xAILAB
EXPLAINABLE MACHINE LEARNING

Benchmarking selected State-of-the-Art Baseline Neural Networks for 2D Biomedical Image Classification, Inspired by the MedMNIST v2 Framework

Bachelor Thesis

Bachelor of Science in Computer Science

Julius Brockmann

October 2, 2023

Supervisor:

1st: Prof. Dr. Christian Ledig
2nd: Sebastian Dörrich, M.Sc.

Chair of Explainable Machine Learning
Faculty of Information Systems and Applied Computer Sciences
Otto-Friedrich-University Bamberg

Abstract

Deep learning in medical imaging has experienced rapid growth. This work introduces benchmarking practices in the domain of biomedical image classification. It highlights the importance of assessing the effectiveness of deep learning models on standardized, high-quality medical databases. This practice, known as benchmarking, enables researchers to compare various algorithms and architectures, to identify strengths and limitations, and make informed decisions for specific tasks. Pre-trained baseline neural networks are utilized as a foundation for benchmarking and contrasts different training setups, shedding light on factors influencing deep learning model performance.

The work's contribution lies in conducting benchmark experiments on four convolutional neural networks and one transformer-based architecture for biomedical image classification. It explores the impact of resolution scaling on performance and provides access to the methodology for creating higher resolution datasets. Additionally, it briefly examines common misclassifications made during the experiments. A summary of key findings, highlighting the improved performance of newer convolutional neural network architectures and confirming the suitability of transformers for larger sample sizes, concludes this work.

Scripts for dataset creation and experiment replication are publicly available.

Abstract

Deep Learning in der medizinischen Bildgebung hat ein schnelles Wachstum erfahren. In dieser Arbeit werden Benchmarking-Verfahren im Bereich der biomedizinischen Bildklassifizierung vorgestellt. Es wird hervorgehoben, wie wichtig es ist, die Effektivität von Deep-Learning-Modellen auf standardisierten, hochwertigen medizinischen Datenbanken zu bewerten. Diese als Benchmarking bezeichnete Praxis ermöglicht es Forschern, verschiedene Algorithmen und Architekturen zu vergleichen, Stärken und Grenzen zu erkennen und fundierte Entscheidungen für bestimmte Aufgaben zu treffen. Vortrainierte neuronale Basis-Netzwerke werden als Grundlage für das Benchmarking verwendet und kontrastieren verschiedene Trainingsverfahren, um die Faktoren zu beleuchten, die die Leistung von Deep-Learning-Modellen beeinflussen.

Der Beitrag dieser Arbeit liegt in der Durchführung von Benchmark-Experimenten mit vier Convolutional Neural Networks und einem Transformer für die biomedizinische Bildklassifizierung. Sie untersucht die Auswirkungen der Auflösung auf die Perfomanz und bietet Zugang zur Methodik für die Erstellung von Datensätzen mit höherer Auflösung. Außerdem wird kurz auf häufige Fehlklassifizierungen während der Experimente eingegangen. Eine Zusammenfassung der wichtigsten Ergebnisse, die die verbesserte Leistung neuerer Convolutional Neural Networks hervorhebt und die Eignung von Transformern für größere Stichprobengrößen bestätigt, schließt diese Arbeit ab.

Skripte für die Erstellung von Datensätzen und die Replikation von Experimenten sind öffentlich zugänglich.

Acknowledgements

I extend my sincere gratitude to Prof. Dr. Christian Ledig for providing me with the opportunity to undertake my bachelor thesis at his new chair at the university. His seminar, which encouraged lively discussions, was a significant source of motivation for me. I am deeply indebted to Sebastian Dörrich, my dedicated supervisor, whose invaluable guidance ensured the successful completion of this work. Lastly, I would like to express my heartfelt appreciation to my mother and uncle for their unwavering support and regular check-ins throughout the creation of this work.

Contents

List of Figures vi

List of Tables ix

1 Introduction 1

1.1 Related Work . . . 1

1.2 Contribution . . . 1

2 Methods 2

2.1 Learning of Neural Networks . . . 2

2.2 Neural Network Functions . . . 5

2.2.1 . . . 6

2.2.2 Convolution . . . 6

2.2.3 Pooling . . . 8

2.2.4 Activation . . . 9

2.2.5 Attention . . . 10

2.2.6 Normalization . . . 11

2.2.7 Linear . . . 13

2.2.8 Regularization . . . 13

2.2.9 Loss . . . 14

2.2.10 Performance measures . . . 15

2.3 Neural Network Architectures . . . 16

2.3.1 ResNets . . . 17

2.3.2 DenseNet . . . 19

2.3.3 EfficientNet . . . 21

2.3.4 Vision Transformer . . . 23

3 Data records 24

3.1 Methods . . . 25

3.2 Details of enhanced MedMNIST2D (MedMNIST2D+) . . . 25

3.2.1 PathMNIST+ . . . 25

3.2.2 ChestMNIST+ . . . 25

3.2.3 DermaMNIST+ . . . 26

3.2.4 OCTMNIST+ . . . 27

3.2.5 PneumoniaMNIST+ . . . 27
3.2.6 RetinaMNIST+ . . . 28
3.2.7 BreastMNIST+ . . . 28
3.2.8 BloodMNIST+ . . . 29
3.2.9 TissueMNIST+ . . . 29

4 Experiments **30**
4.1 Training setup . . . 30
4.2 Results . . . 31
4.3 Analysis . . . 31
4.3.1 Comparison of performance measures . . . 31
4.3.2 Inspection of misclassified images . . . 35

5 Discussion **39**

6 Conclusion **41**

A Appendix **43**
A.1 Code availability . . . 43

Bibliography **44**

List of Figures

1 The receptive field for the purple result of multiple convolutions is shown in blue. 7

2 A cross correlation. The input matrix on the left is multiplied with the kernel in the middle to produce an element in the output matrix on the right. The input matrix is padded with a value of 1. With a stride of two, in two steps, the top right element will be computed via a multiplication of the field indicated by the 3×3 matrix in the top right corner of the input matrix and the kernel. 7

3 Different activation functions. 9

4 The attention mechanism. It is also known as scaled dot-product attention. Illustration taken from Vaswani et al. 11

5 Multi-headed attention drives the transformer architecture. Illustration taken from Vaswani et al. 11

6 Wu and He illustrate the differences in calculating the expectation and variance for the different normalization methods. 12

7 Huang et al. (b) illustrate the drop in survival rate for the generation of the noise tensor. Illustration modified. 14

8 Huang et al. (a) present the dense building block. Residual connections exist from one layer to all subsequent layers in the block. 21

9 Mobile inverted bottleneck block in its basic (left) and fused (right) form. The expansion in channels happens through two different convolutions in the basic version while the fused variant uses only one convolution. It is noted that EfficientNets do not use the squeeze-and-excitation mechanic in fused mobile inverted bottleneck blocks. Illustration taken from Tan and Le (b). 22

10 Illustration from Dosovitskiy et al. visualizes their vision transformer architecture. 24

11 Comparison of the original sample (c), the MedMNIST version (a) as well as the enhanced version (b) on the training image at index 47424 of the Pathology dataset. 25

12 The image at index 3447 of the test set. Symbols from the original image (c) can still be identified in ChestMNIST+ (b). 26

13 The image at index 1133 of the training set of DermaMNIST and DermaMNIST+. 26

14 The shape of the train image at index 84131 is still recognizable in (b). 27

15 The object seen in the source image (c) can also be seen in ChestMNIST+ (b) at index 3507 of the training split. This is not the case for PneumoniaMNIST (a). 27

16 Individual venules and arterioles are still visible in RetinaMNIST+ (b). 28

17 Benign breast tissue is shown. From left to right: ChestMNIST, ChestMNIST+, both at index 2 of the test split, and the source image. 28

18 Comparison of the different versions of an image showing a blood cell. At index 10819 of their training split, the ChestMNIST, ChestMNIST+ and source image are shown from left to right. 29

19 Slices of the source image which can be used to create the maximum projection seen in Figure 20(c). 29

20 The axial-axis maximum projection of the slices shown in Figure 19 are downsampled for TissueMNIST (a) and upsampled for TissueMNIST+ (b). 30

21 Box plots of both performance measures for each model. Newer CNNs perform better than older ones. The newest architecture, the vision transformer, consistently performs the worst. 33

22 AUC of each model is plotted against the number of samples of the dataset. Additionally, linear regression curves are shown. There is no significant relationship between dataset size and AUC for CNNS. The vision transformer's performance scales with dataset size. 34

23 Misclassified image from the enhanced PathMNIST dataset. All models classified this case of cancer-associated stroma as smooth muscle. . 35

24 An x-ray scan from the enhanced ChestMNIST dataset showing a case of infiltration. No model in the experiment detected this case. . . 36

25 In the experiment, all models misclassified this case of melanoma as benign. Image is sampled from the enhanced DermaMNIST dataset. . 36

26 All models in the experiment classified this image from the enhanced OCTMNIST dataset as a case of choroidal neovascularization. However, this is a case of drusen. 36

27 Image from the enhanced PneumoniaMNIST not showing a case of pneumonia. However, all models in the experiment classified it as such. 37

28 All models in the experiment did not ascribe the first level of diabetic retinopathy, but assigned it to the lowest level. Image is taken from the enhanced RetinaMNIST dataset. 37

29 Two cases of malignant breast scans from the enhanced BreastMNIST dataset. All models were unable to classify these instances as such. . 38

30 Two cases of normal breast scans from the enhanced BreastMNIST dataset. In the experiment, all models classified these as maligned cases. 38

31 An image from the enhanced BloodMNIST dataset. In the experiment all models mistook this neutrophil blood cell as an immature granulocyte. 38

32 In the experiment, all models classified this neutrophil blood cell as an eosinophil blood cell. Image is taken from the enhanced BloodMNIST dataset. 39

33 An image showing distal convoluted tubule that was classified by all models in the experiment as collecting duct or connecting tubule. The image is from the enhanced TissueMNIST dataset. 39

34 Different versions of a dermatoscopic image from the HAM1000 dataset (Tschandl et al., 2018; Tschandl). Strictly following the procedure described by the MedMNIST framework (b) does not result in a replica of the DermaMNIST version (a). Only when the source image is center-cropped before being downsized can a replica (c) be produced. 40

List of Tables

1 Overview of the chosen models for the experiment. Top 1 and top 5 accuracies (ACCs) are reported for the ImageNet-1K dataset. 17

2 Prelude and final operations of each CNN. 18

3 Architecture of the ResNets used for the experiments. 19

4 Architecture of DenseNet-121. Dense refers to one dense layer and its multiplication implies one dense block. 20

5 Architecture of EfficientNetV2-S 23

6 Performance of each model on each MedMNIST2D+ dataset. 32

7 Average performance of the chosen models in measures average AUC and average ACC over all enhanced 2D datasets 33

8 Differences in the performance measures of the MedMNIST v2 technical validation compared to the experiment done on the enhanced versions of the datasets. ResNet-{18,50} are equivalent to ResNet-{18,50} (224) in MedMNIST v2 with the addition that their models were not pre-trained. Runs on the enhanced versions are generally marginally better with the exception of DermaMNIST, OCTMNIST and RetinaMNIST. ResNet-50 becomes more accurate. Both variants make AUC improvements on the latter dataset. 34

9 Difference in averaged AUC and ACC of the two ResNet variants compared to the experiments done on the enhanced versions of the MedMNIST2D datasets. 35

Notation

This section provides a concise reference describing notation as used in the book by Goodfellow et al. (2016). If you are unfamiliar with any of the corresponding mathematical concepts, Goodfellow et al. (2016) describe most of these ideas in chapters 2–4.

Numbers and Arrays

a	A scalar (integer or real)
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix
$\mathbf{A}$	A tensor
$\boldsymbol{I}_n$	Identity matrix with n rows and n columns
$\boldsymbol{I}$	Identity matrix with dimensionality implied by context
$\boldsymbol{e}^{(i)}$	Standard basis vector $[0, \dots, 0, 1, 0, \dots, 0]$ with a 1 at position i
$\text{diag}(\boldsymbol{a})$	A square, diagonal matrix with diagonal entries given by $\boldsymbol{a}$
a	A scalar random variable
$\mathbf{a}$	A vector-valued random variable
$\mathbf{A}$	A matrix-valued random variable

Sets and Graphs

$\mathbb{A}$	A set
$\mathbb{R}$	The set of real numbers
$\{0, 1\}$	The set containing 0 and 1
$\{0, 1, \dots, n\}$	The set of all integers between 0 and n
$[a, b]$	The real interval including a and b
$(a, b]$	The real interval excluding a but including b
$\mathbb{A} \backslash \mathbb{B}$	Set subtraction, i.e., the set containing the elements of $\mathbb{A}$ that are not in $\mathbb{B}$
$\mathcal{G}$	A graph
$Pa_{\mathcal{G}}(\mathrm{x}_i)$	The parents of x_i in $\mathcal{G}$

Indexing

a_i	Element i of vector $\boldsymbol{a}$, with indexing starting at 1
a_{-i}	All elements of vector $\boldsymbol{a}$ except for element i
$A_{i,j}$	Element i, j of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	Row i of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{:,i}$	Column i of matrix $\boldsymbol{A}$
$\mathsf{A}_{i,j,k}$	Element (i, j, k) of a 3-D tensor $\mathbf{A}$
$\mathbf{A}_{:,:,i}$	2-D slice of a 3-D tensor
a_i	Element i of the random vector $\mathbf{a}$

Linear Algebra Operations

$\boldsymbol{A}^\top$	Transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A}^+$	Moore-Penrose pseudoinverse of $\boldsymbol{A}$
$\boldsymbol{A} \odot \boldsymbol{B}$	Element-wise (Hadamard) product of $\boldsymbol{A}$ and $\boldsymbol{B}$
$\det(\boldsymbol{A})$	Determinant of $\boldsymbol{A}$

Calculus

$\frac{dy}{dx}$	Derivative of y with respect to x
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\nabla_{\boldsymbol{x}} y$	Gradient of y with respect to $\boldsymbol{x}$
$\nabla_{\boldsymbol{X}} y$	Matrix derivatives of y with respect to $\boldsymbol{X}$
$\nabla_{\mathbf{X}} y$	Tensor containing derivatives of y with respect to $\mathbf{X}$
$\frac{\partial f}{\partial \boldsymbol{x}}$	Jacobian matrix $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ of $f : \mathbb{R}^n \to \mathbb{R}^m$
$\nabla^2_{\boldsymbol{x}} f(\boldsymbol{x})$ or $\boldsymbol{H}(f)(\boldsymbol{x})$	The Hessian matrix of f at input point $\boldsymbol{x}$
$\int f(\boldsymbol{x}) d\boldsymbol{x}$	Definite integral over the entire domain of $\boldsymbol{x}$
$\int_{\mathbb{S}} f(\boldsymbol{x}) d\boldsymbol{x}$	Definite integral with respect to $\boldsymbol{x}$ over the set $\mathbb{S}$

Probability and Information Theory

$\mathrm{a} \perp \mathrm{b}$	The random variables a and b are independent
$\mathrm{a} \perp \mathrm{b} \mid \mathrm{c}$	They are conditionally independent given c
$P(\mathrm{a})$	A probability distribution over a discrete variable
$p(\mathrm{a})$	A probability distribution over a continuous variable, or over a variable whose type has not been specified
$\mathrm{a} \sim P$	Random variable a has distribution P
$\mathbb{E}_{\mathrm{x} \sim P}[f(x)]$ or $\mathbb{E}f(x)$	Expectation of $f(x)$ with respect to $P(\mathrm{x})$
$\mathrm{Var}(f(x))$	Variance of $f(x)$ under $P(\mathrm{x})$
$\mathrm{Cov}(f(x), g(x))$	Covariance of $f(x)$ and $g(x)$ under $P(\mathrm{x})$
$H(\mathrm{x})$	Shannon entropy of the random variable x
$D_{\mathrm{KL}}(P \Vert Q)$	Kullback-Leibler divergence of P and Q
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution over $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

Functions

$f : \mathbb{A} \to \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
$f \circ g$	Composition of the functions f and g
$f(\boldsymbol{x}; \boldsymbol{\theta})$	A function of $\boldsymbol{x}$ parametrized by $\boldsymbol{\theta}$. (Sometimes we write $f(\boldsymbol{x})$ and omit the argument $\boldsymbol{\theta}$ to lighten notation)
$\log x$	Natural logarithm of x
$\sigma(x)$	Logistic sigmoid, $\dfrac{1}{1 + \exp(-x)}$
$\zeta(x)$	Softplus, $\log(1 + \exp(x))$
$\Vert \boldsymbol{x} \Vert_p$	L^p norm of $\boldsymbol{x}$
$\Vert \boldsymbol{x} \Vert$	L^2 norm of $\boldsymbol{x}$
x^+	Positive part of x, i.e., $\max(0, x)$
$\mathbf{1}_{\mathrm{condition}}$	is 1 if the condition is true, 0 otherwise

Sometimes we use a function f whose argument is a scalar but apply it to a vector, matrix, or tensor: $f(\boldsymbol{x})$, $f(\boldsymbol{X})$, or $f(\mathbf{X})$. This denotes the application of f to the array element-wise. For example, if $\mathbf{C} = \sigma(\mathbf{X})$, then $C_{i,j,k} = \sigma(X_{i,j,k})$ for all valid values of i, j and k.

Datasets and Distributions

p_{data}	The data generating distribution
$\hat{p}_{\text{data}}$	The empirical distribution defined by the training set
$\mathbb{X}$	A set of training examples
$\boldsymbol{x}^{(i)}$	The i-th example (input) from a dataset
$y^{(i)}$ or $\boldsymbol{y}^{(i)}$	The target associated with $\boldsymbol{x}^{(i)}$ for supervised learning
$\boldsymbol{X}$	The $m \times n$ matrix with input example $\boldsymbol{x}^{(i)}$ in row $\boldsymbol{X}_{i,:}$

1 Introduction

In the realm of medicine, the utilization of medical imaging has assumed a principal role, facilitating non-invasive diagnostic procedures and enables invaluable insights for treatment planning (European Society of Radiology (ESR); Giardino et al.). Deep learning moved into the spotlight as a pivotal tool for medical imaging in response to increased demand for precise and dependable algorithms (Puttagunta and Ravi; Xiao et al.). However, it becomes important to assess the effectiveness on a standardized, high-quality medical database, given the rapid development of deep learning models (Luca et al.). This practice is referred to as benchmarking (Dhar and Shamir).

To this end, baseline neural networks offer a straightforward and promising starting point to enable the comparison of diverse algorithms and architectures. Researchers can identify strengths and limitations through experimentation on common medical datasets and make judgments regarding the selection of the appropriate model for a particular task (Wang et al., c).

Moreover, benchmarking with baseline neural networks serves as a foundation for contrasting different training setups (Klein and Hutter). This, in turn, allows for analysis and interpretation of the factors that contribute to the performance of deep learning models in the domain of medical imaging.

1.1 Related Work

The MedMNIST v2 (Yang et al., 2023) framework serves as inspiration for this work. It is a compilation of publicly available medical imaging datasets and standardized in a MNIST-like format. Its diversity in terms of scale, covered modalities and classification tasks makes it a desirable biomedical benchmark. However, the low resolution of the datasets make it less attractive since medical diagnoses based on medical imaging rely on identifying small features that are more visible at high resolution.

For image segmentation in a medical context, the medical segmentation decathlon (Simpson et al.) provides a similar database for image volumes. Data is collected from various locations in different settings, but delivered in a standardized format. The aim of the decathlon is the facilitate development of generalisable 3D semantic segmentation in a biomedical realm. Reporting on the results focuses on the end-to-end system that was used to achieve the results (Antonelli et al.).

1.2 Contribution

This work offers a benchmark on four convolutional neural networks and one based on the transformer architecture in biomedical image classification tasks. Furthermore, it provides insight into how an increase in resolution might affect performance as the datasets are upscaled versions of another benchmark and two neural network

models are reused for experimentation. To create the database, methodology outlined by Yang et al. (2023) is followed to create 2D datasets for binary, multi-class and multi-label classification. The method used for obtaining the higher resolution datasets is made publicly available. Lastly, a brief inspection of common misclassifications can be found.

The details of the training and modification scripts can be found in the provided code repository.

2 Methods

Deep learning is a subfiled of machine learning that automates discovery of patterns through data via neural networks. Raw input data is processed by layers of interconnected nodes to extract hierarchical features with a mathematical foundation (Goodfellow et al., 2016). This paradigm shift has led to remarkable advancements in domains like computer vision or natural language processing (LeCun et al., 2015).

2.1 Learning of Neural Networks

In essence, neural networks are chains of functions, i.e., compositions of many functions.

To ease the understanding of this, the computations that are implied by these function are often represented via a graph called the computational graph. It shows the flow of information from the input to the output of the composed functions. During training or inference, the neural network gets an input and calculates output step by step which can be visualized in such a graph.

To evaluate a neural network its performance can be measured based on how well it classified the input. By nudging the network's parameters in the right direction, its performance will increase.

Parameters are the components of functions that are variable and in the context of neural networks learnable. Take the linear function $y = mx + t$ for example. The result y is a function of input x for some slope m and intercept t. In this example, m and t are parameters, that can be adjusted to influence the input of x.

Learning refers to the adjustment of these parameters to influence all future inputs of x in a way such that the performance is increased. The amplitude of learning is controlled by the learning rate. The direction of learning is computable.

Let the linear function be the neural network to be trained and be parameterized by its slope and intercept: $f(x; m, t) = mx + t$. Furthermore, let the goal of the neural network be to predict data from the matrix

$$\boldsymbol{X} = \begin{bmatrix} 3 & 0 & 2 & 1 & 4 \\ 3 & 1 & 2 & 2 & 3.5 \end{bmatrix}^\top .$$

Performance of the neural network is measured by the distance of the predicted point of the neural network to the true pointed given by $\boldsymbol{X}$. The used function for this purpose is also known as the squared error and defined by $l(\hat{y}, y) = (y - \hat{y})^2$ for $i \in \{1, \ldots, 5\}$.

With the training set $\mathbb{X} = \{\boldsymbol{x}^{(i)} \mid 1 \leq i \leq 3\}$ the network can begin its learning process. Say m and t are initialized to 1 and 0.5 respectively. Then,

$$f(x_1^{(1)}) = f(3) = 1 \times 3 + 0.5 = 3 + 0.5 = 3.5$$

would be its prediction and its squared error

$$l(f(x_1^{(1)}), y^{(i)}) = (y^{(i)} - 3.5)^2 = (3 - 3.5)^2 = (-0.5)^2 = 0.25 \ .$$

Calculating the derivative of this error with respect to the individual components of f will yield the direction of learning for the component with the application of the chain rule:

$$\frac{\partial l}{\partial m} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial m}$$

and

$$\frac{\partial l}{\partial t} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial t} \ .$$

The common partial derivative of those partial derivatives is calculable and serves as the first step of propagating gradient information backwards through the network:

$$\frac{\partial l}{\partial f} = \frac{\partial}{\partial f} l = \frac{\partial}{\partial f}(3 - f)^2 = -2(3 - f)$$

and since $f = 3.5$ at this point at the calculation

$$-2(3 - f) = -2(3 - 3.5) = -2 \times (-0.5) = 1 \ .$$

Furthermore, the local gradients for m and t are calculated as follows:

$$\frac{\partial}{\partial m} mx + t = x = 3$$

and

$$\frac{\partial}{\partial t} mx + t = 1$$

which finally can be used to calculate their gradients with respect to the loss function of the squared error by

$$\frac{\partial l}{\partial m} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial m} = 1 \times 3 = 3$$

and

$$\frac{\partial l}{\partial t} = \frac{\partial l}{\partial f}\frac{\partial f}{\partial t} = 1 \times 1 = 1 \ .$$

In this case, these gradients represent the level of influence of the parameters on the loss function. A parameter with a positive gradient increases the loss function's value; with a negative gradient the loss function's value decreases. Since the loss function represents the level of error of the network with respect to the task of the network, letting a parameter's gradient positively influence itself will increase the error in the future. Hence, multiplying by a negative fraction of its gradient, a parameter will be nudged in the direction of minimizing the error of the network.

The fraction that determines the optimizing step size is called the learning rate. Let the learning rate $\gamma \in \mathbb{R}$ be $\gamma = 0.1$. Then, the parameters are updated by $\hat{m} = m - \gamma\frac{\partial l}{\partial m} = 1 - 0.1 * 3 = 1 - 0.3 = 0.7$ and $\hat{t} = 1 - 0.1 = 0.9$.

A more advanced optimizing process is described in the following.

Let the learnable network be $f(\boldsymbol{X}; \boldsymbol{\Theta})$ with parameters $\boldsymbol{\Theta}$ and learning rate γ. The adaptive moment estimation (Adam) optimizer (Kingma and Ba) is defined such that at timestep t of a training batch

$$\boldsymbol{\Theta}^{(t)} = \boldsymbol{\Theta}^{(t-1)} - \gamma\frac{\hat{\boldsymbol{M}}^{(t)}}{\sqrt{\hat{\boldsymbol{V}}^{(t)}} + \epsilon} \ , \tag{1}$$

where ϵ is a small scalar for numerical stability and typically $\epsilon = 10^{-8}$. This essentially prevents a division by 0. Furthermore $\hat{\boldsymbol{M}}^{(t)}$ and $\hat{\boldsymbol{V}}^{(t)}$ are bias-corrected estimates of the first non-raw and second raw moment respectively. They are defined in terms of their non-corrected moments by

$$\hat{\boldsymbol{M}}^{(t)} = \frac{\boldsymbol{M}^{(t-1)}}{(1 - \beta_{\boldsymbol{M}}^t)}$$

and

$$\hat{\boldsymbol{V}}^{(t)} = \frac{\boldsymbol{V}^{(t-1)}}{(1 - \beta_{\boldsymbol{V}}^t)}$$

respectively, where $\beta_{\{\boldsymbol{M},\boldsymbol{V}\}} \in [0, 1)$ is the decay rate of either moment estimate $\boldsymbol{M}$ or $\boldsymbol{V}$. Typically, $\beta_{\boldsymbol{M}} = 0.9$ and $\beta_{\boldsymbol{V}} = 0.999$. They are in turn defined by

$$\boldsymbol{M}^{(t)} = \beta_{\boldsymbol{M}}\boldsymbol{M}_{(t-1)} + (1 - \beta_{\boldsymbol{M}})\boldsymbol{G}^{(t)}$$

and

$$\boldsymbol{V}^{(t)} = \beta_{\boldsymbol{V}}\boldsymbol{V}^{(t-1)} + (1 - \beta_{\boldsymbol{V}})(\boldsymbol{G}^{(t)} \odot \boldsymbol{G}^{(t)}) \ ,$$

where $\boldsymbol{G}^{(t)} = \nabla_{\boldsymbol{\Theta}} l^{(t)}(\boldsymbol{\Theta}^{(t-1)})$ represents the gradient matrix with respect to the loss function l at timestep t. Initial first and second moment are set to 0, i.e., $\boldsymbol{M}^{(0)} = \boldsymbol{V}^{(0)} = 0$.

2.2 Neural Network Functions

Neural networks can be understood as a graph where neurons are nodes in a graph and their connections are edges. These nodes are functions as they take and input which is then transformed by the function to an output. Their basic unit of operation is a tensor, i.e., a multi-dimensional matrix holding elements of a specific data type. As a consequence the usual operations of matrices can be applied to the elements of a tensor such as addition, multiplication or transposition. In the following, more unusual functions operating on that datastructure are presented.

If a function f is defined on scalar, then this is denoted by $y = f(x)$, where y is the output scalar and x is the input scalar.

In the context of this work however, the most basic input is an image which can be understood as a two-dimensional array of pixel values. For gray-scale images for example, the intensity of white can be indicated by an integer between 0 and 255, referring black or white respectively. Since an image has two dimensions, say width w and height h, in mathematical notation a gray-scale image can be an element from the domain of $\{1, \ldots, 256\}^{w \times h}$. Such an element is called a matrix which is a two-dimensional array of values. In this case, the input image can be thought of as a matrix of size $w \times h$ containing values in the domain of $\mathbb{R}$.

While propagating through the network the input image is transformed by functions that produce values that are less interpretable than those indicating a gray scale. That is why it is common to project pixel values into the domain of $\mathbb{R}$, usually in the range of $[0, 1]$. Furthermore, the input and output of a function does not have to be in this range, but can be considered to be in $\mathbb{R}$ entirely. Consequently, $\boldsymbol{Y} = f(\boldsymbol{X})$ denotes the application of f on the input matrix $\boldsymbol{X}$ with an output matrix $\boldsymbol{Y}$, both with a domain of $\mathbb{R}$. Additionally, only square matrices shall be considered, i.e., $\boldsymbol{X}$ is an element of $\mathbb{R}^{d_{\boldsymbol{X}} \times d_{\boldsymbol{X}}}$, where the scalar $d_{\boldsymbol{X}}$ denotes the number of elements in a dimension of $\boldsymbol{X}$. This notation is simplified and extended to the output matrix $\boldsymbol{Y} \in \mathbb{R}^{d_{\boldsymbol{Y}}^2}$.

Not all images are gray-scale. Some images have color which are usually constructed through the combination of the color channels red, green and blue. The two-dimensional matrix can be extended by another dimension resulting in a three-dimensional array of values. For an image with the red, green and blue color channels it can be said that such an image is an element from the domain of $\{1, \ldots, 256\}^{3 \times w \times h}$. Elements with at least three dimensions are called tensors. Therefore, color images can be considered tensors. To generalize, the input and output matrices used for definitions are extended with their own channels. $\mathbf{Y} = f(\mathbf{X})$ denotes the resulting $c_{out} \times d_{\boldsymbol{Y}}^2$ output tensor $\mathbf{Y}$ obtained by applying f on input tensor $\mathbf{X} \in \mathbb{R}^{c_{in} \times d_{\boldsymbol{X}}^2}$. This means that $\mathbf{X}$ and $\mathbf{Y}$ are arrays of c_{in} and c_{out} matrices respectively; one matrix for each of the c_{in} or c_{out} channels. Say the matrix $\boldsymbol{X}$ is in the seventh channel of input tensor $\mathbf{X}$, then slicing $\mathbf{X}$ at the seventh channel yields that matrix $\boldsymbol{X} = \mathbf{X}_{7,:,:}$.

To train neural networks, it is typical to feed it the input in collections with a fixed size. One such collection is called a batch and it can be thought of as a four-dimensional tensor. Hence, $\mathbf{B} \in \mathbb{R}^{d_{\mathbf{B}} \times c_{in} \times d_{\boldsymbol{X}}^2}$ is considered the batch of input tensor

with a batch size of $d_{\mathbf{B}}$. The application of f obtains the $d_{\mathbf{C}} \times c_{out} \times d_{\mathbf{Y}}^2$ batched output tensor $\mathbf{C} = f(\mathbf{B})$.

Unless explicitly stated, if a function f is defined for for a vector, matrix, or tensor, then f is applied across the first dimension of the array until the dimensions fit to the definition of f. For example, if $\mathbf{Y} = \text{softmax}(\mathbf{X})$, then $\mathbf{Y}_{i,j,:} = \text{softmax}(\mathbf{X}_{i,j,:})$ for all valid values of i and j.

2.2.1

2.2.2 Convolution

The direct mapping of every pixel to an artifical neuron was used early when trying to build artificial neural networks. Layers of these neuron represented linear transformation with an additional non-linear component. By scaling up the amount of neuron layers used, both in depth and width, fairly promising results were produced for small-resolution classification of hand-written numbers.

However, when classifying images with the potential of depicting more than one of ten classes, these methods were lacking since parameter count increases rapidly in these fully connected networks. Neurophysiological studies in the middle of the 20th century (Hubel and Wiesel, a,b) discovered different cell types that either responded to stimulus in a specific region of the visual field or regardless of its position. Areas of the visual field that activate these cells is called the receptive field. The potential of the apparently inherent and promising property of shift-invariance of the latter cell type inspired work (Fukushima, b; Zhang et al.; Hinton et al., a; LeCun et al., b) to achieve the recognition of such complex features by the composition of simpler and local features.

Convolutions are employed to extract high-level features that with increasing depth of the network are transformed to even higher-level features. These then form the basis for a the classical fully-connected "sub-network" classifying the image. Additionally, convolutions are translation equivariant which means that shifting the input results in a shifted output. This does not necessarily mean that the architecture employing them are shift-invariant and requires careful designing (Zhang). However, this is in contrast to traditional fully-connected networks that merely "memorize" the pixels of images and do not extract any kind of features.

The receptive field size is referred to as the kernel size. Kernels are the filters that are shifted over images to recognize features. The amount of shift during processing is called the stride. Additionally, the image can be padded.

The receptive field in the context of convolutions is visualized in Figure 1.

If $n, s, p \in \mathbb{N}$ are the kernel size, stride and padding, the cross-correlation $\boldsymbol{Y} = \boldsymbol{K} * \boldsymbol{X}$ is defined to be such that

$$Y_{i,j} = \sum_{k,l} K_{k,j} \hat{X}_{(i-1)s+k,(l-1)s+j} \ , \tag{2}$$

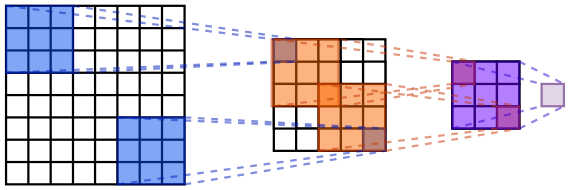


Figure 1: The receptive field for the purple result of multiple convolutions is shown in blue.

where $\boldsymbol{K}$ is a n^2 kernel matrix and $\hat{\boldsymbol{X}}$ is the $(d_{\boldsymbol{X}} + 2p)^2$ 0-padded input matrix such that

$$\hat{X}_{i,j} = \begin{cases} 0 & i \leq p \text{ or } j \leq p \text{ or } i > d_{\boldsymbol{X}} + p \text{ or } j > d_{\boldsymbol{X}} + p \\ X_{i,j} & \text{otherwise} \end{cases} . \tag{3}$$

This process is visualized in Figure 2.

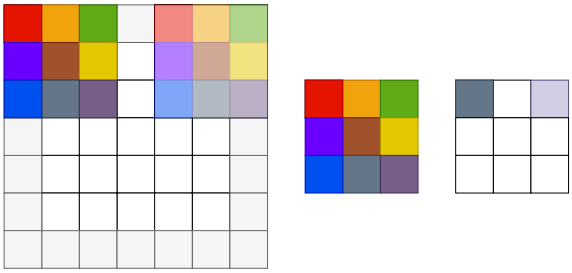


Figure 2: A cross correlation. The input matrix on the left is multiplied with the kernel in the middle to produce an element in the output matrix on the right. The input matrix is padded with a value of 1. With a stride of two, in two steps, the top right element will be computed via a multiplication of the field indicated by the 3 $\times$ 3 matrix in the top right corner of the input matrix and the kernel.

This operation is the basis of a convolution $\mathbf{Y} = \text{conv}(\mathbf{X})$. It is defined to be such that

$$\mathbf{Y}_{i,:,:} = \sum_j \mathbf{K}_{i,j,:,:} * \mathbf{X}_{j,:,:} , \tag{4}$$

where $\mathbf{K}$ is a $c_{out} \times c_{in} \times n^2$ tensor of n^2 kernel matrices, i.e., c_{in} kernel matrices for each of the c_{out} output channels.

The learnable parameters is the set of filter represented here by 2D-slices of $\mathbf{K}$ along the first and second dimension. For each input channel there is one kernel filter. The result of these cross-correlations are then summed up to represent one feature. Hence, the number of output channels is also referrend to as the number of feature maps.

However, this kind of convolution suffers from hight cost. With a constraint on our feature map size a more performant variant of convolution can be employed.

For $c_{out} = c_{in}$, depthwise convolution (Howard et al.) $\mathbf{Y} = \text{dconv}(\mathbf{X})$ is defined such that

$$\mathbf{Y}_{:,:,i} = \hat{\mathbf{K}}_{i,:,:} * \mathbf{X}_{:,:,i} \ , \tag{5}$$

where $\hat{\mathbf{K}}$ is a $c_{out} \times n^2$ tensor of n^2 kernel matrices, i.e., an array of n^2 kernel matrices with c_{out} elements.

For depthwise convolutions, each input channel uses its own set of filters. This way, the kernel tensor can be reduced by one dimension: $\hat{\mathbf{K}}$ is three-dimensional whereas $\mathbf{K}$ is four-dimensional. In any case, convolutions try to capture pattern correlations.

2.2.3 Pooling

Convolutions primarily increase the tensor's channel size during its forward flow in the neural network. As already mentioned, these operations are also expensive and are applied channel-wise. To reduce the amount of strides across the transformed tensor matrices, pooling is employed to reduce its resolution.

With the same antecedent as convolutions, defined in chapter 2.2.2, the max pool (Yamaguchi et al.; Graham) $\mathbf{Y} = \text{maxp}(\mathbf{X})$ is defined to be such that

$$Y_{i,j,k} = \max(\{\hat{X}_{i,(j-1)s+l,(k-1)s+m} \mid l \leq n \text{ and } m \leq n\}) \ , \tag{6}$$

where $\hat{\mathbf{X}}$ is again the padded tensor defined in equation 3. Despite pooling also using the kernel size n as a variable controlling the behavior of the operation, the type of the kernel differs from convolutions. Kernels in convolutions are learnable matrices that are used for multiplication, whereas the kernel of pooling operations only refers to the area over which the type of pooling is applied.

Furthermore, the average pool $\mathbf{Y} = \text{avgp}(\mathbf{X})$ is defined to be such that

$$Y_{i,j,k} = \frac{1}{n^2} \sum_{i=1}^{k} \sum_{j=1}^{k} X_{i,(j-1)s+l,(k-1)s+m} \ . \tag{7}$$

Finally, an adaptive version of these functions will use a stride, kernel size and padding defined by

$$s = \frac{d_{\boldsymbol{X}}}{d_{\boldsymbol{Y}}} \ , \ n = d_{\boldsymbol{X}} - s(d_{\boldsymbol{Y}} - 1) \text{ and } p = 0 \ .$$

2.2.4 Activation

Traditional fully-connected networks used linear transformation to indicate the readiness of a neuron. Non-linear transformations then allowed for non-linear relationships to be encoded in the neural network. One of the oldest activation functions is the (logistic) sigmoid function σ (Verhulst, b,c,a; Han and Moraga).

Although already used in 1969, the rectified linear unit (ReLU) function (Fukushima, a) $y = \mathrm{ReLU}(x)$ gained popularity only around 2010 (Nair and Hinton) because of it allowed for deeper networks. It is defined by

$$\mathrm{ReLU}(x) = x^+ = max(0, x) \ . \tag{8}$$

Essentially, the input is zeroed out when non-positive.

With probabilistic theoretical backing, Hendrycks and Gimpel propose scaling the input according to its magnitude compared to other inputs. The sigmoid linear unit (SiLU) function $y = \mathrm{SiLU}(x)$ is defined by

$$\mathrm{SiLU}(x) = x\sigma(x) \ . \tag{9}$$

Here, the input is scaled according to the cumulative distribution function (CDF) of the logistic distribution $\sigma(x)$. It can be seen as an approximation of the CDF of a normal distribution which is used as the scaling factor for the more performant proposed linear unit.

The Gaussian Error Linear Unit (GELU) function $y = \mathrm{GELU}(x)$ is defined by

$$\mathrm{GELU}(x) = x\Phi(x) \ , \tag{10}$$

where $\Phi(x) = p(\mathrm{x} \leq x)$ and $\mathrm{x} \sim \mathcal{N}(0, 1)$.

These activation functions are visualized in figure 3.

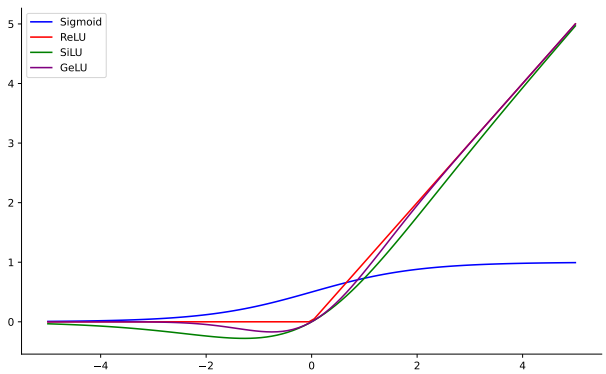


Figure 3: Different activation functions.

For multidimensional scalars a different approach is taken. If $\boldsymbol{x}$ is an input vector, the softmax (Boltzmann; Luce; Bridle, a,b) $\boldsymbol{y} = \text{softmax}(\boldsymbol{x})$ is defined such that

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \ . \tag{11}$$

By exponentiating the values of the input vector, negative values will become positive values. The sum of all inputs is used to normalize the scalars contained in $\boldsymbol{x}$. Output $\boldsymbol{y}$ can then be interpreted as probabilities, since its scalars will add up to 1. This activation function is oftentimes used to normalize the final output of a network in the context of binary- and multi-class classification tasks.

2.2.5 Attention

Data in natural language processing is sequence-based. For machine translation, the direct translation of source word into the target language would result in lacking results. Next to grammar, there is also the meaning of words that change based on the context. By application of a compatibility function to embedded input and other elements of the input sequence an attention score can be calculated. This results in a weighing of each element in the sequence based on its relationship to other elements in the sequence. Finally, the combination of this relationship information with another version of the embedded input yields a context-aware representation of the input.

If $\boldsymbol{Q}$ is a query matrix and $\boldsymbol{K}$ is a key matrix, both of dimension d, and $\boldsymbol{V}$ is a value matrix, attention (Vaswani et al.), or scaled dot-product attention, $\boldsymbol{Y} = \text{att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ is defined by

$$\text{att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}})\boldsymbol{V} \ . \tag{12}$$

Here, the compatibility function is simple matrix-multiplication between query and keys. The square-roots purpose is normalization of the attention score.

The transformer architecture (Vaswani et al.) was the first full self-attention architecture at scale. To introduce different representation subspaces, the notion of attention heads was introduced to allow for parallel computation.

Multi-headed attention (Vaswani et al.) $\mathbf{Y} = \text{mha}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$ is defined by

$$\text{mha}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \begin{bmatrix} \mathbf{H}_{i,:,:} \\ \vdots \end{bmatrix} \hat{\boldsymbol{W}} \tag{13}$$

where $\mathbf{H}$ is an array of head matrices $\boldsymbol{H}$ such that $\mathbf{H}_{i,:,:} = \text{att}(\boldsymbol{Q}\boldsymbol{W}_{\boldsymbol{Q}}^{(i)}, \boldsymbol{K}\boldsymbol{W}_{\boldsymbol{K}}^{(i)}, \boldsymbol{V}\boldsymbol{W}_{\boldsymbol{V}}^{(i)})$, $\boldsymbol{W}_{\boldsymbol{M}}^{(i)}$ is the i-th weight matrix for projecting $\boldsymbol{M} \in \{\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\}$ and $\hat{\boldsymbol{W}}$ is the final weight matrix.

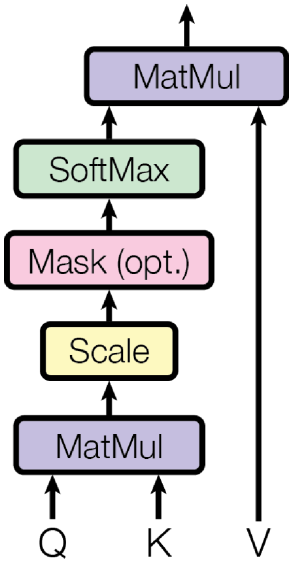


Figure 4: The attention mechanism. It is also known as scaled dot-product attention. Illustration taken from Vaswani et al.

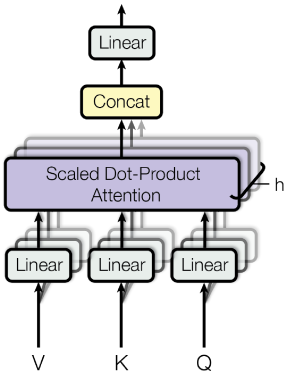


Figure 5: Multi-headed attention drives the transformer architecture. Illustration taken from Vaswani et al.

Linear projections create individual embeddings for each head. A final multiplication with another learnable weight matrix relates all subspace to produce an output with the same shape as the input.

2.2.6 Normalization

Choosing the right hyperparameters for the successful convergence of models cannot be guided by direct calculation. Aspects like weight initialization and regularizations factor into the decision. The introduction of normalization of tensors flowing through the network during training helps with the model's convergence speed and

performance by avoiding distribution shift of the information flowing through the network (Ioffe and Szegedy). There are different types of normalization.

In general, the normalization $\mathbf{C} = \text{norm}(\mathbf{B})$ is defined such that

$$C_{i,j,k,l} = \gamma_j \hat{B}_{i,j,k,l} + \beta_j \ , \tag{14}$$

where $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^{c_{in}}$. These trainable vectors transform the shifted input batch to make up for possible lost representation potential. The shifted element of the input batch $\hat{B}_{i,j,k,l}$ is defined by

$$\hat{B}_{i,j,k,l} = \frac{1}{\sqrt{\text{Var}(B_{i,j,k,l})}}(B_{i,j,k,l} - \mathbb{E}(B_{i,j,k,l})) \ ,$$

with $\text{Var}(B_{i,j,k,l})$ and $\mathbb{E}(B_{i,j,k,l})$ being the expectation and variance of $B_{i,j,k,l}$. They are defined by

$$\mathbb{E}(B_{i,j,k,l}) = \frac{1}{|S_{i,j,k,l}|} \sum_{(m,n,o,p) \in S_{i,j,k,l}} B_{m,n,o,p}$$

and

$$\text{Var}(B_{i,j,k,l}) = \frac{1}{|S_{i,j,k,l}|} \sum_{(m,n,o,p) \in S_{i,j,k,l}} (B_{m,n,o,p} - \mathbb{E}(B_{i,j,k,l}))^2 + \epsilon \ .$$

$\mathbf{S}$ are the $d_{\mathbf{B}} \times c_{in} \times d_{\boldsymbol{X}} \times d_{\boldsymbol{X}}$ coordinate tensor holding sets of indeces for which expectation and variance are computed. $|\cdot|$ is the cardinality of a set, i.e., it denotes the function returning the number of elements that are contained in a set. $\epsilon \in \mathbb{R}$ is a small constant for numerical stability.

Each type of normalization applies equation 14 with different sets contained in tensor $\mathbf{S}$. They are visualized by figure 6.

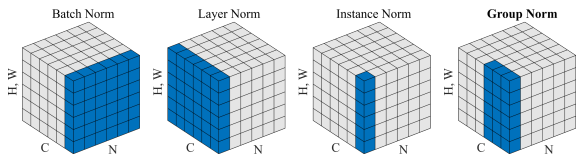


Figure 6: Wu and He illustrate the differences in calculating the expectation and variance for the different normalization methods.

For batch normalization (Ioffe and Szegedy), which introduced this mechanism, $\mathbf{S}$ is defined to be such that

$$S_{i,j,k,l} = \{(m, j, n, o) \mid B_{m,j,n,o}\} \ . \tag{15}$$

This means a single feature map is normalized across the entire batch. Layer normalization (Ba et al.) does the opposite: One sample of the batch is fixed and expecation aswell as variance is calculated across all feature maps. Hence, $\mathbf{S}$ is defined to be such that

$$\mathsf{S}_{i,j,k,l} = \{(i,m,n,o) \mid \mathsf{B}_{i,m,n,o}\} \ . \tag{16}$$

Group normalization (Wu and He) was introduced under the assumption that distributions in groups of channels differ enough such that dividing them into groups increases representational ability. If $g \in \mathbb{N}$ is the number of groups, $\mathbf{S}$ is defined to be such that

$$\mathsf{S}_{i,j,k,l} = \{(i,m,n,o) \mid \mathsf{B}_{i,m,n,o} \text{ and } \lfloor \frac{o}{c/g} \rfloor = \lfloor \frac{l}{c/g} \rfloor\} \tag{17}$$

where $\lfloor \cdot \rfloor$ is the floor operation. It can clearly be seen that if $g = 1$, group normalization is identical to layer normalization.

2.2.7 Linear

The linear transformation $\boldsymbol{Y} = \text{lin}(\boldsymbol{X})$ is defined by

$$\text{lin}(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{A}^\top + \boldsymbol{b} \ , \tag{18}$$

where $\boldsymbol{A}$ is a $c_{out} \times c_{in}$ weight matrix and $\boldsymbol{b}$ an array of c_{out} elements. This transformation was the basic building block of early neural networks. It is the classical linear function extended to higher dimensions. Furthermore, in the context of neural networks, the variable components $\boldsymbol{A}$ and $\boldsymbol{b}$ are learnable.

2.2.8 Regularization

Overfitting occurs when a complex enough network is able to model the training data very well but fails to generalize to perform well on unseen test data. To counteract this in training, elements of the input can be randomly dropped, i.e., zeroed out. This prevents the learning of too complex relationships between the network's components and forces them to generalize.

If $p \in [0,1]$ is the dropout probability, the dropout (Hinton et al., b) $y = \text{dropout}(x)$ is defined by

$$\text{dropout}(x) = x\text{r}\frac{1}{1-p} \ , \tag{19}$$

where r is sampled from a Bernoulli distribution with

$$P(\text{r} = 0) = p$$

and

$$P(\mathrm{r} = 1) = 1 - p \ .$$

An extension of this is stochastic depth. Instead of randomly dropping individual elements of a tensor, entire samples of a batch are ignored.

Thus, the stochastic depth (Huang et al., b) $\mathbf{C} = \mathrm{sdepth}(\mathbf{B})$ is defined by

$$\mathrm{sdepth}(\mathbf{B}) = \mathbf{B}\mathrm{dropout}(\mathbf{N}) \tag{20}$$

where $\mathbf{N}$ is a $d_{\mathbf{B}} \times 1^3$ noise tensor.

For residual neural networks that have skip connections, this can be understood as training an ensemble of networks with varying depth. By linearly increasing the dropout probability of sdepth from a lower bound to an upper bound with increasing depth, there is an increasing chance that only the skip connection is used. This is illustrated in Figure 7. Learning of the network component is omitted in that instance.

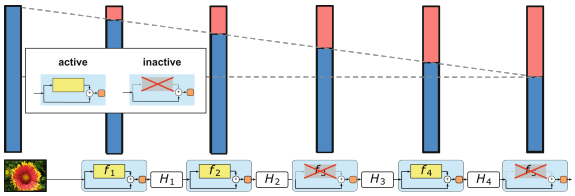


Figure 7: Huang et al. (b) illustrate the drop in survival rate for the generation of the noise tensor. Illustration modified.

2.2.9 Loss

As training is based on gradients that are calculated with respect to the performance of the model, one must be able to differentiate the performance function. Loss functions are such differentiable functions that indicate the error of the network.

If $n \in \mathbb{N}$ is the number of classes, $\boldsymbol{x} \in \mathbb{R}^n$ is an input vector and $c \in \{1, \dots, n\}$ is a class index, the cross entropy loss $y = \mathrm{ce}(\boldsymbol{x}, c)$ is defined by

$$\mathrm{ce}(\boldsymbol{x}, c) = -\log \frac{\exp(x_c)}{\sum_i \exp(x_i)} \ . \tag{21}$$

The input of the negative natural logarithm is recognizable as the softmax calculation of one vector element. Consequently, the model's notion of certainty in predicting the target class is calculated with a result between 0 and 1. Applying the negative natural logarithm of this certainty score will result in loss converging zero the closer its prediction was to 1, while growing towards infinity the closer it was to 0. Certain and correct classifications are rewarded with low error and uncertain and wrong predictions are punished with high error.

However, cross entropy loss is only applicable in classification tasks where only one class is the target class. In the setting of multi-label classification, multiple target classes are possible. Binary cross entropy loss can be an indicator for the error rate of a model.

If $\boldsymbol{v} \in \{0, 1\}^n$ is a target vector, the binary cross entropy loss $y = \text{bce}(\boldsymbol{x}, \boldsymbol{v})$ is defined such that

$$\text{bce}(\boldsymbol{x}, \boldsymbol{v}) = \frac{1}{n} \sum_i -(v_i \log x_i + (1 - v_i) \log(1 - x_i)) \ . \tag{22}$$

This loss function must be used in conjunction with some activation function on $\boldsymbol{x}$ such that the values are within $[0, 1]$. Otherwise, the loss function will be undefined due to the application of the natural logarithm on a value below 0.

For each class, the confidence score of a class is again mapped to the same bounds with the same convergence properties as cross entropy loss. In the end, the loss is averaged across all classes.

To simplify, fix some i. Then, the term inside the sum can be simplified to one of two expressions.

If $v_i = 0$, then the loss for the i-th class is $-\log(1 - x_i)$. A high confidence score x_i will result in a low natural logarithm since it subtracts from 1. Below 1, the natural logarithm takes on negative values. Finally, the sign is flipped by multiplication with -1. In this case, a low value of x_i is rewarded with a loss close to 0; a high value is penalized with a loss growing towards infinity.

Analogously, if the target class is to be classified by the model, i.e., $v_i = 1$, then the term inside the sum is simplified to $-\log x_i$. It should be clear that the same properties hold with switched conditions on x_i.

2.2.10 Performance measures

Next to indicating the performance of a classification model via a loss function, non-differentiable approaches can be taken. For the evaluation of classification models, two measures are considered in this work: the area under the receiver operating characteristic (AUC) and accuracy (ACC).

Both measures are based on the outcomes of the confusion matrix which contrasts the hypothesized classes of the model with the true classes of the samples processed by the model. True classes are either positive (P) or negative (N). In the most basic case of binary classification, the model hypothesizes about the presence or absence of a class. If the hypothesized presence is correct, one speaks of a true positive (TP); is the hypothesized absence correct, one speaks of a true negative (TN). Similarly, if the hypothesized presence is incorrect, one speaks of a false positive (FP). Again analogously, the wrongly hypothesized absence is a case of a false negative (FN).

From these primitives, a range of derivative measures can be calculated. The ACC is the percentage of cases where the hypothesis matches the target:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}} \ .$$

Relating TPs to Ps yield the true positive rate (TPR):

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} \ .$$

The false positive rate (FPR) is defined analogously along the true class' dimension:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}} \ .$$

However, the output of neural networks for classification not directly be mapped to the domain of a confusion matrix. Usually, confidence scores, not necessarily lying in $[0, 1]$, do not make a final decision about what class or classes to finally predict.

This changes with the introduction of a decision threshold. Values under the threshold are interpreted as a negative prediction; values above the threshold as a positive one. Varying the threshold yields multiple confusion matrices. Additionally, the tasks of multi-class and multi-label classification imply the existance of confusion matrices with increased dimension size.

For ACC, a threshold of 0.5 is typically chosen and computed directly. To arrive at the AUC measure, the TPR is plotted against the FPR at different threshold values to yield the receiver operating characteristic. Reading the plot from either axis gives information about the performance of the model at a specific operating threshold. A higher sensitivity of detecting a true positive comes at the cost of also increase probability of predicting a false alarm. However, to get a general understanding of the models performance, the area under the characteristic's curve is taken resulting in the AUC measure.

The advantage of the AUC over the ACC measure is its insensitivity to class-imbalance. That is also the reason why the medical research community relies on it (Hanley and McNeil; Zweig and Campbell; Obuchowski; Pepe).

2.3 Neural Network Architectures

Neural network architectures often consist of multiple layers. Layers are a collection of functions. An architecture of a neural network describes the general skeleton of function compositions. A variant of such architecture, called a model, is explicit about the functions used.

Most of the models that are evaluated in the experiment are based on the convolution operation conv, defined by equation 4. These models are called convolutional neural networks (CNNs) (Li et al.). The models chosen from this family of neural networks are ResNet-18, ResNet-50 (He et al., b), DenseNet-121 (Huang et al., a) and EfficientNetV2-S (Tan and Le, b). The exception to this is a model based on

attention att, defined by equation 12, which found first success in natural language processing (NLP) (Zhou et al.). Vision transformer ViT-B/16 (Dosovitskiy et al.) is the chosen model based on this mechanic.

Table 1: Overview of the chosen models for the experiment. Top 1 and top 5 accuracies (ACCs) are reported for the ImageNet-1K dataset.

Model	ACC@1	ACC@5	Params	GFLOPS
ResNet-18	0.698	0.891	11.690M	1.81
ResNet-50	0.809	0.954	25.557M	4.09
DenseNet-121	0.744	0.920	7.979M	2.83
EfficientNetV2-S	0.842	0.969	21.458M	8.37
ViT-B/16	0.811	0.953	86.578M	17.56

Implementation of the models are gathered from TorchVision (maintainers and contributors), part of the PyTorch (Paszke et al.) project. They are instantiated with pre-trained weights obtained by training on the ImageNet-1K (Russakovsky et al.) dataset. ImageNet is a database based on the lexical database WordNet (Miller et al.; Miller) which groups words into distinct sets of concepts. Additionally, nouns and verbs are ordered hierarchically. For each node of the noun hierarchy, ImageNet provides up to thousands of images. ImageNet-1K is a subset of ImageNet narrowed down to 1000 nodes of the hierarchy.

After instantiation, models are "tuned" such that the component leading to the original output of 1000 classes is adjusted to the number of classes required by the dataset. For the models chosen in this experiment, this means adjusting the output size of a linear transformation lin, defined by equation 18.

All chosen CNNs use convolution with stride 2, i.e., a conv defined by equation 4 with $s = 2$ and $p = 0$, to increase channel size. Batch normalization, defined by equation 15, is applied to this lower-resolution output before being activated by a linear unit. The following function applications are architecture specific and presented in the following. Adaptive average pooling, defined by equation 7, is applied to the 7×7 output of those layers before the tuned linear transformation classifies the image. These general sequences of operations that these CNNs have in common are shown in Table 2.

2.3.1 ResNets

The problem of vanishing gradients (Bengio et al.; Hochreiter et al.) occurs when functions are applied to input increasingly often. It has first been identified by Hochreiter. The parameters of a neural network are optimized during training by calculating gradients of the loss with respect to the parameters of the network.

Table 2: Prelude and final operations of each CNN.

ResNet-18	ResNet-50	DenseNet-121	EfficientNetV2-S
conv1			conv2
Batch normalization			
ReLU			SiLU (eq. 9)
maxp			
Architecture specific			
adaptive avgp			
lin			

With each layer the gradient of a new function composition has to be calculated which requires applying the chain rule. Calculating the gradient of one function requires the gradient of all deeper functions. If any gradient or enough gradients of these deeper functions are too small, then multiplying with them results in a gradient approaching 0 and therefore vanishing. This prevents the neural network from learning. Furthermore, if enough gradients in the depths of the network are too big, then multiplying with them results in a gradient pointing towards infinity. This is the exploding gradient problem and makes learning unstable.

On the other hand, it has been shown, that deeper machine learning models, i.e., models with increasingly many function applications, perform better (Simonyan and Zisserman; Szegedy et al.). To tackle this problem, ideas for the initialization of models (LeCun et al., a; Glorot and Bengio; Saxe et al.; Sutskever et al.) and their components were developed (He et al., a; Ioffe and Szegedy). On an architectural level, shortcut connections were used only during training (Lee et al.; Szegedy et al.) but also kept during inference (Srivastava et al., b,a). Although the latter "highway network" yielded promising results in overcoming the gradient problem they also introduced additional parameters to be trained.

He et al. (b) introduce residual nets (ResNets) that mostly make use of identity feed-forward connections from the input of a convolution stack to their output. This does not elevate the amount of parameters and allows the network to be scaled in depth without being punished by vanishing gradients as hard as before. Input is transformed by the mentioned early layers and activated by a ReLU, defined by equation 8, before max pooling maxp, defined by equation 6, is applied. The building block making up most of architecture's layers depends on the amount of layers used.

Below 50 layers, the basic building block is a sequential application of a 3 by 3 convolution, and batch normalization, defined by equation 15, separated by a ReLU activation. To this output the initial input is directly added before a final ReLU activation finishes the basic building block. This identity mapping is what separates

ResNets from previous architectures. At the beginning of every convolutional layer a projection via a 1×1 convolution with a stride of 2 followed by batch normalization is applied instead of identity mapping. The exception to this is the first convolutional layer as its input has already been reduced by convolution common to all model architectures used for this experiment. This results in a shortcut connection after every other convolution.

For architecture variations with at least 50 layers, this basic building block is enhanced by a "bottleneck design" (He et al., b). One 3×3 convolution is surrounded by two 1×1 convolution, before the identity or projection of the original input is added. This results in a shortcut connection after every three convolutions.

Table 3: Architecture of the ResNets used for the experiments.

ResNet-18			ResNet-50		
Output		Block	Output		Block
Channels	$\sqrt{\text{Resolution}}$		Channels	$\sqrt{\text{Resolution}}$	
64	64	Basic $\times$ 2	256	64	Bottleneck $\times$ 3
128	28	Basic $\times$ 2	512	28	Bottleneck $\times$ 4
256	14	Basic $\times$ 2	1024	14	Bottleneck $\times$ 6
512	7	Basic $\times$ 2	2048	7	Bottleneck $\times$ 3

Layering of the chosen models ResNet-18 and ResNet-50 can be seen in Table 3. From this the input size of the linear classifier can be determined and it is noted that the 50-layer variant has a feature map size 4 times that of the 18-layer ResNet variant.

ResNet-18 pre-training is rather simple. Training was done for 90 epochs in batch sizes of 32. The initial learning rate of 0.1 was reduced by a factor of 0.1 every 30 epochs. Stochastic gradient descent was used with momentum (Sutskever et al.) of 0.9 and a weight decay of 0.0001.

Pre-training for ResNet-50 and other models differs (Vryniotis) which involves a considerable amount of regularization and data augmentation.

It is noted that the chosen models ResNet-18 and ResNet-50 are equivalent to ResNet-18 (224) and ResNet-50 (224) in MedMNIST v2's experiments (Yang et al., 2023).

2.3.2 DenseNet

With the advancements of ResNets (He et al., b) and Highway Networks (Srivastava et al., b,a) depth-wise scaling is more effective. However, by bypassing a random subset of layers every few samples in ResNets performance improvements can be

achieved (Huang et al., b). This stochastic depth of ResNets is taken as evidence by Huang et al. (a) that not all layers contribute to the end result. To shorten the connection between input and output of a network, they elevate the idea of ResNets' feed-forward connections by concatenating the output of a layer to its input. As a consequence information flow is facilitated through the network.

This gives rise to the dense building block. A dense block groups multiple shortcut connections and convolutions together before being processed to reduce resolution. One block is made up out of multiple dense layers which in turn applies two convolutions; the first being a 1×1 convolution and the second being a 3×3 one. Before each of these, batch normalization and ReLU activation is applied. The concatenation of the output of these layers happens in the dense block.

Downsampling the accumulated feature maps is done by the transition layer and is positioned between every dense block. It consists of batch normalization, ReLU activation, a 1×1 convolution and 2×2 average pooling with a stride of 2 which are sequentially applied in that order.

Table 4: Architecture of DenseNet-121. Dense refers to one dense layer and its multiplication implies one dense block.

Output		Block
Channels	$\sqrt{\text{Resolution}}$	
256	56	Dense $\times$ 6
128	28	Transition
512	28	Dense $\times$ 12
256	14	Transition
1024	14	Dense $\times$ 22
512	7	Transition
1024	7	Dense $\times$ 16

The chosen model's architecture for the experiment is DenseNet-121 and it is shown in Table 4. A hyperparameter for DenseNets controls the amount of feature maps that are produced by each dense layer. For DenseNet-121 this "growth rate" is 32. The final feature map size and thus also the final input size of the classifying linear layer is 1280.

The training setting of the used pre-trained DenseNet-121 is identical to the used ResNet-18.

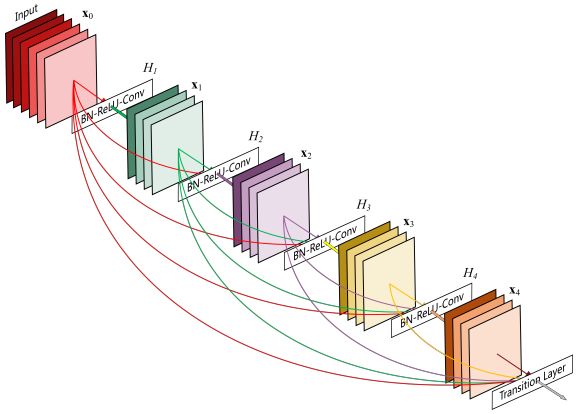


Figure 8: Huang et al. (a) present the dense building block. Residual connections exist from one layer to all subsequent layers in the block.

2.3.3 EfficientNet

Neural architecture search (Zoph and Le) automatically searches for the most optimal neural network architecture. Tan et al. apply this method to obtain MnasNet. In their search space they included variants of the traditional convolution that stray away from the naive application of its filters. Depth-wise convolutions (Sifre; Sifre and Mallat) are convolutions where each input channel uses its own set of kernel filters. With the goal of bringing convolutions to mobile devices, MobileNets (Howard et al.; Sandler et al.) use "mobile inverted bottleneck convolution". This building block is also included in the search space.

Additionally of note is, that they searched for the number of feature maps that should be put out by different stages of the architecture. This can be thought of as scaling the network architecture along the width dimension. Next to the width dimension, there is also the resolution of a network which refers to the width and height of the network's input as well as the depth which refers to the number of layers used. ResNets and DenseNets made breakthroughs by successfully scaling neural networks along the depth dimension. Some efforts were made to also scale neural networks along the other dimensions (Huang et al., c), but not all three dimensions together.

Tan and Le (a) strategically search for the most useful scaling combination along all three dimensions to obtain a family of neural networks called EfficientNets while maintaining performance and reducing parameter count. Their next version, EfficientNetV2 (Tan and Le, b) extends the search space for operations with the fused mobile inverted bottleneck convolution (Gupta and Tan; Gupta and Akin) and optimize for training speed using the same compound scaling along the different dimension of a network as before.

The model used in the experiment is EfficientNetV2-S. 24 initial features are extracted using a convolution with a kernel size of 3, stride of 2 and padding of 1. As is typical, batch normalization is applied and the output is activated by SiLU, defined by equation 9. The mobile inverted bottleneck convolution and its fused

version are the main building block of the architecture. They employ the squeeze-and-excitation technique (Hu et al., c), which recalibrates features channel-wise.

Squeeze-and-excitation calculates a scalar for each channel by reducing the input resolution to 1 via adaptive average pooling. Then, channel size is "squeezed" by some factor through convolution and then activated by a SiLU before being "excited" again back to the original channel size by a second convolution. The scalar is produced by application of the sigmoid function.

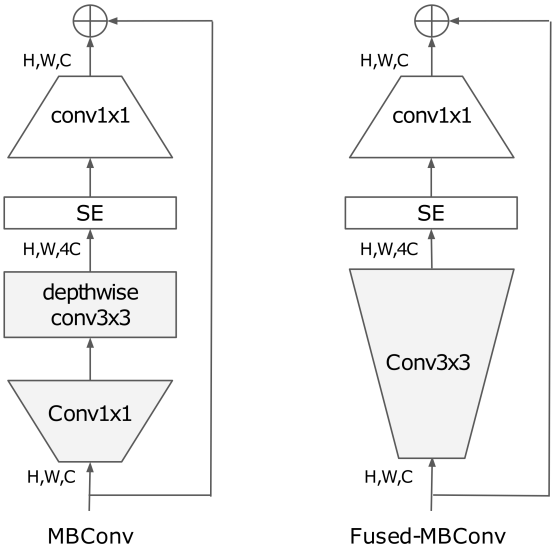


Figure 9: Mobile inverted bottleneck block in its basic (left) and fused (right) form. The expansion in channels happens through two different convolutions in the basic version while the fused variant uses only one convolution. It is noted that EfficientNets do not use the squeeze-and-excitation mechanic in fused mobile inverted bottleneck blocks. Illustration taken from Tan and Le (b).

The mobile inverted bottleneck block expands channel size via a 1×1 convolution followed by a depth-wise convolution, defined by equation 5, with kernel size 3. Both expanding convolutions are supported by batch normalization and activated by a SiLU. After squeeze-and-excitation, channel size is shrinked back down by a 1×1 convolution with batch normalization. Since it is also a residual block, the input is added to the final output. The fused variant only uses one normal 3×3 convolution with batch normalization and activation by a SiLU for expansion and this architecture also omits the squeeze-and-excitation mechanic.

These two building blocks are layered in the way described by Table 5 to obtain EfficientNetV2-S.

The training setup of the pre-trained model used for the experiment does not differ to the one used for the pre-trained ResNet-50.

Table 5: Architecture of EfficientNetV2-S

Output Channels	Output $\sqrt{\text{Resolution}}$	Block
24	112	Fused-MBConv × 2
48	56	Fused-MBConv × 4
64	28	Fused-MBConv × 4
128	14	MBConv × 6
160	14	MBConv × 9
256	7	MBConv × 15
1280	7	Final Convolution

2.3.4 Vision Transformer

The success of the attention mechanism in machine translation (Vaswani et al.) motivated its application in computer vision tasks. Several approaches were taken.

Some of these include combining the novel attention function with the classical convolution function by relating in- and output feature maps of convolutions (Wang et al., b) or processing sequences of feature maps of CNNs with attention (Carion et al.; Hu et al., a; Wu et al.). Self-attention can also be the main actor in a CNN architecture like ResNet and augmented by convolutions (Bello et al.).

There were also efforts made to fully replace convolutions with attention. The main challenge with this approach is the quadratic cost associated with the global application of attention on images. The kernel size dictates the range of local context in convolutions which can be replaced by attention. However, while exchanging all other convolutions with self-attention, not replacing the initial convolution performed best (Ramachandran et al.). This was not the case for full-attention networks where enlargening the space of included neighbouring pixels delivered seemed crucial in delivering promising results (Hu et al., b; Zhao et al.)

More greedy applications of the transformer architecture for solving image classification tasks were tried by self-attending across axes (Wang et al., a) or globally to downsampled images (Chen et al.). The sucess of attention over convolution motivated a the first naive implementation of the transformer architecture at small scale to provide evidence that attention can and in some cases do emulate convolutions (Cordonnier et al.). However, in contrast to CNNs, transformers lack the inductive biases of translational equivariance and locality. Dosovitskiy et al. speculate that by scaling architecture and the training dataset these properties can be learned. Their vision transformer (ViT) is another naive application of the transformer architecture to the computer vision task of image classification.

Instead of the 2 × 2 patches used by Cordonnier et al., they devide the input image into square image patches with a dimension size of at least 14. These patches are projected by convolutions into a sequence of multi-channel values before being concatenated with a learnable "class token" matrix. After a learnable position embedding is added to the sequence which is then transformed by encoding layers.

An encoding layer applies multi-headed-attention, defined by equation 13, to the layer-normed input introspectively. This self-attention means that the input to the attention mechanism is the same: $\boldsymbol{Q} = \boldsymbol{K} = \boldsymbol{V}$ which originate from the concatenation of the class token and the projected image patches. As a residual block, the original input of the layer is added to the attention output. This output is normalized layer-wise and processed by a multilayer perceptron (MLP).

The MLP is a fully connected network simply consisting of linear connections with hidden layers activated by a linear unit. In the case of the encoder block, the hidden layer briefly expands the feature space before being shrinked down again to the original size. The expanded features are activated by a GELU, defined by equation 10.

Like all other architectures, a final linear layer classifies the image.

The variant used for the experiment is ViT-B/16. Its patch size is 16 × 16 with 12 encoding layers. Input is attended to by 12 heads with values in 768 channels and the MLP has a hidden dimension of 3072.

The pre-training setting for ViT-B/16 is similar to the one of ResNet-50 and EfficientNetV2-S with different settings in regularization and data augmentation.

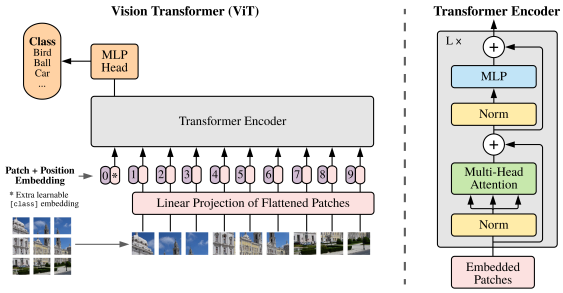


Figure 10: Illustration from Dosovitskiy et al. visualizes their vision transformer architecture.

3 Data records

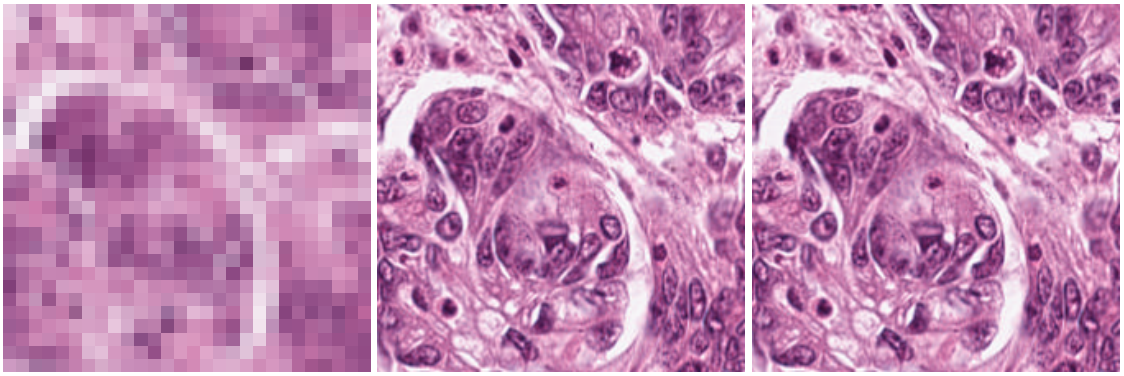
In this chapter, the enhanced versions of the MedMNIST2D datasets are presented. The are denoted by a "+" at the end.

3.1 Methods

Information about the association of a source image with a certain split were taken from MedMNIST v2 by contacting the authors. For scaling operations, bicubic interpolation was used. Following the method described in Yang et al. (2023)'s work results in the following nine enhanced MedMNIST2D datasets.

3.2 Details of enhanced MedMNIST2D (MedMNIST2D+)

3.2.1 PathMNIST+



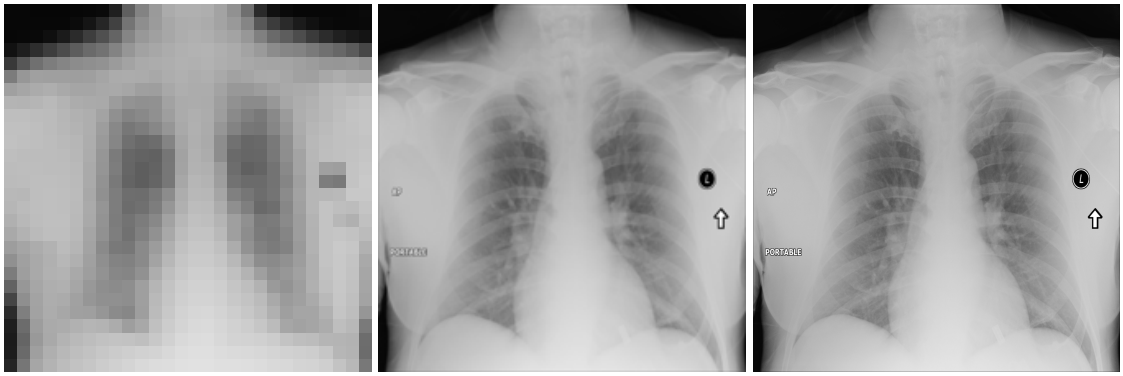
(a) PathMNIST (b) PathMNIST+ (c) Source image

Figure 11: Comparison of the original sample (c), the MedMNIST version (a) as well as the enhanced version (b) on the training image at index 47424 of the Pathology dataset.

The source dataset (Kather et al., 2018) was used by their authors in an earlier study (Kather et al., 2019). 107180 image patches were gathered from slides of human colon tissue. Next to background images, the $3 \times 224 \times 224$ images can be categorized into one of 8 types of colon tissue: Adipose, debris, lymphocytes, mucus, smooth muscle, normal colon mucosa and colorectal adenocarcinoma epithelium. Processing only entails reading and saving the images in the same position as in the PathMNIST dataset as they are already in the desired resolution for our enhanced versions. A comparison of the different versions of a sample can be seen in figure 11.

3.2.2 ChestMNIST+

By text-mining reports associated with scans of over 32,000 patients (Wang et al., 2017), construct a chest X-ray dataset of 112120 images. The grayscale 1024×1024 images of the "ChestX-ray8" dataset are resized into 224×224 pixels 12. An image can be associated with up to 14 disease labels: atelectasis, cardiomegaly, effusion, infiltration, mass, nodule, pneumonia, pneumothorax, consolidation, edema, emphysema, fibrosis, pleural and hernia.



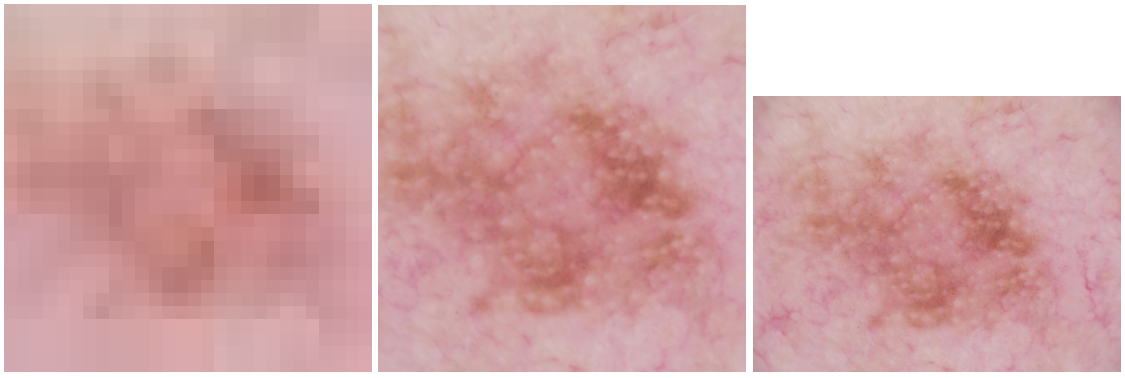
(a) ChestMNIST (b) ChestMNIST+ (c) Source image

Figure 12: The image at index 3447 of the test set. Symbols from the original image (c) can still be identified in ChestMNIST+ (b).

This is the only multi-label classification problem of the dataset collection.

3.2.3 DermaMNIST+

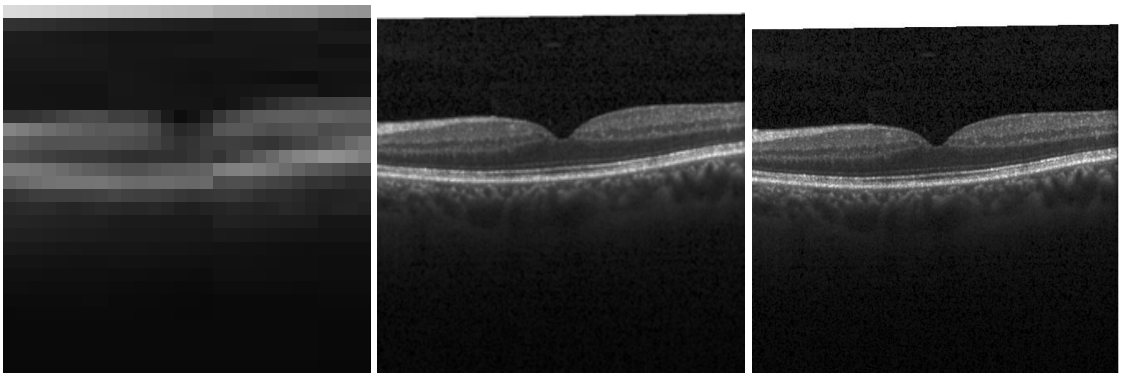
The "Human Against Machine with 10000 training images" (HAM1000) dataset (Tschandl et al., 2018; Tschandl) is a collection of dermatoscopic images from different populations. Multiple modalities are unified into standardized images to serve as representative sample of pigmented skin lesions. The dataset also contributed to a challenge hosted by the International Skin Imaging Collaboration (ISIC) (Codella et al., 2019) for image classification as well as segmentation. Each of the 10,015 dermatoscopic images can be classified as one of 7 different diseases: actinic keratoses and intraepithelial carcinoma, basal cell carcinoma, benign keratosis-like lesions, dermatofibroma, melanoma, melanocytic nevi and vascular lesions.



(a) DermaMNIST (b) DermaMNIST+ (c) Source image

Figure 13: The image at index 1133 of the training set of DermaMNIST and DermaMNIST+.

3.2.4 OCTMNIST+



(a) OCTMNIST (b) OCTMNIST+ (c) Source image

Figure 14: The shape of the train image at index 84131 is still recognizable in (b).

Kermany et al. (2018b,a) demonstrate the effectiveness of a transfer-based deep learning method for biomedical computer vision tasks on two datasets and make their datasets available for research. One of these datasets contains 108,312 retina images gathered via optical coherence tomography (OCT). Each retina image can be evidence of one of four classes: choroidal neovascularization, diabetic macular edema, drusen and normal. The grayscale images range from 384 to 1,536 pixels in height and 277 to 512 pixels in width. They are center-cropped and resized into 224 $\times$ 224.

3.2.5 PneumoniaMNIST+

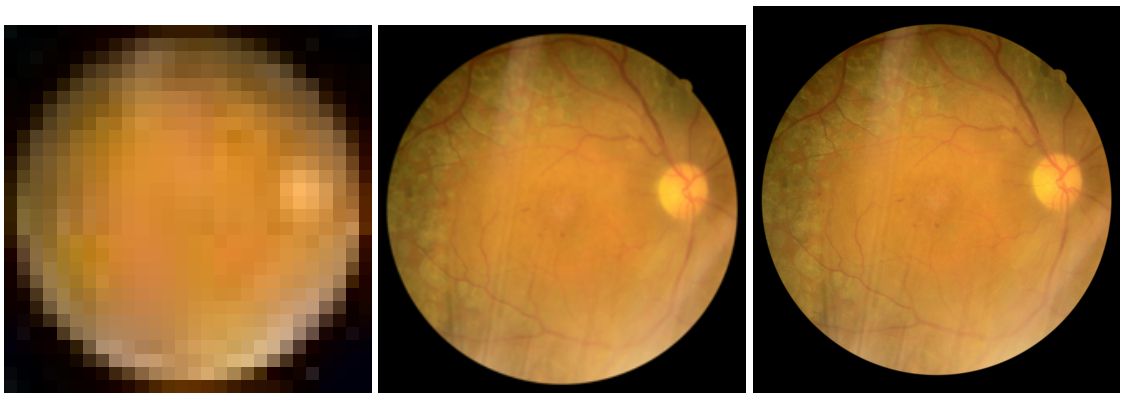
(a) PneumoniaMNIST (b) PneumoniaMNIST+ (c) Source image

Figure 15: The object seen in the source image (c) can also be seen in ChestMNIST+ (b) at index 3507 of the training split. This is not the case for PneumoniaMNIST (a).

The other set of images used (Kermany et al., 2018b) and made available (Kermany et al., 2018a) is another Chest X-Ray dataset. 5,232 grayscale images are gathered

from children possibly displaying a case of pneumonia, making this a binary classification task instead of a multi-label one. Center-cropping is again necessary for the (384 to 2,916) × (127 to 2,713) images before being resized to 224 × 224.

3.2.6 RetinaMNIST+

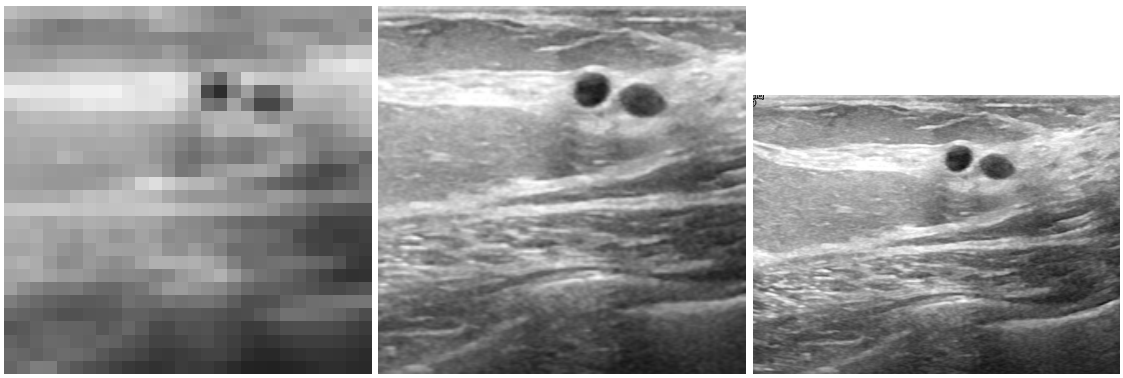


(a) RetinaMNIST (b) RetinaMNIST+ (c) Source image

Figure 16: Individual venules and arterioles are still visible in RetinaMNIST+ (b).

Retinal fundus images are used (deepdrdoc) by the DeepDRiD challenge (Liu et al., 2022) to grade the level of diabetic retinopathy. The problem of ordinal regression with 5 levels is formulated as a multi-class problem and evaluated as such. For processing the 3 × 1,736 × 1,824 source images are center-cropped and resized to 3 × 224 × 224. There are 1600 images.

3.2.7 BreastMNIST+

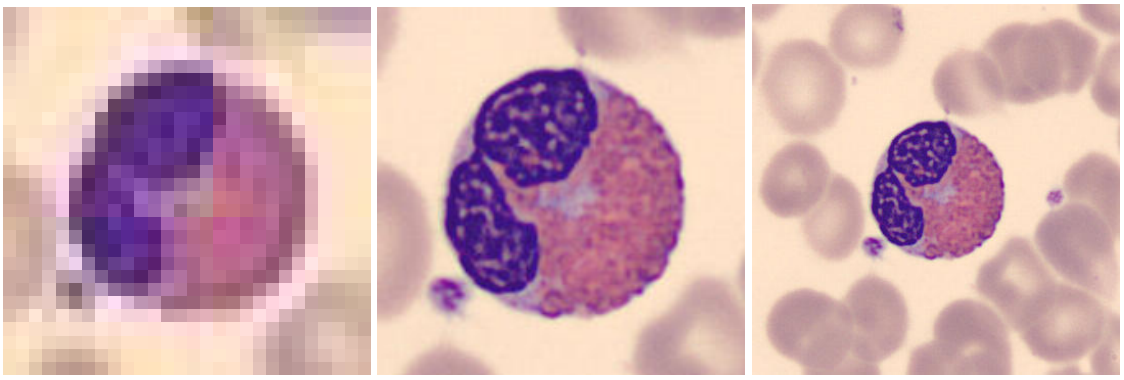


(a) BreastMNIST (b) BreastMNIST+ (c) Source image

Figure 17: Benign breast tissue is shown. From left to right: ChestMNIST, ChestMNIST+, both at index 2 of the test split, and the source image.

780 ultrasound images of women's breasts serve as the basis for this dataset (Al-Dhabyani et al., 2020). Cases of normal and benign breast cancer are combined to model the computer vision task as a binary classification of malignant breast cancer. The grayscale images of 500 pixels in width and height are downsized to 224 pixels each.

3.2.8 BloodMNIST+

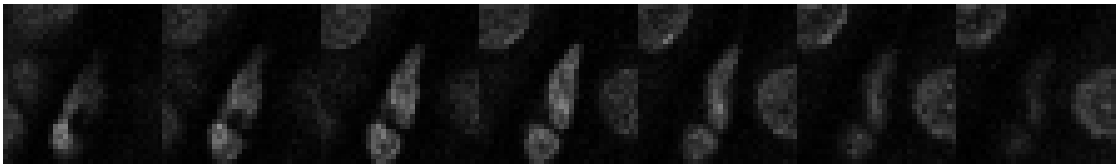


(a) BloodMNIST (b) BloodMNIST+ (c) Source image

Figure 18: Comparison of the different versions of an image showing a blood cell. At index 10819 of their training split, the ChestMNIST, ChestMNIST+ and source image are shown from left to right.

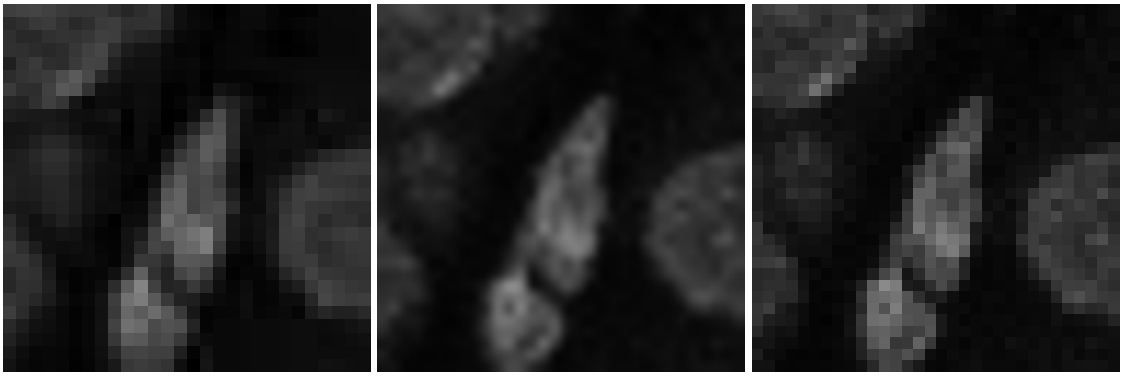
Eight different blood cell types can be classified in a dataset of 17,092 microscopic images (Acevedo et al., 2020b,a) and serve the basis for this dataset. The cell types are basophil, eosinophil, erythroblast, immature granulocytes, which includes myelocytes, metamyelocytes and promyelocytes, lymphocytes, monocyte, neutrophil and platelet. To focus on the individual cell to classify, the $3 \times 360 \times 363$ images are center-cropped using a window size of 200. The resulting $3 \times 200 \times 200$ images are then downsized to $3 \times 224 \times 224$.

3.2.9 TissueMNIST+



(a) Slice 1 (b) Slice 2 (c) Slice 3 (d) Slice 4 (e) Slice 5 (f) Slice 6 (g) Slice 7

Figure 19: Slices of the source image which can be used to create the maximum projection seen in Figure 20(c).



(a) TissueMNIST (b) TissueMNIST+ (c) Maximum projection

Figure 20: The axial-axis maximum projection of the slices shown in Figure 19 are downsampled for TissueMNIST (a) and upsampled for TissueMNIST+ (b).

Images depicting stained human kidney nuclei and their masks of a deep learning study (Woloshuk et al., 2021) are contained in the Broad Bioimage Benchmark Collection (Ljosa et al., 2012). Only the 3D $32 \times 32 \times 7$ grayscale representation of the human kidney tissue are considered. Along the axial-axis of the 7 slices the maximum pixel value is taken to yield a 2D maximum projection. Finally, this projection is upscaled to 224×224. An example of the slices and the comparison between the different versions of the maximum projection can be seen in Figure 19 and in Figure 20 respectively.

4 Experiments

Several state of the art baseline computer vision models are trained and evaluated on the enhanced versions of the MedMNIST 2D datasets. Results are reported for each model and dataset as well as averaged for each model.

4.1 Training setup

Parameters for training are taken from MedMNIST v2 (Yang et al., 2023). Models are trained for 100 epochs in which it is optimized via Adam (Kingma and Ba) with an initial learning rate of 0.001 for minimizing cross-entropy loss (Bishop and Nasrabadi). The batch size is effectively 128. If the memory of the accelerator device is not enough for a normal batch size of 128, the batch size is reduced but the frequency of an optimization step is adjusted accordingly. I.e., like Wang et al. (2017), if the batch size is to be reduced to 64, then every second batch, the optimization step is done. Learning rate is scheduled to be delayed by 0.1 at 50% and 75% of the epochs. For evaluation of a model, its outputs are activated by the softmax function (Bridle, a,b). The exception for this pose versions of the ChestMNIST dataset for which the sigmoid function (Han and Moraga) is used along the second dimension

as well as plain sigmoid followed by binary cross entropy for loss. Performance of a model is measured by the area under the receiver-operating-characteristic curve (AUC) which is calculated with the procedure provided by Yang et al. (2023). The best model is selected by its performance on the validation split after each epoch.

Loaded images are resized to the input resolution of the pre-trained model and ensured to be in RGB mode. Afterwards the image is converted to tensor and normalized with mean and standard deviation of 0.5.

In contrast to MedMNIST v2, these experiments are only run once instead of thrice.

4.2 Results

Performance is measured by the AUC and ACC measures, introduced in chapter 2.2.10. The best performing is highlighted. Results of each model on MedMNIST+ is listed in Table 6. Percentages were rounded to three digits after the decimal point except for AUC results of BloodMNIST+ which would amount to a perfect AUC otherwise. Instead, highlighted performances indicate the superiority over the other models.

Average performance of each model across all MedMNIST+ datasets are shown in Table 7.

4.3 Analysis

In the following, the results from the experiment are analysed. First, AUC and ACC are examined in isolation and briefly correlated with the dataset sample size. Then, common misclassifications of each dataset are shown and elaborated upon.

4.3.1 Comparison of performance measures

EfficientNetV2-S performs well across all enhanced datasets. It achieves the highest AUC in six of the nine datasets. However, its difference to other CNNs is not big: AUC on the enhanced BloodMNIST is the same as DenseNet-121 and at most 3.1 percentage points in the case of enhanced ChestMNIST.

DenseNet-121 is also very accurate, being the most accurate on four of the nine enhanced datasets.

ResNet-50 only achieves the highest performance in one measure on one dataset. It shares the highest ACC on the BloodMNIST+ dataset with DenseNet-121. Despite being the smaller variant, ResNet-18 the highest AUC and ACC in two cases for each measure. The best model in terms of AUC and ACC on the enhanced OCTMNIST dataset is ResNet-18. It is the most accurate model on the enhanced RetinaMNIST dataset and achieves the highest AUC on the small enhanced BreastMNIST dataset.

ViT-B/16 performs the worst. It can never outperform any of the CNNs on any enhanced dataset.

Table 6: Performance of each model on each MedMNIST2D+ dataset.

Methods	PathMNIST+		ChestMNIST+		DermaMNIST+	
	AUC	ACC	AUC	ACC	AUC	ACC
ResNet-18	0.995	0.905	0.800	0.946	0.953	0.823
ResNet-50	0.994	0.918	0.807	0.942	0.968	0.863
DenseNet-121	0.992	0.931	0.818	**0.948**	**0.979**	**0.882**
EfficientNetV2-S	**0.996**	**0.932**	**0.831**	0.947	0.976	0.859
ViT-B/16	0.984	0.884	0.712	0.947	0.940	0.784
Methods	OCTMNIST+		PneumoniaMNIST+		RetinaMNIST+	
	AUC	ACC	AUC	ACC	AUC	ACC
ResNet-18	**0.992**	**0.922**	0.975	0.814	0.849	**0.668**
ResNet-50	0.990	0.884	0.986	0.857	0.860	0.590
DenseNet-121	0.989	0.884	0.990	0.912	0.847	0.638
EfficientNetV2-S	0.990	0.900	**0.991**	**0.917**	**0.862**	0.650
ViT-B/16	0.974	0.798	0.931	0.864	0.747	0.525
Methods	BreastMNIST+		BloodMNIST+		TissueMNIST+	
	AUC	ACC	AUC	ACC	AUC	ACC
ResNet-18	**0.938**	0.872	0.999	0.988	0.939	0.699
ResNet-50	0.866	0.827	0.999	**0.990**	0.942	0.709
DenseNet-121	0.937	**0.917**	**0.999**	**0.990**	0.953	0.744
EfficientNetV2-S	0.921	0.897	**0.999**	0.988	**0.959**	**0.761**
ViT-B/16	0.686	0.609	0.999	0.975	0.919	0.652

Averaged performance of the smaller ResNet-18 is better than that of bigger ResNet-50. Furthermore, newer CNN architectures perform better when averaged across all enhanced datasets. The box plot in Figure 21 also shows that the spread in performance is less the newer the architecture is. It also contrasts the vision transformer's poor performance compared to the CNNs.

AUC is plotted against the number of samples of the enhanced dataset on which the AUC is achieved in Figure 22. It also shows a linear regression curve. There is no significant influence of sample count on the AUC for CNNS. This is not the case for the vision transformer. ViT-B/16 performs better when sample size is bigger.

Table 7: Average performance of the chosen models in measures average AUC and average ACC over all enhanced 2D datasets

Methods	AVG AUC	AVG ACC
ResNet-18	0.938	0.848
ResNet-50	0.935	0.842
DenseNet-121	0.945	**0.872**
EfficientNetV2-S	**0.947**	**0.872**
ViT-B/16	0.877	0.782

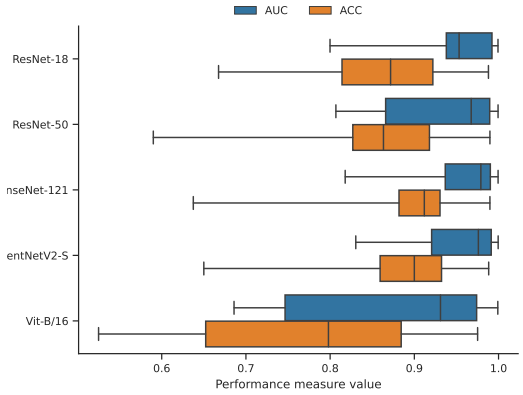


Figure 21: Box plots of both performance measures for each model. Newer CNNs perform better than older ones. The newest architecture, the vision transformer, consistently performs the worst.

Both the experiment in this work as well as the technical validation in MedMNIST v2 use ResNet variants ResNet-18 and ResNet-50. Technical validation results are gathered from three runs whereas the experiments on the enhanced datasets were run only once. Their differences per dataset are shown in Table 8.

Generally, experiments on the enhaned versions of the dataset perform marginally better. There are exceptions for DermaMNIST, OCTMNIST and RetinaMNIST. ResNet-50 becomes noticably more accurate on the first two; ResNet-18 on the last two. AUC improvements can be seen on the enhanced version of the RetinaMNIST dataset.

The averaged differences over the datasets for which enhanced versions exist are shown in Table 9. Noteworthy improvements in terms of both AUC and ACC are not as noticable when averaged over all datasets. Both models are slightly more accurate on the enhanced versions of the datasets.

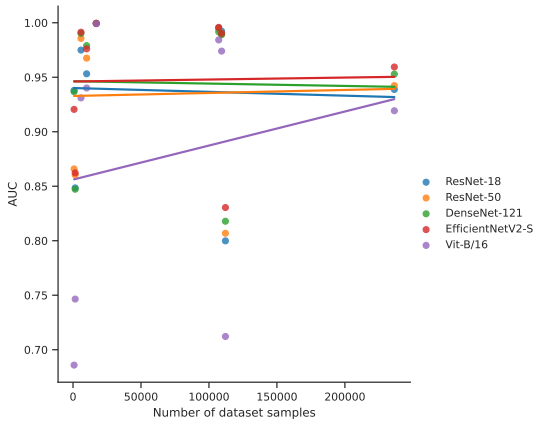


Figure 22: AUC of each model is plotted against the number of samples of the dataset. Additionally, linear regression curves are shown. There is no significant relationship between dataset size and AUC for CNNS. The vision transformer's performance scales with dataset size.

Table 8: Differences in the performance measures of the MedMNIST v2 technical validation compared to the experiment done on the enhanced versions of the datasets. ResNet-{18,50} are equivalent to ResNet-{18,50} (224) in MedMNIST v2 with the addition that their models were not pre-trained. Runs on the enhanced versions are generally marginally better with the exception of DermaMNIST, OCTMNIST and RetinaMNIST. ResNet-50 becomes more accurate. Both variants make AUC improvements on the latter dataset.

Methods	PathMNIST		ChestMNIST		DermaMNIST	
	ΔAUC	ΔACC	ΔAUC	ΔACC	ΔAUC	ΔACC
ResNet-18	-0.006	+0.004	-0.027	+0.001	-0.033	-0.069
ResNet-50	-0.005	-0.026	-0.034	+0.006	-0.056	-0.132
Methods	OCTMNIST		PneumoniaMNIST		RetinaMNIST	
	ΔAUC	ΔACC	ΔAUC	ΔACC	ΔAUC	ΔACC
ResNet-18	-0.034	-0.229	-0.019	+0.050	-0.139	-0.175
ResNet-50	-0.032	-0.108	-0.024	+0.027	-0.144	-0.079
Methods	BreastMNIST		BloodMNIST		TissueMNIST	
	ΔAUC	ΔACC	ΔAUC	ΔACC	ΔAUC	ΔACC
ResNet-18	-0.047	-0.039	-0.001	-0.025	-0.006	-0.018
ResNet-50	±0.000	+0.015	-0.002	-0.040	-0.010	-0.029

Table 9: Difference in averaged AUC and ACC of the two ResNet variants compared to the experiments done on the enhanced versions of the MedMNIST2D datasets.

Methods	ΔAVG AUC	ΔAVG ACC
ResNet-18	-0.035	-0.056
ResNet-50	-0.034	-0.041

4.3.2 Inspection of misclassified images

For every enhanced dataset, a case of the most common misclassification made by all model is inspected.

The most common misclassification made by all models for the enhanced PathMNIST dataset were instances of cancer-associated stroma. Most commonly, they were mistaken for smooth muscle. An instance showing a case of cancer-associated stroma that was misclassified by all models as smooth muscles can be seen in Figure 23.

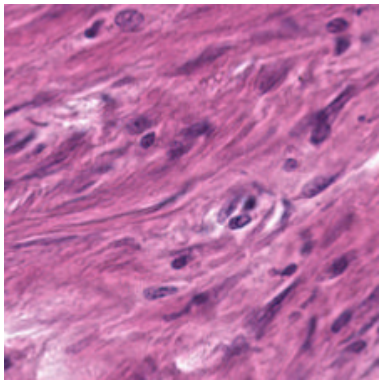


Figure 23: Misclassified image from the enhanced PathMNIST dataset. All models classified this case of cancer-associated stroma as smooth muscle.

All models struggled to classify anything in instances of infiltration in the ChestXray8 dataset. One such instance is shown in Figure 24.

For enhanced DermaMNIST, melanoma, the most serious type of skin cancer, was most frequently identified as melanomamelanocytic nevi, which is benign. A case of this is shown in Figure 25.

The most common misclassification of all models for that enhanced OCTMNIST dataset were cases of drusen, "which are lipid deposits present in the dry form of macular degeneration" (Kermany et al., 2018b). Choroidal neovascularization was the most common prediction made by the models for these cases. One such case is shown in Figure 26.

Figure 24: An x-ray scan from the enhanced ChestMNIST dataset showing a case of infiltration. No model in the experiment detected this case.

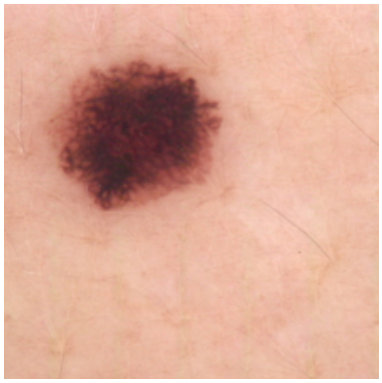


Figure 25: In the experiment, all models misclassified this case of melanoma as benign. Image is sampled from the enhanced DermaMNIST dataset.

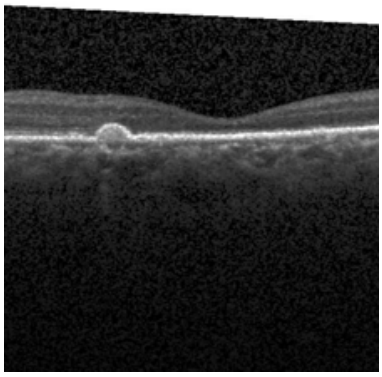


Figure 26: All models in the experiment classified this image from the enhanced OCTMNIST dataset as a case of choroidal neovascularization. However, this is a case of drusen.

Images from the enhanced PneumoniaMNIST dataset were misclassified as showing cases of pneumonia. However, all of them do not show any cases of the three possible variants of pneumonia. An image from this scenario is displayed with Figure 27.

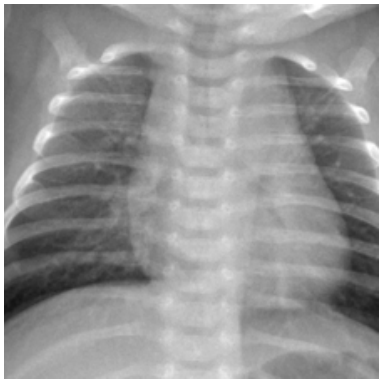


Figure 27: Image from the enhanced PneumoniaMNIST not showing a case of pneumonia. However, all models in the experiment classified it as such.

In the ordinal regression tasks associated with the RetinaMNIST dataset, models had trouble differentiating between exclusively benign cases and the first worse level of diabetic retinopathy. Figure 28 shows one such example that was wrongly identified as having the lowest level of diabetic retinopathy.

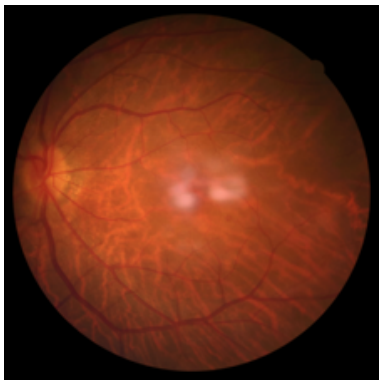


Figure 28: All models in the experiment did not ascribe the first level of diabetic retinopathy, but assigned it to the lowest level. Image is taken from the enhanced RetinaMNIST dataset.

From the 156 test images, two cases of either class exist in which the models misclassified the image as the other class. Figure 29 show the two images that were malignant but classified as normal by all models; Figure 30 show the two images that were benign but classified as malignant.

All models performed very well on the enhanced BloodMNIST dataset. Neutrophil blood cells were most commonly misclassified by all models. Except for one instance, they were classified as immature granulocytes, of which an image is shown in figure 31.

In the exception, shown in Figure 32, models "saw" an eosinophil blood cell.

Cases of distal convoluted tubule were most often misclassified by all models on the enhanced TissueMNIST dataset. Models most often mistook instances of that class

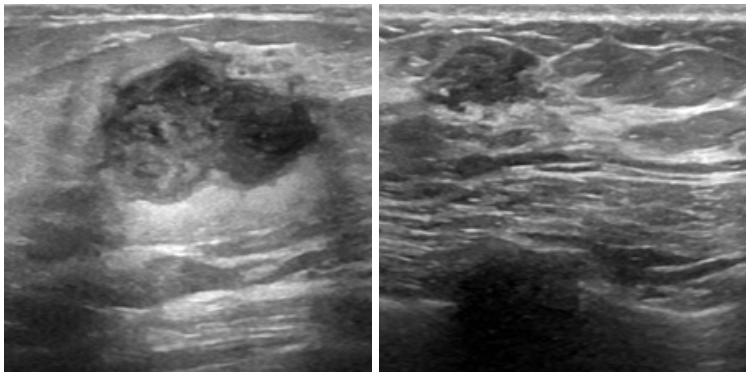


Figure 29: Two cases of malignant breast scans from the enhanced BreastMNIST dataset. All models were unable to classify these instances as such.

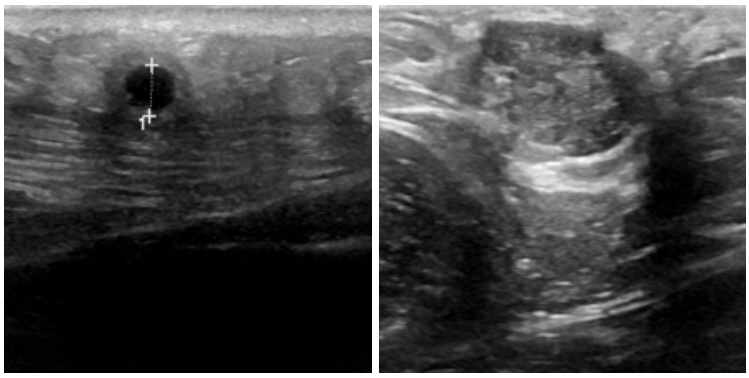


Figure 30: Two cases of normal breast scans from the enhanced BreastMNIST dataset. In the experiment, all models classified these as maligned cases.

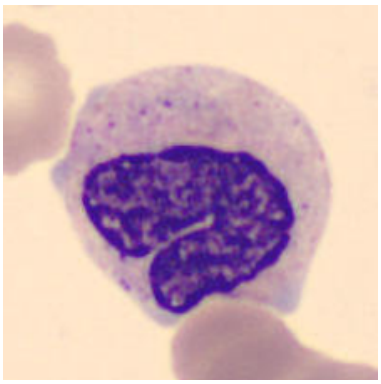


Figure 31: An image from the enhanced BloodMNIST dataset. In the experiment all models mistook this neutrophil blood cell as an immature granulocyte.

as cases of collecting duct, connecting tubule. An image showing this scenario can be seen in Figure 33.

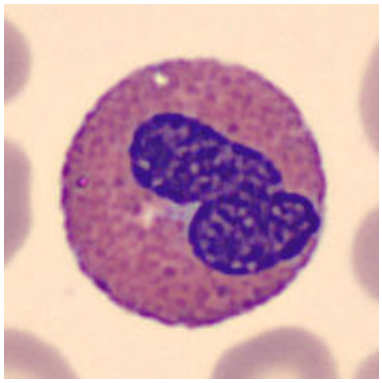


Figure 32: In the experiment, all models classified this neutrophil blood cell as an eosinophil blood cell. Image is taken from the enhanced BloodMNIST dataset.

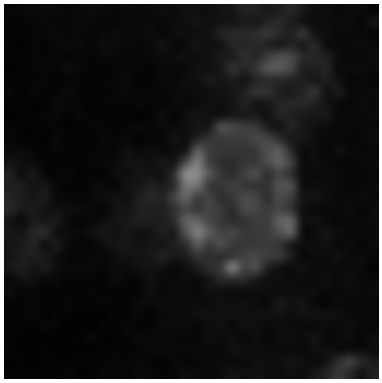


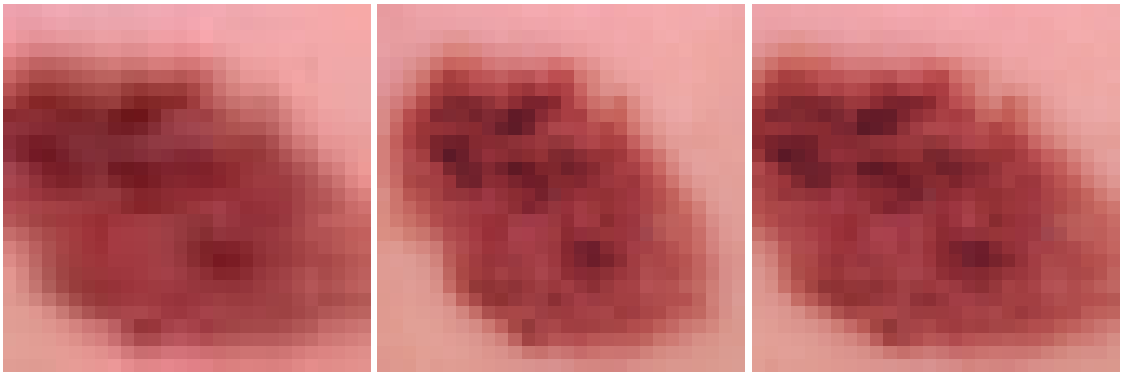
Figure 33: An image showing distal convoluted tubule that was classified by all models in the experiment as collecting duct or connecting tubule. The image is from the enhanced TissueMNIST dataset.

5 Discussion

In this work, five state-of-the-art baseline neural networks are benchmarked on biomedical 2D image classification datasets. Both dataset construction and experiment setup are inspired by the MedMNIST v2 framework (Yang et al., 2023).

Sources for all datasets are identical and treatment of the images followed the framework to the extent that it produced a replica. Moreover, the datasets used for the experiment are enhanced versions of MedMNIST2D. Instead of resizing the images to an MNIST-like size of 28×28, enhanced versions are of resolution 224×224.

For one MedMNIST2D dataset, the method diverged. Following the procedure of DermaMNIST of immediately downsizing the $3 \times 600 \times 450$ images does not yield a comparable result to DermaMNIST due to aspect ratio distortion. Hence, a center-cropping step was introduced before the downsizing step to produce a replica of the original DermaMNIST image. Figure 34 shows the images resulting from the different preprocessing settings.



(a) DermaMNIST (b) Strict (c) Replica

Figure 34: Different versions of a dermatoscopic image from the HAM1000 dataset (Tschandl et al., 2018; Tschandl). Strictly following the procedure described by the MedMNIST framework (b) does not result in a replica of the DermaMNIST version (a). Only when the source image is center-cropped before being downsized can a replica (c) be produced.

Furthermore, enhanced versions could not be obtained for Organ{A,C,S}MNIST. Although volumes of computed tomography and segmentations for the liver in the axial, coronal and sagittal views are publicly accessible (Bilic et al., 2023), the MedMNIST dataset makes use of bounding boxes for additional 10 organs provided by another study (Xu et al., 2019). These bounding boxes were not publicly obtainable. As a consequence, an enhanced version of Organ{A,C,S}MNIST is omitted, making the enhanced version of MedMNIST2D just short of being complete.

The produced datasets were experimented upon using neural networks. Five state-of-the-art baseline neural networks were chosen for this benchmark. The first two are variants of ResNet (He et al., b), which are identical to those used in the MedMNIST v2's technical validation. However, all models chosen in this work are pre-trained on the ImageNet-1k (Russakovsky et al.) dataset.

Compared to the MedMNIST v2 technical validation, ResNets became slightly more accurate. Interesting improvements were seen on the enhanced versions of DermaMNIST, OCTMNIST and RetinaMNIST. These are datasets with small, medium and large sample sizes in the context of all MedMNIST2D datasets.

Additionally, two additional CNNs (Huang et al., a; Tan and Le, b) were chosen which improved the state-of-the-art as well as the attention-mechanism-based vision transformer (Dosovitskiy et al.) which represents the current state-of-the-art.

However, the current state-of-the-art results of ViT were achieved through transfer learning from a huge datasets to smaller ones. ImageNet-1k is not considered huge. As a consequence, ViTs are not able to reproduce their success achieved by solely training on this dataset. This can already be seen in the statistics of the pre-trained models chosen for the experiments in this work. The most advanced CNN used for experimentation EfficientNetV2-S boasts better accuracy than ViT-B/16 using just shy of a quarter the number of parameters.

Performance only worsens in the context of the experiments done in this work. ViT-B/16 is not able to trump any of the CNNs; neither in AUC nor in ACC. Especially on small datasets such as the enhanced BreastMNIST dataset, ViT-B/16 achieves an AUC of only 68,6% whereas CNNs achieve at least 86,6%. On big datasets with multi-class classification tasks however, i.e., on the enhanced PathMNIST, OCTMNIST and TissueMNIST datasets, ViT-B/16's performance closes in on those of the CNNs.

Moreover, training of transformers is more complicated (Popel and Bojar; Liu et al.; Cao et al.; Han et al.). There seem to be more and different factors at play in unlocking a transformer's full potential. Overall, this benchmark should not be taken as a discreditation of ViT's state-of-the-art status. If at all, it should be seen as a limitaion of this work and as motivation for future work to develop a framework fit for benchmarking vision transformers.

Additionally, experiments were run only once. This is in contrast to MedMNIST v2, which ran their experiments three times to get more robust measurements. Increasing the amount of data also encourages more thorough statistical analysis to allow for statistically significant judgements.

Finally, this work briefly inspects misclassifications per dataset common to all models. This analysis is very surface level and would also benefit from more data to identify common misclassifications across runs which could identify insights into edge cases, wrongly labeled data or simply hard classification task instances. Furthermore, feature importance techniques such as Grad-CAM (Selvaraju et al.) attention scores (Shamshad et al.) or others (Barredo Arrieta et al.) can be employed to get a more nuanced view on the differences between evaluated models.

This work provides tools for producing biomedical image classification datasets of variable resolutions based on almost all MedMNIST2D datasets. Enhanced versions of those MedMNIST2D datasets are produced and several state-of-the-art baseline neural networks are benchmarked. Marginal improvements in AUC and ACC can be seen in general. The increase in performance is of note for two multi-class classification tasks and one ordinal regression task. Newer CNN architectures perform better. The vision transformer does not live up to its potential under the same training settings as CNNs which offers room for improvement in the development of image classification benchmarks. Its tendency to perform better on bigger datasets is confirmed. A shallow inspection on misclassifications might offer starting points for researchers with biomedical domain knowledge that want to gain insight into the shortcomings of neural networks for image classification in a biomedical context.

6 Conclusion

This work offers an introduction to image classification in a biomedical context. After motivating the application of artificial intelligence in biomedical imaging with promising successes, a technical introduction into deep learning is given. Next,

monumental baseline computer vision architectures are introduced. Based on the MedMNIST v2 (Yang et al., 2023) 2D datasets, higher resolution versions are obtained with the source material as a basis. Small pre-trained variants from the introduced architectures are benchmarked on these datasets. Analysis of the results shows marginal improvements on the enhanced versions and confirms that newer convolutional neural network architectures perform better than older ones. Furthermore, the transformers affinity for tasks with larger sample sizes is confirmed in this work. Finally, a simple and short inspection of commonly misclassified cases offers inspiration for researchers with domain knowledge. Scripts for deriving such datasets and reproducing the experiments are made available to facilitate 2D biomedical image classification.

A Appendix

A.1 Code availability

Available on GitHub.

Bibliography

Andrea Acevedo, Anna Merino, Santiago Alférez, Ángel Molina, Laura Boldú, and José Rodellar. A dataset for microscopic peripheral blood cell images for development of automatic recognition systems, June 2020a.

Andrea Acevedo, Anna Merino, Santiago Alférez, Ángel Molina, Laura Boldú, and José Rodellar. A dataset of microscopic peripheral blood cell images for development of automatic recognition systems. *Data in Brief*, 30:105474, 2020b. ISSN 2352-3409.

Walid Al-Dhabyani, Mohammed Gomaa, Hussien Khaled, and Aly Fahmy. Dataset of breast ultrasound images. *Data in Brief*, 28:104863, 2020. ISSN 2352-3409.

Michela Antonelli, Annika Reinke, Spyridon Bakas, Keyvan Farahani, Annette Kopp-Schneider, Bennett A. Landman, Geert Litjens, Bjoern Menze, Olaf Ronneberger, Ronald M. Summers, Bram van Ginneken, Michel Bilello, Patrick Bilic, Patrick F. Christ, Richard K. G. Do, Marc J. Gollub, Stephan H. Heckers, Henkjan Huisman, William R. Jarnagin, Maureen K. McHugo, Sandy Napel, Jennifer S. Golia Pernicka, Kawal Rhode, Catalina Tobon-Gomez, Eugene Vorontsov, James A. Meakin, Sebastien Ourselin, Manuel Wiesenfarth, Pablo Arbeláez, Byeonguk Bae, Sihong Chen, Laura Daza, Jianjiang Feng, Baochun He, Fabian Isensee, Yuanfeng Ji, Fucang Jia, Ildoo Kim, Klaus Maier-Hein, Dorit Merhof, Akshay Pai, Beomhee Park, Mathias Perslev, Ramin Rezaiifar, Oliver Rippel, Ignacio Sarasua, Wei Shen, Jaemin Son, Christian Wachinger, Liansheng Wang, Yan Wang, Yingda Xia, Daguang Xu, Zhanwei Xu, Yefeng Zheng, Amber L. Simpson, Lena Maier-Hein, and M. Jorge Cardoso. The medical segmentation decathlon. 13(1):4128. ISSN 2041-1723. doi: 10.1038/s41467-022-30695-9.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. 58:82–115. ISSN 1566-2535.

Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V. Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. 5(2):157–166. ISSN 1941-0093. doi: 10.1109/72.279181.

Patrick Bilic, Patrick Christ, Hongwei Bran Li, Eugene Vorontsov, Avi Ben-Cohen, Georgios Kaissis, Adi Szeskin, Colin Jacobs, Gabriel Efrain Humpire Mamani, Gabriel Chartrand, Fabian Lohöfer, Julian Walter Holch, Wieland Sommer, Felix Hofmann, Alexandre Hostettler, Naama Lev-Cohain, Michal Drozdzal,

Michal Marianne Amitai, Refael Vivanti, Jacob Sosna, Ivan Ezhov, Anjany Sekuboyina, Fernando Navarro, Florian Kofler, Johannes C. Paetzold, Suprosanna Shit, Xiaobin Hu, Jana Lipková, Markus Rempfler, Marie Piraud, Jan Kirschke, Benedikt Wiestler, Zhiheng Zhang, Christian Hülsemeyer, Marcel Beetz, Florian Ettlinger, Michela Antonelli, Woong Bae, Míriam Bellver, Lei Bi, Hao Chen, Grzegorz Chlebus, Erik B. Dam, Qi Dou, Chi-Wing Fu, Bogdan Georgescu, Xavier Giró i Nieto, Felix Gruen, Xu Han, Pheng-Ann Heng, Jürgen Hesser, Jan Hendrik Moltz, Christian Igel, Fabian Isensee, Paul Jäger, Fucang Jia, Krishna Chaitanya Kaluva, Mahendra Khened, Ildoo Kim, Jae-Hun Kim, Sungwoong Kim, Simon Kohl, Tomasz Konopczynski, Avinash Kori, Ganapathy Krishnamurthi, Fan Li, Hongchao Li, Junbo Li, Xiaomeng Li, John Lowengrub, Jun Ma, Klaus Maier-Hein, Kevis-Kokitsi Maninis, Hans Meine, Dorit Merhof, Akshay Pai, Mathias Perslev, Jens Petersen, Jordi Pont-Tuset, Jin Qi, Xiaojuan Qi, Oliver Rippel, Karsten Roth, Ignacio Sarasua, Andrea Schenk, Zengming Shen, Jordi Torres, Christian Wachinger, Chunliang Wang, Leon Weninger, Jianrong Wu, Daguang Xu, Xiaoping Yang, Simon Chun-Ho Yu, Yading Yuan, Miao Yue, Liping Zhang, Jorge Cardoso, Spyridon Bakas, Rickmer Braren, Volker Heinemann, Christopher Pal, An Tang, Samuel Kadoury, Luc Soler, Bram van Ginneken, Hayit Greenspan, Leo Joskowicz, and Bjoern Menze. The liver tumor segmentation benchmark (lits). *Medical Image Analysis*, 84:102680, 2023. ISSN 1361-8415.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer.

Ludwig Boltzmann. Studien über das gleichgewicht der lebendigen kraft zwischen bewegten materiellen punkten. 58:517–560.

John Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, a.

John S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Françoise Fogelman Soulié and Jeanny Hérault, editors, *Neurocomputing*, pages 227–236. Springer Berlin Heidelberg, b. ISBN 978-3-642-76153-9.

Yun-Hao Cao, Hao Yu, and Jianxin Wu. Training vision transformers with only 2040 images. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 220–237. Springer Nature Switzerland. ISBN 978-3-031-19806-9.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 213–229. Springer International Publishing. ISBN 978-3-030-58452-8.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1691–1703. PMLR.

Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers.

deepdrdoc. deepdrdoc/DeepDRiD: The final version of DeepDRiD dataset.

Sanchari Dhar and Lior Shamir. Evaluation of the benchmark datasets for testing the efficacy of deep convolutional neural networks. 5(3):92–101. ISSN 2468-502X.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale.

European Society of Radiology (ESR). Medical imaging in personalised medicine: a white paper of the research committee of the european society of radiology (esr). 6(2):141–155. ISSN 1869-4101. doi: 10.1007/s13244-015-0394-0.

Kunihiko Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. 5(4):322–333, a.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. 36:193–202, b.

Angela Giardino, Supriya Gupta, Emmi Olson, Karla Sepulveda, Leon Lenchik, Jana Ivanidze, Rebecca Rakow-Penner, Midhir J. Patel, Rathan M. Subramaniam, and Dhakshinamoorthy Ganeshan. Role of imaging in the era of precision medicine. 24(5):639–649. ISSN 1076-6332.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256. PMLR.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Benjamin Graham. Fractional max-pooling.

Suyog Gupta and Berkin Akin. Accelerator-aware neural network design using automl.

Suyog Gupta and Mingxing Tan. Efficientnet-edgetpu: Creating accelerator-optimized neural networks with automl.

Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In José Mira and Francisco Sandoval, editors, *From Natural to Artificial Neural Computation*, pages 195–201. Springer Berlin Heidelberg. ISBN 978-3-540-49288-7.

Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. A survey on vision transformer. 45(1):87–110. ISSN 1939-3539. doi: 10.1109/TPAMI.2022.3152247.

J A Hanley and B J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. 143(1):29–36. doi: 10.1148/radiology.143.1.7063747. PMID: 7063747.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, b.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus).

G Hinton, Alexander Waibel, et al. Phoneme recognition using time-delay neural network. 37:328–339, a.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, b.

Sepp Hochreiter. Untersuchen zu dynamischen neuronalen Netzen. 91(1):31.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications.

Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, a.

Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, b.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, c.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, a.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 646–661. Springer International Publishing, b. ISBN 978-3-319-46493-0.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, and zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., c.

David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. 148(3):574, a.

David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. 195(1):215–243, b.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456. PMLR.

Jakob Nikolas Kather, Niels Halama, and Alexander Marx. 100,000 histological images of human colorectal cancer and healthy tissue, April 2018.

Jakob Nikolas Kather, Johannes Krisam, Pornpimol Charoentong, Tom Luedde, Esther Herpel, Cleo-Aron Weis, Timo Gaiser, Alexander Marx, Nektarios A. Valous, Dyke Ferber, Lina Jansen, Constantino Carlos Reyes-Aldasoro, Inka Zörnig, Dirk Jäger, Hermann Brenner, Jenny Chang-Claude, Michael Hoffmeister, and Niels Halama. Predicting survival from colorectal cancer histology slides using deep learning: A retrospective multicenter study. *PLOS Medicine*, 16(1):1–22, 01 2019. doi: 10.1371/journal.pmed.1002730.

Daniel Kermany, Kang Zhang, and Michael Goldbaum. Large Dataset of Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images, June 2018a.

Daniel S. Kermany, Michael Goldbaum, Wenjia Cai, Carolina C.S. Valentim, Huiying Liang, Sally L. Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, Justin Dong, Made K. Prasadha, Jacqueline Pei, Magdalene Y.L. Ting, Jie Zhu, Christina Li, Sierra Hewett, Jason Dong, Ian Ziyar, Alexander Shi, Runze Zhang, Lianghong Zheng, Rui Hou, William Shi, Xin Fu, Yaou Duan, Viet A.N. Huu, Cindy Wen, Edward D. Zhang, Charlotte L. Zhang, Oulan Li, Xiaobo Wang, Michael A. Singer, Xiaodong Sun, Jie Xu, Ali Tafreshi, M. Anthony Lewis, Huimin Xia, and Kang Zhang. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131.e9, 2018b. ISSN 0092-8674.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization.

Aaron Klein and Frank Hutter. Tabular benchmarks for joint architecture and hyperparameter optimization.

Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus Robert Müller. *Efficient BackProp*, pages 9–50. Springer Berlin Heidelberg, a. ISBN 978-3-540-49430-0. doi: 10.1007/3-540-49430-8_2.

Yann LeCun, Lawrence D Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Urs A Muller, Eduard Sackinger, Patrice Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. 261(276):2, b.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-Supervised Nets. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 562–570. PMLR.

Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: Analysis, applications, and prospects. 33(12): 6999–7019. doi: 10.1109/TNNLS.2021.3084827.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers.

Ruhan Liu, Xiangning Wang, Qiang Wu, Ling Dai, Xi Fang, Tao Yan, Jaemin Son, Shiqi Tang, Jiang Li, Zijian Gao, Adrian Galdran, J.M. Poorneshwaran, Hao Liu, Jie Wang, Yerui Chen, Prasanna Porwal, Gavin Siew Wei Tan, Xiaokang Yang, Chao Dai, Haitao Song, Mingang Chen, Huating Li, Weiping Jia, Dinggang Shen, Bin Sheng, and Ping Zhang. Deepdrid: Diabetic retinopathy—grading and image quality estimation challenge. *Patterns*, page 100512, 2022. ISSN 2666-3899.

Vebjorn Ljosa, Katherine L. Sokolnicki, and Anne E. Carpenter. Annotated high-throughput microscopy image sets for validation. *Nature Methods*, 9(7):637–637, 2012. ISSN 1548-7105. doi: 10.1038/nmeth.2083.

Andreea Roxana Luca, Tudor Florin Ursuleanu, Liliana Gheorghe, Roxana Grigorovici, Stefan Iancu, Maria Hlusneac, and Alexandru Grigorovici. Impact of quality, type and volume of data used by deep learning models in the analysis of medical images. 29:100911. ISSN 2352-9148.

Robert Duncan Luce. *Individual Choice Behavior: A Theoretical Analysis*. Wiley NY.

TorchVision maintainers and contributors. TorchVision: PyTorch's Computer Vision library. URL `https://github.com/pytorch/vision`.

George A. Miller. Wordnet: A lexical database for english. 38(11):39–41. ISSN 0001-0782. doi: 10.1145/219717.219748.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet: An On-line Lexical Database*. 3(4): 235–244. ISSN 0950-3846. doi: 10.1093/ijl/3.4.235.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Nancy A. Obuchowski. Receiver operating characteristic curves and their use in radiology. 229(1):3–8. doi: 10.1148/radiol.2291010898. PMID: 14519861.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Margaret Sullivan Pepe. *The statistical evaluation of medical tests for classification and prediction*. Oxford university press.

Martin Popel and Ondř ej Bojar. Training tips for the transformer model. 110(1): 43–70. doi: 10.2478/pralin-2018-0002.

Muralikrishna Puttagunta and S. Ravi. Medical image analysis based on deep learning approach. 80(16):24365–24398. ISSN 1573-7721. doi: 10.1007/s11042-021-10707-4.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. 32.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. 115(3):211–252. doi: 10.1007/s11263-015-0816-y.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. 128(2):336–359. doi: 10.1007/s11263-019-01228-7.

Fahad Shamshad, Salman Khan, Syed Waqas Zamir, Muhammad Haris Khan, Munawar Hayat, Fahad Shahbaz Khan, and Huazhu Fu. Transformers in medical imaging: A survey. 88:102802. ISSN 1361-8415.

Laurent Sifre. Rigid-motion scattering for image classification.

Laurent Sifre and Stéphane Mallat. Rigid-motion scattering for texture classification.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition.

Amber L. Simpson, Michela Antonelli, Spyridon Bakas, Michel Bilello, Keyvan Farahani, Bram van Ginneken, Annette Kopp-Schneider, Bennett A. Landman, Geert Litjens, Bjoern Menze, Olaf Ronneberger, Ronald M. Summers, Patrick Bilic, Patrick F. Christ, Richard K. G. Do, Marc Gollub, Jennifer Golia-Pernicka, Stephan H. Heckers, William R. Jarnagin, Maureen K. McHugo, Sandy Napel, Eugene Vorontsov, Lena Maier-Hein, and M. Jorge Cardoso. A large annotated medical image dataset for the development and evaluation of segmentation algorithms.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., a.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. abs/1505.00387, b.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147. PMLR.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, a.

Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10096–10106. PMLR, b.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Philipp Tschandl. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions.

Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific Data*, 5(1):180161, 2018. ISSN 2052-4463. doi: 10.1038/sdata.2018.161.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

P-F Verhulst. *Deuxième mémoire sur la loi d'accroissement de la population*, volume 269. Hayez, a.

Pierre-François Verhulst. Notice sur la loi que la population suit dans son accroissement. 10:113–129, b.

Pierre François Verhulst. Resherches mathematiques sur la loi d'accroissement de la population. 18:1–41, c.

Vasalis Vryniotis. How to train state-of-the-art models using torchvision's latest primitives. URL `https://web.archive.org/web/20230616172433/https://pytorch.org/blog/how-to-train-state-of-the-art-models-using-torchvision-latest-primitives/`.

Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European conference on computer vision*, pages 108–126. Springer, a.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, b.

Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M. Summers. ChestX-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jul 2017. doi: 10.1109/cvpr.2017.369.

Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, c.

Andre Woloshuk, Suraj Khochare, Aljohara F. Almulhim, Andrew T. McNutt, Dawson Dean, Daria Barwinska, Michael J. Ferkowicz, Michael T. Eadon, Katherine J. Kelly, Kenneth W. Dunn, Mohammad A. Hasan, Tarek M. El-Achkar, and Seth Winfree. In situ classification of cell types in human kidney tissue using 3d nuclear staining. *Cytometry Part A*, 99(7):707–721, 2021.

Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Haonan Xiao, Xinzhi Teng, Chenyang Liu, Tian Li, Ge Ren, Ruijie Yang, Dinggang Shen, and Jing Cai. A review of deep learning-based three-dimensional medical image registration methods. 11(12):4895. doi: 10.21037/qims-21-175.

Xuanang Xu, Fugen Zhou, Bo Liu, Dongshan Fu, and Xiangzhi Bai. Efficient multiple organ localization in ct image using 3d region proposal network. *IEEE Transactions on Medical Imaging*, 38(8):1885–1898, 2019. doi: 10.1109/TMI.2019.2894854.

Kouichi Yamaguchi, Kenji Sakamoto, Toshio Akabane, and Yoshiji Fujimoto. A neural network for speaker-independent isolated word recognition. In *ICSLP*.

Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):41, 2023.

Richard Zhang. Making convolutional networks shift-invariant again. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7324–7334. PMLR.

Wei Zhang, Jun Tanida, Kazuyoshi Itoh, and Yoshiki Ichioka. Shift-invariant pattern recognition neural network and its optical architecture. In *Proceedings of annual conference of the Japan Society of Applied Physics*, volume 564. Montreal, CA.

Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

S. Kevin Zhou, Hayit Greenspan, Christos Davatzikos, James S. Duncan, Bram Van Ginneken, Anant Madabhushi, Jerry L. Prince, Daniel Rueckert, and Ronald M. Summers. A review of deep learning in medical imaging: Imaging traits, technology trends, case studies with progress highlights, and future promises. 109(5):820–838. ISSN 1558-2256. doi: 10.1109/JPROC.2021.3054390.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning.

M H Zweig and G Campbell. Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine. 39(4):561–577. ISSN 0009-9147. doi: 10.1093/clinchem/39.4.561.

Declaration of Authorship

Ich erkläre hiermit gemäß §9 Abs. 12 APO, dass ich die vorstehende Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Des Weiteren erkläre ich, dass die digitale Fassung der gedruckten Ausfertigung der Abschlussarbeit ausnahmslos in Inhalt und Wortlaut entspricht und zur Kenntnis genommen wurde, dass diese digitale Fassung einer durch Software unterstützten, anonymisierten Prüfung auf Plagiate unterzogen werden kann.

Place, Date	Signature