Deep Learning in Skin Lesion Classification Tasks

Renée Vanheule

Supervisor: Prof. dr. ir. Aleksandra Pizurica
Counsellors: Angel Javier Lopez Aguirre, Simon Donné, Dr. ir. Danilo Babin, Laurens Meeus

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Biomedical Engineering

Department of Telecommunications and Information Processing
Chair: Prof. dr. ir. Herwig Bruneel
Faculty of Engineering and Architecture
Academic year 2017-2018

Preface

Delving into the world of deep learning and looking for ways to incorporate my biology-focused engineering masters with the field of artificial intelligence proved to be an interesting, exciting, and challenging at times, journey. I would like to sincerely thank Professor Pizurica for giving me the freedom to explore this field and investigate possible applications for a skin lesion diagnostic task, and guiding me throughout the work.

Furthermore, the input from Laurens Meeus was indispensable to this work. Always gladly offering his knowledge on deep learning as well as his opinions on some of the encountered problems in our discussions, he helped me in focusing my thoughts and moving forwards with the work. Therefore I would like to extend many thanks to him.

Finally, I would like to thank my family for their support throughout my studies.

Permission of use on loan

The author gives permission to make this master dissertation available for consultation and to copy parts of this master dissertation for personal use. In all cases of other use, the copyright terms have to be respected, in particular with regard to the obligation to state explicitly the source when quoting results from this master dissertation.

Renée Vanheule
Ghent, June 18th 2018

DEEP LEARNING IN SKIN LESION CLASSIFICATION TASKS

Renée Vanheule

Supervisor: Prof. Dr. ir. Aleksandra Pizurica
Counsellors: Laurens Meeus, Angel Javier Lopez Aguirre, Simon Donné, Dr. ir. Danilo Babin

Master's dissertation submitted in order to obtain the academic degree of Master of Science in Biomedical Engineering

Department of Telecommunications and Information Processing
Chair: Prof. dr. ir. Herwig Bruneel
Faculty of Engineering and Architecture
Academic year 2017-2018

Abstract- Skin cancer is the most common type of malignancy in the Caucasian population and incidence is on the rise. With only 1 % of all skin cancers being comprised of melanomas, this constitutes the most dangerous form of skin cancer given its propensity to metastasize. Diagnostic processes based on visual assessment portray highly variable results and biopsies are required for confirmation. CAD-systems are therefore desired to allow early and accessible detection and enhance classification performance. This study focuses on the use of deep learning for classification of melanoma vs. non-melanoma skin lesions with a limited labeled dataset. A first line of research delves into the use of transfer learning for this task. Applying SVM-classification on features extracted from pretrained state-of-the-art models, point to high performance (in terms of AUC-score) of the ResNet50 model. Performance can be enhanced through combinations of ResNet50 and InceptionV3 features. Additionally, finetuning of the ResNet50 model was examined and optimal results were achieved through supervised learning of the final softmax layer alongside the deepest residual block. A second line of research explored the use of unlabeled datasets in the training process via an auto-encoder framework. Two convolutional auto-encoder models were proposed (differing in their depth) and the best performance was obtained after a supervised learning phase of an identity block-based top model stacked on the pretrained, deeper encoder. The AUC-scores are in the same ballpark of what was achieved with SVM classifiers trained on VGG16 FC2-features or deep InceptionV3 features, thus portraying a clear window of opportunity for further exploration of unsupervised training of auto-encoders as (partial) pretraining phase for a target-domain and -task specific CNN.

Keywords- skin lesion classification, pretraining, transfer learning, convolutional auto-encoders

Deep Learning in Skin Lesion Classification Tasks

Renée Vanheule[1]

Supervisor: Prof. dr. ir. Aleksandra Pizurica
Counsellors: Laurens Meeus, Angel Javier Lopez Aguirre, Simon Donn, Dr. ir. Danilo Babin

Abstract— **Skin cancer is the most common type of malignancy in the Caucasian population and incidence is on the rise. With only 1 % of all skin cancers being comprised of melanomas, this constitutes the most dangerous form of skin cancer given its propensity to metastasize. Diagnostic processes based on visual assessment portray highly variable results and biopsies are required for confirmation. CAD-systems are therefore desired to allow early and accessible detection and enhance classification performance. This study focuses on the use of deep learning for classification of melanoma vs. non-melanoma skin lesions with a limited labeled dataset. A first line of research delves into the use of transfer learning for this task. Applying SVM-classification on features extracted from pretrained state-of-the-art models, point to high performance (in terms of AUC-score) of the ResNet50 model. Performance can be enhanced through combinations of ResNet50 and InceptionV3 features. Additionally, finetuning of the ResNet50 model was examined and optimal results were achieved through supervised learning of the final softmax layer alongside the deepest residual block. A second line of research explored the use of unlabeled datasets in the training process via an auto-encoder framework. Two convolutional auto-encoder models were proposed (differing in their depth) and the best performance was obtained after a supervised learning phase of an identity block-based top model stacked on the pretrained, deeper encoder. The AUC-scores are in the same ballpark of what was achieved with SVM classifiers trained on VGG16 FC2-features or deep InceptionV3 features, thus portraying a clear window of opportunity for further exploration of unsupervised training of auto-encoders as (partial) pretraining phase for a target-domain and -task specific CNN.**

Keywords— **skin lesion classification, pretraining, transfer learning, convolutional auto-encoders**

I. INTRODUCTION

Anno 2018, skin cancer is recognized to be the most common type of malignancy in the Caucasian population and continues to portray upwards trends in incidence [1], thus placing an enormous health and economical burden on our society. In Belgium, over 37 000 cases have been reported in 2017, with numbers expected to rise by another 5 to 9 % per year depending on the type of skin cancer [2]. Aside from genetic predisposition and particular environmental factors, the leading cause of skin cancer is exposure to ultraviolet radiation [3]. Initiating a complex interplay between direct DNA damage (mainly attributed to UVB) as direct effect, and immunosuppression, as well as the formation of free reactive oxygen species (ROS) responsible for indirect damage (the latter primarily caused by UVA), as long term effects due to recurrent/chronic exposure [4], it may target different cells. A general distinction is made between non-melanoma skin cancer and melanoma skin cancer, representing respectively about 99 % and 1 % of all diagnosed skin cancers [5].

[1]This study was performed at the Department of Telecommunications and Information Processing, Ghent University (UGent), Ghent, Belgium.

Melanoma is allocated its own category as it has a very high propensity to metastasize and is thus the most dangerous form of skin cancer [6]. Early detection and resection of the melanoma, offers the patient a 5-year survival rate of 99 %. This drops however to 63 % as the melanoma enters the regional stage and further down to 18 % upon reaching the distal stage [7]. In the current skin lesion diagnostic process, physicians rely on subjective systems by looking at specific characteristics of the lesion. This renders the diagnosis highly dependent on the physician's training, previous experience and interpretation, and is further complicated by the high inter-patient variability in lesion appearance. Diagnosis is only confirmed upon performing a biopsy, with many lesions thus being biopsied unnecessarily. In light of the need for early and accessible detection to improve survival chances, as well as the high variability in the diagnostic process, instigating the execution of biopsies for finality, there is a clear window of opportunity for the use of CAD-systems in this field.

Classic machine learning approaches, incorporating the steps of feature extraction followed by the training of a classifier [8], have been applied to skin lesion classification tasks. Recent trends in research have however portrayed a shift towards deep learning in this domain, with the landmark paper published by Esteva et al. at Stanford [9]. Adapting the pretrained InceptionV3 network to the task at hand and finetuning it on a large, mostly private, database of labeled skin lesion images, they reported performances matching that of at least 21 dermatologists.

Moving away from the data-driven approach performed at Stanford and in consideration of the limited amount of publicly available high-quality labeled skin lesion images, a more horizontal comparative study at a lower level is desirable and chosen as the main approach in this study. Incorporating deep learning techniques for limited medical datasets requires as a rule pretraining of the architectures to cope with the dimensionality of the problem. Furthermore, attention has to be given to problems of overfitting and

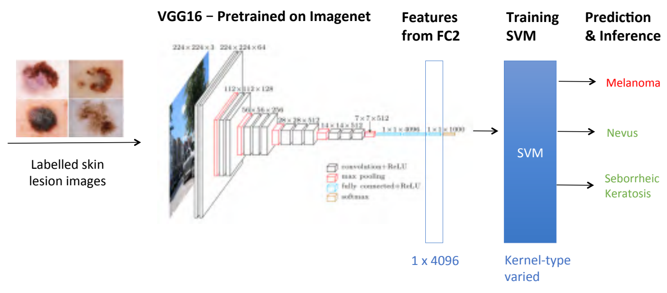

Fig. 1. Illustration of CNN as feature extractor (partially adapted from [10] and [11]).

results skewing to the overrepresented classes, as medical datasets are often heavily imbalanced with healthy samples being over represented.

II. Methodology

Under the incentives specified in the introduction, this study explores a binary classification of melanoma vs. non-melanoma based on the ISIC 2017 labeled skin lesion dataset and the collection of 'untagged' skin lesions published on the ISIC Archive as unlabeled dataset [10]. As the labeled dataset spans three different skin lesion classes, classifiers are trained for this three-class classification taks, from which the binary (i.e. melanoma/non-melanoma) classification results are inferred. A labeled test set is kept separately at all times to allow for unbiased testing. All experiments are performed in Keras, with Tensorflow backend. Three main frameworks w.r.t. pretraining are explored, two in the line of transfer learning and one incorporating the use of unlabeled data via auto-encoders. Techniques for countering overfitting and handling class imbalance are studied alongside the three frameworks.

A. Transfer learning

In a first line of research, two transfer learning schemes on several pretrained state-of-the-art models (VGG16, VGG19 [11], InceptionV3 [12] and Resnet50 [13]) are investigated and classifiers are compared in terms of their AUC-score for the detection of melanoma. All data is preprocessed, through resizing and mean subtraction, conform the requirements of the applied pretrained model.

1) Employing the pretrained model as feature extractor and performing SVM-classification on top of this: For features extracted from the FC2-layer of VGG16, a cross-validation scheme is instigated to study mediating the class imbalance problem through introduction of class weighting or augmentation of samples in the feature space (via ROS or SMOTE [14]). Parametric optimization of the SVM is similarly evaluated with a cross-validation scheme. All optimizations are evaluated on the AUC-score of melanoma classification. This procedure is visualised in Figure 1.
Subsequently, a comparison is made for the classifier, under the chosen hyperparameter settings, trained on features from different pretrained models. Furthermore, combinations of features from different layers of the same model (in case of either of the VGG-nets) and from different models is studied.

2) Finetuning a pretrained state-of-the-art model for the skin lesion classification task: In line with the results of the aforementioned experiments, the ResNet50 model was chosen for further examination. This study analyzes the optimal depth of finetuning of the pretrained ResNet50 model with an output layer adapted to the skin lesion diagnostic task at hand. Specific consideration is given to the architecture of the ResNet50 model as a stacked model of residual blocks (alterations of convolution and identity blocks). Overfitting and class imbalance is addressed through dropout, data augmentation and class weighting. To this end, we introduce a data augmentation scheme based on Krizhevsky et al. [15], as used by Simonyan & Zisserman [11], i.e. isotropically rescaling the images and performing random crops, rotations and flips. Supervised training is performed with the ADAM-optimizer [16] ($\beta_1 = 0.9$, $\beta_2 = 0.999$, ϵ = None, decay = 0) for variable learning rates.

B. Convolutional auto-encoders for pretraining on unlabeled skin lesion data

In light of the large unlabeled skin lesion datasets that are available, the second line of research explores the inclusion of said data in the training process through an auto-encoder framework. We suggest two convolutional auto-encoder (CAE) models with multiple hidden layers as unsupervised learning framework to accomplish this. The encoders will subsequently be complemented by different top models, upon which both supervised learning of the top model and finetuning of the final CNN can be performed.

The unlabeled data is normalized through rescaling to [0,1] and real-time data augmentation is applied following the same procedure as introduced for transfer learning. We propose two different CAE architectures: (1) AE_1 incorporating three stages of convolutional layers, each one followed by a maxpooling layer, in its encoder (Table I), and mirrored in its decoder (Table II) and (2) AE_2 with an increased number of filters in each stage and a fourth stage is added, with parameters $[f_1, f_2, f_3, f_4]$ = [64, 128, 256, 512]. We made the choice to implement two convolutional layers after each other in the first two stages of the encoder to simulate a larger receptive field, while keeping the parametric demand reasonable, similar to what was implemented in the VGG-nets [11]. Training of the models was performed with the ADAM-optimizer and the mean squared error as objective function. In consideration of the time-constraint on this study, training was manually halted upon reaching early convergence area, as followed on training loss and accuracy graphs.

To accelerate the supervised training process, the labeled images are resized via central crops from their isotropically rescaled variants and normalized to [0,1] range. Data augmentation is expected to improve results, but could not be explored due to the time demands on the training

TABLE I

ENCODER ARCHITECTURE OF AE_1 WITH $[f_1, f_2, f_3]$ = [32, 64, 128]

Input	Skin lesion image $\in$ [0, 1]
Layers	2 x Conv2D (nb. filters = f_1, kernel size = (3, 3), 'relu') 1 x MaxPooling2D (pool size = (2, 2)) 2 x Conv2D (nb. filters = f_2, kernel size = (3, 3), 'relu') 1 x MaxPooling2D (pool size = (2, 2)) 1 x Conv2D (nb. filters = f_3, kernel size = (3, 3), 'relu') 1 x MaxPooling2D (pool size = (2, 2))
Output	Encoded vector

TABLE II

DECODER ARCHITECTURE OF AE_1 WITH $[f_1, f_2, f_3]$ = [32, 64, 128]

Input	Encoded vector
Layers	1 x Conv2D (nb. filters = f_3, kernel size = (3, 3), 'relu') 1 x UpSampling2D (pool size = (2, 2)) 2 x Conv2D (nb. filters = f_2, kernel size = (3, 3), 'relu') 1 x UpSampling2D (pool size = (2, 2)) 2 x Conv2D (nb. filters = f_1, kernel size = (3, 3), 'relu') 1 x UpSampling2D (pool size = (2, 2)) 1 x Conv2D (nb. filters = 3, kernel size = (3, 3), 'sigmoid')
Output	Decoded image $\in$ [0,1]

process. Different top models are explored for stacking on top of the encoder: (1) top model with single dense layer (Table III) and (2) top model based on a residual block as used in the ResNet50 architecture (Table IV). In the latter case, both an identity block (For AE_1: k = 3 and $[f_1, f_2, f_3]$ = [64, 64, 128]; For AE_2: k = 3 and $[f_1, f_2, f_3]$ = [128, 128, 512]) and a convolution block (For AE_1: k = 3, $[f_1, f_2, f_3]$ = [128, 128,256], s = 2) is investigated. For the architectural details, we refer to reader to [13] and [17]. Class weighting is applied to cope with class imbalance and the learning rate as well as dropout rate are under investigation as hyperparameters to deal with overfitting. Once again, training is performed with the ADAM optimizer and a categorical cross-entropy loss function.

III. RESULTS & DISCUSSION

A. Transfer learning

1) Employing the pretrained model as feature extractor and performing SVM-classification on top of this: An RBF-kernel SVM obtained the best results on the FC2-features from VGG16. With regard to class imbalance handling, all schemes resulted in models with similar capacities (represented in their AUC-scores). Nevertheless, the use of ROS or SMOTE in the feature space resulted in an immediate increase in F1-scores. Omitting the step of decision thresholding of the classifier, we therefore chose to work with an RBF-SVM on SMOTE-augmented features and report an AUC-score of 0.765 $\pm$ 0.0231 and an F1-score of 0.45 $\pm$ 0.0229 obtained during cross-validation. Penalty parameter C and kernel coefficient γ were subsequently established through a cross-validation scheme, optimizing on the AUC-score.

TABLE III

TOP MODEL WITH A SINGLE DENSE LAYER

Input	Encoded vector
Top layers	Flatten () Dropout (rate = r) Dense (size = 3, 'softmax')
Output	Label prediction: $p_i \in$ [0, 1]

TABLE IV

TOP MODEL BASED ON A RESIDUAL BLOCK (EITHER IDENTITY OR CONVOLUTION BLOCK)

Input	Encoded vector
Top layers	Residual Block (k, $[f_1, f_2, f_3]$, s) AveragePooling2D (size = (7, 7))Flatten () Dropout (rate = r) Dense (size = 3, 'softmax')
Output	Label prediction: $p_i \in$ [0, 1]

Exploring an SVM-based classification of features extracted from different state-of-the-art pretrained models, indicated a high performance, in terms of AUC-score, for features from ResNet50 (AUC-score of 0.809), and adequate results for FC1-features from VGG19 and VGG16, as can be observed from Table V. Combinations of features from different layers of the VGG-nets did not improve performance. Per contra, moving towards the use of features from different models, specifically those from ResNet50 and InceptionV3, did enhance performance (AUC-score of 0.816), indicating that both nets might be able to extract some complementary features. These results are visualised in Figure 2, alongside the documentation of the corresponding AUC-scores. The two best-performing combinations are documented in Table VI. Concerning the F1-score (and related precision and recall) for the positive class (i.e. melanoma), we recognize that further decision thresholding of the SVM classifier can be employed to fully exploit the classifier's abilities, as indicated by its AUC-score.

2) Finetuning a pretrained state-of-the-art model for the skin lesion classification task: Supervised training of the new target-domain specific output layer, freezing all other layers, of the pretrained ResNet50 model was performed. Results indicated the suitability of a scheme that combined real-time data augmentation and class weighting for handling class imbalance. To accelerate further finetuning of more layers, separate top models were created and trained with images read in through the 'central-crop' approach. The architecture of the ResNet50 model lends itself to finetuning in steps of blocks. For varying learning rates and dropout rates, the best obtained results for training up to four blocks deep in the model are reported in Table VII. An increase in AUC-score was obtained by adding an extra trainable block to the process (as compared to only training the softmax layer) (AUC-score of 0.822 $\pm$ 6.39 e-4) and this could be maintained when adding two additional blocks. However, no further increases in performance could be reached and

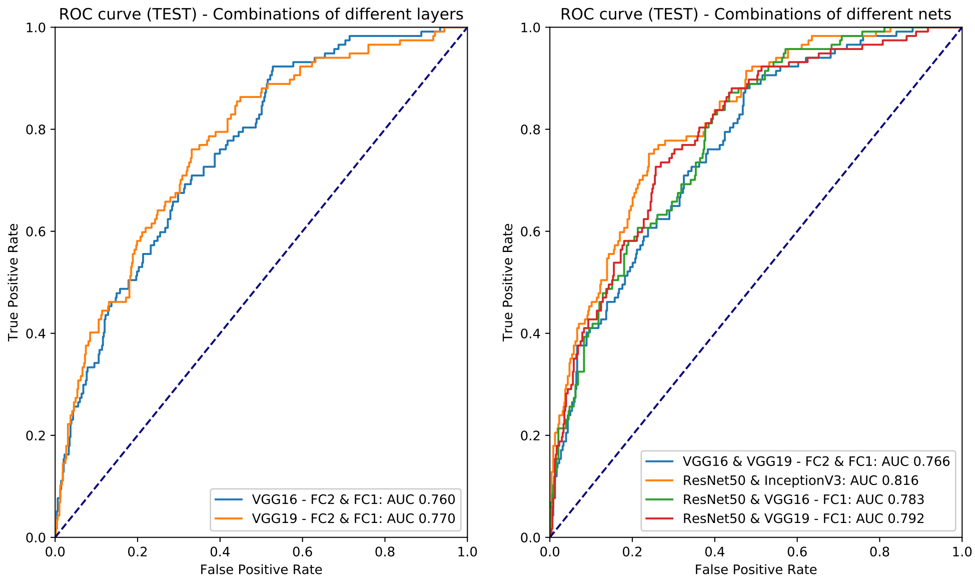

Fig. 2. ROC-curves and AUC-scores on test set for SVM classifier on feature combinations from different layers and different models.

TABLE V
TEST RESULTS SVM CLASSIFIER ON FEATURES EXTRACTED FROM DIFFERENT PRETRAINED MODELS.

CNN	Layer	# feat.	F1	Prec.	Recall	AUC
VGG16	FC2	4096	0.444	0.463	0.427	0.748
VGG16	FC1	4096	0.302	0.571	0.205	0.767
VGG19	FC2	4096	0.436	0.489	0.393	0.744
VGG19	FC1	4096	0.276	0.600	0.179	0.774
ResNet50	flatten_1	2048	0.520	0.482	0.564	**0.809**
InceptionV3	avg_pool	2048	0.457	0.396	0.538	0.729

TABLE VI
TEST RESULTS SVM CLASSIFIER ON FEATURE COMBINATIONS FROM DIFFERENT LAYERS AND DIFFERENT MODELS.

CNN	Layer	# feat.	F1	Prec.	Recall	AUC
ResNet50 & VGG19	flatten_1 & FC1	6144	0.299	0.622	0.197	0.792
ResNet50 & InceptionV3	flatten_1 & avg_pool	4096	0.510	0.485	0.538	**0.816**

decreases in learning rate were required to stabilize the training process. Finetuning of a total of four blocks led to a decrease in performance (at least in an acceptable time-window of 4000 epochs). This thus portrays a limited, but existing window of opportunity to enhance performance on the skin lesion dataset when finetuning starting from a specific depth in the ResNet50 model.

B. Convolutional auto-encoders for pretraining on unlabeled skin lesion datasets

Supervised training results for the three different top models on AE_1 are documented in Table VIII; the unlabeled dataset appeared highly sufficient to train AE_1 with three convolutional stages in its encoder and mirrored in its decoder. Therefore, the deeper convolutional auto-encoder model (AE_2) was also explored and hypothesized to be trainable with the unlabeled dataset. As stable training results were quickly obtained with an identity block-based top model, we chose to only explore the hyperparameters for such a top model stacked on AE_2. The best results are also documented in Table VIII. A visual comparison is portrayed in Figure 3.

Investigating the use of CAE's for (partial) pretraining of a CNN architecture thus indicated adequate performances in terms of AUC-scores in the case of stacking with an identity or convolution block-based top model rather than a dense output layer. The highest scores were obtained with a slightly deeper AE-model (AUC-score of 0.7625 ± 6.66 e-4), which proved to be trainable with the available unlabeled data within an acceptable time frame. Given the time constraint on this study, complete finetuning of the obtained CNN model (pretrained encoder + trained top model) could only briefly be touched upon and further exploration of hyperparameter settings is required, but further increases in AUC-scores are expected.

TABLE VII

BEST RESULTS FOR THE DEPTH OF FINETUNING, IN TERMS OF THE AMOUNT OF BLOCKS THAT WARE INCLUDED IN THE SUPERVISED TRAINING, WITH INDICATION OF THE APPLIED LEARNING AND DROPOUT RATE (* RESULTS DOCUMENTED AS MOVING AVERAGE FOR A WINDOW OF 5 EPOCHS)

Depth of finetuning	Lr.	Drop rate	AUC*
Softmax output layer	1 e-5	0	0.806 $\pm$ 2.49 e-4
Including Block 1	1 e-6	0.1	**0.822 $\pm$ 6.39 e-4**
Including Block 2	1 e-6	0.3	0.819 $\pm$ 1.94 e-4
Including Block 3	5 e-7	0.2	0.818 $\pm$ 4.02 e-4
Including Block 4	1 e-8	0.1	0.779 $\pm$ 1.79 e-4

TABLE VIII

COMPARISON OF BEST SUPERVISED TRAINING RESULTS FROM DIFFERENT TOP MODELS ON BOTH AE_1 AND AE_2 (INVESTIGATED UNDER HYPERPARAMETERS OF LEARNING AND DROPOUT RATE).

Case	Lr.	Drop rate	AUC
AE_1 and dense top	1 e-5	0.2	0.637 $\pm$ 9.05 e-4
AE_1 and identity top	1 e-5	0.2	0.7371 $\pm$ 9.97 e-4
AE_1 and convolution top	5 e-6	0.025	0.7362 $\pm$ 4.94 e-4
AE_2 and identity top	5 e-6	0.05	**0.7625 $\pm$ 6.66 e-4**

IV. CONCLUSIONS

To conclude, we indicate in Table IX some of the most prominent results achieved for the various settings that were explored w.r.t. efficient pretraining of CNN architectures, opting for paths in the direction of transfer learning or auto-encoders. At all stages, class imbalance handling and overfitting was extensively considered. For the case of transfer learning, training SVM's on combinations of features from different nets (especially ResNet50 and InceptionV3) increased performance beyond classification results on stand-alone features. Improvements in these results were obtained through adaptation of the ResNet50 model to the skin lesion classification task and finetuning at least one additional block in the model. This has proven to be the most optimal scheme for the skin lesion classification task under consideration.

Finally, even though only a limited amount of pretraining was performed via auto-encoders (due to time-constraints in this study), the supervised learning phase of a top-model stacked on the pretrained encoder-base indicated AUC-score exceeding what was achieved with SVM's trained on FC2-features from the VGG-nets and the final features from the InceptionV3 model. The results for this scheme are almost in the same ballpark as what was achieved by the second the best feature + SVM scheme (i.e. VGG19, FC1-features). Further exploration of CAE's is therefore desired, both in terms of training time of the auto-encoder an

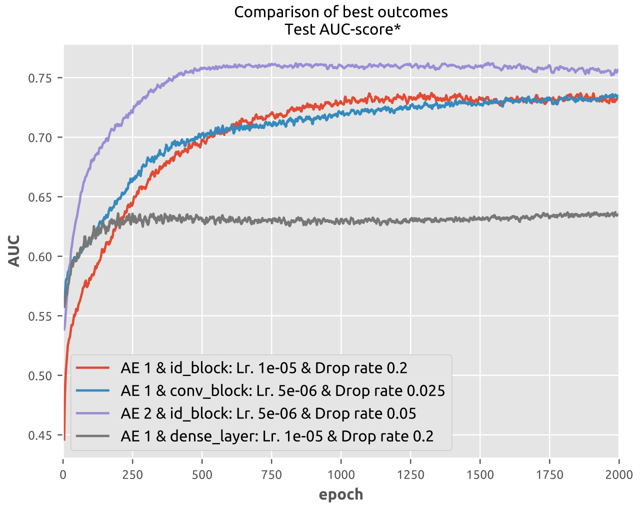

Fig. 3. Comparison of best supervised training results from different top models on both AE_1 and AE_2.

finetuning of the final CNN, as even this limited exploration of the framework indicates the a window of opportunity for performing pretraining of CNN architectures through CAE's.

ACKNOWLEDGMENT

The data used in this work were provided by the International Skin Imaging Collaboration (ISIC) [10]. The author wishes to thank the supervisor, Prof. A. Pizurica, and the counselor, L. Meeus, for their constructive and guiding assistance, which was indispensable to the positive completion of this work.

REFERENCES

[1] Apalla Z, Nashan D, Weller RB, Castellsagu X. Skin Cancer: Epidemiology, Disease Burden, Pathophysiology, Diagnosis, and Therapeutic Approaches. Dermatology and Therapy. 2017;7(Suppl 1):5-19. doi:10.1007/s13555-016-0165-y.
[2] EuroMelanoma. België heeft een huidkanker probleem - Persbericht EuroMelanoma. 2018. www.euromelanoma.org/belgie
[3] Fabbrocini G, Triassi M, Mauriello MC, et al. Epidemiology of Skin Cancer: Role of Some Environmental Factors . Cancers. 2010;2(4):1980-1989. doi:10.3390/cancers2041980.
[4] Matsumura Y, Ananthaswamy HN. Toxic effects of ultraviolet radiation on the skin. Toxicology and Applied Pharmacology. 2004;195(3):298-308. doi.org/10.1016/j.taap.2003.08.019.
[5] Didona D, Paolino G, Bottoni U, Cantisani C. Non Melanoma Skin Cancer Pathogenesis Overview. Biomedicines. 2018;6(1):6. doi:10.3390/biomedicines6010006.
[6] Gray-Schopfer V, Wellbrock C, Marais R. Melanoma biology and new target therapy. Nature. 2007. 445: 851-7.
[7] American Cancer Society. Facts and Figures 2018. 2018. www.cancer.org/research/cancer-facts-statistics/all-cancer-facts-figures/cancer-facts-figures-2018.html
[8] Korotkov K, Garcia R. Computerized analysis of pigmented skin lesions: A review. Artificial Intelligence in Medicine. 2012. 56(2): 69-90
[9] Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, Thrun S. Dermatologist-level classification of skin cancer with deep neural networks. Nature. 2017. 542: 115-8. doi.org/10.1038/nature21056
[10] ISIC Archive. https://isic-archive.com/
[11] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. CoRR. vol. abs/1409.1556. 2014

TABLE IX

COMPARISON OF THE DIFFERENT SETTINGS EXPLORED THROUGHOUT THIS STUDY. WHERE APPLICABLE, THE AUC-SCORES WERE AVERAGED OVER A WINDOW OF 5 EPOCHS AND BOTH MEAN AND STD ARE REPORTED.

Case	**Hyperparameters**	**AUC**
SVM on pretrained VGG19-features	FC1 - features	0.774
SVM on pretrained ResNet50-features	flatten_1 - features	0.809
SVM on combination of pretrained ResNet50 and InceptionV3	flatten_1 from ResNet50 and avg_pool from InceptionV3	0.816
Finetuning ResNet50	Supervised training of pretuned softmax layer and Block 1	0.822 ± 6.39 e-4
Pretraining with CAE-framework	Training of AE_2 on unlabelled data and supervised training of identity-block based top model	0.7625 ± 6.66 e-4

[12] Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. CoRR. vol. abs/1409.4842. 2014

[13] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. CoRR. vol. abs/1512.03385. 2015

[14] Lemaitre G, Nogueira F, Aridas CK. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. Journal of Machine Learning Research. 2017. 18(17):1-5.

[15] Krizhevsky A, Sutskever I, Hinton G. ImageNet classification with deep convolutional neural networks. Proc. Advances in Neural Information Processing Systems. 2012. 25:1090-1098

[16] Kingma DP, Ba J. ADAM: A method for stochastic optimization. CoRR. vol. abs/1412.6980. 2014

[17] Chollet F. Deep Learning Models: ResNet50. https://github.com/fchollet/deep-learning-models/blob/master/resnet50.py

Table of Contents

List of Tables

LIST OF FIGURES

FIGURE **Page**

List of abbreviations

AE	Auto-Encoder
AP	Action Potential
AUC	Area-Under-Curve
BCC	Basal Cell Carcinom
CAD	Computer-Aided Diagnosis
CAE	Convolutional Auto-Encoder
CNN	Convolutional Neural Network
CV	Cross-Validation
EPSP	Excitatory Post-Synaptic Potential
FC	Fully Connected
FPR	False Positive Rate
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IPSP	Inhibitory Post-Synaptic Potential
ISIC	International Skin Imaging Collaboration
Lr.	Learning rate
ML	Machine Learning
MSC	Melanoma Skin Cancer
NMSC	Non-Melanoma Skin Cancer
PCA	Principle Component Analysis
RBF	Radial Base Function

ReLu	Rectified Linear Unit
RF	Receptive Field
RGP	Radial Growth Phase
ROC	Receiver Operating Characteristic
ROS	Random OverSampling
ROS	Reactive Oxygen Species
SCC	Squamous Cell Carcinoma
SGD	Stochastic Gradient Descent
SMOTE	Synthetic Minority Over-Sampling Technique
STD	Standard Deviation
SVM	Support Vector Machine
TPR	True Positive Rate
UVR	UltraViolet Radiation
VGP	Vertical Growth Phase

List of symbols

E	objective function/cost function
Ω	penalty
T_s	training set
$\boldsymbol{x}$	training sample
c	class label
$\boldsymbol{w}$	model weights
b	model bias
ρ	margin
L	Lagrangian
ϵ	slack variable
K	kernel
φ	activation function
v	summed neuronal input
y	neuronal output
Θ	model/system parameters
δ	errors to be backpropagated

Chapter 1

Introduction

In the ever changing field of medicine, clinicians are eternally trying to improve their diagnostic skills. As a proper diagnosis is indispensable for further health care decisions, where errors may lead to wrongful treatment, postponed treatment and/or physiological/psychological damages, ameliorating the quality and accessibility of diagnosis formation can be described as one of the key objective in this fields [1]. The diagnostic process constitutes a complex and multidisciplinary task in which collaboration is vital. It was thus inevitable that the rising popularity and accessibility of computers would lead to the introduction of said systems in the medical field. The first examples of this can be found in the area of radiology in the 1960s, where focus was initially placed on creating an automated computer analysis of X-ray images [2], [3], [4]. These systems however did not only lack in quality due to the limited capabilities of computers and imaging processing techniques, it also rapidly became clear that no single computer system could replace a clinician. Rather, a trend was observed to move from a computer automated diagnostic process towards a system to support clinicians in their diagnostic process, which is known as computer-aided diagnosis (CAD) [5].

Nowadays, CAD systems designed on a variety of machine learning (ML) approaches are actively being used in a number of medical areas. Applications include, but are not limited to, detection of breast cancer, lung cancer, certain brain pathologies (e.g. brain tumors and pathological hallmarks of Alzheimer's disease) and diabetic retinopathy. The use of machine learning is of course not restricted to diagnostic processes, but further extends to image processing, personalized treatment, electronic health records, big data studies (e.g. genome-wide association studies or GWAS), drug discovery and robot surgery. The list keeps expanding.

With regard to image-based applications, the recent growth in strength and popularity of deep learning can also be portrayed in applications in the medical world. Deep learning relies on a multi-stage processing of images to extract increasingly more complex features from them. As the features get extracted at higher-levels, they tend to progressively represent more distinctive and integrated characteristics of the image, thus resulting in high performances for e.g. image classification.

1.1 Objectives

This work will focus on the use of deep learning for skin cancer classification purposes. Rises in skin cancer, both benign and malignant, have been observed in the Belgium, with over 37 000 cases reported in 2017.[1] Skin cancer incidence is furthermore expected to rise by another 5 to 9 % per year, depending on the type of skin cancer [6]. As will be explained in reference to literature in Chapter 3, melanoma, representing only about 1 % of all skin cancer cases, is the most dangerous form given its propensity to metastasize. Early detection and resection of melanoma, offers the. patient excellent 5-year survival rates, but this drops severely as the melanoma progresses into the regional or distant stage. In the current diagnostic practice for skin lesions, physicians rely on subjective systems by looking at specific characteristics of the lesions. This renders the diagnosis highly dependent on the physician's training, previous experience and interpretation. Certainty is only obtained upon performing a biopsy, with many lesions thus being biopsied unnecessarily. In light of the need for early and accessible detection to improve survival chances as well as the high variability in the diagnostic process, instigating the performance of biopsies for finality, there is a clear window of opportunity for the use of CAD.

Machine learning for skin lesion classification has long been a 'hot' topic with many attempts at improving the diagnostic performances of such systems [7]. Past approaches have been incorporating two main steps: (1) hand-designing a feature extractor after image processing and (2) training a classification algorithm on these features. The bottleneck in these approaches is clearly the strength of the hand-crafted feature extractor, which tend to imitate the diagnostic tools physicians use to visually assess lesions. However, the high variability in diagnostic performances of physicians portrays that they might not be able to establish the most characteristic features of skin lesions. Therefore, recent trends in research have portrayed a shift towards deep learning in this domain, with the aim of letting the algorithm itself now decide on features, possibly discovering ones that humans have not been able to notice or think of.

The use of deep learning for skin lesion classification will be the main focus point of this work. Working with a limited and unbalanced labeled dataset, different deep learning approaches will be compared in their performance for obtaining a differential diagnosis of melanoma. In a first line of research, the use of transfer learning on this dataset will be explored. Secondly, given the large, unlabeled datasets of skin lesions that are publicly available, an exploration of the use of auto encoders as alternative pretraining scheme will be investigated.

[1]In comparison, there were around 67 000 invasive tumors of other natures observed in 2017 in Belgium [6].

1.2 Structure

This work is subdivided in different topics:

- *Chapter 2: Deep Learning* aims to provide the reader with the general concepts of deep learning. Given the extent of literature existing around this scientific area, only a small scoop can be tackled here. In light of the biological nature of the application area in this work, the machine learning notions most relevant to this study will be introduced alongside their parallels in neuroscience, as a clear synergy can be observed between both research fields. Furthermore, a brief overview of benchmark performances of deep neural networks is presented alongside the principles of transfer learning for image classification. Finally, autoencoders are touched upon as alternative route of pretraining a neural network.
- *Chapter 3: Skin Cancer Classification* presents a broad overview of skin physiology and skin cancer pathophysiology, to provide the reader with the necessary biological insights in the problem. Furthermore, an exploration of relevant literature details what has been done with machine learning in the field of skin cancer classification. The chapter then concludes with an examination of current challenges in deep learning approaches for skin lesion classification and introduces the open lines of research.
- *Chapter 4: Methods* delineates more in detail the specific research objectives and research questions of this work. It further discusses the datasets, preprocessing steps, machine learning approaches and performance measures used to investigate said questions.
- *Chapter 5: Results & Discussion* presents the results obtained throughout this work and comments on said results as well as the challenges and practical possibilities for overcoming them.
- *Chapter 6: Conclusion* reports a summary of the accomplished work as well as a brief exploration of future research opportunities.

1.3 Main contributions

In light of the popular research area that targets the development of CAD-systems for skin lesion classifications, numerous research studies have been published concerning this topic. However, different studies tackle individual deep learning applications and delve into specific aspects of them. Furthermore, even though in the past the majority of research has been focused on the use of supervised learning on labeled datasets, some very recent research starts to incorporate the use of large, unlabeled skin lesion databases.

Given the numerous different factors at play with respect to datasets and techniques, a more global, comparative overview becomes somewhat more strenuous to obtain. This work aims to explore the effectiveness of different supervised learning methods on a limited, labeled dataset, their performance w.r.t.

overfitting and imbalanced classes, as well as the use of unlabeled data to improve performance. Thus, we hope to provide a more structured horizontal comparison between different routes for classification and dealing with often-encountered problems in this field.

The research in this works aims to contribute to the following key aspects of the use of deep learning for skin lesion classification:

1. *Evaluating the best-performing transfer learning scheme for skin lesion classification*

   Different pretrained model will be compared in their performance on the skin lesion dataset. Furthermore, the effectiveness of using the pretrained model as a feature extractor and building a classifier on top will be compared to finetuning the pretrained model for the task at hand. For both cases, a sub-research question will be further investigated. For the former, the effect of merging features from different pretrained models is examined; for the latter, the manner and depth of finetuning is studied.

2. *Exploring different strategies to cope with class imbalance and early overfitting on the limited dataset*

   Different techniques of data augmentation, both in the data and feature space will be explored, penalizing misclassification in underbalanced classes by incorporating class weights during training of the model and proper evaluation measures will be explored for the class imbalance problem. Furthermore, the effects of data augmentation and dropout as measures to prevent overfitting will be established.

3. *Proposing a convolutional auto-encoder to exploit the use of unlabeled data*

   Two convolutional auto-encoder models, one shallow and one more deep model, are trained on the unlabeled data. Different top-models stacked on the encoder are explored to investigate whether this unsupervised, pretraining scheme may improve classification performances.

Chapter 2

Deep learning

The ideas put forward by Alan Turing in the midst of the 20th century with regard to the rise of machine learning are increasingly gaining momentum. The (total) Turing Test states that an artificial system can be considered intelligent if it can interact with a human (either in written manner or combined with visual simulations), without the human being able to know the nature of the system (be it a machine or an actual human) [8]. As machine learning becomes thus embedded in our society, with applications ranging from online search suggestions to chatbots, being used for personalized marketing purposes, entering the financial world for stock predictions, etc., it has also managed to make its way into the medical world [9]. For purposes of computer-aided diagnosis and personalized treatment planning, machine learning is treading a clear path forward. Given their current popularity, it is however important to note that these, undeniably powerful, techniques should however be handled with care and understanding, which is even more imperative in the medical world. Therefore, this chapter has as purpose to ease the reader into the concepts of deep learning to improve understanding and point out the critical aspects in dealing with machine learning techniques.

The chapter commences with a brief exploration of general machine learning notions and techniques. In light of the biological nature of the problem, it is interesting to explore the existing synergy in research between the fields of neuroscience and machine learning. This enables us to introduce the concepts of deep neural networks. From this discussion, the need for other deep learning techniques will naturally surface and subsequently be explored. It is natural that given the years of extensive research in these fields and the broad range of topics it encaptures, only a brief representation can be given in this work. Therefore, this chapter aims to introduce all necessary concepts to guide the way towards the applications for skin lesion CAD. As the goal entails an image classification problem, all concepts will be translated to this level.

2.1 Machine learning

2.1.1 Introduction to machine learning

Machine learning can be seen as a research area where data about the world are introduced to a computer or acquired through interactions with said world and upon receiving this, the computer is able to learn and generalize from this to obtain knowledge [10]. Computational learning in itself was defined by Mitchell [11] as: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Task The task at hand typically entails mapping a certain input, that is defined through a collection of features (either raw or extracted), to a desired output. Translating this in an image classification task, the describing features would be the pixel-values (or some visual characteristics in the case of feature extraction) and the target is a certain label. The strength of machine learning lies in its ability to discover mappings between inputs and outputs that cannot be statically coded or discovered by humans [10]. A bottleneck in such tasks is the quality of the features to define an example, which used to be hand-crafted and therefore strongly relied on the experience of the researcher. Recent emergence of deep learning techniques allows the learning algorithms themselves to discover features that can aid the task at hand [12].

Performance Machine learning algorithms are typically trained on a set of N examples $\{\mathbf{x_1}, ..., \mathbf{x_N}\}$, referred to as the training set, with as result an input-output mapping f($\mathbf{x}$). Subsequently, a quantitative measure is put in place to assess how well the algorithm performs. Typical performance measures are accuracy, i.e. proportion of correctly mapped samples, and error rate, i.e. proportion of samples for which the model has produced the wrong output. However, numerous measures exist and the choice of adequate performance measures is often task-dependent. To allow for an unbiased assessment of the algorithm, it is crucial that the model is assessed on an unseen dataset, usually referred to as the test set.

Experience The 'experience' through which an ML algorithm can learn a certain task may differ depending on the available data introduced throughout the training process [13].

- *Supervised learning:* By far the most popular approach, supervised learning consists of training the ML system on a labeled dataset. The task at hand consists of mapping an example $\mathbf{x}$ to a target output t. In the case of discrete target outputs, this is referred to as a classification task with class labels c as outputs, whereas for continuous outputs, this is known as regression.

- *Unsupervised learning:* The ML algorithm is in this case fed an unlabeled dataset and must infer knowledge from this data, typically looking at the hidden structure or representations in the data. Popular examples include clustering, to group similar datapoints in clusters, and density estimation, which targets the discovery of the probability distribution of the data in the input space.

- *Reinforcement learning:* In this case, the system only receives unlabeled data. However, through interactions with an environment, the system receives either positive or negative feedback on its performance upon which it may adapt.

- *Semi-supervised learning and self-taught learning:* On the borderline between supervised and unsupervised learning problems, the technique of semi-supervised learning deals with partially labeled datasets. Being costly and time-consuming to acquire, labeled datasets are often limited in size, in contrast to often large unlabeled datasets that are freely available. Therefore, the aim of semi-supervised learning is to supplement the limited labeled data with unlabeled data to improve learning. Even broader are the self-taught learning techniques [14], which supplement with unlabeled data that is not presumed to be from any of the classes of the labeled data.

Model building Building an ML algorithm or model during the training phase is typically approached in the same manner: data are fed to the model, a cost function is established and the model will adapt by changing its internal parameters to optimize the cost. Often, the aim is to the maximize the likelihood that the established model gives rise to the presented data (unsupervised) or the likelihood of the model mapping the given inputs to the target outputs (supervised). In the case of supervised learning, this can be translated into a minimization of the negative log-likelihood as cost function. Alternatively, another popular optimization procedure is the least-mean square rule, which targets the minimization of the sum of squared errors between model outputs and targets.

Model selection and evaluation Generally, a model is not only characterized by its internal parameters $\mathbf{p}$ (e.g. means, covariance matrices, weights), but is more correctly described as a function $m(\mathbf{p}, \mathbf{h})$ with $\mathbf{h}$ the hyperparameters (e.g. the type of model, the input-features, the dimensionality). If all model hyperparameters are fixed, the training phase optimizes the parameters p and the model is subsequently evaluated on the test dataset. However, in the case where only a limited amount of data is available, cross-validation is typically used to exploit the acquired data as best as possible [15]. This technique relies on the splitting of the dataset in k folds (k-fold cross-validation) and iteratively leaving 1 fold out, training the model on the other k-1 folds and testing on the left-out fold. In the case of k = N (N being the number of training examples), this is known as leave-one-out cross-validation. A performance estimation is subsequently obtained by averaging the obtained performances over the different iterations. The model can then be trained on the complete dataset, with the averaged performance documented as indication of test performance.

If on the contrary also the hyperparameters of the model have to be established, this necessitates the construction of another, separate dataset, the validation set. The different models $m_1 = m(\mathbf{p}, \mathbf{h_1})$, $m_2 = (\mathbf{p}, \mathbf{h_2})$, ..., $m_j = m(\mathbf{p}, \mathbf{h_j})$ are trained on the training set, evaluated on the validation set and the model that generalizes best on the validation data, according to a chosen model-selection rule (e.g. minimal validation error), will present the optimal hyperparameters $\mathbf{h}^*$. Subsequently, the model parameters are obtained from training on the merged training and validation set and the model performance is assessed on the test set, resulting in an unbiased indication of model performance. In the complication of small datasets, a nested cross-validation scheme is used. This technique aims to assess model performance on outer folds, while it mimics cross-validation on inner folds (obtained from the k-1 outer folds during each iteration) to select the best hyperparameters on inner folds.

Overfitting and underfitting The aim of learning is to build a model on the training set that also performs well on the test set. The extent to which a trained model performs well on unseen data is known as the generalization capability of the model. Given that the model parameters are fitted on a training set of limited size, two challenges can occur. If the model complexity is too high, the problem is likely underdefined, leading to a model that is perfectly fitted to the training data, but does not capture the necessary trends to extend to new test data, known as overfitting. Contrarily, in the case of underfitting, the model capacity might be too low to perform well on even the training set. Thus, for appropriate model design, a trade-off is required between model bias (underfitting leads to a high bias) and model variance (overfitting results in high variance) [16]. This aims for a model that is complex enough to capture underlying trends in the training data, but does not fit to noise and specifics of this data.

Regularization To prevent overfitting, one path of action lies in controlling the complexity of the model. This is done by extending the cost function (E) used to optimize the model to include a penalty term for model complexity. The amount of regularization in the modified cost-function ($E' = E + \lambda\Omega$) is determined by a hyperparameter λ [17]. Typical choices for the regularization term may target the mapping function y($\mathbf{x}$) through a penalization of e.g. high curvature, or the weights themselves, by controlling the L_1 or L_2 norm of the weights. The latter approach is motivated by the tendency of weights to increase as the model starts to overfit.

Feature selection and extraction Overfitting can also be prevented through using carefully selected features as training input, thus reducing the dimensionality of the input space. Feature selection targets to use only those features that are useful for capturing trends in the data. Alternatively, the process of feature extraction intends to project the training samples in the high-dimensional input space onto a lower dimensional space, still containing sufficient information, by the process of constructing strong new features from which a limited number can be selected. This projection is often performed through the means of unsupervised learning algorithms, which amongst others include the popular Principal Component Analysis (PCA). PCA minimizes the square error between the original data and the projection. It should be noted that in any machine learning problem dealing with a multi-dimensional input space, the input features should preferably be normalized to even out their importance.

Motivation for deep learning Classic machine learning algorithms perform well on numerous tasks, but portray a decay in generalization ability as the dimension of the problems increase. In the case of more complex problems, such as object recognition (computer vision), the classic approaches tend to fail in capturing the complex functions embedded in the high-dimensional space.

In the case of object recognition, it is apparent that instead of relying on the raw pixel values, the use of higher levels of abstraction (i.e. looking at specific structures inside the image) simplifies the classification task. In the past, this problem has been bypassed through splitting the task in two main operations. The first phase involves the construction of an intelligent feature extractor, which is highly task-specific and requires expert knowledge. Subsequently, the easier part of training a classifier on the extracted features can be performed. Deep learning is however able to discover higher levels of abstraction that are useful for classification without interference of the designer. This is also referred to as representation-learning and it is exactly where the strength of deep learning lies [12].

2.1.2 Support Vector Machines

To illustrate a general machine learning technique, the example of Support Vector Machines (SVM's) is given here in concise form, as this will be used in this work. Focusing first on a linearly separable binary classification problem, the basic concepts will be introduced. This can then be extended to non-linearly separable problems, as well as to multiclass classification problems. SVM's have also been applied on regression problems, for which we refer to [18], [19], [17].

Optimal separating hyperplane Considering a binary, linearly separable classification problem, the objective is to find a classifier that generalizes well to unseen data. As depicted in Figure 2.1, different linear classifiers, which can be conceptualized as hyperplanes separating the feature space, can separate the training data. Intuitively, the hyperplane that is expected to generalize best is the one that maximizes the *margin*, which is the distance between the hyperplane and its nearest datapoints of the different classes [1], known as the *support vectors* [17], [20].

Let a training set T_S containing N vectors belonging to two different classes,

$$T_S = \{(\mathbf{x_1}, c_1), ..., (\mathbf{x_N}, c_N)\}, \quad \mathbf{x} \in \mathbb{R}^D, c \in \{-1, 1\} \tag{2.1}$$

be separated by a hyperplane H($\mathbf{w}$, b) described by

$$\mathbf{w}^T \mathbf{x} + b = 0. \tag{2.2}$$

The corresponding decision function for classification is f($\mathbf{x}$) = sign($\mathbf{w}^T \mathbf{x} + b$). The hyperplane is said to be optimally separating if (1) all vectors are separated correctly and (2) the distance of the hyperplane

[1]Alternatively, the margin is referred to as the distance between the support vectors.

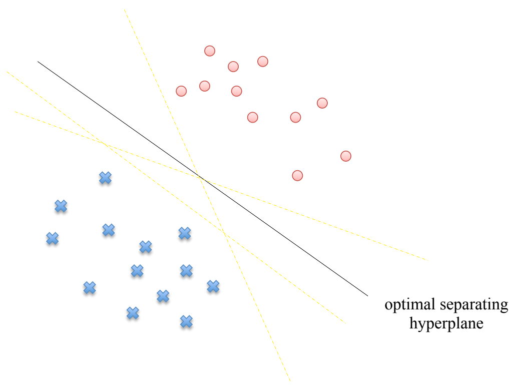

Figure 2.1: Optimal separating hyperplane for binary classification.

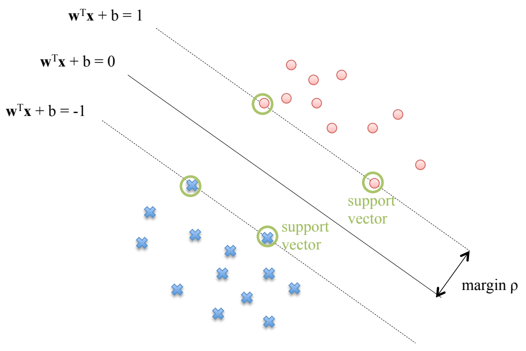

Figure 2.2: Maximum margin solution for linearly separable case.

to the support vectors is maximal. As **w** and b are not unique under these conditions, the choice for a canonical hyperplane [21] is commonly made where **w** and b must fulfill the condition

$$\min_n\left[c_n(\mathbf{w}^T\mathbf{x}+b)\right]=1 \quad \forall n=1,...,N \tag{2.3}$$

which entails a normalization of the weight vector. Thus, the optimal separating hyperplane must comply with the conditions of separating all vectors correctly

$$c_n\cdot(\mathbf{w}^T\mathbf{x_n}+b)\geq 1 \quad \forall n=1,...,N \tag{2.4}$$

and maximizing the margin ρ

$$\begin{aligned}\rho(\mathbf{w},b)&=\min_n[d(H,x_n)] \quad \forall n=1,...,N \qquad \text{with } d(H,x_n)=\frac{c_n(\mathbf{w}^T\mathbf{x_n}+b)}{||\mathbf{w}||}\\ &=\frac{1}{||\mathbf{w}||}\min_n[c_n(\mathbf{w}^T\mathbf{x_n}+b)]\\ &=\frac{1}{||\mathbf{w}||}.\end{aligned}$$

Consequently, the optimization problem $\max_{\mathbf{w},b}[\rho]$ reduces to maximizing $||\mathbf{w}||^{-1}$ or equivalently minimizing $||\mathbf{w}||^2$, as expressed in (2.5) where an extra factor 1/2 is commonly introduced, under the constraint of (2.4). [17]

$$\text{minimize} \quad \left[\frac{1}{2}||\mathbf{w}||^2\right] \tag{2.5}$$

This quadratic optimization problem under linear constraint can be solved through the introduction of a Lagrange multiplier (α_n) for each constraint: $\boldsymbol{\alpha}=(\alpha_1,...,\alpha_N)^T$. The Lagrangian function (2.6) should now be minimized with respect to the primal variables **w** and b and maximized with respect to the dual variables $\boldsymbol{\alpha}$.

$$L(\mathbf{w},b,\boldsymbol{\alpha})=\frac{1}{2}||\mathbf{w}||^2-\sum_{n=1}^{N}\alpha_n\cdot(c_n(\mathbf{w}^T\mathbf{x_n}+b)-1) \tag{2.6}$$

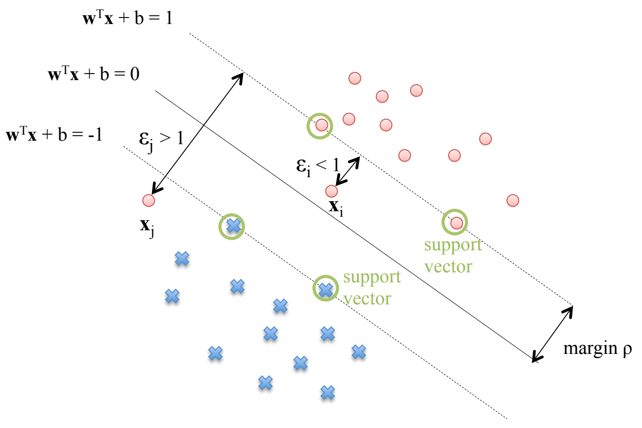

Figure 2.3: Optimal separating hyperplane for binary classification

As the solution is a saddle point of this Lagrangian, setting the derivatives of the Lagrangian with respect to the primal variables to zero results in the conditions: $\sum_{n=1}^{N} \alpha_n c_n = 0$ and $\mathbf{w} = \sum_{n=1}^{N} \alpha_n c_n \mathbf{x_n}$. The optimization can thus be rewritten by eliminating $\mathbf{w}$ and b:

$$\begin{aligned} &\text{maximize} \qquad L^*(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m c_n c_m (\mathbf{x_n}^T \mathbf{x_m}) \\ &\text{subject to constraints} \qquad \sum_{n=1}^{N} \alpha_n c_n = 0, \quad \alpha_n \geq 0 \quad \forall n = 1, ..., N \end{aligned}$$

Solving this dual problem for $\boldsymbol{\alpha}$, the solution for the primary problem ($\mathbf{w}$ and b) can be inferred from:

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n c_n \mathbf{x_n}$$

$$b = c_k - \sum_{n=1}^{N} \alpha_n c_n \mathbf{x_n}^T \mathbf{x_k} \quad \text{for any } \alpha_k > 0$$

where each non-zero α_n indicates that $\mathbf{x_n}$ is a support vector. The decision function can be reformulated as $f(\mathbf{x}) = sign(\sum_{n=1}^{N} \alpha_n c_n \mathbf{x_n}^T \mathbf{x} + b)$, where explicit computation of $\mathbf{w}$ is no longer required [17].

Generalized optimal separating hyperplane A more realistic binary-class dataset is in most cases not perfectly linearly separable. Whereas in the previous model the prerequisite was to classify each training vector correctly, this condition can be made more flexible through the introduction of a penalty term which assigns a higher cost the further the samples are on the wrong side of the margin boundary. This is expressed by the slack variables $\epsilon \geq 0$, typically a linear function of the distance to the margin boundary: $\epsilon_n = |c_n - (\mathbf{w}^T \mathbf{x_n} + b)|$. Thus, $\epsilon_n = 0$ indicates a sample on the margin, $0 < \epsilon_n \leq 1$ a correctly classified sample on the wrong side of the margin boundary and $\epsilon_n > 1$ a misclassified sample on the wrong side of the decision boundary, as depicted in Figure 2.3 [17]. The optimization problem can now

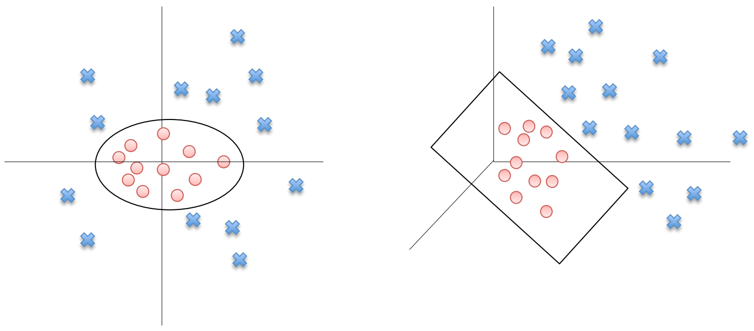

Figure 2.4: Mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$.

be formulated as a primary problem

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{n=1}^{N}\epsilon_n \\ &\text{subject to constraints} \quad c_n \cdot (\mathbf{w}^T\mathbf{x_n} + b) \geq 1 - \epsilon_n, \quad \epsilon_n \geq 0 \qquad \forall n = 1, ..., N \end{aligned}$$

where C can be considered as the inverse of a regularization parameter, controlling the trade-off between model complexity (margin) and training error (slack variable penalty). [22] Alternatively, Lagrange multipliers $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ can respectively be introduced for the constraints $c_n \cdot (\mathbf{w}^T\mathbf{x_n} + b) \geq 1 - \epsilon_n$ and $\epsilon_n \geq 0$, which allows the following dual problem statement:

$$\begin{aligned} &\text{maximize} \quad L^*(\boldsymbol{\alpha}) = \sum_{n=1}^{N}\alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m c_n c_m (\mathbf{x_n}^T\mathbf{x_m}) \\ &\text{subject to constraints} \quad \sum_{n=1}^{N}\alpha_n c_n = 0, \quad C \geq \alpha_n \geq 0 \quad \forall n = 1, ..., N \end{aligned}$$

This expression is very similar to the formulation for the linearly separable case, except for the boxed constraints on α_n. Again, a non-zero α_n indicates that $\mathbf{x_n}$ is a support vector. This problem is known as a soft margin classification problem, which includes an extra hyperparameter C to tune the amount of slack you allow. The larger the C, the more complicated the margin becomes, consequently increasing the chance of overfitting.

Generalisation to high-dimensional feature space Margin and soft-margin classification assume the data are approximately linearly separable. In the case of non-linearly separable data, the idea is to project the feature space to one of a higher dimension in which the data are (approximately) linearly separable. The aforementioned theory thus holds in the higher-dimensional feature space where the training data are now

$$T_s = \{(\phi(\mathbf{x_1}), c_1), ..., (\phi(\mathbf{x_N}), c_N)\} \quad \mathbf{x_n} \in \mathbb{R}^D, \phi(\mathbf{x_n}) \in \mathbb{R}^{D'}, c_n \in \{-1, 1\} \tag{2.7}$$

with transformation function $\phi(.)$ and $D' > D$. This is illustrated in Figure 2.4.

Table 2.1: Common kernels and their hyperparameters

Kernel type	**Inner product kernel**	**Hyperparameters**
Polynomial Kernel	$K(\mathbf{x_n},\mathbf{x_m}) = (\mathbf{x_n}^T\mathbf{x_m} + \theta)^d$	Degree of polynomial d and threshold θ
Gaussian Radial Base Function	$K(\mathbf{x_n},\mathbf{x_m}) = exp(-\frac{\|\|\mathbf{x_n}-\mathbf{x_m}\|\|^2}{2\sigma^2})$	Width σ^2 or alternatively $\gamma = \frac{1}{2\sigma^2}$
Sigmoid Kernel	$K(\mathbf{x_n},\mathbf{x_m}) = tanh(\eta\mathbf{x_n}^T\mathbf{x_m} + \theta)$	Only for specific values of η and θ

This non-linear mapping to a higher-dimensional feature space might introduce two problems to the optimisation problem. Firstly, the substantial increase in parameters to estimate in a higher-dimensional feature space may result in overfitting. This can be countered through the choice of an appropriate value for C in the soft-margin approach. Secondly, applying the transformation $\phi(\mathbf{x_n})$ on all vectors $\mathbf{x_n}$ becomes computationally demanding. Therefore, SVM's rather rely on the fact that samples only appear in inner products $(\mathbf{x_n}^T\mathbf{x_m})$ in the optimisation problem. Thus, a kernel $K(.)$ can be introduced to compute the inner product of vectors in the higher-dimensional feature space, where $K(\mathbf{x_n},\mathbf{x_m}) = \phi(\mathbf{x_n})^T\phi(\mathbf{x_m})$, without requiring the explicit transformation of all vectors separately. As the complexity of the problem now only depends on the dimensionality of the input space and not the higher-dimensional feature space, the input features can be mapped to an infinite dimensional space [23]. The dual problem can now be formulated as

$$\begin{aligned} \text{maximize} \quad & L^*(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_n\alpha_m c_n c_m K(\mathbf{x_n},\mathbf{x_m}) \\ \text{subject to constraints} \quad & \sum_{n=1}^{N} \alpha_n c_n = 0, \quad C \geq \alpha_n \geq 0 \quad \forall n = 1,...,N \end{aligned}$$

The corresponding decision function now becomes $f(\mathbf{x}) = sign(\sum_{n=1}^{N} \alpha_n c_n K(\mathbf{x_n},\mathbf{x}) + b)$, where a non-zero α_n is again indicative of $\mathbf{x_n}$ being a support vector [24], [25].

The kernel itself can be considered as a type of similarity measure between the two input vectors. Not all functions represent an inner product kernel function; Mercer's theorem states a necessary and sufficient condition on a function to be characterized as such. Commonly used kernels and their hyper-parameters are given in Table 2.1 [22], [23].

Generalisation to multiclass classification Two possibilities exist to extend the SVM classifier from a binary to a multiclass classification task. A single classifier can be build to solve the optimization problem in one go, whilst another technique splits up the optimization problem in several subproblems that have to be solved [26]. Of the latter methodology, two popular approaches are given here. Firstly,

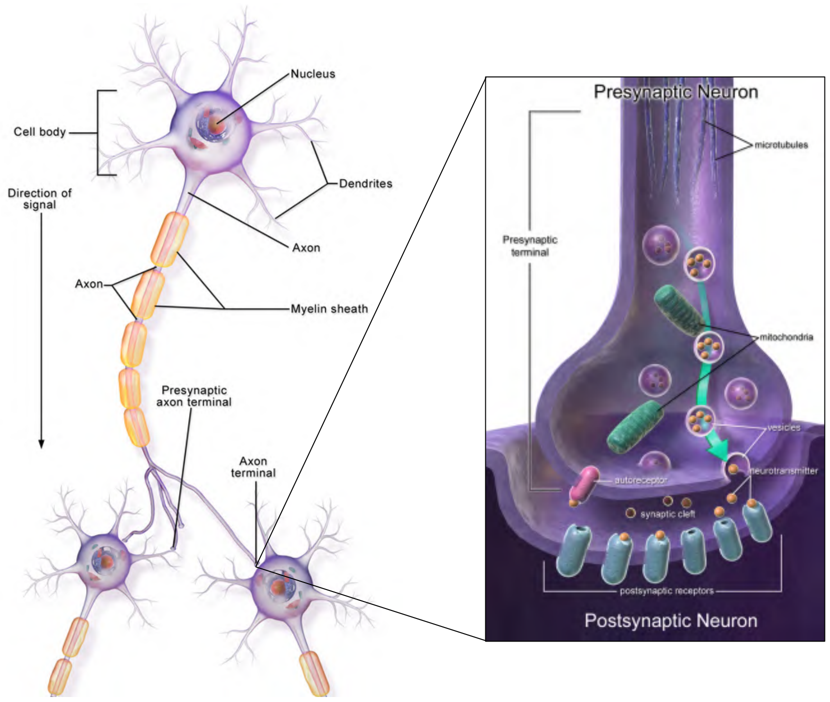

Figure 2.5: Structure of a neuron [27] with a detailed view of synaptic transmission [28].

the one-versus-all approach aims to build K different SVM classifiers for the K different classes that seperate the vectors of the K$^{\text{th}}$ class from the vectors in the K-1 other classes. Samples are subsequently classified in the class with the highest probability (i.e. which SVM places the sample the furthest in the positive region). Secondly, the one-versus-one approach considers all pairs of classes and constructs K(K-1)/2 different SVM's. Samples are assigned to the class that obtains the highest score, where each trained SVM's votes for one class [17].

2.2 Neural networks

2.2.1 From neurons to perceptrons

Neurons form the structural building blocks of the brain and can be considered as the primary information-carrying units. The pyramidal neuron depicted in Figure 2.5, shows the three main neuronal components: the dendrites, cell body and axon. Information flows along the neuron in the form of an electric signal, known as the action potential (AP). The neuronal membrane with a resting membrane potential of a value ranging from -90 mV to -40 mV depending on the type of neuron, gets depolarized by a sodium influx through voltage-gated sodium channels. Upon reaching the action potential threshold, often between -50 mV to -40 mV, an explosive influx of sodium occurs, followed by a repolarization due to a potassium efflux through slower-responding voltage-gated potassium channels [29]. These ion currents are driven by their electrochemical gradient. A refractory period is kept in place by the inactive state of the sodium channels, during which they will not respond to depolarization, thus preventing the signal from traveling backwards. The passing of an AP (nerve impulse) is referred to as the firing of a neuron. As a consequence of the way AP's are created, they are all equal and independent of stimulation strength. Information on the type of stimulus the neuron receives is instead encoded in the neuronal firing rate, with stronger impulses leading to higher frequency of action potentials [30].

Communication between different neurons is enabled by synapses, either electrical or chemical, the latter being the most common type and discussed here. In response to a depolarization, the pre-synaptic terminal will release neurotransmitters that are contained in presynaptic vesicles into the synaptic cleft by exocytosis. The neurotransmitter subsequently diffuse across the cleft and by binding to specific receptors, the neurotransmitters can incite a change in the postsynaptic membrane potential, thereby allowing the dendrites to receives information. Two of the most common types of neurons are the excitatory neurons, releasing glutamate and causing a depolarization or excitatory post-synaptic potential (EPSP) and inhibitory neurons releasing GABA, which cause a hyperpolarization or inhibitory post-synaptic potential (IPSP) [31].

Neurons generally receive their input at the dendrites, which are often covered with small appendages, known as dendritic spines. A pyramidial neuron in the CA1 region of the hippocampus has over 30 000 dendritic spines through which it may receive input from other neurons [32]. This physiological hallmark underlies the ability of the adult human brain to adapt. Through the formation of new dendritic spines, which may receive neurotransmitter through volume transmission or can connect to other neurons, thereby creating new synapses, or adaptation of existing spines in response to e.g. high-frequency stimulation of the pathway, plasticity in the human brain is enabled [33].

The presence of such large numbers of input-channels portrays the need for a strong synaptic integration process. Electric signals generated at postsynaptic terminals will spread along the membrane towards the cell body and eventually the axon hillock. Due to the behavior of cell membranes as an RC-transmission line, the passive diffusion will distort the signals. Furthermore, the individual PSP's generated at the numerous postsynaptic terminals are most often in the subthreshold range. In light of the large number of connections however, they are very likely to sum together in either time or space, thus evoking larger changes in the membrane potentials [34]. This summation of signals that were distorted due to their travels along the dendritic arborizations provides the means for neuronal information-integration [35]. As the main dendritic branches come together at the soma, this is were the final integration takes place. Subsequently, the signal goes to the axon inital segement (i.e. axon hillock), a region characterized by a high sodium-channel density. The corresponding local decrease of AP-threshold renders it a favored region for action potential initiation.

To construct a model for a neuron, the key aspects of biological neurons were taken into account and mimicked. Viewing the neuron as an information-processing unit and the AP as the signal, it must entail the following:

- **Input**: On average, a neuron receives inputs from 10^3 to 10^4 other neurons [36]. Therefore, the input will be a vector of signals $\mathbf{x} = [x_1, x_2, ..., x_n]$, with n denoting the vector length.
- **Weighting**: Synapses modify to which degree an AP can influence the membrane potential of the postsynaptic terminal through various means, e.g. type of neurotransmitter (evoking an EPSP or IPSP), variable amount of neurotransmitter release, variable size of the dendritic spine and modifications by volume transmission. Therefore, each signal will be multiplied by a certain weight to represent the strength of the synaptic connection [33]. A neuron k receives as input the outputs from a set of n other neurons, which are modified through a set of weights $\mathbf{w} = [w_{1k}, w_{2k}, ..., w_{nk}]$.
- **Integration**: The intricate process of summation of distorted signals throughout dendritic arborizations and finally at the soma will be simplified to a summation of the weighted inputs: $u_k = \sum_{i=1}^{n} w_{ik}x_i$.
- **Activation**: The integrated current will move towards the axon hillock, where an AP will be fired if the evoked change in membrane potential surpasses a certain threshold. Thus, an activation function φ indicates when a neuron will fire and confines the output range of said neuron. Often a bias term b_k is introduced to modify the input of the activation function, which now becomes $v_k = u_k + b_k$, with v_k representing the activation potential or local field of the neuron k [33]. Evoking the use of an activation function serves as a representation of the strong non-linear nature of the neuronal information integration process. Different types of activation functions can be used, with common choices represented in Table 2.2 [33]. The non-linear behavior of neurons can be encaptured by the use of a non-linear activation function, e.g. the logistic sigmoid or the hyperbolic tangent. Furthermore, it is important to remark that the activation function can also be interpreted as the probability of the neuron being in the 'on' state. Thus, a stochastic binary neuron can be in two states, either 'on' during which it fires (state +1) or 'off' (state -1).
- **Output**: The resulting output is a scalar $y_k = \varphi(v_k) = \varphi(u_k + b_k) = \varphi(\sum_{i=1}^{n} w_{ik}x_i + b_k)$. An alternative representation can be found by replacing the bias by an extra synaptic input term: $x_0 = 1$ with weight $w_{0k} = b_k$, rendering the output to be: $y_k = \varphi(\sum_{i=0}^{n} w_{ik}x_i)$.

From a historical point of view, the first artifical neuron model was implemented by McCullogh and Pitts [37] in 1943 as a hard-threshold classifier, firing when the linear combination of inputs exceeds a certain threshold. With the intent of creating a computational model of a boolean agent, they mimicked the neuronal all-or-none firing process and supplied the neuron with equal-sized excitatory (w=1) and inhibitory (w=-1) inputs. The way neurons were perceived changed however tremendously with the discovery by Donald Hebb in 1949 of an increasing strength in the connection between neurons that

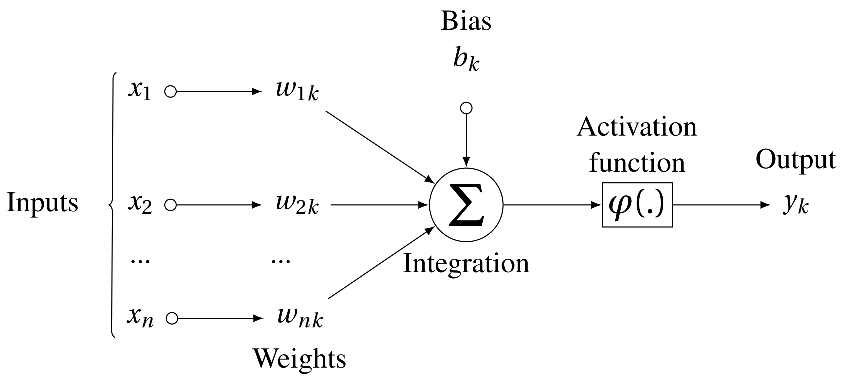

Figure 2.6: Schematic of a neuronal model: Neuron k receiving n neuronal inputs.

Table 2.2: Types of activation functions

Neuron	**Activation function**	**Output**
Linear neuron	$\varphi(v) = v$	$y = v_k$
Binary threshold neuron	$\varphi(v) = \begin{cases} 1, & \text{if } v \geq 0 \\ 0, & \text{if } v > 0 \end{cases}$	$y_k = \begin{cases} 1, & \text{if } v_k \geq 0 \\ 0, & \text{if } v_k > 0 \end{cases}$
Rectified Linear Neuron	$\varphi(v) = \max(v, 0)$	$y_k = \begin{cases} v_k, & \text{if } v_k \geq 0 \text{ (Linear)} \\ 0, & \text{otherwise (Hard decision)} \end{cases}$
Sigmoid neurons	$\varphi(v) = \frac{1}{1+exp(-v)}$	$y_k = \frac{1}{1+exp(-v_k)}$
Stochastic binary neurons	$\varphi(v) = P(v) = \frac{1}{1+exp(-v)}$	$y_k = \begin{cases} \text{State } 1, & \text{with probability P(v)} \\ \text{State } -1, & \text{with probability 1-P(v)} \end{cases}$

fire together. This concept of synaptic plasticity translated in the famous Hebbian learning rule [38]: "Neurons that fire together, wire together."

Spurred on by the work of McCullogh and Pits and the Hebbian learning rule, Rosenblatt [39] constructed the first perceptron in 1957. He proposed a framework for binary classification tasks, implementing both real-valued inputs and weights and including a learning rule. Originally intended as a machine for image classification rather than a program, it consisted of three types of units: the input sensory units (S), the associative units (A) and the response unit (R) which returns the weighted sum of the associative units, which is subsequently passed through a (h-)Heaviside function. The weights between the sensory layer and the associative layer were predetermined [36], but the weights of the branches connecting the associative units to the response units were variable. Rosenblatt came up with a way of training these latter weights of the connections to improve classification results. Every time the algorithm misclassified a sample, an update of the weight vector was performed moving the weight vector towards the feasible weight space, by decreasing the squared distance to all feasible weight vectors [40].

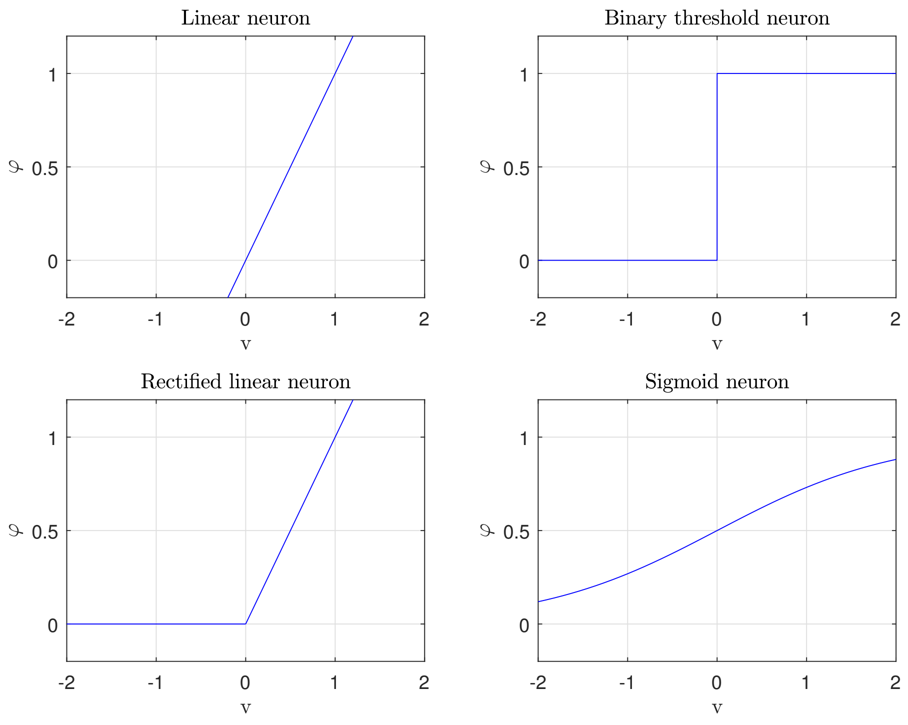

Figure 2.7: Activation functions of different types of neurons.

Simplified by Minsky and Papert in 1969 [41], the single-layer perceptron consisting of 1 layer of input nodes and 1 layer of output nodes emerged, making it the most simple type of neuronal net architecture. However, it soon became apparent that single-unit perceptrons could only handle linear classification tasks (e.g. impossible to learn the XOR function) [36]. Furthermore, the classification power of single-layer perceptrons is strongly dependent on the features it receives as input. To overcome the limitations of the single-layer perceptron, hidden layers consisting of non-linear neurons were added to the network architectures [42]. Said multi-layer perceptrons, which are more commonly known as multilayer or deep neural networks (DNN's), will be the main subject of the following section.

2.2.2 Architecture of feedforward neural networks

Architecture A general representation of a feedforward neural network is shown in Figure 2.9. Consisting of an input layer, which receives the input vector $\mathbf{x} = [x_1, x_2, ..., x_n]$, multiple hidden layers (of which one is shown here, with hidden nodes $\mathbf{z} = [z_1, ..., z_m]$) and an output layer with outputs $\mathbf{y} = [y_1, ..., y_l]$. Connections between a unit i in layer 1 and unit 2 in layer 2 are weighted with a factor $w_{ij}^{(1)}$, with superscript (1) indicating that the weights originate from units in the first layer. Each layer, with exception of the output layer, also contains its own bias term, notated here by the nodes x_0 and z_0 with bias weights. For clarity, the transformation formula from one layer to the next for the simplified

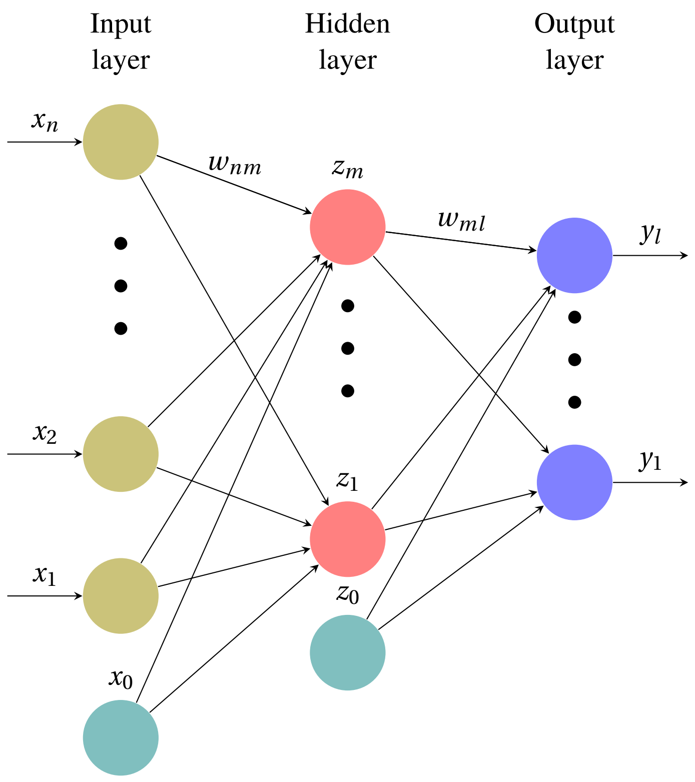

Figure 2.8: Architecture of a feedforward neural network.

network depicted in Figure 2.9 for a hidden unit is

$$z_j = \varphi(v_j) = \varphi(\sum_{i=0}^{n} w_{ij}^{(1)} x_i) = \varphi(\sum_{i=1}^{n} w_{ij}^{(1)} x_i + w_{0j}^{(1)}) \tag{2.8}$$

whereas for the output units this is

$$y_k = \sigma(v_k) = \sigma(\sum_{j=0}^{m} w_{jk}^{(1)} x_j) = \sigma(\sum_{j=1}^{m} w_{jk}^{(1)} x_j + w_{0k}^{(1)}) \tag{2.9}$$

in which the activation function of the output layer can differ from he previous ones.

Forward propagation The output of the network $\mathbf{y}$, at each output neuron y_k

$$y_k(\mathbf{x},\mathbf{w}) = \sigma\left(\sum_{j=0}^{m} w_{jk}^{(2)} \varphi\left(\sum_{i=0}^{n} w_{ij}^{(1)} x_i\right)\right) = \sigma\left(\sum_{j=1}^{m} w_{jk}^{(2)} \varphi\left(\sum_{i=1}^{n} w_{ij}^{(1)} x_i + w_{0j}^{(1)}\right) + w_{0k}^{(2)}\right) \tag{2.10}$$

is obtained by propagating the input $\mathbf{x}$ through the various layers of the net characterized by weights $\mathbf{w}$, explaining the name of a feedforward network [17]. The idea behind the propagation is that every layer corresponds to a different representation of the input vector, extracting increasingly more complex features from the input, until finally achieving a classification layer. A key characteristic of feedforward nets is the non-linear nature of their units (i.e. non-linear activation functions), which is employed to extend the capability of neural nets to solving non-linear classification tasks [33]. Furthermore, to enable training of the network (as will become clear in Section 2.2.3), continuous activation functions are opted for rather than the hard decision thresholds used by McCulloch & Pitts and Rosenblatt. A common choice for the activation function is the sigmoid or logistic function: $\varphi(v) = \frac{1}{1+exp(-v)}$. However, recent advances in convolutional neural network research show the importance of ReLu's as activation function [43]. The rectified linear unit combines a hard decision threshold with differentiability in its 'on'-domain and has been found to accelerate computations.

Output Using the neural network for supervised machine learning purposes requires some further thought to be given to the outputs of the net. The activation function for the output units follows naturally from the learning task at hand [17].

- *Regression*: In the case of a regression problem, the network should be able to approximate any continuous function in response to its inputs. Thus, the output unit is chosen as a linear neuron, where the identity function allows the output to reach any value.
- *Binary classification*: For a task that aims to assign any input sample **x** to either of two classes C_1 or C_2, the single output unit is represented by a non-linear neuron. A common choice is the sigmoid neuron. Under the assumption that an output y=1 represents a classification to class C_1, the sigmoid activation function can subsequently be interpreted as the probability of the input belonging to class C_1, i.e. $y(\mathbf{x}, \mathbf{w}) = p(C_1|\mathbf{x})$. By augmenting the number of output units, this methodology can be extended to the task of K separate binary classifications. In this case, the activation function of each of the K output units is the sigmoid function.
- *Multiclass classification*: For the case of a classification task involving K mutually exclusive classes, this information can be explicitly embedded in the neural network. One way to achieve this is by forcing the outputs of the net to sum to 1, rendering them interpretable as the class probabilities under a specific input sample **x**. The activation function of the output units is then chosen to be the softmax, resulting in outputs as represented in (2.11).

$$\sigma(\mathbf{z}) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.11}$$

2.2.3 Training multilayer neural networks

Having established the architecture of the neural network and the corresponding notations, the discussion can now proceed to the training of neural networks. As mentioned in the introduction about ML (Section 2.1.1), supervised machine learning techniques rely on an optimization of the parameters of the classifier through a comparison of the target output with what the classifier is actually doing. However, as the target outputs of the hidden units are unknown to the designer, this introduces a new difficulty in the training procedure of multilayer neural networks. The following section will give a general outline of the popular backpropagation algorithm to train multilayer neural networks and does so by building up to the complexity of the problem.

Notation Supervised learning for classification aims to construct a input-output mapping $f(.)$

$$f : (X, \Theta) \rightarrow Y \tag{2.12}$$

based on a labeled training set T_s,

$$T_s = \{(X, C)\} = \{(\mathbf{x_n}, c_n) : n = 1..N\} \tag{2.13}$$

where c_n represents the class label for training sample x_n. The system parameters Θ, which are in this case the weights and biases in the net ($\Theta = (\mathbf{w}, b)$), are determined through optimization of an objective function, which contains information on how well the system approximates the target output. The notation in this work will include the biases in the weights (i.e. x_0 input or z_0 hidden unit and w_0 weight for the bias), thus confining the parameters to be a set of weights $\mathbf{w}$.

Training scheme Different training schemes exist with regard to how the training data is handled [17].

- *Batch learning*: All training examples are processed, the objective function is evaluated and adjustments are made to the system parameters.
- *Online learning*: The objective function is evaluated after processing only one example and adjustments are made correspondingly. This is subsequently repeated for all training examples.
- *Mini-batch learning*: Instead of using one example, this method relies on a random subset of training examples, i.e. a mini-batch, to make adjustments to the system parameters.

Objective function A common choice for the objective function in the case of a regression or classification model is the mean-square error.

$$E(w) = \sum_{n=1}^{N} E(w)^n = \sum_{n=1}^{N} \sum_{k=1}^{K} \frac{1}{2}(y_k(x^n, w) - c_k^n) \tag{2.14}$$

However, interpreting the outputs of the neural network as probabilities opens the road to a new objective function by aiming to maximize the likelihood that the chosen model maps the input samples to the target labels, noted as $p(\mathbf{c}|\mathbf{x}, \Theta)$. This can be translated in the minimization of the negative log-likelihood, which is the cost-function known as the cross-entropy cost

$$E(w) = \sum_{n=1}^{N} E(w)^n = -\sum_{n=1}^{N} \sum_{k=1}^{K} c_k^n log(y_k(x^n, w)) \tag{2.15}$$

shown here for the case of multiclass classification with K mutually exclusive classes.

Model optimization Optimization of the model parameters to achieve the best possible classification results now reduces to finding a set of weight **w** that minimizes the error function E(**w**). Analytically, this would translate into finding the set of weights that minimizes the smooth and continuous error function, i.e. $\nabla E(\mathbf{w}) = 0$ (assuming the convex nature of the error function or under the second constraint of a positive, second derivative). However, visualizing the error space portrays that there are many local minima in which the gradient (nearly) vanishes, indicative of the non-convex character of the error function [17]. Thus, an analytical approach is very unlikely to lead to the optimal solution and iterative methods are adopted to tackle the problem. Many approaches exist to solve this non-linear optimization problem and typically rely on the idea of initializing the parameters at step zero to certain values $\mathbf{w}^{(0)}$ and updating them at each time step t. Different techniques exist for to implement this weight vector update.

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \Delta\mathbf{w}^{(t)} \tag{2.16}$$

Gradient descent algorithm A popular algorithm to tackle this iterative approach is the gradient descent algorithm. As the gradient indicates the direction of highest ascent, the aim to move towards the global minimum of the error function can be achieved by repeatedly moving in the negative gradient direction. The weight update at each step is thus

$$\Delta\mathbf{w}^{(t)} = -\varepsilon\nabla E(\mathbf{w}^{(t)}), \tag{2.17}$$

where ε signifies the learning rate. Batch gradient descent, which involves calculating the error function on the entire dataset, evaluating the gradient and updating the weights, does not deliver adequate results [17]. A more effective training scheme for large neural networks is that of stochastic gradient descent (SGD), which involves the evaluation of the gradient on one sample at a time (online learning) or on a subset of samples (mini-batch learning). At each iteration, these samples/subsets are randomly selected, assuming that the resulting weight updates behave like their batch opponents [44]. The main advantages of SGD over batch gradient descent are that it: (1) eliminates the heavy computational demand of evaluating the error function and corresponding gradient on the entire dataset, (2) becomes more time-effective, (3) takes into account redundancy in training set (where gradients for weights on two subsets may be very similar) and (4) decreases the risk of getting trapped in a local minimum of the global error function (as different samples/subsets will have different local minima) [17].

To illustrate this process, the simple example of training a sigmoid neuron ($y = \varphi(v) = \frac{1}{1+e^{-v}}$) is given through the process of stochastic gradient descent on the basis of a sum-squared error measure. The error on the output is in this case

$$E = \frac{1}{2}(c-y)^2 \tag{2.18}$$

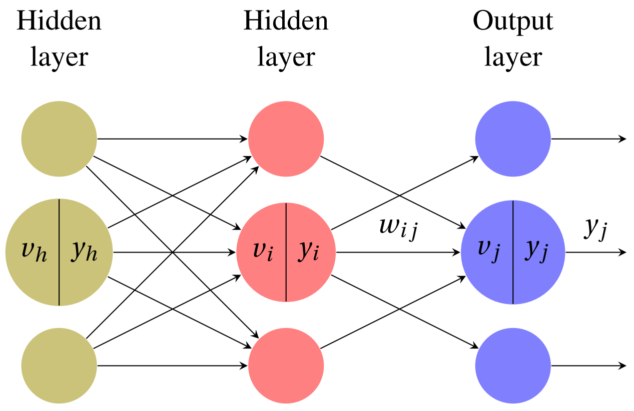

Figure 2.9: Illustration of error backpropagation through hidden layers of a neural network with explicit mention of the weighted sum of inputs to the neuron (v) and the output of the neuron ($y = \varphi(v)$). Two hidden layers are depicted, as well as the output layer, from which neurons are respectively denoted by (h), (i) and (j). For clarity, not all weighted connections between to the deepest hidden and final layer are portrayed.

where we only consider one training example. The error with respect to the summed input v can be calculated with the chain rule,

$$\frac{\partial E}{\partial v} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial v} = \frac{\partial E}{\partial y} y(1-y) = -(c-y)y(1-y) \tag{2.19}$$

knowing that the derivative $\frac{\partial y}{\partial v}$ of the sigmoid function is $y(1-y)$. The resulting weight update is thus equal to $\Delta w_i = -\varepsilon \frac{\partial E}{\partial w_i}$, where

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial z}\frac{\partial z}{\partial w_i} = \frac{\partial E}{\partial z} x_i = -(c-y)(1-y)yx_i. \tag{2.20}$$

The algorithm can now be extended to multilayer neural networks. The mathematics are described for the updates of weights w_{hi} and w_{ij}. For the weights w_{ij} connecting the output layers to the deepest hidden layer, the methodology is straightforward. However, the difficulty arises in determining the weight update of the weights between the two hidden layers. Drawing parallels with what has been done, there is a need to calculate the partial derivative of the error with respect to the output of hidden neuron i (2.27). To make the concept more concrete, the equations are assessed for the case of sigmoid neurons with a sum-squared error measure.

(2.21) Objective function E

$$E = \sum_{j \in output} \frac{1}{2}(c_j - y_j)^2 \tag{2.22}$$

$$\delta_j \triangleq \frac{\partial E}{\partial v_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial v_j} \tag{2.23}$$

$$\frac{\partial E}{\partial v_j} = \frac{\partial E}{\partial y_j}(1-y_j)y_j = -(c_j - y_j)(1-y_j)y_j \tag{2.24}$$

(2.25) $$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial v_j}\frac{\partial v_j}{\partial w_{ij}} = \delta_j y_i$$ (2.26) $$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial v_j} y_i$$

(2.27) $$\frac{\partial E}{\partial y_i} = \sum_{j \in output} \frac{\partial E}{\partial v_j}\frac{\partial v_j}{\partial y_i}$$ (2.28) $$\frac{\partial E}{\partial y_i} = \sum_j -(c_j - y_j) y_j (1 - y_j) \cdot w_{ij}$$

Subsequently, the error with respect to the weighted sum of inputs to the hidden neuron can be obtained.

(2.29) **Backpropagation:** $$\delta_i = \frac{\partial E}{\partial v_i} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial v_i} = \varphi'(v_i) \sum_j \delta_j w_{ij}$$

This is known as 'backpropagating' the error throughout the network [17]. Thus, the error with respect to the weighted summed input of each hidden unit (δ) can be calculated nonetheless that the target output of the hidden units is unknown. This allows to calculate the weight updates for all hidden units (according to the error measures δ) [17].

(2.30) **Weight updates:** $$\frac{\partial E}{\partial w_{hi}} = \frac{\partial E}{\partial v_i}\frac{\partial v_i}{\partial w_{hi}} = \delta_i y_h$$

The algorithm for the online training of a multilayer network thus exists of a forward and backward pass and can be summarized as follows:

1. Pass a training sample $x^{(q)}$ through the net and compute the activation of all hidden units and the output $y^{(q)}$.
2. Compute the error measures δ for all output units based on (2.23).
3. Propagate the errors δ throughout the network to the underlying hidden layers using the backpropagation formula (2.29).
4. Compute the weight update rule for those units following (2.30).
5. Repeat from Step 1 for all training samples.

Prevent overfitting In light of the large amounts of weight parameters that characterize a feedforward neural network with a certain number of hidden units, it is apparent that preventing the network from overfitting is a critical factor. The most obvious way to prevent overfitting, is to gather more data or otherwise opt for the simplest model that is capable of capturing trends in the available data (i.e. Occam's razor), which will generally entail a reduction in the amount of hidden units. Furthermore, regularization, as applied in many machine learning applications, can be used to accomplish this, e.g. through penalizing the L1- or L2- norm of the weights. Another option is to exploit the iterative training process of the neural network. By evaluating the network on a validation set after each training iteration, the start of overfitting can be visualized as the point where validation error starts to increase regardless of a continuing decreasing training error. The training process can thus be halted at the critical point with lowest validation error. Overfitting can also be delayed through data augmentation, which performs operations on the data (e.g.translations, rotations, color modification, noise addition) to simulate a larger dataset. Yet another path forward targets the complexity of the model's architecture. During the training process of the network, the weights of the respective hidden units are optimized to reduce the output error in light of what the other units are doing. This can consequently lead to co-adaptation of the units, where they will rely on the activities of one another to optimize performance. It is natural that this optimization will not extend to the test set. The technique of dropout [45] aims to reduce or avoid as much as possible co-adaptation by making the presence of other units unreliable. This is achieved through stochastically dropping some hidden units at each iteration and training the net as such. It is important to remark that these are merely some of the most popular techniques in the battle against overfitting. For more information on the many other mechanisms, we refer to [10], [17].

2.2.4 Feedforward neural networks for object recognition

Feedforward neural networks can be considered as concise universal function approximators: architectures with sufficient hidden units can be trained via gradient descent/backpropagation to approximate any function between the input and output layers. Applying such deep learning techniques to object recognition aims to eliminate the challenging feature-engineering task of machine learning. There are however limitations to the performance of feedforward neural networks for object recognition on images.

The primary concern is the ignorance of feedforward neural networks to the character of images, which contain locally-grouped 2D information and recurring patterns at different locations. Furthermore, the sensitivity of the feedforward neural nets to translations, scaling and local distortions of the input requires the need for strong preprocessing to standardize the inputs. Both aspects thus stress the importance of incorporating this prior knowledge about the nature of object recognition in the algorithms. A final concern is the dimensionality of an input image, which generally contains a high number of pixels. Applying a first fully-connected layer of a feedforward neural network to all pixels would result in a high number of weights at this layers, thus resulting in a high-dimensional problem.

2.2.5 From the visual hierarchy pathway to CNN's

To overcome the problems encountered in the field of computer image recognition, further inspiration was drawn from the concepts of human visual perception, which has lead to Convolutional Neural Networks (CNN's) [46].

Biological visual perception Biological visual perception was classically considered to occur through two separate retino-cortical pathways. The existence of a 'what'-pathway and a 'where'-pathway was

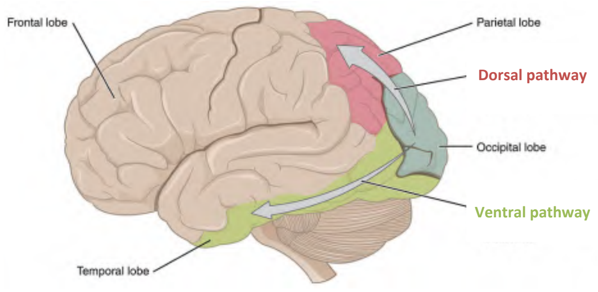

Figure 2.10: Simplified depiction of the two visual streams: (1) the dorsal pathway and (2) the ventral pathway [47].

observed after the Second World War in veterans with cerebral missile wounds by Newcombe & Russell [48]. Lesions in the parietal lobe impaired the maze-learning performance of the patients, whereas lesion in the temporal lobe resulted in a decreased closure ability, i.e. mentally reconstructing an image based on incomplete information [49]. Further evidence for this model was acquired in a study by Ungerleider and Mishkin in which macaque monkeys were impaired with focal lesions in the cerebrum and investigated on an anatomical and electrophysiological level [50]. Their resuls indicated a distinction between two pathways originating from the primary visual cortex V1: a dorsal pathway projecting to the posterior parietal lobe and inferior ventral pathway to the temporal lobe, which are respectively involved in the visuospatial and visuoperceptual functioning of biological vision [51], as illustrated in Figure 2.10.

As indicated in Figure 2.11, both pathways receive their input from the photoreceptors at the retina and pass through many of the same cortical areas before parting ways spatially. They are however segregated on an anatomical level from as early as the ganglion cells on the retina, which is reflected by the two pathways synaptically mapping at different layers in the same cortical areas they cross. At the lateral geniculate nucleus, the visuospatial system springs from larger neurons at the ventral side and is correspondingly known as the magnocellular-pathway, which processes luminescence-based spatial inputs. Contrarily, the parvocellular-pathway, which refers to the visuoperceptual system stemming from smaller neurons at the dorsal side, conveys colour-sensitive input without regard to spatial aspects [51], [52].

Further distinctions can be made between the pathways with regard to their required outputs. The dorsal pathway projecting to the parietal lobe conveys the information required to execute an action, i.e. a spatial description of the scene (position, orientation and structure), which is supported by the high temporal sensitivity of said pathway. The posterior parietal lobe receives input both sensitive to the spatial scene of objects and self-initiated movements, correlating the body's position to its surroundings and it further projects to the frontal lobe, where it thus plays a role in reaching movements of limbs and hand and finger grasping actions, as well as ocular control [54]. The ventral pathway on the other hand

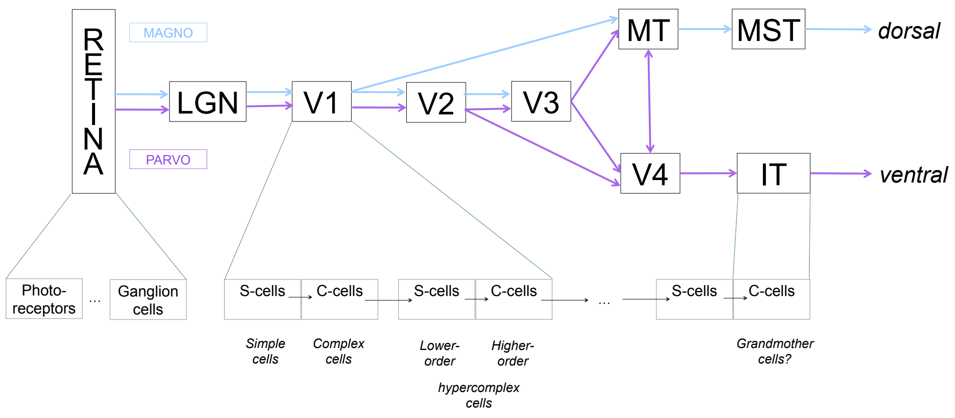

Figure 2.11: Comparative illustration of the organization of the two visual pathways [52] and the stacking of S- and C-cells [53].

provides higher-resolution information more slowly to the temporal lobe with detailed perception as objective. The structures involved in the ventral stream play a role in visual perception, memory and learning. [49], [52], [55]

Object recognition in the ventral visual pathway Delving deeper into the functioning of the ventral stream, fundamental discoveries concerning biological image processing were made by neurophysiologists D. Hubel and T. Wiesel in the 1960s [56], [57]. Their experiments with cats firstly showed oriented slits of light to be the most effective in stimulating certain striate cortex neurons, whereas an oriented line of point sources was more effective in stimulating neurons in the lateral geniculate nucleus and, as previously shown by Kuffler [REF], neurons in the retina. This indicated the differences in perception between retina and cortex, as well as the possible importance of inhibitory fields surrounding the receptive fields of input neurons. Secondly, they established the topographical structure of the visual cortex, i.e. retinal perception maps are represented in similar topographical maps in the cortex, where columnar structures represent cells processing the same information. Thirdly, they observed a hierarchical structure in the organization of cells in the visual pathway. Distinguishing simple, complex, lower-order and higher-order hypercomplex cells, information processing was assumed to be hierarchical through this sequence of increasingly more complex cells. More complex cells were less sensitive to the exact position of the lines, were thought to rely on information integration and appeared to have larger receptive fields as they did not respond greatly to a line of point sources. Thus, Hubel and Wiesel showed hierarchical processing of information in the primary visual cortex V1 through different cell types, distinguished based on their receptive field and tolerance. Furthermore, they also alluded to the existence of similar neuronal types in other visual cortex areas (V2, V3 and V4). Finally, cells in the inferotemporal cortex can handle highly complex patterns under complicated image variations. [58], [59], [60]

Stepping away from a simple two-stream, feedforward visual model: Some recent advances Moving forward from the fundamental research, it becomes increasingly obvious that, as for any physiological process, biological vision is not this straightforward. On a structural level, it serves to be mentioned that the two pathways are in reality not that clearly delineated. Not only does the dorsal stream require information seemingly entering only the ventral stream to accomplish any visuospatial task and vice versa [55], new research also shows that the two streams interact along the way [49]. L. Cloutman discusses three possible models of inter-stream communication in her work [61] and recent research documented cross-talk via specific white tract pathways linking the two streams with diffusion imaging tractography during hand actions [62]. On a processing level, it is important to note that a mere feed-forward model cannot capture the dynamics of visual perception. For the ventral stream, hierarchical feedforward projections have the upper hand during recognition of an image shown in a time window of 100-200 ms (i.e. 'blink-of-an-eye' timeframe), with the objective of transforming information to a neuronal representation in the inferotemporal cortex [60]. However, when the viewing window lengthens in time, strong feedback connections come into play to enhance perception through incorporation of finer details obtained at lower levels of the pathway. Thus, attentive image recognition on a longer time scale is hypothesised to exist as a complex interplay between feedforward and feedback connections in the ventral stream [63], [64]. Thirdly, the classification of cells into simple, complex and hypercomplex cells is more intricate than previously assumed. Rush et al. showed that V1 neurons portray different combinations of quadratic filters, possibly indicative of simple and complex cells being just two of many possible configurations of the neuronal receptive fields [59], [65].

Computer vision spurred on by neuroscience Inspiration for computer vision has been found from a combination of factors mentioned above. The core idea drawn from biology entailed that the ventral pathway aims to transform raw images through a number of stages into a compact, but descriptive and distinctive representation regardless of the variational input [66]. Focusing on the feedforward pass during 'vision-at-a-glance', the idea of a hierarchical model sprung forth which targets to incorporate the increasing selectivity and tolerance characteristics of the cells. To achieve this, the notion of simple and complex cells as found in V1 has been mimicked by creating alternating layers of cells that simulate their respective behaviors in models.

Fukushima's Neocognitron The first artificial neural network that achieved this was designed by Fukushima in 1980 [67]. The multilayer neural net consists of alternating layers of S-cells and C-cells. As illustrated in Figure 2.12, these alternations, with each S-C pair defined as a stage, extract increasingly more complex features from its input. The columnar information processing structure of the cortex is ensured by retinotopically structured connections between cells of subsequent layers, i.e. cells only receive input from a limited area of the previous layer. Further biological inspiration was drawn from object recognition in humans relying on specific receptive fields for the detection of similar patterns at different locations in images. This was incorporated in the neocognitron by creating cell planes in

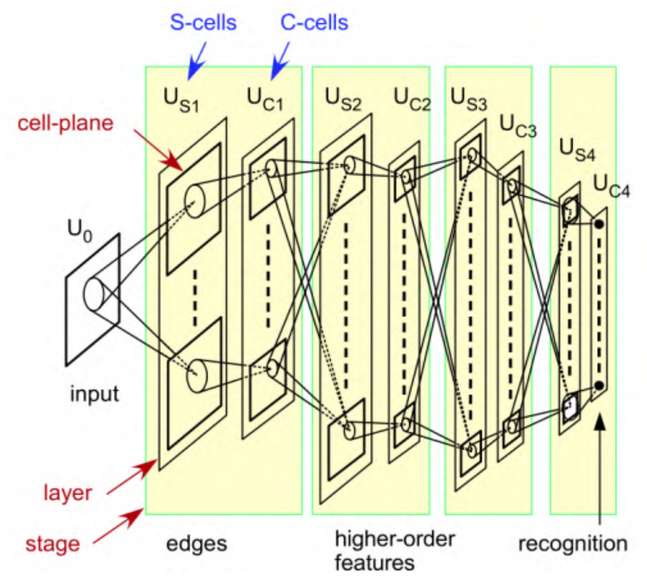

Figure 2.12: General architecture of the neocognitron [53]

each layer, which represent a group of retinotopically organized cells receiving the same set of input connections [53].

In this architecture, the S-cells are responsible for feature extraction, where the features to which they will respond are determined through a learning process of the variable input connections. Simple features, such as specific orientations of lines and edges, will be extracted by lower-level S-cells, while higher-level S-cells extract more global features (e.g. parts of learned patterns). The C-cells receive their inputs through fixed connections from S-cells in the previous layers and are responsible for a robust pattern recognition (i.e. decreasing sensitivity to a deformation or location shift of a pattern). Each C-cell receives inputs from a number of S-cells ('the input window') in one cell plane, i.e. from S-cells that extract the same feature, but at different spatial locations. The C-cell will fire, if it receives an input from at least one of the S-cells. Consequently, the feature will be detected even under a shift in the location of the input features, rendering the system less sensitive to the exact feature locations. The behavior of the C-cells can however also be interpreted from an alternative point of view. The input windows for the different C-cells strongly overlap thus C-cells can be considered as performing a spatial blur on the excitatory signals they receive from the S-cells. This spatial pooling is obtained by averaging these signals from the input window, which are S-cells that perceive the same feature at slightly different locations. Furthermore, the excitatory S-cell input window is often framed by a small inhibitory region. This enables the C-cells to distinguish a continuous line from the end-point of the line and allows the network to still recognize features after blurring. A consequence of the fixed input windows of the C-cells is a reduction in size of the cell-planes, thus allowing the network to compress data as it progresses to higher stages [53].

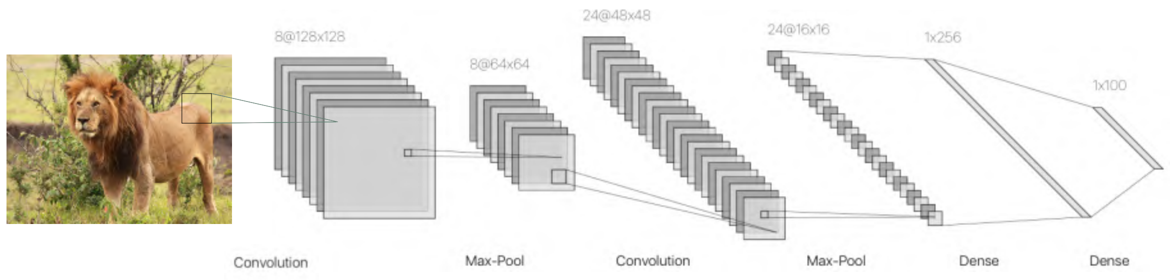

Figure 2.13: Illustration of a convolutional neural network (adapted from [46], [68] and [69]).

LeCun's CNN Modern convolutional neural networks originated from the work of Y. LeCun whose network has a similar architecture as Fukushima's neocognitron. It translates the behaviour of the S- and C-cells in mathematical formulations, respectively as convolutions and subsampling. The network primarily distinguishes itself however from the neocognitron in its training process. Whereas the neocognitron relies on a layer-wise, unsupervised training process of the S-cell layers and supervised training of the output layer, LeCun's CNN is trained to find a global minimum over all the parameters through backpropagation. The key notions that mimic biology and render CNN's so apt for image classification are summarized below.

2.3 Powerful CNN's for object recognition

2.3.1 Key concepts of CNN's

Convolutional layer As images contain highly correlated 2D information (generally in three RGB-channels), the idea of *local connections* has been exploited by creating units in a convolutional layer which receive weighted input from local patches in the previous layer, referred to as their receptive fields (RF's). By relying on local RF's, this introduces sparse connections, rather than the whole-scale image statistics passed on to units and causing high-dimensional problems in fully-connected feedforward networks. The weight vector of a unit is also known as filter bank and through training the weights will be updated to filter for specific features, giving the units the behavior of filter detectors. The weighted inputs and additive bias are then passed through the activation function of said unit, where ReLU's have portrayed much faster learning in deep networks compared to other activation functions [46], [43].

Furthermore, images tend to portray similar features at different spatial locations. Accordingly, units with receptive fields at different locations will have the same weight vectors to extract similar features, which is known as *parameter sharing* and further reduces the dimensionality of the problem [12].

On a structural level, the outputs of units with the same weight vector are organised in feature maps. Multiple feature maps (with different weight vectors) in each convolutional layer enable scanning for different features in each local image patch. The optimal number of feature maps needed to capture the image dynamics tends to increase with the complexity of the input images [12].

Mathematically, the filtering operation performed by a unit with a small RF is in fact a discrete convolution. Assume a gray scale image I

$$I : (1, ..., d_1) \times (1, ..., d_2) \rightarrow \mathbb{R}, (i, j) \mapsto I_{(i,j)} \tag{2.31}$$

representing a d_1 x d_2 dimensional array. A kernel K $\in \mathbb{R}^{2h_1+1 \times 2h_2+1}$, i.e. size of the corresponding receptive field, filters the image by means of a discrete convolution

$$S_{(i,j)} = (I \star K)_{(i,j)} = \sum_{u=-h_1}^{h_1} \sum_{v=-h_2}^{h_2} I_{(i+u,j+v)} K_{(u,v)} \tag{2.32}$$

which, given the discrete nature of the image, entails a matrix multiplication of image and kernel in the receptive field. [2] At edges, padding or inhibition on applying the kernels there prevents the appearance of negative indices. The notion of parameter sharing, as translated in this convolution of image with filter, now introduces the characteristic of the convolutions to be equivariant to translations. Thus, the CNN obtains robustness against distortions and shifts of the input image [10], [46].

Pooling layer The feature maps in the convolutional layer serve as input for the next stage, being a pooling layer. Given that the relative location of features with respect to each other contains significantly more information than their precise pixel-location, further stressed by the notion that small shifts in the precise feature locations are to be expected in different images, the idea of *local pooling* of features to reduce spatial resolution is implemented at this stage. This results in a network invariant to small shifts in the images and the stages of convolutional/pooling layers equip the network to handle variable size inputs.

Pooling units will scan the feature maps with (non-)overlapping $p_1 \times p_2$ receptive fields and output a single value. Typical choices for pooling units are subsampling, where the output of the a unit in the j-th layer is the average of units in the receptive field, supplemented with additive bias b and passed through an activation function [46] and max pooling,

$$y_j = \max_{p_1 \times p_2} (y_i^{(p_1 \times p_2)}) \tag{2.33}$$

which outputs the maximal value in the receptive field [70].

[2] It should be noted that this implementation is a cross-correlation which can be preferred on a practical level over flipped convolutions. [10]

Deep hierarchies Finally, the use of *many layers* represents the different stages in the simplified model of the ventral stream. Features at higher-level stages results from combinations of lower-level features, thus enabling deep structures to capture the compositional hierarchy of natural images [12].

2.3.2 Recent advances in CNN Architectures and design elements

Given the current hype of deep learning, optimizations of CNN architectures and designs is in hot pursuit. This section will therefore be devoted to some of the powerful CNN models that have been developed in the past few years and will be used throughout this work. Specific, revolutionary design-elements that have assisted these improvements are marked throughout this discourse.

AlexNet and VGG's AlexNet, as designed by Krizhevsky A., Sutskever I. and Hinton G. in 2012, consists of five convolutional layers, response-normalisation layers, max-pooling layers with overlapping pooling windows and finally fully-connected layers at the output end. They demonstrated the enhanced efficiency in training through the use of *ReLu's* and applied *dropout* on the fully-connected layers to reduce overfitting. [43] In contrast with the 11x11-sided kernels used in the first convolutional layer of AlexNet, the VGGNet, designed by Simonyan K. and Zisserman S. in 2014, made use of 3x3-kernels. Relying on blocks of 2 to 3 convolutional layers before a max-pooling layer, they simulated the effect of a large receptive-field filter, while decreasing the number of parameters tied to large filters. Best known are the VGG16 and VGG19, respectively containing 16 and 19 weight layers, which demonstrated the effectiveness of deeper models with numerous layers to improve performance. On an architectural level, 3x3 filter kernels are used in each convolutional layer, but increasing numbers of filters are applied at deeper layers, thus reducing the spatial dimension, while increasing the width of the layer when moving through the net [71].

Inception networks A more intricate CNN architecture was developed at Google by Szegedy C. et al. Aware of the limited window for improving the performance of CNN architectures by creating deeper and wider models, given the increasing demands on both dataset size and computational power, they introduced the concept of *inception blocks*. At each new layer in a CNN architecture, conceptual hyperparameters are the choice between a convolutional or maxpooling layer and the respective size of the receptive fields, i.e. extracting smaller or more general features. The inception block aims to allow the net discover itself what is more useful at each layer. Using 1x1 convolutional layers, the width of each layer is reduced before performing a computationally more expensive convolution, thus allowing the introduction of deeper structures in a computationally more efficient manner. This architecture is also known as the Inception-V1 net, where modifications led to the Inception-V2, -V3 and -V4 models [72].

Residual networks Another advance was made through the introduction of *residual blocks* as done by He et al. at Microsoft in their ResNet Architecture. These blocks (starting from a representation $\boldsymbol{x}$)

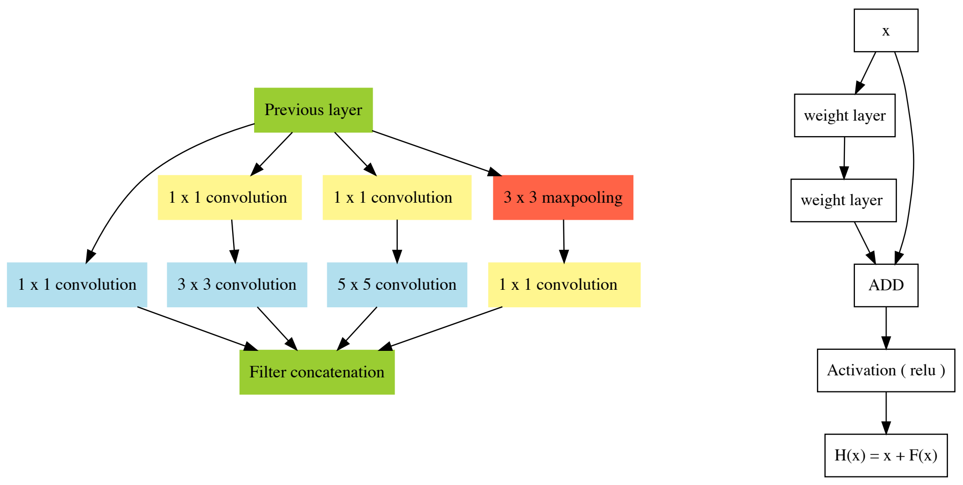

Figure 2.14: Illustration of (a) an inception block [72] and (b) a residual block [73].

consist of two pathways, one in which computations are performed ($F(\boldsymbol{x})$) and one shortcut connection, mapping the underlying representation $\boldsymbol{x}$ to a deeper level, where these two paths are added. They hypothesised that it was easier for a deep layer to learn the residual mapping $H(\boldsymbol{x}) = F(\boldsymbol{x}) + \boldsymbol{x}$ than to merely learn the original, underlying representation $\boldsymbol{x}$. Results on ImageNet indicated that the deep ResNet's with up to 152 layers were easier to train than their non-residual opponents and portrayed a higher accuracy [73].

2.3.3 Benchmark performances of deep CNN architectures

Often cited to demonstrate the power of CNN's for image recognition tasks is the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Consisting of a publicly available dataset, labeled through crowd-funding approaches, spanning 1000 different object classes, it yearly sets up a competition and is often used as bench mark for new CNN architecture performances for object recognition. [74] In the proceeding discussion, the ILSVRC will serve as a guideline to discuss the most poignant results obtained with deep CNN's with indication of some remarkable new elements.

To give an indication of the performance of these models, the top-5 error rate on the ImageNet dataset of the aforementioned nets is presented in Table 2.3. It has to be noted that the submitted nets, training and testing schemes offer a myriad of other hyperparameters, which were tuned to achieve these results (e.g. a 7-network ensemble of VGG-architectures). From these results, it can be observed that ResNet [73], as well as other more recent nets (e.g. InceptionV3 [75], InceptionResNetV2 [76]), succeed in surpassing the documented human performance (5.1 % [74]) on this dataset.

Table 2.3: Results ILSVRC 2012-2015 ([43]), [71], [72], [73]).

CNN	Nb. layers	Top-5 error rate
AlexNet	7	16.4 %
VGG16-19	16 - 19	7.3 %
GoogleLeNet	19	6.67 %
ResNet	152	3.57 %

2.4 Transfer learning

2.4.1 Motivation for transfer learning

However, the ILSVRC poses a unique problem with millions of annotated images in which each class is sufficiently represented, thus rendering it highly dissimilar from the often scarce and unbalanced real-life datasets. Furthermore, the training of the aforementioned deep CNNs generally required multiple GPU's and was a time-consuming process (up to 3 weeks). Thus, it is clear that training the same, high-performance CNN architectures on real-life data is in most cases unfeasible. The CNNs trained to recognise the ImageNet dataset have nevertheless a strong capability to extract very distinctive features from natural images, rendering them useful for datasets of other images. This is supported by the idea that the lower levels of these architectures extract general features from the images and become increasingly more task-specific deeper in the network. From this the idea of transfer learning sprung forth, which aims to kickstart the process of the random weight initialisation by using the pretrained weights of deep CNN's to facilitate the training process.

2.4.2 Key notions concerning transfer learning

Transfer learning entails training a source network on a source dataset for a source task, transferring the learned knowledge (e.g. features or weights) to a new network for a different learning task and/or on a different target domain [77]. In formal terms, the definition of Pan and Yang is given here.

> "Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T, transfer learning aims to help improve the learning of the target predictive function $f_T(.)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$ or $T_S \neq T_T$." [78]

Transfer learning is used with the objective of increasing performance on the target task by exploiting the trained knowledge of the source task. It can either serve as a 'kickstarter' for the process improving initial performance, increase final performance or speed up the training of the target task [79]. Three main types of transfer learning can be distinguished with respect to similarities in the source/target domains/tasks: (1) inductive transfer learning with the goal of inducing a generalizable, predictive model for the target task from a different source task; (2) transductive transfer learning where a model should be derived in the target domain for the target task lacking labeled data, based on the source task containing a substantial amount of labeled data; and (3) unsupervised transfer learning which targets unsupervised learning in the target domain based on a different source task [78], [79].

For purposes of image classification, the approach of inductive transfer learning has gained momentum in the past years. Given the large training time and requirements on labeled data, architectures (as specified in Table 2.3) are pretrained on the ImageNet (source domain) for 1000-class classification purposes (source task) and knowledge is transferred to a labeled dataset of choice (target domain) for e.g. classification as target task. Thus, it primarily exploits the learned feature-representations in the source domain and aims to transfer these to the target domain with the aim of improving classification performance whilst making use of deep, data-greedy CNN architectures [78].

The transferability of different levels of features was touched upon in the work of of Razavian et al. [80], with results confirming that the deepest layers are most task-specific and better results can be obtained by using features from levels upwards in the model. This was confirmed by the work of Zheng et al. [81], in which furthermore was suggested that pooling of features as well as combinations of features may improve performance. Yosinki et al. [77] added to the evidence by documenting higher performances for CNN's initiate with pretrained weights and finetuned for the task at hand, rather than through random weight initialization and training from scratch. Furthermore they investigated the transferability of specific feature layers and reported co-adaptation of intermediate feature layers, where splitting between frozen and trainable layers at this level might result in a drop in performance.

2.5 Auto-encoders

From the previous discourse follows the key message of deep learning, in that it is a powerful technique for CAD of signals and images in the medical world, which is equipped to learn powerful internal representations of the data on which it may perform classification. CNNs in particular rely on supervised learning, with backpropagation of the error throughout the net. However, the question arose in the past as to how to learn an internal representation without the availability of target outputs [82]. Thereto, the structure of auto-encoders (AE) was suggested, which aims to use the input as target output, exploiting error backpropagation to achieve an optimal 'mirroring' of the input. [3]

[3] Recirculation has been proposed [83] as an alternative to using backpropagation for training autoencoders. This method omits the perils of the supervised learning requirements for backpropagation by eliminating the output-neurons and merely considering 'visible' and 'hidden' units. These visible units are then tracked throughout time, with the initial state being the original input and further states representing the reconstructed inputs. The weights to the hidden units are updated according to the discrepancy of the states of the visible units.

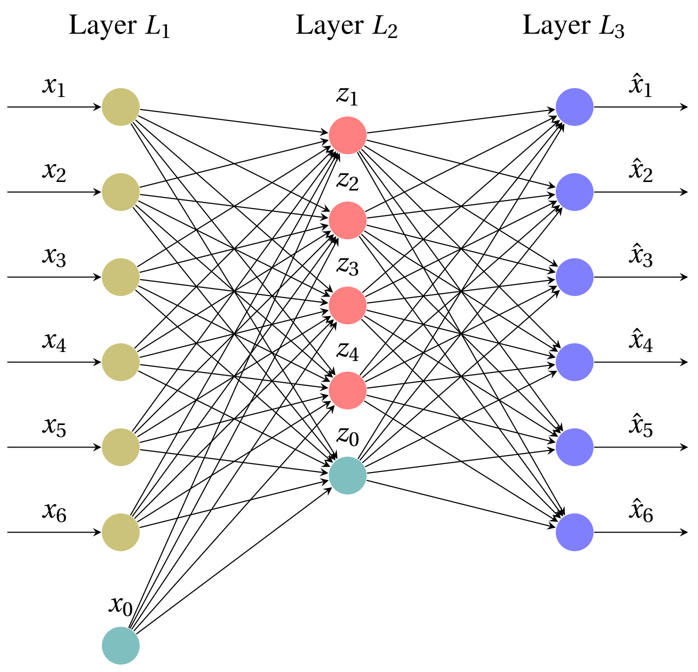

Figure 2.15: Architecture of an undercomplete auto-encoder building block. The fully-connected auto-encoder contains a single hidden layer of lower dimensionality than its input layer.

2.5.1 Undercomplete auto-encoders

Undercomplete AE building block As shown in Figure 2.15, the original proposition of the autoencoder relied on a single hidden layer. It was purposed for dimensionality reduction, with the internal representation being of a lower dimension than the input and containing the most characteristic features from which the input can be reconstructed. The one-layered encoder produces the latent representation $\mathbf{z}$ through a mapping function $\mathbf{z} = f(\mathbf{x})$, from which the output $\hat{\mathbf{x}}$ is reconstructed by the decoder $\hat{\mathbf{x}} = g(\mathbf{z}) = g(f(\mathbf{x}))$. Moving towards this compact representation of the input clearly contains a loss of information from which we have to recapture $\hat{\mathbf{x}}$. The training objective will thus be to minimize this reconstruction loss $\mathscr{L}(\mathbf{x}, \hat{\mathbf{x}})$, which is generally the cross-entropy or the mean-squared error to quantize how dissimilar the input and its mapping are [84]. This technique can thus be considered as an extension of PCA, as it can learn non-linear mappings between input and hidden layer under non-linear activation functions for the neurons (or cross-entropy loss) [10].

Limitations Regardless of the possible strength of the aforementioned principle for non-linear dimensionality reduction, allowing the auto-encoder to be of too high capacity places its generalization capacity in jeopardy [10]. The auto-encoder can in that case move towards learning the identity function on the training set rather than discovering underlying data characteristics, thus resulting in severe overfitting. Consequently, severe restrictions should be placed on the dimensionality of the architecture. Alternatively, regularization techniques have been proposed to counter these trends, which will be described in the following section.

Training AE building block The mapping functions of the encoder and decoder are typically described by

$$f(\mathbf{x}) = \varphi_e(\mathbf{w}^{(e)} \cdot \mathbf{x} + b^{(e)}) \tag{2.34}$$

$$g(\mathbf{z}) = \varphi_d(\mathbf{w}^{(d)} \cdot \mathbf{z} + b^{(d)}) \tag{2.35}$$

with activation functions of encoder and decoder respectively being φ_e and φ_d [85]. The AE parameters are thus $\Theta = \left(\mathbf{w}^{(e)}, b^{(e)}, \mathbf{w}^{(d)}, b^{(d)}\right)$, where the weights of encoder and decoder can either differ or be tied, in which case $\mathbf{w}^{(d)} = (\mathbf{w}^{(e)})^T$. The AE can thus be trained as a conventional multi-layered perceptron with stochastic gradient descent backpropagating the error to the hidden layer.

2.5.2 Overcomplete auto-encoders

Overcomplete AE building block Augmenting the dimensionality of the hidden layer and allowing it to surpass the dimensionality of the input layer, gives rise to an overcomplete AE, as depicted in Figure 2.16. A naive implementation of this framework would result in an exact activity mapping from input to output, by merely dropping the function of the surplus of hidden neurons (i.e. they become ‘inactive’). However, extra regularization terms are used throughout training to enforce the hidden layer to learn not merely an identical mapping, but also e.g. sparsity of hidden representation, robustness against noise or limitation on gradient of hidden neuronal activations [85].

Regularized auto-encoders In a *sparse auto-encoder*, sparsity is introduced in the hidden layer, thus promoting a greater number of neurons to be in their inactive state. The sparsity constraint (Ω) can either be placed on the bias (b) or on the output of the activation function (e.g. $\sum_i |z_i|$) and training now aims to minimize $\mathscr{L}(\mathbf{x}, \hat{\mathbf{x}}) + \lambda\Omega$, where $\hat{\mathbf{x}} = g(f(\mathbf{x}))$ [10]. Another option does not interfere with the hidden layer, rather it lets the auto-encoder learn a noisy version $\tilde{\mathbf{x}}$ of the input $\mathbf{x}$ to incorporate robustness against noise in the model. The idea behind this is to let the model learn some distribution around the true inputs, rather than an exact mapping of the training data [85]. The corresponding loss function which has to be optimized in a *denoising auto-encoder* thus becomes $\mathscr{L}(\mathbf{x}, g(f(\tilde{\mathbf{x}})))$. A final variant is known as a *contractive auto-encoder*, which introduces a regularization constraint on the gradients of the hidden neuronal activations with respect to the inputs. This aims at letting the model learn an internal representation that can cope with slight perturbations of the input [84].

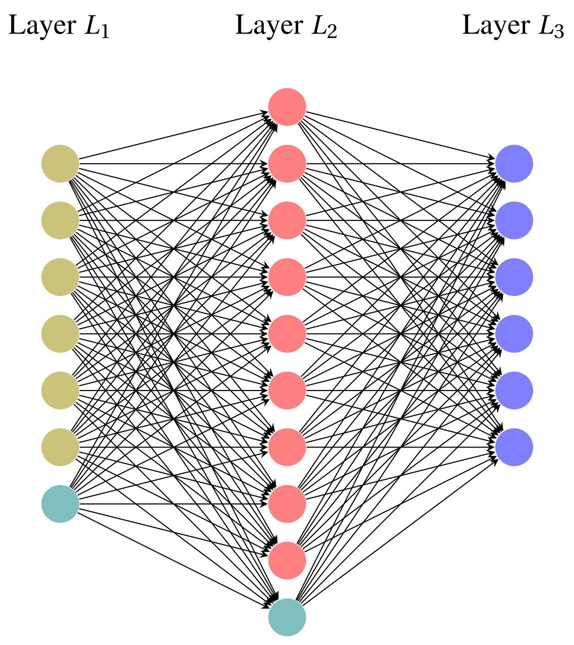

Figure 2.16: Architecture of an overcomplete auto-encoder building block.

2.5.3 Deep auto-encoders

The importance of auto-encoders has also surfaced in deep learning. Originally done by stacking several auto-encoders on top of each other and training them in a consecutive bottom-up manner to learn the subsequent stages of internal representations, the trained encoder can be singled out. Subsequently, stacking a top-model on this encoder gives rises to a deep neural network that has been partially pre-trained [86]. It thus exploits the use of unlabeled datasets for 'kick-starting' the supervised training of deep architectures. The use of pretraining a neural network has been suggested to act as a regularizer [87], giving rise to the starting point and restricting the parameters of the optimization process during supervised learning to a specific, favorable parameter space. When expanding the depth of encoder and decoder, both are in essence feedforward neural networks and can thus alternatively be trained with backpropagation and gradient descent [10].

Finally, it is important to note that the use of a deep AE for pretraining can also be extended to applications of image classification. In this regard, stacked convolutional auto-encoders have been proposed [88] as alternative to random weight-initialization of a CNN. This architecture allows for unsupervised pretraining of a CNN-base model, followed by supervised learning of the top model and finetuning of the base model weights.

Chapter 3

Skin Cancer Classification

Having introduced the general concepts of deep learning, the biological framework of skin lesions and the literature on machine learning techniques applied in this field will be discussed throughout this chapter. The chapter commences with a concise exploration of the pathology of skin lesions and by extension cancer, as well as the incidence of skin cancer and the current diagnostic processes. Subsequently, an overview will be presented of the use of deep learning in the field of dermatology to aid the skin lesion diagnostic process. The chapter concludes with a concise reflection upon the the reviewed literature and a formulation of the windows of opportunity for further research in this field.

3.1 Skin Lesions and Cancer

3.1.1 Skin biology

As depicted in Figure 3.1, the skin exists of three main layers. The bottom layer is the hypodermis, also referred to as subcutaneous tissue, consisting of loose connective tissue, primarily adipose tissue capable of storing fat. This layer mainly contains fat lobules, thus functioning as an energy reserve, insulation and protection layer, as well as nerves and larger blood and lymphatic vessels. On top of the hypodermis lies the dermis, a layer composed of collagen fibers and elastic fibers (i.e. connective tissue) and an extracellular matrix, all of which are produced by fibroblasts. It contains blood and lymphatic vessels, sensory receptors, hair follicles and sweat glands [89]. The upper skin layer is the epidermis, which can again be structurally subdivided in different layer (Figure 3.2). At the interface between dermis and epidermis lies the stratum basale, a columnar layer of *basal cells*, primarily keratinocytes. The proliferative capacity of the skin has been observed to be restricted to the stratum basale, which is due to the presence of epidermal stem cells. Portraying both self-maintenance and self-renewal,

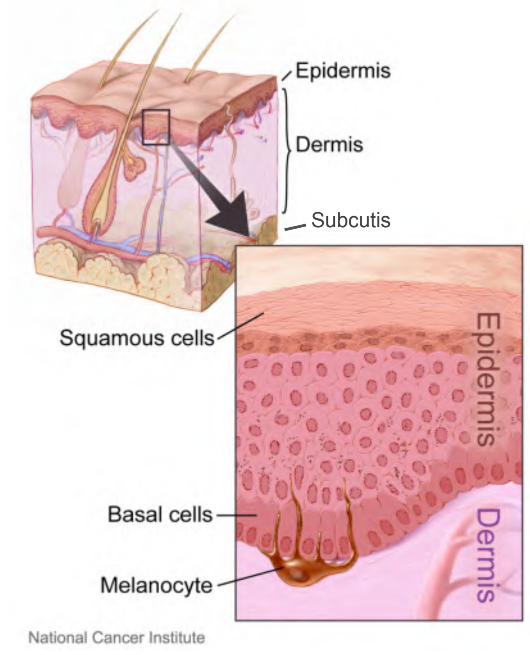

Figure 3.1: Structure of the skin [92]

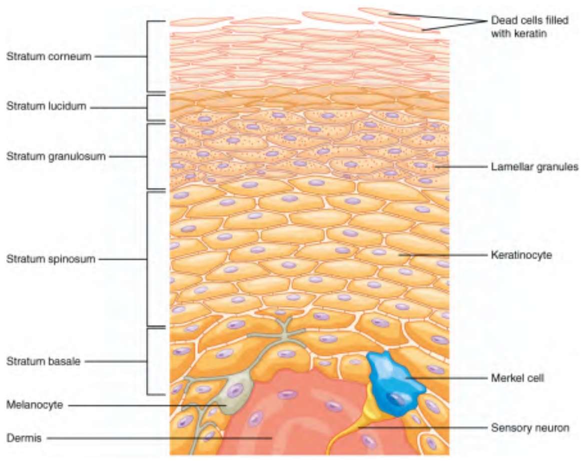

Figure 3.2: Structure of the epidermis [93]

they give upon (asymmetric) division rise to one daughter stem-cell and one transit-amplifying cell. The latter cells have the capacity to undergo a limited number of cell divisions before producing fully differentiated keratinocytes, thus giving rise to a short-lived stem-cell lineage. The differentiated cells move upwards into the suprabasal layers, which consists largely of *squamous epithelial cells*, whereas the daughter stem cell remains in the stratum basale to maintain the stem cell pool. Upon reaching the outermost skin layer, the keratinocytes have undergone a further maturation process and have lost their nucleus and cytoplasmic organelles and are now referred to as corneocytes [90]. Stacked layers of corneocytes compose the outermost barrier of the epidermis, the stratum corneum, which serves as a physical and chemical protection barrier and regulates skin hydration [89]. The non-viable corneocytes are continuously being shed through the process of desquamation and new keratinocytes enter from the underlying epidermal skin layers, thus preserving skin tissue homeostasis [91].

The epidermis contains a number of other cells, amongst others: Langerhans cells which belong to the skin immune system [94], Merkel cells which play a part in mechanoreception and are in contact with neurons [95] and, of importance here, melanocytes. Melanocytes are neural-crest derived cells and can also be found in the hair, and iris, as well as the inner ear, nervous system and heart [96]. Their main function is the production of melanin in dedicated organelles, known as melanosomes. Melanin is a natural pigment that comes in different forms: brown/black eumelanin, red/yellow pheomelanin and brown/black neuromelanin. The most prominent type in the skin, eumelanin, acts as a photo-absorber and has been reported to dissipate over 99.9% of absorbed visible and UV light through non-radiative pathways [97]. The melanosomes are subsequently transported to the keratinocytes, where melanin accumulates in a strategical manner, providing a cap-like cover on the UV-exposed side of the cells [98]. Differences in skin pigmentation can be attributed to a difference in the amount of melanogenesis and the distribution, size and content of melanosomes, rather than to a difference in the amount of melanocytes [99]. The melanogenesis in response to UV exposure is suggested to be under control of the keratinocytes themselves [100].

3.1.2 Skin cancer pathology

A cutaneous or skin lesion can be described as an abnormal change in skin tissue. It can be classified as benign, pre-malignant or malignant with respect to the nature of the lesion being non-cancerous, pre-cancerous or cancerous. In this respect, cutaneous or skin cancer refers to the proliferation of aberrant skin cells. A range of different factors is associated with skin cancer, amongst others particular environmental pollutants and arsenic in the drinking water. The most poignant cause however is exposure to ultraviolet radiation (UVR) (either from sun or artificial light) [101].

UVR is subdivided in three distinct categories according to the wavelengths it encompasses: (1) UVA with wavelengths of 315 - 400 nm, which makes up majority of UVR from sunlight as it is able to penetrate the ozone efficiently; (2) UVB in the 280 - 315 nm range is largely blocked by the ozone, only 1 - 10 % reaches the earth's surface (where increases in UVB exposure have been seen due to ozone depletion); and (3) the highest-energy UVC radiation (wavelengths of 100 - 280 nm) is completely filtered by the ozone layer and does not reach the earth's surface [101]. Furthermore, UVA can penetrate the skin relatively deeply and enter the dermis. The higher-energy UVB radiation remains more restricted to the upper cell layers [102].

Acute UVR exposure is associated with DNA damage, where direct effects are especially related with shorter wavelengths of UVB radiation. The photo-induced DNA lesions generally lead to covalent bonds between pyrimidines, where the two primary types are: (1) pyrimidine (6-4) photoproducts and (2) cyclobutane pyrimidine dimers (CPD's) in a ratio of 1:4 (varying up to 1:10) [98]. Specific repair pathways of the affected cells are put in place to remove the lesions. If repair fails or the lesions are not repaired, they can result in sustained DNA mutations passed to daughter cells upon cell division. If the damage is too severe or for cells that have escaped repair, other mechanisms are called upon: (1) the cell can be arrested in the G1-phase of its cell-cycle, thus broadening the time window for DNA-repair before the cell enters the S-phase during which it replicates; (2) apoptotic pathways can be activated to induce cell death. The key regulator in these processes is the p53 protein, which is furthermore involved in a number of other protection mechanisms (e.g. inhibits angiogenesis, enhances correct DNA replication and repair, maintains genomic stability) [103]. Acute effects also include sun burn erythema (inflammation due to photo-damage presenting as red skin), which is largely contributed to DNA damage from UVB exposure. Furthermore, the tanning process is initiated, which involves the construction of nuclear caps on the exposed cell-side to protect the underlying DNA, as mentioned before. Finally, UVR results in immuno-suppression of the immune system at work in the skin, which is composed of the Langerhans cells, cytokines produced by the keratinocytes and lymph nodes [103].

Long-term effects of chronic or recurrent UVR have as clinical effect photo-aging of the skin. Whilst UVB radiation is highly efficient in causing direct DNA damage, UVA exposure causes the formation of free reactive oxygen species (ROS). ROS have been show to interact with cell membranes, initiating a photo-aging pathway, and to indirectly damage the DNA, thus also being involved in skin cancer initiation [104]. The most prominent chronic UVR effect is associated with accumulation of DNA damage and sustained immuno-suppression. Respectively resulting in DNA mutations and enabling cells to escape immuno-surveillance, these factors consequently form a highly favorable climate for carcinogenesis [103]. This will be the primary subject of the following paragraphs, where the distinction is made between cancer affecting melanocytes or other skin cell types.

Different factors lie at the base of the carcinogenesic process. Mutations due to DNA damage generally result in an inactivation or under-activation of tumor-suppressant genes, whereas oncogenes are overactivated. Furthermore, through new research the existence of cancer stem cells has surfaced and indicated their impact on treatment strategies.

3.1.3 Non-melanoma skin cancer

Non-melanoma skin cancer (NMSC) is the most frequently diagnosed type of skin cancer in the Caucasian population and consists primarily of keratinocyte tumors, subdivided in basal cell carcinoma's (BCC) and squamous cell carcinoma's (SCC) [104]. To give an indication of their prevalence, we report he statistics documented by the American Cancer Society in the US: an estimated of 5.4 million BCC's and SCC's were diagnosed in 2012, spread out over 3.3 million Americans [105]. BCC is the most common type and is responsible for up to 80% of all NMSC cases [106]. They occur mostly on sun-exposed parts of the body, such as the neck and the head, but are however slow-growing and unlikely to spread to other parts of the body. SCC represents 19% of skin cancer cases and it also affects sun-exposed areas such as hands, ears, arms and legs, as well as damaged skin (e.g. by chemicals or radiation or scar tissue). Even though it is more likely to spread to other areas, with 2-5 % of SCC's portraying this behavior, it still remains rather unlikely. More rare types of NMSC, together contributing only to 1% of all NMSC cases, include Merkel cell carcinoma, cutaneous lymphoma and Kaposi sarcoma [104], [106].

Given that NMSC's can predominantly be found in sun-exposed areas, the role of UV radiation is especially pronounced in these cases. The accumulation of DNA-damage generally results in mutations, both in tumor suppressant genes (which tend to be inhibited) and oncogenes (which are subsequently overexpressed). In over 50 % of all human skin cancers (and more broadly, in over 50 % of all cancers), mutations can be found in TP53 gene, the tumor suppressant gene responsible for producing p53, which is as discussed before a key player in maintaining the cell in a healthy state under external stress. More specifically, about 50 % of all BCC and up to 90 % of all SCC have a TP53 mutation [107], [108]. Other genes are implicated in the NMSC carcinogenesis, including: CDKN2A locus (encoding for two different promoters that arrest the cell cycle) and RAS oncogenes. [103]

3.1.4 Melanoma skin cancer

Even though invasive melanomas only make up 1% of all diagnosed skin cancers in the US, they are responsible for over 75% of all skin-cancer induced deaths. According to the American Cancer Society, melanoma of the skin is the 5th cancer with highest incidence in the US, following breast cancer, lung and bronchus cancer, prostate cancer and colorectal cancer. An estimated number of 87 100 of new cases were observed in 2017, as well as 9730 melanoma-induced deaths. The trends over the years in the US show an increase in incidence of melanomas, whereas the mortality rate remained quasi-stable [105]. The increase in incidence can be related to changes in human behavior (e.g. sun tanning), as well as the aging population. The number might however be biased by an increase in screening methods as well.

The documented incidence in Belgium reported the diagnosis of 2635 new cases of malignant melanoma in 2013. At that time, it was the 4th most frequent cancer in females and 7th most frequent cancer in males [109]. Projections about cancer incidence for the period 2014-2024 in Belgium point out a similar increase in malignant melanoma incidence. An increase of about 49% in the yearly incidence is expected, which in absolute numbers entails an increase of 2925 to 4356 new cases each year [110].

Through a disturbed intracellular signaling process, melanocytes escape the strict control of the keratinocytes and consequently start to proliferate and spread. The different phases of this process as portrayed in Figure 3.3, shows an initial proliferation confined to the epidermis, dermis or both, resulting in a typically benign naevus or dark spot. The radial growth phase (RGP) is generally considered as the primary malignant phase and results in some local micro-invasions of the dermis. Upon reaching the more dangerous vertical growth phase (VGP), clusters of proliferated cells can be found in the dermis, which can easily metastasize to other regions [111]. [1] Their propensity to metastasize is exactly what makes these melanoma's so dangerous. The 5-year survival rate for melanoma's in the localised state is 99%, this drops to 63% as the melanoma enters the regional stage and further to 18% upon reaching the distal stage [114].

Two major genetic processes underlie the propensity of aberrant melanocytes to proliferate and metastasize. Firstly, gain-of-function mutations in genes responsible for handling melanocyte proliferation and survival are overactivated. Identified oncogenes include BRAF (mutated in 50-70% of melanomas and) and NRAS (mutated in 15-30% of melanomas) which stimulate a sustained ERK activity. ERK is a cytosolic protein kinase involved in the Ras/Raf/MEK/ERK pathway and its hyperactivity results in augmented

[1]The aforementioned process of melanoma formation to reaching the metastasizing stage is the classic, chronological Clark model in which metastasis is only possible after the previous stages have been undergone [112]. However, new research portrays that the process is more intricate and malignant cells can already spread earlier in this process [113].

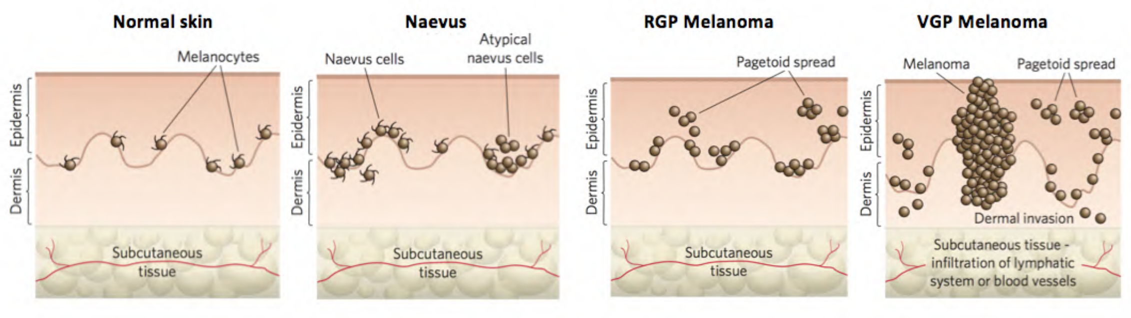

Figure 3.3: Melanoma progression [111]

melanocyte proliferation and survival. Furthermore, mutated BRAF genes facilitate tumor maintenance and neo-angiogenisis. New studies show potential hyperactivation of the PI(3)K-pathway, involved in cell survival, proliferation, growth and motility, in many melanomas. Furthermore, melanocyte functioning seems to be critically interwoven with the concentrations of MITF (micropthalmia-associated transcription factor), where only at intermediate concentrations proliferation occurs, and it can collaborate with the mutated BRAF to stimulate proliferation. Secondly, genetic alterations in tumor suppressor genes can allow melanocytes to evade entering a state of senescence (i.e. complete stop of cell proliferation), which normally occurs upon cell exposure to potentially oncogenic stress. Identified gene loci include the CDKN2A gene, responsible for producing (amongst others) the tumor suppressor protein $p16^{INK4a}$, and mutated p53 functioning, both causing an inactivation of these tumor suppressant pathways [111].

3.1.5 Skin cancer risk factors and prevention

The two main types, melanoma and non-melanoma skin cancers, are currently the most common type of malignancy in the Caucasian population. Even though their mortality rate is in general rather low compared to their prevalence, they place a huge economic burden ($650 million on annual basis in the US [115]) on society and can be extremely dangerous in the case of melanomas. Consequently, there is a strong need for skin cancer prevention. A bi-fold program to achieve these goals should be focused on behavioral changes in the population on the one hand and early detection on the other.

In the case of NMSC, there are strong personal and environmental risk factors. The personal risk factors include gender, genetic predisposition and age. The prolonged life expectancy encountered in current times may go hand in hand with an increased incidence of skin cancers. Exposure to sunlight is the main environmental risk factor and should be the primary target of a prevention strategy. In the case of SCC, the effect of long-term, cumulative sunlight exposure will directly affect the risk. However, studies show a more intricate connection in the case of BCC's, where the risk seems to increase with excessive exposure during childhood, as well as indoor tanning [116]. This thus offers an opportunity to put in place a prevention program focused on limiting direct sun exposure, supported by a strong educational

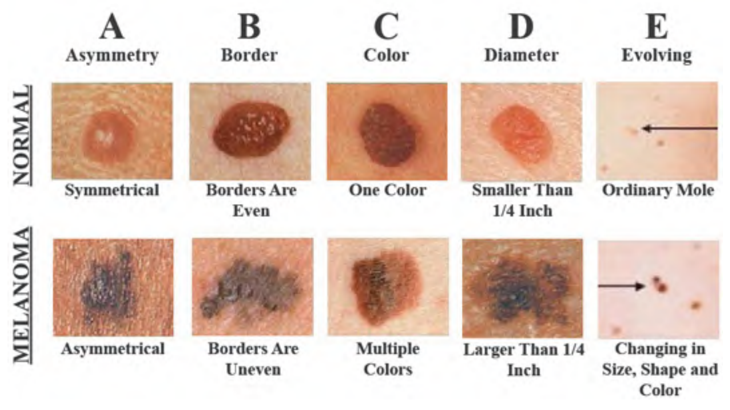

Figure 3.4: ABCDE-system for melanoma detection [120]

component to inform the public. For MSC's, the relation with UVR exposure is less straightforward.

The second window of the prevention strategy is the establishment of an early detection program, which includes self-skin assessment and screening by the physicians [117]. In the case of melanomas less than 1 mm in depth, the 5 year survival rate is over 95% after local excision of the lesion. As stated before, this drops tremendously as the melanoma transgresses in more advanced stages. This indicates the importance of early detection of melanomas. Furthermore, a correct classification of the type of skin tumor (benign or malignant and further subdivisions) is required for establishing the proper treatment plan, as different types require different handling.

3.1.6 Skin lesion diagnosis: Differential diagnosis of malignant melanoma

The diagnostic process for skin lesions generally includes an interview concerning the risk factors for different types, establishment of location, age, gender and arrangement of the lesions is taken into account and a visual screening is performed. With regard to melanoma's, the ABCD(E)-system, published by Friedman et al. in 1985 to detect early stage malignant melanoma's [118] and is now used by physicians for a visual screening of pigmented skin lesions. As can be seen from Figure 3.4, the system relies on specific characteristics of melanoma's, i.e. asymmetry, border, color and diameter. As rapidly evolving skin lesions may be indicative of melanomas, evolution was added as a criterion to this rule [119].

Even though this technique offers some semi-quantitative measures to assess skin lesion, physicians rely strongly on their own knowledge, experience and subjective perception for the classification of the lesions. Furthermore, the technique does not allow to assess depth of the skin lesion. Breslow depth or thickness, which can be defined as the depth of tumor invasion with respect to the granular layer of the epidermis, is however a feature with strong prognostic value and should thus be taken into account [121]. Finally, deviant lesions that do not show the typical characteristics of melanoma's will go unnoticed through this assessment. Other methods have since been proposed, which include the seven-point checklist, the

Menzies method and the CASH algorithm (color, architecture, symmetry and homogeneity) [122], [123].

Another factor complicating the diagnosis of malignant melanoma, is its similarity to pigmented skin lesions (referred to as naevi, Latin for birth marks, or moles) as both share their melanocytic origin. Naevi are benign lesions caused by an accumulation of melanocytes and the most common types include: the common naevus, blue nevus, dysplastic naevus and congenital naevus (which is the actual birthmark the Romans referred to) [7]. These naevi may however be precursors for malignant melanoma's and thus stress the importance of the follow-up of lesions (or the 'E'-feature, as mentioned before) [118].

The recent emergence of dermatoscopy (referred to in short as dermoscopy), also known as epiluminescence microscopy, relies on a reduction of skin surface reflectivity (stratum corneum with refractive index n = 1.55) to allow for an examination of subsurface structures under magnification. The term is generally used to refer to two different hand-held set-ups: (1) non-polarized light microscopy in an immersion set-up to reduce light reflection of the stratum corneum; (2) polarized light microscopy with filter (i.e. cross polarizer) to reduce reflection, which can penetrate deeper in the skin [124], [125]. Inspection of the dermatoscopic images primarily extracts information from the color and structures/patterns of the lesions, where especially the latter is of importance for lesion classification [126]. This noninvasive method has greatly enhanced diagnostic performances (in terms of sensitivity and specificity) of trained dermatologists/physicians [127].

Nevertheless, due to strong inter-patient variability in lesion appearances, visual classification remains difficult and subjective. For final confirmation of the diagnosis, this therefore requires the physicians to resort to histopathological examinations. In regard to this, it serves to be mentioned that dermatoscopy has shown successful in reducing the amount of unnecessary skin biopsies [128]. Furthermore, dermatoscopy has also been proven useful in the follow-up of lesions. Not only can it be employed to track the evolution, it is also a powerful tool to evaluate new lesions in high-risk patients (previously diagnosed with melanoma or having a genetic predisposition for developing melanoma), where biopsying every skin lesion is not feasible [122], [124].

3.2 Machine learning for skin lesion classification

Summarizing the previous discussion, it is possible to state that early detection of melanomas increases survival rate, visual diagnosis is complicated due to inter-patient variability in lesion appearances and subject to the physicians prior knowledge/experience and final diagnosis requires skin biopsies. As dermatoscopy provides high-resolution, magnified images of skin lesions, computer-aided diagnosis has surfaced as a tool to support skin lesion diagnosis.

3.2.1 Creating a large skin lesion database

The International Skin Imaging Collaboration (ISIC) has been created in an international attempt to construct a large, publicly available database for skin lesions images, both clinical and dermatoscopic. The primary goal of the ISIC project is to improve melanoma diagnosis through the accessibility of digital images, in order to enhance early detection and reduce the amount of unnecessary biopsies. Furthermore, the collaboration focuses on creating standards for current dermatoscopic/clinical imaging practices, as they tend to vary in type of technology, technique and terminology while taking into account privacy and possibility for sharing the data between different platforms. [129] A yearly challenge has been set up since 2016 which has, based on a selected training, validation and test set from the ISIC Data Archive, a three-fold goal: (1) lesion segmentation, (2) dermatoscopic feature classification task and (3) disease classification [130].

3.2.2 Classical machine learning approaches

Classical machine learning approaches have been abundantly applied on skin lesion classification tasks. These diagnostic systems generally follow the same methodology, which will be concisely discussed below. As the literature on this topic is very extended, we refer to review papers [7] and [131] for more information.

1. *Image Preprocessing:* To counter the presence of artifacts (e.g. hair, ruler, dark corners) and variability in photographic conditions (e.g. contrast, intensity, angle, perspective), the dermoscopic images are generally preprocessed. Preprocessing steps can be focused on image enhancement, image noise/blur removal or hair removal specifically [132].

2. *Segmentation:* With the goal of removing all background information from the image, segmentation is regularly applied to retain only the region of interest (ROI), i.e. skin lesion. Typical techniques are edge-based segmentation, region-based segmentation and threshold-based segmentation and segmentation based on active contours. Furthermore, artificial intelligence (e.g. DNN's and fuzzy logic) have been applied for segmentation purposes [133].

3. *Feature extraction:* To reduce the computational cost of classification, a selected number of features is to be extracted which may serve as input for the machine learning algorithms. The classification performance for skin lesions has been documented to highly influenced by the quality of the features. Typically, the hand-designing of features aims to mimic the criteria evaluated with existing diagnostic techniques (e.g. ABCDE-rule, 7-point checklist, Menzies method) or are based on pattern analysis [134].

4. *Classification:* Numerous different ML algorithms can be applied on the extracted features, including SVM's, k-nearest neighbor classification and decision trees. We refer to [135] for a comparative study of a number of different techniques.

We stress again that the use of hand-crafted features is the primary bottleneck in this approach, even more so considering that they are generally based on the same diagnostic tools used by dermatologists, which have been proven to be highly subjective and unreliable. Furthermore, all steps (and specifically the preprocessing and feature extraction) result in a strenuous process involving a substantial degree of human intervention. Therefore, a shift can be seen in literature stepping away from said methods and moving towards deep learning, with the aim of letting the neural networks themselves extract features that are perhaps more indicative of the pathology than the common diagnostic tools.

3.2.3 Related work: Deep-learning based methods

Deep learning has surfaced a tool of invaluable measure in the movement towards computer-aided diagnosis in medicine and has been applied on skin lesions. A dire point for the application of supervised deep learning methods such as CNNs is their enormous need for training data. Medical images and their correct labels are often not collected on open-source, integrated platforms, thus rendering this approach seemingly impossible. However, through the creation of the ISIC Archive, such methods are - to some extent - now available to the general public. Alternatively, data could be obtained through large and generally private collaborations with hospitals.

Specifically in the field of skin lesion diagnosis, two main research trends can be visualized. A first strategy exploits deep CNN architectures through transfer learning schemes, thus overcoming the limited availability of limited data to some extent. Another approach aims to use the larger, unlabeled datasets that are available through specific unsupervised learning techniques, after which supervised learning is applied for classification. Papers of relevance in both lines of research will be briefly discussed below.

Related work focused on CNN-based supervised learning

The effectiveness of transfer learning for skin lesion classification (in specific melanoma vs. benign lesions) was portrayed in the study by Codella et al. (2015) [136]. Two layers (FC8 = 1000-dim concept detector layer and FC6 = 4096-dim dense layer) from the Caffe CNN, pretrained on the ILSVRC, were used as feature extractors, whose outputs were subsequently used as SVM inputs. The obtained results were equivalent to previous performance based on hand-crafted features, portraying the functionality of using feature detectors trained on photographs of real-world images to extract features relevant for skin cancer classification, even though the two domains used in this transfer learning set-up are quite distant. Furthermore, sparse coding was applied as unsupervised technique, both in grayscale and RGB color space, where the latter increased performance. Finally, the strength of ensemble techniques was observed, where combinations of deep features and sparse coding features (with global averaging over all separate performances) outperformed the stand-alone methods.

Ensemble methods for skin lesion diagnosis can be justified as mimicking the diagnostic process of dermatologists that relies on combining different visual characteristics from the lesions and even different opinions to improve performance. Its importance was further established by the work of Codella et al. (2017) [137]. Non-linear SVM's were build on hand-crafted features, sparse coding features, Caffe CNN features and Deep Residual Network (DNR) features separately. The global average of the probabilistic predictions of the separate classifiers served as performance measure and showed improvements with respect to the stand-alone performances. On the level of CNNs, a similar ensemble-learning approach was applied on features extracted from different layers of the VGG-nets and AlexNet, where SVMs were build for each feature-layer input and the probabilistic predictions of the different classifiers were averaged to compute the final classification results. In this study by Mahbod et al. [138], similar improvements in performance could be observed.

Further justification for the use of pretrained CNNs on the ILSVRC is given by Kawahara et al. (2016) [139]. Relying on the AlexNet architecture for transfer learning, features extracted from the pretrained net were highly performing inputs for classification with a logistic regressor. Furthermore augmentation of the dataset by rotations and left-right flips improved results and the lack of image segmentation did not endanger classification performance. The effectiveness of data augmentation was further portrayed in different studies, amongst others in [140] where the pretrained GoogleLeNet (as released in 2014 [72]) was finetuned.

The specific effect of transfer learning methods on performance in melanoma classification was investigated by the group of Menegola et al [141], where attention was given to the type of datasets used for transfer learning. The implemented architecture was the VGG-M net, ignoring the (fully-connected) output layer and using the output for an SVM classifier. Different schemes were subsequently tested, differing in their transfer learning scheme (no pretraining, pretrained on the ImageNet, pretrained on a retinopathy dataset, pretrained on both), fine-tuning (retraining only the SVM layer or retraining the whole net on the skin lesion set, then inserting the SVM layer). The best performance was achieved for the simple transfer learning scheme with weights trained on the ImageNet dataset and finetuning all layers on the skin lesion data. Thus, (supplementary) pretraining on a medical dataset and transferring this knowledge did not appear to be beneficial in the skin cancer classification task.

Further noteworthy are the winning results of the RECOD Titans team (Menegola et al.) [142] in the ISIC 2017 challenge for skin lesion classification, which focused on three specific classes of lesions: melanoma, seborrheic keratosis and benign nevi. The goal was a two-fold binary classification task: melanoma-others and seborrheic keratosis-others. The goal of their research targeted an extra push in the AUC-scores, for which they relied on the pretrained InceptionV4 and ResNet-101 models. The best results were achieved by relying on extending the dataset, both through supplementary data as through data augmentation, per-image normalization and using the combination of the probabilistic outputs of different models as input (i.e. feature vector consisting of the concatenated decisions from the different models) for an SVM classifier or an alternative high-level learning scheme. This shows the added strength of combining decision of different models in an extra learning scheme rather than for instance global averaging of the separate performances.

The landmark paper for skin-cancer classification with neural networks was published by Esteva A, Kuprel B, Novoa R et al. from Stanford University [143], and showed performance results at the same level of dermatologists. Opting for a data-driven approach, with a, mostly private, dataset of 129 450 clinical images (including 3374 dermatoscopic images), in which 2032 different diseases were represented, the need for extensive image-preprocessing, lesion segmentation and the establishment of hand-crafted features was eliminated. A transfer learning scheme was implemented based on the pretrained Inception-V3 model. The final classification layer was replaced by the skin cancer classification task (757 disease classes, generated from a disease partitioning algorithm) and the whole CNN was finetuned with RMSProp. For the three-class classification task between benign, malignant and non-neoplastic lesions (inferred from the broader disease partitions), a nine-fold cross-validation scheme portrayed an overall accuracy of $72.1 \pm 0.9\%$, which is the averaged accuracy over the individual classes. Furthermore, they reported an AUC-score of 94% for melanoma's.

Recent advances in incorporating unsupervised learning to improve classification performance

Throughout the duration of this work, a number of studies have been published in which unsupervised learning techniques were introduced, thus indicating exactly how 'hot' this field is. These techniques have been targeted to profit from the availability unlabeled skin lesion datasets or to render the feature learning process partially autonomous.

Work currently submitted to IET Computer Vision Journal by Creswell et al. [144] explored the use of auto-encoders for aiding benign vs. malignant classification of skin lesions. They proposed a semi-supervised training method for a denoising adversarial autoencoder model to tackle classification with a limited labeled dataset and exploiting a large unlabeled datapool.

A three-step process for melanoma classification was proposed by Arasi et al. [145], spanning preprocessing, feature extraction and classification. Classification was in their approach based on a stacked auto-encoder model, allowing each AE-layer to be trained in an unsupervised manner on the features it received from the preceding layer, with incorporation of a softmax layer on top of this framework to obtain classification. They reported adequate sensitivity and specificity results on features extracted based on discrete wavelet transforms.

Also dealing with dermatoscopic images, but focusing on the detection and localization of cutaneous vasculature, the work by Kharazmi et al. [146] similarly used stacked sparse auto-encoders for unsupervised learning of deeper levels of features, followed by a supervised training phase of an output classification model.

3.2.4 Challenges

Through the availability of the ISIC Archive, CAD for skin lesion diagnosis has been a popular research area in recent years, as illustrated in the previous discussion. As each study tackles individual components and focuses on specific aspects, a more global, comparative overview becomes somewhat more strenuous to obtain. Furthermore, the use of unlabeled data in this field in still in its early stages with only a limited amount of research focused on this.

Open lines of research may focus on squeezing the very last improvements out of existing frameworks to improve the AUC-scores. Alternatively, a more horizontal approach at a lower level can be chosen, in which an exploration is performed on just what is possible with a limited and unbalanced dataset. Not only will attention have to be given to problems of overfitting and results skewing to the overrepresented classes, the question as to what type of ‘pretraining’ of the network is most feasible and delivers the best results. It is clear by now that such limited datasets do not allow using deep learning techniques in itself, i.e. by random weight initialization and training. Therefore, focus can be placed on transfer learning and supervised finetuning of pretrained models. Another course of action targets the use of unlabeled data via e.g. auto-encoders as an alternate type of pretraining.

Chapter 4

Research Questions & Methods

After this extensive discourse on both deep learning and skin lesion (patho-)physiology, supported by existing tools in this field, we have now paved the way for introducing what contributions we will make in the field of skin lesion classification. This chapter will commence by stating the objective, remaining rather nebulous. This is then deciphered in precise research questions (RQs) that will guide our investigations. After introducing the datasets and implementation frameworks used for this work, respectively in Sections 4.2 and 4.3, the methods used to explore the RQ's will be detailed. Already lifting a corner of the veil, one of the primary focuses of this work is concerned with pretraining of CNN architectures. Hereto, three primary frameworks will be investigated, two following the transfer learning approach and one delving into the use of unlabeled data via autoencoders. This chapter is by consequence structured to mimic our research goals, with Section 4.5 devoted to the use of transfer learning and Section 4.6 describing the proposed auto-encoders. Finally, a brief discussion of the performance measures used for the evaluation of the models is given in Section 4.7.

4.1 Objective and research questions

The objective of this work will be to use deep learning for a skin lesion classification task on a **limited** and **unbalanced labeled dataset**, with indication of **adequate performance measures** given the **medical nature** of the problem, and further extending this application to the use of **unlabeled data**.

The objective illustrates several of the key aspects to which we will give attention. For each of them, we will delve into more depth on specific facets where the research questions will serve as a guideline for a coherent investigation. The RQs can be described as follows:

1. *Which transfer learning scheme offers the most opportunity for the skin lesion dataset?*

    - In terms of the type pretrained neural networks, do the trends in their performance on the ILSVRC hold op when transferring to the skin lesion dataset? A comparison will be made of the performance of the pretrained state-of-the-art models VGG16, VGG19, Inception-V3 and Resnet50.
    - With regard to the type of transfer learning scheme, a comparison will be made between the two main cases. Furthermore, for each case, a specific subquestion will be investigated. Firstly, the pretrained model can be used as a feature extractor and a classifier can be build on top of this. For this case, could combining the strengths of different models be merging features also increase performance and which combinations are more powerful? Secondly, the pretrained model can be finetuned after replacing the task-specific output layer(s) of the model. As only a limited labeled dataset is available, is there any existing correlation between the depth of finetuning the model and the performance on the skin lesion dataset?

2. *In light of the limited and imbalanced medical dataset, in which healthy classes tend to be more represented, what measures can be taken to apply deep learning whilst limiting the effects of overfitting and skewing the algorithm towards the largest class?*

    - Different strategies can be lined out to cope with an unbalanced dataset and a comparison will be made between an array of approaches. Before training, data augmentation is often applied, which can be performed both in the data and in the feature space. This technique can target to upsample only the minority classes by duplications or artificial renderings based on the available, minority class samples. Alternatively, samples of all classes can be modified at random to simulate a larger dataset. During training, class weighting can be applied to penalize misclassification of samples in the underrepresented classes. After training, evaluation metrics should be handled with care and a valid measure should be chosen which can reflect performance w.r.t. unbalanced classes. Which of these strategies is most apt for applying on the skin lesion classification problem?
    - As in any machine learning application, attention should be paid to overfitting. Specifically for the case of finetuning pretrained models, when will overfitting occur (correlated with depth of finetuning) and how can early overfitting be reduced? Different strategies will be explored, which include random data augmentation in the data space and dropout as a regularization measure.

3. *Given the extent of skin lesion images available publicly over a wide variety of categories, but lacking an (official) diagnosis, can this unlabeled data pool be exploited to improve performance through the use of unsupervised learning?*

    - As an alternative to ‘kick starting’ the training of skin lesion classification with transfer learning, can auto-encoders be used for pretraining? An auto-encoder framework will be

Table 4.1: Dataset 2017 ISIC Challenge

Dataset	**Total**	**Melanoma**	**Seborrhoeic Keratosis**	**Nevus**
Train	2000	374	254	1372
Validation	150	30	42	78
Train_total	2150	404	296	1450
Test	600	117	90	393

explored for a large, unlabeled dataset of skin lesions, after which the encoder can be used for supervised learning based on the labeled dataset.

4.2 Datasets

Labeled data The data used to investigate transfer learning schemes in this work is the ISIC 2017 Challenge: 'Skin Lesion Analysis towards Melanoma Detection' [147]. This dataset is composed of 2000 training images (374 malignant and melanocytic melanoma, 254 benign and non-melanocytic seborrheic keratosis and 1372 melanocytic and benign nevus), 150 validation images and 600 test images, indicated in Table 4.1. The training and validation sets can also be merged for the purpose of increasing data availability for training through cross-validation schemes. This will be referred to as the train_total dataset.

Unlabeled data In addition to the yearly release of labeled datasets for the challenges, the ISIC Archive offers large datasets of different skin lesions. The collection of 'untagged' skin lesions as published on the ISIC Archive is used in this work as unlabeled dataset, which consists of a total of 10 916 images [147].

4.3 Implementation

4.3.1 Programming language and framework

Python is used as programming language throughout this work. For the implementation of the neural networks, the Keras library [148] is used in light of it being highly suitable for handling neural models. The Keras backend is chosen to be TensorFlow [149], which offers the high performance numerical computations. The entirety can run on either CPU or GPU and is rapidly becoming a favorite in industry and research.

4.3.2 Experiments and hardware

The pretrained neural networks were loaded from the Keras Applications library and are referenced w.r.t. the original authors throughout this work. All experiments concerning finetuning and training of neural networks are run on GeForce GTX 770 GPU (with 3.64 GiB of free memory).

4.4 Classification task

Before commencing on the discourse of used methods to tackle the RQ's, we first state the precise classification objective of this work. Given the three types of skin lesions, we will focus on malignant versus benign classification, which entails a binary classification task of melanoma versus seborrhoeic keratosis/nevus.

However, to avoid losing the information contained in class-specific features of the merged lesions, all implemented CNN schemes will be trained for a 3-class classification task. To achieve the binary classification results, the probabilistic outputs of the latter two classes will be merged. This draws inspiration from the approach of the group at Stanford [143], where probabilistic inference was used on a much larger range of skin lesion diagnoses.

4.5 Transfer learning

4.5.1 Preprocessing

Of importance in the application of deep neural networks are certain preprocessing steps. Following [142] and [143], it can be remarked that opting for larger datasets might replace general preprocessing steps such as contrast enhancement and lesion segmentation. In light of this, such measures will not be implemented here. However, another critical component is preparing the images for passage through neural networks. Many of these CNNs have been built to perform optimally on images of specific input size, e.g. VGG16, VGG19 and ResNet50 on 224x224 images and Inception-V3 and -V4 on 299x299 sized inputs. Most applications in skin lesion classification literature rely on the standard resizing functions in Keras. When reading in the images, this resize function, available from the Pillow-toolbox, will resample the images to the target size by the default of nearest-neighbor interpolation algorithm. It can be remarked however that squeezing these skin lesions into square images, both leads to a loss in resolution and a loss of information on the shape of said lesion, thus endangering the classification task.

Therefore, we opted to implement the approach of Krizhevsky et al. (2012) [43], which inspired the preprocessing of the ImageNet images to train the VGG16 and VVG19 architectures, as done by Simonyan & Zisserman (2014) [71]. The jpg-training images are read in and isotropically rescaled to the smallest side having a length S, known as the training scale. Given the small size of the dataset, data augmentation is required ([140], [142], [143]). This is achieved through various means. Firstly, a random 224x224 crop is performed on the image. If S = 224, then one side will be completely encaptured by the crop. However, more often S is taken to be 256 and cropping will result in some loss of information. In this work, S = 256 for VGG16, VGG19 and ResNet50 applications and S = 384 for InceptionV3 preprocessing. Secondly, the images are randomly rotated (over 0, 90, 180 or 270 degrees) and flipped horizontally with a probability of 50%. Furthermore, the images are normalized with respect to the

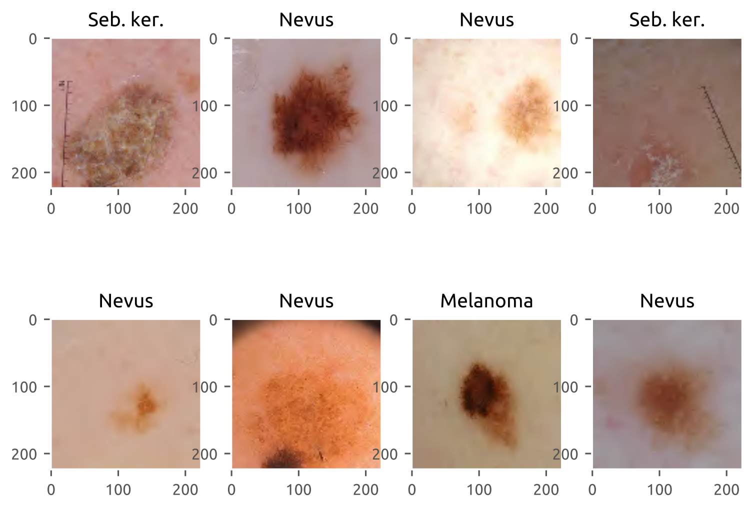

Figure 4.1: Illustration of our data augmentation for VVG/ResNet; Images courtesy to [147]. In general, the random 224-crops from 256-scaled images are able to capture the majority of the skin lesions.

ImageNet-data RGB-means as used to pretrain the specific CNN. These steps were implemented in a custom training datagenerator in Keras. Relying on real-time augmentation of the images, this decreases the computational memory requirements.

With regard to the testing and validation set, data augmentation is possible in these cases as well. However, as indicated by Simonyan & Zisserman [71], averaging the predictions over a large set of crops leads to minor improvements. Therefore, we leave this open as a future research path to get more stable predictions, but will not implement this here. The validation and test set are preprocessed by isotropically rescaling them to a smallest side of Q, i.e. the test scale. This value does not have to be equal to S, but in the case of VGG16, good performance was obtained with Q = S, thus Q was set to be 256 here for VGG/ResNet and 384 for InceptionV3. Subsequently, mindful of the importance of reserving the proportions, three methods are proposed to resize to 224 x 224: (1) following Krizhevsky, a central 224x224 crop is performed, possibly losing information along the outer edges; (2) Q is reduced to 224 and a central crop was performed, thus capturing whole-image statistics; (3) the Q x L (with L being the long side) images are padded to square L x L images, which are then resized to 224x224 format.

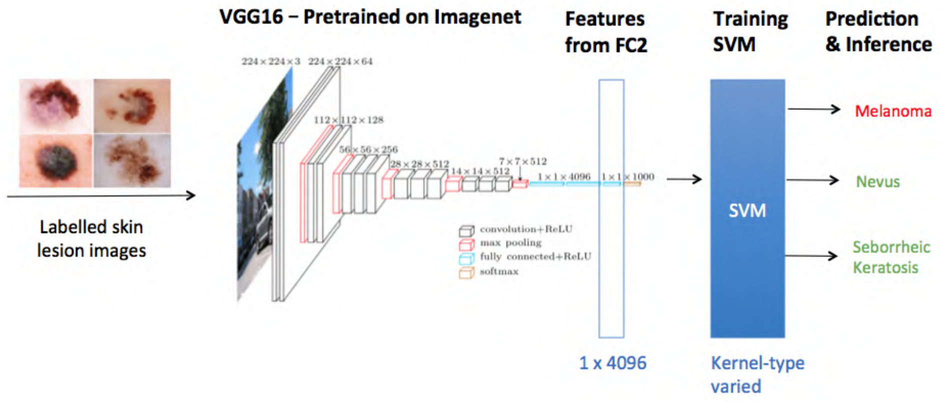

Figure 4.2: Illustration of CNN as Feature Extractor (partially adapted from [71] and [147])

4.5.2 Transfer learning schemes

As mentioned before, two major categories of transfer learning can be distinguished: (1) employing the pretrained CNN as a feature extractor for the skin lesion dataset and using these features as input for a classifier; (2) modifying the pretrained CNN for the task at hand through adaptation of the output layers and finetuning of the weights. For both cases, different aspects w.r.t. hyperparameters, class imbalance and overfitting problems will be explored according to the methods presented in this section.

4.5.2.1 CNN as feature extractor and classifier on top

A first line of research treats the CNN as a feature extractor, i.e. samples are run through the net and features are extracted from certain layers of the network, mostly deeper layers as they tend to be more specific. To investigate certain hyperparameters, our baseline methodology exists of extracting features from the last fully connected layer (FC2-layer, dim = 4096) of the VGG16-net and training an SVM classifier on said features. Comparisons were made between the three different read-in frames (and standardization conform VGG16-preprocessing), as discussed in 4.5.1, and the best results were obtained with the ‘central-crop’ approach, thus motivating this as choice for static read-in throughout this work.

> Establishing an SVM classifier equipped to handle the features extracted from FC2 of VGG16, as well as class imbalance through a 4-fold cross-validation scheme with exploration of different performance measures, reported as their means and std’s over the folds.
>
> *Hyperparameters SVM-kernel:* *{linear, polynomial, rbf, sigmoid}*
> *Hyperparameters class imbalance:* *{train, weighted, ROS, SMOTE, Augmented}*

In a first step, we explore the hyperparameters of kernel type and class imbalance handling through a 4-fold cross-validation scheme on the train_total dataset, reporting both performance means and standard deviations over the folds. The SVM-kernel will be varied between linear, polynomial, rbf and sigmoid. Furthermore, class imbalance handling will be investigated, as clinical diagnostic data often tends to be heavily skewed towards the healthy samples. As a straight-forward classification algorithm relies on the optimization of a certain loss-function that considers the classification accuracy of all samples in equal respect, this tends to cause the algorithm to fail for the minority class. Expressed through the accuracy measure, it can be observed that a very high accuracy may be due to classifying all samples in the majority class. Consequently, this not only renders the accuracy measure quite useless, it also points out the need for remedying this trap. We survey five options in more detail, namely (1) passing the raw data as baseline ('train'); (2) weighting the different classes according to their appearance ('weighted'); (3) ROS-upsampling of the CNN-extracted features ('ROS'); (4) SMOTE-upsampling of the CNN-extracted features ('SMOTE') and (5) Data-augmentation in the data space ('Augmented').

1. *Class-weighting*
   Class imbalance can be countered during training through the introduction of cost-sensitivity. Class weights are introduced in the learning algorithm and as such higher costs can be attributed to the misclassification of samples in user-specified classes. In our case, higher penalties are attributed to misclassification of samples in the underrepresented cases, which is reflected by the class weights of the melanoma (374 samples), sebKer (254 samples) and nevus (1372 samples) being respectively 1.7823, 2.62467 and 0.4859 to counter their imbalance.

2. *Upsamling of the underrepresented classes in the feature space*
   The second approach to tackle class imbalance targets the dataset rather than the learning algorithm and has as objective to upsample the underbalanced classes. As this transfer learning scheme operates on two levels (from raw data to features with CNN and from features to labels with SVM), it is possible to augment the features of the underrepresented classes in the feature space. Hereto, two different techniques are applied: (1) random oversampling (ROS, [150]) relies on an at random duplication of samples of the minority class and adding those to the dataset; (2) a more refined method is known as Synthetic Minority Over-Sampling Technique (SMOTE, [150]). In this case, a number of neighbors (k_neighbors = 5 in this case) of an underrepresented sample will be considered and subsequently synthetic samples can be generated randomly along the line in the feature space connecting the sample and one of its neighbors.

3. *Data augmentation in the data space*
   Finally, we consider augmenting the data in the data space. According to the approach detailed in Section 4.5.1, new images are generated through random crops, rotations and a horizontal flip. Careful treading is required when working with data augmentation to avoid contaminating a validation/test set with augmented samples from the training set. Thus, it is worth specifying that in each iteration of the 4-fold CV scheme, the training data is doubled through data augmentation,

while the validation set remains untouched. In this approach, all images are augmented with equal probability, thus upsampling all classes. A modification a this level would be to implement data augmentation of only the underrepresented classes.

> Optimizing the rbf-SVM through a 3-fold cross-validation scheme.
>
> *Parameter γ: {0.00001, 0.0001, 0.001, 0.01, 0.1, 1}*
> *Parameter C: {0.0001, 0.01, 0.1, 1, 10, 100, 1000}*

After having established the proper way of data handling and the best-performing SVM kernel (disclosing here that the rbf-kernel outperformed the others should not be too large of a spoiler), the SVM model parameters can now be optimized through a new cross-validation scheme. At this stage, a 3-fold CV-scheme on the train_total-dataset was implemented to discover the best performing penalty parameter C and kernel coefficient γ. The performance measure chosen to evaluate the optimization on in this gridsearch is the AUC-score of the melanoma-class (subsequently referred to as the positive class).

> Investigating features extracted from different pretrained models, as well as different layers. Combining features of both from different layers, as well as models to examine whether this may improve classification performance.
>
> *Pretrained models: {VGG16, VGG19, ResNet50, InceptionV3}*
> *Layers: {FC1, FC2, flatten_1, avg_pool}*

Finally, the confirmation of all model parameters and hyperparameters, allows the use of the established classifier for different features. Accordingly, features are extracted from specific layers in different pretrained CNNs to identify which serve the best purpose as feature extractors. Features will be extracted from the two last fully connected layers of the VGG16 and VGG19, from the flatten_1 layer of the ResNet50 and of the avg_pool layer of the InceptionV3 net.

To examine the usefulness of combining different models to improve performance, the choice is made here to work with feature combinations in the feature space, simply by concatenating them. Thus, features from different layers of the VGG-nets are combined and features extracted from different nets entirely are merged to investigate whether or not this could improve classification performance. The performance of the different schemes is reported on the test set (600 samples), which remained unused up to this stage.

4.5.2.2 Finetuning a pretrained neural network

Another possibility to benefit from the existing pretrained neural networks is finetuning them to the task at hand. As only limited data is available, finetuning the entire neural network (as done by Esteva et al. at Stanford for the Inception-V3 model [143]) can be considered too daunting a task to achieve. Therefore, the primary objective of this part of the work is to research any (causal) correlation between depth of finetuning the CNN and performance for a limited dataset. This is often referred to as the difference between shallow finetuning, signifying that only the final layers can be tuned, and deep finetuning, which allows increasingly more layers to be adapted to the task at hand.

Our approach consists of replacing the final softmax-layer in the pretrained model, which is specific for the ImageNet classification task (i.e. 1000-dim), by a 3-neuron softmax-layer to obtain probabilistic outputs for the three classes. In light of the random weight initialization of the final layer, immediately training this alongside certain pretrained layers, results in disruptive gradients (too large over the final connections), thus endangering the training process. Therefore, ‘pretuning’ of the final softmax-layer is required ahead of finetuning pretrained layers. This was further supported by our preliminary finetuning tests on the VGG16-network (which were exploratory performed in light of its straightforward architecture). Supervised training of the model is done with the ADAM-optimizer [151], for which both learning rate and decay are varied, and the categorical cross-entropy loss.

To investigate the depth of finetuning research questions, we chose the pretrained ResNet50 neural network, as released in Keras by [73], as framework. Two incentives can be identified to motivate this choice. Firstly, results with the pretrained CNNs indicated the higher strength of the ResNet50 as feature extractor (cf. Chapter 5). Secondly, the preliminary VGG16 finetuning experiments resulted however in much lower AUC-scores than what was achieved with the previous method. To attempt to increase said scores, ResNet50 is used in further stages. Finally, it has to be noted that in order to allow comparison with the previous method, the training and validation data is merged and used as the ’train_total’ dataset and testing is performed on the test set.

> Examining depth of finetuning in the pretrained ResNet50 model (with replaced output-layer), specifically taking into account the nets architecture. Furthermore, attention is paid to overfitting and class imbalance, with the line between measures to tackle these aspects separately starting to blur. Finally, care is taken w.r.t. the learning rate (Lr.) throughout training.
>
> *Depth of finetuning:* *{Block 1, Block 2, Block 3, Block 4}*
> *Overfitting and class imbalance:* *{Dropout, Data augmentation, Class weighting, (Lr.)}*

The primary concern for this approach is naturally overfitting, given the limited 2150-sized dataset. Furthermore, training of some layers of the deep ResNet50 structure with real-time data-augmentation is a time-consuming process. Therefore, we consider different techniques to reduce overfitting, whilst taking into account class imbalance, in separate steps, as discussed below.

1. *Finetuning the net with data augmentation*
   As a baseline, the 2150-sized training set is used for pretuning the new softmax output layer, while freezing all other parameters of the pretrained CNN. Subsequently, different schemes are investigated to counter overfitting: (1) augmentation of the under-represented samples to simulate class balance (results in 4350-sized dataset), (2) real-time data augmentation without class weighting and (3) real-time data augmentation supplemented by class weighting. After establishing the best-performing configuration, more layers are tuned to investigate the research question.

2. *Dropout with trainable part of CNN*
   After encountering significant overfitting in the previous approach and results below expectations (cf. Chapter 5), new regularization techniques were introduced at this stage. To avoid the time-intensive data augmentation and training of the whole ResNet50, features were extracted from the base model (i.e. the frozen layers) and used as input for the top model (i.e. the trainable layers with 3-class softmax layer on top to which dropout was added).

   Similarly to the previous approaches, a ground-comparison is made for a single trainable softmax-layer on the 2150 features extracted from the flatten_1 layer, to which subsequently dropout is added with different dropout rates and training is performed with variable learning rates. To investigate the depth of finetuning of the ResNet50-model, attention should be paid to its specific architecture. For a detailed overview of this design, we refer the reader to Appendix A. We first investigate finetuning inside the deepest identity block of the model. Next, we explore the finetuning of more blocks of the model to investigate the research question. At all stages, careful handling of the learning rate as well as the added dropout rate is required.

4.6 Use of unlabeled datasets

As was abundantly stressed throughout Chapter 2, pretraining of large CNNs forms a critical component in their applicability to new, smaller datasets. Up until now, we have explored the use of transfer learning for kickstarting the learning process of skin lesion classification. However, given the large abundance of unlabeled medical data, the question arises as to how one can exploit this in our training process. Therefore, we suggest two convolutional auto-encoder (CAE) models (with multiple hidden layers) as unsupervised learning framework to accomplish this. The encoders will subsequently be complemented by different top models, upon which both supervised learning of the top model and finetuning of the final CNN can be performed. Our aim is at this part to investigate how useful pretraining via CAE's of limited depth (as compared to e.g. InceptionV3 or ResNet) may be in comparison with transfer learning from another domain.

4.6.1 Preprocessing

Unlabeled data: Real-time data augmentation is applied to the images, identical to what was used for transfer learning as described in Section 4.5.1. As normalization procedure, the images are merely rescaled from the [0, 255] to the [0, 1] range.

Labeled data: To accelerate the supervised training process, the labeled images are read in through the 'central-crop' framework and similarly rescaled to the [0, 1] range. As demonstrated before, training data augmentation will increase performance, but places higher time demands on the training process, and is hence omitted at this stage.

4.6.2 Unsupervised learning: Convolutional auto-encoder (CAE)

Architecture of CAE We propose two different CAE architectures: (1) AE_1 incorporates three stages of convolutional layers, each one followed by a maxpooling layer, in its encoder and (2) AE_2 has an increased number of filters in these stages and a fourth stage is added. The design of the encoder and decoder part of the CAE are respectively detailed in Tables 4.2 - 4.3 and Tables 4.4 - 4.5. The choice is made to implement two convolutional layers after each other in the encoder to simulate a larger receptive field, while keeping the parametric demand reasonable, similar to what was implemented in the VGG-nets [71].

For the decoder, the architecture is mirrored to obtain the decoded image, with upsampling layers replacing the maxpooling layers. In the convolutional layer making up the final stage, a sigmoid activation function is chosen to obtain mapping of values to the [0, 1] range. This is supported by the choice of preprocessing, which incorporates a scaling of the input/target values to [0,1].

Training of CAE The ADAM optimizer is used for training the CAE, with the following hyperparameters: lr. = 1 e-4, β_1 = 0.9, β_2 = 0.999, ϵ = None, decay = 0 and no AMSGrad (batch size = 32). The loss-metric chosen for optimization is the mean squared error to reflect the difference between the value of the input and decoded image. In light of the time-demanding training process, training was manually halted upon reaching early convergence area, as followed on training loss and accuracy graphs. This resulted in training times of approximately 55 h for AE_1 (200 epochs) and 93 h for AE_2 (350 epochs).

Table 4.2: Encoder architecture of AE_1 (# parameters = 139 424); Strides of the convolutional layer = (1,1) and of the maxpooling layer = pool size.

Input	Skin lesion image ∈ [0, 1]: size (224, 224, 3)
Layers	Conv2D (nb. filters = 32, kernel size = (3, 3), activation = ’relu’)
	Conv2D (nb. filters = 32, kernel size = (3, 3), activation = ’relu’)
	MaxPooling2D (pool size = (2, 2))
	Conv2D (nb. filters = 64, kernel size = (3, 3), activation = ’relu’)
	Conv2D (nb. filters = 64, kernel size = (3, 3), activation = ’relu’)
	MaxPooling2D (pool size = (2, 2))
	Conv2D (nb. filters = 128, kernel size = (3, 3), activation = ’relu’)
	MaxPooling2D (pool size = (2, 2))
Output	Encoded vector: size (28, 28, 128)

Table 4.3: Decoder architecture of AE_2 (# parameters = 286 883); Strides of convolutional layer = (1,1).

Input	Encoded vector: size (28, 28, 128)
Layers	Conv2D (nb. filters = 128, kernel size = (3, 3), activation = ’relu’)
	UpSampling2D (size = (2, 2))
	Conv2D (nb. filters = 64, kernel size = (3, 3), activation = ’relu’)
	Conv2D (nb. filters = 64, kernel size = (3, 3), activation = ’relu’)
	UpSampling2D (size = (2, 2))
	Conv2D (nb. filters = 32, kernel size = (3, 3), activation = ’relu’)
	Conv2D (nb. filters = 32, kernel size = (3, 3), activation = ’relu’)
	UpSampling2D (size = (2, 2))
	Conv2D (nb. filters = 3, kernel size = (3, 3), activation = ’sigmoid’)
Output	Decoded image ∈ [0, 1]: size (224, 224, 3)

Table 4.4: Encoder architecture of AE_2 (# parameters = 1 735 488); Strides of the convolutional layer = (1,1) and of the maxpooling layer = pool size.

Input	Skin lesion image ∈ [0, 1]: size (224, 224, 3)
Layers	Conv2D (nb. filters = 64, kernel size = (3, 3), activation = ’relu’)
	Conv2D (nb. filters = 64, kernel size = (3, 3), activation = ’relu’)
	MaxPooling2D (pool size = (2, 2))
	Conv2D (nb. filters = 128, kernel size = (3, 3), activation = ’relu’)
	Conv2D (nb. filters = 128, kernel size = (3, 3), activation = ’relu’)
	MaxPooling2D (pool size = (2, 2))
	Conv2D (nb. filters = 256, kernel size = (3, 3), activation = ’relu’)
	MaxPooling2D (pool size = (2, 2))
	Conv2D (nb. filters = 512, kernel size = (3, 3), activation = ’relu’)
	MaxPooling2D (pool size = (2, 2))
Output	Encoded vector: size (14, 14, 512)

Table 4.5: Decoder architecture of AE_2 (# parameters = 4 094 787); Strides of convolutional layer = (1,1).

Input	Encoded vector: size (14, 14, 512)
Layers	Conv2D (nb. filters = 512, kernel size = (3, 3), activation = ’relu’)
	UpSampling2D (size = (2, 2))
	Conv2D (nb. filters = 256, kernel size = (3, 3), activation = ’relu’)
	UpSampling2D (size = (2, 2))
	Conv2D (nb. filters = 128, kernel size = (3, 3), activation = ’relu’)
	Conv2D (nb. filters = 128, kernel size = (3, 3), activation = ’relu’)
	UpSampling2D (size = (2, 2))
	Conv2D (nb. filters = 64, kernel size = (3, 3), activation = ’relu’)
	Conv2D (nb. filters = 64, kernel size = (3, 3), activation = ’relu’)
	UpSampling2D (size = (2, 2))
	Conv2D (nb. filters = 3, kernel size = (3, 3), activation = ’sigmoid’)
Output	Decoded image ∈ [0, 1]: size (224, 224, 3)

Table 4.6: Top model with a single dense layer, depicted for stacking on AE_1 (# parameters = 301 059).

Input	Encoded vector: size(28, 28, 128)
Top layers	Flatten ()
	Dropout (rate = r)
	Dense (size = 3, activation = 'softmax')
Output	Label prediction: $p_i \in [0,1]$ (i = 1, 2, 3)

4.6.3 Supervised learning: CNN

Architecture of CNN To exploit the pretrained encoder-part of the CAE, a top model is to stacked on the encoder-base. We explore three different top models in this work to maximally make use of the labeled, supervised data. A first and obvious choice is a mere fully-connected layer, applied on the flattened encoder-output with added dropout, as shown in Table 4.6. In light of the high parametric demand this top-model places on the learning process, we opted to also investigate performance with residual blocks as top model. Not only are these residual blocks hypothesized to facilitate training by learning the residual mapping, they also decrease the number of parameters while allowing a multi-layered top model (e.g. adding another fully-connected layer to the dense top model increases the number of parameters by millions). For the precise details on the ResNet architectures (specifically ResNet50), we refer the reader to Appendix A. We investigate both a top model configuration based on an identity block and one on a convolutional block, as shown in Tables 4.7 and 4.8. For all top models, the output fully-connected layer has a softmax activation function to obtain the probabilistic 3-class classification results, from which again binary classification will be inferred.

Training of CNN Supervised training is done with the same ADAM optimizer as used for the unsupervised learning of the CAE and a batch size of 32. Class weighting is incorporated during training to cope with class imbalance. The hyperparameters of the training process will be the learning rate (lr.) and the amount of dropout (dropout rate r). Mimicking our procedure for transfer learning, supervised learning is performed in two stages: (1) a supervised learning phase of the top model on the encoded vectors of the training set and (2) a finetuning phase of the stacked encoder and top model.

> Comparing the different top models throughout a supervised learning phase of said top models over 2000 epochs.
>
> *Hyperparameter Lr.: {1 e-4, 5 e-5, 1 e-5, 5 e-6, 1 e-6}*
> *Hyperparameter Dropout rate r: {0.025, 0.05, 0.1, 0.2}*

Table 4.7: Top model based on identity block (cf. Appendix A), depicted for stacking on AE_1 (# trainable parameters = 60 163 and # non-trainable (BN) parameters = 512). For the use on top of AE_2, the filters are adapted to [f1, f2, f3] = [128, 128, 512].

Input	Encoded vector: size(28, 28, 128)
Top layers	identity_block (k = 3, [f1, f2, f3] = [64, 64, 128])
	AveragePooling2D (size = (7, 7))
	Flatten ()
	Dropout (rate = r)
	Dense (size = 3, activation = 'softmax')
Output	Label prediction: $p_i \in [0,1]$ (i = 1, 2, 3)

Table 4.8: Top model based on convolution block (cf. Appendix A), depicted here for stacking on AE_1 (# trainable parameters = 304 899 and # non-trainable (BN) parameters = 2560). For the use on top of AE_2, the filters are adapted to [512, 512, 2048].

Input	Encoded vector: size(28, 28, 128)
Top layers	convolution_block (k = 3, [f1, f2, f3] = [128, 128, 256], s = 2)
	AveragePooling2D (size = (7, 7))
	Flatten ()
	Dropout (rate = r)
	Dense (size = 3, activation = 'softmax')
Output	Label prediction: $p_i \in [0,1]$ (i = 1, 2, 3)

4.7 Evaluation of the models

The performance of the different frameworks for classification is evaluated on the test set containing 600 samples, which remained unused up to this point. Classification results for the binary classification task are obtained from inference of the probabilistic results from the three-class classification training.

With regard to the evaluation of the models, caution must be taken regarding the interpretation of the model performance measures. As mentioned before, the overall accuracy tends to be a less useful measure in this respect and will here be reported as a mere illustration. Of high importance in computer-aided diagnostic systems is the confusion matrix, as depicted in Table 4.9, and its related measures.

In light of the desire of a CAD-system for melanomas focusing on early prevention measures, the implemented classification scheme would ideally target to not miss any positive samples, i.e. rather a false positive than a false negative. This can be expressed through the precision and recall measures,

Table 4.9: Confusion matrix

True label	*Melanoma*	TP	FN
	Seb. ker. / Nevus	FP	TN
		Melanoma	*Seb. ker. / Nevus*
		Predicted label	

with precision being the amount of percentage of correctly positive identified lesions,

(4.1) $$\text{Precision:} \quad P = \frac{TP}{TP + FP}$$

whereas recall (also known as the sensitivity) entails the amount of truly positive lesions that have been identified as such.

(4.2) $$\text{Recall:} \quad R = \frac{TP}{TP + FN}$$

Often used for the evaluation of CAD-systems in the medical world, is a measure referred to as the AUC-score. The performance of the CAD-system (classifier in this case) can be described in terms of both the benefits of classification (correctly classifying the disease/condition) and its costs (chances of false alarms or false positives) [152]. This is accordingly visualized in a graph, depicting the true positive rate (TPR, sensitivity) versus the false positive rate (FPR, 1 - specificity). For any probabilistic classifier (or a deterministic classifier from which probabilistic outputs can be obtained, as is e.g. done for SVM outputs through logistic regression), its classification results on the data will depend on the decision threshold used by the system. Therefore, a range of different classifiers based on these varying thresholds can be expressed through TPR(FPR) and visualized on the graph as such. This is known as the Receiver Operating Characteristic (ROC) of the classifier. The diagonal (TPR = FPR) depicts a random classifier and moving upwards from this, indicates an increase in performance of the classifier [153]. The area-under-curve (AUC) is typically used as overall measure indicating the performance of the classifier, where AUC = 0.5 thus indicates a random classifier and AUC = 1 a perfect one.

(4.3) $$AUC = \int_0^1 TPR \cdot FPR \ \ d(FPR)$$

For supervised training of the CNN architectures used throughout this work, observations are made over a sufficient number of epochs to visualize overfitting. A moving average with a window of 5 over the epochs is applied on the results and results are consequently documented as the mean and std over 5 epochs; this will be denoted with an asterisk. The best performing models are subsequently chosen as those that either minimize test loss* or maximize test AUC*-score.

Chapter 5

Results & Discussion

Guided by the research questions and founded on the methods as described in Chapter 4, the obtained results will be presented throughout this chapter. Focus was placed throughout the previous chapter on the three implemented frameworks in relation to the choice of pretraining of CNN architectures, two in the line of transfer learning and one exploring the use of unlabeled data via auto-encoders. To facilitate readability, we will adopt in the respective sections the same structure and answer the research questions in the corresponding sections.

5.1 Transfer learning

5.1.1 CNN as feature extractor and classifier on top

5.1.1.1 Countering class imbalance in the skin lesion dataset

For the SVM-classifier on FC2-features from VGG16, as visualized in Figure 4.2, both kernel-type and class imbalance handling were investigated through a 4-fold cross-validation scheme. The obtained means and STD's of five different performance measures, the overall accuracy and specifically for the positive class the F1-score, precision, recall and the AUC-score, are reported in Tables 5.1 - 5.8. The accuracy is mentioned for the sake of completeness, but comparing the accuracy and F1-scores of higher-scoring models clearly portrays that a higher accuracy does not entail a higher recall/precision, with the latter being desirable as it specifies how the models cope with class imbalance.

As discussed in the previous chapter, the commonly used AUC-score measure for the positive class is a much more reliable indicator of the models performance. As can be observed from Table 5.9, the RBF-kernel outperforms the other kernel types. These SVMs are for all class imbalance handling schemes, with the exception of data augmentation, in the same ballpark. From Table B.11 significantly

Table 5.1: Averaged Accuracy-scores (4-fold CV)

	Linear	Poly	RBF	Sigmoid
Train	0.816	0.815	0.822	0.813
Weighted	0.816	0.823	0.824	0.815
ROS	0.799	0.79	0.79	0.76
SMOTE	0.799	0.79	0.79	0.76
Augment	0.806	0.817	0.817	0.812

Table 5.2: STD Accuracy-scores (4-fold CV)

	Linear	Poly	RBF	Sigmoid
Train	0.00472	0.00663	0.00472	0.00618
Weighted	0.00369	0.00840	0.00979	0.00886
ROS	0.00991	0.0137	0.0128	0.0167
SMOTE	0.00632	0.0134	0.0136	0.0151
Augment	0.00770	0.00796	0.00586	0.000829

Table 5.3: Averaged F1-scores (4-fold CV)

	Linear	Poly	RBF	Sigmoid
Train	0.16	0.21	0.29	0.13
Weighted	0.17	0.25	0.29	0.22
ROS	0.34	0.43	0.45	0.44
SMOTE	0.35	0.43	0.45	0.43
Augment	0.12	0.17	0.20	0.01

Table 5.4: Std F1-scores (4-fold CV)

	Linear	Poly	RBF	Sigmoid
Train	0.0593	0.0121	0.0348	0.0122
Weighted	0.0668	0.0520	0.0181	0.0369
ROS	0.0482	0.0231	0.0229	0.0353
SMOTE	0.0482	0.0231	0.0229	0.0337
Augment	0.0775	0.0356	0.0183	0.0112

Table 5.5: Averaged Precision-scores (4-fold CV)

	Linear	Poly	RBF	Sigmoid
Train	0.55	0.54	0.59	0.54
Weighted	0.55	0.61	0.61	0.55
ROS	0.44	0.44	0.44	0.39
SMOTE	0.45	0.44	0.44	0.39
Augment	0.38	0.59	0.56	0.21

Table 5.6: Std Precision-scores (4-fold CV)

	Linear	Poly	RBF	Sigmoid
Train	0.0901	0.0720	0.0491	0.0116
Weighted	0.0630	0.0770	0.0882	0.0926
ROS	0.0487	0.0313	0.0292	0.0352
SMOTE	0.0230	0.0318	0.0314	0.0320
Augment	0.137	0.0214	0.0833	0.25

Table 5.7: Averaged Recall-scores (4-fold CV)

	Linear	Poly	RBF	Sigmoid
Train	0.10	0.131	0.19	0.072
Weighted	0.11	0.16	0.20	0.14
ROS	0.28	0.43	0.45	0.49
SMOTE	0.29	0.43	0.45	0.49
Augment	0.072	0.10	0.12	0.0050

Table 5.8: Std Recall-scores (4-fold CV)

	Linear	Poly	RBF	Sigmoid
Train	0.0392	0.00821	0.0317	0.00821
Weighted	0.0482	0.0404	0.0162	0.0283
ROS	0.0456	0.0363	0.0214	0.0364
SMOTE	0.0366	0.0377	0.0214	0.0364
Augment	0.0526	0.0214	0.0128	0.00572

Table 5.9: Averaged AUC-scores (4-fold CV)

	Linear	Poly	RBF	Sigmoid
Train	0.724	0.755	0.765	0.706
Weighted	0.723	0.753	0.768	0.741
ROS	0.722	0.750	0.765	0.716
SMOTE	0.722	0.750	0.765	0.716
Augment	0.701	0.703	0.721	0.655

Table 5.10: Std AUC-scores (4-fold CV)

	Linear	Poly	RBF	Sigmoid
Train	0.0147	0.0272	0.0174	0.0259
Weighted	0.0144	0.0298	0.0240	0.0242
ROS	0.0163	0.0305	0.0231	0.0247
SMOTE	0.0158	0.0306	0.0231	0.0247
Augment	0.0514	0.0363	0.0272	0.0383

higher F1-scores (0.45 ± 0.03 versus 0.29 ± 0.02 or 0.03) can however be observed in the case of feature space augmentation through ROS or SMOTE (with the difference in performance between the two being non-significant). We remark that optimization procedures on sensitivity/specificity (by changing the decision threshold) of any of the SVMs with similar AUC's would result in similar F1-scores. However, the immediate increase in offset of the F1-scores (attributed here to increases in recall), are favorable to circumvent the optimization procedure (as this is not the main point of focus in our work), but nonetheless give an indication of what the classifier may be able to do in terms of recall/precision. The recall-score being the number of positive samples that have been correctly identified as such (i.e. how many of the melanoma's were detected by the classifier), is a highly desired characteristic of the skin lesion CAD-system.

As can be observed, data augmentation in the data space, which was employed to double the dataset with randomly augmented images leads to a drop in performance. This is possibly due to the data augmentation approach implemented in this work, where random, off-center crops are made, thus making it possible for some edges of the lesion to be missing. This approach is often used in training CNNs with real-time data augmentation over a substantial amount of epochs. Thus, not only will the CNN receive different patches of the same lesion, which allows it to assess the whole image, the CNN will also be able to learn useful features from only patches. These results could therefore possibly indicate that the SVM requires a more standardized, standard-crop input for training. However, repeating these experiments would be necessary to make any sure statements in this regard.

> An rbf-kernel SVM obtained the best results on the FC2-features from VGG16. With regard to class imbalance handling, with the exception of data augmentation in the data space, all schemes resulted in models with similar capacities (represented in their AUC-scores). Nevertheless, the use of ROS or SMOTE in the feature space resulted in an immediate increase in F1-scores. Therefore, we chose to work with an rbf-SVM on SMOTE-augmented features.

5.1.1.2 Tuning of the SVM

In a next step, the SVM-specific parameters were fitted to the training data in an extensive cross-validation scheme. With established hyperparameters of an rbf-kernel and SMOTE data augmentation, the penalty parameter C and the kernel coefficient γ are evaluated and ranked according to their AUC-scores. The best results are achieved for C = 10 and $\gamma = 0.0001$.

> To summarize, the best cross-validation performance, taking into account both the AUC-score and the F1-score of the positive class, is an SVM with rbf-kernel, with C = 10 and $\gamma = 0.0001$, that was trained on features extracted from the second fully-connected layer of the VGG16, which were augmented with SMOTE.

Table 5.11: Test results SVM classifier - Features extracted from different nets.

CNN	Layer	Nb. features	Acc.	F1	Prec.	Recall	AUC
VGG16	FC2	4096	0.792	0.444	0.463	0.427	0.748
VGG16	FC1	4096	0.815	0.302	0.571	0.205	0.767
VGG19	FC2	4096	0.802	0.436	0.489	0.393	0.744
VGG19	FC1	4096	0.817	0.276	0.600	0.179	0.774
ResNet50	flatten_1	2048	0.797	0.520	0.482	0.564	**0.809**
InceptionV3	avg_pool	2048	0.750	0.457	0.396	0.538	0.729

5.1.1.3 Evaluation classification performance on combinations of features from different layers and different CNNs

The critical factor in the previous framework is the quality of the features extracted from the neural network. Therefore, the best SVM-classifier was trained on the total training set (2150 samples) and evaluated on the test set (600 samples) for different CNNs and feature-layers to compare how powerful the features of the different nets are. The results are presented in Table 5.11. It serves to mention the dimensionality of the problem, as the 2150 feature vectors were upsampled by SMOTE to 4350 vectors. The dimensionality of said vectors are presented alongside the results. Performance is expressed in terms of the overall accuracy (for the sake of completeness), the positive class F1-score, precision and recall, and ultimately the AUC-score of the positive class as most indicative measure. The corresponding ROC-curves are portrayed in Figure 5.1.

In the case of the VGG-nets, it can be observed from the AUC-scores given in Table 5.11, that performance increases for features extracted from FC1 rather than FC2. This is in line of our expectations and indicates that there is indeed a significantly higher source domain-adaptation for the deepest layer (FC2) as compared to a less deep layer (FC1). Investigation of the deeper models portrayed indeed a substantial gain in performance for ResNet50. However, counter to our expectations, as InceptionV3 has a higher performance in the ILSVRC than the VGG19 net, features extracted from InceptionV3 did not give rise to any increased performance.

Precision and recall, or alternatively the F1-score as their harmonic mean, can be further optimized by choosing appropriate decision thresholds. By default the SVM-classifier will assign a sample to a class upon 50 % probability. However, according to the ROC-curve, a new threshold can be selected based on the calculated recall- and precision-scores at different thresholds. It is important to note that this thresholding might offer opportunities in a medical framework to select those criteria that have the highest diagnostic value and tune the classifier as such. As an illustration, thresholding is performed to optimize the sum of sensitivity (= recall) and specificity, with results presented in Table 5.12. Increases

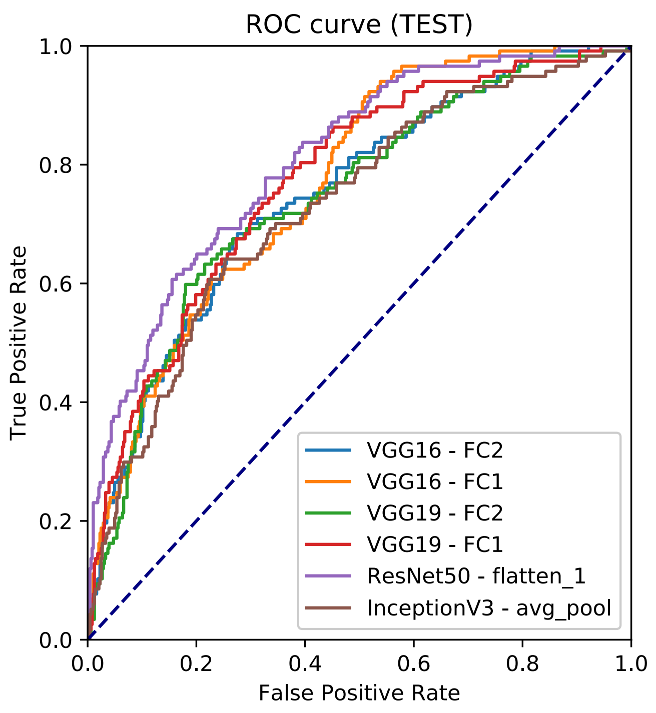

Figure 5.1: ROC-curves for testing of SVM classifiers trained on features extracted from the pretrained VGG16 and VGG19 (layers FC2 and FC1), ResNet50 (layer flatten_1) and InceptionV3 (layer avg_pool).

Table 5.12: Test results SVM classifier with thresholding (optimizing on sum of sensitivity and specificity) - Features extracted from different nets

CNN	Layer	Nb. features	Acc	F1	Prec.	Recall	AUC
VGG16	FC2	4096	0.770	0.465	0.426	0.513	0.748
VGG16	FC1	4096	0.807	0.0169	1.00	0.0085	0.767
VGG19	FC2	4096	0.802	0.457	0.490	0.427	0.744
VGG19	FC1	4096	0.805	0.0168	0.500	0.00855	0.774
ResNet	flatten_1	2048	0.809	0.520	0.440	0.632	**0.809**
InceptionV3	avg_pool	2048	0.710	0.463	0.362	0.641	0.729

in the F1-scores, attributed to increases in recall, can be observed except for the FC1-layers of both VGG-models.

We note that as the SVM classifier was not optimized for each case separately (only for FC2 from VGG16), this may have played a role in the below-expectation behavior of classification of features extracted from InceptionV3. The latter case may require a different parametric configuration of the

Table 5.13: TEST Results SVM classifier - Feature Combinations extracted from different nets

CNN	Layer	Nb. features	Acc	F1	Prec.	Recall	AUC
VGG16	FC2 & FC1	8129	0.812	0.261	0.556	0.171	0.760
VGG19	FC2 & FC1	8129	0.820	0.308	0.615	0.205	0.770
VGG16 & VGG19	FC2 & FC1	16384	0.815	0.255	0.594	0.162	0.766
ResNet50 & InceptionV3	flatten_1 & avg_pool	4096	0.798	0.510	0.485	0.538	**0.816**
ResNet50 & VGG16	flatten_1 & FC1	6144	0.815	0.335	0.500	0.239	0.783
ResNet50 & VGG19	flatten_1 & FC1	6144	0.820	0.299	0.622	0.197	0.792

SVM to separate the data for classification. Furthermore, no decision thresholding was possible for the FC1-layers of either of the VGG-nets, which upon inspection was attributed to overfitting of those SVMs on the training features (i.e. AUC-scores of 1 prevent choosing optimal thresholds). Considering that these overfitted SVMs nonetheless give high performances on the test set (AUC's of 0.767 and 0.774 for FC1 from VGG16 and VGG19), it is likely that establishing SVM parameters through cross-validation for the cases separately may further drive up performance. This was however omitted here due to time constraints.

Next, we investigated the fusion of features extracted from different layers and different nets. The obtained performance scores are reported in Table 5.13. For features extracted from the same net, as investigated for the VGG-nets, combining these features offered no improvement. However, this was subsequently extended to combinations of features extracted from different nets, for which we opted to combine the powerful ResNet50-features, with the best features from all the other nets. A combination of the two best cases (FC1 from VGG19 and ResNet50), did not improve performance compared to ResNet50 separately. However, a combination of ResNet50 with the seemingly less powerful InceptionV3 features did achieve this (81.9% AUC for combination versus 80.9% AUC achieved with features solely from ResNet50), indicating that both nets might be able to extract some complementary features.

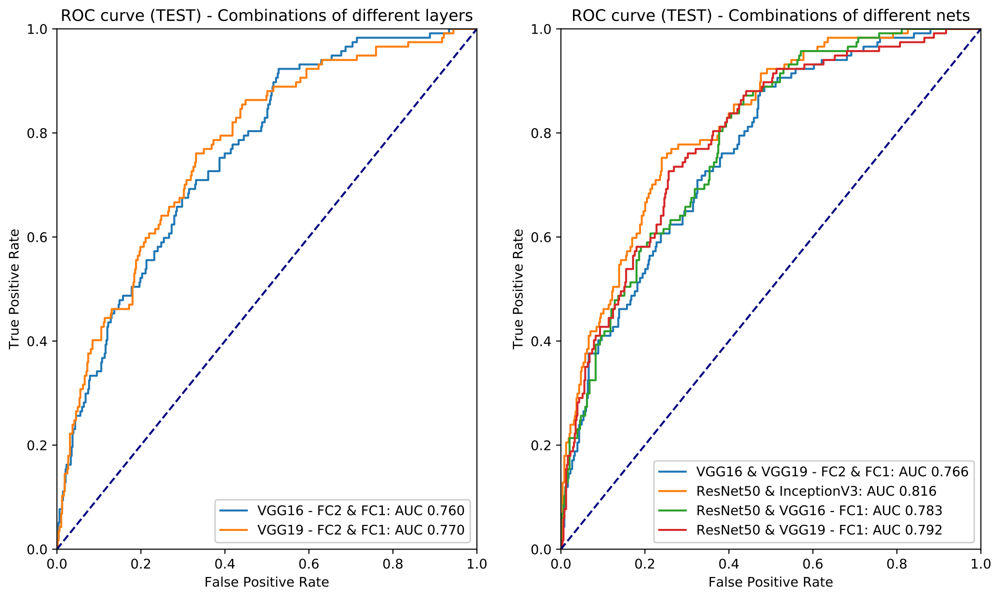

Figure 5.2: ROC-curves on test set for SVM classifier on feature combinations from different layers and different nets

> Exploring an SVM-based classification of features extracted from different state-of-the-art pretrained models, indicated a high performance, in terms of the AUC-score, for features from ResNet50, and adequate results for the FC1-features from VGG19 and VGG16. Combinations of features from different layers of the VGG-nets did not improve performance. Per contra, moving towards the use of features from different models, specifically those from ResNet50 and InceptionV3, did enhance performance.

We remark that notwithstanding the lengthy optimization procedure that was undergone, there is substantial room for improvement in these scores. This might however be intrinsically connected to the number of samples, as the training dataset was limited to the 2150-sized ISIC train_total data. It serves to mention that the higher scores , e.g. obtained in [142] and [143] with transfer learning approaches, always necessitate the use of more labeled data. To illustrate that our results are in the same ballpark as what has been achieved in literature on smaller datasets, we cite an AUC-score of 80.7 % and Recall of 47.6 % as achieved by the RECOD-group on the 2016 ISIC dataset [141].

Routes for improvement naturally include gathering more data, which is not the main study point of this work. Aside from that, another possibility would be to repeat the optimization procedure, implemented here for FC2-features from the VGG16-net, for all different feature-layer and CNN settings. Additionally, as mentioned before, the window opens here to further tune the decision threshold of the classifier to the specific needs of the classification task.

Table 5.14: ADAM Optimizer hyperparameters.

Lr.	Decay	Epoch	Loss* μ	Loss* σ	AUC* μ	AUC* σ
1e-4	0	16	**0.843**	17.7 e-3	0.628	2.10 e-3
		28	0.868	5.57 e-3	**0.641**	8.43 e-4
1e-4	1e-6	16	**0.843**	1.78 e-2	0.628	2.09 e-3
		28	0.868	5.52 e-3	**0.641**	8.39 e-4
1e-5	0	173	**0.830**	4.79 e-4	0.639	1.68 e-4
		203	0.839	5.42 e-4	**0.641**	1.30 e-4

5.1.2 Finetuning a pretrained state-of-the art model for our skin lesion classification

5.1.2.1 Preliminary exploration of VGG16 finetuning

Preliminary tests were performed focused on finetuning the VGG16 model in light of its simple architecture. As two key observations were made throughout this process, which supported our methodology choices for the following investigations, it is worthwhile to mention here.

Firstly, the random weight initialization of the new 3-neuron softmax layer requires ‘pretuning’, i.e. separate training of the said layer with the entire pretrained base model in frozen state. This can of course be attributed to the gradients over this last layer being too high initially, thus being highly disruptive to full training of the model. Secondly, even after finetuning of different layers, the obtained AUC-scores were substantially lower than what was achieved with SVMs. Consequently, and in light of the promising results of the ResNet50 model, the latter was chosen as further framework to investigate the depth of finetuning research questions.

5.1.2.2 Finetuning based on whole ResNet50-model

As discussed in the Chapter 4 and supported by the observations stated above, the pretrained ResNet50 with new, 3-class output layer was used to study transfer learning with finetuning. We discuss pretuning and finetuning of more layers separately for clarity.

Pretuning: Learning rate and decay Firstly, the learning rate and decay factor for pretuning the softmax layer were investigated with the ADAM optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, no AMSGrad) on a batch size of 16. The test loss and AUC-score are averaged over a window of 5 epochs and documented alongside their STD in Table 5.14 for the case of minimal test loss and maximal AUC-score. It can be observed that the highest AUC-scores were achieved at slight overfitting on the training data. Overfitting occurs before decay becomes relevant and stable results are achieved with Lr. = 1e-4, resulting in the choice of said learning rate without decay as hyperparameters.

Table 5.15: Class imbalance hyperparameters.

Scheme	Class weights	Lr.	Epoch	AUC* μ	AUC* σ
Train_total (size = 2150)	✓	1e-4	28	0.641	8.43 e-4
Balanced Train_total (size = 4350)	No	1e-5	61	0.604	2.59 e-4
Augmentation generator	No	1e-4	48	0.642	1.81 e-3
Augmentation generator	✓	1e-4	32	**0.652**	2.04 e-3

Pretuning: Overfitting and class imbalance In a second step, we investigated the class imbalance problem, of which the maximum AUC-scores (averaged over a window of 5 epochs) are reported in Table 5.15. Balancing the data by augmenting images of the underrepresented classes does not increase performance. It was observed that the substantial augmentation of the minority classes increases the chances of overfitting, and even at lower learning rates (Lr. = 1 e-5) this technique proved to be inefficient. Real-time data augmentation that creates new samples at random, with an equal probability of augmentation for samples in each class, during training of the model is more useful in this regard. Further improvements were achieved by adding the class weights, thus incorporating a class-penalty in the categorical cross-entropy loss. Finally, it serves to mention that in generally a slight postponing of overfitting was obtained when using data augmentation. In the case of augmentation generator with class weight a minimum test loss (0.891 ± 0.00258) was obtained at epoch 22, whereas this previously occurred at epoch 16, thus indicating that data augmentation also has a slight effect on early overfitting with limited datasets.

Finetuning: Training more layers Using this pretuned output layer and unfreezing other layers in the ResNet50 architecture showed in all tests with varying hyperparameters overfitting after only a handful of epochs. To give an indication, for finetuning of the final block in ResNet50 alongside the output layer, substantial overfitting occurred after 3 epochs and resulted in a high varations in AUC-score, with a maximum of approximately 0.7 vs. 0.652 ± 2.04 e-3 before finetuning extra layers.

Discussion: Discrepancy AUC-scores These results illustrate a very low training capacity of the pretrained ResNet50: max AUC*-score of 65.2% versus 81.6% as achieved by an SVM classifier on features combined from ResNet50 and InceptionV3. After extensive testing, the cause for this discrepancy was established to be due to the implementation of the 'freezing' of batch normalization layers in Keras.

Batch normalization (BN) layers, as proposed by Ioffe S. and Szegedy C. [154], proved to be effective at speeding up training, beneficial at countering explosive/vanishing gradients and has also been reported to have some regularization effects. The Keras implementation of BN-layers subdivides the BN parameters in trainable and non-trainable ones, respectively γ (scaling component) and β (shift) which are updated through backpropagation and μ and σ^2 representing the mini-batch statistics (mean and variance) during training or their running averages E and VAR during testing. Problems will however be experienced during finetuning of a pretrained model, where only part of the model is trainable. Freezing of the BN-layers entails a locking of the parameters γ, β, E and VAR, which were learned on the source-domain. During training, the principles of the BN-layer still hold up and the mini-batch statistics of the target-domain are propagated towards the deepest, trainable layers. During testing on the other hand, the stored E and VAR of the source-domain (in this case ImageNet) are used, which causes the trainable layers to receive inputs that are scaled differently than what they were trained with. This results in a drop in performance, but respective trends in performance as discussed before should however stand.

We chose to circumvent this problem by redefining a top model with the desired number of trainable layers based on the ResNet50-architecture and transferring the pretrained weights and biases to this architecture. Omitting the step of data augmentation, features are extracted from desired, frozen depth of the ResNet50-model which runs in inference mode and the top model can be trained on these features. Alternatively, the base-model could be explicitely hard-coded to run in inference mode to allow for testing with data-augmentation. However, given the time-demand of data-augmentation and as trends w.r.t. this hyperparamater were already observed before, this will not be explored here.

> A first series of experiments was conducted w.r.t. finetuning of the ResNet50-model, tackling the two stages of (1) ‘pretuning’ the target-domain specific output layer, freezing all other layers, and (2) ‘finetuning’ of multiple layers in the model pretrained on the source-domain (ImageNet), freezing up to a certain depth of the model. Results w.r.t. handling class imbalance indicated the suitability of a scheme that combined data augmentation and class weighting during training. The observations made during finetuning of more layers of the model did not aid us in the exploration of our research questions. However, a key observation was made concerning the workings of the batch-normalization layers in transfer learning. This led to an alternative methodology to investigate the depth of finetuning.

5.1.2.3 Finetuning based on a two-stage model

Pretuning: Learning rate and Dropout Constructing a separate top-model, we investigated pretuning of the final softmax layer for different learning rates and incorporation of dropout before this dense layer. The investigated hyperparameters are in the range of Lr. $\in \{1e-4, 1e-5, 1e-6\}$ and Dropout rate r $\in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ with the ADAM optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, no AMSGrad) and a batch size of 32. The results are presented in Table 5.16 and indicate similar performances of Lr. = 1 e-4 and Lr. = 1 e-5 in an acceptable time frame. However, as can be observed from Figure 5.3, the smaller learning rate of 1 e-5 gave more stable results with limited overfitting. The results regarding the incorporation of dropout are given in Table 5.17 and shown in Figure 5.4. A small dropout rate of 0.1 does not offer any merits at this stage, but can be useful for deeper finetuning. Too large dropout rates will be avoided as long as overfitting can be handled otherwise, as they result in a decrease in performance. The optimal weights based on lr = 1e-5 without dropout were thus implemented in the model.

Table 5.16: The test loss and test AUC-score during pretuning of the softmax layer with the ADAM optimizer for different learning rates.

Lr.	Dropout rate	Epoch	Loss	AUC* μ	AUC* σ
1 e-4	0	229	0.804	0.809	1.14 e-3
1 e-5	0	2445	0.810	0.806	2.49 e-4
1 e-6	0	4000	0.871	0.769	4.86 e-5

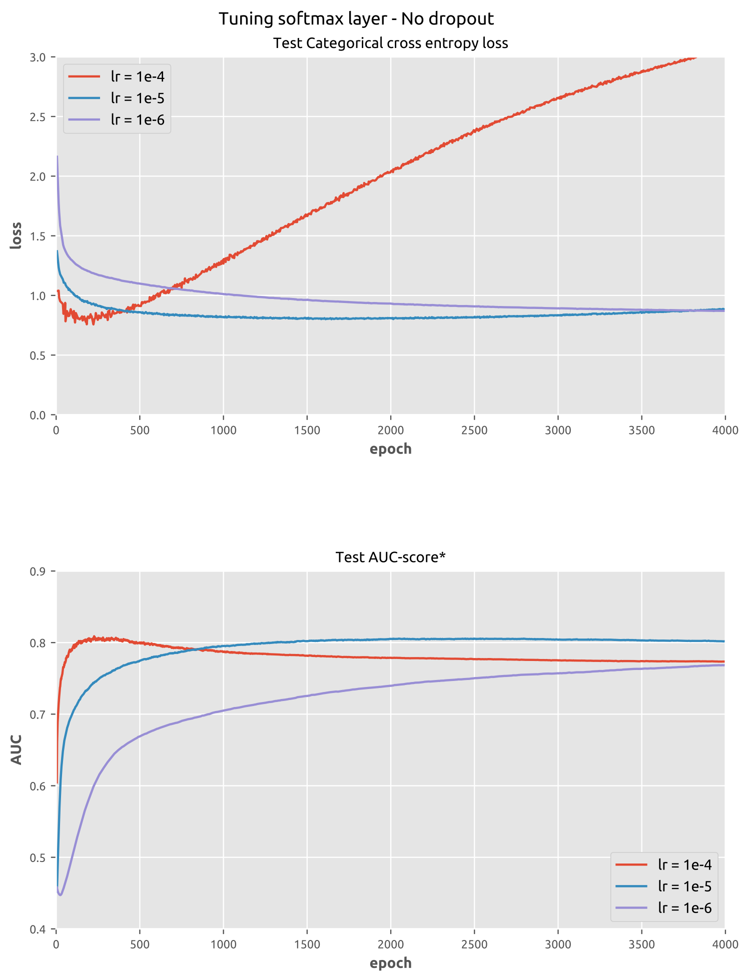

Figure 5.3: The test loss and test AUC-score during pretuning of the softmax layer with the ADAM optimizer for different learning rates.

Table 5.17: Investigation of dropout rate during pretuning of softmax layer.

Lr.	Dropout rate	Epoch	Loss	AUC* μ	AUC* σ
1 e-5	0	2445	0.810	**0.806**	2.49 e-4
1 e-5	0.1	2137	0.815	0.800	4.42 e-4
1 e-5	0.2	1942	0.816	0.792	2.25 e-4
1 e-5	0.3	3085	0.823	0.783	6.08 e-4
1 e-5	0.4	3894	0.836	0.774	3.03 e-4
1 e-5	0.5	3053	0.8398	0.765	8.57 e-4

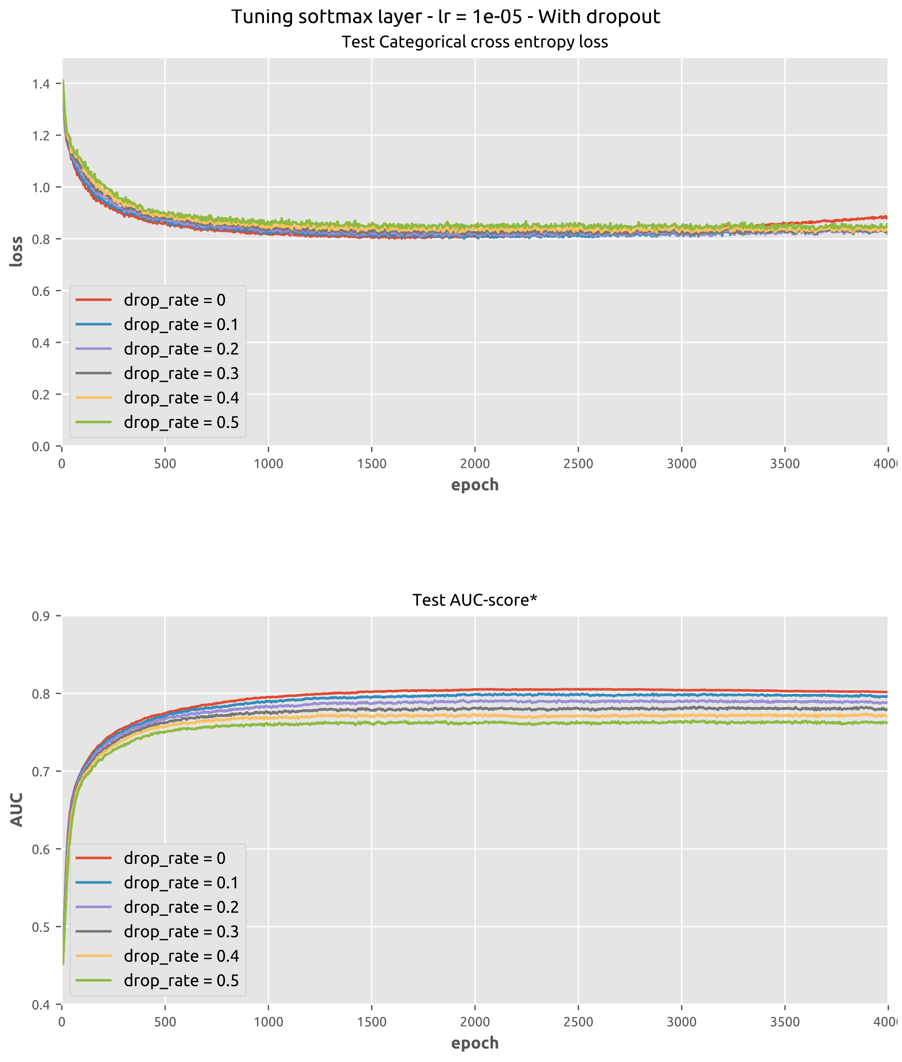

Figure 5.4: Investigation of dropout rate during pretuning of softmax layer.

Table 5.18: Final implementation of pretuning scheme and obtained max AUC*-score.

Baseline	# Epochs	Lr.	Drop rate	Epoch	AUC* μ	AUC* σ
Pretuning	4000	1 e-5	0	2445	0.806	2.49 e-4

Table 5.19: Finetuning inside Block 1 (cf. Appendix A) with varying learning rates and dropout rates. Finetuning of the deepest BN layer was attempted for the documented Lr.'s, but due to high variations in the results, reporting any scores is pointless. Tuning the deepest convolutional layer required a decrease in learning rates, as for Lr. = 1 e-5 resulted in heavy fluctuations.

Layer	# Epochs	Lr.	Drop rate	Epoch	AUC* μ	AUC* σ
bn5c_branch2c	1000	1 e-4	0	/	/	/
	1000	1 e-5	0	/	/	/
	1000	1 e-6	0	/	/	/
res5c_branch2c	1000	1 e-5	0	/	/	/
	1000	1 e-6	0	299	0.805	9.50 e-5
res5c_branch2b	1500	1 e-6	0	648	0.804	6.48 e-4
	1500	1 e-6	0.1	210	0.802	2.50 e-4
	1500	1 e-7	0.1	1455	0.799	5.40 e-5

In the proceeding work, deeper finetuning of the ResNet50-architecture was explored. We refer the reader to Appendix A, where this architecture and the related terminology are discussed in more detail. For the finetuning of different blocks, we will employ a simplified notation with Block i referring to the i^{th} block starting from the deepest level and moving upwards in the model. We thus denote the deepest identity block as Block 1, the identity block above it as Block 2, the next convolution block as Block 3, and so on (cf. Appendix A). The baseline after pretuning of the softmax layer is repeated in Table 5.18.

Finetuning inside Block 1 In a first step, finetuning inside Block 1 (i.e. the final identity block) is explored and results are reported in Table 5.19.[1] Inclusion of the uppermost BN-layer for training does not lead to any increase in performance and causes fluctuating results, in line with our expectations. Subsequently, the convolutional layers are explored inside this block. Again, no increase in performance is visible. Nevertheless, observations about the learning rate indicate that lower learning rates are required to exploit the pretrained weights, with higher learning rates leading to substantial fluctuations in the training results. These results can meet our expectations, as blocks are merely designed as a 1x1 convolution to reduce dimensionality, a functional 3x3 convolution, and 1x1 convolution to recapture dimensionality. Thus interfering in an intermediate stage of the computation of $F(\mathbf{x})$, which is used by

[1]For completeness and as they support the observations, hyperparameters of the experiments leading to fluctuating results are reported here, omitting the documentation of any values.

Table 5.20: Finetuning of different blocks in the ResNet50 model for different learning rates and dropout rates.

Block	# Epochs	Lr.	Drop rate	Epoch	AUC* μ	AUC *σ
Block 1	1500	1 e-6	0.1	640	**0.822**	6.39 e-4
	1500	1 e-7	0.1	1468	0.811	1.55 e-4
Block 2	1000	1 e-6	0.1	678	0.819	7.98 e-4
	1000	1 e-6	0.2	386	0.819	4.66 e-4
	3000	1 e-6	0.3	303	**0.819**	1.94 e-4
	4000	5 e-7	0.2	678	0.818	3.95 e-4
	4000	1 e-7	0.1	3250	0.818	5.65 e-5
Block 3	2500	5 e-7	0.2	400	**0.818**	4.02 e-4
	5000	1 e-7	0.1	2288	0.813	1.36 e-5
Block 4	4000	1 e-8	0.1	3967	**0.779**	1.79 e-4

the residual block to compute the mapping function $H(\mathbf{x}) = F(\mathbf{x}) + \mathbf{x}$, was indeed not expected to lead to any noticeable improvements in performance.

Finetuning of different blocks After an inspection of finetuning inside Block 1, our focus shifted to the finetuning of multiple blocks to investigate the depth of finetuning and its possible enhancement of performance. We finetuned up to 4 blocks in the ResNet50 model and the most relevant experimental results are shown in Table 5.20. An increase in performance can be observed when incorporating the first block in finetuning process (AUC of 0.822 compared to 0.806 after pretuning).

This enhanced performance can be maintained, but not surpassed (despite extensive testing with hyperparameter settings), when incorporating Block2, and additionally Block 3. Furthermore, to achieve stable training results when adding said blocks, a decrease in learning rate was required to avoid destroying the pretrained weights. Adding new blocks introduces a substantial amount of trainable parameters to the model and the trade-off between increasing AUC-scores and avoiding overfitting becomes more difficult to balance. This becomes increasingly clear when adding further blocks, as can be observed from the results for Block 4. A decrease of the learning rate to 1e-8 was required to allow for training and over a duration of 4000 epochs (approximately 18h) the AUC-score only managed to arrive at a value of 0.779.

To study the depth of finetuning of the ResNet50 model to enhance its classification performance on the skin lesion dataset, focus was placed on the finetuning of additional blocks moving upwards in the model. An increase in AUC-score was obtained by adding an extra trainable block to the process and this could be maintained when adding two additional blocks. However, no further increases in performance could be reached and decreases in learning rate were required to stabilize the training process. Finetuning of a total of four blocks led to a decrease in performance (at least in an acceptable time-window). This thus portrays a limited, but existing window of opportunity to enhance performance on the skin lesion dataset when finetuning starting from a specific depth in the ResNet50 model.

5.2 Convolutional auto-encoders for pretraining on unlabeled skin lesion data

Two architectures for convolutional auto-encoders models were described in the previous chapter, one more shallow incorporating three convolutional stages in its encoder (further referred to as AE_1) and one slightly deeper model with an additional convolutional stage (denoted as AE_2). Both AE-models were trained according to the procedure described in the Methods chapter.

We subsequently explored the supervised training of different top models on the pretrained encoders of both AE_1 and AE_2, of which the obtained results are documented in Section 5.2.1. All results are presented stating the used learning rate and dropout rate and indicating the maximal AUC-score (mean and STD over a window of 5 epochs) and the corresponding epoch at which the maximum was achieved. The results were also visualized, for which we refer to reader to Appendix B. Finally, the supervised finetuning of the entire CNN-model (pretrained encoder and trained top model) was initiated. Only a minimal exploration was possible and the acquired result is reported in Section 5.2.2.[2]

5.2.1 Supervised training phase of different top models on both AE_1 and AE_2

Supervised training of a dense top model, indicated that only a single dense layer (after dropout on the flattened features extracted from the encoder) was trainable. Adding another dense layer entailed too high an increase in parameters to allow for supervised training with the limited labeled dataset. The best results obtained for this model indicated a maximal AUC-score of 0.6370 ± 9.05 e-4, as documented in Table 5.21.

[2]In light of the limited time window of this work, the hyperparameter investigation of finetuning the entire CNN model could only briefly be touched upon.

Table 5.21: Results from supervised training phase of a dense layer as top model on AE_1, the latter being frozen throughout this process.

Lr.	Drop rate	Epoch	AUC* μ	AUC* σ
1 e-5	0.2	1987	0.6370	9.05 e-4

Table 5.22: Results from supervised training phase of the identity block as top model stacked upon AE_1, the latter being frozen throughout this process.

Lr.	Drop rate	Epoch	AUC* μ	AUC* σ
1 e-4	0.025	157	0.707	4.75 e-3
	0.05	360	0.694	5.05 e-3
	0.1	114	**0.720**	4.04 e-3
	0.2	667	0.709	1.40 e-3
5 e-5	0.025	177	0.712	1.56 e-3
	0.05	226	0.715	2.02 e-3
	0.1	948	0.727	6.22 e-3
	0.2	348	**0.729**	2.08 e-3
1e-5	0.025	1512	0.716	1.29 e-3
	0.05	954	0.703	1.63 e-3
	0.1	1345	0.705	1.53 e-3
	0.2	1327	**0.7371**	9.97 e-4
5e-6	0.025	1917	**0.7132**	0.430 e-3
	0.05	1877	0.6932	6.95 e-4
	0.1	1926	0.7057	9.18 e-4
	0.2	1932	0.7037	9.97 e-4
1e-6	0.025	1960	0.6607	4.18 e-4
	0.05	1996	0.6623	3.73 e-4
	0.1	1990	0.6411	4.23 e-4
	0.2	2000	**0.6661**	5.62 e-4

Table 5.23: Results from supervised training phase of convolution block as top model stacked upon AE_1, the latter being frozen throughout this process.

Lr.	Drop rate	Epoch	AUC* μ	AUC* σ
1 e-5	0.025	440	0.6997	6.32 e-4
	0.05	1778	0.6971	7.63 e-4
	0.1	339	**0.708**	1.48 e-3
	0.2	1811	0.695	1.50 e-3
5 e-6	0.025	1977	**0.7362**	4.94 e-4
	0.05	1521	0.708	1.81 e-3
	0.1	1384	0.708	1.24 e-3
	0.2	1533	0.706	1.50 e-3
1 e-6	0.025	1971	**0.6799**	6.94 e-4
	0.05	1977	0.6672	8.77 e-4
	0.1	1956	0.6641	6.01 e-4
	0.2	1983	0.6734	8.15 e-4

Table 5.24: Results from supervised training phase of identity block as top model stacked upon AE_2, the latter being frozen throughout this process.

Lr.	Drop rate	Epoch	AUC* μ	AUC* σ
1 e-5	0.025	241	0.757	1.57 e-3
	0.05	279	0.7472	5.38 e-4
	0.1	1437	0.7474	7.36 e-4
	0.2	260	**0.7627**	6.81 e-4
5 e-6	0.025	653	0.740	1.34 e-3
	0.05	1528	**0.7625**	6.66 e-4
	0.1	756	0.756	1.00 e-3
	0.2	485	0.749	1.27 e-3
1 e-6	0.025	1991	0.7411	4.00 e-4
	0.05	1976	0.7449	4.76 e-4
	0.1	1999	**0.7589**	3.68 e-4
	0.2	1907	0.7315	3.33 e-4

Instantiating a top model based on the identity block (from the ResNet50 architecture), led to substantial improvements in AUC-scores, as can be seen from Table 5.22. From the different learning rates and dropout rates that were tested, an optimal AUC-score of 0.7371 ± 9.97 e-4 was obtained with Lr. = 1 e-5 and Drop rate = 0.2.

In the case of a convolutional block-based top model, similar results in AUC-score (max AUC of 0.7362 ± 4.94 e-3) could be obtained with a Lr. of 5e-6 and a dropout rate of 0.025, with results indicating that slightly lower learning rates were needed for stable training. All results are presented in Table 5.23.

As can be clearly observed from the previous results, the first AE_2 model is clearly useful for partial pretraining of the CNN for skin lesion classification. The unlabeled dataset appeared highly sufficient to train AE_1 constructed with 3 convolutional stages in its encoder and mirrored in its decoder. Therefore, a deeper convolutional auto-encoder model (AE_2) was also explored and hypothesized to be trainable with the unlabeled dataset. As stable training results were quickly obtained with an identity block-based top model, we chose to only explore the hyperparameters for such a top model stacked on AE_2. Results can be found in Table 5.24 and show increases in AUC-scores, as would be expected for a deeper CNN that lies within the training capacities of the available data. Maximum AUC-scores (0.7625 ± 6.66 e-4) are obtained with Lr. = 5 e-6 and drop rate = 0.05, giving more stable results than Lr. = 1e-5 and drop rate = 0.2 (cf. figures in Appendix B).

To conclude, an illustrative comparison of the best obtained configurations (AE-type, top model type, Lr. and Dropout rate) is shown in Figure 5.5. For AE_1, the dense top model results in a significantly lower performance than both the identity and convolution block-based top models. The latter two give rise to comparable results, with slightly smaller learning rates required for stable training of the convolution-block based model. The deeper AE_2 model was trainable with the available unlabeled dataset and helped to improve performances when stacked with an identity block-based top model.

5.2.2 Finetuning the entire CNN (AE and top model)

Based on the aforementioned results, we chose to finetune the CNN constructed of the pretrained encoder of AE_2 and the trained identity block-based top model. The hyperparameter setting of Lr. = 1 e-6 and Drop rate = 0.2 was used for training the whole CNN over 1000 epochs, with results given in Table 5.25. We note that due to computational memory requirements, a reduction of the batch size to 12 was necessary. Minor improvements could be seen in the AUC-score, we remark however that a thorough exploration of the further hyperparameter settings for this case is required. This is expected to further increase the AUC-score.

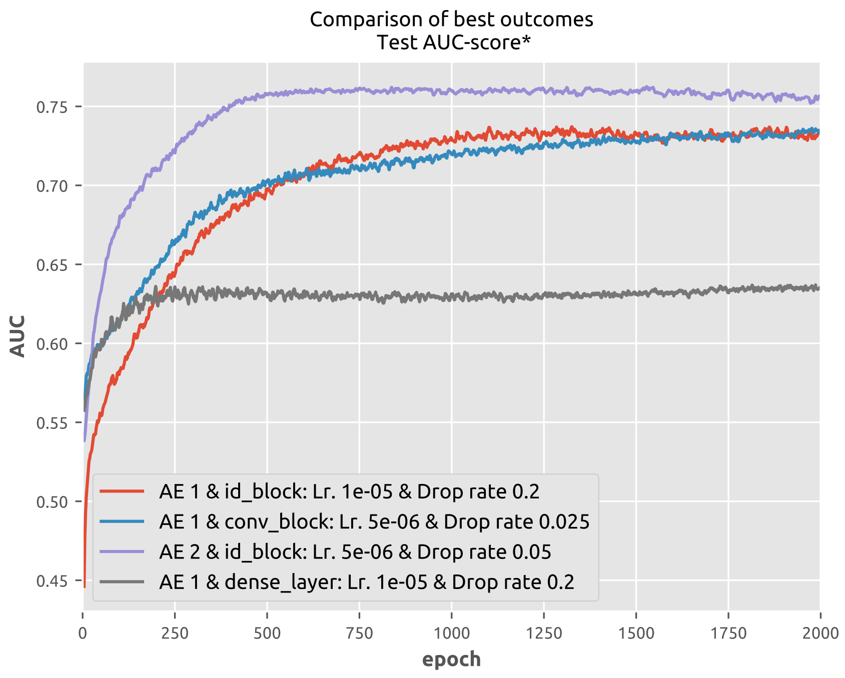

Figure 5.5: Comparison of the best supervised training results for different top models (identity block, convolution block and dense layer) on both AE_1 and AE_2 and learning rate and dropout rate as hyperparameters.

Table 5.25: Results from supervised training of the whole CNN-model for one hyperparameter setting.

Lr.	Drop rate	Epoch	AUC* μ	AUC* σ
1 e-6	0.2	427	0.7643	6.72 e-4

Investigating the use of CAEs for (partial) pretraining of a CNN architecture indicated adequate performances in terms of AUC-scores in the case of stacking with an identity or convolution block-based top model rather than a dense output layer. The highest scores were obtained with a slightly deeper AE-model, which incorporated 4 convolutional stages in its encoder and proved to be trainable with the available unlabeled data within an acceptable time frame. Further exploration of the hyperparameter settings is required for finetuning the entire CNN, but further increases in AUC-scores are expected.

Table 5.26: Illustrative comparison of the different settings explored throughout this work. Where applicable, the AUC-scores were averaged over a window of 5 epochs and both mean and STD is reported.

Case	**Hyperparameters**	**AUC**
SVM on pretrained VGG19-features	FC1 - features	0.774
SVM on pretrained ResNet50-features	flatten_1 - features	0.809
SVM on combination of pretrained ResNet50 and InceptionV3	flatten_1 from ResNet50 and avg_pool from InceptionV3	0.816
Finetuning ResNet50	Supervised training of pretuned softmax layer and Block 1	0.822 ± 6.39 e-4
Pretraining with CAE-framework	Training of AE_2 on unlabelled data and supervised training of identity-block based top model	0.7625 ± 6.66 e-4

5.3 Comparison of the different frameworks explored for pretraining and classification of skin lesions

To conclude this chapter, we indicate in Table 5.26 some of the most prominent results achieved for the various settings that were explored w.r.t. efficient pretraining of CNN architectures, opting for paths in the direction of transfer learning or auto-encoders. For the case of transfer learning, training SVMs on combinations of features extracted from different pretrained models (specifically from ResNet50 and InceptionV3) was successful in improving classification performance compared to SVM-classification on stand-alone features. Improvements in these results can be obtained through adaptation of the ResNet50 architecture to the skin lesion classification task and finetuning at least one additional block in the model. Finally, even though only a limited amount of pretraining was performed via the auto-encoders (due to time-constraints), the supervised learning phase of a top-model stacked on the pretrained encoder-base indicated AUC-scores exceeding than what was achieved with SVMs trained on FC2-features from the VGG-nets and the final features from the InceptionV3 net. The results for this scheme are almost in the same ballpark as what was achieved by the second to best feature + SVM framework (i.e. VGG19, FC1-features). Further exploration of CAEs is therefore desired, both in terms of training time of the auto-encoder and finetuning of the final CNN, as even this limited exploration of the framework indicates the strong capability performing pretraining of CNN architectures through CAEs.

CHAPTER 6

CONCLUSION & OUTLOOK

The development of CAD-system for skin lesion diagnostic tasks is in hot pursuit, with special attention being given to the diagnosis of melanomas. Given its high propensity to metastasize, going hand in hand with severe decreases in survival rates, and the high inter-patient variability in lesion appearance, as well as a strong requirement on the training of the physician/dermatologist properly diagnosing a melanoma can be considered a daunting task. Recent work has been focused on the use of deep learning for skin lesion diagnostic purposes and was the line of research explored throughout this work. Given the limited public availability of labeled datasets, the use of deep CNN architectures requires pretraining of the weights to 'kickstart' the supervised training process. We therefore chose this as our primary research objective, whilst tackling class imbalance and overfitting throughout.

A first methodology consisted of bening/malignant classification with SVM's on features extracted from state-of-the-art models (VGG16, VGG19, InceptionV3, ResNet50). Best performance was achieved with deep ResNet50-features, followed by FC1-features from VGG19. Combining features of different layers was investigated for the VGG-nets, but did not enhance classification performance. Furthermore, combining features from different networks through concatenation was investigated and an increase in performance was observed when combining ResNet50 and InceptionV3 features, possibly indicating that both nets extract slightly complementary features. The RBF-kernel SVM classifier was equipped to handle the class imbalance at hand, as indicated by the AUC-scores of the different schemes explored. An immediate offset, without requiring decision thresholding of the classifier, was observed when employing augmentation in the feature space through ROS or SMOTE.

Another approach in the transfer learning realm tackled the adaptation of the pretrained ResNet50 model to task at hand (by changing the final output layer), pretuning the weights of this layer and finetuning the structure. The depth of finetuning was studied for this approach and an increase in performance was documented upon training an additional block in the model. This increase could be maintained through careful tuning of learning rate and dropout over the incorporation of an additional two blocks, but no further rise could be obtained. Finetuning a total of four blocks resulted in inevitable overfitting and thus a decrease in performance. Incorporating data augmentation alongside class weighting during training enhanced performance and the use of dropout was critical to cope with overfitting.

Moving away from transfer learning to kickstart the learning process, pretraining of a new CNN architecture was explored via an auto-encoder framework. This has as strong advantage that it may exploit the large datasets of unlabeled skin lesions. To this end, two convolutional auto-encoder architectures were proposed (differing in the number of convolutional stages in its encoder, either three or four). The CNN architecture for classification consisted of stacking the encoder with a top model, exploring top models based on a dense layer, an identity block or a convolution block (as incorporated in the ResNet50 model). Supervised training of the top model, with class weighting, indicated an augmented performance of either an identity or convolution block-based top model, rather than a dense output layer. Furthermore, the deeper model proved to still be within the training capacity of the unlabeled dataset and thus enhanced classification performance, as could be expected. Results for the final configuration of a deep pretrained encoder with a trained identity block top model indicated AUC-scores surpassing the performance of SVM's on features extracted from the FC2-layers of the VGG-nets and of the deepest features from the InceptionV3-net. This thus illustrates that pretraining with unlabeled datasets may offer a strong window of opportunity, even though this model could not surpass classification results obtained with the ResNet50 model.

We conclude that in the research line of transfer learning, training an SVM on feature combinations from different pretrained state-of-the-art models (ResNet50 and InceptionV3) leads to an enhanced classification performance compared to stand-alone features. A more apt use of finetuning however proved to be adapting the pretrained model, ResNet50 in this case, to the task at hand and finetuning up to a certain depth in the model. This scheme gave rise to the optimal results achieved in this study. Finally, these results were compared to an auto-encoder framework for pretraining. The best performance results here entered the same ballpark as the lower-scoring SVM classifiers on pretrained features framework, but could not compete with the ResNet50 finetuning scheme. We note however that in light of the trends observed in the auto-encoder framework, a more thorough exploration (of architecture, training time and hyperparameter settings) could potentially improve these results and open a window of opportunity to thoroughly using unlabeled datasets for kickstarting the supervised training phase with limited labeled datasets.

APPENDIX A

APPENDIX A

The entire ResNet-50 architecture [73], as detailed in Table A.1, has two main, residual building blocks. These are referred to as the identity block and the convolutional block with structures, respectively depicted in Figures A.1 and A.2. The residual blocks are constructed from three convolutional layers in the first branch and either an identity mapping or another convolutional layer (+BN) in the second branch. The kernels of the convolutional layers are respectively 1x1, 3x3 and 1x1. The first 1x1 convolution is applied to reduce dimensions, after which a functional 3x3 convolution is applied at this bottleneck and dimensions are recaptured by another 1x1 convolution. .

Table A.1: ResNet50 Architecture (for skin lesion classification)

Input	Skin lesion image: size (224, 224, 3)
Input layers	ZeroPadding2D(padding = (3, 3)) Conv2D (nb. filters = 64, kernel size = (7, 7), padding = ’valid’) BatchNormalization () Activation (’relu’) MaxPooling2D (size = (3, 3), strides = (2, 2)
Residual blocks	1 x conv_block (k = 3, [f1, f2, f3] = [64, 64, 256], s = 1) 2 x identity_block (k = 3, [f1, f2, f3] = [64, 64, 256]) 1 x conv_block (k = 3, [f1, f2, f3] = [128, 128, 512], s = 2) 3 x identity_block (k = 3, [f1, f2, f3] = [128, 128, 512]) 1 x conv_block (k = 3, [f1, f2, f3] = [256, 256, 1024], s = 2) 5 x identity_block (k = 3, [f1, f2, f3] = [256, 256, 1024]) 1 x conv_block (k = 3, [f1, f2, f3] = [512, 512, 2048], s = 2) 2 x identity_block (k = 3, [f1, f2, f3] = [512, 512, 2028])
Output layers	AveragePooling2D (size = (7, 7)) Flatten ()
Output	Dense (size = 3, activation = ’softmax’)

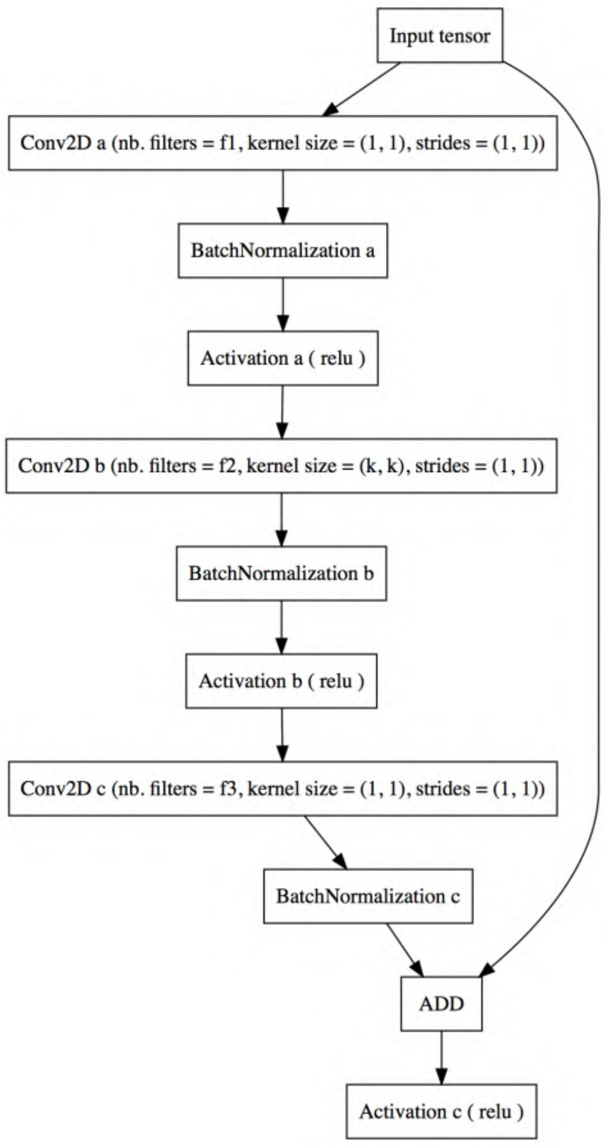
Input tensor
Conv2D a (nb. filters = f1, kernel size = (1, 1), strides = (1, 1))
BatchNormalization a
Activation a (relu)
Conv2D b (nb. filters = f2, kernel size = (k, k), strides = (1, 1))
BatchNormalization b
Activation b (relu)
Conv2D c (nb. filters = f3, kernel size = (1, 1), strides = (1, 1))
BatchNormalization c
ADD
Activation c (relu)

Figure A.1: Identity block - ResNet

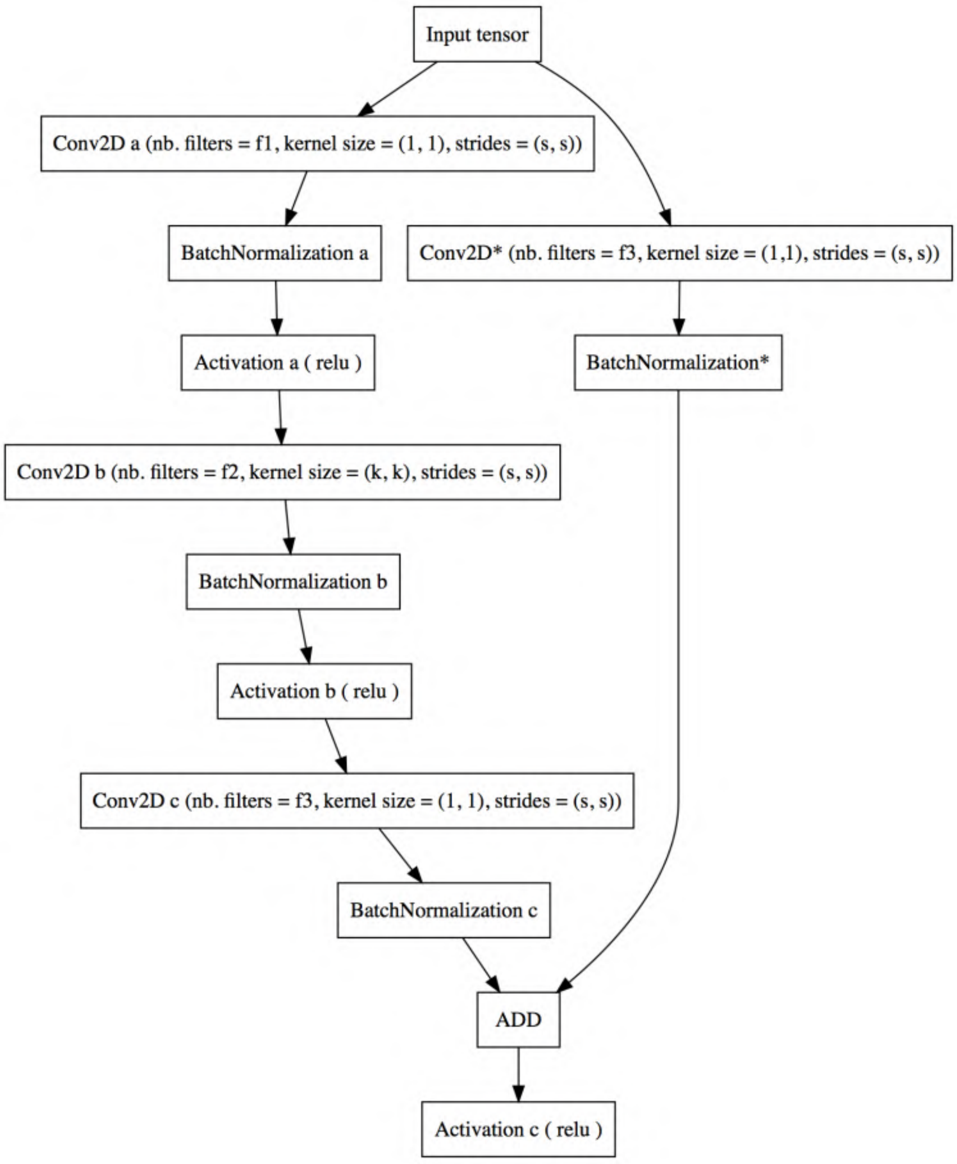
Input tensor
Conv2D a (nb. filters = f1, kernel size = (1, 1), strides = (s, s))
BatchNormalization a
Conv2D* (nb. filters = f3, kernel size = (1,1), strides = (s, s))
Activation a (relu)
BatchNormalization*
Conv2D b (nb. filters = f2, kernel size = (k, k), strides = (s, s))
BatchNormalization b
Activation b (relu)
Conv2D c (nb. filters = f3, kernel size = (1, 1), strides = (s, s))
BatchNormalization c
ADD
Activation c (relu)

Figure A.2: Convolution block - ResNet

APPENDIX B

APPENDIX B

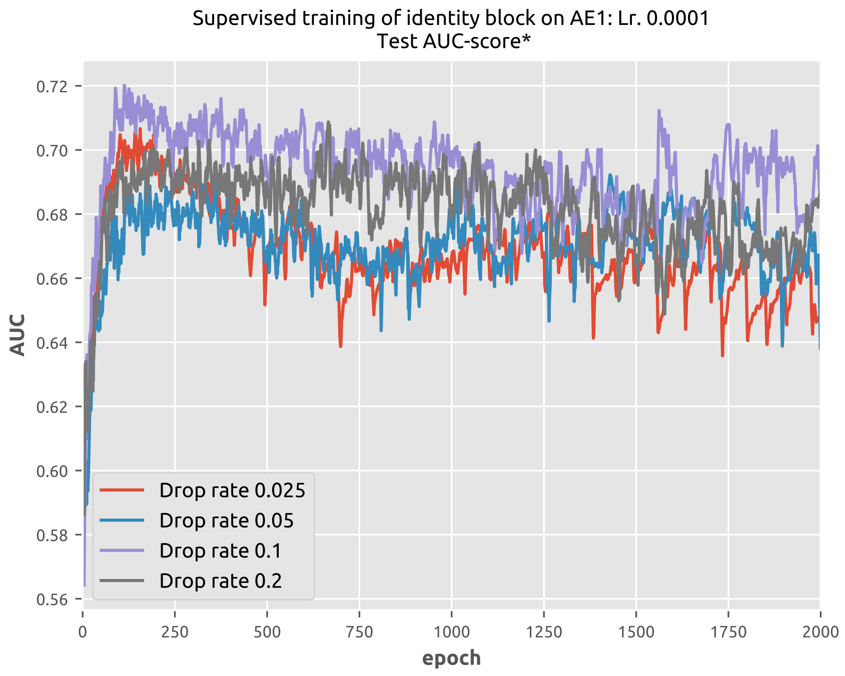

Figure B.1: Supervised training phase of AE_1 and identity block-based top model with a learning rate of 1e-4 for different dropout rates.

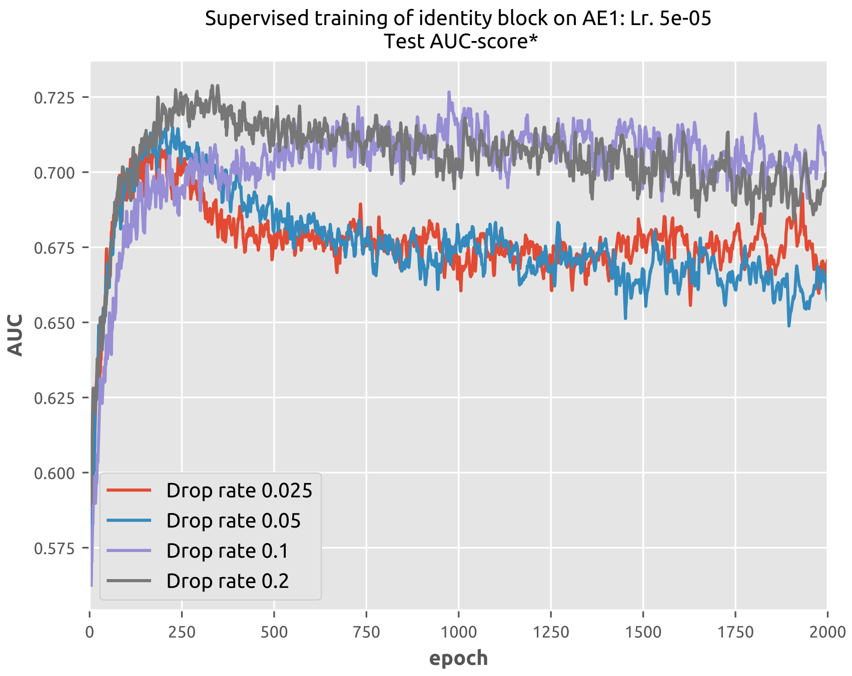

Figure B.2: Supervised training phase of AE_1 and identity block-based top model with a learning rate of 5e-5 for different dropout rates.

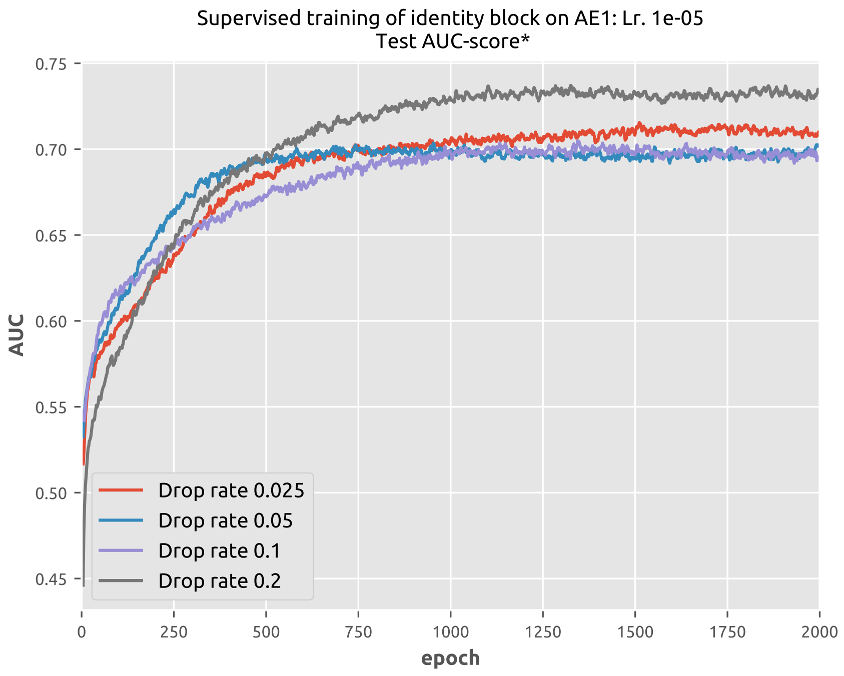

Figure B.3: Supervised training phase of AE_1 and identity block-based top model with a learning rate of 1e-5 for different dropout rates.

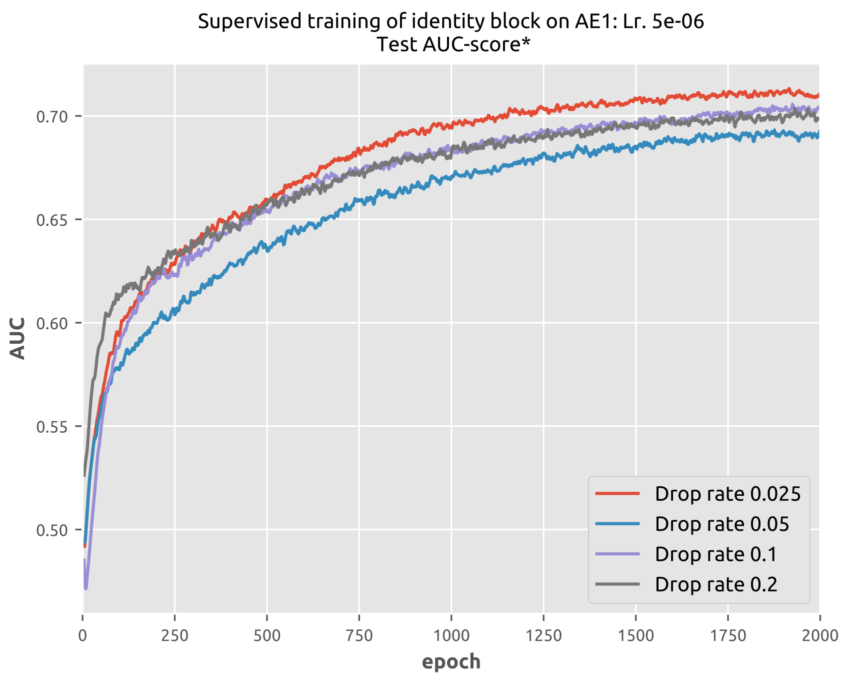

Figure B.4: Supervised training phase of AE_1 and identity block-based top model with a learning rate of 5e-6 for different dropout rates.

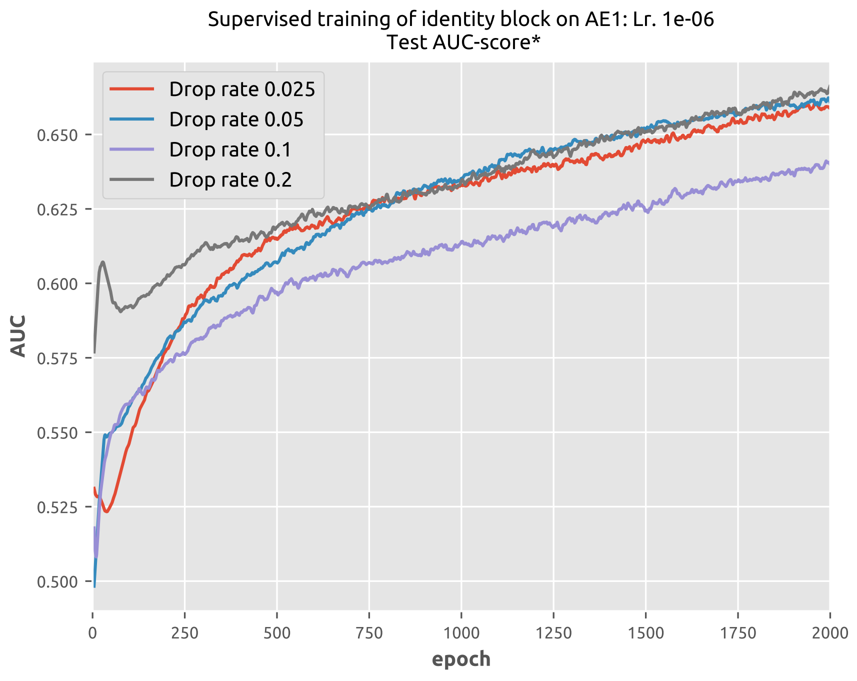

Figure B.5: Supervised training phase of AE_1 and identity block-based top model with a learning rate of 1e-6 for different dropout rates.

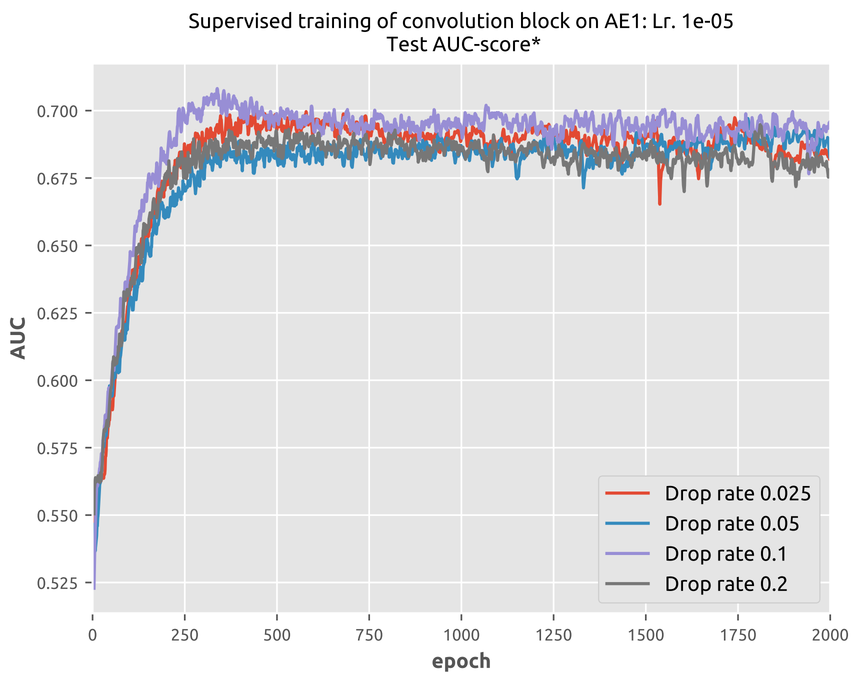

Figure B.6: Supervised training phase of AE_1 and convolution block-based top model with a learning rate of 1e-5 for different dropout rates.

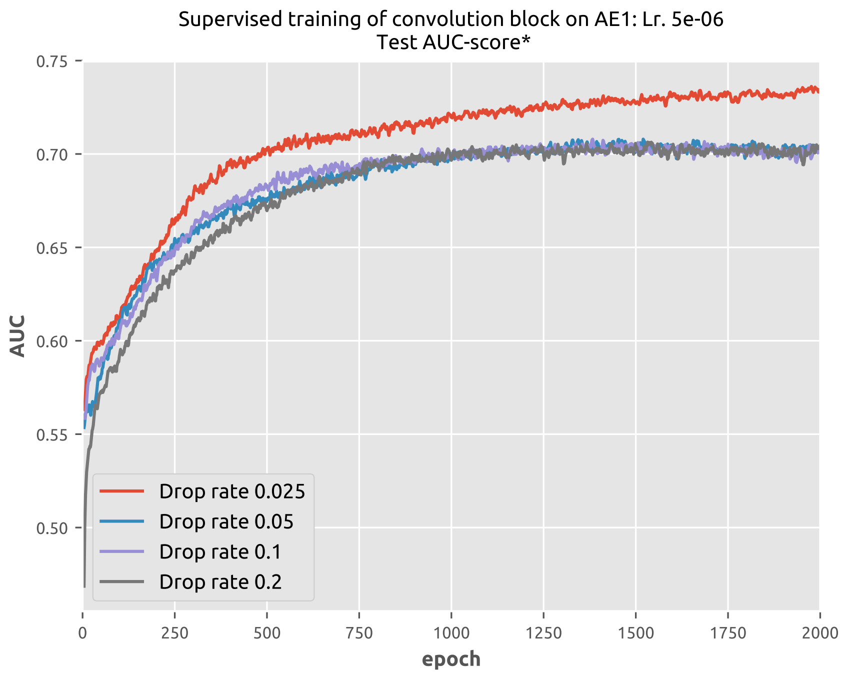

Figure B.7: Supervised training phase of AE_1 and convolution block-based top model with a learning rate of 5e-6 for different dropout rates.

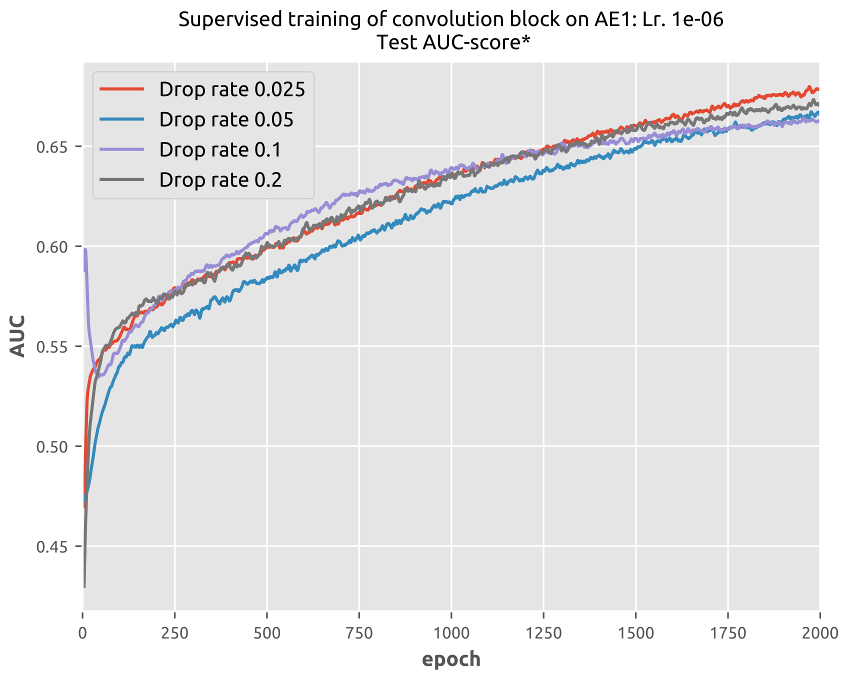

Figure B.8: Supervised training phase of AE_1 and convolution block-based top model with a learning rate of 1e-6 for different dropout rates.

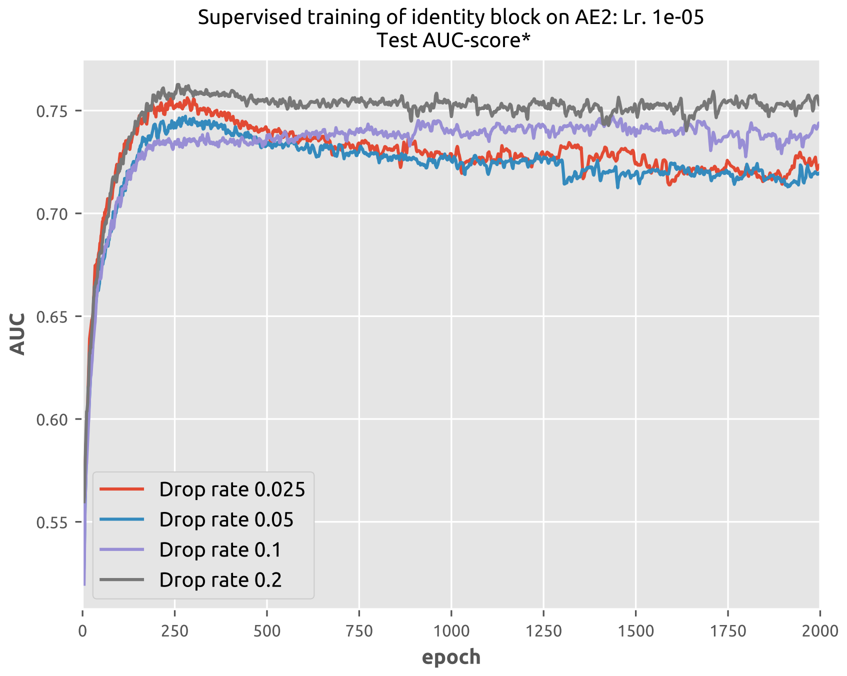

Figure B.9: Supervised training phase of AE_2 and identity block-based top model with a learning rate of 1e-5 for different dropout rates.

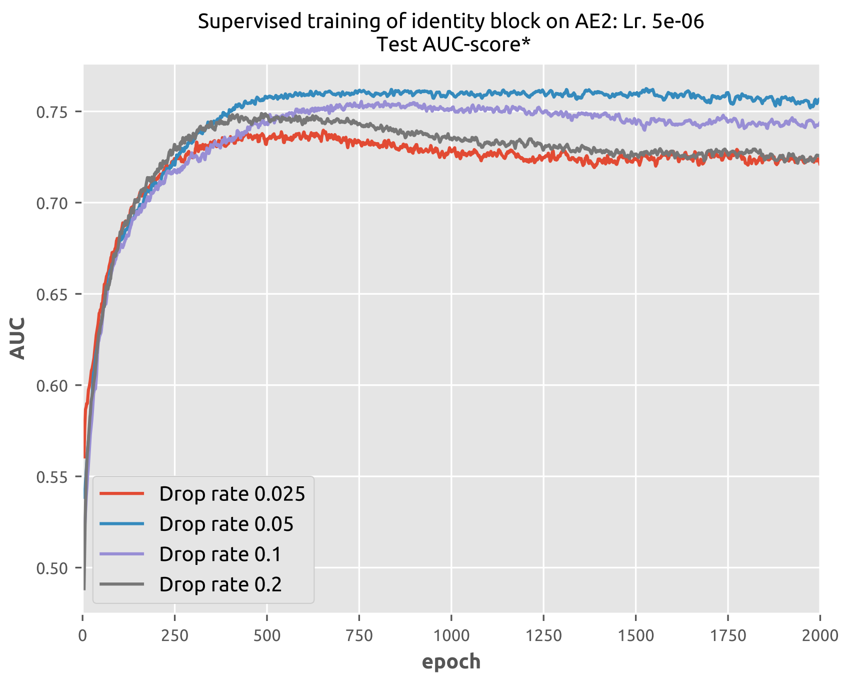

Figure B.10: Supervised training phase of AE_2 and identity block-based top model with a learning rate of 5e-6 for different dropout rates.

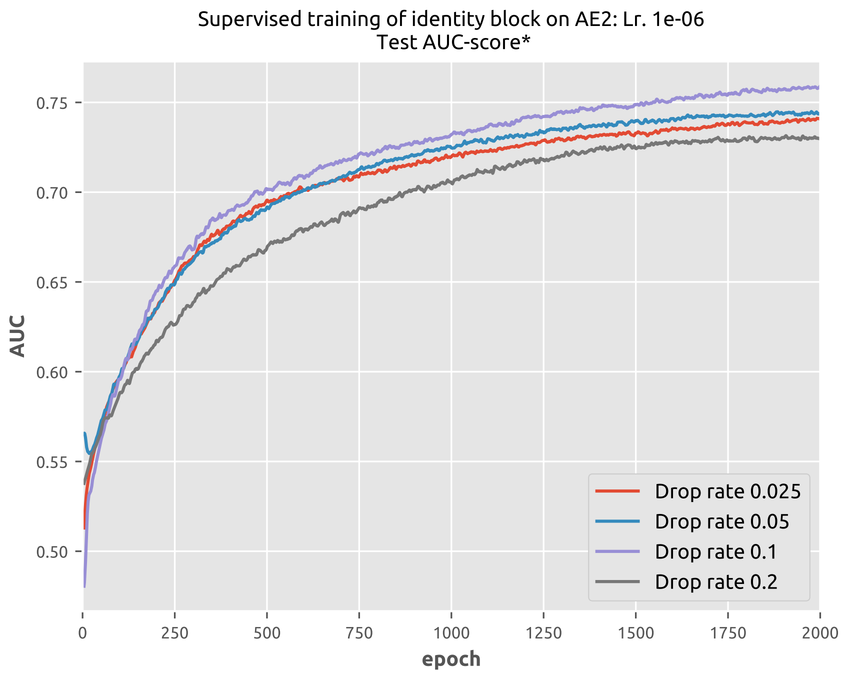

Figure B.11: Supervised training phase of AE_2 and identity block-based top model with a learning rate of 1e-6 for different dropout rates.

Bibliography

[1] Committee on Diagnostic Error in Health Care, Board on Health Care Services, Institute of Medicine, The National Academies of Sciences, Engineering and Medicine, *Improving Diagnosis in Health Care*.
National Academies Press (US), 2015.

[2] P. H. Meyers, J. Charles M. Nice, H. C. Becker, J. Wilson J. Nettleton, J. W. Sweeney, and G. R. Meckstroth, "Automated computer analysis of radiographic images," *Radiology*, vol. 83, no. 6, pp. 1029–1034, 1964.
PMID: 14226800.

[3] R. P. Kruger, J. R. Townes, D. L. Hall, S. J. Dwyer, and G. S. Lodwick, "Automated radiographic diagnosis via feature extraction and classification of cardiac size and shape descriptors," *IEEE Transactions on Biomedical Engineering*, vol. BME-19, no. 3, pp. 174–186, 1972.

[4] J.-I. Toriwaki, Y. Suenaga, T. Negoro, and T. Fukumura, "Pattern recognition of chest x-ray images," *Computer Graphics and Image Processing*, vol. 2, no. 3, pp. 252 – 271, 1973.

[5] K. Doi, "Computer-aided diagnosis in medical imaging: Historical review, current status and future potential," *Computerized medical imaging and graphics : the official journal of the Computerized Medical Imaging Society*, vol. 31, no. 4-5, pp. 198–211, 2007.

[6] EuroMelanoma, "België heeft een huidkanker probleem - Persbericht EuroMelanoma," 2018.
`https://www.euromelanoma.org/belgie`.

[7] K. Korotkov and R. Garcia, "Computerized analysis of pigmented skin lesions: A review," *Artificial Intelligence in Medicine*, vol. 56, no. 2, pp. 69 – 90, 2012.

[8] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[9] P. Kashyap, *Industrial Applications of Machine Learning*, pp. 189–233.
Berkeley, CA: Apress, 2017.

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

[11] T. Mitchell, *Machine Learning*.
New York: McGraw-Hill, 1997.

[12] Y. LeCun, Y. Bengio, G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[13] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*.
Pearson Education, 2 ed., 2003.

[14] R. Raina, A. Battle, H. Lee, B. Packer, A. Y. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, (New York, NY, USA), pp. 759–766, ACM, 2007.

[15] P. Refaeilzadeh, L. Tang, and H. Liu, "Encyclopedia of database systems," pp. 1–7, New York, NY: Springer New York, 2016.

[16] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, pp. 78–87, Oct. 2012.

[17] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*.
Berlin, Heidelberg: Springer-Verlag, 2006.

[18] V. Vapnik, S. E. Golowich, and A. J. Smola, "Support vector method for function approximation, regression estimation and signal processing," in *Advances in Neural Information Processing Systems 9* (M. C. Mozer, M. I. Jordan, and T. Petsche, eds.), pp. 281–287, MIT Press, 1997.

[19] D. Basak, S. Pal, and D. Chandra Patranabis, "Support vector regression," in *Neural Information Processing: Letters and Reviews*, vol. 11, 11 2007.

[20] V. N. Vapnik, "An overview of statistical learning theory," *Trans. Neur. Netw.*, vol. 10, no. 5, pp. 988–999, 1999.

[21] V. N. Vapnik, *The Nature of Statistical Learning Theory*.
Berlin, Heidelberg: Springer-Verlag, 1995.

[22] K. P. Bennett and C. Campbell, "Support vector machines: Hype or hallelujah?," *SIGKDD Explor. Newsl.*, vol. 2, pp. 1–13, Dec. 2000.

[23] M. Hoffman, "Support vector machines — kernels and the kernel trick," 2006.
`http://www.cogsys.wiai.uni-bamberg.de/teaching/ss06/hs_svm/slides/SVM_Seminarbericht_Hofmann.pdf`.

[24] S. R. Gunn, "Support vector machines for classification and regression," 1998.

[25] C. Campbell, "Radial basis function networks 1," ch. An Introduction to Kernel Methods, pp. 155–192, Vienna, Austria, Austria: Physica Verlag Rudolf Liebing KG, 2001.

[26] J. Milgram, M. Cheriet, and R. Sabourin, "“One Against One” or “One Against All”: Which One is Better for Handwriting Recognition with SVMs?," Oct. 2006.
http://www.suvisoft.com.

[27] Blausen Medical Communications, Inc.
Accessed from Wikimedia Commons; `https://commons.wikimedia.org/wiki/File:Neuron_Part_1.png`.

[28] Blausen Medical Communications, Inc.
Accessed from Wikimedia Commons; `https://commons.wikimedia.org/wiki/File:Synapse_Presynaptic_Neuron.png`.

[29] *Human anatomy and Physiology*.
Boston: Pearson, 2012.

[30] W. Gerstner, A. K. Kreiter, H. Markram, and A. V. M. Herz, "Neural codes: Firing rates and beyond," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 94, pp. 12740–12741, 11 1997.

[31] D. Purves, G.J. Augustine, D. Fitzpatrick et al., *Neuroscience - 2nd edition*, ch. Summation of Synaptic Potentials.
Sunderland (MA): Sinauer Associates, 2001.

[32] F. Irie ,Y. Yamaguchi, "Encyclopedia of neuroscience," ch. Eph Receptor Signaling and Spine Morphology, Elsevier, 2009.

[33] S. Haykin, *Neural Networks: A Comprehensive Foundation - 2nd edition*.
Pearson Education, 1999.

[34] M. Hiroyoshi, "Encyclopedia of neuroscience," pp. 3952–3956, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[35] "Fundamental neuroscience," ClinicalKey 2012, ch. Information Processing in Dendrites and Spine, Elsevier/Academic Press, 2013.

[36] D. Kriesel, *A Brief Introduction to Neural Networks*.
2007.

[37] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[38] D. Hebb, *The Organization of Behavior. A Neuropsychological Theory*.
NewYork, NY: Wiley, 1949.

[39] F. Rosenblatt, *Principles of neurodynamics: perceptrons and the theory of brain mechanisms*. Report (Cornell Aeronautical Laboratory), Spartan Books, 1962.

[40] C. Van Der Malsburg, "Frank rosenblatt: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms," in *Brain Theory* (G. Palm and A. Aertsen, eds.), (Berlin, Heidelberg), pp. 245–248, Springer Berlin Heidelberg, 1986.

[41] M. Minsky, S. Papert, *Perceptrons*.
Oxford, England: M.I.T. Press, 1969.

[42] L. N. Kanal and P. B. Baltes, "International encyclopedia of the social & behavioral sciences," pp. 11218–11221, Oxford: Pergamon, 2001.

[43] A. Krizhevsky, I. Sutskever, G. Hinton, "Imagenet classification with deep convolutional neural networks," *Proc. Advances in Neural Information Processing Systems*, vol. 25, pp. 1090–1098, 2012.

[44] L. Bottou, *Stochastic Gradient Descent Tricks*, pp. 421–436.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[46] Y. Le Cun, L. Bottou, Y. Bengio, P. Haffner, "Gradient based learning applied to document recognition," *Proceedings of IEEE*, vol. 86, pp. 2278–2324, 1998.

[47] Illustration from Anatomy and Physiology, Connexions Web site.
Accessed from Wikimedia Commons; `https://commons.wikimedia.org/wiki/File:1424_Visual_Streams.jpg`.

[48] F.G. NewCombe, "Missile wounds of the brain: a study of pschological deficits," 1969.

[49] E.H. De Haan, A. Cowey, "On the usefulness of 'what' and 'where' pathways in vision," *Trends in Cognitive Sciences*, pp. 460–466, October 2011.

[50] L.G. Ungerleider, M. Mischkin ch. Two cortical visual systems, Cambridge: MIT Press, 1982.

[51] R.L. Mapou, J. Spector, "Clinical neuropsychological assessment: A cognitive approach," ch. Assessment of visuocognitive processes, pp. 138–139, Springer Science & Business Media, 1995.

[52] N.V.K. Medathati, H. Neumann, G.S. Masson, P. Kornprobst, "Bio-inspired computer vision: Towards a synergistic approach of artificial and biological vision," *Computer Vision and Image Understanding*, April 2016.

[53] K. Fukushima, “Artificial vision by multi-layered neural networks: neocognitron and its advances,” *Neural Networks*, vol. 37, pp. 103–119, 2013.

[54] M.A. Goodale, A.D. Milner, “Separate visual pathways for perception and action.,” *Trends in Neuroscience*, pp. 20–25, January 1992.

[55] D.J. Kravitz, K.S. Saleem, C.I. Baker, L.G. Ungerleider, M. Mishkin, “The ventral visual pathway: an expanded neural framework for the processing of object quality,” *Trends in Cognitive Sciences*, pp. 26–49, January 2013.

[56] D.H. Hubel, T.N. Wiesel, “Receptive fields of single neurones in the cat’s striate cortex,” *The Journal of Physiology*, vol. 148, pp. 574–591, 10 1959.

[57] D.H.Hubel, T.N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” *The Journal of Physiology*, vol. 160, pp. 106–154.2, 01 1962.

[58] R.H. Wurtz, “Recounting the impact of hubel and wiesel,” *The Journal of Physiology*, vol. 587(Pt 12), p. 2817–2823, 2009.

[59] D.L. Ringach, “Mapping receptive fields in primary visual cortex,” *The Journal of Physiology*, vol. 558(Pt 3), p. 717–728, 2004.

[60] A. Afraz, D.L. Yamins, J.J. DiCarlo, “Neural mechanisms underlying visual object recognition,” *Cold Spring Harb Symp Quant Biol*, vol. 79, pp. 99–107, 2014.

[61] L. Cloutman, “Interaction between dorsal and ventral processing streams: Where, when and how?,” *Brain and Language*, vol. 127, November 2013.

[62] S. Budisavljevica, F. Dell’Acqua, U. Castiello, “Cross-talk connections underlying dorsal and ventral stream integration during hand actions,” *Cortex*, vol. 103, pp. 224–239, 2018.

[63] S. Hochstein, M. Ahissar, “View from the top: hierarchies and reverse hierarchies in the visual system,” *Neuron*, vol. 36, pp. 791–804, December 2002.

[64] J.J. DiCarlo, D. Zoccolan, N.C. Rust, “How does the brain solve visual object recognition?,” *Neuron*, vol. 73, p. 415–434, 2012.

[65] N.C. Rust, O. Schwartz, J.A. Movshon, E.P. Simoncelli, “Spatiotemporal elements of macaque v1 receptive fields,” *Neuron*, vol. 46, pp. 945–956, June 2005.

[66] D.D. Cox, T. Dean, “Neural networks and neuroscience-inspired computer vision,” *Current Biology*, vol. 24, pp. 921–929.

[67] K. Fukushima, “Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.

[68] OMPE.
Accessed from `http://www.ompe.org/en/theme/save-the-lion/`.

[69] Accessed from `http://alexlenail.me/NN-SVG/LeNet.html`.

[70] Scherer D., Muller A., Behnke S., ch. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition.
2010.

[71] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[72] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[73] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[74] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[75] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015.

[76] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016.

[77] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *CoRR*, vol. abs/1411.1792, 2014.

[78] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[79] L. Torrey and J. Shavlik, *Handbook of Research on Machine Learning Applications*, ch. Transfer Learning.
IGI Global, 2009.

[80] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," *CoRR*, vol. abs/1403.6382, 2014.

[81] L. Zheng, Y. Zhao, S. Wang, J. Wang, and Q. Tian, "Good practice in CNN feature transfer," *CoRR*, vol. abs/1604.00133, 2016.

[82] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1,” ch. Learning Internal Representations by Error Propagation, pp. 318–362, Cambridge, MA, USA: MIT Press, 1986.

[83] G. E. Hinton and J. L. McClelland, “Learning representations by recirculation,” in *Neural Information Processing Systems* (D. Z. Anderson, ed.), pp. 358–366, American Institute of Physics, 1988.

[84] M. A. Ponti, L. S. F. Ribeiro, T. S. Nazare, T. Bui, and J. Collomosse, “Everything you wanted to know about deep learning for computer vision but were afraid to ask,” in *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*, pp. 17–41, 2017.

[85] Y. Bengio, A. C. Courville, and P. Vincent, “Unsupervised feature learning and deep learning: A review and new perspectives,” *CoRR*, vol. abs/1206.5538, 2012.

[86] P. Baldi, “Autoencoders, unsupervised learning, and deep architectures,” in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning* (I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver, eds.), vol. 27 of *Proceedings of Machine Learning Research*, (Bellevue, Washington, USA), pp. 37–49, PMLR, 02 Jul 2012.

[87] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, “Why does unsupervised pre-training help deep learning?,” *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, Mar. 2010.

[88] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, “Stacked convolutional auto-encoders for hierarchical feature extraction,” in *Artificial Neural Networks and Machine Learning –ICANN 2011* (T. Honkela, W. Duch, M. Girolami, and S. Kaski, eds.), (Berlin, Heidelberg), pp. 52–59, Springer Berlin Heidelberg, 2011.

[89] R. K. Freinkel and D. Woodley, *Biology of the Skin*.
CRC Press, March 2001.

[90] R. Lanza, J. Gearhart, B. Hogan, D. Melton, R. Pedersen, E. D. Thomas, J. Thomson, and I. Wilmut, eds., *Essentials of Stem Cell Biology (Second Edition)*.
San Diego: Academic Press, second edition ed., 2009.

[91] C. Blainpain and E.Fuchs, “Epidermal homeostasis: a balancing act of stem cells in the skin,” *Nature Reviews - Molecular Cell Biology*.

[92] National Cancer Institute.
Accessed from Wikimedia Commons; `https://commons.wikimedia.org/wiki/File:Layers_of_the_skin.jpg`.

[93] Connexions Website, “Anatomy and physiology.” http://cnx.org/content/col11496/1.6/, 2013.

[94] S. Jaitley and T. Saraswathi, “Pathophysiology of langerhans cells,” *Journal of Oral and Maxillofacial Pathology*, vol. 16, pp. 239–244, 2012.

[95] N. Boulais and L. Misery, “Merkel cells,” *Journal of the American Academy of Dermatology*, vol. 57, no. 1, pp. 147 – 165, 2007.

[96] M. Cichorek, M. Wachulska, A. Stasiewicz and A. Tymińska, “Skin melanocytes: biology and development,” *Allergology/Postepy Dermatologii I Alergologii*, vol. 30, pp. 30–41, 2013.

[97] M. Paul and R. Jennifer, “Radiative relaxation quantum yields for synthetic eumelanin,” *Photochemistry and Photobiology*, vol. 79, no. 2, pp. 211–216.

[98] B. Michaela and H. V. J., “The protective role of melanin against uv damage in human skin,” *Photochemistry and Photobiology*, vol. 84, no. 3, pp. 539–549.

[99] J. Y. Lin and D. E. Fisher, “Melanocyte biology and skin pigmentation,” *Nature*, vol. 445, p. 843–850, February 2007.

[100] T. Hirobe, “Keratinocytes regulate the function of melanocytes,” *Dermatologica Sinica*, vol. 32, no. 4, pp. 200 – 204, 2014.
Special Issue: Pigmentary Disorders-Bringing Colors to Our Specialty.

[101] G. Fabbrocini, M. Triassi, M.C. Mauriello, G. Torre, M.C. Annunziata, V.D. Vita, et al., “Epidemiology of skin cancer: Role of some environmental factors,” *Cancers*, vol. 2, pp. 1980–1989, 2010.

[102] D.L. Narayanan and R.N. Saladi and J.L. Fox Joshua, “Review: Ultraviolet radiation and skin cancer,” *International Journal of Dermatology*, vol. 49, no. 9, pp. 978–986.

[103] Y. Matsumura and H. N. Ananthaswamy, “Toxic effects of ultraviolet radiation on the skin,” *Toxicology and Applied Pharmacology*, vol. 195, no. 3, pp. 298 – 308, 2004.
Toxicology of the Skin.

[104] R. Gordon, “Skin cancer: An overview of epidemiology and risk factors,” *Seminars in Oncology Nursing*, vol. 29, no. 3, pp. 160 – 169, 2013.
Skin Cancer.

[105] American Cancer Society, “Facts and Figures 2017,” 2017.
https://www.cancer.org/research/cancer-facts-statistics/all-cancer-facts-figures/cancer-facts-figures-2017.html.

[106] M.R. Albert and M.A. Weinstock, “Keratinocyte carcinoma,” *CA: A Cancer Journal for Clinicians*, vol. 53, no. 5, pp. 292–302.

[107] V. Madan, J. T. Lear, and R.-M. Szeimies, “Non-melanoma skin cancer,” *The Lancet*, vol. 375, no. 9715, pp. 673 – 685, 2010.

[108] G.-M. Giuseppina and S. Alain, “Tp53 mutations in human skin cancers,” *Human Mutation*, vol. 21, no. 3, pp. 217–228.

[109] “Cancer burden in Belgium 2004-2013,” 2015.

[110] “Cancer Incidence Projections in Belgium, 2015 to 2025,” 2017.

[111] V. Gray-Schopfer, C. Wellbrock, and R. Marais, “Melanoma biology and new target therapy,” *Nature*, vol. 445, pp. 851–7, 03 2007.

[112] W. H. Clark, D. E. Elder, D. Guerry, M. N. Epstein, M. H. Greene, and M. V. Horn, “A study of tumor progression: The precursor lesions of superficial spreading and nodular melanoma,” *Human Pathology*, vol. 15, no. 12, pp. 1147 – 1165, 1984.

[113] W.E. Damsky, N. Theodosakis, M. Bosenberg, “Melanoma metastasis: new concepts and evolving paradigms,” *Oncogene*, vol. 33, pp. 2413 EP –, 06 2013.

[114] American Cancer Society, “Facts and Figures 2018.” `https://www.cancer.org/research/cancer-facts-statistics/all-cancer-facts-figures/cancer-facts-figures-2018.html`.

[115] T. Mudigonda, D.J. Pearce, B.A. Yentzer, P. Williford, S.R. Feldman, “The economic impact of non-melanoma skin cancer: a review,” *J Natl Compr Canc Netw.*, vol. 8, pp. 888–896, 2010.

[116] Z. Apalla and A. Lallas, E. Sotiriou, E.Lazaridou, D. Ioannides, “Epidemiological trends in skin cancer,” *Dermatology Practical & Conceptual*, vol. 7, pp. 1–6, 04 2017.

[117] K. Linden, “Screening and early detection of skin cancer,” *Current Oncology Reports*, vol. 6, no. 6, pp. 491–496, 2004.

[118] R.J. Friedman and D.S. Rigel and A.W. Kopf, “Early detection of malignant melanoma: The role of physician examination and self-examination of the skin,” *CA: A Cancer Journal for Clinicians*, vol. 35, no. 3, pp. 130–151.

[119] H. Tsao, J. M. Olazagasti, K. M. Cordoro, J. D. Brewer, S. C. Taylor, J. S. Bordeaux, M.-M. Chren, A. J. Sober, C. Tegeler, R. Bhushan, and W. S. Begolka, “Early detection of melanoma: Reviewing the abcdes,” *Journal of the American Academy of Dermatology*, vol. 72, no. 4, pp. 717 – 723, 2015.

[120] "Skin Cancer at Arta Dermatology, Newport Beach." `http://www.artaderm.com/skin-cancer-newport-beach/`.

[121] J.L. Messina and V.K. Sondak, "Prognostic factors in localized cutaneous melanoma with particular reference to thin primary lesions," *G Ital Dermatol Venereol*, vol. 142, pp. 123–129, 2007.

[122] G. Argenziano and H. P. Soyer, "Dermoscopy of pigmented skin lesions &#x2013; a valuable tool for early," *The Lancet Oncology*, vol. 2, pp. 443–449, 2018/05/24 2001.

[123] J. Henning, J.A. Stein, J. Yeung et al., "Cash algorithm for dermoscopy revisited," *Archives of Dermatology*, vol. 144, no. 4, pp. 554–555, 2008.

[124] L. Thomas and S. Puig, "Dermoscopy, digital dermoscopy and other diagnostic tools in the early detection of melanoma and follow-up of high-risk skin cancer patients," *Acta Derm Venereol*, July 2017.

[125] B. Nirmal, "Dermatoscopy: Physics and principles," *Indian Journal of Dermatopathology and Diagnostic Dermatology*, vol. 4, pp. 27–30, 7 2017.

[126] S. Bajaj, M. A. Marchetti, C. Navarrete-Dechent, S. W. Dusza, K. Kose, and A. A. Marghoob, "The role of color and morphologic characteristics in dermoscopic diagnosis," *JAMA dermatology*, vol. 152, pp. 676–682, 06 2016.

[127] C. Benelli, E. Roscetti, V. D. Pozzo, G. Gasparini, and S. Cavicchini, "The dermoscopic versus the clinical diagnosis of melanoma.," *European journal of dermatology : EJD*, vol. 9 6, pp. 470–6, 1999.

[128] D.S. Rigel, J. Russak, R. Friedman, "The evolution of melanoma diagnosis: 25 years beyond the abcds," *CA: A Cancer Journal for Clinicians*, vol. 60, pp. 301–316, 2018/06/17 2010.

[129] "ISIC Project." `https://isdis.net/isic-project`, 2018.

[130] D. Gutman, N. C. F. Codella, M. E. Celebi, B. Helba, M. A. Marchetti, N. K. Mishra, and A. Halpern, "Skin lesion analysis toward melanoma detection: A challenge at the international symposium on biomedical imaging (ISBI) 2016, hosted by the international skin imaging collaboration (ISIC)," *CoRR*, vol. abs/1605.01397, 2016.

[131] M. Arasi, E.-S. El-Dahshan, E.-S. M El-Horbaty, and A.-B. M. Salem, *Malignant Melanoma Detection Based on Machine Learning Techniques: A Survey*.
09 2016.

[132] A. N. Hoshyar, A. Al-Jumaily, and A. N. Hoshyar, "The beneficial techniques in preprocessing step of skin cancer detection system comparing," *Procedia Computer Science*, vol. 42, pp. 25 – 31, 2014.

Medical and Rehabilitation Robotics and Instrumentation (MRRI2013).

[133] R. B. Oliveira, M. E. Filho, Z. Ma, J. P. Papa, A. S. Pereira, and J. M. R. Tavares, "Computational methods for the image segmentation of pigmented skin lesions: A review," *Computer Methods and Programs in Biomedicine*, vol. 131, pp. 127 – 141, 2016.

[134] P. Mehta and B. Shah, "Review on techniques and steps of computer aided skin cancer diagnosis," *Procedia Computer Science*, vol. 85, pp. 309–316, 2016.

[135] S. Dreiseitl, L. Ohno-Machado, H. Kittler, S. Vinterbo, H. Billhardt, and M. Binder, "A comparison of machine learning methods for the diagnosis of pigmented skin lesions," *Journal of Biomedical Informatics*, vol. 34, no. 1, pp. 28–36, 2001.

[136] N. Codella, J. Cai, M. Abedini, R. Garnavi, A. Halpern, and J. R. Smith, "Deep learning, sparse coding, and svm for melanoma recognition in dermoscopy images," in *Machine Learning in Medical Imaging* (L. Zhou, L. Wang, Q. Wang, and Y. Shi, eds.), (Cham), pp. 118–126, Springer International Publishing, 2015.

[137] N. C. F. Codella, Q. Nguyen, S. Pankanti, D. Gutman, B. Helba, A. Halpern, and J. R. Smith, "Deep learning ensembles for melanoma recognition in dermoscopy images," *CoRR*, vol. abs/1610.04662, 2016.

[138] A. Mahbod, R. Ecker, and I. Ellinger, "Skin lesion classification using hybrid deep neural networks," *CoRR*, vol. abs/1702.08434, 2017.

[139] J. Kawahara, A. BenTaieb, and G. Hamarneh, "Deep features to classify skin lesions," *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pp. 1397–1400, 2016.

[140] C. N. Vasconcelos and B. N. Vasconcelos, "Increasing deep learning melanoma classification by classical and expert knowledge based image transforms," *CoRR*, vol. abs/1702.07025, 2017.

[141] A. Menegola, M. Fornaciali, R. Pires, F. V. Bittencourt, S. E. F. de Avila, and E. Valle, "Knowledge transfer for melanoma screening with deep learning," *CoRR*, vol. abs/1703.07479, 2017.

[142] A. Menegola, J. Tavares, M. Fornaciali, L. T. Li, S. E. F. de Avila, and E. Valle, "RECOD titans at ISIC challenge 2017," *CoRR*, vol. abs/1703.04819, 2017.

[143] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, pp. 115 EP –, 01 2017.

[144] A. Creswell, A. Pouplin, and A. A. Bharath, "Denoising adversarial autoencoders: Classifying skin lesions using limited labelled training data," *CoRR*, vol. abs/1801.00693, 2018.

[145] M. A. Arasi, E. S. M. El-Horbaty, A. B. M. Salem, and E. S. A. El-Dahshan, "Stack auto-encoders approach for malignant melanoma diagnosis in dermoscopy images," in *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, pp. 403–409, 2017.

[146] P. Kharazmi, J. Zheng, H. Lui, Z. Jane Wang, and T. K. Lee, "A computer-aided decision support system for detection and localization of cutaneous vasculature in dermoscopy images via deep feature learning," *Journal of Medical Systems*, vol. 42, no. 2, p. 33, 2018.

[147] "ISIC Archive." `https://isic-archive.com/`.

[148] F. Chollet *et al.*, "Keras." `https://github.com/fchollet/keras`, 2015.

[149] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015.
Software available from tensorflow.org.

[150] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.

[151] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[152] B. Ozaydin, J. M. Hardin, and D. C. Chhieng, *Data Mining and Clinical Decision Support Systems*, pp. 45–68.
Cham: Springer International Publishing, 2016.

[153] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, and G. D. Tourassi, "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance," *Neural Networks*, vol. 21, no. 2, pp. 427–436, 2008.

[154] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.