m), \tag{2.6}$$

where $\boldsymbol{I}$ is the input, for example an image, $\boldsymbol{K}$ is the kernel, and $\boldsymbol{Y}$ is the output. $\boldsymbol{Y}$ can be an image as well, with $\boldsymbol{Y}(i,j)$ being a single pixel at position (i,j). The kernel describes the weights of the input elements in the receptive field, which have to be learned by the network. The same kernel is used regardless of the position of the moving window, which leads to a significant reduction of the number of parameters compared to fully connected layers with the same number of inputs. As another consequence, a translation – in contrast to rotation and scaling – in the input causes an equal translation in the output. If the kernel size is not one, the convolution operation leads to a reduction of the size of the output compared to the input. In order to compensate for this, the input is often padded at the edges, usually with zeros. [18]

A very common activation function in CNNs is the rectified linear unit (ReLU), which is defined as

$$f_{\mathrm{ReLU}}(x) = \max(0, x). \tag{2.7}$$

In ReLU, positive inputs are passed to the next layer, while negative values are blocked and replaced by zero. [16, 18]

If the input has multiple channels, for example three in an RGB image, the same kernel can be used for each input channel. On the other hand, it is also possible to have multiple kernels, causing the output to have more channels, called feature maps, than the input. Following this principle, the data passed through a convolutional network with multiple convolutional layers successively become smaller (if there is no padding) and deeper. Another way to reduce the size of the data being passed from one layer to the next is pooling. Common examples are taking the maximum (*max-pooling*), the minimum or the average value in a non-overlapping moving window. A classification network usually consists of multiple convolutional and possibly pooling layers together with a few fully connected layers at the end. [16, 18]

2.2.3 Training of Neural Networks

Losses

In a supervised training scenario, the network is presented with examples of known classes. These samples are passed through the network and the output is observed.

The basic quantity to evaluate the quality of a neural network, not only during training, is the error, which is estimated by a so-called loss, quantifying the correctness of the observed output of the network. One very common loss for regression and binary (two-class) classification is the mean squared error (MSE), which is defined as

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - t_i)^2 \tag{2.8}$$

where y_i is the network output, i.e. the predicted label in case of classification or the estimated value in case of regression, and t_i is the target, i.e. the true label or value, which is known during training. For classification, often the cross-entropy (CE) loss is used, which in the multi-class case with n classes is defined as

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{n} t_{i,c} \cdot \log(y_{i,c}). \tag{2.9}$$

This means that, if the target is 1 for one single class $\tilde{c}$ and 0 otherwise, only the output $y_{i,\tilde{c}}$ is relevant for the loss. The overall loss is the average of these contributions over all samples. Depending on the particular task and network architecture, there are many other possible loss definitions, which often have to be engineered for the particular task. [16]

Backpropagation and Weight Updates

During training, the weights of the neural network are updated based on the loss on the current samples. To determine appropriate adjustments of the weights, a backpropagation of that loss is performed. In order for this to work, it is important that the weights of the network are initialised for example randomly in a small interval around zero [10].The problem of initialisation is discussed more thoroughly in Section 3.1.3.

Backpropagation of the loss involves the calculation of its partial derivative with respect to the individual weights

$$\frac{\partial \mathcal{L}}{\partial w_i}. \tag{2.10}$$

Since the weights can be represented as a vector, this can also be written as a gradient of the loss $\nabla_{\boldsymbol{w}}\mathcal{L}$. In a gradient descend, the goal is to decrease the loss by moving in the direction of the negative gradient [18] In vector notation, the weights are updated according to

$$\boldsymbol{w_{new}} = \boldsymbol{w_{old}} - \alpha \cdot \nabla_{\boldsymbol{w}}\mathcal{L}, \tag{2.11}$$

where α is the so-called learning rate, which is a positive scalar [18] hyperparameter, which has to be selected very carefully. A large learning rate can make training faster, but might cause suboptimal results, because the influence of an individual

learning step can become too big. On the other hand, a low learning rate makes training slower and more prone to become stuck in a local minimum [16].

In an attempt to avoid getting stuck in local optima, there are several approaches, including adaptive learning rates – usually α is decreased during training – and an extension of the gradient descent with momentum [10]. One popular algorithm based on the latter idea is adaptive moment estimation (Adam) [19], which uses two moments, an estimate of the first moment (the mean) and an estimate of the second moment (the variance) of the gradient. For this purpose, moving exponential averages are used, where the two additional parameters $\beta_1, \beta_2 \in [0, 1)$ control the exponential decay rate of these averages.

Dataset Splits

In practice, the number of samples available to train neural networks is usually limited. Often the data are divided into three disjoint sets: The training set, the validation set and the test set. While the actual training is only performed on the training set, possibly repeatedly in multiple epochs, the other two sets are used solely for evaluation. This is necessary because of the effect of overfitting, which occurs when the noise in the training data is modelled. In case of overfitting, the performance on the training data is very high, but the model does not generalise well to unseen data [10]. The validation set is used to track the performance of the network during training by measuring the loss on unseen data. It is often used to find an appropriate stopping point of the training, as an alternative to a much more naive stopping after a fixed number of epochs, or to select the best-performing model state that occurred during training. In many cases, it can be observed that during training the validation loss reaches a minimum and then begins to rise again due to overfitting, indicating that from this point the model loses its capability to generalise [10]. The test set on the other hand is used only for the evaluation of the final, trained and selected network. This third set is required to obtain an unbiased measure, which the validation set cannot provide, because it has already been used for model state selection. [16]

Practical Optimisation Techniques

Often, the samples are passed to a neural network in batches of multiple samples, instead of the entire training set at once. This method is called stochastic gradient descent. Stochastic gradient descent often converges much quicker compared to full gradient descent, because the random samples frequently allow a good approximation of the overall gradient. However, it might fail in cases where the gradient is not very steep [16]. In batch-wise training, it is advisable to shuffle the training set such that successive examples are most likely dissimilar. Additionally, it is useful to normalise the inputs by element-wise subtraction of the mean $\boldsymbol{\mu}$ of the whole dataset (or training set) and then scaling the input variables such that the covariances are similar by division by the standard deviation $\boldsymbol{\sigma}$ of the dataset:

$$\boldsymbol{x}_{norm} = \frac{\boldsymbol{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \tag{2.12}$$

Normalisation removes a potential bias of weight updates and ensures equal contribution of all input variables. [20]

To address a similar problem within the network, batch normalisation [11] can be employed, which is batch-wise normalisation of the input of a layer. Batch normalisation addresses the problem that a layers' input distribution changes due to changes of all previous layers during training.

2.2.4 Evaluation of the Performance of a Classifier

Not only in deep learning, but for any classification approach, one or multiple meaningful measures are required to objectively evaluate a classifier's performance. Such a measure is important because of multiple reasons – firstly to (at least approximately) know the reliability of the output when working with a trained model in general, and secondly for model selection, either between different alternative models, or to select the best model state that occurred during training based on the performance on the validation set [10].

For each example of a class in the dataset, the classifier can either correctly predict it to be a member of this class, giving a true positive (TP) result, or falsely predict another class, leading to a false negative (FN) result. Conversely, every sample of all other classes can either be predicted to be of (any) other class, leading to a true negative (TN) result with respect to the considered class, or be predicted as member of the class, meaning that the prediction is false positive (FP).

The overall results can be represented by a confusion matrix [21], in which an entry at position (i, j) represents how often a sample from class i has been predicted to be member of class j. Diagonal entries ($i = j$) represent the number $N_{\mathrm{TP},i}$ of true positive predictions. The sum of all non-diagonal entries in row i is the number $N_{\mathrm{FN},i}$ of false negative predictions, assuming that rows correspond to the true and columns to the predicted class. Conversely, the sum of non-diagonal entries in the j-th column $N_{\mathrm{FP},j}$ represents the number of false positive outcomes for the corresponding class. Lastly, $N_{\mathrm{TN},i}$ of class i can be calculated as the sum of all entries that are located neither in the i-th row, nor in the i-th column. From these numbers, different evaluation measures can be calculated.

Accuracy and Error Rate

The two most basic measures for the performance of a classifier are the accuracy and the error rate. The accuracy is defined as the relative amount of correct classifications. For general multi-class classification, it can be written as

$$\text{accuracy} = \frac{\sum_{i=1}^{n} N_{\mathrm{TP},i}}{\sum_{i=1}^{n} N_{\mathrm{TP},i} + \sum_{i=1}^{n} N_{\mathrm{FP},i}}. \tag{2.13}$$

The error rate is defined as

$$\text{error rate} = 1 - \text{accuracy} \tag{2.14}$$

and describes the relative frequency of errors with respect to the total number of predictions [10].

F-score

The approach of using error rate and accuracy as the only measures of a classifier's performance can be problematic, especially in case of a significant imbalance in the dataset between the number of examples from the individual classes. If there are relatively few examples of a certain class, a classifier that never predicts this class is practically useless, but still achieves a high accuracy. In such cases, more appropriate measures are precision and recall, and derived from these, the F-score. The precision of a classifier

$$\text{precision} = \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FP}}} \tag{2.15}$$

is defined as the relative amount of positively classified samples which are indeed positive. On the other hand, the recall

$$\text{recall} = \frac{N_{\text{TP}}}{N_{\text{TP}} + N_{\text{FN}}} \tag{2.16}$$

describes the relative number of positive examples that are classified as such. Now, if there is a class imbalance and the classifier never predicts a small class, precision and recall would be low (or undefined) for that class, even though the accuracy would be high and the error rate low. [10]

Depending on the particular task, the precision might be more important than the recall, or vice-versa. A single measure that combines precision and recall is the F-score

$$F_\beta = \frac{(\beta^2 + 1) \cdot \text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}. \tag{2.17}$$

The parameter $\beta \in [0, \infty)$ allows to weigh precision and recall: If $\beta > 1$, recall is given a higher importance, while for $\beta < 1$ the precision is emphasized. Often, β is simply chosen as 1, leading to the F_1-score

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{2.18}$$

The F_1-score can be generalised in two ways to multi-class classifications – the micro-averaged and the macro-averaged F_1-score. Calculating precision and recall from the overall numbers of correct and incorrect classifications

$$\text{precision}_{\text{micro}} = \frac{\sum_{i=1}^{n} N_{\text{TP},i}}{\sum_{i=1}^{n} (N_{\text{TP},i} + N_{\text{FP},i})} \tag{2.19}$$

$$\text{recall}_{\text{micro}} = \frac{\sum_{i=1}^{n} N_{\text{TP},i}}{\sum_{i=1}^{n} (N_{\text{TP},i} + N_{\text{FN},i})} \tag{2.20}$$

leads to the micro-averaged F_1-score

$$F_{1,\text{micro}} = \frac{2 \cdot \text{precision}_{\text{micro}} \cdot \text{recall}_{\text{micro}}}{\text{precision}_{\text{micro}} + \text{recall}_{\text{micro}}}. \tag{2.21}$$

From averaging of all individual decisions, it follows that the micro-averaged F_1-score implicitly assigns a higher weight to classes with more examples. In fact, in the most common case where each sample is classified into exactly one class, the micro-averaged F_1-score is equivalent to the accuracy [2].

Because of the weaknesses of the micro-averaged F_1-score in case of class imbalances, often the macro-averaged F_1-score is preferred, which is defined as

$$F_{1,\text{macro}} = \frac{1}{n} \cdot \sum_{i=1}^{n} F_{1,i}. \tag{2.22}$$

The F_1-scores of all n classes are averaged directly, consequently all classes have the same importance.

[2] see Appendix A.1 for proof.

3 State of the Art

While the previous chapter has given important, but general background knowledge about haematopoiesis and deep learning, this chapter introduces more specific state-of-the-art methods. In Section 3.1, two very successful network architectures are described in detail. Section 3.2 addresses the application of machine learning methods on a hierarchical domain, while Section 3.3 gives examples of prior work employing some kind of cascade of neural networks. Section 3.4.1 gives a brief introduction to clustering, focusing on spectral clustering algorithms. Finally, Section 3.5 provides more information about relevant prior work and findings of our haematology project.

3.1 Successful Deep Learning Architectures

In this work, two established deep learning architectures are used that have shown the most promising results on our dataset in previous investigations [3]. When comparing neural network architectures, it is common to evaluate their performance on benchmark datasets. One example for such a benchmark is the ImageNet Large Scale Visual Recognition Competition (ILSVRC) with approximately 1.2 million training images, 50 000 validation and 100 000 test images from 1000 classes, ranging from different animals over food to objects like vehicles or electronics [22, 23]. The performance on the ImageNet dataset is usually reported in terms of the top-1 error-rate, measuring the rate of cases where the correct class has not been assigned the highest confidence by the network, and the top-5 error rate, which is the relative frequency of cases where the correct class is not among the 5 classes with the highest predicted probability. The two architectures introduced in the following are ResNet (Section 3.1.1) and DenseNet (Section 3.1.2).

3.1.1 ResNet

Late 2015, He et al. introduced the idea to include residual connections within deep networks to overcome problems of performance degradation when networks become too deep. The resulting network architecture is called *ResNet* [12].

Its basic building block, as shown in Figure 3.1, comprises of two successive convolutional layers, and a shortcut connection, which skips both of these layers. Alternatively, to reduce training time, the building blocks may consist of three layers, with reduced dimensions in the intermediate so-called bottleneck layer. Regardless which building block design is used, the input is simply added to the output of the two successive layers, without being modified apart from dimensions matching. Within

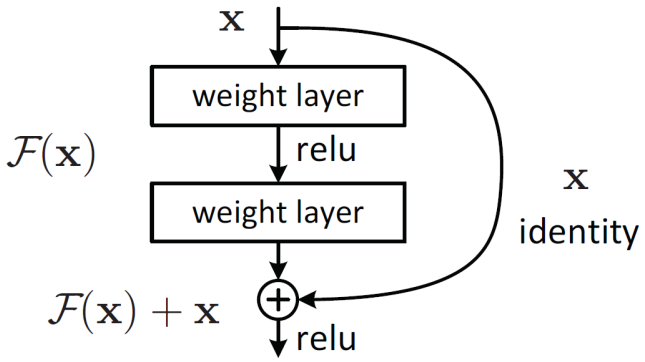


Figure 3.1: Basic building block of a ResNet. Via skip connections, the input is added to the output. Source: [12]

the ResNet, at some points downsampling is performed by application of a stride of 2. At these points, the number of filters is doubled, while the feature map size is halved. Consequently, due to the dimension mismatch, the residual connection has to be adjusted, either by padding or by using a linear projection, implemented with 1×1 convolutions.

ResNets allowed to train deeper networks than before. As a consequence, ResNets have been very successful, for example on ILSVRC, on which, when they were published, ResNets of different depths achieved lower error rates (top-1-error-rate: 19.38 % and top-5-error-rate 4.49 % with a depth of 152) than any previous contender. Common depths for ResNets are 18, 34, 50, 101 or 152, the corresponding numbers and types of building blocks are given in Figure 3.2. Additionally, the number of operations performed by the network (FLOPs) is stated, assuming classification on the ImageNet dataset with images of size 224×224 belonging to 1000 classes. Regardless of the depth, the networks contain one convolutional layer with a kernel size of 7×7, followed by 4 parts comprised of the basic building blocks, and are concluded by a single fully connected layer followed by a softmax activation. [12]

3.1.2 DenseNet

In 2017, Huang et al. published an architecture called *DenseNet* [13], which also involves skip connections to allow particularly deep networks. However, in contrast to ResNets, DenseNets have multiple shortcut connections from every block not only to the next, but to all successive layers within blocks. The basic building block of a DenseNet consists of a sequence of batch normalisation, ReLU and convolution, to which, as in ResNets, an additional bottleneck layer can be added. The principle of a dense block is illustrated in Figure 3.3

Each dense block with L layers contains $\frac{L(L+1)}{2}$ connections. Within these so-called dense blocks, all feature maps have the same dimensions, however, between the dense blocks there are transition convolutional and pooling layers, which reduce the feature map sizes. A key distinction to ResNets is that in DenseNets the inputs

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 3.2: Possible ResNet configurations when applied on ImageNet. All variants have in common that they consist of 5 blocks. The individual cells show the basic building blocks in brackets together with their number. Source: [12]

and outputs are concatenated, and not added. Due to this concatenation, the l^{th} layer within a dense block has

$$k_0 + k \cdot (l - 1) \tag{3.1}$$

input feature maps, where k is the growth rate, a parameter that can be relatively small (for example $k = 32$). Consequently, the individual layers in DenseNets can be narrow, meaning the number of filters is relatively low. Because of the additional shortcut connections, feature maps learned in early network layers do not need to be re-learned by later layers, reducing feature redundancy and allowing for a decreased number of parameters. Conversely, while the performance in the ImageNet challenge has been similar to ResNets (top-1-error-rate: 20.80 % and top-5-error-rate 5.29 %), DenseNets achieved these results with significantly fewer parameters. Figure 3.4 shows the architectures of DenseNets in more detail, when applied to classification on the ImageNet dataset with 224×224 images from 1000 classes. Common depths are 121, 161, 169 and 201. There are certain similarities to the ResNet architecture (Figure 3.2): In both architectures, all variants consist of four blocks, comprised of the respective basic building block, and similar first and final layers. [13]

3.1.3 Weight Initialisation and Transfer Learning

As already mentioned in 2.2.3, it is important to initialise the weights of the neural network before training. While an initialisation with zeros would prohibit any convergence, too large initial weights which lead to saturation in an output sigmoid would lead to very small gradients and therefore slow training. Ideally, every neuron's weights should be selected randomly with zero mean, and the standard deviation $\sigma_w = m^{-1/2}$, where m is the number of inputs to a certain unit. [20]

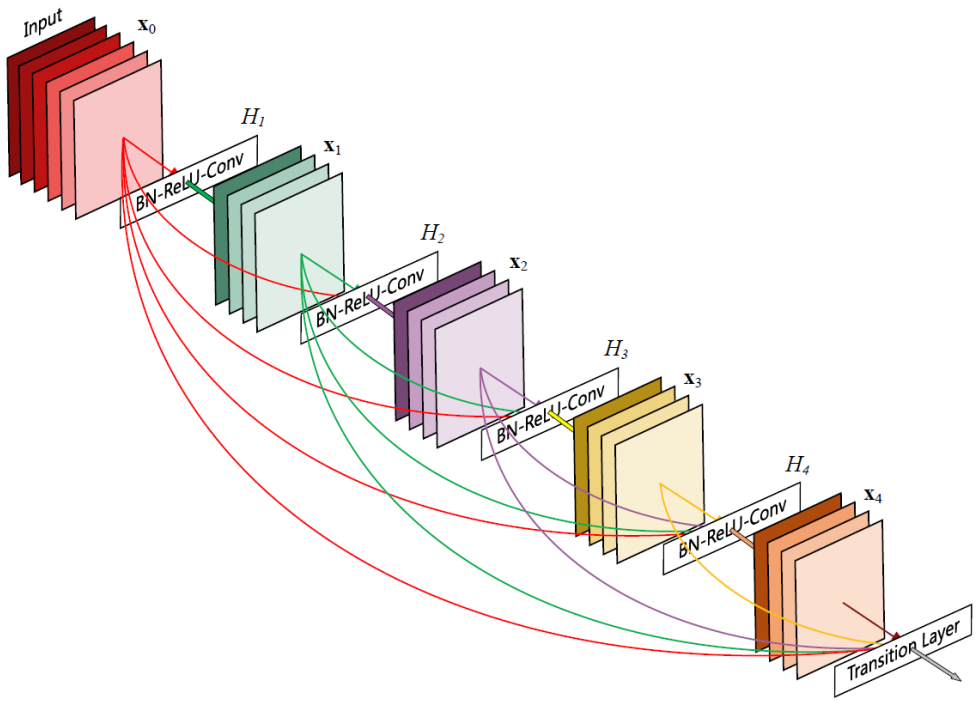


Figure 3.3: Illustration of a dense block with a growth rate of $k = 4$. Each layer gets the concatenated feature maps of all previous layers as its input. Source: [13]

Another way a neural network's parameters can be initialised is to adopt the weights of another network with the same architecture which has already been trained on a different dataset [24]. This is called *transfer learning*.

In case of insufficient training data, starting from a model which is pretrained on a large-scale dataset can improve the network's performance. Especially features learnt in early layers of a network, such as edges or corners in images, are simple but universal. Exploiting this, transfer learning can both speed up the training and also be helpful especially in cases of insufficient training data, even when pretrained on a very different domain [24, 25].

One commonly used dataset for pre-training is the ILSVRC dataset [22, 23]. Even though, naturally, it would be preferable to perform the pretraining with data from a similar domain, transfer learning based on the heterogeneous ImageNet dataset has successfully been utilised for many different tasks.

3.2 Hierarchical Classification

3.2.1 Basic Terminology and Task Definition

Task Definition

In a 2010 survey, Silla and Freitas [26] defined hierarchical classification as classification given a pre-defined taxonomy, either in form of a tree, where every node has

Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 36$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figure 3.4: Possible DenseNet configurations when applied on ImageNet. All variants have in common that they consist of 4 blocks. The individual cells show the basic building blocks in brackets together with their number. As *conv*, a sequence of batch normalisation, ReLU and convolution is abbreviated. Source: [13]

at most one parent, or a directed acyclic graph (DAG). Formally, such a taxonomy can be defined as a partially ordered set $(C, \prec)$ using the "is-a" operator $\prec$, as was done in 2005 by Wu et al. [27]: Let $C = \{c_0, ..., c_n\}$ be the set of classes ordered within a tree, and let c_0 denote its only root. Then it holds

$$\forall c_i \in C \backslash \{c_0\} : c_i \prec c_0, \tag{3.2}$$

meaning that all other classes are at the same time also members of the root class c_0. Furthermore, the operation is defined as anti-reflexive

$$\forall c_i \in C : c_i \not\prec c_i \tag{3.3}$$

and transitive

$$\forall c_i, c_j, c_k \in C : c_i \prec c_j \; and \; c_j \prec c_k \Rightarrow c_i \prec c_k, \tag{3.4}$$

which means that if c_j is an ancestor of c_i, all ancestors of c_j are also ancestors of c_i. Extending this definition, Silla and Freitas [26] require the "is-a" relation to be asymmetric

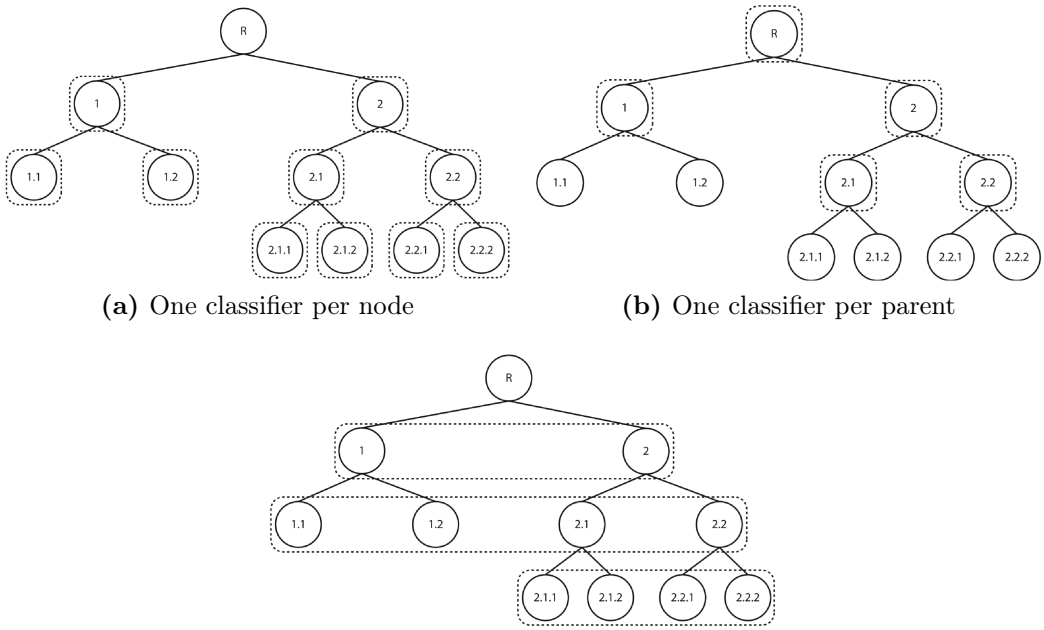
$$\forall c_i, c_j \in C : c_i \prec c_j \Rightarrow c_j \not\prec c_i, \tag{3.5}$$

assuring that the tree or DAG is acyclic.

When classifying within such a taxonomy without allowing inconsistencies, every assigned class c_i implicitly also assigns the partially ordered set $(C_i = \{c_i\} \cup \{c_j \in C : c_i \prec c_j\}, \prec)$, because when classifying into a class, the sample is implicitly also classified into all of its ancestor classes because of the properties of the "is-a" operation. The set C_i is called the structured label, or multi-label, containing all assigned labels, the root being the most general and the leaves representing the most specific labels [27].

Local and Global Approaches in Contrast to Flat Classifiers

Given a taxonomy, there are different approaches to incorporate its information [26]. In *global* approaches, the entire hierarchy is taken into account, but its knowledge is used to learn a single classifier, while *local* approaches apply several local classifiers to propagate through the tree or DAG. Consequently, local approaches have the advantage of modularity, because of the separate local classifiers, which can be selected and trained individually. A *flat* classification method on the other hand ignores the relationships between classes completely, in many cases just predicting the leaf nodes of the taxonomy. In comparison to individual local classifiers, this usually means that the classifier has to deal with more possible classes at once.



(a) One classifier per node

(b) One classifier per parent

(c) One classifier per level

Figure 3.5: Visualisation of the three approaches to local hierarchical classification. Circles represent classes, with R indicating the root, while dashed boxes show the position of classifiers. In contrast, global approaches incorporate the domain knowledge by other means, while a simple flat network just considers the final classes, i.e. usually the leaves, in a single classifier. Source: [26]

Local approaches to hierarchical classification can be one of three types, shown in Figure 3.5. The first possible method is to have one binary classifier *per node* (Figure 3.5(a)) except for the root. Each local classifier then performs a binary classification, predicting whether or not a given sample belongs the node's class. Another approach is to establish one multi-class classifier *per parent* (Figure 3.5(b)), which predicts to which of its children a sample belongs. This leads to fewer – although there is a classifier at the root now, the leaves have no classifier anymore – but more complex

classifiers, which have to take different numbers of classes into account. The third possibility, further decreasing the number of classifiers, is to have a single multi-class classifier *per level* (Figure 3.5(c)) across all branches.

In all these approaches, the prediction is usually performed using a top-down propagation through the hierarchy. Starting from the root, the sample is at first assigned to more general classes, before moving to the most specific classes represented by the leaves. Because of this stepwise approach, it must be noted that in general, depending on the exact implemented decision strategy, this approach can be sensitive to error propagation – in case whole branches are neglected because of a high-level classification. It can also lead to class inconsistencies, contradicting the transitivity (3.4) due to assigning labels from different branches at different nodes. [26]

3.2.2 Hierarchical Evaluation Measures

Given a taxonomy, it is possible to adjust the evaluation metrices to account for different levels of correctness of a prediction. In many cases, it is much better to misclassify two very close examples than samples from more distant classes in the hierarchy. For example, classifying a cat as a dog still implies that the object is an animal, while predicting a cat to be a car would potentially be much worse. Classical evaluation measures for flat classification, such as the F-score, cannot distinguish between misclassifications between close and distant classes. Similarly, standard classification losses such as the cross-entropy loss do not distinguish how far off a misclassification really is. Therefore, it makes sense to also consider other evaluation metrices and losses.

In 2005 Kiritchenko et al. [28] defined a hierarchical F-score. Given a DAG or a tree, they defined the micro-averaged hierarchical precision as

$$hP = \frac{\sum_{i=1}^{N} |\hat{C}_i \cap \hat{C}'_i|}{\sum_{i=1}^{N} |\hat{C}'_i|} \tag{3.6}$$

and the micro-averaged hierarchical recall as

$$hR = \frac{\sum_{i=1}^{N} |\hat{C}_i \cap \hat{C}'_i|}{\sum_{i=1}^{N} |\hat{C}_i|}, \tag{3.7}$$

where $\hat{C}_i = C_i \backslash \{c_0\}$ and $\hat{C}'_i = C'_i \backslash \{c_0\}$ are the true and predicted multi-labels of the i-th sample, excluding the root. In other words, hP measures the relative amount of correctness of the predicted multi-label, while hR quantifies the correctly predicted fraction of the true multi-label. From hP and hR, the hierarchical F-score can be calculated as usual:

$$hF_\beta = \frac{(\beta^2 + 1) \cdot hP \cdot hR}{\beta^2 \cdot hP + hR}, \ \beta \in [0, \infty) \tag{3.8}$$

3.2.3 Hierarchical Losses

To incorporate the hierarchical knowledge not only into evaluation, but already into training, some researchers also introduced hierarchical losses. In an article published in 2006, Cesa-Bianchi et al. [29] proposed a very simple loss called, h-loss. For the h-loss, the tree is propagated from the root and whenever there is a mismatch between the correct and the predicted – not necessarily consistent – multi-label, the loss is increased by one and the propagation of the current path is stopped. In 2019, Wu et al. [30] proposed a different hierarchical loss for the special case of ultrametric trees, which are trees where every leaf has the same number of ancestors. They define the win of a prediction from the probabilities for all nodes, as given for example by the output of a Softmax function, weighted by $1/2^{\text{depth}}$. By counting the leaves twice, the weights sum up to one. The loss is then simply defined as the negative win, or alternatively its logarithm. However, in their article, the authors come to the conclusion that this loss is not well suited for plain classification training, but may be more beneficial when optimised using a hierarchical training process.

3.3 Cascades of Neural Networks

Already in 1993, Poddar and Rao [31] employed a hierarchical ensemble using neural networks as per-parent local classifiers, following a Bayesian approach to estimate a posteriori probabilities of the classes. Individual, simple neural networks are trained for particular classification sub-tasks. This training is performed only on the for this sub-task relevant subset of the dataset. Using the Bayes rule, propagation through the network is then carried out recursively: Let C_i be the multi-label of a non-root node ($i \neq 0$), and $C_j = C_i \cup \{c_j\}$ the multi-label of one of its children, and let $\boldsymbol{x}$ be a data sample to be classified. For any direct child k of the root, the estimate for $P(C_k|\boldsymbol{x})$ is directly taken from the root's network output. Given an input $\boldsymbol{x}$, the a posteriori probability assigned to all other nodes C_j is calculated as

$$P(C_j|\boldsymbol{x}) = \frac{P(C_j, \boldsymbol{x})}{P(\boldsymbol{x})} \tag{3.9}$$

$$= \frac{P(C_j, \boldsymbol{x}, C_i)}{P(\boldsymbol{x})}, \text{ because } C_i \subset C_j \tag{3.10}$$

$$= \frac{P(C_j|\boldsymbol{x}, C_i) \cdot P(C_i, \boldsymbol{x})}{P(\boldsymbol{x})} \tag{3.11}$$

$$= P(C_j|\boldsymbol{x}, C_i) \cdot P(C_i|\boldsymbol{x}). \tag{3.12}$$

By training the individual networks only on their relevant subset, their output can be viewed as a direct estimate of $P(C_j|\boldsymbol{x}, C_i)$. Furthermore, the authors showed that, in principle, such a neural network is equivalent to another, single neural network, however, for neural networks that are not deep, they found that their approach outperformed such a single network in terms of both required time for training and classification error.

The idea of using a cascade of much more complex *deep* neural networks is, however, not very common. Nevertheless, there are some examples where deep neural networks are cascaded for other tasks, with similar ideas how they are trained and how outputs are combined.

In 2014, Girshick et al. [32] published their method called *R-CNN* for object detection. Region proposals, obtained by any suitable method, are given to a convolutional neural network, which then performs feature extraction. These features are finally fed to a classifier. In 2015, Girshick published an improvement which he called *Fast R-CNN* [33], where the classifier is a feed-forward neural network consisting of several fully connected layers, leading to a cascade of two neural networks. In the same year, Ren et al. [34] proposed to use a region proposal network, so that there is a neural network at every step, which however share convolutional layers. Training of the region proposal network and the Fast R-CNN detection network is done in an alternating fashion, which consists of four steps. The first two steps involve separate training of the region proposal network and the detection network. Finally, in steps three and four, the networks share convolutional layers which are then fixed. During these training phases, only the layers unique to first the region proposal and lastly the detection network are fine-tuned to optimise the overall performance.

In 2017, Sobokrou et al. [35] published a casade of Deep Neural Networks used for anomaly detection and localisation in crowded scenes, which employs a similar idea as Faster R-CNN. Their approach involves two major stages. As a first step, a deep autoencoder detects a large number of initial region proposals. These proposals are then resized and fed to a deep convolutional network for affirmation or rejection. Additionally, both of these networks are cascaded in themselve, having weak gaussian classifiers at the output of intermediate layers.

Also in 2017, Schlemper et al. [36] developed a cascade of deep convolutional networks for reconstruction of undersampled sequences of magnetic resonance images. Their approach involves successively concatenating neighbouring samples and feeding them through a series of neural networks to perform de-aliasing and image reconstruction. The individual networks in this approach are a series of convolutional layers, bypassed by a single shortcut connection, so that the input is added to the output of the final layer of each network.

3.4 Other Machine Learning Methods

In this section, two machine learning algorithms are introduced, which are relevant for some parts of this work. These algorithms are spectral clustering and UMAP.

3.4.1 Spectral Clustering

Clustering as a machine learning task is defined as grouping unlabelled objects into disjoint subsets, or clusters, such that samples within clusters are as similar as possible, while samples from different clusters are dissimilar. There are several popular

and relatively simple clustering techniques, such as k-means [37,38] or its nonlinear variants like kernel k-means , or the expectation maximisation algorithm [39]. Another popular clustering algorithm is spectral clustering [40].

The input to a spectral clustering algorithm are a symmetric similarity matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ and the number k of clusters to be found. From $\boldsymbol{S}$, an undirected similarity graph $G = (V, E)$ with a weight matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ is built. S should model the local neighbourhoods only. This can be done by different means, for example by using heuristics like an ϵ-neighbourhood or connecting a node only with a specified number of nearest neighbours, or by having a fully connected graph with distances according to $\boldsymbol{S}$. In this graph, each node has a certain degree

$$d_i = \sum_{j=1}^{n} w_{i,j}, \tag{3.13}$$

which can be written in a diagonal degree matrix $\boldsymbol{D}$ with entries $d_1, ..., d_n$ on the main diagonal, and zeros in all other positions. [40]

In the next step, the Laplacian matrix

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}, \tag{3.14}$$

is calculated from $\boldsymbol{W}$ and $\boldsymbol{D}$. There are different variants of spectral clustering, with the main distinction between unnormalised and normalised [41,42] variants. In the unnormalised version, the next step is to calculate the first k eigenvalues and eigenvectors $\boldsymbol{u_1}, ..., \boldsymbol{u_k}$ of $\boldsymbol{L}$ [40]. In the normalised spectral clustering according to Shi and Malik [41], the generalised eigenproblem $\boldsymbol{LU} = \lambda \boldsymbol{Du}$ is solved instead. Regardless of which version of the algorithm is used, an eigenvector matrix $\boldsymbol{U} \in \mathbb{R}^{n \times k}$ is built from the first k eigenvectors with the smallest eigenvalues, which are inserted as the columns. $\boldsymbol{U}$ represents a transformed space, in which then the actual clustering is performed for example using k-means for every row of $\boldsymbol{U}$ representing one sample. [40]

Often, the number k of clusters is not known in advance. In this case, there is a useful heuristic that can help to decide on how many clusters to search for – the *eigengap heuristic*. When looking at the sorted eigenvalues, the goal is to choose k such that $\lambda_1, ..., \lambda_k$ are close to zero, while λ_{k+1} is significantly larger. [40]

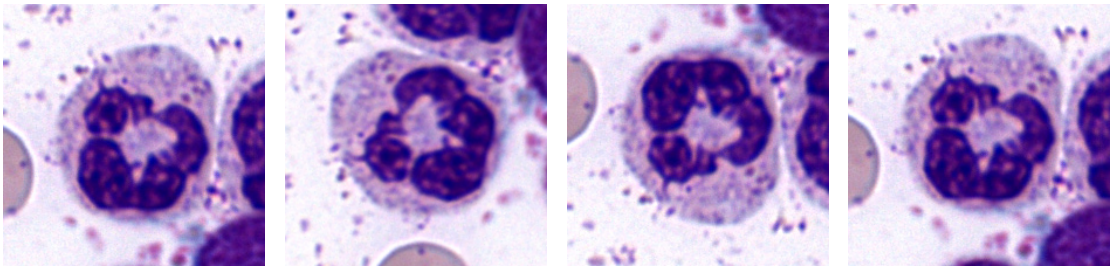
3.4.2 UMAP

The goal of dimensionality reduction is to represent high-dimensional data in a low-dimensional space while preserving certain characteristics, such as clusters. Uniform Manifold Approximation and Projection (UMAP), introduced by McInnes et al. in 2018 [43], is a non-linear algorithm for this task. It is based on Riemann geometry and algebraic topology and involves two phases. In the first step, a weighted k-neighbourhood graph is created, connecting samples that are similar in the high-dimensional space. Afterwards, a low-dimensional layout of this graph is computed in a second step.

3.5 Previous Work and Findings of the Haematology Project

As already mentioned in the introduction (Chapter 1), this work is part of a haematology project at RWTH Aachen University. The goal of this project is to determine the distribution of the different cell types in whole slide bone marrow images. In the following, some of the relevant previous work connected to this project is introduced.

In 2018, different approaches for classification of cell images from our dataset were tested and compared. Apart from deep learning in form of a pretrained ResNet with depth 18 and 152, classical approaches that involve a feature extraction and a classification step were evaluated. Different features such as the histogram of oriented gradients [44] and local binary patterns [45] as well as features extracted by convolutional networks, particularly ResNet-18 and ResNet-152, were evaluated. The classifiers that were applied on these features were linear [46] as well as radial basis function support vector machines [47], random forest classifiers [48] and AdaBoost [49]. Among all these methods, the deep learning approach outperformed the classical approaches in terms of the (logarithmic) F-score. [2]



(a) Original **(b)** Rotation **(c)** Mirroring **(d)** Cropping

Figure 3.6: Visualisation of the three approaches to data augmentation that yielded the best results in prior examinations [3].

A deeper analysis of the application of single deep networks on our dataset, including the evaluation of different hyperparameter setups, was performed in 2019 [3]. Out of several different compared setups, DenseNet-121, closely followed by ResNet-152, performed best, while deeper DenseNets performed worse possibly because of stronger overfitting on the used subset of our dataset. In the same work, different further optimisation techniques were examined. Transfer learning based on networks pretrained on the ILSVRC dataset have proven to significantly improve the results and it was shown that the weights especially of early layers, providing the most general features, barely changed from that pretrained state. For the learning rate, it was shown that in this setting, values in the range 10^{-4} down to 10^{-6} gave the best results while not prolonging the training too extensively. Analysis showed that the influence of the batch size is not very large for batch sizes between 4 and 64. To partly improve training given the relatively small dataset, it was shown that simple image transformations for data augmentation [50] can help to improve the classification performance. Successful strategies were mirroring the image in the vertical or

horizontal direction, rotation around a random angle, and cropping the image with a small, random displacement, effectively slightly translating the cell within the image. Examples for these transformations are shown in Figure 3.6. Counteracting the large class imbalance, weighted cross-entropy loss (WCE), where each class receives a weight [51] based on its inverse frequency, was shown to slightly increase the performance of the classifier. Other methods that were analysed, but did not improve the results, were undersampling of overrepresented classes, and oversampling of underrepresented classes.

Another, unpublished work in 2019 showed that, in principle, it is possible to apply regression instead of classification for maturity estimation in a specific lineage, in this case neutrophilic granulocytes. It can be converted into classification by thresholding. This approach, by exploiting the ordinal character of these specific class labels, proved to reduce the confusion between more distant maturation stages.

For automated detection of cells in whole slide images, a neural network based on the successful RetinaNet architecture [52] is used, however using circular anchors instead of rectangular bounding boxes to improve detection accuracy. In case of overlapping anchors, a strategy called *Area-based Nonmaximum Suppression* is employed to reduce the number of false positives. If a candidate area overlaps with another area, such that the relative overlap exceeds a certain threshold, and if its confidence is lower than the confidence of the area it overlaps with, it is discarded. [53]

4 Neural Network Cascades

The methods and principles developed in this work are introduced in this chapter. In Section 4.1 the haematology dataset is introduced and analysed. Section 4.2 provides an overview over the hierarchical classification approach using deep neural networks employed on these data. Within section 4.2, two alternative ways to traverse through a hierarchical cascade of neural networks are described in 4.2.2. Next, the training strategies for such a cascade are discussed in Section 4.3, before Section 4.4 introduces several different classification hierarchies for the haematopoietic domain. A cascade of networks instead of a single, flat classifiers allows more diversity, for example by using regression for certain sub-tasks. This possibility is covered in Section 4.5. Different options to employ transfer learning to a cascade of multiple networks are discussed in Section 4.6. In Section 4.7, common convolutional blocks which can be shared between different networks are defined. Section 4.8 briefly covers methods to compensate class imbalance, and how they are used in the context of a cascade of multiple neural networks. Section 4.9 introduces feature forwarding, an approach to connect the otherwise independent networks using skip connections. Finally, in Section 4.10, a macro-averaged hierarchical F-score for model evaluation is defined.

4.1 Data Analysis

In this work, classification is performed on a dataset of haematological images, with the goal to incorporate specific domain knowledge about this dataset into the classification process. A brief overview over the entire dataset, its acquisition and basic properties is given in Section 4.1.1. Relevant implications for this work in particular, including the exclusion of certain cell types and the resulting class sizes, are discussed in 4.1.2.

4.1.1 Dataset

At the time this work is carried out, the dataset contains eight (partly) annotated bone marrow smears, acquired with a magnification of 63 with immersion oil. Figure 4.1 shows an example of such a whole slide image. Samples are acquired, pre-processed and digitised by the Department of Haematology, Oncology, Haemostaseology and Stem Cell Transplantation at the University Hospital of the RWTH Aachen University. The samples are Pappenheim stained, allowing a good distinction between cell nuclei and cytoplasm. For classification, image patches of adjustable size can be extracted from the whole slide images. Individual patches

Figure 4.1: Whole slide image of a bone marrow smear.

are centred around the cell to be classified, but frequently also contain neighbouring cells. In some cases on the other hand, depending on the patch size, the patch might not contain the entire cell. Practically this is mainly the case for cells of megakaryopoiesis due to the very large cell sizes. The slices have been annotated by medical experts, however, to obtain a reasonably large dataset given the limited resources, an inter-rater study to validate those annotations has not been performed yet. The labels are given in a hierarchy motivated by the actual cell lineages (compare Figure 2.1), however with some practical modifications. These modifications are done in medical practice to simplify the annotation process. For example, for eosinophilic and basophilic granulocytes the cells are divided only into immature and mature cells rather than the individual maturity stages, because this is often sufficient for diagnosis and because these cells are significantly less frequent compared to neutrophilic granulocytes. In Pappenheim stain, myeloblasts, lymphoblasts and monoblasts cannot be distinguished reliably and are therefore merged into a single class called blasts, while proerythroblasts are more easily distinguishable and therefore labelled as such. Blasts are further separated into physiological ungranulated blasts and granulated blasts, which can occur for example in case of leukaemia. In Table 4.1 the number of cells of each type is listed.

4.1.2 Data Selection

As indicated in Table 4.1, the dataset is highly unbalanced, with some classes having only very few samples. For experiments, such low numbers of samples are not suitable, therefore, some classes have to be merged or excluded. These considerations are important already for method design because the hierarchical classification naturally is very much dependant on the classes that are present within the class taxonomy. The goal of these adjustments is not to balance the dataset, but to ensure

Cell type	Number of samples
Myeloblasts	130
Ungranulated (physiological) myeloblasts	76
Granulated myeloblasts	54
Neutrophil granulocytes	4859
Promyelocytes	1171
Myelolocytes	486
Metamyelocytes	703
Band forms	1104
Segmented	1355
Pseudo-Pelger	40
Eosinophilic granulocytes	351
Immature eosinophils	196
Mature eosinophils	155
Basophilic granulocytes	82
Immature basophils	16
Mature basophils	63
Atypical basophils	3
Erythropoietic cells	1099
Proerythroblasts	44
Basophilic erythroblasts	163
Polychromatic erythroblasts	646
Orthochromatic erythroblasts	244
Dysplastic erythroblasts	2
Lymphopoietic cells	283
Lymphocytes	243
Plasma cells	40
Monopoietic cells	298
Promonocytes	54
Monocytes	244
Megakaryopoietic cells	19
Megakaryoblast	9
Megakaryocyte	8
Dysplastic megakaryocyte	2
Reticulum cells	5
Mast cells	5
Core macrophage	14
Osteoblasts	1
Artefacts	2747
Total	9893

Table 4.1: Number of cells from each class in our dataset.

that the classes all have a reasonable size, while not excluding too many cell types.

Firstly, some classes are neglected completely, excluding the corresponding samples from training, validation and testing. This is the case for all classes – except cells of megakaryopoiesis – with less than 20 samples, for example reticulum cells, mast cells, core macrophages and osteoblasts. Additionally, some atypical cell types such as pseudo-Pelger cells and other dysplastic cells are not included in any experiments or incorporated into the models themselves.

There are different, heterogeneous types of artefacts such as nucleus and cell

shadows, damaged cells and cell-like artefacts. Although having many samples, these artefacts are neither considered in the developed methods, nor used in the experiments.

Another possibility to avoid too small classes, which retains the samples instead of neglecting them, is to merge related classes into a single class. This is possible due to the hierarchical structure of the class labels. The first classes that are merged in the following are immature and mature basophilic granulocytes, resulting in the class of simply basophilic granulocytes. Similarly, physiological undifferentiated blasts and granulated blasts are merged into a single class. Megakaryopoietic cells are also merged and not dropped, even though the resulting class is still very small. The reasoning behind this is that due to their size, these cells are very unique, and dropping this entire group entirely would neglect one important lineage of haematopoiesis.

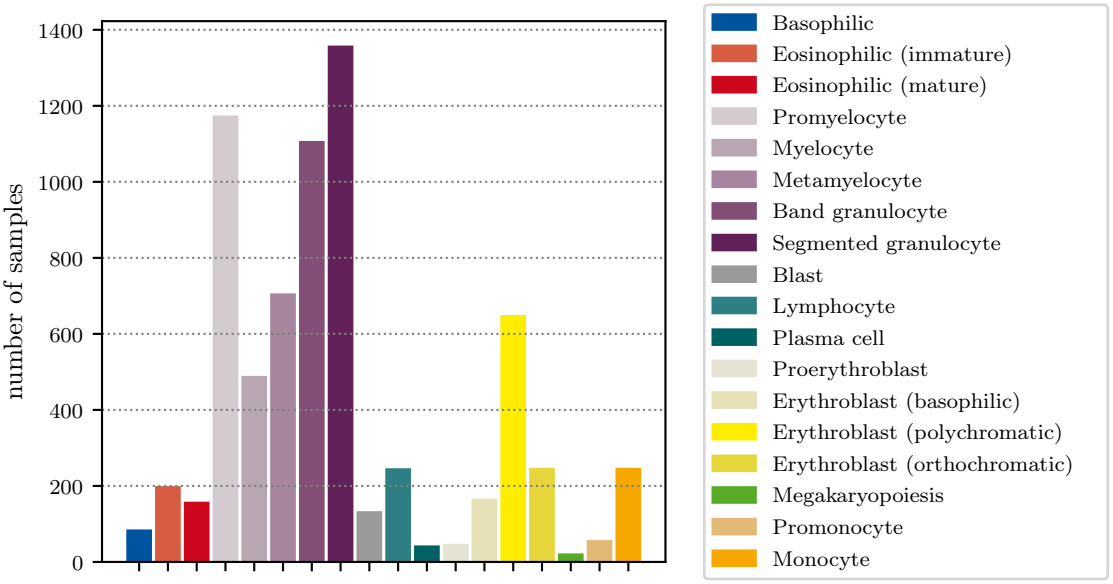


Figure 4.2: Class frequencies of the final classes after merging and dropping of some classes. See Table 4.1 for the exact numbers.

As shown in Figure 4.2, the remaining dataset remains highly unbalanced. For example, neutrophilic granulocytes of all stages are much more frequent than most other cell types. Other classes are still very infrequent due to the compromises that had to be made between getting a reasonable size on the one hand and not excluding too many classes on the other. These imbalances are later adressed by other methods, which are discussed in other parts of this chapter.

4.2 A Cascade of Deep Classifiers

In this work, the domain knowledge is incorporated into the classification in the form of hierarchical cascades. The motivation for this approach is not only a potential

improvement of the classification performance. A hierarchical classification also provides more insight into the classification process and is more informative, as provides additional output at all intermediate levels of the underlying taxonomy. Furthermore, it makes the whole classification more adaptable, as individual parts of the cascades can be altered without affecting the rest of the hierarchy.

4.2.1 Basic Principle

Given a hierarchy tree, deep neural networks can be placed at every parent node that has multiple children. All theses networks receive the image $\boldsymbol{I}$ as their input and perform a prediction to which child class the sample belongs. The example in Figure 4.3 shows a simple cascade for granulopoiesis, where a first prediction aims to identify the lineage, before a specialised second network determines only the exact cell type within its particular lineage. Mostly, these networks are classification networks – another possibility in form of regression networks is introduced later in Section 4.5.

A classification network at parent c_i provides output confidences for each child class by applying the softmax function $S(\boldsymbol{y_i})$ to the final fully connected layer's output vector $\boldsymbol{y_i}$. As part of this work, several different groupings of the labels specifically for the haematological domain are investigated. These are introduced in Section 4.4, along with their individual motivation. The following sections provide more specific detail on how such a cascade is traversed in general to obtain a final prediction.

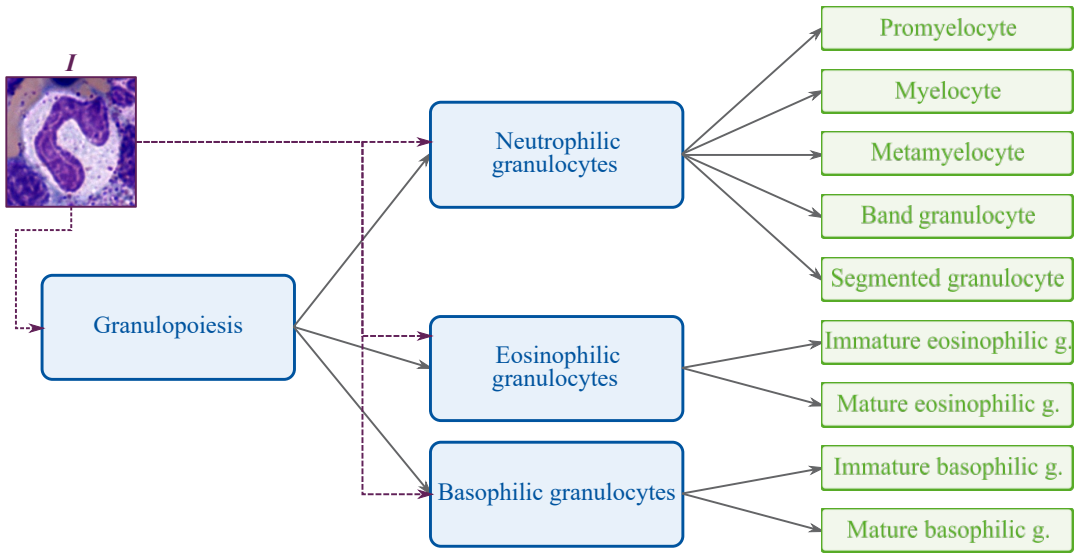


Figure 4.3: Example of a hierarchical cascade of classifiers (blue boxes), derived from the cell hierarchy of granulocytes.

4.2.2 Propagation Through the Cascade

To perform the hierarchical classification and obtain final leaf-class labels from individual predictions, propagation strategies need to be defined. In this work, two approaches are examined – a deterministic approach, where every network provides a definitive decision which branch to follow, and a probabilistic approach, based on the calculation of the a posteriori class confidences.

Deterministic (Hard) Propagation

Each network's output can be used to make a definitive decision by always only following the highest confidence, given by the softmax output $S(\boldsymbol{y_i})$. At every network, a hard classification is performed by discarding non-maximum entries. Consequently, the hierarchy is followed only along the most promising path in a greedy fashion. This effectively represents pruning of the hierarchy, as the input only needs to be forwarded through networks on this one path. Hence, the predicted multi-label C' is extended at each parent along the path. Algorithm 1 describes the procedure in pseudo-code.

Algorithm 1: Hard, deterministic propagation through the hierarchy

Input: An input image $\boldsymbol{I}$, cascade structure T with networks in nodes with at least 2 children
Output: Predicted multi-class label C'

$C' \leftarrow \{c_0\}$; `// initialise predicted multi-label, containing only the root class` c_0
node $\leftarrow$ root(T) ; `// the currently visited node, starting at the root`
while node is not a leaf **do**
  if node has only one child **then**
    node $\leftarrow$ child(node);
    continue
  end
  $\boldsymbol{y}_{\text{node}} \leftarrow$ network output given $\boldsymbol{I}$ at the current node;
  $i = \arg\max S(\boldsymbol{y}_{\text{node}})$; `// greedily follow the highest confidence`
  let c_i be the class of the i-th child of the current node;
  $C' \leftarrow C' \cup \{c_i\}$; `// extend the multi-label with child-class` c_i
  node $\leftarrow$ i-th child of node;
end
return C'

A direct consequence of this *hard propagation* approach is a guaranteed consistency of individual networks' decisions and the finally assigned multi-label. However, it does not take the confidences assigned by the softmax functions into account. Very confident decisions, where the confidence of one class is much higher than of all siblings, have the same impact as very close decisions.

Probabilistic (Soft) Propagation

Alternatively, the cascade can be traversed completely, putting the input through every network, and calculating the a posteriori probabilities of all nodes. Each softmax provides a confidence in the range between 0 and 1 for all children. Summing up to 1, these confidences are – mathematically – probabilities and can be used to calculate an a posteriori confidence value. For this purpose, as a first step the cascade is traversed in an arbitrary order, and the input image $\boldsymbol{I}$ is forwarded through every network without any pruning. The softmax outputs $S(\boldsymbol{y_i})$ of all networks are collected. Afterwards, the cascade is traversed a second time, starting from the root, such that no node is visited before all of its ancestors have been visited. In this step, the a posteriori probabilities of the node classes are calculated from the previously collected confidence values. For the root, the network output is directly used as an estimated probability of the j-th child class

$$P(C_j|\boldsymbol{I}) = S(y_{0,j}) \tag{4.1}$$

to be the correct sub-class at this level in the taxonomy. For all other nodes, the softmax output is used only as an estimate of the probability of the children, assuming that the sample is also member of the parent's multi-class C_i

$$P(C_j|\boldsymbol{I}, C_i) = S(y_{i,j}) \tag{4.2}$$

for $i \neq 0$ and $c_j \in \text{children}(c_i)$. These collected confidences are used to iteratively calculate the a posteriori probabilities

$$P(C_j|\boldsymbol{I}) = P(C_j|\boldsymbol{I}, C_i) \cdot P(C_i|\boldsymbol{I}). \tag{4.3}$$

For prediction of the final leaf class, the multi-class of the leaf with the maximum a posteriori probability

$$C' = \underset{C_i \in \text{leaves}}{\arg\max}\, P(C_i|\boldsymbol{I}) \tag{4.4}$$

is selected. The detailed procedure is summarised in Algorithm 2.

In the following, this probabilistic approach is also called *soft propagation* due to the lack of hard decisions in intermediate levels. Every network only contributes its confidence values, but a hard decision is only performed at the very end of the cascade by comparing the a posteriori probabilities of all leaves. In contrast to hard decision, it is possible that the finally assigned multi-label is inconsistent with individual decisions when performing predictions at a non-leaf level. Such a case is illustrated in Figure 4.4. For soft decision, the leaf with the highest a posteriori confidence (0.405) would be predicted, while for hard decision, a leaf with a slightly lower confidence (0.3465) is selected. This can occur especially in cases of low confidence, as in this example for the network with confidences of 0.45 and 0.55 for its two children. Soft decision has the advantage of using the confidence values provided by the networks, making high confidences more important and providing additional information about the output. An increasing depth of the

Algorithm 2: Soft, probabilistic propagation through the hierarchy

Input: An input image $\boldsymbol{I}$, cascade structure T with networks in nodes with at least 2 children
Output: Predicted multi-class label C'
foreach network i in T **do** `// in an arbitrary order`
  forward $\boldsymbol{I}$ through network i and store softmax output $S(\boldsymbol{y}_i)$;
  foreach child j of node i **do**
    assign $P(C_j|\boldsymbol{I}, C_i) = S(y_{i,j})$
  end
end
`// calculate a posteriori confidences:`
foreach child j of the root node 0 **do**
  assign $P(C_j|\boldsymbol{I}) = S(y_{0,j})$;
end
foreach non-root node i with at least 2 children **do** `// from root to leaves`
  foreach child node j **do**
    $P(C_j|\boldsymbol{I}) = P(C_j|\boldsymbol{I}, C_i) \cdot P(C_i|\boldsymbol{I})$
  end
end
`// perform prediction:`
return $C' = \arg\max_{C_i \in \text{leaves}(T)} (P(C_i|\boldsymbol{I}))$

hierarchy poses an additional challenge for the probabilistic approach, as the number of competing nodes increases. The repeated multiplications and the possibility that multiple a posteriori confidences in different branches can become relatively close might potentially cause numerical instability. In practice, deep neural networks tend to be overconfident [54], therefore it can be expected that the outcome will be the same in most cases.

4.3 Cascade Training

To train a hierarchy of networks, an appropriate method must be defined. In general, it is possible to train individual networks separately (Section 4.3.1) or all at once (Section 4.3.2) under different premises. Both methods are motivated and formally defined in the following.

4.3.1 Separate Training of the Individual Networks

The individual networks in the cascade can be trained separately and in an arbitrary order. This procedure represents a local *per parent* approach to hierarchical classification. The individual networks are trained, validated and tested only with samples that are a member of the parent node's class, and consequently belong to

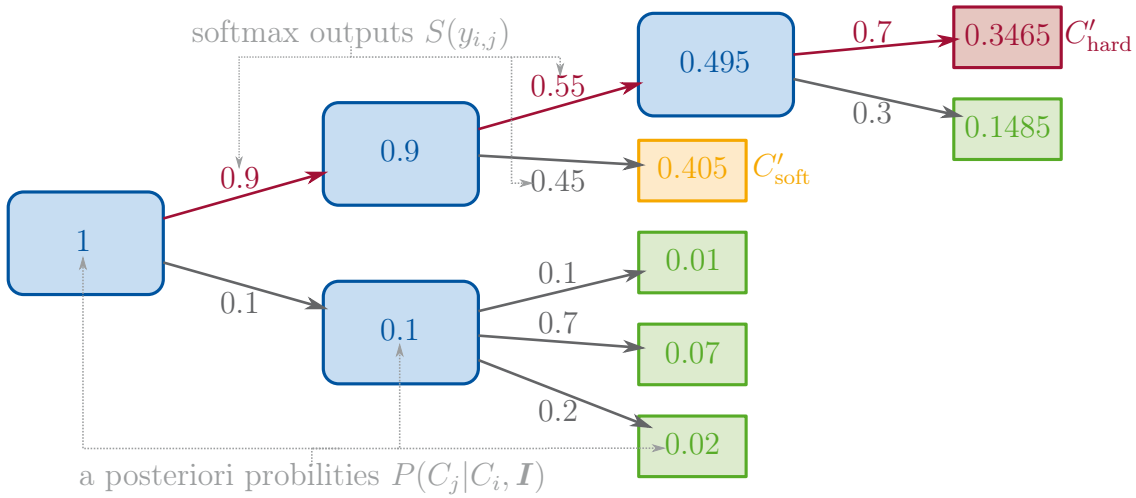


Figure 4.4: Comparison of hard and soft propagation. In this example, the resulting prediction is not the same. The highlighted edges indicate the path followed during hard propagation. All a posteriori probabilities are only calculated during soft propagation.

one of its descendant leave classes.

This assures validity of (4.2) – the task of prediction whether a given sample belongs to a parent multi-class C_i is left solely to the previous network, while it is presumed by the current network. Consequently, the network's output can be, when using soft, probabilistic propagation through the hierarchy, interpreted as an estimate of the conditional probability $P(C_j|\boldsymbol{I}, C_i)$. The resulting training procedure, given in Algorithm 3, is relatively simple. It allows for a modular approach, with individually adjustable training parameters $\boldsymbol{\theta}_i$ allowing for example different amount of training depending on the individual performance, or different loss functions for different networks. Furthermore, separate training makes it possible to employ model selection for each individual network based on its validation score, in contrast to selecting the optimal overall model. Also, since the propagation is not relevant for training single networks, it is possible to switch between the different propagation strategies without requiring any other adjustments or retraining. The networks can be assembled into the cascade after training, during training they are fully independent, as the cascade is never traversed in its entirety.

4.3.2 End-to-End Training of the Cascade as a Whole

When performing the soft, probabilistic propagation strategy, it is possible to train the entire cascade end-to-end (Algorithm 4). Looking only at the leaves, the a posteriori probabilities of the leaves sum up to 1

$$\sum_{i=1}^{n} P(C_i|\boldsymbol{I}) = 1, \tag{4.5}$$

Algorithm 3: separate training of individual networks

```
Input: Set of training parameters θ, classification hierarchy T
forall nodes i in T with at least 2 children do  // in an arbitrary order
    initialise a local network at node i;
    foreach epoch do                    // number of epochs specified in θ_i
        for batches of data from the subtree rooted in node i do
            train local network with training parameters θ_i;
             // backpropagation of loss, weight update...
        end
        determine current validation performance;
    end
    select model state with best local validation score;
end
// end of training:  Only now the cascade needs to be composed
select a propagation strategy;      // can arbitrarily be changed later
 without a need to retrain the networks
```

as they do in flat classifiers. Consequently, the whole network can be seen as a single black box network, which can be trained using a single loss. The multiplication of the individual network outputs in soft propagation leads to well-defined gradients. This end-to-end training can be seen as a global approach to hierarchical classification, at least during the training phase. The structure of the cascade directly incorporates the hierarchy, but is trained as a whole.

Hierarchical Tree Loss

It is possible to train the whole cascade using any appropriate loss, for example the cross-entropy loss (2.9). Such standard losses however do not account for the given taxonomy. Therefore, to extend the global hierarchical classification approach, it might be desirable to incorporate the class relationships into the loss. The hierarchical loss defined by Wu et al. [30] (see Section 3.2.3) is applicable only when the hierarchical classification is performed on ultrametric trees. A good hierarchical loss should penalise more distant misclassifications more heavily than close ones. Then, applied to a cascade of classifiers which is trained end-to-end, a high loss means that the error is located also at earlier levels in the taxonomy. Since potentially more networks are involved in causing this misclassification, a larger training step might be beneficial. A low, non-zero loss on the other hand should indicate a misclassification close to the leaves.

With these considerations, another hierarchical loss is defined. It is based on the cross-entropy loss, but weighs the individual loss components differently

$$\mathcal{L}_{\text{tree}} = -\frac{1}{N}\sum_{i=1}^{N} w(\boldsymbol{t_i}, \boldsymbol{y_i}) \cdot \sum_{c=1}^{n} t_{i,c} \cdot \log(y_{i,c}). \tag{4.6}$$

Algorithm 4: End-to-end training of the entire cascade

Input: Training parameters $\boldsymbol{\theta}$, classification hierarchy T

```
// the cascade needs to be composed before the actual training
   begins:
```

initialise individual networks in all nodes of T with at least 2 children;

foreach epoch **do** `// number of epochs specified in` $\boldsymbol{\theta}$

  foreach batch in the training set **do**

    forward images in batch through entire cascade using soft propagation;

    calculate and backpropagate loss through all networks in T;

    update weights of all networks at once;

  end

  evaluate performance of whole cascade on validation set;

end

select best performing model state based on validation score;

Here, the weight function w is defined as

$$w(\boldsymbol{t_i}, \boldsymbol{y_i}) = \sum_{j=1}^{n} y_{i,j} \cdot \operatorname{dist}\left(j, \arg\max(\boldsymbol{t_i})\right), \tag{4.7}$$

where

$$\operatorname{dist}\left(j, \arg\max(\boldsymbol{t_i})\right) = |C_j \setminus C_{\arg\max(\boldsymbol{t_i})}| + |C_{\arg\max(\boldsymbol{t_i})} \setminus C_j| \tag{4.8}$$

is the distance between the j-th leaf and the correct leaf, defined as the number of mismatches between the multi-labels of these leaves. Relaxing the directionality of edges within the tree, this is the same as the length of the path between the true leaf and the current leaf.

4.4 Cascade Structures

Part of this work is the inspection of different hierarchical structures to guide the classification based on the haematological dataset as introduced in Section 4.1. Several different possibilities are introduced, each in form of a tree with networks placed at each node with two or more children. These different structures are described in this section. As mentioned in Section 4.1, artefacts are not considered by these hierarchies, but could always be included by adding a new root to the hierarchy, with two children corresponding to either artefacts or blood cells in general.

4.4.1 Full Hierarchy Directly Based on Cell Hierarchy

A first possibility to define a hierarchy to guide classification is simply orienting on the developmental cell hierarchy. In many cases, cells of the same haematopoietic lineages are more similar to each other than to cells of other lineages. Following

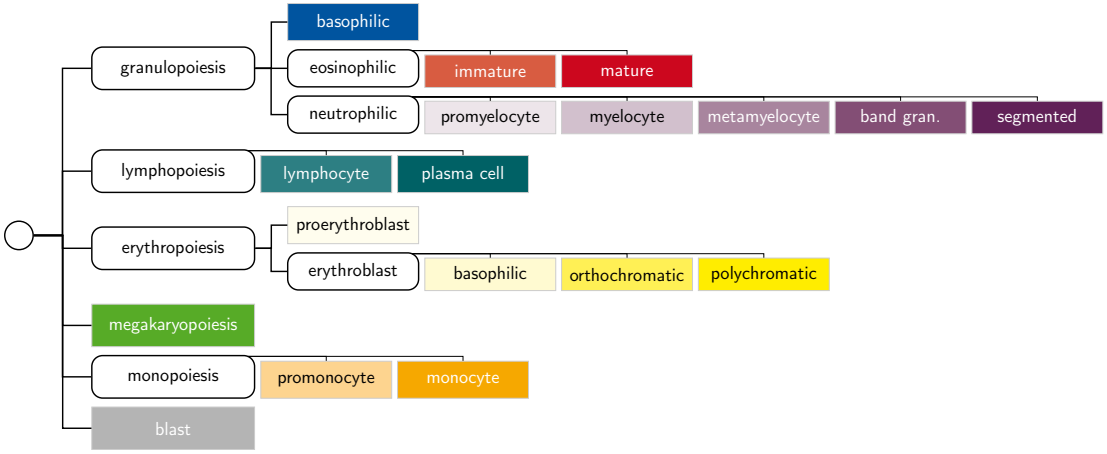


Figure 4.5: Full hierarchy classification tree. The circle indicates the root, rectangles with rounded edges represent networks.

the cell hierarchy for classification incorporates these lineage-dependant characteristics. Furthermore, every classification within the cascade represents a biologically well-founded distinction. The hierarchical structure, which is shown in Figure 4.5, contains several intermediate steps, as the classification is becoming more and more specific. First, more general lineages are separated, for example for haematopoietic cells whether they belong to granulopoiesis, erythropoiesis, lymphopoiesis et cetera. More specialised networks then distinguish different maturation stages within a single lineage.

4.4.2 Simplified, Ultrametric Hierarchy

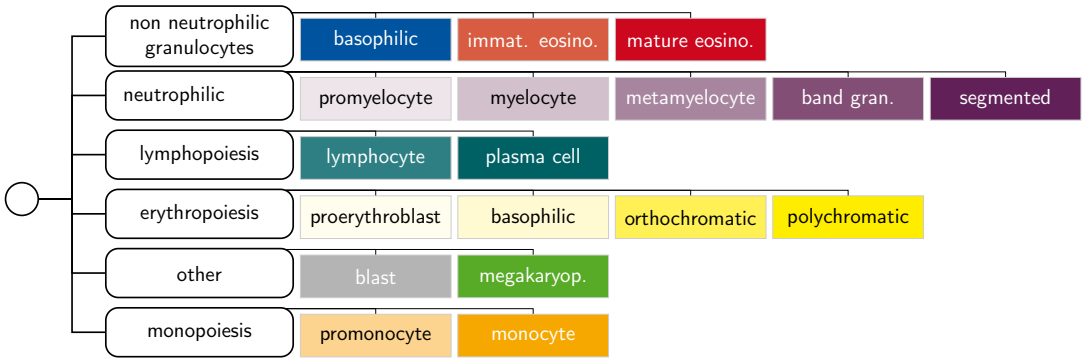


Figure 4.6: Simplified, ultrametric hierarchy classification tree.

The full hierarchy contains many networks, often with only relatively few classes. Additionally, it is not ultrametric, meaning that different leaf classes can have a different number of preceding networks. By removing some internal nodes along with their networks, the hierarchy can be simplified, while still roughly representing the haematopoietic cell hierarchy. With some re-grouping such a simplified taxonomy can be made ultrametric. The resulting tree is shown in Figure 4.6. Compared

to the full hierarchy, blood cells are directly divided into the groups of neutrophilic granulocytes, non-neutrophilic granulocytes, lymphopoiesis, erythropoiesis, monopoiesis, and others. Therefore, each path from the root to any leaf passes two networks, one to classify the group, and one to specify the class within these groups. As a side note, other than in the other hierarchies, artefacts should not be inserted with a new root node, but for example as a child of the *other* node to keep the tree ultrametric.

4.4.3 Feature-Based Cell Grouping

Another possibility to group the cell classes is to solely focus on certain features at certain levels in the cascade. This approach does not directly take cell-type relationships into account but is based only on visual characteristics. On the contrary, completely new multi-classes are introduced, each representing certain feature characteristics. A drawback of such an artificial taxonomy is that it is difficult to find an appropriate structure which is significantly different from the hierarchy based on the cell development, because many characteristics are shared between cells from the same lineages. Furthermore, features may vary even for cells of the same class and it can be difficult to assign a cell type to a fixed feature group. In the following, three such feature-based groupings are detailed.

Hierarchy based on Cytoplasm Colour and Granulation

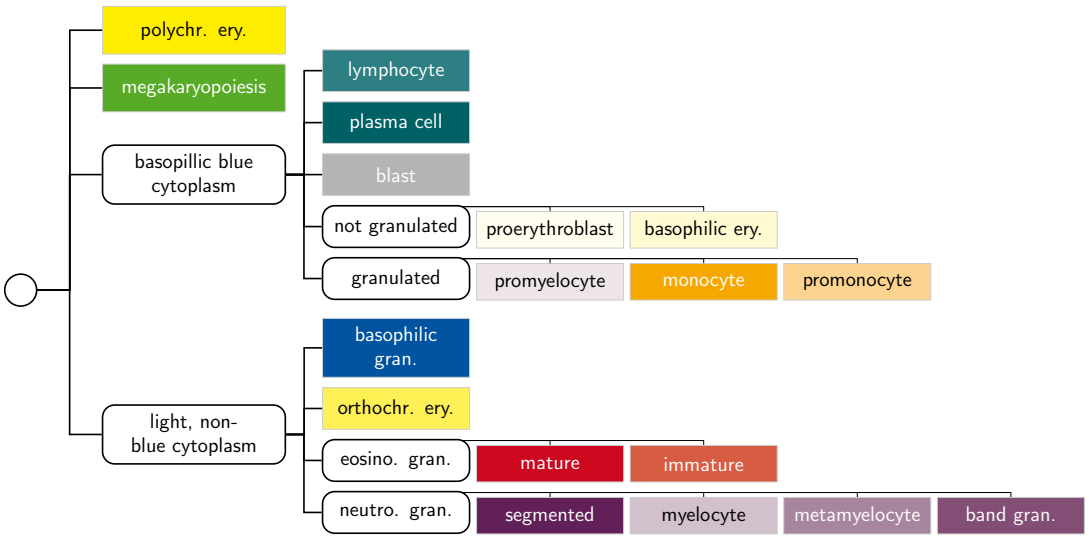


Figure 4.7: Feature-based classification tree based on cytoplasm colour and granulation.

A first hierarchy separates cells based on the underlying cytoplasm colour in the first step, and based on granulation in a second step. It is visualised in Figure 4.6. Polychromatic erythroblast have a diverse cytoplasm colour and cannot be assigned to any of the classes, while cells of megakaryopoiesis are often too large for image

patches of 224×224 pixels. For these reasons, these two classes are not assigned to any of the groups, but inserted as leaves already at the first level of the tree.

The first group of cells is characterised by a relatively light cytoplasm colour. This includes all more mature stages of granulopoiesis starting from myelocytes and orthochromatic erythroblasts. Immature basophilic and eosinophilic granulocytes are also accounted to this group; however, this reveals a problem of the feature-based approach. The cell types that are summarised as immature contain also promyelocytes which for neutrophilic cells are accounted to the other group. However, both classes are relatively small, and since the promyelocytes stand opposed to myelocytes, metamyelocytes and band forms, are assigned to this group. Especially for basophilic granulocytes, an additional difficulty for classification into this group is the fact that the strong granulation often obscures the actual cytoplasm colour. Within the cells with a light, non-basophilic cytoplasm, there is a further division based on the granulation. The resulting four groups are basophilic, eosinophilic and neutrophilic cells, and the non-granulated cells, only containing orthochromatic erythroblasts.

The second group contains cells with a more or less basophilic cytoplasm. Within this group, lymphocytes and plasma cells are directly treated as leaves. The other cells are grouped into granulated and non-granulated cells. Granulated cells include both monocytes and promonocytes as well as promyelocytes, while the group of ungranulated cells includes proerythroblasts and basophilic erythroblasts. Blasts include both physiological ungranulated blasts and granulated blasts, and therefore are not assigned to any of these two groups, as well as lymphocytes and plasma cells. Further separations are not performed because of the relatively small sizes of the resulting groups.

Hierarchy based on Nucleus-to-Cytoplasm-Ratio and Nucleus Shape

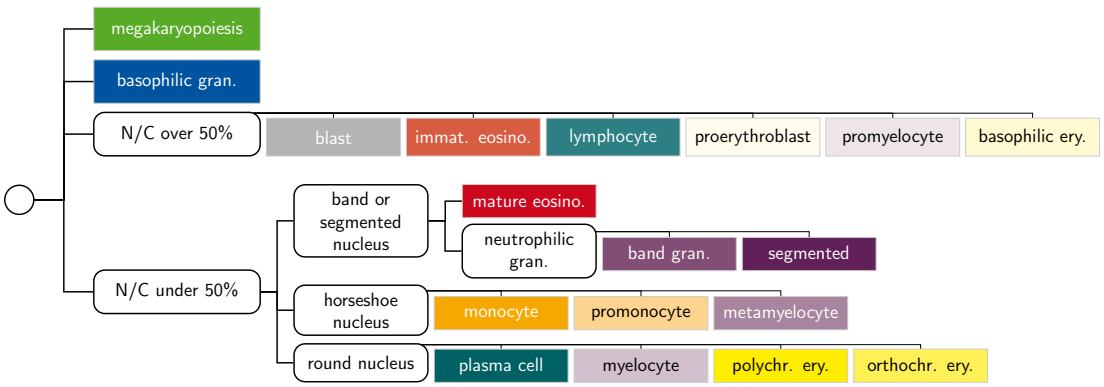


Figure 4.8: Feature-based classification tree based on the nucleus-to-cytoplasm ratio (N/C) and nucleus shape.

In the second feature-based hierarchy, shown in Figure 4.8, cells are first divided into large and small nucleus-to-cytoplasm ratios and then grouped by nucleus shape. Again, cells of megakaryopoiesis are not assigned to any of these groups, because they

are usually exceeding the image patches. Similarly, the merged basophilic granulocytes are also inserted here, because the group membership is unclear – depending on the maturity of the specific sample. The threshold for the two groups was selected to be 50 %, based on literature [7] values. Large nucleus-to-cytoplasm ratios are a common characteristic of many immature stages such as immature basophilic and eosinophilic granulocytes, myeloblasts and promyelocytes and proerythroblasts as well as basophilic erythroblasts. Additionally to these blast stages, the group of cell types with a relatively large nucleus also contains the more mature lymphocytes. Since all these cells have round or oval nuclei, there is no further division within this group.

The second group of cells consists of more mature stages, and is sub-divided based on the nucleus shape. The first sub-group is comprised of cells with a band-form or segmented nucleus, and itself is divided into neutrophilic, basophilic and eosinophilic cells. Again, basophilic granulocytes are a special challenge, since the nucleus is often obscured by the basophilic granules, so neither the nucleus-to-cytoplasm ratio nor the segmented character of the nucleus are visible. The second sub-group are cells with a horseshoe shaped nucleus, which are metamyelocytes and both stages of monopoiesis. The third and final sub-group are cells with a round or an oval nucleus, which are orthochromatic and polychromatic erythroblasts, myelocytes and plasma cells.

Hierarchy Based on Nucleus Shape and Visibility

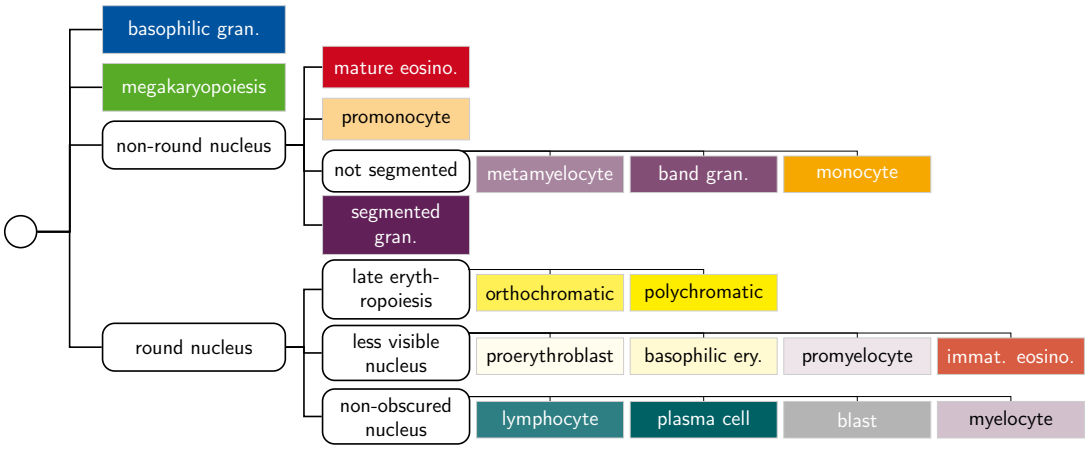


Figure 4.9: Feature-based hierarchy classification tree based on nucleus shape and visibility.

Lastly, the third feature-based hierarchy is again based on nucleus shape in a first step, and then the more subjective nucleus visibility and overall granulation strength, as shown in Figure 4.7. At the first stage, cell classes are separated based on whether their nucleus is round (or oval) or indented. Basophilic granulocytes and cells of megakaryopoiesis are treated as separate leaves at this stage already, because these features are not necessarily visible or definitively assignable to one of these two groups.

Within the group of cells with a round nucleus, the first sub-group is late erythropoiesis, containing polychromatic and orthochromatic erythroblast, which are both defined by a relatively small, condensed nucleus. Cells in the second group often have a rather low contrast between nucleus and cytoplasm, for example due to granulation or a strongly basophilic cytoplasm. Classes belonging to this group are proerythroblasts, basophilic erythroblasts, promyelocytes and immature eosinophilic granulocytes. All other cells with round or oval nucleus are constituting the third sub-group. These cell types are myeloblasts, myelocytes, lymphocytes and plasma cells.

Cells with an indented nucleus are further separated depending on whether they have a segmented or a non-segmented nucleus. Promonocytes and mature eosinophilic granulocytes are not assigned to any of these classes. For promonocytes, a correct assignment to these binary groups is difficult, while eosinophilic granulocytes often only show two, large segments, and furthermore, are well characterised by their eosinophilic granulation, justifying their treatment as a separate class. The group of cells with a segmented nucleus therefore only consists of a single class – segmented neutrophilic granulocytes – while the non-segmented cells are metamyelocytes, band granulocytes and monocytes.

4.4.4 Using Spectral Clustering to Build a Hierarchy

Previous hierarchies are either oriented on the actual cell hierarchy or have manually been designed based on certain features. An alternative idea is to obtain a classification tree not by manual design, but in an automated fashion by performing clustering, based on an appropriate similarity measure.

Clustering of the Confusion Matrix

In many cases, a classifier confuses some classes more than others. Often, but not exclusively, such misclassifications occur between more closely related classes. It is possible to divide a classification process hierarchically by grouping more easily confusable classes together and then training a specialised network that solely aims to distinguish these classes. An appropriate measure for the misclassifications of a classifier is given by its confusion matrix. Figure 4.10 shows the combined confusion matrix obtained on the test data from all cross-validation runs of an experiment using a flat DenseNet-121 (see 5.1 for the general experimental setup). A confusion matrix $\boldsymbol{M} \in \mathbb{N}_0^{n \times n}$ can be interpreted as an asymmetrical similarity matrix between the classes. However, spectral clustering (see Section 3.4.1) requires a symmetrical similarity matrix $\boldsymbol{S}$. Such a symmetrical matrix can be obtained by

$$\boldsymbol{S} = \boldsymbol{M} + \boldsymbol{M}^T \in \mathbb{N}_0^{n \times n}. \tag{4.9}$$

In principle, $\boldsymbol{S}$ can be used for spectral clustering, but it is affected by class imbalances, since it contains the absolute number of confusions rather than the relative frequencies. In extreme cases, a large class may be misclassified as another class only relatively few times, but compared to a much smaller class, the absolute number of

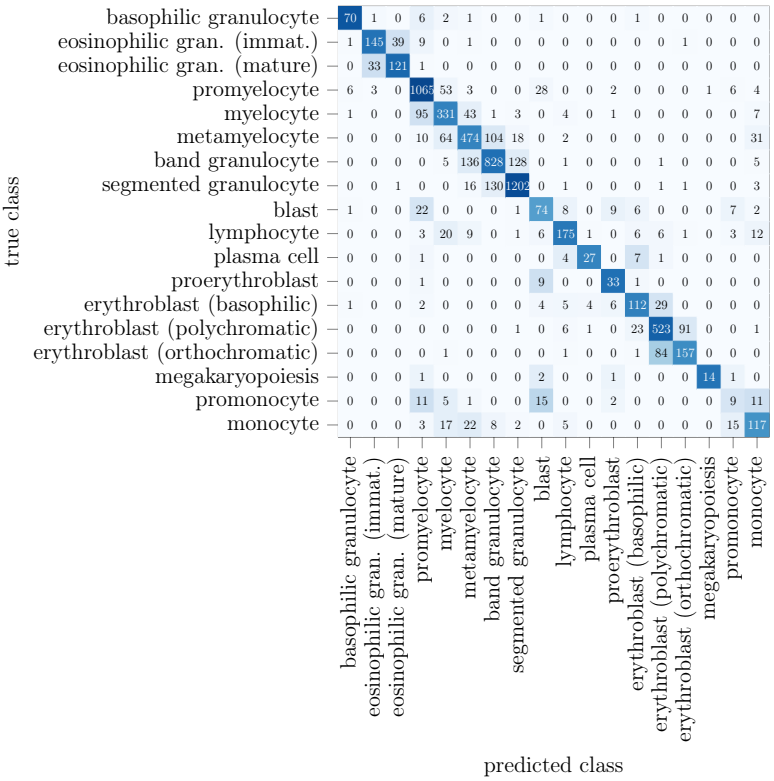


Figure 4.10: The confusion matrix of a flat DenseNet-121 used for spectral clustering.

these misclassifications may still be large. Therefore, either the rows or columns of $\boldsymbol{S}$ can be normalised, unless a whole row or column is zero. After normalising the rows, $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ describes the relative frequencies of the predicted classes given the actual class. Vice-versa, normalising the columns yields a matrix $\boldsymbol{S} \in \mathbb{R}^{n \times n}$ representing the frequencies of the actual classes given the predicted class label.

Regardless if and how the similarity matrix is normalised, spectral clustering can be performed. For the number of clusters, the eigengap heuristic is used. The confusion matrix shown in Figure 4.10 is used to derive a similarity matrix. It should be noted that, in general, it is problematic to design a method based on the test set on which the resulting performances are compared. In general, it would be preferable to use a separate, disjoint subset to obtain the confusion. However, due to the already limited size of the dataset, the combined test results are used here. The upper plot in Figure 4.11 displays the absolute values of the eigenvalues of the Laplacian matrix $\boldsymbol{L}$ for the case of row normalisation. The largest gap between subsequent eigenvalues occurs from $k = 4$ to $k = 5$. The eigenvalues λ_1 up to λ_4 are relatively close to zero compared to all following eigenvalues. Therefore, the number of clusters made in the first stage is selected to be 4, which results in the tree shown in Figure 4.12.

The first cluster contains plasma cells and basophilic, polychromatic and orthochromatic erythroblasts. In the second cluster, metamyelocytes, band and segmented granulocytes are combined, while the third cluster only contains mature and

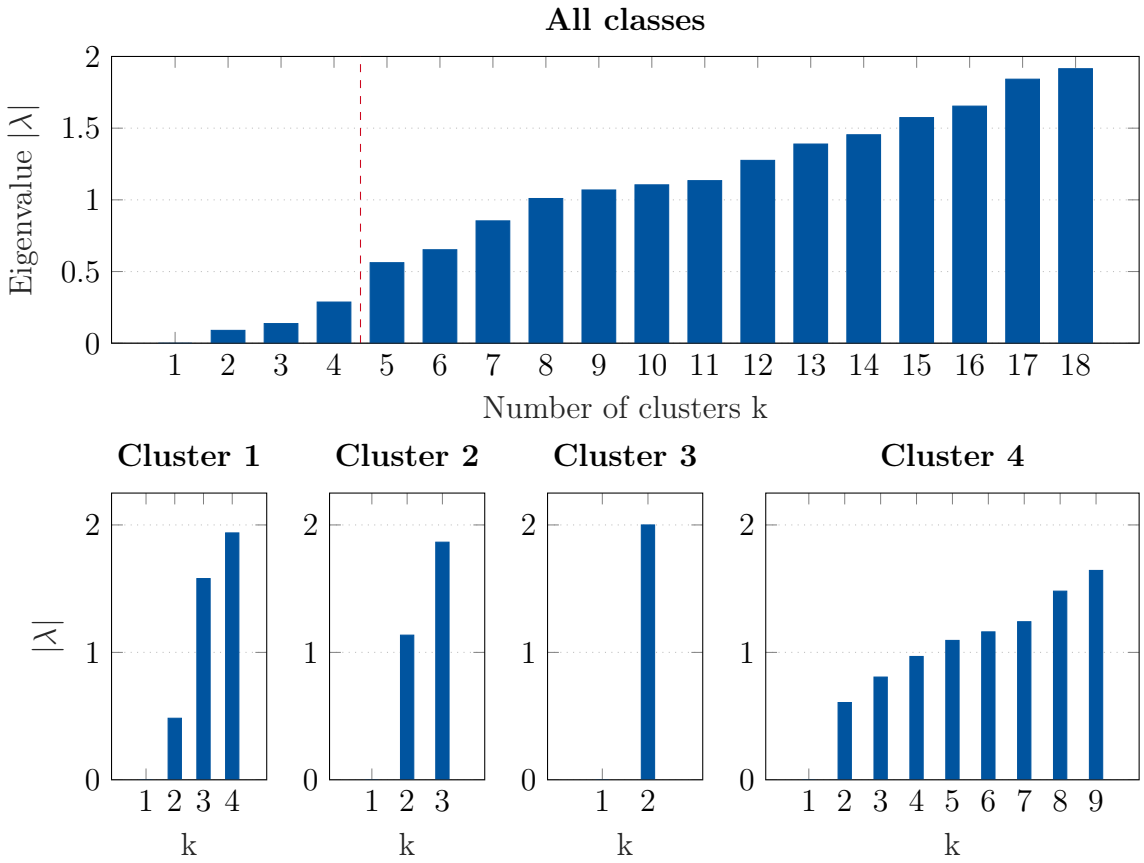


Figure 4.11: Eigenvalues of the Laplacian matrix for spectral clustering based on the confusion matrix of a flat network. The plots in the lower row show the eigenvalues within the four resulting clusters.

immature eosinophilic granulocytes. The fourth and final cluster contains all other cell types. Analysis of these clusters based on the biological background shows that they tend to contain relatively similar cells. For example, the second cluster only consists of relatively mature stages of neutrophilic granulocytes. The first cluster contains three types of erythroblasts. Polychromatic and orthochromatic erythroblasts both are characterised by a relatively condensed nucleus, conversely, plasma cells also have a relatively small nucleus compared to their size. However, in other cases, no particular characteristic is obvious. From this clustering, it is theoretically possible to further divide the resulting clusters. However, it showed that within these clusters, the eigengap occurs already between $k = 1$ and $k = 2$, as shown in the lower plots in Figure 4.11. Consequently, these clusters are not further divided. As a side node, all three possibilities of defining $\boldsymbol{S}$ result in the same final clusters here. Even though the eigenvalues differ slightly, the overall pattern is similar.

Clustering of a Similarity Matrix as Defined by Medical Experts

Medical experts from the Department of Haematology, Oncology, Haemostaseology and Stem Cell Transplantation at the University Hospital of the RWTH Aachen

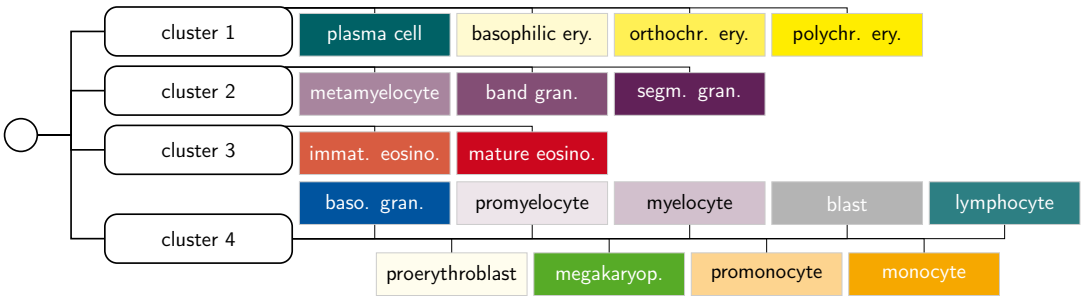


Figure 4.12: Classification tree from spectral clustering of a confusion matrix.

University have rated the similarity between 15 cell types. The doctors were asked to rate the confusability of a given cell type with the other types on a scale from 1 (not confusable) to 5 (not distinguishable), only assuming non-pathological cell forms. For the resulting similarity matrix $\hat{\boldsymbol{S}}$ (see Appendix A.2) to be a valid similarity matrix for spectral clustering, is again made symmetrical by adding its transposed matrix

$$\boldsymbol{S} = \left(\hat{\boldsymbol{S}} + \hat{\boldsymbol{S}}^T\right) \quad \in \mathbb{N}_0^{n \times n}. \tag{4.10}$$

By additionally subtracting 2 from $\boldsymbol{S}$, all matrix elements corresponding to the lowest similarity value (1 in the survey, 2 after adding the transposed matrix) are set to zero. This effectively deletes many edges in the similarity graph and assures that $\boldsymbol{S}$ models only the local neighbourhood of the classes, as required for spectral clustering [40].

On this similarity matrix, spectral clustering is performed. The eigenvalues of the resulting Laplacian matrix are shown in the upper plot in Figure 4.13. Although it is not very clear in this case, a relatively large gap occurs between $k = 5$ and $k = 6$, so following the eigengap heuristic, 5 clusters are created.

The classification tree is shown in Figure 4.14. A first cluster contains basophilic and eosinophilic granulocytes together with neutrophilic promyelocytes and myelocytes. The second group consists of the more mature neutrophilic granulocytes, namely metamyelocytes, band and segmented granulocytes. Cells of erythropoiesis make two clusters, one containing the more mature polychromatic and orthochromatic erythroblasts, and the other containing proerythroblasts and basophilic erythroblasts. The final cluster contains the other cells covered by the survey, which are blasts, lymphocytes, promonocytes and monocytes. Again, these clusters are not further divided, because the resulting clusters would be very small and because of the eigenvalues: As shown in the lower plots in Figure 4.13, all eigenvalues of the Laplacian matrices from these clusters for $k > 1$ are larger than 0.5.

Cell types that were not included in the survey are placed at appropriate positions based on the biological background. Eosinophilic granulocytes are simply split into mature and immature cells, but kept at the same position. Cells of megakaryopoiesis are inserted as a separate group of their own, because of much larger different size. Plasma cells are inserted as a sibling of their closest relative, which are lymphocytes.

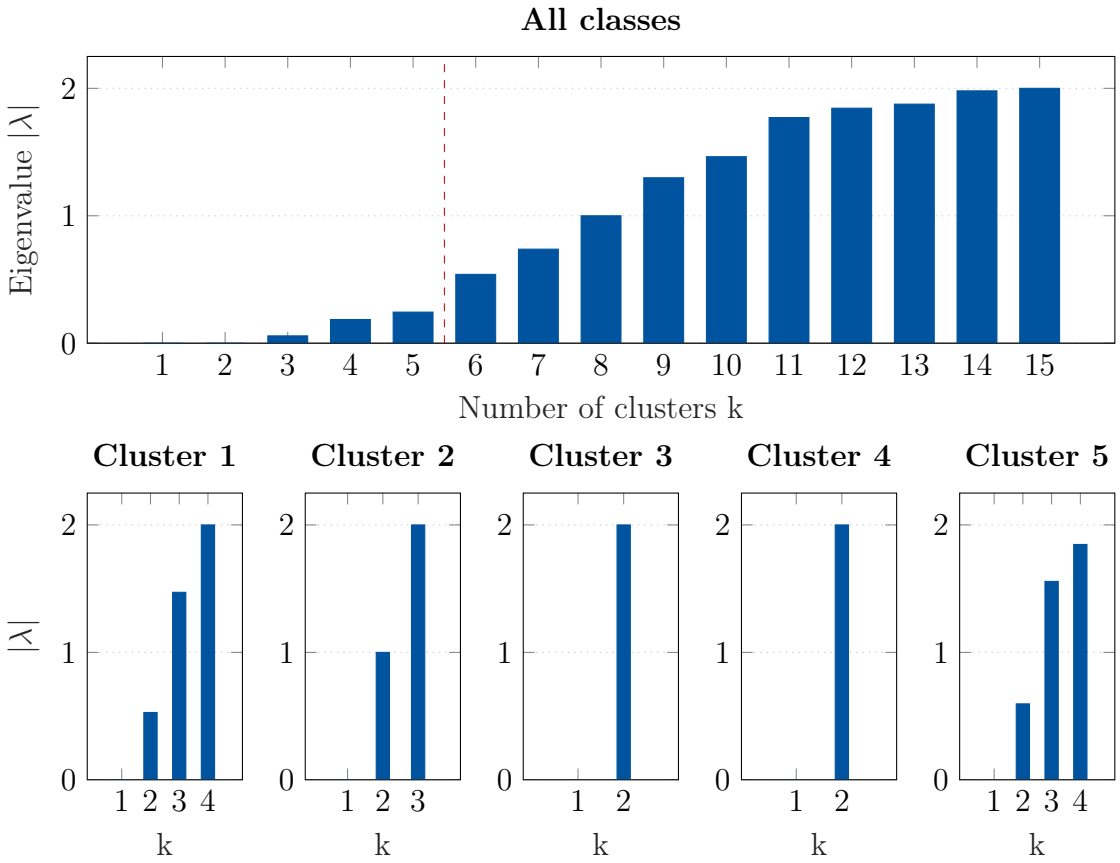


Figure 4.13: Eigenvalues of the Laplacian matrix for spectral clustering, based on cell type similarity rated by medical experts. The plots in the lower row show the eigenvalues within the five resulting clusters.

4.4.5 Hierarchy Reducing Class Imbalances

A large practical problem in training deep classifiers is class imbalance. In a hierarchy, class imbalances can in many cases be less extreme compared to a flat, single network. The reason is the combination of several classes into intermediate classes in early levels, and the exclusion of many classes at later levels in specific branches of the hierarchy. However, at least in early levels, the imbalances may even be amplified if – coincidentally – large classes are in the same sub-class. This is, for example, the case for neutrophilic granulocytes, which are all relatively frequent in the dataset (see Figure 4.2) and can, when combined, outnumber any other class at an intermediate level in the hierarchy. Based on these considerations, it is possible to design a hierarchy specifically reducing class imbalance at all levels. Such a hierarchy in general does not directly incorporate the biological background, on the contrary, in many cases it might contradict the domain knowledge by separating similar classes for the sake of achieving more balanced class sizes.

A simple hierarchy with only three networks that achieves more balanced classes is visualised in Figure 4.15. Cells are grouped into frequent cells with at least 350 samples, infrequent classes with between 80 and 350 samples, and rare classes for

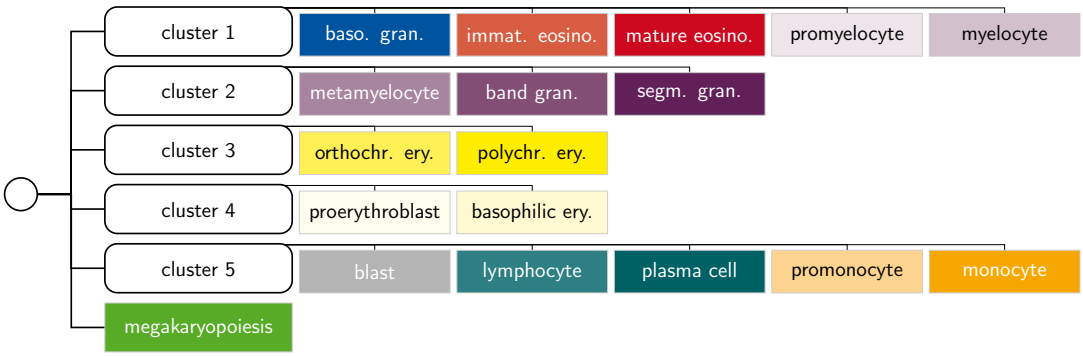


Figure 4.14: Classification tree from spectral clustering of similarities as rated by medical experts.

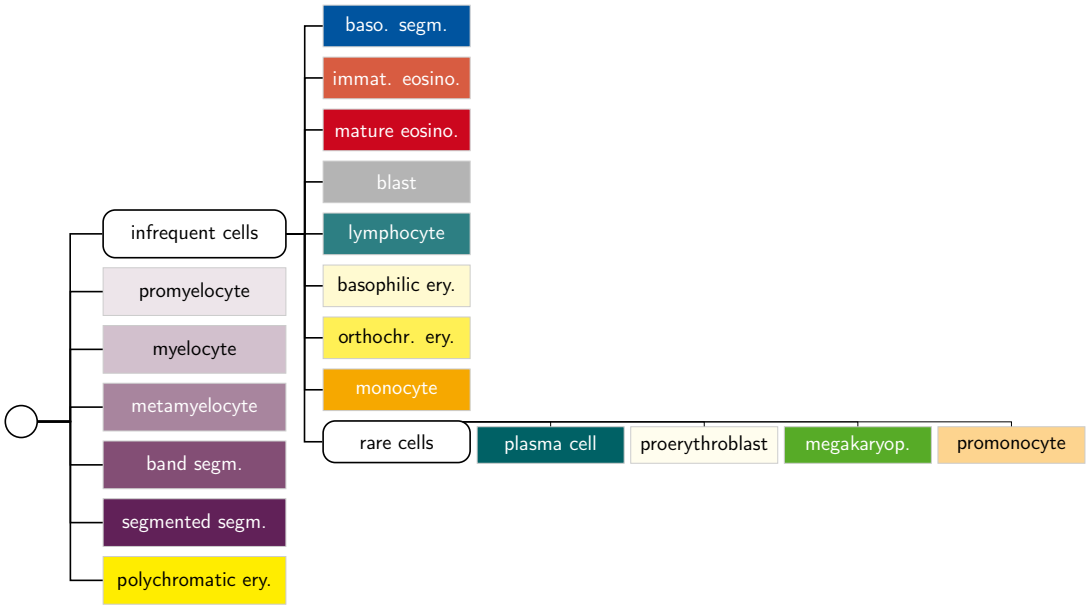


Figure 4.15: Classification tree with similar class-sizes for every network.

all cell types that have less than 80 samples. These groups and their boundaries were selected based on large gaps in the class size statistics while trying to keep the number of groups as low as possible. Basophilic granulocytes could also be assigned to rare cells, but in that case they would outnumber all other rare classes, which are especially vulnerable to class imbalances. Figure 4.16 displays the resulting class frequencies of the different groups in the hierarchy.

4.4.6 Nonsensical Hierarchy for Reference

For evaluation of the impact of cascading the classification process, another hierarchy is introduced. Instead of being inspired by the biological background or the cell's appearances, an existing hierarchy is taken and the leaves are randomly shuffled, leaving the overall structure untouched. Figure 4.17 shows the classification tree which, exemplary, is used in this work, and based on the full hierarchy that was

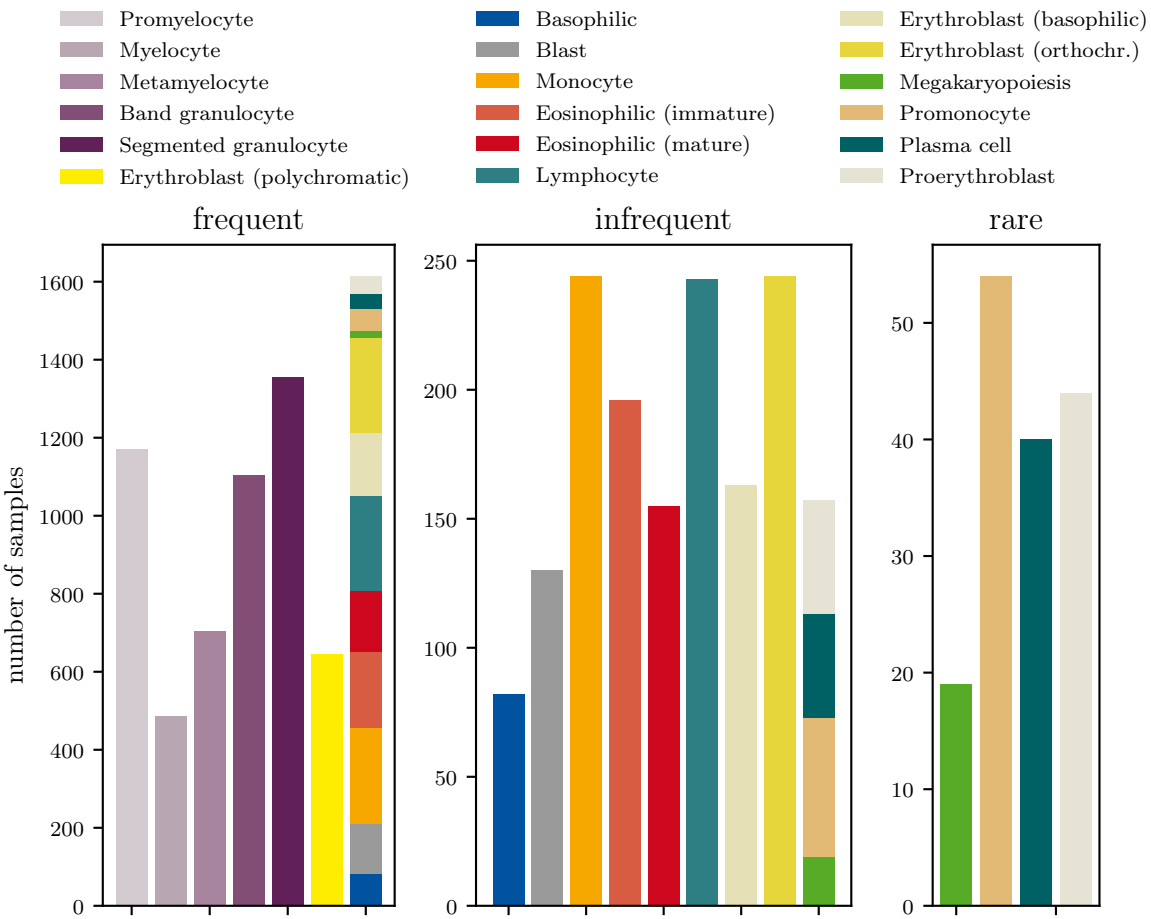


Figure 4.16: Class sizes of the three networks in the hierarchy with reduced class imbalances.

introduced in Section 4.4.1. The structure as well as the networks, including the number of parameters, remain unchanged. However, the random shuffling might make the predictions of the individual networks – especially at early levels, where the classes are composed of many diverse cell types – much more difficult because the intra-class variance is increased. This reference allows an investigation of the effects of the cascading itself by removing the potential gains due to the incorporation of the domain knowledge.

4.5 Maturity Estimation Using Regression

Within early stages of the cell hierarchy, classification is the only appropriate approach to distinguish the unordered individual lineages. However, within these lineages, the cell types can be ordered by maturity. This is especially true for granulocytes, where the different stages are ordinal classes from a continuous process, meaning that the class membership of a cell is not always certain. In particular, this is the case after reaching the state of metamyelocytes, when there are no further

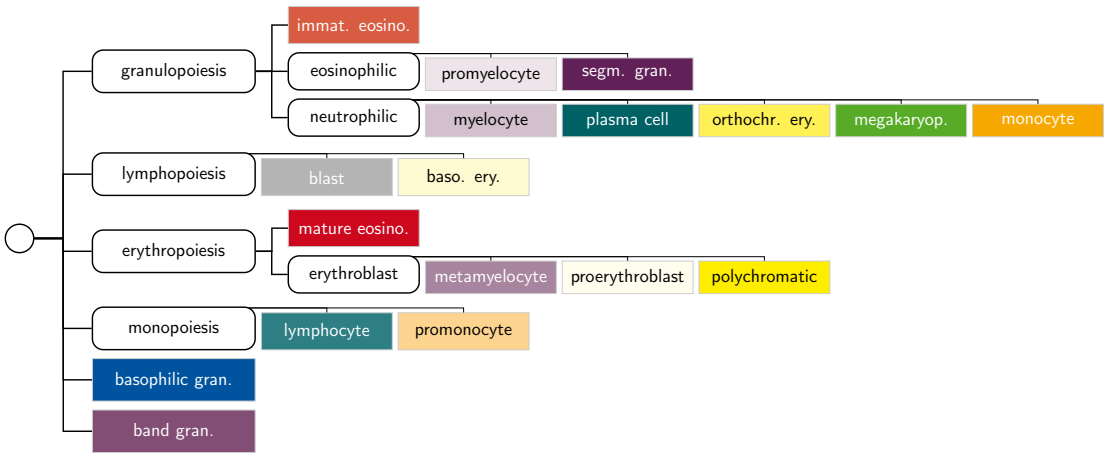


Figure 4.17: Classification tree with no sensible cell groupings. The inner structure is the same as for the full hierarchy as shown in Figure 4.5, only leaves have been interchanged randomly.

cell divisions.

4.5.1 Introducing Regressors for Maturity Estimation

Using a single, flat network, the network must be a classifier, because no global ordering of the classes is possible. However, a cascade of neural networks allows for more variety in the network types. Networks specialised on distinguishing different cell types within a certain lineage can be replaced by regressors to account for the ordinal character of the cells. Such linear regressors only have a single output – instead of one for each class – and are not further processed by an activation function such as softmax. In the following, this is examined for the three lineages of granulocytes. For neutrophilic granulocytes, five stages are distinguished, while for basophilic – if not merged into a single class – and eosinophilic granulocytes there are only two. The maturation stages are converted into linearly separated targets, ranging from zero for the most immature to four for the most mature stage. Figure 4.18 illustrates the targets for the regression.

4.5.2 Transformation of Regression into Classification

Using regression in context of the entire cascade requires hard, deterministic propagation, because the regressor's output y does – other than a classifier's softmax function – not provide a probability estimate. When the task of the overall cascade remains classification, a class label must be derived from y. The regression output y can be converted into classification by simple rounding. Here, the class represented by the closest target to y is selected, as indicated in Figure 4.18. The resulting decision intervals to predict a class label based on the regression output are also indicated in Figure 4.18. It should be noted that, by rounding to the next valid target, the decision boundaries are located in the middle between two neighbouring targets. This means that the output values y generally have different meanings

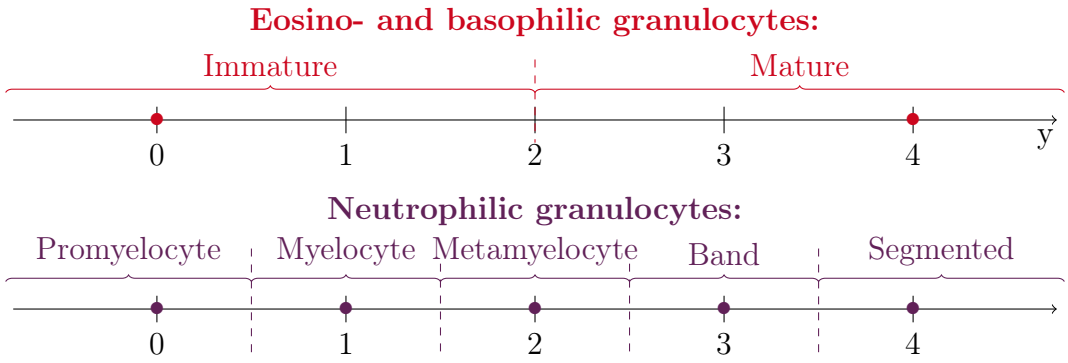


Figure 4.18: Regression labels and decision intervals for granulocytes. Dots mark regression targets, the dashed lines mark the decision boundaries for conversion into classification.

for neutrophilic granulocytes compared to eosino- and basophilic granulocytes. For the latter two lineages, cells are classified as mature – which usually only includes segmented and sometimes band granulocytes – if y exceeds two. However, for neutrophilic granulocytes the same output value might still indicate an immature stage, such as metamyelocyte or band granulocyte. Compared to standard classification with an output value per class and application of softmax function, this approach might be better suited to reduce misclassifications between more distant maturity stages because of its ordinal character.

Alternatively to this conversion into classification, it would also be possible to stop classification at the previous level, assigning only the parent class – for example "neutrophilic granulocyte" – and use the maturity estimation provided by the regressor as additional information. This however would require adjustments to the evaluation procedure. Therefore, in this work, the regression is always converted into classification again.

4.6 Weight Initialisation from Pretrained Networks

A cascade of multiple networks already requires more training than a single network by nature. This makes it even more important to employ transfer learning and initialise the networks from a pretrained state.

4.6.1 Pretrained on ImageNet

The first possibility is pretraining on the ImageNet dataset [22,23]. Such pretrained models are widely available and therefore need no further training before being used for transfer learning. Using ImageNet models as a starting point has proven to lead to major improvements for flat classifications on our dataset [3] and therefore might be useful for a cascade of classification networks as well.

4.6.2 Pretrained on the Haematology Dataset

Another possibility to pretrain the individual networks is to start from a network already trained on the dataset. The individual networks have to specialise on different aspects, but all perform cell classification on the same underlying dataset. First training a network on the entire dataset leads to features generally useful on that domain. Instead of having to relearn those features, the individual networks can then focus on a specialisation process, depending on their individual task within the cascade. Starting from a general, flat network is preferable to, for example, using the previous network in the cascade for initialisation. Ideally, individual networks in the cascade should mostly focus on different features at different positions. Due to this orthogonality of classifiers, transfer from other networks in the cascade would not be helpful, because the feature sets are, in theory, disjoint. A general network, pretrained on the entire dataset at once, on the other hand could possibly provide at least some of the relevant features for an individual network of the cascade. Furthermore, from a more practical point of view, initialisation from another network that is part of the cascade would lead to the loss of the ability to train the networks in an arbitrary order.

4.7 Parameter Sharing Between Individual Networks

Especially low-level features, such as edges and corners, might be redundant among the individual networks. In a cascade of networks of the same architecture, this redundancy might lead to an increased training time and an increased memory requirement.

4.7.1 Division of ResNet and DenseNet into Four Blocks

Both DenseNets and ResNets are comprised of four convolutional blocks with some additional in- and output layers (see Section 3.1). In DenseNets, the four dense blocks (see Figure 3.4) are an obvious choice where the network can be split. The input convolution layer and max pooling layer are not treated as a separate block, but always combined with the first dense block because of the very low number of parameters.

For ResNets, the blocks can be defined similarly (see Figure 3.2). Here, *conv1* and *conv2* together make a first block, and *conv3*, *conv4* and *conv5* each represent another block. Table 4.2 gives an overview on the number of learnable parameters in the four blocks of both architectures at the smallest available depths. Having multiple networks of the same architecture allows to share some or all of these blocks. The accumulated number of parameters in Table 4.2 when using the first k blocks shows a main difference for the two architectures. For ResNet-18, the first three blocks together only make up approximately 25% of the total number of learnable parameters, all others are found in the fourth block, while for DenseNet-

121, the first three blocks combined contain around 69% of the parameters. This leads to the assumption that the effect of sharing blocks may be much stronger for DenseNet than for ResNet.

	ResNet-18			DenseNet-121		
	parameters	in %	accumulated	parameters	in %	accumulated
Block 1	157 504	1.41%	1.41%	377 856	5.43%	5.43%
Block 2	525 568	4.70%	6.11%	1 051 776	15.13%	20.56%
Block 3	2 099 712	18.79%	24.90%	3 364 096	48.38%	68.94%
Block 4	8 393 728	75.10%	100%	2 160 128	31.06%	100%
Total	11 176 512	100%	100%	6 953 856	100%	100%

Table 4.2: Number of learnable parameters in different blocks of ResNet-18 and DenseNet-121. For the final, fully connected layer this number is variable, depending on the number of outputs, and therefore not included in the table.

4.7.2 Sharing of Blocks in a Cascade of Neural Networks

It is possible to share the first one, two, three or even all four of these blocks between individual networks in the cascade. The extreme case of sharing all blocks, including the whole convolutional part of the networks, would not allow for any specialised features at certain points in the cascade whatsoever. Such a cascade would essentially be nothing but a single, flat network – with the only part being adjusted during cascade training being the final fully connected layer. Shared blocks should be taken from a network trained on the same dataset, and then be fixed during cascade training. Adjusting the shared layers as well would not reduce the training time. Furthermore, the separable training of the individual networks would become incompatible with shared layers. The reason is that the order of training could make a significant difference, as the training data for these shared layers would not be shuffled but determined by the currently trained individual network.

The common blocks of the networks comprise an entirely convolutional neural network, as shown in Figure 4.19, with the sole purpose of feature extraction from the input image $\boldsymbol{I}$. When forwarding a sample through the cascade, the image is first processed by this *extractor network*, which provides an intermediate feature representation $\boldsymbol{F}$ as its output. Passing the image through these layers only needs to be performed once. Then, the cascade is traversed as usual, but each – now smaller – network is given the feature map $\boldsymbol{F}$ instead of the entire image.

4.8 Compensation of Class Imbalances

The class sizes in the dataset are strongly imbalanced. In a hierarchy, classes are combined in early levels, and the resulting class sizes are the sum of the sizes of their

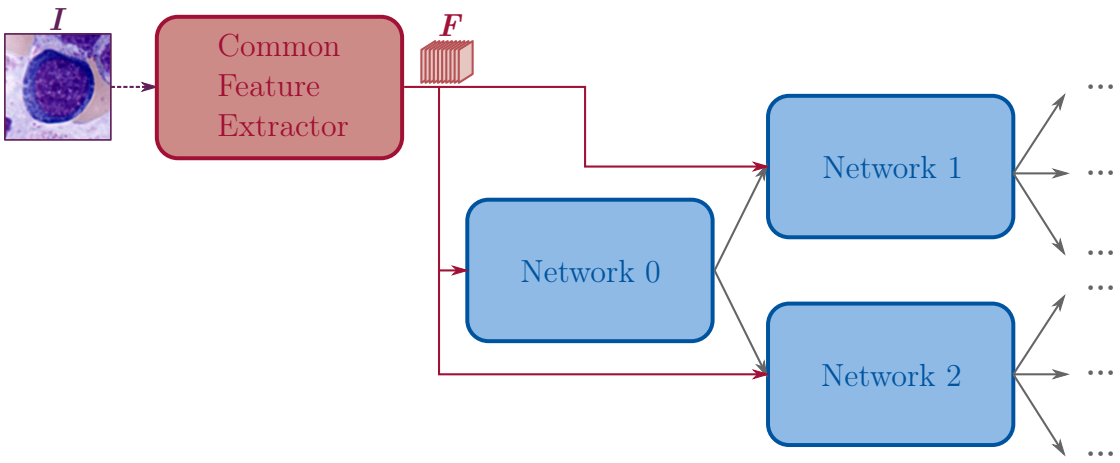


Figure 4.19: Principle of block sharing. The image $\boldsymbol{I}$ is given to the feature extractor network, and the resulting feature maps $\boldsymbol{F}$ to the individual networks.

member-classes. Section 4.4.5 already addressed a structure designed to compensate for the class imbalance. Two other ways of partly compensating class imbalance that can be used in flat networks as well as in the individual networks in a cascade are loss weights and oversampling. These two methods are introduced in the context of hierarchical classification in the following.

4.8.1 Loss Weights

Plain cross-entropy loss (2.9) is impacted by class imbalances. Averaged over the entire dataset, more frequent classes contribute more terms to the sum, while small classes have a smaller impact. To compensate this, weight factors γ_i can be introduced into the cross-entropy loss

$$\mathcal{L}_{\mathrm{WCE},\gamma} = -\frac{1}{N}\sum_{i=1}^{N}\gamma_i\sum_{c=1}^{n}t_{i,c}\cdot\log(y_{i,c}), \tag{4.11}$$

giving small classes a higher weight [51]. Such loss is called the weighted cross-entropy loss (WCE). The weight factor can for example be defined based on the inverse class frequencies to compensate class imbalances. Let N_i be the size of the i-th out of in total n classes such that

$$N = \sum_{i=1}^{n} N_i. \tag{4.12}$$

Then, a possible way to calculate loss weights of the individual classes is

$$\gamma_i = \frac{n}{N_i^{\delta}\cdot\sum_{j=1}^{n}\frac{1}{N_j^{\delta}}}. \tag{4.13}$$

The parameter δ can be used to tweak the weight. For $\delta = 0$, this results in the standard, unweighted cross-entropy loss.

In a cascade of networks, class sizes at intermediate levels are given by the sum of the combined class sizes of all descendant leaf classes. Compared to a flat classifier, the loss weights can be calculated for every individual network in hierarchical classification separately. Over the course of the cascade, this means that the loss from a given sample can be weighted differently, based on the level in the hierarchy and the size of the combined class at that position.

4.8.2 Oversampling of Underrepresented Classes

Another possibility to address class imbalance is oversampling of underrepresented classes in combination with data augmentation. Oversampling means that samples from smaller classes are repeated within an epoch randomly. In combination with data augmentation, the training data added this way become more heterogeneous. For a given class with N_i samples in the dataset, the resulting number of samples N_i^* can, for example, be determined by

$$N_i^* = N_i \cdot \left(\frac{\max\limits_{j \in \{1,\dots,n\}} (N_j)}{N_i} \right)^{\nu} . \tag{4.14}$$

Similar to the loss weight, the oversampling can be tweaked by a parameter ν. The class frequencies are unchanged for $\nu = 0$, while any $\nu > 0$ oversamples the underrepresented classes. For $\nu = 1$, the resulting numbers of samples per class are calculated as the fraction of samples in a given class compared to the largest class, while $\nu = 0.5$ is the square root of this fraction.

As for the loss weights, oversampling can be performed separately for all individual networks within a cascade of networks. Different class sizes at different positions in the hierarchy require different oversampling to achieve balanced class sizes.

4.9 Feature Forwarding

So far, the individually trained networks in the cascade have been completely independent. There is no need for any dependencies if the features distinguishing the classes at different levels are orthogonal, meaning that the sets of features relevant for different decisions in the cascade are disjoint. If this assumption holds, the defining features at a given level are completely irrelevant for all later levels. For example, one network might only aim to decide whether a cell has a round or segmented nucleus, while a subsequent network only judges the granulation, completely independently of the nucleus shape. However, in many cases, this assumption does not hold, because there are correlations of such cellular characteristics. For this reason, *feature forwarding* is introduced. It introduces embeddings into the cascade,

which can also be useful in other aspects, for example for visualisation of the model's feature space.

4.9.1 Principle

Features learned by early networks for prediction of more general classes may also be useful for later, more specific classification. The reasons are that, in many cases, more than a single feature might influence the decision, and even if different cell types in a certain group within the hierarchy share a certain characteristic, it can be more pronounced for some than for others. To exploit these already learnt features, feature forwarding is introduced. Its idea is inspired by residual connections in deep architectures in general, and in particular by DenseNets [13], as it introduces skip connections from one stage to all following stages in the cascade. This is illustrated in Figure 4.20.

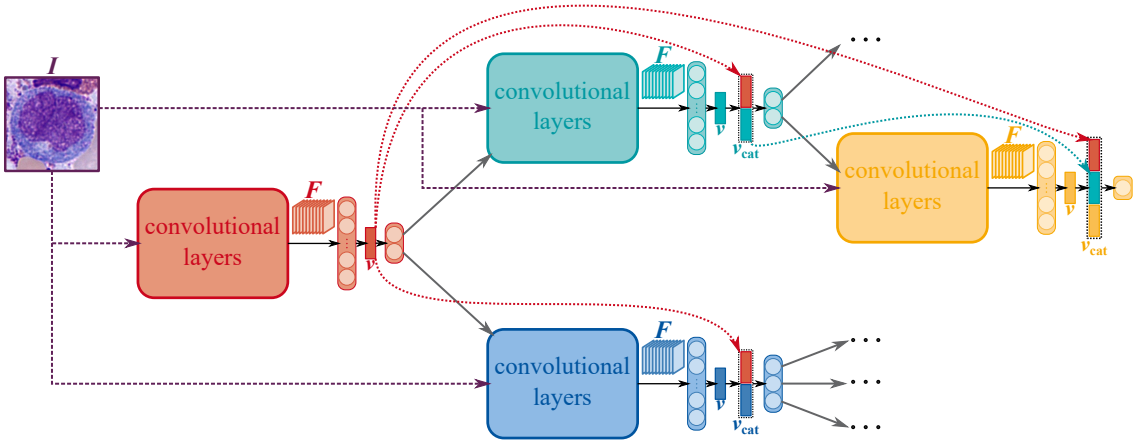


Figure 4.20: Principle of feature forwarding. In an intermediate step, each network learns a feature map, represented by a small, coloured box, which is concatenated with the feature maps of all following networks, as indicated by the dashed boxes.

In all networks, an additional fully connected layer is inserted right before the final linear layer. This additional layer receives the feature maps $\boldsymbol{F}$ from the convolutional layers and has a fixed output size, the embedding length n_{emb}. The networks each learn an n_{emb}-dimensional feature representation $\boldsymbol{v_i}$ which – ideally – allows the best possible separation between classes at its level. These embedding vectors are passed to all descendants in the hierarchy, providing the learnt features to all following networks along a path in the cascade. In the root network, its embedding vector is directly provided to the classification layer to obtain its output as usual. The final classification layers of all non-root networks receive the concatenation

$$\boldsymbol{v}_{\text{cat},i} = [\boldsymbol{v_0},\ \ldots,\ \boldsymbol{v_i}] \tag{4.15}$$

$$= \left[\boldsymbol{v}_{\text{cat,parent}(i)},\ \boldsymbol{v_i}\right] \tag{4.16}$$

of all preceding feature maps as well as the embedding given by the convolutional layers at their own stage. These concatenations are performed successively along paths from the root to the leaves. Consequently, for a network at depth l in the cascade – starting with $l = 1$ for the root – the final linear layer receives an input vector with

$$n_{\text{cat}}(l) = l \cdot n_{\text{emb}} \tag{4.17}$$

dimensions

4.9.2 Training with Feature Forwarding

For more efficient training when using feature forwarding, the training algorithm is adjusted. The networks are always trained separately in order as given by the hierarchy, starting from the root. Each network is fully trained before moving to the next one because of the one-way root-to-leaf dependency. This way, later networks receive the final, fixed feature representations from their predecessors for a given input, such that training is performed with as much information about the final model as possible. As a side note, this procedure is only possible when only the feature maps of the ancestors are provided to a network. If other feature maps were included, for example from other networks at the same level in the hierarchy, it would be unclear in which order to train the networks. The motivation for this training procedure is to create different, complementing features, and to reduce redundancy. Although the outcome now also depends on all earlier networks, their weights remain unchanged to prevent interference with the earlier stage classification as well as with the siblings, which are also relying on the same embedding vectors. As a consequence, even though it is theoretically possible to resume training for more epochs, it has to be noted that this will not necessarily yield the same outcome as training a fresh model for the increased total number of epochs.

4.9.3 Use of Embeddings for Training Visualisation

Often, embeddings are, after application of dimensionality reduction, used for visualisation of the learned feature space. In a cascade of networks using feature forwarding, there are different embeddings at different positions in the network. This allows for visualisation of the individual decisions at any position in the hierarchy, by only considering the concatenated embeddings used for the individual network at that position.

However, a visualisation of the entire training process including all classes at once is not trivial. In a cascade of neural networks, it is not obvious how to combine the different feature maps at different levels. Furthermore, using the concatenated embeddings which are used by a specific network at leaf level for visualisation is problematic, because, unless it is an ultrametric tree, the length of these combined feature vectors is not the same for all leaves. A straightforward way to compensate for this is, for example, zero padding to achieve a common length, or to collect and concatenate the embeddings from all networks just for this purpose.

4.10 Macro-averaged Hierarchical F-score

The hierarchical F-score introduced by Kiritchenko et al. in 2005 [28] is micro-averaged. On an unbalanced dataset such as the haematological dataset onto which this work is focused, it might be preferable to use macro-averaging. Inspired by the original hierarchical F-score, such a macro-averaged hierarchical F-score is introduced in this section.

A hierarchical, class-wise precision of class c_i can be defined as

$$hP(c_i) = \frac{\sum_{j=1}^{n_i^*} |\hat{C}_i \cap \hat{C}_j^*|}{\sum_{j=1}^{n_i^*} |\hat{C}_j^*|}, \tag{4.18}$$

where

$$n_i^* = N_{\mathrm{TP},i} + N_{\mathrm{FP},i} \tag{4.19}$$

is the number of samples that were predicted to be from class c_i, regardless of whether this prediction is correct or incorrect. The numerator of (4.18) describes the sum of common ancestors of the multi-label $\hat{C}_i$ of class c_i excluding the root and the true multi-labels $\hat{C}_j^*$ of all samples that were predicted to be member of C_i. This sum is divided by the sum of the numbers of ancestors of the correct labels of all samples predicted to be of class c_i. Following this definition, the hierarchical class-wise precision contains contributions of all true positive and all false positive samples, as does the precision in non-hierarchical evaluation (see equation (2.19)).

Conversely to the non-hierarchical recall (2.20), the hierarchical, class-wise recall should consider all true positive and false negative samples. Therefore, it can be defined as

$$hR(c_i) = \frac{\sum_{j=1}^{n_i} |\hat{C}_i \cap \hat{C}_j'|}{\sum_{j=1}^{n_i} |\hat{C}_i|}. \tag{4.20}$$

For all

$$n_i = N_{\mathrm{TP},i} + N_{\mathrm{FN},i} \tag{4.21}$$

samples which are truly members of class c_i, the numbers of common ancestors of the correct multi-label $\hat{C}_i$ and the predicted multi-label $\hat{C}_j'$, always excluding the root, are summed, and divided by sum of true ancestors of the correct class.

The class-wise F-scores

$$hF_1(c_i) = \frac{2 \cdot hP(c_i) \cdot hR(c_i)}{hP(c_i) + hR(c_i)} \tag{4.22}$$

are then averaged over all classes, resulting in the macro-averaged hierarchical F_1-score

$$hF_{1,\mathrm{macro}} = \frac{1}{n} \sum_{i=1}^{n} hF_1(c_i). \tag{4.23}$$

5 Experiments

In this chapter, the different experiments conducted during this work are presented. Each experiment is divided into three sub-sections. At first, the experiment setup is described, before the results are presented, and lastly analysed and discussed.

5.1 General Experimental Setup

Some parameters are shared among most or all experiments. These common setting are described in the following.

5.1.1 Default Hyperparameter Settings

Unless explicitly stated otherwise, all experiments are conducted with the following default settings and hyperparameters:

- Learning rate: $\alpha = 1 \cdot 10^{-5}$
- Image patch size: 224 px × 224 px
- Optimiser: Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$
- Transfer learning: initialisation from convolutional layers pretrained on ImageNet (ILSVRC); parameters are not fixed during further training
- Data augmentation: rotation, mirroring and random cropping
- Classification loss: cross-entropy loss
- Training duration: 200 epochs
- Batch size: 16 samples per batch
- Validation score for model selection: $F_{1,\text{macro}}$

These parameters are selected based on a previous hyperparameter optimisation study [3] with additional constraints to keep GPU-memory requirements reasonably low. Individual networks within cascades as well as the flat reference networks are trained with these settings. The image patches are fixed to a size of 224 px × 224 px because this corresponds to the image size of the pretrained ResNet and DenseNet in torchvision [51]. Networks are never trained from scratch, but usually pretrained on the ImageNet dataset. Only if specified, the networks are pretrained on the haematological dataset. This pretraining itself is also performed utilising

transfer learning from ILSVRC. Input images are therefore normalised based on the characteristics of the ImageNet dataset. For this purpose, following equation (2.12), the mean vector

$$\boldsymbol{\mu} = 255 \cdot \begin{pmatrix} 0.485 \\ 0.456 \\ 0.406 \end{pmatrix} \tag{5.1}$$

is subtracted from all pixel values of the RGB images, which are subsequently divided by the corresponding component of the standard deviation

$$\boldsymbol{\sigma} = 255 \cdot \begin{pmatrix} 0.229 \\ 0.224 \\ 0.225 \end{pmatrix} . \tag{5.2}$$

5.1.2 Dataset Splits and Cross-Validation

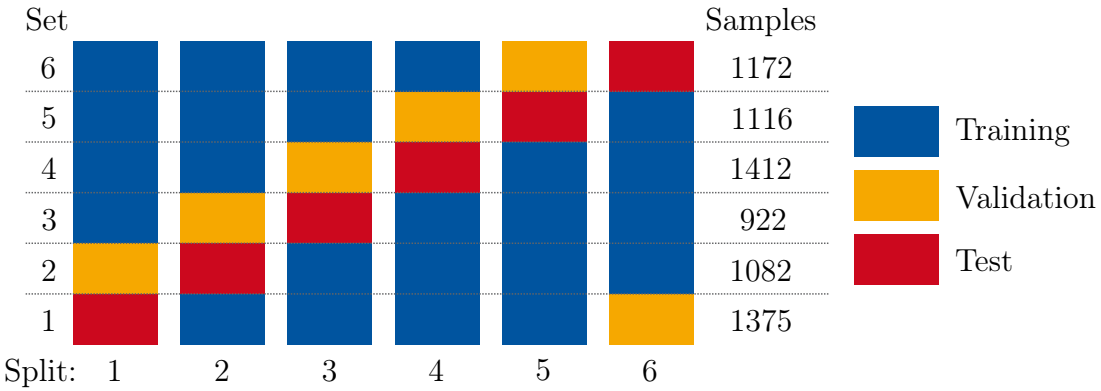


Figure 5.1: Dataset splits for cross-validation and number of samples in each subset.

All experiments are performed with 6-fold cross-validation, as visualised in Figure 5.1. For this purpose, the dataset is divided into six sets, of which four, containing approximately two thirds of the samples, are used for training, while $\frac{1}{6}$ of the data are reserved for validation and testing, respectively. Samples are not assigned randomly to the subsets. Instead, their position within the slices is taken into account. The slices are divided into blocks of 4096 px × 4096 px, and all cells within these blocks are assigned to the same subset, while cells at their edges are discarded. This results in more independent sets. However, as a side effect, the resulting set sizes can differ relatively strongly, as is also indicated in Figure 5.1.

Every experiment is performed six times, once for each split. It should be noted that, in theory, $6 \times 5 = 30$ unique dataset splits can be created from six folds. However, because of the large number of experiments given limited computational and temporal resources, in the following all experiments are performed only on these six splits. Each subset, and consequently each sample, is member of the test and validation set exactly once, respectively, hence having the same impact on the final, averaged test score. It was empirically shown that such a 6-fold cross-validation

can sufficiently approximate the means and standard deviations of a complete cross-validation with all possible combinations of subsets [3].

In a cascade, the individual networks are trained only on the relevant subsets of these sets, as discussed in Section 4.3. These subsets are created while preserving the overall subset membership. This means that a sample that is, for example, in the first set of the overall dataset is also in the first set of all subsets used to train individual networks. Hence, it is assured that no sample is in the training set for one network, but in the validation or test set for another network or the overall cascade, which would lead to partial overfitting of the cascade and consequently invalidate the validation and test results.

For evaluation of the performances of the classifiers, the macro-averaged F_1-score, the micro-averaged F_1-score – which, as shown in Appendix A.1, is the same as the accuracy – and the macro- and micro-averaged hierarchical F_1-scores are observed. Therefore, there is a macro-averaged measure and a micro-averaged measure both in the usual sense as well as taking the hierarchy into account. In all experiments, the mean values

$$\text{mean} = \frac{1}{k} \cdot \sum_{i=1}^{k} p_i \tag{5.3}$$

as well as the unbiased standard deviations

$$\text{std} = \sqrt{\frac{\sum_{i=1}^{k} (p_i - \text{mean})^2}{k-1}} \tag{5.4}$$

are reported, calculated from all $k = 6$ folds, where p_i is the respective performance score on the i-th fold.

5.1.3 Individual Networks in the Cascades and Cascade Propagation

For individual networks in the cascades, two architectures are investigated: ResNet and DenseNet, which have previously shown the most promising results on the dataset [3]. Because there are multiple networks in every cascade, only their most shallow versions are considered, which are ResNet-18 and DenseNet-121. Deeper networks would require much more GPU-memory and time for training, and are therefore not feasible. The different hierarchies, including the number of networks and the resulting number of learnable parameters, are listed in Table 5.1. For more detailed descriptions of the different hierarchies, the sections of their introduction are also listed. In most experiments, soft, probabilistic traversal of the cascade is performed, and – unless stated otherwise – the networks are trained individually. Specialised methods such as pretraining on a network trained on the haematology dataset, block sharing for feature extraction, feature forwarding, loss weights or upsampling are not employed unless explicitly mentioned.

Hierarchy	See	Networks	Learnable parameters (in million) ResNet 18	ResNet 152	DenseNet 121	DenseNet 201
Flat (ref.)	[12, 13]	1	11.2	58.2	7.1	18.3
Full	4.4.1	8	89.4	-	55.7	-
Ultrametric	4.4.2	7	78.2	-	48.7	-
Cytoplasm features	4.4.3	7	78.2	-	48.7	-
Ratio/shape features	4.4.3	7	78.2	-	48.7	-
Nucleus features	4.4.3	7	78.2	-	48.7	-
Confusion clustering	4.4.4	5	55.9	-	34.8	-
Similarity clustering	4.4.4	6	67.0	-	41.7	-
Balanced	4.4.5	3	33.5	-	20.9	-
Random	4.4.6	8	89.4	-	55.7	-

Table 5.1: Summary of the different hierarchies, including the nomenclature used throughout this chapter. The total number of learnable parameters is listed for different base architectures.

5.1.4 Reference Networks

For reference, multiple flat networks are trained on the same data as the cascades. The architectures used as reference are the previously best-performing DenseNet-121 and ResNet-152 [3], and the deeper DenseNet-201. Additionally, a flat ResNet-18 is also trained for comparison with the cascades consisting of multiple of these networks. The numbers of learnable parameters are stated in the first row of Table 5.1. For some experiments, an additional DenseNet-121 is trained as well, if the methods investigated in these experiments are applicable also to single, flat classification networks. In all cases, the general training parameters are the same as for the cascades.

5.2 Comparison of Cascade Hierarchies

In the first experiment, the different hierarchies, as introduced in Section 4.4, are examined in the most basic setup. The experiment serves multiple purposes: on the one hand, the general performances and other aspects of the different cascades are evaluated and compared to flat classifiers. Furthermore, the comparison of the hierarchies among each other could provide insight into the underlying mechanisms and principles.

Experimental Setup

For this experiment, cascades of all nine hierarchies introduced in Section 4.4 are evaluated in the basic setup, including separate training of the individual networks and soft, probabilistic propagation for decision making. As reference, different flat networks as described in 5.1.4 are trained.

Results

The results are listed in Table 5.2, the macro averaged F-scores are additionally visualised in Figure 5.2. The highest performances are achieved by the deepest flat reference classifiers. In terms of the micro-averaged scores, ResNet-152 attains the highest performance, while DenseNet-201 achieves slightly higher macro-averaged scores. The non-hierarchical scores and their hierarchical counterparts are strongly correlated. The Pearson correlation coefficient of $F_{1,\text{macro}}$ and $hF_{1,\text{macro}}$ is 0.941, while the correlation coefficient between $F_{1,\text{micro}}$ and $hF_{1,\text{micro}}$ is 0.967.

		$F_{1,\text{macro}}$ [%]	$F_{1,\text{micro}}$ [%]	$hF_{1,\text{macro}}$ [%]	$hF_{1,\text{micro}}$ [%]
		mean ± std	mean ± std	mean ± std	mean ± std
Flat	RN-18	70.3 ± 2.6	76.8 ± 1.4	88.8 ± 1.4	93.8 ± 0.6
	RN-152	71.4 ± 4.3	**78.7** ± 1.6	89.0 ± 2.4	**94.3** ± 0.7
	DN-121	70.7 ± 3.1	77.9 ± 1.7	88.9 ± 1.7	94.1 ± 0.6
	DN-201	**71.8** ± **3.1**	77.9 ± 1.3	**89.8** ± 1.5	94.1 ± 0.5
Full	RN-18	70.3 ± 2.2	77.1 ± 1.2	89.1 ± 1.2	94.0 ± 0.5
	DN-121	70.1 ± 2.1	77.4 ± 1.0	89.2 ± 0.9	94.0 ± 0.4
Ultrametric	RN-18	70.6 ± 2.3	77.4 ± 0.9	89.3 ± 1.3	94.0 ± 0.4
	DN-121	70.1 ± 4.4	77.5 ± 1.9	88.5 ± 2.8	94.0 ± 0.6
Cytoplasm features	RN-18	69.1 ± 2.6	76.5 ± 0.9	87.7 ± 2.1	93.5 ± 0.5
	DN-121	66.4 ± 2.4	77.6 ± 1.7	84.6 ± 1.7	93.9 ± 0.7
Ratio/shape features	RN-18	66.0 ± 4.1	76.1 ± 1.4	84.6 ± 5.0	93.3 ± 0.6
	DN-121	63.8 ± 3.0	76.1 ± 1.4	82.8 ± 3.0	93.4 ± 0.4
Nucleus features	RN-18	65.4 ± 3.6	75.2 ± 1.6	84.7 ± 3.4	93.2 ± 0.5
	DN-121	65.1 ± 2.3	76.1 ± 1.5	83.9 ± 2.6	93.5 ± 0.5
Confusion clustering	RN-18	66.2 ± 1.1	76.1 ± 1.0	82.5 ± 1.6	93.6 ± 0.4
	DN-121	65.9 ± 0.8	77.2 ± 1.3	81.6 ± 1.8	93.8 ± 0.4
Similarity clustering	RN-18	64.4 ± 1.8	76.4 ± 1.2	83.5 ± 0.6	93.6 ± 0.5
	DN-121	66.9 ± 2.2	76.9 ± 1.5	85.0 ± 1.3	93.9 ± 0.5
Balanced	RN-18	69.2 ± 3.5	77.3 ± 1.6	88.6 ± 1.5	93.9 ± 0.5
	DN-121	70.5 ± 3.2	78.2 ± 1.8	88.8 ± 1.4	94.1 ± 0.7
Random	RN-18	62.3 ± 5.1	74.8 ± 1.1	81.2 ± 5.8	93.1 ± 0.6
	DN-121	67.4 ± 1.1	76.2 ± 1.4	87.7 ± 0.6	93.6 ± 0.5

Table 5.2: Average performances of the different hierarchies with ResNet (RN) and DenseNet (DN) as base architectures.

The highest-performing cascades reach scores within the standard deviations of the best-performing reference networks. Hierarchies with the highest scores are, in general, the full cascade and the ultrametric taxonomy, closely followed by the balanced hierarchy. The two hierarchies obtained by spectral clustering of the confusion matrix and the medical experts' similarity survey do not reach the scores of the cascades oriented more closely on the biological lineages. Out of the feature-based cascades, only the hierarchy based cytoplasm characteristics achieves an $F_{1,\text{macro}}$-score close to the reference, and this only with ResNet-18 networks as base learners.

Compared to the full hierarchy, the same tree with randomly swapped leaves reaches much lower performances, especially in terms of the macro-averaged scores. This is visualised in Figure 5.3, which shows the difference of the confusion matrices

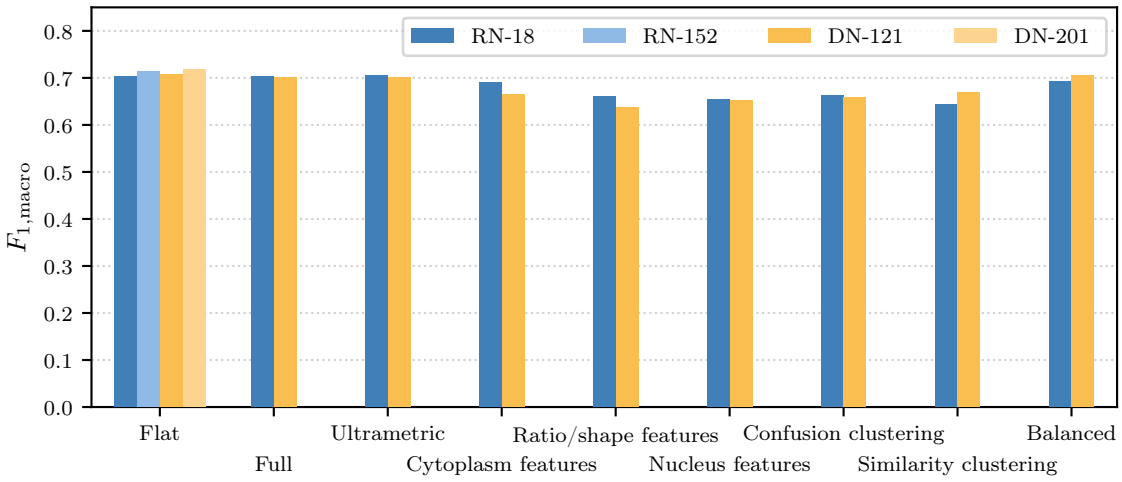


Figure 5.2: Macro-averaged F-scores of different hierarchical cascades with different ResNet (RN) and DenseNet (DN) networks. Flat network performances are given as reference.

of the full hierarchy and the random hierarchy. Table 5.3 lists the macro- and micro-averaged F-scores achieved by the individual networks in these two cascades. Hierarchical scores are not listed, because the individual networks only handle classes from a single level in the hierarchy. While the full cascade attains considerably higher scores in the root network and the granulopoiesis network, the networks in the random hierarchy reach higher scores in all other nodes of the hierarchy.

	Full cascade (ResNet-18)		Random (ResNet-18)	
	$F_{1,\text{macro}}$ [%]	$F_{1,\text{micro}}$ [%]	$F_{1,\text{macro}}$ [%]	$F_{1,\text{micro}}$ [%]
Node name	mean ± std	mean ± std	mean ± std	mean ± std
Root	78.8 ± 4.6	95.0 ± 1.0	67.6 ± 8.1	79.0 ± 0.8
Granulopoiesis	87.6 ± 9.7	97.3 ± 2.4	80.0 ± 14.1	85.9 ± 7.2
Lymphopoiesis	80.2 ± 18.9	97.6 ± 2.3	85.2 ± 13.8	89.2 ± 7.8
Erythropoiesis	83.3 ± 17.2	97.9 ± 2.1	88.9 ± 13.5	91.9 ± 8.2
Monopoiesis	80.8 ± 16.7	94.7 ± 7.7	86.4 ± 16.0	92.6 ± 7.7
Eosiniphilic gran.	80.3 ± 15.5	92.2 ± 9.6	88.7 ± 15.4	93.8 ± 7.5
Neutrophilic gran.	80.1 ± 14.4	90.6 ± 9.6	89.2 ± 14.4	93.8 ± 7.0
Erythroblast	79.4 ± 13.6	88.9 ± 10.2	90.4 ± 13.9	94.6 ± 6.8

Table 5.3: Network-wise test scores of the full hierarchy (Figure 4.5) cascade compared to the random hierarchy cascade (Figure 4.17), which have the same underlying structure, but with swapped leaves in the random hierarchy. The node names therefore represent the position in the full hierarchy, for the random hierarchy they do not imply actual sub-class memberships.

Table 5.4 lists the class-wise F_1-scores of a flat DenseNet-121 and the three highest-performing cascades (see Appendix A.3 for the class-wise scores of all hierarchies). The individual scores are relatively similar. In all architectures, the promonocytes

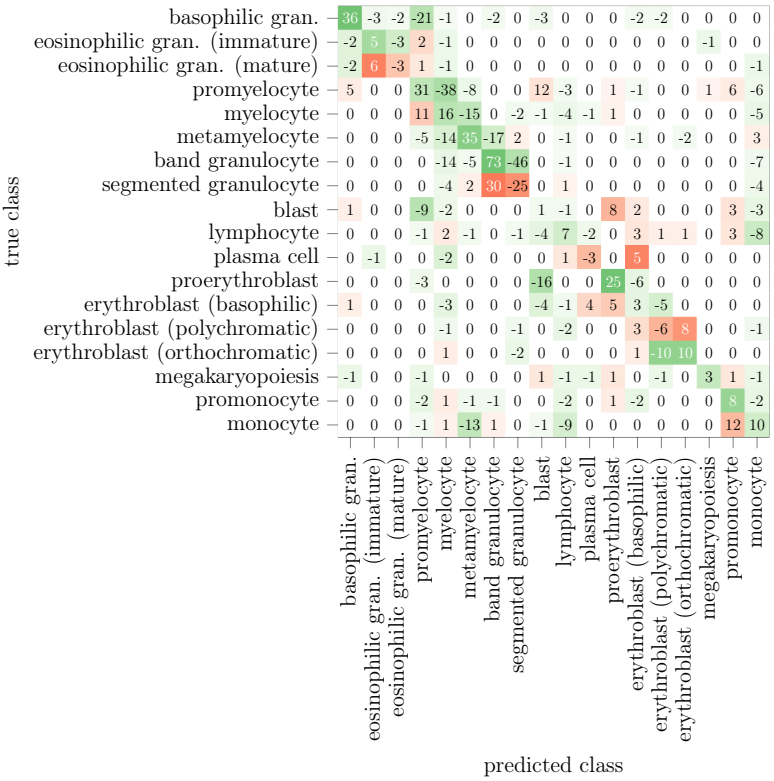


Figure 5.3: Difference of the confusion matrices to evaluate the performance of the full hierarchy compared to the random hierarchy. Positive values indicate that there are more predictions from the full hierarchy cascade.

stand out with a very low score ranging from 18.8 % for the reference and 26.3 % achieved by the balanced hierarchy.

Regarding the individual network architectures, ResNet-18 achieve a higher $F_{1,\text{macro}}$ in six out of nine cases compared to hierarchies with DenseNet-121 networks. However, the mean macro-averaged F-scores are, with 67.1 % for ResNet-18 and 67.4 % for DenseNet-121, very close – and slightly higher for DenseNet-121.

Table 5.5 lists the test scores of the individual networks in a full hierarchy cascade for both base architectures. Both $F_{1,\text{macro}}$ an $F_{1,\text{micro}}$ are relatively similar, both in terms of the mean values as well as for the standard deviations. The macro-averaged F_1-scores of DenseNet-121 networks are, on average, larger than their ResNet-18 counterpart by approximately 0.2 %. For the micro-averaged F-score, the average absolute difference is 0.1 %.

Discussion

The experimental results allow comparison of the performance of the hierarchical cascades with individual training and soft propagation against flat networks as well as an analysis of the different hierarchies with the two evaluated individual network architectures.

	DenseNet-121 (ref.)	Full ResNet-18	Ultrametric ResNet-18	Balanced DenseNet-121
	mean ± std	mean ± std	mean ± std	mean ± std
Basophilic granulocyte	86.5 ± 3.5	90.0 ± 5.0	88.0 ± 4.0	88.3 ± 4.9
Eosino. gran. (immat.)	76.8 ± 7.2	79.0 ± 6.3	76.9 ± 6.7	79.1 ± 7.2
Eosino. gran. (mature)	76.1 ± 10.2	75.5 ± 8.2	78.1 ± 5.1	76.1 ± 7.4
Promyelocyte	89.0 ± 2.0	87.7 ± 2.6	88.1 ± 2.3	89.2 ± 2.1
Myelocyte	67.3 ± 6.0	66.5 ± 4.9	65.9 ± 2.8	68.8 ± 5.3
Metamyelocyte	67.1 ± 4.0	65.1 ± 2.4	64.5 ± 3.4	66.6 ± 3.0
Band granulocyte	75.9 ± 3.4	73.9 ± 2.5	75.2 ± 1.7	74.9 ± 2.8
Segmented granulocyte	88.5 ± 1.4	88.4 ± 1.1	89.2 ± 0.8	89.4 ± 1.2
Blast	55.4 ± 6.8	56.0 ± 8.7	53.5 ± 7.6	54.1 ± 3.9
Lymphocyte	76.8 ± 5.7	75.7 ± 6.7	75.9 ± 6.0	76.7 ± 5.8
Plasma cell	67.6 ± 35.2	76.1 ± 21.0	71.4 ± 17.4	72.0 ± 20.1
Proerythroblast	66.4 ± 17.5	66.5 ± 9.6	63.6 ± 22.5	59.0 ± 15.3
Erythroblast (basophilic)	70.2 ± 7.1	65.4 ± 6.7	70.6 ± 10.2	74.6 ± 10.8
Erythroblast (polychr.)	80.8 ± 4.2	80.2 ± 3.4	80.9 ± 3.7	81.4 ± 2.9
Erythroblast (orthochr.)	63.0 ± 7.0	63.0 ± 5.2	64.0 ± 6.2	60.7 ± 11.7
Megakaryopoiesis	84.9 ± 18.9	80.4 ± 21.5	87.1 ± 17.0	71.0 ± 38.3
Promonocyte	19.0 ± 19.0	18.8 ± 14.0	21.0 ± 3.5	26.3 ± 14.6
Monocyte	60.5 ± 6.9	57.9 ± 4.2	56.7 ± 11.6	61.1 ± 9.2

Table 5.4: Class-wise F_1-scores of a flat DenseNet-121 and the three best-performing cascades.

	Full cascade (ResNet-18)		Full cascade (DenseNet-121)	
	$F_{1,\text{macro}}$ [%]	$F_{1,\text{micro}}$ [%]	$F_{1,\text{macro}}$ [%]	$F_{1,\text{micro}}$ [%]
Node name	mean ± std	mean ± std	mean ± std	mean ± std
Root	78.8 ± 4.6	95.0 ± 1.0	78.9 ± 4.6	94.9 ± 0.8
Granulopoiesis	87.6 ± 9.7	97.3 ± 2.4	87.7 ± 9.8	97.2 ± 2.4
Lymphopoiesis	80.2 ± 18.9	97.6 ± 2.3	79.7 ± 18.7	97.3 ± 2.4
Erythropoiesis	83.3 ± 17.2	97.9 ± 2.1	83.1 ± 17.3	97.7 ± 2.2
Monopoiesis	80.8 ± 16.7	94.7 ± 7.7	81.5 ± 16.2	95.2 ± 5.9
Eosiniphilic gran.	80.3 ± 15.5	92.2 ± 9.6	80.8 ± 14.9	92.5 ± 8.5
Neutrophilic gran.	80.1 ± 14.4	90.6 ± 9.6	80.6 ± 13.8	91.0 ± 8.6
Erythroblast	79.4 ± 13.6	88.9 ± 10.2	79.7 ± 13.2	89.1 ± 9.6

Table 5.5: Comparison of network-wise scores for ResNet-18 and DenseNet-121 as base architectures in a full hierarchy cascade.

Performance Comparison with Flat Networks

In general, the results indicate that, without further adjustments, hierarchical splitting of the classification process does not inherently increase the classification performance. The cascades, at best, come close to the reference scores, and often perform much worse. An analysis of the class-wise F_1-scores reveals that there are no major differences between the cascades and a flat network, showing that no gains are achieved by cascading. The worst performance, by a relatively large margin, is

achieved for promonocytes, which are confused with several different classes. With only 54 samples, this class is vastly underrepresented. Furthermore – in contrast to the even rarer, but uniquely large megakaryopoietic cells – they share more features with other, similar cell types. For example for a flat DenseNet-121, they are mainly confused with promyelocytes, myelocytes, blasts and monocytes, as can be seen from the confusion matrix, which is shown in Figure 4.12 already in Section 4.4.4. All these types have a similar appearance, including a relatively basophilic stain.

Comparison of the Different Hierarchies

The best results are achieved by the full and the ultrametric hierarchies, which are closest related to the biological lineages of haematopoiesis. Apart from these two hierarchies, the hierarchy with reduced class imbalance also performs reasonably well.

The worst performing cascade is the full cascade with randomly shuffled leaves, which consequently does not incorporate any domain knowledge. It shows that just having more classifiers per se does not positively impact the results, and that, if classification is performed hierarchically, the cell groupings are important.

An analysis of the difference of the confusion matrices (Figure 5.3) and the test scores achieved by the individual networks (Table 5.3) provides some insight into the reasons of the performance difference, as well as error propagation within the cascades. The main reason for the performance gap lies in the root network, where the difference both in the macro-averaged and the micro-averaged F_1-score is very large. At this first stage, the sub-classes are still a combination of many leaf-classes. Without following the biological background knowledge by grouping similar classes together, the intra-class variance is increased, which appears to make classification more difficult. The same effect can be observed for the network placed at the granulopoiesis node, which, apart from the root, has the largest child classes in terms of the number of combined leaf-classes. Networks which are closer to the leaves generally have less heterogeneous child-classes. For these networks, the random interchanges of the leaves often lead to less similar – therefore easier distinguishable – classes, which result in very good test scores of these individual networks.

The overall scores of these two cascades show that error propagation is an important mechanism in a cascade. Generally, it is very difficult for networks to overrule their ancestors' decisions even with the probabilistic soft propagation – in hard propagation, it is completely impossible. This is especially the case if the networks are overconfident, which is commonly the case for deep networks [54].

Any hierarchy that deviates from the haematological lineages, by artificially grouping the cells based on single features or on clustering of either the confusion matrix or a similarity score given by experts, does not lead to equally good results. This suggests that the biological background directly provides the most suitable cell groupings. The fact that the performance gap compared to flat networks is larger in terms of the macro-averaged scores than for the micro-averaged scores shows that the main reason of this difference lies in the performance on the underrepresented classes. Potentially, larger sub-classes of a combined inner class outweigh the smaller

classes. If on the other hand more similar classes are generally combined, the smaller classes are easier to classify within early levels, simply because they are grouped together with the most similar cells. This assumption is supported by the fact that – as an exception – the balanced hierarchy reaches performances relatively close to the reference. This cannot be explained solely from the hierarchy itself, although closely related classes often have similar sizes, but from the fact that all classifiers in this cascade handle classes of roughly similar sizes. By partly compensating for the class imbalances of the dataset, it is less likely that one class outweighs another class.

Comparison of the Individual Network Architectures

The comparison of base learners, either ResNet-18 or DenseNet-121, does not reveal major advantages of one architecture over the other within a cascade. In some experiments, one may achieve higher scores than the other, but on average, the performances are very similar – and much more dependent on the hierarchy itself. This is clearly visible in Figure 5.2 and in the comparison of the network-wise scores of a full cascade in Table 5.5. Notably, the individual scores of DenseNet-121 are slightly higher, while the cascade with ResNet-18 still achieves the higher overall performances. In soft propagation, definitive decisions are not made at the individual networks, but are determined from the overall confidences. These confidences are not accounted for by the individual scores, as well as other effects, such as the influence of samples that are not from a child class of a network. Which architecture leads to higher overconfidences, and how this affects the performances of the cascades, requires further investigation.

5.3 Comparison of Training Strategies

In Chapter 4.3, two different training strategies for hierarchical cascades have been introduced. In the following, separate training of the individual networks is compared to end-to-end training of the entire cascade at once.

Experimental Setup

The two training procedures are employed on the full (Section 4.4.1) and the ultrametric (Section 4.4.2) hierarchy cascade, which have achieved the highest performances in the previous experiment. End-to-end training requires significantly more GPU-memory compared to individual training, because gradients within all networks have to be stored at once. For this reason, only ResNet-18 is selected as the base architecture due to lower memory requirements compared to DenseNet-121. For the end-to-end training, either the cross-entropy loss or the tree loss as defined in 4.3.2 is applied. Regardless of the cascade, the tree loss is calculated from the distances in the actual haematological cell hierarchy. The tree loss introduces weights to the cross-entropy components that are, on average, larger than one. For evaluation of their effect, the experiments using the standard cross-entropy loss are

performed twice, once with the standard learning rate of $\alpha = 1 \cdot 10^{-5}$ and once with an increased learning rate $\alpha = 5 \cdot 10^{-5}$. Because hard propagation does not guarantee well defined gradients, probabilistic (soft) propagation is performed within the cascades both during training and test time.

Results

Table 5.6 lists the performances of individual training and end-to-end training for cascades in form of the full and the ultrametric hierarchy. Additionally, the macro-averaged F-scores are visualised in Figure 5.4. For all four scores and both hierarchies, the individual training outperforms end-to-end training. The differences are especially pronounced for the macro-averaged scores.

	Training	Loss	$\frac{\alpha}{10^{-5}}$	$F_{1,\text{macro}}$ [%] mean ± std	$F_{1,\text{micro}}$ [%] mean ± std	$hF_{1,\text{macro}}$ [%] mean ± std	$hF_{1,\text{micro}}$ [%] mean ± std
Full	separate	CE	1	**70.3** ± 2.2	**77.1** ± 1.2	**89.1** ± 1.2	**94.0** ± 0.5
	end-to-end	CE	1	66.4 ± 3.8	76.9 ± 1.7	84.7 ± 3.7	93.9 ± 0.7
	end-to-end	CE	5	64.0 ± 4.3	76.9 ± 2.1	79.9 ± 3.4	93.7 ± 0.5
	end-to-end	tree	1	60.1 ± 1.2	75.1 ± 1.0	75.7 ± 0.7	93.1 ± 0.4
Ultrametric	separate	CE	1	**70.6** ± 2.3	**77.4** ± 0.9	**89.3** ± 1.3	**94.0** ± 0.4
	end-to-end	CE	1	67.2 ± 3.7	76.6 ± 1.3	85.3 ± 2.8	93.8 ± 0.4
	end-to-end	CE	5	62.7 ± 3.3	77.3 ± 1.9	77.5 ± 2.7	93.9 ± 0.6
	end-to-end	tree	1	62.7 ± 2.6	75.5 ± 1.8	80.0 ± 1.0	93.4 ± 0.5

Table 5.6: Performance of different training strategies, separated by the hierarchy of the cascade.

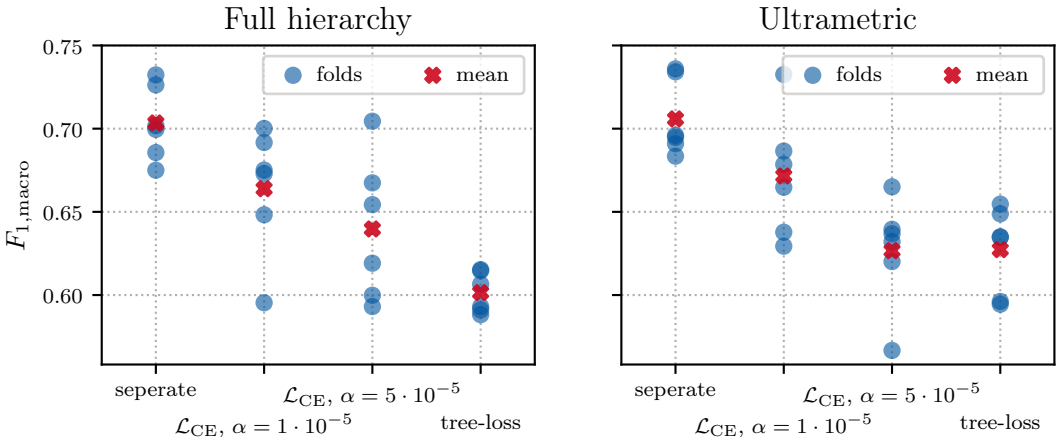


Figure 5.4: Scatter plot of the test results for the different training strategies.

Figure 5.5 displays the development of the macro-averaged validation-F_1-score and the training and validation losses for the ultrametric hierarchical cascade. Similar plots for the full hierarchy can be found in Appendix A.4. Consistent with the test

result, the validation score is very similar to the score of the cross-entropy loss with $\alpha = 5 \cdot 10^{-5}$ and, when converged, slightly lower than with the lower learning rate. Initially, the tree loss is considerably higher than the unweighted cross-entropy loss. During training, the slope of the tree loss is much larger than for the cross-entropy loss, as is the variability of the loss. While the initial validation loss is larger for the tree loss, it converges to a similar mean level as the cross-entropy losses, albeit with a much higher variability.

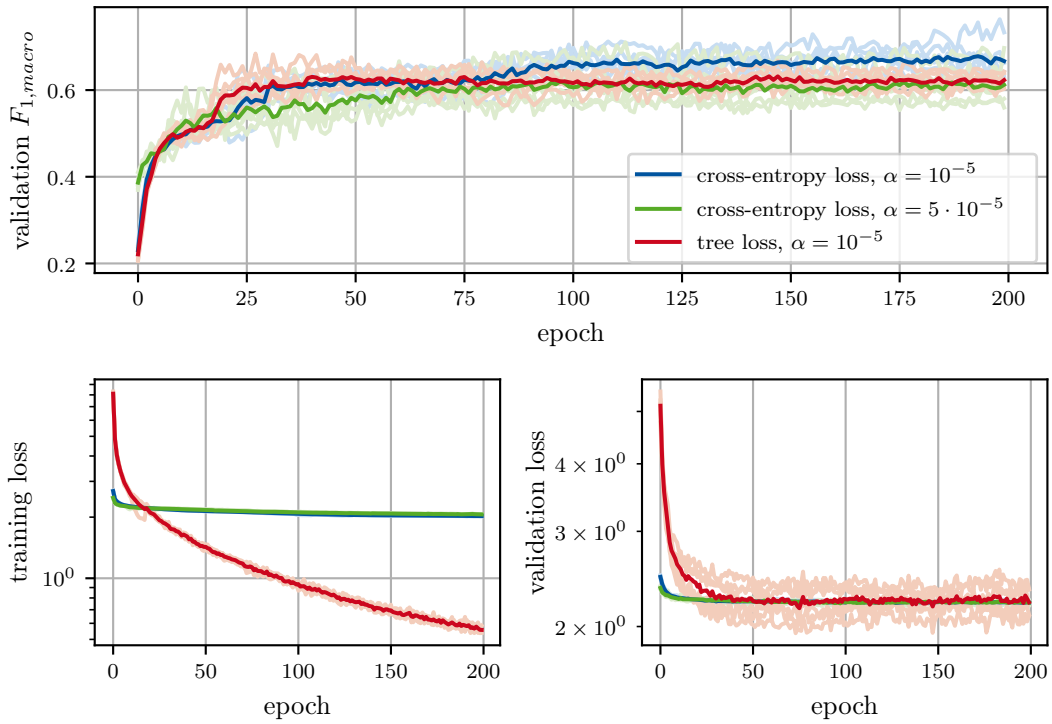


Figure 5.5: Development of the validation-$F_{1,\text{macro}}$-score and the different training and validation losses for end-to-end training of the ultrametric cascade. The highlighted lines are the respective averages over all six folds.

Discussion

The results of this experiment show that individual training of the networks clearly outperforms end-to-end training of the entire cascade at once. This is especially true for the macro-averaged scores, which indicates that the individually trained networks can handle underrepresented classes better.

In end-to-end training, the different loss functions and learning rates have an influence on the performance. Although it accounts for the cell hierarchy, the hierarchical loss does not generally improve the results, which agrees with conclusions

Wu et al. [30] have drawn for their own hierarchical loss. From the development of the loss during training (Figure 5.5), some insight into the reasons can be deducted. Firstly, the additional hierarchical weights cause the initial loss to be much larger than the plain cross-entropy loss. This leads to large gradients, which might cause overshoot during training. While this could be compensated by either a constant factor to the loss or a reduced learning rate, the weights also lead to higher variability in the loss. This variability can be explained from the different weight factors that get introduced as soon as one or more samples are misclassified with a relatively distant class. Even though these factors are weighted by the confidence of the network, such false predictions have a strong influence as soon as the networks are overconfident. To reduce this problem, it might be beneficial to drastically increase the batch size to average over more samples at once, however, this has practical limitations, and requires further investigation.

Regardless of the loss, individual training is preferable, especially for unbalanced datasets. Apart from the higher performance scores, it provides other benefits such as a higher adaptability of the cascades. Furthermore, separate training has significantly lower computational requirements, because only the gradients of a single network need to be calculated and stored, instead of having to adjust all weights in all networks in one step. Based on the results of this experiment, individual training is the method of choice in all subsequent experiments.

5.4 Comparison of Deterministic and Probabilistic Propagation

After the different hierarchies have been compared using soft propagation in Section 5.2, its deterministic alternative is examined in the following experiment.

Experimental Setup

For the first experiment (Section 5.2), the networks in the nine different cascades have been trained separately. Therefore, these networks are re-used for this experiment, since the propagation method has no influence on the networks themselve. Consequently, for a comparison of the two prediction methods, only the testing phase needs to be repeated, this time using hard instead of soft propagation.

Results

The results for probabilistic traversal have already been presented in Section 5.2. The scores for hard propagation are given in Table 5.7.

Performance differences between the two strategies are marginal, as visualised in Figure 5.6. Averaged over all 18 experiments – nine different hierarchies each with ResNet-18 and DenseNet-121 as base architecture – the probabilistic traversal leads to approximately 0.05 % higher macro-averaged F-scores. In 14 cases, the

		$F_{1,\text{macro}}$ [%]	$F_{1,\text{micro}}$ [%]	$hF_{1,\text{macro}}$ [%]	$hF_{1,\text{micro}}$ [%]
		mean ± std	mean ± std	mean ± std	mean ± std
Full	RN-18	70.2 ± 2.1	77.1 ± 1.2	89.1 ± 1.1	94.0 ± 0.5
	DN-121	70.2 ± 2.1	77.4 ± 1.0	89.3 ± 0.9	94.0 ± 0.4
Ultrametric	RN-18	70.8 ± 2.7	77.5 ± 1.0	89.4 ± 1.4	94.0 ± 0.4
	DN-121	70.1 ± 4.6	77.4 ± 1.9	88.5 ± 2.9	93.9 ± 0.6
Cytoplasm features	RN-18	69.1 ± 2.6	76.6 ± 0.9	87.7 ± 2.2	93.5 ± 0.5
	DN-121	66.2 ± 2.4	77.5 ± 1.6	84.6 ± 1.8	93.9 ± 0.7
Ratio/shape features	RN-18	66.0 ± 4.1	76.1 ± 1.5	84.6 ± 5.0	93.3 ± 0.6
	DN-121	63.7 ± 2.9	76.0 ± 1.4	82.7 ± 3.0	93.4 ± 0.4
Nucleus features	RN-18	65.5 ± 3.5	75.2 ± 1.6	84.8 ± 3.3	93.2 ± 0.5
	DN-121	65.0 ± 2.3	76.1 ± 1.6	83.9 ± 2.6	93.5 ± 0.5
Confusion clustering	RN-18	66.2 ± 1.1	76.1 ± 1.0	82.5 ± 1.6	93.6 ± 0.4
	DN-121	65.8 ± 0.8	77.2 ± 1.3	81.6 ± 1.8	93.8 ± 0.4
Similarity clustering	RN-18	64.3 ± 1.9	76.4 ± 1.1	83.5 ± 0.7	93.6 ± 0.5
	DN-121	66.9 ± 2.2	76.9 ± 1.5	85.0 ± 1.3	93.9 ± 0.5
Balanced	RN-18	69.1 ± 3.5	77.2 ± 1.6	88.5 ± 1.5	93.9 ± 0.5
	DN-121	70.5 ± 3.2	78.1 ± 1.8	88.7 ± 1.4	94.1 ± 0.7
Random	RN-18	62.3 ± 5.1	74.8 ± 1.1	81.6 ± 5.2	93.1 ± 0.5
	DN-121	67.1 ± 1.4	76.1 ± 1.5	87.5 ± 0.7	93.5 ± 0.5

Table 5.7: Scores for hard, deterministic cascade propagation.

probabilistic propagation performs better in terms of $F_{1,\text{macro}}$, by an absolute increase of at most 0.3 %, while the deterministic method achieves a higher score in four cases, with a margin up to 0.2 %. For the hierarchical macro-averaged F-score, the average difference is below 0.1 %.

Discussion

Even though, on average, the probabilistic traversal attains marginally higher non-hierarchical F-scores than hard propagation, the differences are negligible.

A possible explanation for this observation lies in the overconfidence of deep networks with softmax activation [54]. Usually, one component of the output from the softmax function is very close to one, while all others are nearly zero. Consequently, it is very unlikely for a later network to overrule a predecessor.This means that, given overconfident networks, the two strategies are practically equivalent with respect to the overall outcome when performing classification. It also indicates a high dependency on the performances of early networks when aiming to improve hierarchical evaluation measures, because later networks are rarely able to outweigh the confidences of their ancestors. An error in the output of an early network most likely leads to a hierarchically very distant misclassification – in hard and soft propagation alike. For this reason, the difference of the results might become much more pronounced if the softmax function is replaced or extended in the future to achieve more accurate confidence values. However, this requires further investigation.

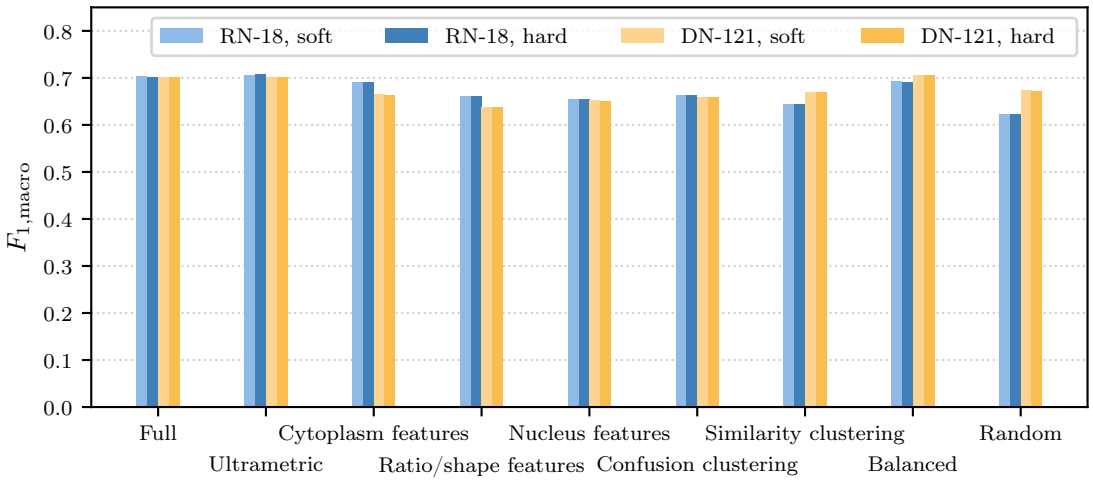


Figure 5.6: Mean macro-averaged F-scores with hard and soft propagation, compared for the different hierarchies with cascades of ResNet-18 and DenseNet-121 networks.

5.5 Regressors

A cascade of networks allows for more diversity of network types. For the following experiment, some classifiers are replaced by regressors to estimate the maturation of neutrophilic and eosinophilic granulocytes. The goal of this experiment is to evaluate the output of such regressors compared to classifiers individually as well as when embedded into a cascade of networks.

Experimental Setup

For this experiment, only the full hierarchy cascade (Section 4.4.1) is examined, because it contains specialised networks for neutrophilic and eosinophilic granulocytes, respectively. These two classifiers are replaced by regressors with the same base architecture to estimate the maturation of the samples. Consequently, the only architectural difference compared to previous experiments is the output linear layer – in regressors only consisting of a single neuron with one output – and the lack of an output activation function. The networks, pretrained on ImageNet, are trained using targets as discussed in Section 4.5, ranging from zero for the most immature to four for the most mature stage. Mean squared error (MSE) is applied as the loss function of the regressors. In context of the whole cascade, the regression output is transformed into classification by thresholding, as described in Section 4.5. The same transformation is also performed during validation. Because the regressors do not provide probabilistic confidences, the overall predictions are obtained by employing the hard propagation method. As for the classifiers, the model state adopted for testing of the regressors is selected based on the highest macro-averaged F_1-score reached on the validation set.

Results

		$F_{1,\text{macro}}$ [%]	$F_{1,\text{micro}}$ [%]	$hF_{1,\text{macro}}$ [%]	$hF_{1,\text{micro}}$ [%]
		mean ± std	mean ± std	mean ± std	mean ± std
Classification	RN-18	70.2 ± 2.1	77.1 ± 1.2	89.1 ± 1.1	94.0 ± 0.5
	DN-121	70.2 ± 2.1	77.4 ± 1.0	89.3 ± 0.9	94.0 ± 0.4
Regression	RN-18	69.3 ± 2.2	76.2 ± 1.6	88.8 ± 1.2	93.7 ± 0.7
	DN-121	68.9 ± 2.4	76.7 ± 1.1	88.7 ± 1.1	93.9 ± 0.4

Table 5.8: Overall cascade performances with regression compared to classification when using deterministic cascade propagation in the full hierarchy.

Results of the entire cascades with regression and with classification with deterministic cascade propagation are shown in Table 5.8. With regression, all performance measures are slightly lower than with classification. For further inspection, the individual scores of the relevant networks which have been replaced by regressors are examined. Neither the macro-averaged, nor the micro-averaged F_1-score or any of the hierarchical scores previously defined distinguishes between close or more distant misclassifications if the classes are siblings in the hierarchy. For this purpose, the frequency that cells are classified neither correctly nor as a direct predecessor or successor of their true maturity stage is observed. Table 5.9 lists both F_1-scores, as well as this *non-neighbour-rate (NNR)* only of the regressors, compared to the corresponding classifiers.

Lineage	Architecture	Type	$F_{1,\text{macro}}$ [%]	$F_{1,\text{micro}}$ [%]	NNR
			mean ± std	mean ± std	%
neutrophilic	ResNet-18	classification	78.6 ± 1.6	81.5 ± 1.1	1.86
		regression	76.7 ± 1.4	80.0 ± 1.0	0.73
	DenseNet-121	classification	79.3 ± 0.9	82.4 ± 0.8	1.40
		regression	78.7 ± 0.7	81.4 ± 0.8	0.61
eosinophilic	ResNet-18	classification	78.1 ± 7.9	79.7 ± 8.3	-
		regression	76.7 ± 4.8	78.9 ± 5.4	-
	DenseNet-121	classification	77.0 ± 4.9	78.6 ± 5.4	-
		regression	78.9 ± 6.7	80.6 ± 6.8	-

Table 5.9: Performance comparison of the individual regression and classification networks for neutrophilic and eosinophilic granulocytes. As an additional measure of closeness of predicted classes to the true labels, the non-neighbour-rate (NNR) is given.

The F-scores are very close, although in three out of four cases the scores are slightly higher for classification. However, there is a considerable difference in the non-neighbour-rates. These are also listed in Table 5.9 for the neutrophilic granulocytes. For all approaches, the non-neighbour-rate is below 2 %. Compared to unordered classification, it is more than halved for both architectures when using regression. This is also visualised by the difference confusion matrix shown in Figure

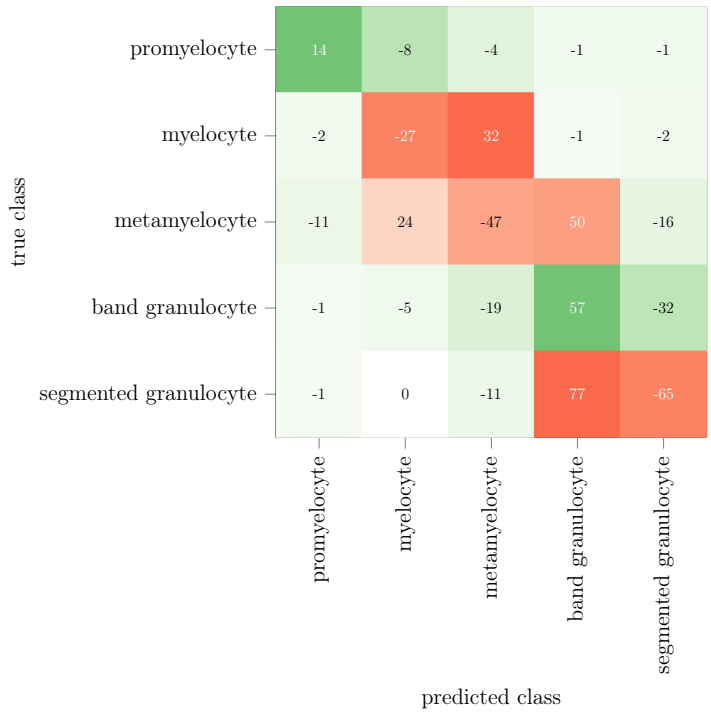


Figure 5.7: Difference of the confusion matrices for regression and classification on neutrophilic granulocytes. Positive values indicate more predictions from the regressor, negative values indicate more predictions from the classifier.

5.7. For eosinophilic granulocytes with only two classes, the non-neighbour-rate is not a meaningful measure.

Figure 5.8 shows the outputs y of a regressor compared to the corresponding correct target labels t in form of box plots. The targets are ordered by maturity, starting from 0 for neutrophilic promyelocytes or immature eosinophilic granulocytes up to 4 for segmented neutrophilic granulocytes and mature eosinophilic granulocytes.

For neutrophilic granulocytes, all boxes completely lie within the correct decision intervals and in most cases are nearly centred therein. All values lie within the range between -0.5 and 4.5, therefore the distance to the closest target never exceeds 0.5. The largest shifts with respect to the true label are observed for metamyelocytes (label 2) and segmented granulocytes (label 4), which both are slightly shifted towards band granulocytes (label 3). The boxes have different sizes, especially for promyelocytes and – to a lesser extent – segmented granulocytes the distributions are very narrow. Outliers only relatively rarely lie in non-neighbouring decision intervals.

For the eosinophilic granulocytes, the boxes are larger, as the output values are more spread. Futhermore, the output of the regressor is clearly biased towards the centre for both classes, such that the boxes are not centred on the target values.

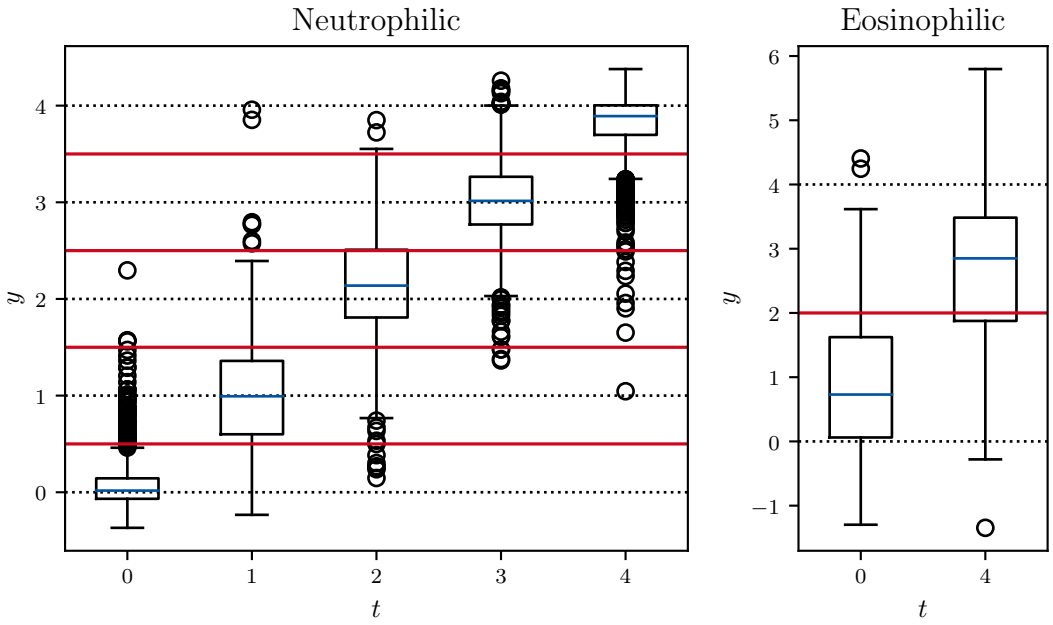


Figure 5.8: Box plots of the ResNet-18 regressor test outputs y for neutrophilic and eosinophilic granulocytes depending on the true class label t. Target values are ordered by increasing maturity.

Discussion

The classification performance of the regressors is slightly inferior to classifiers in terms of the classical evaluation scores. However, although the relative amount of misclassifications into non-neighbouring classes is very low already for classification, regression shows an even smaller number of distant misclassifications. This is illustrated by the difference confusion matrix in Figure 5.7. While in total there are 68 more false predictions for regression – calculated from the sum of values outside the main diagonal – all cases where there are more misclassifications with regression lie on the secondary diagonals. This can be explained from the fact that regression incorporates the ordinal character of the classes, and that the mean squared error penalises the distances of the output values to the targets quadratically instead of treating every false prediction equally. Therefore, even if for example $F_{1,\text{macro}}$ is similar or even slightly higher for classification, regression might be the more appropriate approach, depending on the particular use case and requirements to the classifier. Additionally, it provides an actual maturity measure, which goes beyond class membership. The cascading allows this measure to be combined with classification for all classes, where regression is not suitable. On the other hand, a disadvantage of using regression in the context of a whole cascade of networks is the lack of a confidence measure. This essentially limits the passage through the cascade to the greedy, deterministic propagation method, unless some additional confidence measure is introduced in the future.

The output values for neutrophilic promyelocytes are very narrowly distributed around the correct label. This indicates that, for the regressor, a distinction to myelocytes is relatively easy and sharp. This may for example be caused by the different cytoplasm colour – basophilic (blue) in promyelocytes and pink in myelocytes [7] – which allows a good distinction between these two classes. A relatively narrow distribution is also observed for cells of the class of segmented granulocytes, which lie on the other end of the scale. A possible reason for the slight shift of the output of both segmented granulocytes and metamyelocytes towards band granulocytes is that between these three classes there are no cell divisions, but the changes happen gradually. The differences between these three classes therefore are less clear and based solely on definitions, for example by the ratio of the narrowest to the widest section of the nucleus. Therefore, there often might be samples where it is particularly difficult to exactly determine the correct class membership. This might be true not only for the regressors, but also for the annotations of the cells by the medical experts. Especially given only the small performance differences between classification and regression, it is not clear how much influence potentially incorrect annotations in borderline cases might have. This requires further investigation in the form of an inter-rater study to further validate the existing annotations. The regression value can potentially provide a measure of this uncertainty for the individual samples, as it assigns not only a fixed class label, but also a measure of closeness to the other classes.

For eosinophilic granulocytes, the analysis is more difficult than for neutrophilic granulocytes, because there are significantly fewer samples in the dataset. Additionally, the classes are more diverse, because of the culmination of different classes into only two, more general classes. Both these factors may contribute to the lower classification scores as well as the larger spread of the output values, including the bias towards the decision boundary.

5.6 Weight Initialisation

So far, all networks have been initialised from weights of networks trained on the ImageNet dataset. In this experiment, this approach is compared to pretraining on the haematological dataset.

Experimental Setup

In this experiment, the weights of all individual networks are initialised from a flat network trained on the haematology dataset, instead of the ImageNet dataset. The experiment is conducted for the two best-performing cascades, and additionally for the feature-based cascade based on nucleus features. For initialisation, the weights from reference networks are adapted, which have been trained for 200 epochs with the same basic training parameters as listed in Section 5.1.1. Importantly, the cross-validation splits, as described in Section 5.1.2, for this initialisation network training are assured to be consistent with the subsets used for cascade training.

Apart from the changes to transfer learning, the setup is kept as basic as possible. The initialised individual classification networks are trained separately for 200 more epochs – with weight updates both for the fully connected output layers as well as for the initialised convolutional layers. During the testing phase, soft propagation is performed to determine the predictions.

Results

Table 5.10 lists the performance scores of the pretrained cascades compared to the cascades trained based on ImageNet. The macro-averaged F_1-scores are visualised in Figure 5.9. For all three hierarchies, the cascade starting from networks pretrained on the haematology dataset show increases of all performance measures. Regarding the macro-averaged F_1-score, an average absolute improvement of approximately 2.3 % is achieved, for the micro-averaged F-score the average improvement is 0.8 %. For the hierarchical scores, the improvements are slightly smaller than for their non-hierarchical counterparts. With pretraining on the haematology dataset, $hF_{1,\text{macro}}$ is increased by 1.4 % on average, $hF_{1,\text{micro}}$ by 0.2 %. The biggest relative and absolute improvements are achieved for the feature-based hierarchy, however, the resulting scores remain slightly lower to the other two hierarchies, both with and without pretraining on the haematology dataset.

		$F_{1,\text{macro}}$ [%]	$F_{1,\text{micro}}$ [%]	$hF_{1,\text{macro}}$ [%]	$hF_{1,\text{micro}}$ [%]
		mean ± std	mean ± std	mean ± std	mean ± std
Full	RN-18	70.3 ± 2.2	77.1 ± 1.2	89.1 ± 1.2	94.0 ± 0.5
	RN-18, pre.	71.6 ± 3.5	77.6 ± 1.6	89.6 ± 1.8	94.1 ± 0.6
	DN-121	70.1 ± 2.1	77.4 ± 1.0	89.2 ± 0.9	94.0 ± 0.4
	DN-121, pre.	71.1 ± 2.9	77.8 ± 1.7	89.2 ± 1.6	94.1 ± 0.5
Ultrametric	RN-18	70.6 ± 2.3	77.4 ± 0.9	89.3 ± 1.3	94.0 ± 0.4
	RN-18, pre.	71.8 ± 3.0	77.4 ± 1.6	89.5 ± 1.7	94.0 ± 0.6
	DN-121	70.1 ± 4.4	77.5 ± 1.9	88.5 ± 2.8	94.0 ± 0.6
	DN-121, pre.	71.5 ± 3.7	78.1 ± 1.3	88.8 ± 2.5	94.2 ± 0.5
Nucleus features	RN-18	65.4 ± 3.6	75.2 ± 1.6	84.7 ± 3.4	93.2 ± 0.5
	RN-18, pre.	70.1 ± 4.1	76.9 ± 1.2	88.1 ± 2.8	93.7 ± 0.5
	DN-121	65.1 ± 2.3	76.1 ± 1.5	83.9 ± 2.6	93.5 ± 0.5
	DN-121, pre.	69.5 ± 3.2	77.6 ± 1.8	87.8 ± 2.2	93.9 ± 0.7

Table 5.10: Results with (pre.) and without pretraining on the haematological dataset for three different cascades, the full hierarchy, the ultrametric hierarchy and the hierarchy based on nucleus features.

Discussion

Pretraining on the haematology dataset consistently leads to improvements compared to transfer learning based on the ImageNet dataset. The increase of especially the macro-averaged scores indicates that a considerable gain is achieved for under-represented classes. The greatest improvement is attained for the feature-based

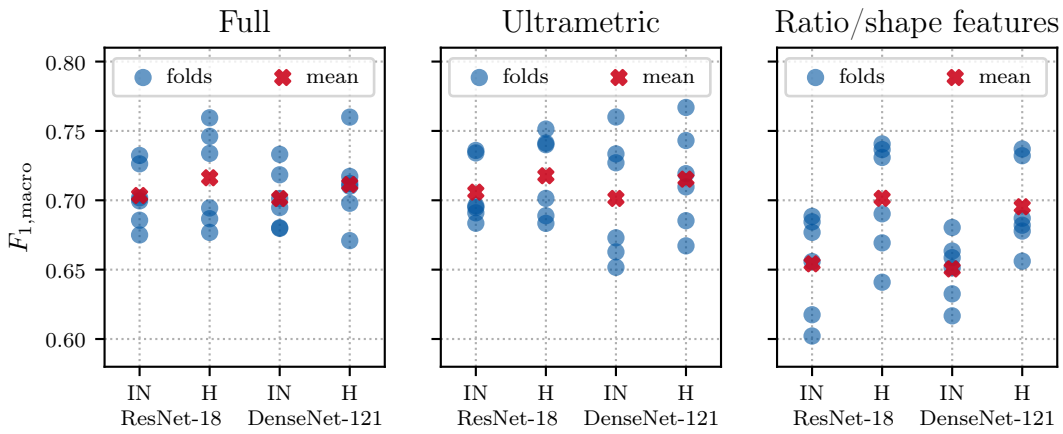


Figure 5.9: Scatter plot of the test results for pretraining on ImageNet (IN) the haematological dataset (H).

hierarchy that performed worse than the others when pretrained on ImageNet. This indicates that it is difficult to learn appropriate features if the cell types are grouped as in this hierarchy. It underlines that even with pretraining on the target dataset, the hierarchies remain an important factor influencing the classification performances.

The major disadvantage of the pretraining is that it already requires the same time and resources as the whole training of a flat network of the same architecture. On the other hand, this additional effort is profitable, as the performances of the best-performing cascades with pretraining exceed those of flat reference networks with the same base architecture (compare Table 5.2). Equal gains for continued training of the flat networks alone are questionable, as the validation scores of these networks converge relatively early already during the initial training (see Figure A.2 in Appendix A.5).

5.7 Loss Weights and Oversampling

In this experiment, the effects of the methods addressing the class imbalance on hierarchical cascades are evaluated when applied to cascades as well as flat networks.

Experimental Setup

In the first part of the experiment, the individual networks are trained not with the standard, unweighted cross-entropy loss, but with weighted cross-entropy loss (WCE). Loss weights are calculated for each network in the cascade separately. The experiment is performed for the exponents

$$\delta \in \{0,\ 0.5,\ 1,\ 2\}, \tag{5.5}$$

where $\delta = 0$ is the standard, unweighted case.

In the second part of the experiment, oversampling of underrepresented classes is performed. The upsampling is employed individually to each network of the cascades with oversampling exponents of

$$\nu \in \{0,\ 0.5\}. \tag{5.6}$$

This means that for every network the classes are either not oversampled, or oversampled by the square root of the relative frequency compared to the most frequent class handled by that network.

In both parts of the experiment, all networks are classifiers pretrained on the ImageNet dataset, and soft propagation is performed to determine the predictions of the cascades. As the cascades, the full hierarchy directly derived from the cellular relationships and the ultrametric hierarchy are examined. For comparison, WCE and oversampling are also employed on a flat DenseNet-121.

Results

The performances with WCE and oversampling listed in Table 5.11 and are visualised in Figure 5.10. For the micro-averaged F-score and the micro-averaged hierarchical F-score, the highest results are consistently achieved for the unweighted case ($\delta = 0$). Both the non-hierarchical and the hierarchical macro-averaged scores show a slight increase for $\delta = 0.5$ compared to $\delta = 0$, however, this is not the case for the ultrametric hierarchical cascade. For the most extreme loss weights ($\delta = 2$), the macro-averaged scores are much lower than for the unweighted case.

			$F_{1,\text{macro}}$ [%]	$F_{1,\text{micro}}$ [%]	$hF_{1,\text{macro}}$ [%]	$hF_{1,\text{micro}}$ [%]
	δ	ν	mean ± std	mean ± std	mean ± std	mean ± std
Flat	0	0	70.7 ± 3.1	77.9 ± 1.7	88.9 ± 1.7	94.1 ± 0.6
	0.5	0	71.5 ± 2.5	77.9 ± 1.7	89.4 ± 1.4	94.1 ± 0.5
	1	0	70.3 ± 2.3	76.5 ± 1.3	88.9 ± 1.2	93.7 ± 0.6
	2	0	70.0 ± 2.3	75.5 ± 1.9	88.7 ± 1.4	93.4 ± 0.6
	0	0.5	72.3 ± 2.7	77.8 ± 1.2	89.7 ± 1.4	94.0 ± 0.5
Full	0	0	70.3 ± 2.2	77.1 ± 1.2	89.1 ± 1.2	94.0 ± 0.5
	0.5	0	70.9 ± 3.1	76.8 ± 1.4	89.4 ± 1.6	93.8 ± 0.6
	1	0	69.4 ± 3.4	75.7 ± 1.9	88.5 ± 1.6	93.4 ± 0.6
	2	0	66.9 ± 3.2	72.6 ± 1.8	87.5 ± 1.6	91.5 ± 0.9
	0	0.5	70.9 ± 3.0	76.8 ± 1.6	89.4 ± 1.5	93.8 ± 0.6
Ultrametric	0	0	70.6 ± 2.3	77.4 ± 0.9	89.3 ± 1.3	94.0 ± 0.4
	0.5	0	69.6 ± 2.5	76.3 ± 1.1	89.1 ± 1.2	93.7 ± 0.4
	1	0	70.1 ± 3.6	76.3 ± 1.7	89.0 ± 1.9	93.7 ± 0.7
	2	0	63.3 ± 4.4	70.3 ± 2.1	85.5 ± 2.9	90.6 ± 0.9
	0	0.5	69.6 ± 2.1	76.9 ± 0.9	88.8 ± 1.1	93.8 ± 0.5

Table 5.11: Performances of flat DenseNet-121 and the cascades with ResNet-18 as base architecture for different loss weights.

Consistently, oversampling is correlated with decreased micro-averaged performance scores. Such a decrease is also observed in the macro-averaged scores for the ultrametric cascade. However, for the full hierarchy cascade and especially for the flat

DenseNet-121 classifier, the oversampling leads to increases of both macro-averaged scores.

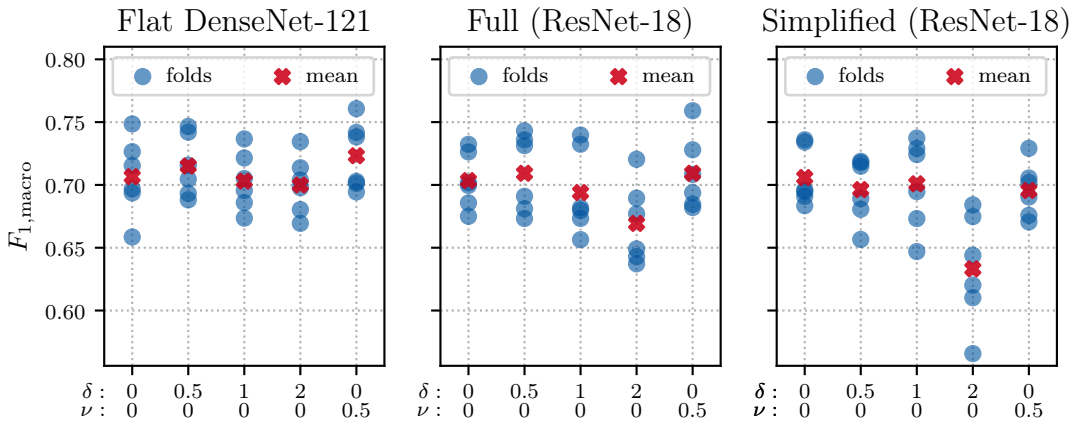


Figure 5.10: Scatter plots of the test results with loss weights and oversampling in a flat DenseNet-121 and the full and ultrametric hierarchy cascades with ResNet-18 as base architecture.

Discussion

The effects of both loss weights and oversampling to compensate for class imbalances are much stronger for the flat classifier than for the cascades. For the weighted cross-entropy loss, the improvements of the macro-averaged scores are the largest for the most moderate weights ($\delta = 0.5$). The oversampling only considerably improved the macro-averaged performance of the flat DenseNet-121, but has a much smaller effect on the cascades. This is a small deviation from results from previous investigations [3], which found that oversampling only led to overfitting without improving performance of a flat network. Whether this improvement is caused by the inclusion of more classes, or by the more moderate oversampling – in [3], effectively $\nu = 1$ was employed – is beyond the scope of this work and requires further investigation. For the hierarchical, cascaded approach, the results show that these two methods, if at all, can improve the performances only by a small margin, and should be employed carefully.

5.8 Shared Blocks for Feature Extraction

The goal of the following experiment is to investigate the effects of sharing a certain amount of convolutional layers as a common feature extraction network, as described in Section 4.7.

Experimental Setup

For this experiment, the full hierarchy is examined, because it has the largest number of networks, so a potential difference is expected to have the biggest effect. The shared, fully convolutional feature extractor network is not further adjusted during training, as motivated in Section 4.7. For this reason, both the shared and the individual networks are pretrained on the haematology dataset.

Results

The results for block sharing based on the full hierarchy are presented in Table 5.12. For ResNet-18 as the base architecture, the differences between the scores are marginal. On the other hand, only the scores for no shared blocks and one shared block in DenseNet-121 are very similar, but subsequently decrease when more blocks are shared. As visualised in Figure 5.11, there is a very small improvement in the macro-averaged, non-hierarchical F-score when sharing one block compared to completely separate networks.

		$F_{1,\text{macro}}$ [%]	$F_{1,\text{micro}}$ [%]	$hF_{1,\text{macro}}$ [%]	$hF_{1,\text{micro}}$ [%]
		mean ± std	mean ± std	mean ± std	mean ± std
ResNet-18	0 blocks shared	71.6 ± 3.5	77.6 ± 1.6	89.6 ± 1.8	94.1 ± 0.6
	1 blocks shared	71.9 ± 2.8	77.5 ± 1.2	89.4 ± 1.3	94.0 ± 0.5
	2 blocks shared	71.7 ± 2.6	77.3 ± 1.2	89.4 ± 1.2	94.0 ± 0.5
	3 blocks shared	71.4 ± 3.4	77.4 ± 1.1	89.3 ± 1.6	94.1 ± 0.5
DenseNet-121	0 blocks shared	71.1 ± 2.9	77.8 ± 1.7	89.2 ± 1.6	94.1 ± 0.5
	1 blocks shared	71.3 ± 3.4	77.8 ± 1.1	89.4 ± 1.8	94.1 ± 0.6
	2 blocks shared	69.8 ± 3.1	76.8 ± 1.7	88.8 ± 1.7	93.8 ± 0.6
	3 blocks shared	66.9 ± 4.4	75.9 ± 1.4	86.7 ± 3.1	93.3 ± 0.6

Table 5.12: Results of block sharing with the full hierarchy for both base architectures.

Discussion

The results show that the classification performance of a hierarchy with ResNet-18 as base learners is not strongly affected by the block sharing. For DenseNet-121 on the other hand, fixing two or more blocks considerably reduces both the macro-averaged as well as the micro-averaged F-score and the hierarchical scores. This can be explained from the number of parameters in the respective blocks of the two architectures (compare Table 4.2). In ResNet-18, the fourth block contains approximately 75 % of the learnable parameters, therefore, even sharing the first three blocks means that only a quarter of the parameters are fixed. For DenseNet-121 on the other hand, the parameters are more evenly distributed among the blocks. Therefore, a much larger relative amount of features is shared – and consequently fixed – than in its ResNet counterpart. In DenseNet-121, sharing three blocks leaves only approximately 30% of the parameters to be trained individually.

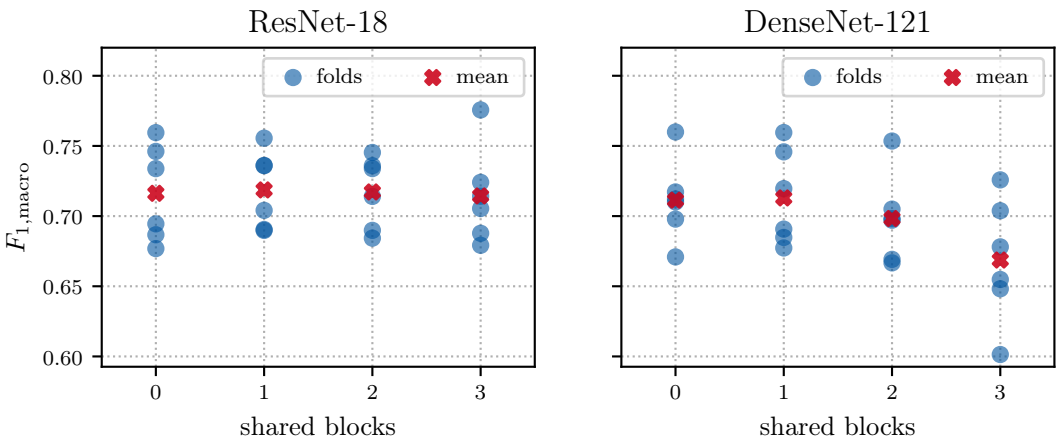


Figure 5.11: Scatter plot of the test results with a shared feature extractor.

The first convolutional layers cover relatively basic features such as edges. As the results show, sharing these layers and not training them further for the specific subtasks does not decrease the performance, which indicates that these basic features are relatively universal. On the other hand, the more advanced features in the second and third block profit from individual training, as can be seen especially for DenseNet-121. This indicates that in a cascade, the networks do learn specialised features suitable for the specific task within the hierarchy, instead of relying only on the general features which are provided by the pretrained, fixed and shared layers. Consequently, the amount of shared convolutional layers should be selected carefully to preserve the abilities of the networks to learn appropriate features for their individual tasks. Sharing only a low number of parameters, however, also entails that the improvements in training and general prediction speed are negligible.

5.9 Feature Forwarding

In the last experiment, the cascades are extended by feature forwarding, as introduced in Section 4.9. The main goals of this experiment are not only to evaluate potential effects on the classification performance of the cascades, but also to examine the use of the embeddings for model visualisation.

Experimental Setup

This experiment is conducted for the two best-performing cascades, the full and ultrametric hierarchy. The networks within the cascades are extended by an additional fully connected layer to obtain embedding vectors with $n_{\text{emb}} = 100$ elements from the output of the convolutional parts of the networks. Both ResNet-18 and DenseNet-121 are examined as the underlying architecture of the networks within the cascade. The individual networks are pretrained on the haematology dataset. As described in more detail in Section 4.9, the training of the networks is performed

separately but in order as given by the hierarchy, starting from the root. During test time, soft traversal is performed to obtain the final predictions, extended by the successive concatenation of the embedding vectors introduced by feature forwarding. In additional runs, the feature forwarding is applied together with moderate loss weights with $\delta = 0.5$ or oversampling with an exponent of $\nu = 0.5$.

Apart from the performance aspects, one goal of this experiment is to employ the embedding vectors for visualisation of the learnt feature spaces. Unsupervised UMAP [43] can be utilised to visualise the embeddings of all samples in the two-dimensional space. For the overall model, all embedding vectors are concatenated before applying the dimensionality reduction.

Results

The performances of the full and the ultrametric hierarchy with feature forwarding are given in Table 5.13. For the full cascade, feature forwarding leads to an average absolute improvement of $F_{1,\text{macro}}$ by 0.8 % for ResNet-18 as base architecture and 0.9 % for DenseNet-121. On the other hand, for the ultrametric hierarchical approach, there is a small improvement by 0.5 % for ResNet-18 and a small performance decrease by 0.3 % for DenseNet-121. These cross-validation results are, exemplary for $F_{1,\text{macro}}$, visualised in Figure 5.12. The other measures show similar changes.

		$F_{1,\text{macro}}$ [%]	$F_{1,\text{micro}}$ [%]	$hF_{1,\text{macro}}$ [%]	$hF_{1,\text{micro}}$ [%]
		mean ± std	mean ± std	mean ± std	mean ± std
Full	ResNet-18, pre.	71.6 ± 3.5	77.6 ± 1.6	89.6 ± 1.8	94.1 ± 0.6
	ResNet-18, pre., FF	72.4 ± 3.5	78.0 ± 1.9	90.0 ± 1.6	94.2 ± 0.8
	DenseNet-121, pre.	71.1 ± 2.9	77.8 ± 1.7	89.2 ± 1.6	94.1 ± 0.5
	DenseNet-121, pre., FF	72.0 ± 2.4	78.4 ± 1.3	89.6 ± 1.1	94.3 ± 0.6
Ultrametric	ResNet-18, pre.	71.8 ± 3.0	77.4 ± 1.6	89.5 ± 1.7	94.0 ± 0.6
	ResNet-18, pre., FF	72.3 ± 4.3	78.2 ± 1.6	89.9 ± 1.9	94.2 ± 0.7
	DenseNet-121, pre.	71.5 ± 3.7	78.1 ± 1.3	88.8 ± 2.5	94.2 ± 0.5
	DenseNet-121, pre., FF	71.2 ± 3.5	77.8 ± 1.5	88.8 ± 2.0	94.1 ± 0.6

Table 5.13: Results with feature forwarding (FF) in the full hierarchical cascade and the ultrametric cascade.

Table 5.14 lists the results of combinations of feature forwarding with weighted cross-entropy loss ($\delta = 0.5$) and with oversampling ($\nu = 0.5$). In no cases, these combinations attain higher scores than feature forwarding with pretraining alone.

Figure 5.13 provides an example visualisation of the embeddings of the three networks in the full hierarchy cascade on the path from the root to the different types of neutrophilic granulocytes. The embedding vectors generated for individual samples by the convolutional part of the respective networks are visualised as points in a scatter plot. Figure 5.14 shows a dimensionality reduction of the concatenated feature vectors from all eight networks in a full hierarchy cascade with feature forwarding.

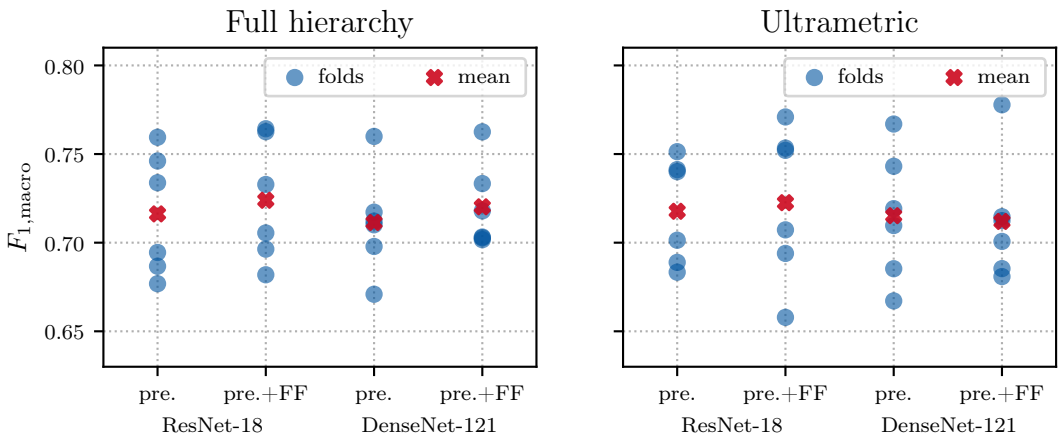


Figure 5.12: Scatter plots of macro-averaged F-scores achieved in the six different folds with feature forwarding.

	δ	ν	$F_{1,\text{macro}}$ [%] mean ± std	$F_{1,\text{micro}}$ [%] mean ± std	$hF_{1,\text{macro}}$ [%] mean ± std	$hF_{1,\text{micro}}$ [%] mean ± std
ResNet-18	0	0	72.4 ± 3.5	78.0 ± 1.9	90.0 ± 1.6	94.2 ± 0.8
	0.5	0	71.2 ± 3.4	77.4 ± 1.4	89.4 ± 1.9	94.0 ± 0.5
	0	0.5	72.4 ± 3.3	77.5 ± 1.5	89.9 ± 1.6	94.0 ± 0.7
DenseNet-121	0	0	72.0 ± 2.4	78.4 ± 1.3	89.6 ± 1.1	94.3 ± 0.6
	0.5	0	71.3 ± 2.7	77.7 ± 1.0	88.6 ± 2.4	94.0 ± 0.4
	0	0.5	71.4 ± 2.7	77.6 ± 1.0	89.3 ± 1.7	94.0 ± 0.4

Table 5.14: Scores of feature forwarding combined with loss weights or oversampling on the full hierarchy cascade with both basic architectures.

Discussion

In three out of four cases, feature forwarding combined with pretraining leads to small improvements of the performance scores when compared to pretraining alone. These small improvements are stronger for the full hierarchy compared to the ultrametric hierarchy, however still very small, and therefore might be negligible. Potential gains arise from the additional information the networks receive from their predecessors in form of the concatenated embedding vectors, which might explain the slightly larger gains for the deeper full hierarchy cascade compared to the ultrametric hierarchy.

A main advantage of including the embeddings is their potential to deliver feature space representations, usable for example for visualisation of the trained model. As indicated in the plots in Figure 5.13, the different groups and classes predicted by every network are visible as clusters. The annotations show the majority class in these clusters, however, some misclassifications are visible where points lie not close to points of the same class, but between samples from other classes. The clusters in the first two networks consist of multiple combined leaf-classes. In some but not all cases, there are visible trends within these clusters, rather than just approximately

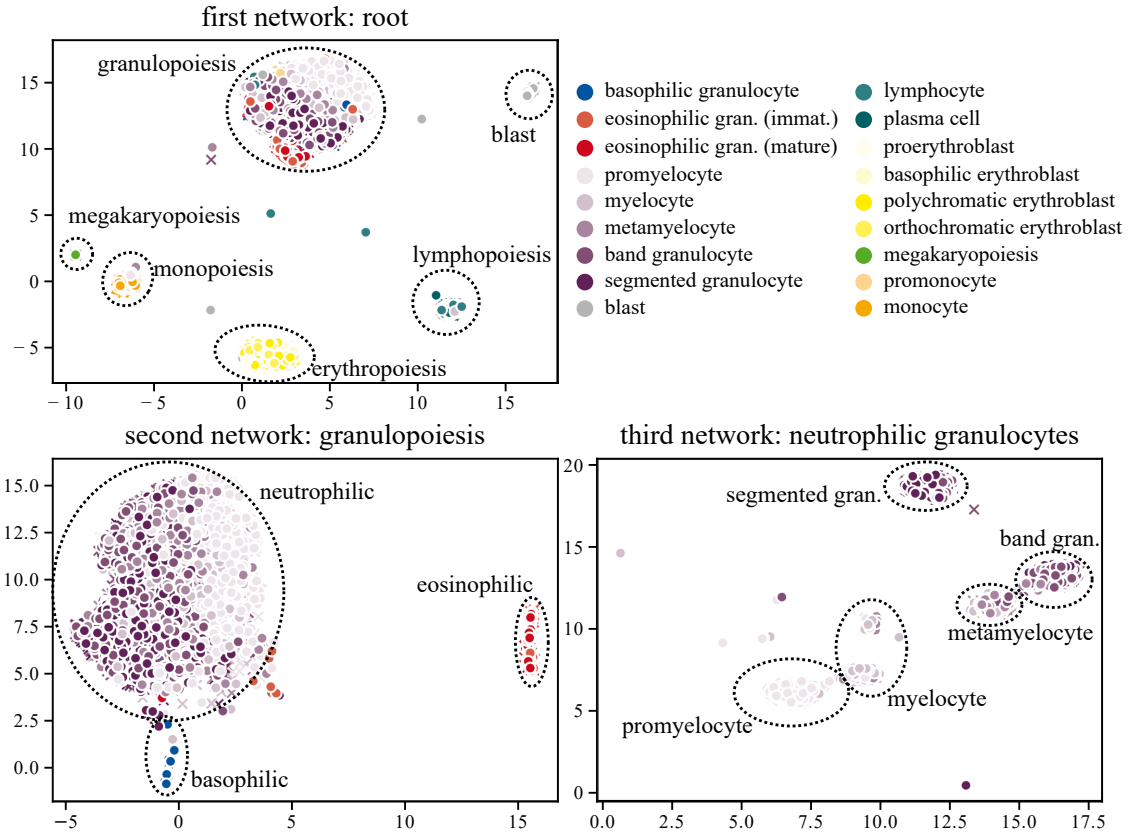


Figure 5.13: Embeddings of three networks on the path from the root to the leaf-classes of neutrophilic granulocytes. Each scatter point represents the 100-element embedding obtained from the respective convolutional network after dimensionality reduction to two dimensions using UMAP [43]. The ellipses and the annotations have been added manually only for illustrative reasons.

uniform distributions of the individual leaf classes. For example, in the group of neutrophilic granulocytes in the second network, more mature samples are located predominantly on the left side and immature stages such as promyelocytes on the right side in the two-dimensional space. These small trends even within the combined classes might be a contributing factor to the slight performance increases, because they indicate that some information about the leaf-class membership is already encoded in some earlier embeddings, potentially making them useful for later stages in the cascade. In the third network, predicting the cell type within neutrophilic granulocytes, the clusters majorly consisting of band granulocytes and metamyelocytes overlap. This is consistent with the previous observation and the assessment by medical experts (see Appendix A.2) that these two classes are especially easily confusable. For the clusters to be visible in the visualisations of the embeddings extracted at the second and third network, these embeddings must contain most of the relevant information required for the classification at these stages. This proves that each stage encodes new knowledge, and at the same time underlines that the

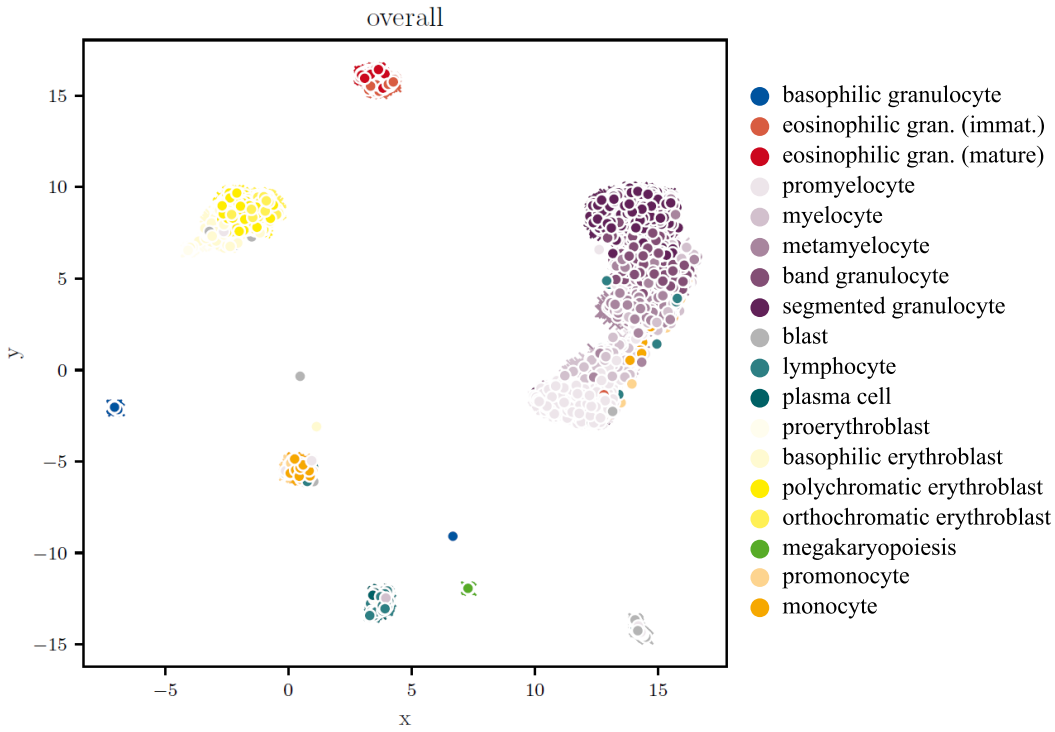


Figure 5.14: Visualisation of the learnt overall feature space from all networks in the full hierarchy cascade. Dimensionality reduction of concatenated embeddings from all networks was performed using UMAP [43].

performance improvements of feature forwarding are limited: the predictions seem to be made mainly from the embedding that has been added at that particular stage of the cascade. This is caused by the fact that no networks are trained before their predecessors. The earlier networks are trained only to produce embeddings optimised for their particular task, and not for their descendants. Because – once finished – these are not further trained, the ability to produce more complementary embeddings, largely increasing the classification performance, is limited.

The dimensionality reduction of the concatenated embedding vectors, shown in Figure 5.14, provides a visualisation of the entire cascade model. In this plot, there are several different clusters visible, which largely correspond to different classes. There are clusters containing different cell types, for example eosinophilic granulocytes, or cells of erythropoiesis. The different stages of neutrophilic granulocytes make up a large cluster – at least in the two-dimensional space. Other than in the dimensionality reduction of only the embedding from the root network, a general gradient of maturity from the lower to the upper part of the cluster is clearly visible, although without any sharp boundaries. A similar gradient is also visible in the scatter plot from the network for granulopoiesis, but it is much more pronounced in the concatenated embedding space.

Embeddings could also be calculated for each network without forwarding them by concatenation. While this would not utilise the already extracted features in other networks in any way, it would still allow similar visualisations, with the possible advantage of more independence of the individual networks, especially during training.

6 Discussion and Outlook

In the following, the developed methods are discussed on a more general level in consideration of the experiment results. Important findings of this work are mentioned, combined with a summary of their implications and practical advice for the application of hierarchical classification cascades. In addition to the major findings, aspects that might be interesting for the future but go beyond the scope of this work are discussed.

The results of the experiments show that, in general, hierarchical cascades of deep networks, as developed in this work, can successfully be employed for classification. However, the underlying hierarchy must be selected very carefully, otherwise hierarchical classification can potentially decrease the performance. Generally, similar classes should be grouped together at early levels in the hierarchy. In the haematological domain, the best performance is achieved when performing the hierarchical classification oriented on the actual cell taxonomy. While a hierarchy with more balanced class sizes can also provide good results, it does not achieve higher scores than single classifiers or cascades oriented on the domain. The class sizes therefore should, if possible, be considered when implementing hierarchical classification, but not as the only criterion to build a hierarchy. Other approaches, focused only on individual characteristics or even without biological justification, perform considerably worse, which shows that the number of networks alone does not improve the classification.

Predictions can be made probabilistically by calculation of the a posteriori probabilities for all classes based on the network confidences, or deterministically by greedily following only the highest confidences of the networks in the hierarchy. In theory, these two algorithms do not always lead to the same predictions, as the deterministic algorithm does not make use of the confidences of the individual networks. However, the experimental results show that, when applying the softmax function, the differences between the results from both methods are negligible. One reason for this is most likely the overconfidence of the networks. An interesting aspect for the future is to introduce more realistic confidence measures, which conform better with the observable classification reliability. With such a measure, the difference between the probabilistic and the greedy, deterministic algorithm to determine the final cascade prediction should be re-evaluated. While there is effectively no difference in the outcome with overconfident networks using the softmax activation function, other probabilistic measures might increase - or decrease – the performance obtained with soft propagation. Additionally, networks could be trained with an additional class representing rejected samples, which do not belong to any of

the children's classes. Strictly speaking, such an approach violates the assumption that in soft propagation a network only estimates the conditional probabilities of its children, given that its own class is true. In practice, however, it could make the cascades more robust, but would require alterations to the probabilistic decision-making process.

The best performance of cascades is achieved with pretraining on the target dataset, followed by separate training of the individual networks. Other methods, such as weighted losses or oversampling to partly compensate for the class imbalance do not lead to large improvements. On the contrary, the results indicate that they should be employed very conservatively, because too large weights or too excessive oversampling can strongly decrease the performance. This is also true for block sharing between the individual networks. While sharing the first conventional layers, providing basic and therefore universal features, does not strongly affect the performance, later blocks should remain separate, such that they can be trained for the particular sub-task within the cascade. In general, the largest gains could most likely be achieved by extending the dataset, since for all approaches the worst scores are mainly achieved for the underrepresented classes. While the exclusion of certain classes is inevitable for further research as long there are too few samples, it is not possible in real-world applications. This could not only involve additional annotations, but further data augmentation techniques, including generative approaches. Additionally, an inter-rater study to increase the reliability of the existing annotations might also be beneficial.

Hierarchical cascading of a classification task using deep neural networks does not increase the classification performance in all cases compared to flat networks. As the hierarchical evaluation scores show, flat deep networks are already capable to distinguish different lineages relatively reliably, while most misclassifications occur between closely related classes. To maximise performance, flat but very deep networks, for example ResNet-152 or DenseNet-201, might be a more promising approach than cascades of more shallow networks, especially in combination with moderate loss weights and oversampling. However, if the additional effort for pretraining of the individual networks on the target dataset is acceptable, similar performances can be achieved by the cascades. Cascades require significantly more training than flat networks, simply because multiple networks have to be trained, instead of only one. However, in terms of resource occupation, the cascades are not generally more demanding than a deeper single network. The required GPU-memory for the cascades is – when the networks are trained separately – not higher than the demands of the deeper flat networks with the same overall training setup. A somewhat similar alternative approach that might be worth further investigation are ensembles of flat networks. In contrast to the hierarchical cascades, these could combine multiple networks not hierarchically, but in parallel and make predictions for example by majority vote. The differences between the parallel networks could either be methodological by selection of different architectures or varying hyperparameters and optimisation techniques, or they could only be random Possible random approaches

might involve arbitrary shuffling or subsampling, for example by bagging [55], of the training set. Although such ensembles could consist of a similar number of networks as hierarchical cascades, they would most likely require more extensive training, because all networks would have to be trained on the entire dataset, instead of only subsets.

Cascading has other advantages, mainly in introducing modularity that a conventional, flat network cannot offer. If not relevant for a particular task, whole branches of the hierarchy including the networks could be neglected. On the other hand, an existing cascade could potentially be extended, for example when more samples become available. Classes that have been merged – for example basophilic granulocytes – could then be split again, which would only require an additional network at the leaf level, specifically trained only on a relatively small subset of the data. If a network were to be added at the root level on the other hand, for example to include artefacts, it would need to be trained on the entire dataset. When extending an existing cascade combined with academic evaluation of the resulting model, it must be ensured that the dataset splits used for training, validation and testing are preserved also for the additional networks. Another advantage is that local networks can be optimised more specifically. When using separate training, which is preferable for classification performance reasons and GPU-memory consumption, different training parameters, including different loss functions, could be used for each individual network. As was shown in this work, it is even possible to include other network types such as regressors. Regression at some points in the hierarchy is biologically well founded because of the ordinal character of the individual lineages, but it is not an appropriate approach at other nodes. This is also illustrated by the feature space representations derived from the dimensionality reductions of the embeddings obtained in the last experiment. The clusters of some lineages, most prominently neutrophilic granulocytes, but also eosinophilic granulocytes and erythroblasts, appear connected, often with some gradient of increasing maturity within the cluster. In a single, classical network, such combinations of classification and regression are not possible. When incorporating different network types into the cascade, it is important to consider whether they provide a probabilistic confidence measure, because soft propagation is not applicable otherwise. For regressors, a classification can be achieved by simple rounding, which limits the cascade decision making process to hard propagation. Other additional information which can directly be provided by a hierarchical cascade are intermediate level confidences, although such group-wise confidences can also be calculated from the leaf confidences provided by flat classification networks.

Another example for more in-depth information provided by hierarchical cascades are the multiple local embedding vectors introduced with feature forwarding. Even though the performance gains achieved with feature forwarding are small, these embedding vectors allow feature representations not only of an overall model, but also at intermediate levels. In the experiments, the embeddings of all networks were concatenated in a fixed order to obtain a feature representation of the entire model

as a whole. This method could be problematic in some cases, and might have to be adjusted in the future, based on the particular use case. Firstly, the concatenated vector can become very long, and the entries from some networks might not be relevant for a given sample. A naive approach to only concatenate the vectors on the path from the root to the predicted leaf leads to the problem that an entry at a given position represents different characteristics in different samples. Therefore, it might be necessary to reduce the embedding length, to subsample the vector in some way, or to find other methods to obtain a suitable size. One potential, although relatively intricate approach might be to introduce an additional fully-connected layer to map this vector to a smaller size, trained to allow a good overall classification based on this mapping. Secondly, concatenation of the embeddings from all networks implicitly assumes soft propagation, where the samples are parsed to every network. In hard propagation on the other hand, the tree is pruned, and not all embeddings are retrieved. This problem could, for example, be addressed by zero-padding. The resulting vector should have the same length as the concatenation of all vectors, which is n_{emb} times the number of networks in the whole cascade. The individual vectors on the path from the root to the leaves could then be inserted at designated positions. Whether these ideas are viable requires further examination in the future, if the necessity arises.

7 Conclusion

In this work, domain knowledge was incorporated into the classification of haematopoietic cells by splitting it into hierarchical sub-tasks. Different training strategies for deep neural networks within these cascades were examined, whereof separate training of the individual networks showed a higher performance than end-to-end training of the entire cascade.

Two methods to obtain final predictions were investigated. Whether the full, probabilistic approach, or the greedy, deterministic algorithm was used did not make a large difference in terms of the overall performance.

While reaching similar classification scores, the cascades were not able to outperform flat state-of-the art networks, especially for networks with large depths such as ResNet-152 and DenseNet-201.

By evaluating different classification hierarchies, it was shown that the best classification results are achieved by following the biological cell lineages. Hierarchies deviating strongly from the haematopoietic lineages, for example based on single characteristics, achieved lower classification scores both than other cascades as well as flat classifiers. Therefore, the hierarchical classification has to be applied carefully, and should only be performed based on a strong theoretical foundation. If such a taxonomy is given, hierarchical cascades offer some advantages over flat networks.

Main advantages include a larger adaptibility, for example by the possibility to incorporate other network types such as regressors, which can account for partly ordinal classes. Such diverse networks can provide additional measures that go beyond plain classification, for example a cell maturity estimate in the haematopoietic domain. On the other hand, cascades require significantly more resources in form of training time and memory compared to single networks with the same architecture.

Further optimisation techniques were evaluated, of which pretraining of the individual networks showed the largest improvement in terms of several different classification performance scores, including hierarchical measures. A method called feature forwarding was introduced, which involves passing of embedding vectors through the hierarchy. While no large improvement of the model was achieved, it was shown that the embedding vectors can be useful for visualisation of the feature spaces not only of the overall model, but also at intermediate stages.

Overall, in terms of the macro-averaged F-score, the highest classification performance of 72.4 % was achieved for a hierarchical cascade following the haematopoietic cell hierarchy, consisting on ResNet-18 networks pretrained on the haematological domain and employing feature forwarding. For comparison, out of different flat networks, DenseNet-201 reached an F-score of 71.8 %, only surpassed by DenseNet-121 combined with oversampling of underrepresented classes, which also attained a performance of 72.4 %.

Bibliography

[1] T. H. Brümmendorf and S. Koschmieder, *Molekular zielgerichtete Therapie der chronischen myeloischen Leukämie (CML)*. UNI-MED Verlag AG, 2014.

[2] P. Gräbel, M. Crysandt, R. Herwartz, M. Hoffmann, B. M. Klinkhammer, P. Boor, T. H. Brümmendorf, and D. Merhof, "Evaluating out-of-the-box methods for the classification of hematopoietic cells in images of stained bone marrow," in *Computational Pathology and Ophthalmic Medical Image Analysis*, pp. 78–85, Springer, 2018.

[3] P. Gräbel, G. Nickel, M. Crysandt, R. Herwartz, M. Hoffmann, B. M. Klinkhammer, P. Boor, T. H. Brümmendorf, and D. Merhof, "Systematic analysis and automated search of hyper-parameters for cell classifier training," 2020.

[4] R. Munker, E. Hiller, J. Glass, and R. Paquette, *Modern hematology: biology and clinical management*, vol. 864. Springer Science & Business Media, 2 ed., 2007.

[5] H. Löffler and J. Rastetter, *Atlas of clinical hematology*. Springer Science & Business Media, 6 ed., 2005.

[6] D. P. Berger, M. Engelhardt, H. Henß, and R. Mertelsmann, *Concise manual of hematology and oncology*. Springer Science & Business Media, 2008.

[7] R. Fuchs and P. Staib, *Mikroskopierkurs Hämatologie*. Nora-Verlag GmbH, 2013.

[8] T. Pollard, W. Earnshaw, and J. Lippincott-Schwartz, *Cell Biology*. Elsevier Health Sciences, 2008.

[9] A. V. Hoffbrand and J. E. Pettit, *Sandoz Atlas Klinische Hämatologie*. Gower Medical Publishing, 1989.

[10] M. Kubat, *An introduction to machine learning*, vol. 2. Springer, 2017.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, p. 448–456, JMLR.org, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[13] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[14] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[15] P. De Wilde, *Neural network models: an analysis*. Springer, 1996.

[16] S. Skansi, *Introduction to Deep Learning: from logical calculus to artificial intelligence*. Springer, 2018.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[19] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[20] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, pp. 9–48, Springer, 1998.

[21] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2009.

[24] L. Y. Pratt, J. Mostow, C. A. Kamm, and A. A. Kamm, "Direct transfer of learned information among neural networks.," in *Aaai*, vol. 91, pp. 584–589, 1991.

[25] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*, pp. 270–279, Springer, 2018.

[26] C. N. Silla and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, no. 1-2, pp. 31–72, 2011.

[27] F. Wu, J. Zhang, and V. Honavar, "Learning classifiers using hierarchically structured class taxonomies," in *International symposium on abstraction, reformulation, and approximation*, pp. 313–320, Springer, 2005.

[28] S. Kiritchenko, S. Matwin, A. F. Famili, *et al.*, "Functional annotation of genes using hierarchical text categorization," in *Proc. of the ACL Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*, 2005.

[29] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Incremental algorithms for hierarchical classification," *Journal of Machine Learning Research*, vol. 7, no. Jan, pp. 31–54, 2006.

[30] C. Wu, M. Tygert, and Y. LeCun, "A hierarchical loss and its problems when classifying non-hierarchically," *Plos one*, vol. 14, no. 12, 2019.

[31] P. Poddar and P. Rao, "Hierarchical ensemble of neural networks," in *IEEE International Conference on Neural Networks*, pp. 287–292, IEEE, 1993.

[32] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[33] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[34] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[35] M. Sabokrou, M. Fayyaz, M. Fathy, and R. Klette, "Deep-cascade: Cascading 3d deep neural networks for fast anomaly detection and localization in crowded scenes," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1992–2004, 2017.

[36] J. Schlemper, J. Caballero, J. V. Hajnal, A. N. Price, and D. Rueckert, "A deep cascade of convolutional neural networks for dynamic mr image reconstruction," *IEEE transactions on Medical Imaging*, vol. 37, no. 2, pp. 491–503, 2017.

[37] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[38] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.

[39] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[40] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[41] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[42] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, pp. 849–856, 2002.

[43] L. McInnes, J. Healy, N. Saul, and L. Grossberger, "Umap: Uniform manifold approximation and projection," *The Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.

[44] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE, 2005.

[45] T. Ojala, M. Pietikainen, and D. Harwood, "Performance evaluation of texture measures with classification based on kullback discrimination of distributions," in *Proceedings of 12th International Conference on Pattern Recognition*, vol. 1, pp. 582–585, IEEE, 1994.

[46] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.

[47] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[48] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[49] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[50] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[51] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Pytorch documentation." https://pytorch.org/docs/stable/index.html, 2015. Accessed: 2019-09-30.

[52] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[53] P. Gräbel, Ö. Özkan, M. Crysandt, R. Herwartz, M. Baumann, B. M. Klinkhammer, P. Boor, T. H. Brümmendorf, and D. Merhof, "Circular anchors for the detection of hematopoietic cells using retinanet," in *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, 2020.

[54] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*, pp. 1321–1330, 2017.

[55] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

A Appendix

A.1 Proof of Equivalence of the Micro-Averaged F-Score and Accuracy

The first steps of the following proof of the equivalence of the micro-averaged F_1-score and the accuracy for multi-class problems are based on https://simonhessner.de/why-are-precision-recall-and-f1-score-equal-when-using-micro-averaging-in-a-multi-class-problem/. We want to show

$$F_{1,\text{micro}} = \text{accuracy}. \tag{A.1}$$

If each sample is assigned to exactly one class, each misclassification contributes a pair of one false positive and one false negative, such that the total numbers of false positives and false negatives are always equal when looking at all classes together:

$$\sum_{i=1}^{n} N_{FP,i} = \sum_{i=1}^{n} N_{FN,i} \tag{A.2}$$

With the definitions of micro-averaged precision (2.19) and recall (2.20) this immediately leads to

$$\text{precision}_{\text{micro}} = \text{recall}_{\text{micro}}. \tag{A.3}$$

In fact, from (A.3) and the definition of $F_{1,\text{micro}}$ (2.21) it follows that micro-averaged precision, recall and F_1-score are equal:

$$F_{1,\text{micro}} = \frac{2 \cdot \text{precision}_{\text{micro}} \cdot \text{precision}_{\text{micro}}}{\text{precision}_{\text{micro}} + \text{precision}_{\text{micro}}} = \text{precision}_{\text{micro}}. \tag{A.4}$$

The accuracy was defined in (2.13) as

$$\text{accuracy} = \frac{\sum_{i=1}^{n} N_{TP,i}}{\sum_{i=1}^{n} N_{TP,i} + \sum_{i=1}^{n} N_{FP,i}}.$$

This is nothing but the fraction of correct predictions relative to the total number of predictions. It is apparent that this is the same as the micro-averaged precision (2.19), and because of (A.4) also as the micro-averaged F_1-score.

A.2 Similarity score based on survey of medical experts

Two medical experts from the Department of Haematology, Oncology, Haemostaseology and Stem Cell Transplantation at the University Hospital of the RWTH Aachen University were asked to rate the similarity between 15 cell types on a scale from 1 to 5:

1. not confusable
2. hard to confuse
3. sometimes confusable
4. easy to confuse
5. not distinguishable

The resulting matrix is shown in Table A.1. It should be noted that this matrix is not fully symmetrical. The pre-processing of this matrix to be a valid similarity matrix for spectral clustering is described in Chapter 4.4.4.

actual cell type \ supposed cell type	basophilic granulocyte	eosinophilic gran.	promyelocyte	myelocyte	metamyelocyte	band gran.	segmented gran.	blast	lymphocyte	proerythroblast	basoph. erythrobl.	polychr. erythrobl.	orthochr. erythrobl.	promonocyte	monocyte
basophilic granulocyte	-	1	2	1	1	1	1	1	1	1	1	1	1	1	1
eosinophilic gran.	2	-	1	1	1	1	1	1	1	1	1	1	1	1	1
promyelocyte	2	1	-	3	1	1	1	2	1	1	1	1	1	1	1
myelocyte	1	1	3	-	2	1	1	1	1	1	1	1	1	1	1
metamyelocyte	1	1	1	2	-	3	1	1	1	1	1	1	1	1	1
band gran.	1	1	1	1	3	-	3	1	1	1	1	1	1	1	1
segmented gran.	1	1	1	1	1	3	-	1	1	1	1	1	1	1	1
blast	1	1	2	1	1	1	1	-	2	2	1	1	1	1	2
lymphocyte	1	1	1	1	1	1	1	2	-	1	1	1	1	1	2
proerythroblast	1	1	1	1	1	1	1	2	1	-	2	1	1	1	1
basoph. erythrobl.	1	1	1	1	1	1	1	1	1	2	-	1	1	1	1
polychr. erythrobl.	1	1	1	1	1	1	1	1	1	1	1	-	2	1	1
orthochr. erythrobl.	1	1	1	1	1	1	1	1	1	1	1	2	-	1	1
promonocyte	1	1	1	1	1	1	1	1	1	1	1	1	1	-	3
monocyte	1	1	1	1	1	1	1	1	2	1	1	1	1	3	-

Table A.1: Similarity survey results.

A.3 Class-Wise Scores for Flat Networks and Different Hierarchies

In the following, the class wise F-scores of different hierarchies are listed in the most basic experimental setting using seperate training of networks pretrained on ImageNet and soft propagation.

	Flat			
	DenseNet-121	ResNet-18	DenseNet-201	ResNet-152
	mean ± std	mean ± std	mean ± std	mean ± std
Basophilic granulocyte	86.5 ± 3.5	89.0 ± 4.0	88.9 ± 4.0	89.0 ± 6.2
Eosino. gran. (immat.)	76.8 ± 7.2	77.5 ± 8.2	77.8 ± 6.5	81.6 ± 3.3
Eosino. (mature)	76.1 ± 10.2	78.1 ± 6.9	73.8 ± 8.6	79.7 ± 4.0
Promyelocyte	89.0 ± 2.0	87.9 ± 3.3	87.6 ± 2.2	88.9 ± 2.4
Myelocyte	67.3 ± 6.0	64.8 ± 3.7	65.7 ± 3.1	69.8 ± 4.8
Metamyelocyte	67.1 ± 4.0	65.4 ± 3.6	67.0 ± 3.1	67.6 ± 4.4
Band granulocyte	75.9 ± 3.4	73.6 ± 1.9	77.2 ± 1.2	77.3 ± 1.0
Segmented granulocyte	88.5 ± 1.4	87.5 ± 1.6	88.9 ± 1.5	89.4 ± 1.8
Blast	55.4 ± 6.8	55.9 ± 10.9	55.7 ± 7.2	59.8 ± 7.2
Lymphocyte	76.8 ± 5.7	77.5 ± 7.3	78.2 ± 4.3	78.1 ± 7.6
Plasma cell	67.6 ± 35.2	75.1 ± 16.1	82.2 ± 22.2	71.1 ± 35.8
Proerythroblast	66.4 ± 17.5	63.3 ± 11.0	65.6 ± 13.4	63.9 ± 19.5
erythroblast (basophilic)	70.2 ± 7.1	70.8 ± 5.2	73.5 ± 9.8	71.5 ± 4.9
erythroblast (orthochr.)	63.0 ± 7.0	60.3 ± 4.6	61.1 ± 6.1	60.5 ± 3.2
erythroblast (polychr.)	80.8 ± 4.2	79.7 ± 3.7	80.0 ± 3.1	79.8 ± 3.2
Megakaryopoiesis	84.9 ± 18.9	84.9 ± 18.9	87.3 ± 19.9	79.4 ± 18.5
Promonocyte	19.0 ± 19.0	13.8 ± 8.8	20.3 ± 3.9	19.6 ± 12.0
Monocyte	60.5 ± 6.9	60.8 ± 8.1	61.3 ± 7.8	57.9 ± 5.6

Table A.2: Class-wise F-scores of flat networks.

	Full		Ultrametric	
	RN-18	DN-121	RN-18	DN-121
	mean ± std	mean ± std	mean ± std	mean ± std
Basophilic granulocyte	90.0 ± 5.0	90.9 ± 6.3	88.0 ± 4.0	89.3 ± 6.5
Eosino. gran. (immat.)	79.0 ± 6.3	77.6 ± 2.7	76.9 ± 6.7	79.0 ± 6.1
Eosino. (mature)	75.5 ± 8.2	76.5 ± 5.9	78.1 ± 5.1	78.0 ± 8.1
Promyelocyte	87.7 ± 2.6	87.6 ± 2.1	88.1 ± 2.3	88.4 ± 3.3
Myelocyte	66.5 ± 4.9	63.8 ± 3.7	65.9 ± 2.8	64.4 ± 4.2
Metamyelocyte	65.1 ± 2.4	66.7 ± 1.4	64.5 ± 3.4	64.9 ± 3.7
Band granulocyte	73.9 ± 2.5	76.8 ± 2.2	75.2 ± 1.7	76.4 ± 2.3
Segmented granulocyte	88.4 ± 1.1	89.5 ± 1.3	89.2 ± 0.8	89.6 ± 1.3
Blast	56.0 ± 8.7	53.0 ± 12.2	53.5 ± 7.6	55.4 ± 8.2
Lymphocyte	75.7 ± 6.7	74.7 ± 5.8	75.9 ± 6.0	72.2 ± 7.7
Plasma cell	76.1 ± 21.0	69.0 ± 15.5	71.4 ± 17.4	68.5 ± 34.8
Proerythroblast	66.5 ± 9.6	62.8 ± 6.9	63.6 ± 22.5	68.6 ± 16.6
erythroblast (basophilic)	65.4 ± 6.7	67.0 ± 5.0	70.6 ± 10.2	71.1 ± 7.9
erythroblast (orthochr.)	63.0 ± 5.2	61.8 ± 6.0	64.0 ± 6.2	61.6 ± 8.9
erythroblast (polychr.)	80.2 ± 3.4	79.1 ± 2.8	80.9 ± 3.7	79.7 ± 4.5
Megakaryopoiesis	80.4 ± 21.5	83.1 ± 17.6	87.1 ± 17.0	83.4 ± 21.8
Promonocyte	18.8 ± 14.0	22.7 ± 15.6	21.0 ± 3.5	14.2 ± 9.0
Monocyte	57.9 ± 4.2	59.6 ± 7.7	56.7 ± 11.6	57.9 ± 4.1

Table A.3: Class-wise F-scores of the full and the ultrametric hierarchy with the two different network architectures.

	Cytoplasm features		Ratio/shape features	
	RN18	DN-121	RN18	DN-121
	mean ± std	mean ± std	mean ± std	mean ± std
Basophilic granulocyte	85.4 ± 4.3	88.9 ± 7.1	89.7 ± 6.5	88.9 ± 6.2
Eosino. gran. (immat.)	76.2 ± 5.4	78.5 ± 6.0	78.7 ± 3.7	78.5 ± 4.3
Eosino. (mature)	77.6 ± 4.5	75.7 ± 8.3	74.0 ± 2.8	77.3 ± 4.5
Promyelocyte	86.7 ± 1.9	88.0 ± 2.3	86.3 ± 1.6	87.4 ± 2.2
Myelocyte	62.5 ± 3.2	64.5 ± 5.2	61.6 ± 2.3	65.8 ± 5.6
Metamyelocyte	66.1 ± 3.5	67.7 ± 3.3	61.6 ± 4.6	63.0 ± 3.8
Band granulocyte	74.3 ± 2.8	76.0 ± 2.9	71.8 ± 3.6	73.7 ± 2.0
Segmented granulocyte	88.2 ± 1.6	89.0 ± 1.9	88.1 ± 1.1	89.0 ± 1.1
Blast	49.6 ± 8.7	49.1 ± 8.8	43.8 ± 7.0	46.7 ± 12.3
Lymphocyte	71.5 ± 7.0	74.4 ± 6.9	72.1 ± 4.7	74.8 ± 6.1
Plasma cell	61.5 ± 32.3	75.2 ± 18.5	71.8 ± 21.0	72.5 ± 20.9
Proerythroblast	70.0 ± 8.8	65.1 ± 17.1	54.5 ± 16.1	58.1 ± 23.5
erythroblast (basophilic)	71.4 ± 9.1	70.9 ± 9.8	69.2 ± 8.8	67.0 ± 9.2
erythroblast (orthochr.)	61.7 ± 7.2	64.6 ± 10.5	61.7 ± 7.9	58.6 ± 9.2
erythroblast (polychr.)	81.8 ± 2.0	82.1 ± 4.3	80.6 ± 2.8	77.0 ± 5.3
Megakaryopoiesis	81.6 ± 17.4	13.3 ± 32.7	44.3 ± 48.9	23.8 ± 38.0
Promonocyte	23.0 ± 10.2	16.2 ± 10.7	15.4 ± 16.3	10.4 ± 9.7
Monocyte	54.3 ± 13.8	55.7 ± 9.7	52.4 ± 9.8	58.4 ± 8.2

Table A.4: Class-wise F-scores of cytoplasm and ratio/shape feature-based hierarchies.

	Nucleus features	
	RN18	DN-121
	mean ± std	mean ± std
Basophilic granulocyte	88.4 ± 8.3	85.4 ± 9.8
Eosino. gran. (immat.)	77.2 ± 6.4	78.6 ± 3.8
Eosino. (mature)	76.8 ± 5.1	77.3 ± 6.8
Promyelocyte	87.7 ± 1.9	88.1 ± 2.5
Myelocyte	62.4 ± 3.7	63.8 ± 4.8
Metamyelocyte	65.0 ± 3.9	63.8 ± 4.5
Band granulocyte	74.4 ± 3.0	74.2 ± 1.5
Segmented granulocyte	87.2 ± 2.0	88.1 ± 1.7
Blast	53.9 ± 7.1	53.0 ± 7.4
Lymphocyte	71.6 ± 7.4	72.9 ± 5.9
Plasma cell	62.0 ± 34.6	59.9 ± 33.5
Proerythroblast	68.5 ± 11.2	58.7 ± 14.9
erythroblast (basophilic)	66.5 ± 4.7	63.1 ± 5.2
erythroblast (orthochr.)	62.8 ± 6.8	60.3 ± 10.1
erythroblast (polychr.)	79.1 ± 2.3	79.0 ± 3.3
Megakaryopoiesis	48.3 ± 38.7	11.1 ± 27.2
Promonocyte	6.1 ± 14.8	16.5 ± 9.6
Monocyte	50.9 ± 9.6	55.3 ± 7.8

Table A.5: Class-wise F-scores of hierarchy based on nucleus features.

	Confusion clustering		Similarity clustering	
	RN18	DN-121	RN18	DN-121
	mean ± std	mean ± std	mean ± std	mean ± std
Basophilic	86.6 ± 6.9	90.7 ± 4.7	84.2 ± 8.9	89.3 ± 6.9
Eosinophilic (immature)	79.0 ± 7.0	80.2 ± 5.0	78.9 ± 4.7	78.6 ± 7.7
Eosinophilic (mature)	80.3 ± 5.2	78.4 ± 6.2	75.5 ± 5.2	80.4 ± 6.0
Promyelocyte	87.6 ± 2.5	87.9 ± 1.9	88.0 ± 1.5	87.8 ± 2.0
Myelocyte	64.9 ± 1.7	65.6 ± 4.6	65.4 ± 2.1	64.3 ± 4.2
Metamyelocyte	63.4 ± 3.1	65.1 ± 1.9	64.2 ± 5.4	67.7 ± 4.1
Band granulocyte	72.1 ± 1.8	75.7 ± 1.7	73.6 ± 2.3	75.7 ± 1.7
Segmented granulocyte	87.6 ± 1.5	90.0 ± 1.4	88.0 ± 1.6	88.6 ± 1.7
Blast	53.9 ± 7.9	51.0 ± 6.4	48.4 ± 4.7	48.1 ± 10.7
Lymphocyte	75.1 ± 5.2	71.8 ± 8.5	73.1 ± 7.7	72.9 ± 7.2
Plasma cell	60.5 ± 31.5	81.2 ± 13.8	75.5 ± 21.2	76.8 ± 23.9
Proerythroblast	51.8 ± 28.2	21.2 ± 29.1	67.4 ± 11.9	63.8 ± 14.9
Erythroblast (basophilic)	67.0 ± 5.3	67.3 ± 4.5	73.7 ± 6.0	70.3 ± 9.2
Erythroblast (orthochr.)	63.4 ± 6.6	61.0 ± 10.6	59.9 ± 5.3	57.5 ± 9.8
Erythroblast (polychr.)	80.3 ± 3.0	79.4 ± 3.9	80.2 ± 3.6	77.6 ± 5.0
Megakaryopoiesis	63.1 ± 34.1	61.7 ± 33.7	0.0 ± 0.0	11.1 ± 27.2
Promonocyte	0.0 ± 0.0	0.0 ± 0.0	5.2 ± 9.1	30.5 ± 11.3
Monocyte	55.7 ± 8.4	57.8 ± 9.1	58.2 ± 8.2	62.6 ± 6.0

Table A.6: Class-wise F-scores of clustering based hierarchies.

	Balanced		Random	
	RN18	DN-121	RN18	DN-121
	mean ± std	mean ± std	mean ± std	mean ± std
Basophilic gran.	91.0 ± 3.0	88.3 ± 4.9	41.9 ± 46.1	86.1 ± 8.9
Eosino. gran. (immature)	77.9 ± 5.0	79.1 ± 7.2	75.7 ± 7.5	76.1 ± 3.5
Eosino. gran. (mature)	75.2 ± 4.6	76.1 ± 7.4	76.7 ± 5.5	74.8 ± 2.1
Promyelocyte	88.4 ± 2.4	89.2 ± 2.1	86.5 ± 2.7	88.0 ± 1.3
Myelocyte	66.8 ± 3.6	68.8 ± 5.3	60.9 ± 4.0	63.0 ± 4.4
Metamyelocyte	64.5 ± 2.2	66.6 ± 3.0	61.9 ± 3.8	63.3 ± 3.0
Band granulocyte	73.9 ± 3.2	74.9 ± 2.8	72.1 ± 2.7	75.3 ± 2.0
Segmented granulocyte	88.5 ± 1.4	89.4 ± 1.2	88.0 ± 1.3	88.8 ± 1.6
Blast	52.1 ± 8.6	54.1 ± 3.9	52.4 ± 8.1	52.4 ± 4.4
Lymphocyte	76.3 ± 6.9	76.7 ± 5.8	71.4 ± 3.1	75.8 ± 9.4
Plasma cell	71.2 ± 17.1	72.0 ± 20.1	79.7 ± 12.0	66.2 ± 17.1
Proerythroblast	57.6 ± 14.4	59.0 ± 15.3	21.5 ± 33.4	58.7 ± 14.2
Erythroblast (basophilic)	72.4 ± 7.7	74.6 ± 10.8	69.4 ± 7.9	68.8 ± 5.0
Erythroblast (orthochr.)	62.6 ± 8.5	60.7 ± 11.7	61.2 ± 5.9	57.1 ± 9.3
Erythroblast (polychr.)	82.1 ± 3.4	81.4 ± 2.9	80.5 ± 2.6	79.3 ± 3.5
Megakaryopoiesis	69.8 ± 29.0	71.0 ± 38.3	67.3 ± 37.2	79.4 ± 18.5
Promonocyte	17.8 ± 16.2	26.3 ± 14.6	2.8 ± 6.8	9.7 ± 11.5
Monocyte	57.5 ± 9.8	61.1 ± 9.2	52.3 ± 6.3	49.8 ± 10.3

Table A.7: Class-wise F-scores of the balanced and the random hierarchy.

A.4 Validation Score and Losses for End-To-End Training of the Full Cascade

The development of the validation F-score and the individual losses for end-to-end training of the ultrametric cascade is given in Figure 5.5 in Section 5.3. For the sake of completeness, the same plots for the full hierarchy are given in Figure A.1

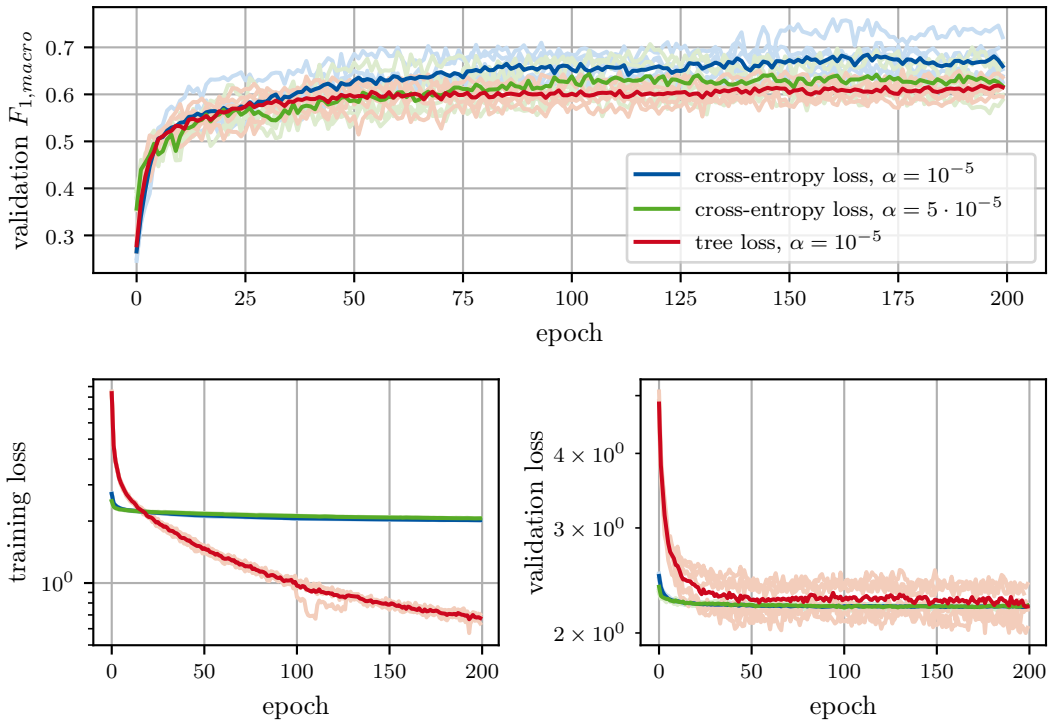


Figure A.1: Development of the validation-$F_{1,\text{macro}}$-score and the different training and validation losses for end-to-end training of the full cascade. The highlighted lines are the respective averages over all six folds.

A.5 Validation Score Plots for Flat Reference Networks

The validation scores of flat ResNet-18 and DenseNet-121 are plotted in Figure A.2. It can be observed that the validation score converges well before the end of training.

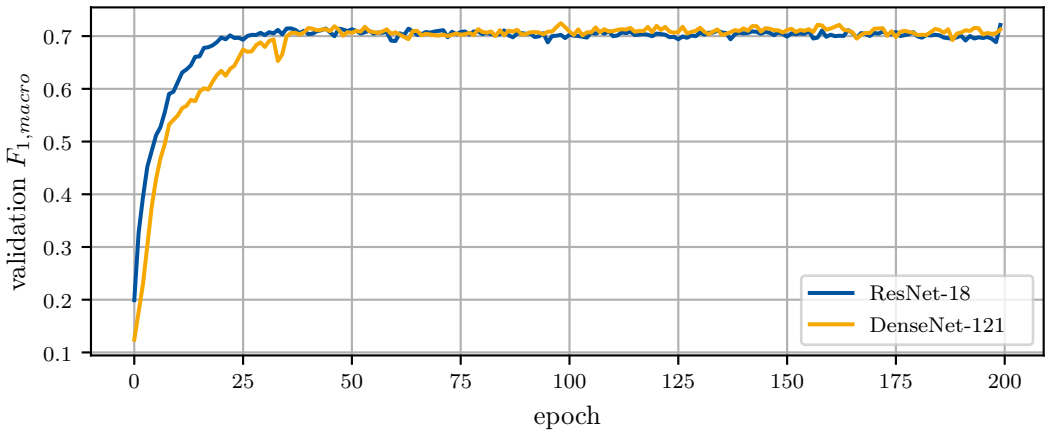


Figure A.2: Average validation scores of flat ResNet-18 and DenseNet-121.