Designing a Benchmark and Leaderboard System for Assessing the Generalizability of Deep Learning Approaches for Medical Image Classification using the MedMNIST+ Dataset Collection

Bachelor Thesis

Bachelor of Science in Applied Computer Science

Marius Ludwig Bachmeier

October 28, 2024

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

The application of Deep Learning methods to solve problems in the biomedical domain has surged. This can be attributed to the increased provision of publicly available datasets from the biomedical domain. By now image classification, image segmentation and even Natural Language processing on medical texts address biomedical issues.
Over the years, several benchmarks have been proposed to establish a standardized, regulated framework to foster scientific competition and advance the field of Deep Learning while tackling innovative tasks. Thus, this thesis introduces a new benchmark focused on image classification in the biomedical domain.
As more and more medical datasets become accessible to people with limited domain knowledge, researchers become more open to working on these issues which historically had been left for experts in medical issues. The publication of MedMNIST+, a collection of biomedical datasets, opens the door for those with limited medical knowledge but great technical expertise to work with this biomedical data.
The Biomedical Image Generalization benchmark is introduced in this thesis to seize the opportunity MedMNIST+ offers — establishing a benchmark in order to investigate the generalization capabilities of Deep Learning models on biomedical datasets. Furthermore, a website is described which can host a challenge based on the Biomedical Image Generalization.
Lastly, a training method focused on the generalization problem is investigated. In particular, a comparison is drawn between Vision Transformers and Convolutional Neural Networks. Part of these models are also used to provide a baseline for the proposed benchmark.
All of the custom code written for this thesis can be found at a GitHub repository[1].

[1] `https://github.com/mariusbachmeier/Bachelor-Thesis`

Abstract

Durch die Bereitstellung öffentlich zugänglicher Datensätze aus der biomedizinischen Domäne stieg die Anwendung von Deep Learning Methoden auf biomedizinische Fragestellungen rasant an. Mittlerweile werden bereits Bildklassifizierungsprobleme, Bildsegmentierungsprobleme aber auch Fragestellungen rund um das Verarbeiten natürlicher Sprache aus medizinischen Texten angegangen.
Über die Jahre hinweg wurden einige Benchmarks vorgeschlagen, die einen standardisierten, regulierten Rahmen zur Förderung des wissenschaftlichen Wettbewerbs bereitstellen und so Deep Learning durch das Lösen ergiebiger Fragestellungen voranbringen wollten. Daher führt diese Bachelorarbeit einen neuartigen Benchmark ein, der die Bildklassifizierung in der biomedizinischen Domäne anspricht.
Auch medizinische Laien können durch die Veröffentlichung leicht zugänglicher und handhabbarer medizinischer Datensätze ihre technische Expertise einbringen, um Probleme in dieser sonst abgeschotteten Domäne zu lösen. Hierzu wurde die Sammlung an Datensätzen namens MedMNIST+ veröffentlicht. Diese beinhaltet viele verschiedene Datensätze aus der biomedizinischen Domäne. Das Arbeiten mit diesen Bilddaten erfordert kein spezifisches medizinisches Domänenwissen.
Der Biomedical Image Generalization Benchmark stellt einen der entscheidenden Beiträge dieser Abschlussarbeit dar. Durch ihn soll ein Rahmen geschaffen werden, innerhalb dessen die Generalisierungsfähigkeit von Deep Learning-Modellen im biomedizinischen Bereich untersucht werden kann. Außerdem wurde eine Webseite eigens für ein mögliches Austragen eines wissenschaftlichen Wettbewerbs erstellt. Dieser Wettbewerb fußt auf dem Biomedical Image Generalization Benchmark. Letztlich werden Modelle basierend auf einer Methode, die sich insbesondere mit der Generalisierungsfähigkeit von Deep Learning-Modellen befasst, trainiert werden. Die Auswertung dieser Modelle fließt in die Bereitstellung von Ausgangswerten für den wissenschaftlichen Wettbewerb ein.

Acknowledgements

I want to thank Professor Ledig for accepting my thesis at the chair, my supervisor Sebastian for staying in long meetings despite already going well past the assigned time, my family for supporting me throughout my studies and of course God, for everything.
Ich möchte mich bei Professor Ledig für das Annehmen meiner Bachelorarbeit am Lehrstuhl bedanken, bei meinem Betreuer Sebastian für die Geduld in den Meetings die häufig länger als angesetzt waren, bei meiner Familie für die Unterstützung über meine Studienzeit hinweg und bei Gott, für alles.

Contents

List of Figures

List of Tables

List of acronyms

DL Deep Learning

ViT Vision Transformer

CNN Convolutional Neural Network

DA Domain Adaptation

DG Domain Generalization

VDD Visual Domain Decathlon

MSD Medical Segmentation Decathlon

CT Computed Tomography

BIG Biomedical Image Generalization

TP True Positive

FP False Positive

TN True Negative

FN False Negative

ACC Accuracy

ERR Error Rate

BALACC Balanced Accuracy

AUC Area under the ROC Curve

ROC Receiver Operating Characteristic

TPR True Positive Rate

FPR False Positive Rate

CO Cohen's Kappa

ML Machine Learning

RL Representation Learning

ANN Artificial Neural Network

SA self-attention

MSA multiheaded self-attention

MLP multilayer-perceptron

ILSVRC ImageNet Large-Scale Visual Recognition Challenge

ReLU Rectified Linear Unit

MIM Masked Image Modeling

VGG Visual Geometry Group

ResNet Residual Network

DenseNet Dense Convolutional Network

CLIP Contrastive Language-Image Pre-training

TrV Transform Vision

GELU Gaussian Error Linear Unit

RoPE Rotary Position Embedding

BiT Big Transfer

SiLU Sigmoid Linear Unit

SAM Segment Anything Model

mm-PT multi-domain multi-task pre-training

1 Introduction

1.1 Motivation

The boom in Deep Learning (DL) has opened up the possibility to employ its algorithms in the medical domain. As Deep Learning took off with the success of AlexNet in the 2012 ImageNet Large Scale Visual Recognition Challenge (Krizhevsky et al., 2012), the application to the medical domain followed soon. Around 2015 to 2016, the number of papers published which were concerned with deep learning for medical applications took off (Litjens et al., 2017). As medical datasets become available to a wider audience with less domain knowledge, the field of Deep Learning in the meantime enjoys further advances with the advent of the Vision Transformer (ViT) architecture (Dosovitskiy et al., 2020). Since ViTs have a different architecture than traditional Convolutional Neural Networks (CNNs) and are subject to different underlying assumptions and biases, these models based on the novel Vision Transformer architecture represent an exciting new avenue for research to be conducted.
At the same time, the corpus of available medical datasets has tremendously expanded. Paradoxically, although medical institutions collect vast amounts of data, publicly available annotated data in the medical domain is hard to come by. One reason for this is that labeled medical data is much more scarce compared to unlabeled medical data (Litjens et al., 2017). This is in part due to the high cost and difficulty (Lundervold and Lundervold, 2019) associated with medical professionals or experts labeling the data. Another reason is privacy issues, where the regulations of a country and thus also the concerns around the usage of the data might vary. Nonetheless, the notoriously void landscape of medical datasets has transformed over the years, with new datasets popping up and old ones expanding. Perhaps the most prominent case is the introduction of the MedMNIST+ collection of datasets, which expands the MedMNIST v2 collection of datasets (Yang et al., 2023) by providing the images not just in resolutions of 28×28 but also in resolutions of 64×64, 128×128 and 224×224. Previously, the MedMNIST Classification Decathlon Yang et al. (2021) was organized as a challenge to work on a collection of biomedical images. It was based on an older, less extensive version, the MedMNIST v1 dataset collection. The expansion of biomedical datasets makes challenges such as the MedMNIST Classification Decathlon possible, where results are compared against a baseline and other participants, in the fashion of a benchmark.
Benchmarks establish a regulated, standardized way to compare model performances. They provide a level playing field for friendly competition which has the potential to spark innovation to a great extent. Benchmarks aim for comparable, standardized performance measures with which an evaluation is conducted. Generally, benchmarks provide a baseline against which the performance of a submitted model is evaluated. With the MedMNIST Classification Decathlon, the spotlight is shone on the ability of models to generalize to other domains.
The advent of the Vision Transformer architecture and the novel MedMNIST+ collection of datasets demand not just a reevaluation but also the development of novel algorithms to take advantage of the aforementioned innovations.

1.2 Contribution

There is the exciting opportunity to make use of the multiple resolutions provided in the MedMNIST+ collection of datasets to evaluate a model's performance as theoretically the information captured in greater image resolutions increases. Furthermore, the diverse datasets in MedMNIST+ make it possible to examine how well a model can perform over multiple medical domains. A comparison of the standard CNN architecture versus the newer ViT architecture, especially as it pertains to medical applications, is subject to research as well. This thesis aims to build on these opportunities. Thus, the contribution of this thesis can be divided into three distinct yet related parts. Firstly, the Biomedical Image Generalization (BIG) benchmark using the MedMNIST+ collection of datasets is detailed. Secondly, a website has been developed with which a challenge based on the proposed benchmark could be hosted. Thirdly, the training of a model on the union of MedMNIST+ datasets is described. This model represents a baseline for the benchmark this thesis contributes. The theoretical foundation as well as various benchmarks which can be seen as predecessors to the BIG benchmark will be given in the following section 1.3. For the contribution of the baseline model, some theory is given in section 2. Aspects like the use of specific evaluation metrics or cheating prevention measures for the benchmark will be picked up on later in section 3. The website for hosting a possible challenge for the BIG benchmark is briefly touched on in section 3 as well. The experiments in which the baseline was established are then described in section 4.

1.3 Related Works

1.3.1 Preliminaries

Domain Adaptation Daumé III (2009) describes Domain Adaptation (DA) as the development of learning algorithms which can be easily ported across domains. The author gives an example for Natural Language Processing, where a model trained on newswire is adapted to biomedical documents. This concept of domain adaptation implies that the model is trained on (at least) both a source and a target domain. The concept of DA might be defined differently throughout the literature but this thesis sticks to the given general definition and doesn't introduce intricacies for reasons of practicality.

Domain Generalization In Domain Generalization (DG) no further updates of the model take place after the initial training, as per Li et al. (2018). So the model is evaluated on one or more target domains after having been trained on at least one source domain. The underlying thought and to some extent assumption is that it's possible for a model to extract visual primitives which the model can learn using the source domain and which then can be used to perform well on the target domain(s). The capability of a model to extract such domain agnostic features determines how well it performs in regards of DG.

Multiple-domain learning Multiple-domain learning is described by Rebuffi et al. (2017) as the performance of feature abstractors in several different image domains. So the capability to learn universal representations, thus leading to a better performance across several different image domains, resembles better multiple-domain learning performance. Examples for different image domains could be natural images taken outside with a photo camera, medical images from e.g. scans or handwritten characters or digits which often have uniform backgrounds and are only black/white or just in grayscale as opposed to colorful natural images.

1.3.2 Benchmarks

Distinction Benchmark and Challenge A distinction between benchmark and challenge can and will be given here. However, for the purpose of this thesis, those two terms can be assumed to be roughly equivalent, as creating a benchmark usually also entails hosting a challenge for participants to compete in. This is where usually the distinction between benchmark and challenge can be made. A benchmark in the context of image classification determines a model's performance given a set of evaluation metrics on some specified data. A challenge on the other hand is more so the ruleset for a competition built around the benchmark. So a challenge would include making submissions according to some rules such that a comparison can be drawn between the performance of submissions on the benchmark. A challenge necessitates that a benchmark exists to host such a challenge that's based on the benchmark, so in the context of this thesis the two terms can be used somewhat interchangeably.

Visual Domain Decathlon The objective of the Visual Domain Decathlon (VDD) is to provide a benchmark for the evaluation of the capability of an algorithm to learn to perform well in multiple domains at the same time, as stated by Rebuffi et al. (2017). Multiple-domain learning was detailed in section 1.3.1. The VDD includes an evaluation of a model's performance over ten representative visual domains to determine the multiple-domain learning capability of a model. There's an enumeration of the various domains and datasets given in table 7. The domains range from handwritten digits in grayscale to RGB images of flowers or other natural images. For the evaluation a bespoke metric was devised, given in equation 1.

$$S = \sum_{d=1}^{10} \alpha_d \max\{0, E_d^{\max} - E_d\}^{\gamma_d}, \quad E_d = \frac{1}{|\mathcal{D}_d^{\text{test}}|} \sum_{(x,y)\in\mathcal{D}_d^{\text{test}}} \mathbf{1}_{\{y\neq\Phi(x,d)\}} \tag{1}$$

These equations implicate that each dataset $\mathcal{D}_d, d = 1, \ldots, 10$ is formed of pairs $(x, y) \in \mathcal{D}_d$ with x being an image and y a label such that $y \in \{1, \ldots, C_d\} = \mathcal{Y}_d$. The goal is stated to be to train the best possible model to address all ten classification tasks using only the provided training and validation data, with no external data being allowed. The evaluation metric is the single scalar score S, with the authors noting that a good performance in this metric would require the model to perform well across all tasks. E_d denotes the average test error for each domain, whereas

$E_d^{\max}$ is the error of a baseline given by Rebuffi et al. (2017). The exponent γ_d then is set to two for all domains, as it rewards more reductions of the classification error. The authors give α_d as $1{,}000\,(E_d^{\max})^{-\gamma_d}$. That way a perfect result for one dataset of the ten VDD datasets would receive a score of 1000. Given that there are ten datasets, a perfect score across all datasets would thus be a score of 10,000. In order to make a submission, participants can upload a results.json file containing the classifications. The results.json file should be containing one annotation for each test image for all ten domains. A devkit[2] (VDD) is provided which contains code, annotations and one TAR archive with the images of the entire datasets. An evaluation code example is given in this devkit, too. Moreover, a competition was hosted from May 1, 2017 to July 10, 2017 and submissions are still possible past this date. Submission limits can be viewed at CodaLab[3] (Cod), where the submission process is hosted. A maximum of three submissions can be made daily and in total 100 can't be exceeded.

Medical Segmentation Decathlon The Medical Segmentation Decathlon (MSD) was presented by Antonelli et al. (2022) and is described as a biomedical image analysis challenge which is geared towards evaluating the generalization capabilities of models performing segmentation. The authors define semantic segmentation as the process of transforming raw medical images into clinically relevant, spatially structured information, such as outlining tumor boundaries, and highlight that segmentation is an essential prerequisite for a number of clinical applications, such as radiotherapy planning and treatment response monitoring (Antonelli et al., 2022). The MSD ties in with the concept of DG and is presented as a benchmark to find an algorithm leading to a better generalization capability of a model. A comparison between a generalistic model that has been trained on multiple tasks and performs well on them versus models trained just on and for a specific dataset is a focal point of the MSD. The idea that such a generalistic model might approach or even surpass the performance of a custom-designed model is of particular interest.
The MSD challenge was split into two distinct phases: first the development phase and second the mystery phase. In phase one, seven open training datasets were provided which could be used to train models. The participants downloaded the data themselves and were expected to train their algorithms on the training data of each task separately. Task-specific manual parameter setting was forbidden. Segmentation results garnered by running the model on the test data were submitted after training. Each team could make one submission per day, with the results having been displayed in a live leaderboard on the challenge website[4](med). Then in the second phase, the mystery phase, three previously unseen data sets could be downloaded and trained on. Further training on these new datasets was permitted, however no changes to the method itself. Only one single submission of the segmentation results was allowed for the mystery phase. Even after the original challenge hosted in 2018 was closed, it's still possible to see the scores as well a live leader-

[2] `https://www.robots.ox.ac.uk/~vgg/decathlon/#res`
[3] `https://zeus.robots.ox.ac.uk/competitions/competitions/9#results`
[4] `https://medicaldecathlon.com/`

board, and more importantly, make submissions[5] (med). Each team can only make 15 submissions altogether. Multiple registrations for one team are strictly forbidden.

MedMNIST Classification Decathlon Resembling the predecessor of the BIG benchmark presented in this thesis, the MedMNIST Classification Decathlon was built upon the ten medical datasets of MedMNIST v1 (Yang et al., 2021). These ten datasets all come from the medical domain, which the authors note novices usually are hesitant to get into, as working with medical data often requires not just in-depth DL knowledge but also knowledge in the medical domain. Thus, MedMNIST v1 provides a pre-processed, publicly available dataset for which people don't require in-depth knowledge about the imaging modalities such as Computed Tomography (CT) or ultrasound to work with it. With the images all being of resolution 28×28, the MedMNIST v1 dataset is considered lightweight as it relates to computational load. Furthermore, the MedMNIST Classification Decathlon makes use of AutoML, which provides open-source AutoML tools Roberts et al. (2023). Those are often used and intended for researchers with not as much technical in-depth knowledge when it comes to e.g. preprocessing data. Different models can also be easily selected and made use of. Various metrics for evaluation are provided in those AutoML tools as well. For evaluation, the MedMNIST Classification Decathlon chooses to use the metrics Area under the Receiver Operating Characteristic Curve and the Accuracy.

AutoML Decathlon The AutoML Decathlon is presented by Roberts et al. (2023) as a machine learning competition focused on diverse tasks. There was a challenge based on the AutoML decathlon hosted at NeurIPS 2022. The decathlon focused on training models on 10 different tasks. These tasks were selected with regards to their diversity in terms of domain, input dimension, output dimension, output type, objective function, and scale. The challenge that was hosted at NeurIPS 2022 was divided into a development and a test phase. During the development phase, both the maximum time a model was allowed to train as well as the number of submissions were limited. The test phase then consisted of an evaluation on 10 withheld datasets. Submissions to the competition were made via CodaLab[6] (Pavao et al., 2023). For the AutoML decathlon, submissions were made in the form of code uploaded via CodaLab. The submitted code was run on infrastructure the challenge organizers provided. The maximum time a model was allowed to run for ranged from 0.2 to 20 hours, depending on the complexity of the task. Ten submissions per day and in total 100 submissions were permitted to be made during the development phase.

WILDS The WILDS benchmark was developed at Stanford university and is specifically designed for research on distribution shifts (Koh et al., 2021a). The authors detail several kinds of distribution shifts. First, when dealing with a related notion of DG, the training and test distributions might stem from similar, yet different domains. They give the example of a model being trained on data from just a few hospitals and then being deployed to a lot of other hospitals which weren't part of the training. This is reflected in WILDS for example in the CAMELYON17-

[5] `https://medicaldecathlon.com/`

[6] `https://codalab.lisn.upsaclay.fr/competitions/6325`

WILDS dataset, where the training data originates from several hospitals and the test data from a different hospital not included in the training data. Secondly they mention the subpopulation shift, where test distributions are subpopulations of the training distribution. The proportions of the training and test domains will often not be the same. The aim there is to not have a strong drop-off in performance with the worst-case subpopulation. They note the poor performance of standard models on under-represented demographic groups. An example of subpopulation shift is given in the CIVILCOMMENTS-WILDS dataset, where some demographics might be underrepresented in the training data but a performance similar to those of the demographics with more representation share is still desired. Thirdly, some hybrid scenarios might occur where an algorithm has to deal with both domain generalization and subpopulation shift, for this the authors mention the FMoW-WILDS dataset. Koh et al. (2021a) state that they substantially modified most of the datasets they include in their benchmark, such that they exhibit a more pronounced distribution shift. They also standardized data splits and preprocessed the data for use in standard ML frameworks.
Submissions for the benchmark can be made via a bespoke website[7] (wil, b)organizing the challenge for the benchmark. This custom-tailored website houses not just a leaderboard but many other resources like a guide about how to turn in submissions or a description of datasets too. Both newly developed algorithms but also re-implementations are allowed to be used. This distinction is visible in the leaderboard, which is marked via bold highlighting for original algorithms. It's stated that all submissions must use the dataset classes and evaluators which are available via the WILDS package (Koh et al., 2021b). Submissions are also divided into standard and non-standard submissions.
The guidelines for standard submissions, as taken from the submission page[8] (wil, a), are as follows: (1) Results must be reported on at least 3 random seeds. The following datasets must have more replicates: 5 random seeds for CIVILCOMMENTS; 10 random seeds for CAMELYON17; and 5 folds for POVERTYMAP. (2) The test set must not be used in any form for model training or selection. (3) The validation set must be either the official out-of-distribution (OOD) validation set or, if applicable, the official in-distribution (ID) validation set. (4) The validation set should only be used for hyperparameter selection. For example, after hyperparameters have been selected, do not combine the validation set with the training set and retrain the model. (5) Training and model selection should not use any additional data, labeled or unlabeled, beyond the official training and validation data. (6) To avoid unintended adaptation, models should not use batch statistics during evaluation. BatchNorm is accepted in its default mode (where it uses batch statistics during training, and then fixes them during evaluation). (7) Other dataset-specific guidelines: - For Camelyon17, models should not be pretrained on external data. Note: We have relaxed the constraint that models should not use color augmentation, since unlabeled data methods typically rely on data augmentation suites that include color augmentation. - For iWildCam, models should not be pretrained on

[7] https://wilds.stanford.edu/
[8] `https://wilds.stanford.edu/submit/`

external data. This includes off-the-shelf detectors (e.g., MegaDetector) that have been trained on external data.
Non-standard submissions on the other hand only need to follow the first two guidelines and will be highlighted as such. Those non-standard submissions are described to be: using unlabeled data from external sources, specialized methods for particular datasets/domains, such as color augmentation for Camelyon17 or using leave-one-domain-out cross-validation instead of the fixed OOD validation set. The submission itself is then made by uploading .csv-files with the predictions except for the GLOBALWHEAT dataset, for which a .pth-file is uploaded. A .tar.gz or .zip file is expected to be uploaded and a submission form to be filled out. This form contains various fields meant to store metadata but also offers the upload for the final archive file containing the predictions. A link to a paper detailing the method used as well as a link to a GitHub where the code is uploaded to are mandatory fields for the submission. It's important to note that the .pth-file for GLOBALWHEAT is used to store predictions and not the model. These predictions are evaluated by the WILDS organizers and said to be added to the leaderboard within a week.

1.4 Biomedical Image Generalization Benchmark

The BIG benchmark follows in the footsteps of the MedMNIST Classification Decathlon as a challenge including datasets from the MedMNIST family. For this benchmark, the MedMNIST+ collection of datasets is used. Hence the proposed benchmark focuses chiefly on image classification in the biomedical realm. A detailed description of the MedMNIST+ collection of datasets is given in section 4. Most importantly, the MedMNIST+ collection of datasets contains images from multiple biomedical domains, with imaging tasks ranging from binary classification to multi-class classification, multi-label classification and even ordinal logistic regression.
Firstly, this poses the challenge of creating a model which performs well across many different tasks. Secondly, given that MedMNIST+ consists of 12 datasets including but not limited to X-rays, microscopy and CT, there is a wide range of imaging modalities over which data was captured. Additionally, the classification tasks stem from different parts of the body, with ChestMNIST containing X-ray images of patients while BloodMNIST for instance pictures individual cells (Yang et al., 2023). The scale alone of the examined areas is vastly different, with cells being vastly different in size compared to the upper part of the torso, by many orders of magnitude. Also, blood cells are part of the body fluid blood, while the chest X-rays depict large areas of lung tissue.
Taking all these differences of modalities, tasks and medical domains into consideration, one goal of the BIG benchmark can be clearly stated: to evaluate the generalization capability of a model. This goal ties into the previously described concepts of DA and DG. However, for this benchmark, no dataset is withheld to purely use for evaluation. This implicates that every imaging task and modality has been seen during training. An approach to assess the generalization capability of

a model trained on the combined datasets of MedMNIST+ to an even greater extent would be to add a dataset to the benchmark which isn't publicly available and perform the evaluation on that dataset. So a withheld dataset that is only used for evaluation. However, this is just a possibility to explore DG to a greater extent and not per se the case for the BIG benchmark proposed in this thesis. One evaluation which stems from Doerrich et al. (2024) is used as a baseline for performance on each dataset individually. Two approaches treated in this extensive evaluation include evaluating pre-trained models using a kNN approach or linear probing. These two approaches are treated by Doerrich et al. (2024). Another approach related to DA and DG is training a model on the combinated collection of datasets. This entails feeding every sample of each dataset to the model during training and using different classification heads — one per classification task. The theoretical concept is elucidated in section 2.3 and its implementation described in section 3.3. This approach to training is one of several approaches which can be employed to make use of the unique properties of the MedMNIST+ collection of datasets, adding to the standard end-to-end training paradigm. The evaluation of the BIG benchmark focuses chiefly on three evaluation metrics: the Area under the Receiver Operating Characteristic Curve, the balanced accuracy and Cohen's Kappa. The theory behind those metrics and the reasons for using them is laid out in section 3.

2 Theoretical Foundations

The BIG benchmark proposed in this thesis represents a unique comparison of traditional CNNs and the more recent ViTs on various biomedical domains. The generalization capability of these architectures and models which are used especially are of great interest here. To establish a common ground for this undertaking, section 2.1 provides the necessary background and vocabulary on the field of Deep Learning. Section 2.1.2 then introduces the basic architectural concepts of CNNs. This focus on architectural specifics continues in 2.1.3 where ViTs are treated. A comparison between CNNs and ViTs completes the discussion of the conceptual and architectural differences of these two popular architecture types. As this thesis also includes a reevaluation of the models presented by Doerrich et al. (2024), those architectures are described in section 2.2. The ResNet architecture and the ViT architecture, both also used in Doerrich et al. (2024), are the two main architectures around which the experiments are structured in this thesis. The DL methods discussed throughout this section are not meant to be an exhaustive list that encompasses everything in the field of DL, as this would almost be impossible. Instead it is mainly focused on the architectures used for the baseline of the BIG benchmark.

2.1 Deep Learning

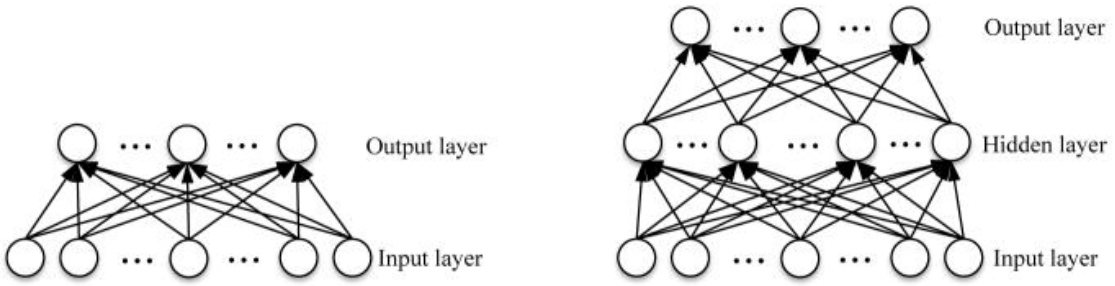
2.1.1 Machine Learning

Machine Learning Machine Learning (ML) is described by Jordan and Mitchell (2015) to chiefly focus on the following two questions: (1) How can one create computer systems which automatically improve with experience? (2) What are the fundamental laws that govern all learning systems, among them computers, humans, and organizations, as described by statistics, the theory of computation, and information theory? Both Jordan and Mitchell (2015) and LeCun et al. (2015) mention Machine Learning to deal with issues in computer vision, natural language processing, object identification, speech recognition, and robotics.

Representation Learning LeCun et al. (2015) define Representation Learning (RL) as a "set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification". In the context of images this might mean feeding a computer with images of a common format and finding a way to efficiently represent this image data, often expressed in pixels with colour values. This pixel-wise representation quickly blows up the number of computations needed to handle even a single image if one were to operate on the raw pixel data. To just represent a single image of dimensions $3 \times 224 \times 224$, where 3 might be the number of color channels in a RGB color scheme, and 224×224 is the number of pixels the image is in width and height, this already are 150,528 pixel values. Representation learning thus also is about finding more efficient ways to handle such data in order to be able to use an efficient representation to perform further computations with this representation.

Deep Learning Deep Learning DL is a subfield of ML, LeCun et al. (2015) highlight how deep learning models are composed of multiple processing layers such that representations of data with multiple levels of abstraction might be learned. Naturally, this also ties DL closely to RL. According to the authors, DL is characterized by those multiple layers being composed of simple but non-linear building blocks. These building blocks, or modules, each transform the representation such that a representation at a higher and more abstract level is obtained.

Artificial Neural Networks Feed-Forward Neural Networks are a type of Artificial Neural Network (ANN). Shen et al. (2017) bring up that ANNs are meant to emulate the computational principles of neural systems, which are supposed to learn patterns in observations. The architecture of such a Feed-Forward Neural Network is given in Fig. 1a. The left-hand schematic pictures a simple single-layer neural network where each neuron in the output layer receives information from each neuron in the input layer. Fig. 1b on the other hand shows a multi-layer neural network, where one or multiple so-called hidden layers are introduced between the input and output layer. A simple single-layer neural network can only approximate linear functions, whereas multi-layer neural networks, given general conditions and enough hidden units, are able to approximate any function (Hornik, 1991). Hidden units are individual neurons in the hidden layer. In fully-connected layers, any given neuron of a layer is connected to each neuron of the next layer as can be seen in Fig. 1.



(a) Single-layer neural network (b) Multi-layer neural network

Figure 1: Architectures of two feed-forward networks, reprinted from Shen et al. (2017)

2.1.2 Convolutional Neural Networks

Convolutional Neural Networks architecture Yamashita et al. (2018) list multiple building blocks which make up a CNN. There are convolutional layers, pooling layers and fully-connected layers. The input to a CNN will commonly be given as an array of numbers, a so-called tensor. This tensor, especially for image classification which this thesis is focused on, represents the image data. A convolutional layer is characterized by its use of convolutions, which are linear operations. Fundamentally, convolutions use a kernel, which is a tensor, and calculate the

element-wise product between a specified slice of the input and the kernel, where the slice of the input must fit the dimensions of the kernel. This kernel is then applied to each location of the input. For images, this means sliding the kernel sequentially over all the pixels of the image, covering each location along the dimensions. The result is a so-called feature map. An example of a convolution including the sliding of the kernel over an input and the resulting feature map is depicted in Fig. 2. Various kernels with different values for their elements can be used, resulting in an arbitrarily large amount of feature maps which can be produced starting from a single input. As Simonyan and Zisserman (2014) mention, a kernel of size 3×3 is the smallest possible kernel size with which the concepts of left/right, up/down and center can still be captured in the feature maps.

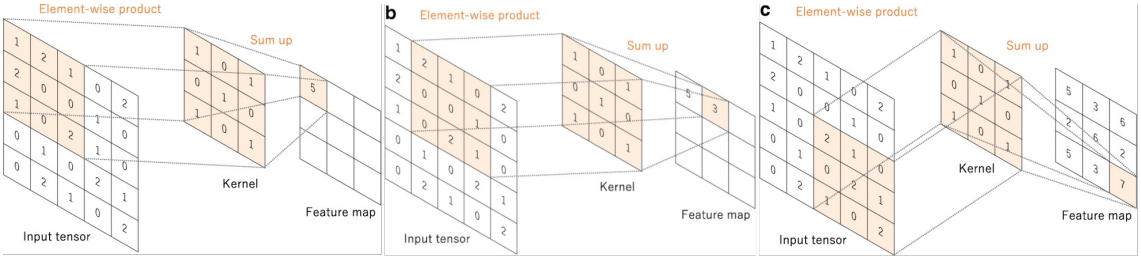


Figure 2: Convolutions, adapted from Yamashita et al. (2018)

Translation Equivariance Bias Lenc and Vedaldi (2015) define translation equivariance such that a "representation ϕ is equivariant with a transformation g of the input image if the transformation can be transferred to the representation output". They go on to describe how Translation Equivariance deals with how based on different transformations of an input image the representation changes. CNNs are found to perform simple linear transformations of the representation given transformations of the input image. The authors describe the changes of the representation as predictable.

Translation Invariance Bias The translation invariance bias is a special case of translation equivariance. The translation invariance bias is defined to be present in a system if the output is not influenced by any translation of the input (Myburgh et al., 2020). It's a characteristic that CNNs possess to some degree. Due to the kernels of a CNN sliding over the entire input and taking information from highly local areas, CNNs are commonly deemed translation invariant to some degree. Myburgh et al. (2020) also note that complete translation invariance, where the output is not influenced by a translation of the input in any way, is rarely achieved. Usually, for practical reasons a more pragmatic concept of translation invariance is used, where the sensitivity of a system to translated inputs is examined. There is some discussion around whether CNNs can be thought of as translation invariant but generally they are to some degree, using a less strict notion of the concept.

Pooling Layers Yamashita et al. (2018) elaborate on Pooling layers and explain how they're typically used for downsampling. This downsampling of the feature maps has the effect of introducing a translation invariance bias to small shifts and distortions. With downsampling, information is locally accumulated in a smaller space, thereby making the network less susceptible to translations applied to the input. Additionally, pooling decreases the number of subsequent learnable parameters. Different types of pooling operations exist, with two popular options being Max Pooling and Global Average Pooling. In Max Pooling, a cutout of the input feature map is taken and just the maximum value of this cutout is then determined to be the output, ultimately dropping the other values in that cutout. Global Average Pooling does reduce a feature map of dimensions height×width by means of taking the global average of the feature map. The depth, or number of channels, of feature maps is preserved in Global Average Pooling.

Locality Bias CNNs use much smaller receptive fields in the lower layers, thus capturing more acutely local information. This too stems from the fact that kernels slide over small parts of the image, extracting information locally. A receptive field is given as the size of the region in the input that produces the resulting feature Araujo et al. (2019). Building on the description from before of how a kernel is slid over an image it follows that for the first layer of a CNN, the size of the kernel matches the receptive field size. In subsequent layers, the receptive field size grows, as not just the kernel influences the part of the image which is considered as input but also the neurons from the layers before, which might have distilled representations of other parts of the input image. Pooling operations aggregate information from previous layers into a smaller resolution feature map and contribute to the receptive field size getting larger the more layers there are.

Two-dimensional neighborhood structure A feature of CNNs is that a sort of understanding of the 2D spatial structure and the concept of neighborhood is inherent in the architecture (Dosovitskiy et al., 2020). As Yamashita et al. (2018) describe, kernels usually have filter sizes of $3 \times 3, 5 \times 5$ or 7×7. Thus they capture spatial information of the immediate neighborhood of the 2D input they slide over. Not just the pixels used for one calculation are related to each other but given a low stride, the subsequent areas the kernel performs calculation on are also immediate neighbors.

Overfitting and Generalizability Overfitting as described by Yamashita et al. (2018) refers to a model learning statistical regularities specific to the training set, thus causing worse performance on other datasets such as the validation set. This according to Shorten and Khoshgoftaar (2019) leads to poor generalizability. They give the term generalizability as "the performance difference of a model when evaluated on previously seen data (training data) versus data it has never seen before (testing data)".

Residual Learning Residual Learning is formulated by He et al. (2016) with the underlying idea of assuming that if multiple nonlinear layers can asymptotically approximate complicated functions, then assuming they can asymptotically approx-

imate the residual functions follows from that initial assumption. Thus, the stacked layers of a network can't just be used to approximate a mapping H(x) but also the residual function given in equation 2. The initial mapping H(x) can then be viewed as equation 3, where both equation 2 as well as equation 3 should be able to be asymptotically approximated by nonlinear layers.

$$F(x) := H(x) - x \tag{2}$$

$$H(x) := F(x) + x \tag{3}$$

For this learning of the residual function, residual blocks are introduced. They're given by (He et al., 2016) such that they use so-called shortcut connections, connecting a layer not just with its following layer but also to a later layer. This is done by means of an identity mapping. With a shortcut connection, the input fed into the residual block is being added element-wise onto the output of the stacked layers. These identity shortcut connections neither introduce additional parameters nor additional computational complexity. Most importantly, residual blocks with their shortcut connections facilitate learning of identity mappings by nonlinear layers. The inability of converging deeper plain networks (without residual blocks) to achieve the same training error or accuracy as compared to less deep converging networks with the same number of parameters is coined the degradation problem by He et al. (2016). The authors conjectured that the degradation problem might be due to plain deeper networks struggling to approximate identity mappings with multiple nonlinear layers. This would explain why plain deeper networks suffer from the degradation problem, as they struggle to approximate the case where the input x is already very close to the target function H(x). Deeper networks with residual blocks have a way of skipping a few layers by means of the shortcut connections, feeding the identity mapping x to later layers and only approximating the residual F(x) with the multiple stacked layers, and thus combating the degradation problem. He et al. (2016) propose a bottleneck design for the residual blocks, using a stack of

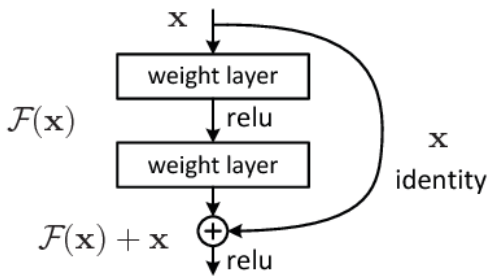


Figure 3: Residual learning: a building block, reprinted from He et al. (2016)

three layers instead of two. The first of the three layers is of dimensions 1×1, while the second is of dimensions 3×3 and the third layer is again of dimensions 1×1. Here, the first 1×1 layer reduces the dimensions, the 3×3 layer then is applied to this lower-dimensional input and the final 1×1 layer restores the initial dimensions. This bottleneck design is visualized on the right in Fig. 4. Computational efficiency

is the main reason for employing such a bottleneck design. The reduction of dimensions by the first 1×1 layer makes it possible to have just one 3×3 convolutional layer with 64 channels instead of two 3×3 convolutional layer with 64 channels like there are pictured on the left in Fig. 4.

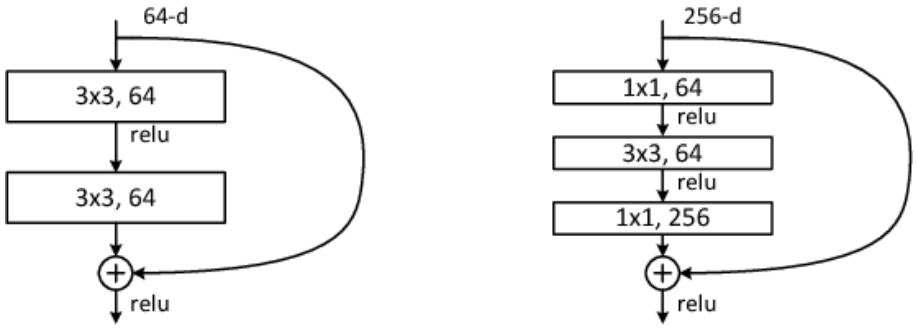


Figure 4: A deeper residual function F for ImageNet, reprinted from He et al. (2016)

Dropout Yamashita et al. (2018) describe dropout as setting randomly selected activations to zero during training. Krizhevsky et al. (2012) mention how they set the output of each hidden neuron to zero with probability 0.5, causing those neurons which have their output set to 0 to neither play a role in the forward pass nor the backward pass anymore. The authors state that using dropout, a network is forced to learn more robust features, providing the rationale that complex co-adaptations of neurons are reduced, as a neuron can't rely on the presence of particular other neurons. A robust feature, as they describe it, is thus characterized by a neuron firing with a variety of different subsets of neurons, as each time that an input is given, dropout will set different sets of neurons to zero. Dropout hence aims to prevent a neuron relying disproportionately much on another neuron or another set of neurons to play its part in representing a feature.

Nonlinear Activation Functions Yamashita et al. (2018) describe how the outputs of a linear operation such as convolution are subsequently passed through a nonlinear activation function. Fig. 5 depicts some of the most well-known nonlinear functions, with the sigmoid and hyperbolic tangent trying to emulate biological neurons and the nowadays more popular ReLU simply taking the max.

ReLU The function for the non-saturating nonlinearity Rectified Linear Unit (ReLU) is given by Krizhevsky et al. (2012) as follows:

$$f(x) = \max(0,x) \tag{4}$$

The significance of ReLU is also highlighted by the authors who state that compared to tanh units, deep CNNs employing ReLU train several times faster. ReLUs are not dependent on input normalization to stop them from saturating.

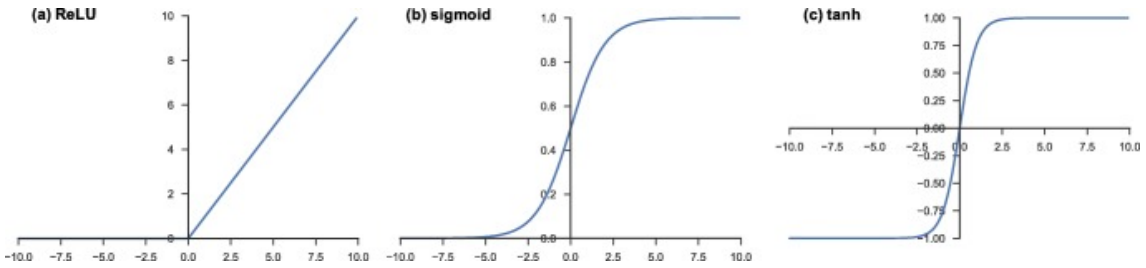


Figure 5: Activation functions commonly applied to neural networks, reprinted from Yamashita et al. (2018)

2.1.3 Vision Transformers

The Vision Transformer architecture as described by Dosovitskiy et al. (2020) in its essence is based on splitting an image into patches and feeding those patches as input to a Transformer. ViTs are usually pre-trained on large datasets and fine-tuned on smaller ones. The authors note that "Vision Transformers generally outperform ResNets with the same computational budget" (Dosovitskiy et al., 2020). Furthermore, the authors note that Big Transfer (BiT) ResNets perform better than ViTs on small datasets, however ViTs surpass them when pre-trained on larger datasets. Fig. 6 shows an overview over the model architecture. The input is given as the sequence of linear embeddings corresponding to the patches. A position embedding is added onto the linear projection of flattened patches. In the next step, a multilayer-perceptron (MLP) head is used for obtaining the predicted class.

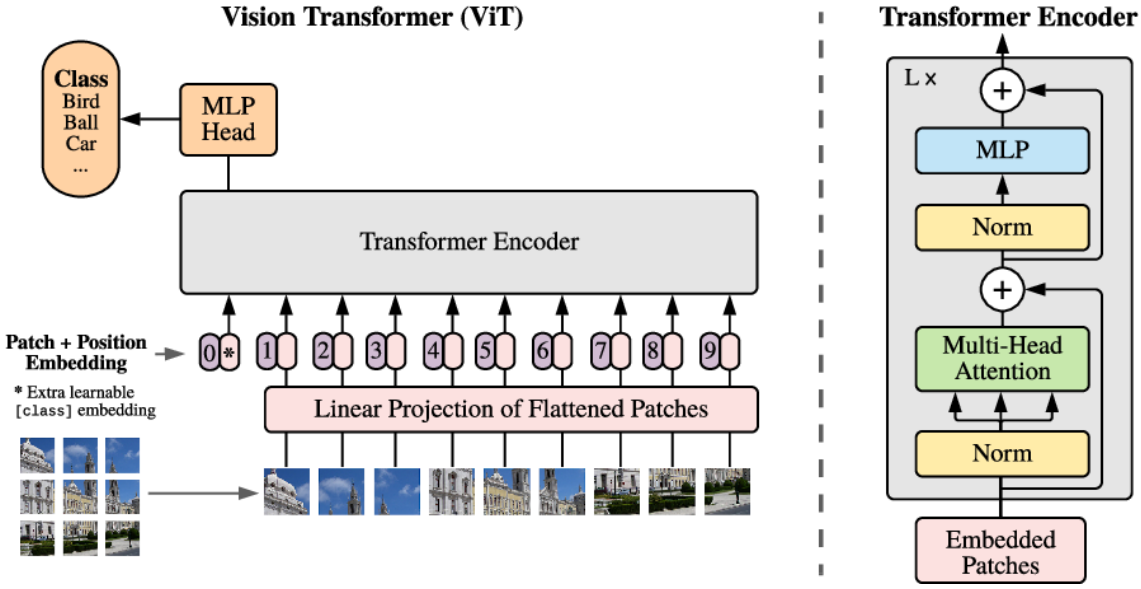


Figure 6: Model Overview, reprinted from Dosovitskiy et al. (2020)

Dosovitskiy et al. (2020) state how previous works by Cordonnier et al. (2019) used a patch size of just 2×2 pixels, leading their approach to only be applicable to small resolutions. So Dosovitskiy et al. (2020) used patches of size 16×16, allowing for medium resolution images to be used for training. As Transformers typically receive an 1D sequence of token embeddings (Dosovitskiy et al., 2020), the 2D images need to be manipulated in order to be processed by the Transformer architecture. Input images are split into patches of dimensions 16×16 pixels and are subsequently linearly embedded. These embeddings are coined "patch embeddings" by Dosovitskiy et al. (2020). Class tokens are prepended to these patch embeddings. Position embeddings are added to their corresponding linear projection of flattened patches. Both of these are vectors, so adding means vector addition here. This resulting 1D sequence is then fed into the Transformer Encoder. Said Transformer Encoder consists of layers of multiheaded self-attention (MSA) and MLP blocks, both alternating. Dosovitskiy et al. (2020) give the equations for self-attention (SA) and MSA. Here, standard qkv self-attention (equation 5 - equation 7) is described such that for each element in an input sequence z $\epsilon \mathbb{R}^{N \times D}$ a weighted sum over all values v in that sequence is computed. Also, the attention weights A_{ij} are said to be based on the pairwise similarity between two elements of the sequence, their respective query q^i and key k^j representations.

$$[\text{q,k,v}] = \text{z}U_{qkv} \qquad U_{qkv} \in \mathbb{R}^{D \times 3D_h} \tag{5}$$

$$\text{A} = \text{softmax}(qk^T/\sqrt{D_h}) \qquad A \in \mathbb{R}^{N \times N} \tag{6}$$

$$SA(z) = Av \tag{7}$$

MSA then extends SA, parallely running k-self-attention operations which are called "heads". The outputs of these operations are then concatenated.

$$\text{MSA(z)} = [SA_1(z); SA_2(z); ...; SA_k(z)]U_{msa} \qquad U_{msa} \in \mathbb{R}^{k \cdot D_h \times D} \tag{8}$$

The MLP blocks are mentioned to contain two layers with a GELU nonlinearity with the equations given by Dosovitskiy et al. (2020)

$$z_0 = [x_{class}; \quad x_p^1 E; x_p^2 E; ...; x_p^N] + E_{pos}, \qquad E \in \mathbb{R}^{(P^2 \cdot C) \times D}, E_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{9}$$

$$z'_l = MSA(LN(z'_{l-1})) + z'_{l-1} \qquad l = 1...L \tag{10}$$

$$z'_l = MLP(LN(z'_l)) + z'_l \qquad l = 1...L \tag{11}$$

$$z'_l = LN(z_L^0) \tag{12}$$

Dosovitskiy et al. (2020) describe how Layernorm is applied before every block while residual connections are applied after each block. Finally, an MLP head with one hidden layer for pre-training and a single linear layer for fine-tuning is used.

Masked Image Modeling Masked Image Modeling (MIM) is presented by Bao et al. (2021) as a task to pre-train vision transformers. With MIM, a ViT is fed both image patches and visual tokens. As specified by the authors, MIM uses both

image patches and visual tokens as representations of the input images. To obtain the image patches, the image $x \in \mathbb{R}^{H \times W \times C}$ is split into a $N = HW/P^2$ sequence of patches $x^p \in \mathbb{R}^{N \times (P^2 C)}$ with C being the number of channels, (H,W) the input image resolution, and (P, P) the resolution of each patch. The patches are then flattened into vectors and linearly projected. For obtaining the visual tokens, the image $x \in \mathbb{R}^{H \times W \times C}$ is tokenized into $z = [z_1, \ldots, z_M] \in V^{h \times w}$ with V being the vocabulary $V = \{1, \ldots, |V|\}$. The authors refer to the discrete variational autoencoder following the principles of Ramesh et al. (2021) as the tokenizer that they used. During pre-training, some percentage of image patches is masked with the objective of predicting the visual tokens for the masked image patches. During training, a reconstruction loss is used to evaluate the model's predictions and based on that updates to the weights are performed. For image classification purposes, a linear layer is attached to the pre-trained transformer. Average pooling is used to aggregate the patch embeddings before this representation is fed into the linear layer and a softmax gives the probabilities for the prediction after the representations have passed through the linear layer.

Global Attention Bias The self-attention layers of ViTs are global in the way they process information (Dosovitskiy et al., 2020). Raghu et al. (2021) write about how even in the lowest layers of ViTs, self-attention layers exhibit a mix of local heads, so small distances, and global heads with large distances. Distance here refers to the distance between patches in the latent space. This also leads to somewhat large effective receptive fields already in the lower layers (Raghu et al., 2021). The authors note that at higher layers, all self-attention heads are global.

Positional Encoding Bias The positional embeddings are meant to give ViTs a sense of locality which otherwise they wouldn't have due to their pronounced global attention bias (Dosovitskiy et al., 2020). There is no concept of spatial structure in the transformer architecture, as inputs are sequences of tokens. Positional Embeddings address this to some extent.

Comparison of standard feedforward networks, convolutional neural networks and vision transformers Per Krizhevsky et al. (2012), there come fewer connections and parameters with CNNs as compared to feedforward networks, which results in a theoretically slightly worse best performance. However, this comes with the upside of vastly greater computational efficiency, as in CNNs not every neuron is connected to every other neuron. As Dosovitskiy et al. (2020) elucidate, transformers miss the inductive biases of CNNs like translation equivariance and locality. This, according to the authors, leads transformers to generalize poorly provided that training is done on insufficient amounts of data. Furthermore, Dosovitskiy et al. (2020) claim that ViTs didn't just use computationally fewer resources during pre-training compared to state-of-the-art convolutional neural networks, they also approached or even beat the performance of state-of-the-art convolutional neural networks. Dosovitskiy et al. (2020) even state that "large scale training trumps inductive bias". So while it's true that ViTs generally require more data than CNNs to perform well, they are more computationally efficient during training, requiring less computational resources. Moreover, according to Dosovitskiy et al. (2020) ViTs have much

less image-specific inductive bias as compared to CNNs. Whereas for CNNs two-dimensional neighborhood structure as well as translation equivariance and locality are inherently present throughout the entire model, for ViT only MLP layers are local and exhibit translational equivariance. The self-attention layers of ViTs are largely global, as previously described. CNNs don't process the input as a sequence of tokens and thus do not use the positional encoding, which introduces some locality in ViTs that they otherwise lack. This locality isn't nearly as strong as the locality of CNNs. Whereas CNNs have fixed, local receptive fields (Raghu et al., 2021) due to using a kernel to slide over the image, ViTs have larger effective receptive fields. The authors also observed much greater similarity comparing lower and higher layers in ViTs than it was the case for CNNs.

2.2 Architectures

AlexNet The introduction of what is nowadays commonly known as "AlexNet", a convolutional neural network presented by Krizhevsky et al. (2012), resembles a major stepping stone in the field of Deep Learning. AlexNet is hailed by Singh et al. (2020) as a "grand success". With citations in the tens of thousands and growing, AlexNet is commonly brought up when reviewing DL and its beginnings, such as in Alzubaidi et al. (2021), where it is said to have spurred an "innovative research era in CNN applications" Alzubaidi et al. (2021). While not necessarily being the first to invent the concepts, Krizhevsky et al. (2012) were some of the first researchers to employ the regularization method "dropout" as well as making use of Rectified Linear Units, given in equation 4. As per Alzubaidi et al. (2021), the previously widely popular and state-of-the-art CNN "LeNet" had five feature extraction stages, which AlexNet brought up to seven. This implicated its own challenges, such as having to find a way to deal with overfitting, but certainly aided in Krizhevsky et al. (2012) ultimately winning the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) in 2012 (Russakovsky et al., 2015), with the authors stating that removing even one convolutional layer would have resulted in inferior performance. The problem of overfitting was tackled in part by making use of the dropout method described in 2.1.2.

The detailed AlexNet architecture is depicted in Figure 7. In the AlexNet architecture, five convolutional layers are followed by three fully connected layers, summing up to eight layers for which the weights have to be adjusted. For classification, the output of the last fully connected layer is given as input to a 1000-way softmax function in the case of the ILSVRC classification task. After the first and second convolutional layers, response-normalization layers were utilized. Subsequently to those response-normalization layers there come max-pooling layers as well as after the fifth convolutional layer. After each convolutional and fully connected layer, the output of that individual layer is fed into the ReLU nonlinearity described in 2.1.2. The AlexNet architecture expects an input of dimensions $224 \times 224 \times 3$, so images with 224×224 pixels and 3 RGB channels. The first convolutional layer then uses 96 kernels of size $11 \times 11 \times 3$ with a stride of 4 pixels. Receiving the response-

normalized and max-pooled output of the first convolutional layer as input, the second convolutional layer uses 256 kernels of size $5 \times 5 \times 48$. The authors give the kernels of the third convolutional layer as 384 kernels of size $3 \times 3 \times 256$ while noting that the third convolutional layer receives the response-normalized and max-pooled outputof the second convolutional layer. The remaining third, fourth and fifth layers will not have any response-normalization layers or max-pooling layers in-between. That fourth convolutional layer comes with 384 kernels of size $3 \times 3 \times 19$ and the fifth and final convolutional layer with 256 kernels of size $3 \times 3 \times 192$. Lastly, each fully connected layer has 4096 neurons. The authors give the number of parameters for their model as 60 million. 650 000 neurons make up the network.

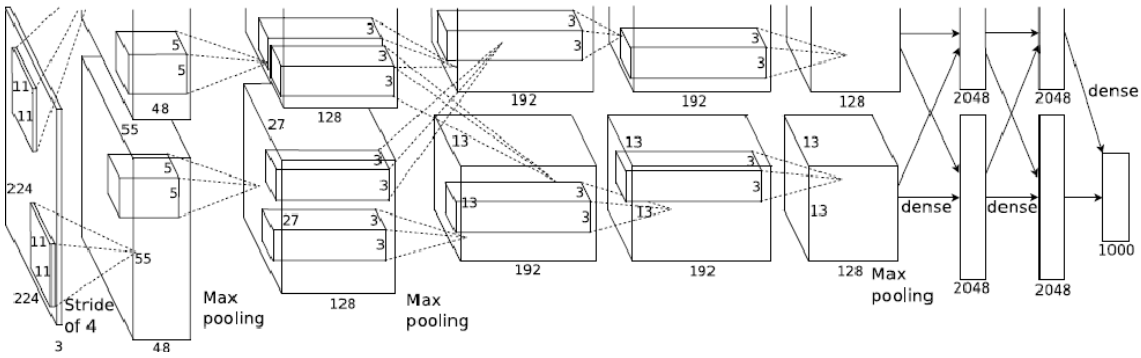


Figure 7: An illustration of the AlexNet architecture, reprinted from Krizhevsky et al. (2012)

VGG The Visual Geometry Group (VGG) architecture has been introduced by Simonyan and Zisserman (2014) with the idea of building onto the AlexNet architecture by Krizhevsky et al. (2012) by adding more network layers, ultimately extending it to create a much deeper network. The VGG-16 and the VGG-19 architectures are the most important here, having 16 and 19 weight layers respectively. The architecture is depicted in Fig. 8. VGG nets expect as input an RGB image of size 224×224. This input is put through convolutional layers with a kernel size of 3×3. A convolutional stride of one is used as well as a padding of one pixel for the 3×3 convolutional layers. In total, for the VGG16 five max pooling layers are used which follow some of the convolutional layers. The Max-pooling is performed with a stride of two over an area of 2×2 pixels. Three FC layers follow the stack of convolutional layers. Those first two of the FC layers each have 4096 channels, while the third has 1000, corresponding to the 1000 classes that needed to be classified for the ILSVRC 2014 challenge the authors participated in and won. Each hidden layer of the network is followed by a ReLU nonlinearity described in section 2.1.2.

ResNet He et al. (2016) present the Residual Network (ResNet) architecture in their paper. The ResNet architecture is based on the implementation of the Residual Blocks from Section 2.1.2. A visualization of a comparison of the VGG-19 architecture, a plain 34-layer CNN and a 34-layer ResNet is given in the appendix as Fig. 18. It is important to note that despite the increase in depth of the 34-layer

Figure 8: The architecture of VGG-16 model, reprinted from Shi et al. (2018)

ResNet as compared to the VGG networks, the ResNet still has fewer filters and fewer computational complexity than the VGG networks.

The convolutional layers are described by the authors to mostly have 3×3 filters, with the depicted 34-layer ResNet having 3×3 filters in all convolutional layers except for the very first one, which has 7×7 filters. Two design rules are given by the authors: (1) the layer should have the same number of filters for the same output feature map size and (2) when the feature map size is halved, the number of filters gets doubled in order to retain the time complexity per layer. Downsampling is realized by the convolutional filters with stride two. Global Average Pooling is conducted before the single fully-connected layer, the final layer. Given the participation at ILSVRC 2015 with their ResNet architectures, the authors chose a 1000-way fully-connected layer with softmax for the final classification layer, meeting the requirements for being able to classify the 1000 different classes of the ILSVRC 2015 challenge.

The shortcut connections, which make the ResNet a Residual Network, come in two variants. If input and output dimensions match, the identity shortcuts can be used without further adaptations needed, this is resembled by the solid line shortcuts in Fig. 18. However, if the input and output dimensions do not match, He et al. (2016) give 2 options. In option one the shortcut still performs an identity mapping but zero-padding is used to match the dimensions. With option two, 1×1 convolutions are used to match dimensions. These cases, where the input and output dimensions do not match, are visualized by the dotted lines in Fig. 18. ResNet receives input images with pixels 224×224. Batch normalization is used after each convolutional layer and before each activation. Common ResNet-variants which the authors also presented results for are ResNets with 18 layers, 34 layers, 50 layers, 101 layers and 152 layers. All of these variants, while significantly deeper than a VGG model, still have lower complexity than a VGG model. Deeper ResNet architectures may rely

more on the bottleneck design for residual blocks detailed in section 2.1.2. These help reduce training time, being more computationally efficient. The 50-layer ResNet is constructed from the 34-layer ResNet by simply using a 3-layer bottleneck residual block for each 2-layer residual block that was in the 34-layer ResNet. Even deeper version like ResNet101 and ResNet152 are constructed by simply adding more of those 3-layer residual blocks.

DenseNet The term Dense Convolutional Network (DenseNet) was coined by Huang et al. (2017). Their DenseNet builds on the principles of ResNets. DenseNets, as opposed to ResNets, however don't just connect a layer to its subsequent layer as well as to a later layer, as ResNets do. In DenseNets, inside a so-called Dense Block, any layer is connected to all its subsequent layers. Likewise, any given layer receives as input all the feature-maps of all the preceding layers inside a Dense Block. This principle is visualized in Fig. 9. Another difference to ResNets is that in DenseNets, features passed into a Dense Block are combined by concatenation, not by summation. The authors go on to state how the l^{th} layer of a DenseNet has l inputs, them being the feature-maps of all the preceding layers inside a Dense Block. Knowing that the feature-maps of a layer are fed to all $L - l$ subsequent layers, one arrives at $\frac{L(L+1)}{2}$ connections in an L-layer network. This would be just L connections for other architectures where the output of a layer isn't fed to all its subsequent layers. Due to this mechanism, Huang et al. (2017) state that DenseNets actually require fewer parameters than traditional CNNs, as there are no redundant feature-maps which must be learned. With often just 12 filters per layer, DenseNet layers seem narrow. DenseNets come in various depths, with depths of 121, 169, 201 and 264 presented by Huang et al. (2017). All these DenseNet variations have in common an initial convolutional layer with 7×7 convolutions and stride 2. This initial layer is followed by a Pooling layer using 3×3 max pooling and stride 2. In each variation, the aforementioned Pooling layer is followed by four Dense Blocks, each separated by a convolutional layer with kernel size 1×1 and a 2×2 Average Pooling layer with stride 2. The authors name these two layers separating the Dense Blocks transition layers. The fourth and final Dense Block is followed by a 7×7 Global Average Pooling layer. Ultimately, after that Global Average Pooling layer comes a classification layer, specifically a 1000D fully-connected layer (used for the 1000 classes of the ILSVRC 2012 challenge) which is then finally followed by a softmax function. The specification details for DenseNets of different depths as presented by Huang et al. (2017) are included in Fig. 19.

EfficientNet The core idea behind the EfficientNet architecture presented by Tan (2019) is to introduce a scaling method which uniformly scales the dimensions depth, width and resolution. A constant ratio, so a fixed set of scaling coefficients is employed to scale the network. These coefficients were determined by a small grid search on the original small model. The compound scaling method is given by the authors as shown in equation 13.

$$\begin{aligned} \text{depth: } d = \alpha^{\phi} \qquad & width : w = \beta^{\phi} \qquad resolution : \gamma^{\phi} \\ \text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \qquad & with\ \alpha \geq 1,\ \beta \geq 1,\ \gamma \geq 1 \end{aligned} \tag{13}$$

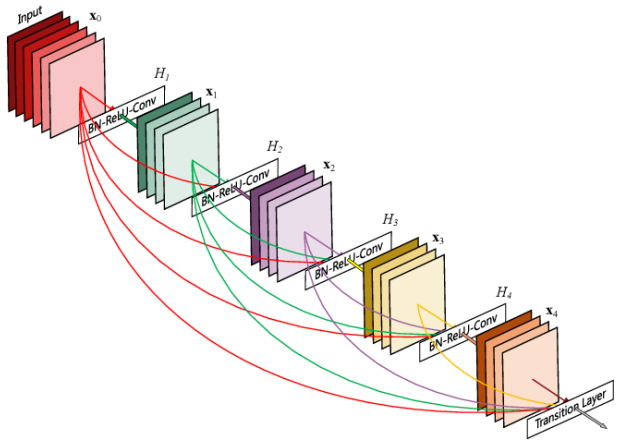


Figure 9: Dense Block structure, reprinted from Huang et al. (2017)

The α, β *and* γ constants are found via a grid search. ϕ is user-defined and depends on how many extra resources the network might be granted. The authors constrained $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ so the total FLOPS will increase by approximately 2^ϕ for any new ϕ. The architecture of EfficientNet is specified in Fig. 10. The 0 in "EfficientNet-B0" stands for $\phi = 0$, the B stems from mobile inverted bottleneck convolutional layers being used. The layers, also called MBConv, are detailed by Sandler et al. (2018). Tan (2019) added a squeeze-and-excitation optimization to the MBConv-layers, given by Hu et al. (2018). The key concept of EfficientNet is illustrated in Fig. 20.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figure 10: EfficientNet-B0 baseline network, reprinted from Tan (2019)

ViT-B/16 The architecture of ViT-B/16 is essentially depicted in Fig. 6. The B in ViT-B/16 denotes a "Base", meaning it used the base version for this vision transformer, as presented in section 2.1.3. The 16 denotes the patch size, so input images are split into images of size 16×16 before feeding it into the transformer encoder.

CLIP ViT-B/16 This architecture is based on Contrastive Language-Image Pre-training (CLIP). Radford et al. (2021) detail that CLIP works by using both a text encoder and an image encoder in order to predict the pairings of a batch of (image, text) training examples which belong together. To realize this, the text encoder and the image encoder are jointly trained to learn a multi-modal embedding space. A visualisation of this is provided in Fig. 11. The joint training is done through maximization of the cosine similarity of the image and text embeddings of the N real pairs in a batch. Simultaneously, the cosine similarity of the embeddings of the $N^2 - N$ incorrect pairings is minimized. The authors describe optimizing a symmetric cross entropy loss over these similarity scores. For CLIP ViT-B/16 specifically, the ViT-B/16 architecture given in Fig. 6 is used as the image encoder with the only modifications being the addition of an additional layer normalization to the combined patch and position embeddings before the transformer. The authors also used a slightly different initialization scheme. As the text encoder a Transformer described by Radford et al. (2019) is used. Crucially, using the CLIP ViT-B/16 architecture for image classification with preceding fine-tuning entails a training step analogous to step (1) of Fig. 11. The training images of the dataset which the CLIP ViT-B/16 is about to be trained on are used to adjust the weights of the encoder as calculated based on the symmetric cross entropy loss over the similarity scores. To perform image classification, the class names are taken and fed into the text encoder. The class labels are represented as a sort of prompt template in the form of "A photo of class label" Radford et al. (2019) (see supplementary materials). This is depicted as step (2) in Fig. 11. For the final classification of an unseen test image, the similarity between the text embedding and the image embedding is then calculated and the label with the highest similarity score is then predicted as the class label for the image.

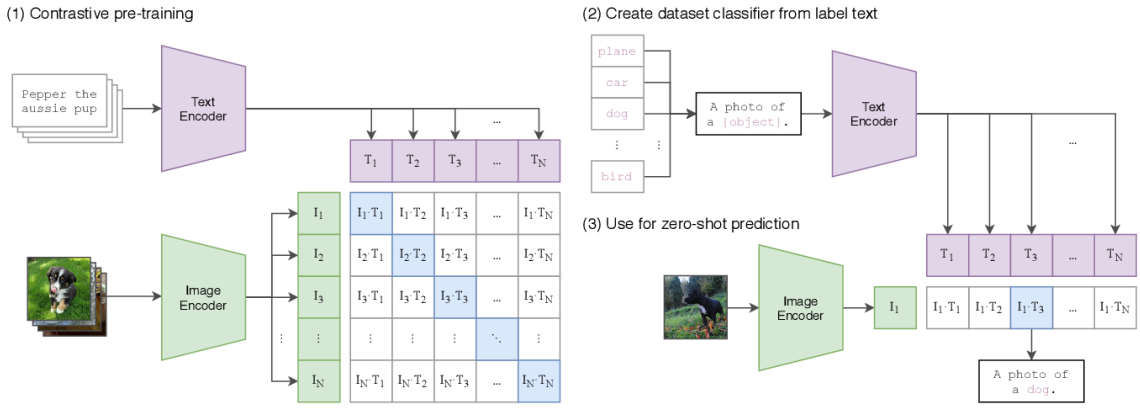


Figure 11: CLIP Architecture, reprinted from Radford et al. (2021)

EVA-02 EVA-02 is a transformer architecture presented by Fang et al. (2024). The EVA acronym stems from the description "Explore the limits of Visual representation at scAle [sic!]" Fang et al. (2023). EVA-02 is a ViT-inspired architecture, using several enhancements shown in Fig. 12. The characteristics of EVA-02 are that it is based on Transform Vision (TrV). Trv builds onto the foundation of the ViT architecture and modifies it. EVA-02 is based on TrV but sufficiently pre-trained from EVA-CLIP using MIM, which was described in section 2.1.3. The main architectural differences between TrV and ViT which was introduced in section 2.1.3 are that TrV uses Rotary Position Embedding (RoPE) instead of RPE, a Sigmoid Linear Unit (SiLU) instead of a Gaussian Error Linear Unit (GELU) and an additional layer normalization. Where not otherwise specified, switch gated linear units are used as part of the feed forward network. For a detailed description of those enhancements, see Fang et al. (2024), Hendrycks and Gimpel (2016), Ba (2016) and Su et al. (2023) as this would go far beyond the scope of this thesis.

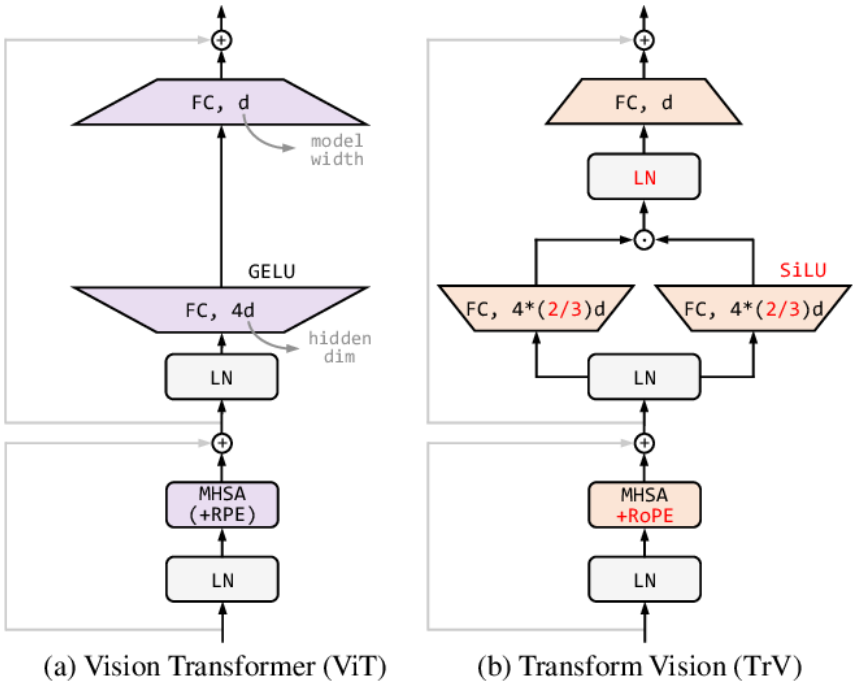


Figure 12: ViT and TrV Blocks, reprinted from Fang et al. (2024)

DINO The term Self-DIstillation with NO labels grants DINO its name. Caron et al. (2021)'s visualization of this is shown in Fig. 13. For an input image, two global views are created by cropping the image to retain more than 50% of the area of the original image. Likewise, several local views are obtained by cropping the image such that less than 50% of the area of the original is contained. This distinction of differently cropped views is important, as different sets of those cropped inputs will be fed into different branches. As Fig. 13 displays, DINO uses both a student and a teacher network. Only one of the global views is fed into the teacher network, while the student network receives both the local views and the remaining global view, so

the entire crops excluding the crop used for the teacher network, as input. Both the student and the teacher network are ViT-based (see section 2.1.3 for a description of the ViT architecture). The ViT-based backbone is followed by a 3-layer MLP, an l_2 normalization and a weight normalized fully connected layer. The teacher network is directly linked to the student network as its weights are computed by means of an exponential moving average on the student network's weights. Caron et al. (2021) go on to explain how knowledge distillation works using DINO. For this, an input image is given to the student network g_{θ_s} which is trained to match the output of the teacher network g_{θ_t}. The student and the teacher network each output probability distributions. The goal is the minimization of the cross-entropy loss with regards to the parameters of the student network as shown in equation 14 and equation 15.

$$\min_{\theta_s} \sum_{x \in \{x_1^g, x_2^g\}} \sum_{\substack{x' \in V \\ x' \neq x}} H(P_t(x), P_s(x')). \tag{14}$$

$$\min_{\theta_s} \sum_{x_t \in x_1^g, x_2^g} \sum_{x_s \in V} H(P_t(x_t), P_s(x_s)), \tag{15}$$

One important detail to note is that the ViT-based networks receive the sequence of patches and an extra learnable token which is added to the sequence as input. This token is coined a [CLS] token although the authors note that it doesn't have a relation to a label or supervision in the case of DINO. For pre-training, DINO was pretrained on the ImageNet dataset Russakovsky et al. (2015) without labels. The previously described projection head is discarded for image classification and instead a fully connected layer is added for classification.

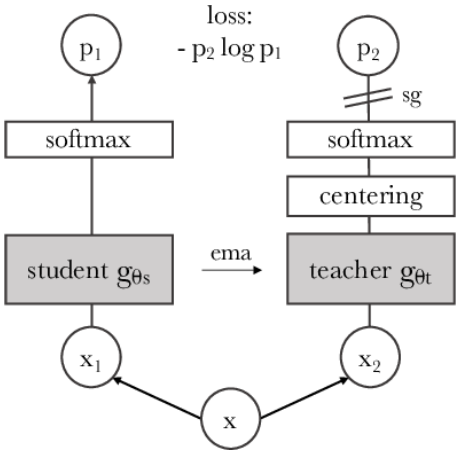


Figure 13: Self-distillation with no labels, reprinted from Caron et al. (2021)

SAM The Segment Anything Model (SAM) is primarily designed for segmentation tasks but can be employed for image classification tasks as well. Kirillov et al. (2023) describe how SAM uses an image encoder to compute an image embedding and a prompt encoder which processes the prompts accompanying the image. Using these two as sources of information then enables SAM to output segmentation masks. This process is depicted in Fig. 14. After pre-training, which has already been done as presented in the paper, the image encoder of SAM can be loaded with pre-trained weight and then be taken advantage of to apply it to image classification tasks.

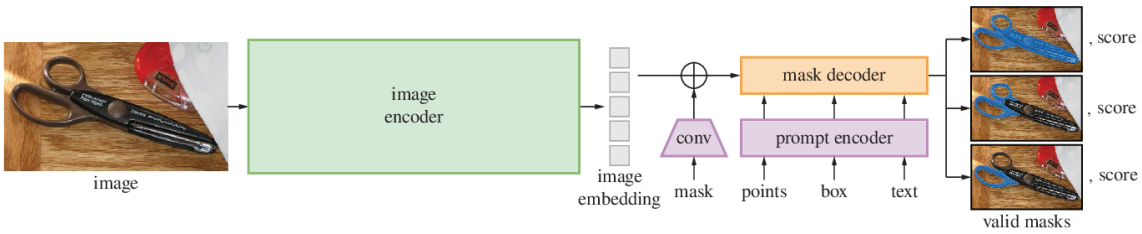


Figure 14: Segment Anything Model (SAM) overview, reprinted from Kirillov et al. (2023)

Model overview Table 1, which was adapted from Doerrich et al. (2024), displays an overview over the models used by the authors in their evaluations of the MedMNIST+ collection of datasets. Evidently, the parameter count among the ViTs is quite similar for the four Vision Transformers ViT-B/16, CLIP ViT-B/16, EVA-02 ViT-B/16 and DINO ViT-B/16. The SAM ViT-B/16 architecture has 89.7 parameters and thus remains in a similar range of the other ViTs where e.g. ViT-B/16 has 86.6 parameters. However, with its additional 3.1M more parameters than ViT-B/16 the SAM model used is the largest of the ViTs and also the farthest away from the original ViT-B/16 in terms of parameter count. The CNN architectures used by Doerrich et al. (2024) exhibit a much wider range of parameter count with the smallest being DenseNet-121 with roughly 8M parameters and the largest being VGG16 with 138.4M parameters, more than 17 times as many parameters as DenseNet-121. ResNet-18 is given as having 11.7M parameters. So a ViT-B/16 model has roughly more than seven times the number of parameters than a ResNet18. It's important to note that these models have very different architectures. They don't just differ in the way that some are CNNs and some ViTs but also in their general architecture. ResNets for example were developed to fix some of the issues plaguing AlexNet and VGGs and thus even among CNNs there can't simply be drawn a comparison solely focused on higher parameter count, as the underlying architecture itself also contains different building blocks such as residual blocks with their residual connections, detailed in section 2.1.2.

Model	Params (M)	Activations (M)	GMACs	# Output Dimension
VGG16	138.4	13.6	15.5	4096
AlexNet	62.3	0.6	0.36	4096
ResNet-18	11.7	2.5	1.8	512
DenseNet-121	8	6.9	2.9	1024
EfficientNet-B4	19.3	34.8	3.1	1792
ViT-B/16	86.6	16.5	16.9	768
CLIP ViT-B/16	86.6	16.5	16.9	768
EVA-02 ViT-B/16	86.3	16.5	16.9	768
DINO ViT-B/16	85.8	16.5	16.9	768
SAM ViT-B/16	89.7	1343.3	486.4	256

Table 1: Model Overview, adapted from Doerrich et al. (2024)

2.3 Training Paradigm

Woerner et al. (2024) introduce a specialized training paradigm focused on the capability of a model to perform well over multiple datasets and thus tasks. The authors talk about the notion of catastrophic forgetting introduced by Kirkpatrick et al. (2017) which describes the poor performance of neural networks to learn different tasks sequentially. Thus catastrophic forgetting is the forgetting of knowledge about one or multiple source tasks, leading to a much worse performance on the evaluation of these source tasks which were forgotten as opposed to a newly trained task. So sequential training leads to a model forgetting about the original tasks it was trained on, causing a drop in classification performance on these tasks. The approach of Woerner et al. (2024) aims to combat this catastrophic forgetting by training a model on the union of various datasets with different tasks. They name their approach multi-domain multi-task pre-training (mm-PT). The basis of their training paradigm is that during training of the model, a batch is randomly sampled from one of the tasks. This usually will mean that from a dataset, for which there exists a dedicated classification task, a batch is sampled. Then this batch is fed as input to the model and used to adjust the parameters of the model. Such a model can reasonably be expected to possess not just one, but instead multiple classification heads, of which only the classification head corresponding to the randomly sampled task is active. Meaning only the weights and the bias of this classification head are being updated with the rest of the backbone while the other classification heads remain frozen. The pseudocode for an algorithm describing mm-PT is given by Woerner et al. (2024) and shown in algorithm 1.

3 Methods

3.1 Benchmark

Classification problems are the origin of the evaluations in this thesis. They themselves can be largely divided into binary classification, multi-class classification and multi-label classification, as given by Hossin and Sulaiman (2015). Binary classification is classification where a decision has to be made between exactly two classes. Multi-class classification implicates three or more target classes. In multi-label classification an instance (e.g. an image in image classification) can have zero or more labels, even multiple labels at once. There is no mutual exclusivity of classes as there is in binary classification or in multi-class classification. Making the transfer to image classification with neural networks, evaluation metrics measure a model's performance on a classification task. Hossin and Sulaiman (2015) also make the distinction of evaluation metrics into threshold, probability and ranking metric. They also note how the evaluation metrics from each of those categories ultimately boil down to being a single scalar value. When doing image classification, the evaluation commonly looks like this: a model is trained in a training phase and in a testing phase that model's predictions are collected in order to determine how many predictions were right and how many were wrong. For this notion, the authors introduce so-called discriminator metrics for binary classification, which are shown in table 2.

	Actual Positive Class	**Actual Negative Class**
Predicted Positive Class	True positive (*tp*)	False negative (*fn*)
Predicted Negative Class	False positive (*fp*)	True negative (*tn*)

Table 2: Discriminator Metrics, reprinted from Hossin and Sulaiman (2015)

Here, True Positive (TP) denotes the correctly predicted positive classes, False Negative (FN) the prediction that the positive class was predicted although it was the negative class, the False Positive (FP) that the negative class was predicted although it was the positive class and the True Negative (TN) that the negative class was predicted when it actually was the negative class.
True Positive Rate (TPR): the True Positive Rate is given by Bocchieri et al. (2023) as shown in equation 16. They describe it as the "hit" rate. It essentially describes the share of correct classifications. Another common name for the TPR is Sensitivity (Hossin and Sulaiman, 2015) or Recall.

$$\text{TPR} = \frac{\text{TP}}{\text{Positives}} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{16}$$

False Positive Rate (FPR): the False Positive Rate is given by Bocchieri et al. (2023) as shown in equation 17. They describe it as the "false alarm" rate. It measures the share of actual negatives which have been predicted as being positive by the model.

It is related to the specificity given in equation 18 by being its complement, as in equation 19.

$$\text{FPR} = \frac{\text{FP}}{\text{Negatives}} = \frac{\text{FP}}{\text{FP + TN}} \tag{17}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{Negatives}} = \frac{\text{TN}}{\text{TN + FP}} \tag{18}$$

$$\text{Specificity} = 1 - FPR \tag{19}$$

Accuracy (ACC): the accuracy is mentioned to be the most used evaluation metric by the authors. It essentially is the percentage of correct predictions made over the total instances over which predictions were made. The formula for the accuracy is given in equation 20.

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \tag{20}$$

Error Rate (ERR): the error rate is the complement to the accuracy. This relationship is displayed in equation 22. The error Rate can also be given in terms of TP, FP, TN and FP, as it is in equation 21.

$$ERR = \frac{FP + FN}{TP + FP + TN + FN} \tag{21}$$

$$ErrorRate = 1 - Accuracy \qquad Accuracy = 1 - ErrorRate \tag{22}$$

Multi-class classification metrics like the accuracy follow the same principles of those metrics for the binary case, but extended to accomodate for multiple classes. In the following the metrics will be presented as given by sklearn (sci), which was used to compute the metrics in the experiments presented in 4. The ACC for the multi-class classification case is given by equation 23. $1(\hat{y}_i = y_i)$ is the indicator function which equals one if $\hat{y}_i = y_i$ where y_i is the true class label of sample i and $\hat{y}_i$ the predicted class label.

$$ACC = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i) \tag{23}$$

Area under the ROC Curve (AUC): the AUC is a single scalar metric intended to measure the performance of a classifier. It is derived from the Receiver Operating Characteristic (ROC) curve. The ROC curve can be obtained by plotting the TPR and the FPR against each other as the threshold is varied. For the binary case, different thresholds at which samples are classified as positive or negative might result in different TPRs and FPRs. For this, the predicted scores, so the probabilities output by the model are fed into the roc_auc_score function by sklearn. The multi-class case then entails that for one class, the ROC is determined in a One-vs-Rest fashion, where the ROC is constructed as in the binary case with the current class being considered against all the other classes combined. Based on the resulting ROC the AUCs are then computed for all classes by simply averaging them. Bradley (1997) explain how the AUC represents the probability that a randomly chosen

positive example is correctly rated with greater suspicion than a randomly chosen negative example.

Balanced Accuracy (BALACC): the BALACC introduces the concept of weights to the ACC metric. This weight is determined by the inverse share a class takes up in the dataset, as shown in equation 24. $1(y_j = y_i)$ again is the indicator function, whereas y_i is the true value of sample i, with w_i being the accompanying sample weight.

$$\hat{w}_i = \frac{w_i}{\sum_j 1(y_j = y_i) w_j} \tag{24}$$

So the weight is inversely proportional to the proportion of samples that belong to a class inside of a dataset. That way classes with less samples will not be underrepresented in their contribution to the (balanced) accuracy score. The equation for the BALACC is given in equation 25.

$$\text{BALACC} = \frac{1}{\sum \hat{w}_i} \sum_i 1(\hat{y}_i = y_i) \hat{w}_i \tag{25}$$

Cohen's Kappa (CO): the CO is commonly used to measure interrater reliability. McHugh (2012) give the interrater reliability as the reliabiltiy across multiple data collectors. The key idea behind CO is to also account for the share of agreement which might be attributed to pure chance. As a form of correlation coefficient, the CO ranges from -1 to 1. Values of 0 are seen as measuring no agreement between the two raters, while values from 0.81-1.00 resemble "almost perfect agreement" (McHugh, 2012), whereas the more negative the CO becomes below 0, the worse the agreement between the raters is. Transferring this concept to the image classification domain, the two raters which are compared are a model's predictions and the existing ground truth. The formula for the CO is given in equation 26 where p_o is the probability of agreement of the two raters on the label assigned to any sample. On the other hand, p_e is defined as the expected agreement of both annotators assigning labels randomly. In sklearn this is an empirical prior over the class labels, calculated using the confusion matrix. This estimation is made by taking the outer product of the total predictions (columns) and the total amount of times the class is the true class (rows) and ultimately dividing this product by the total number of samples to normalize the result.

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{26}$$

Cheating Prevention One key question when designing any benchmark will be whether to use publicly available datasets as test sets or to withhold datasets for the evaluation. Crucial factors influencing this decision might be the availability of datasets which are fit to be part of the benchmark, the available computational resources and infrastructure in the form of a server and whether one wants to host a challenge on a custom-made website for example or use a popular site like for instance Kaggle[9] (kag) or Grand Challenge[10] (gra). More importantly, if publicly

[9] `https://www.kaggle.com/`

[10] `https://grand-challenge.org/`

available datasets are used, then that opens the door to a greater extent for cheating than conducting the evaluation on withheld datasets. However, given some domains like biomedical image datasets suffering from a lack of high-quality datasets that are available, it might not always be a question of whether one wants to withhold any datasets but also whether there are even datasets available which could be withheld. The benchmark presented in section 1.4 is based on the MedMNIST+ dataset collection[11] that's connected to MedMNIST v2 (Yang et al., 2023). This choice was made for several reason, one certainly being that MedMNIST+ is a large collection of biomedical datasets and datasets in the biomedical domain already are generally hard to come by as noted by Shen et al. (2017), who state that for most medical applications there are fewer than 1,000 images available. Also, there's a python package associated with MedMNIST+ and MedMNIST v2, allowing for a streamlined usage of the datasets, greatly facilitating downloading and handling the data, even providing code to evaluate the models trained on MedMNIST+. For the BIG benchmark there currently aren't any datasets withheld for evaluation but this might be something to add in the future. One adversarial approach to such a challenge would be training the model on the test datasets such that it performs well on them. Withholding datasets for evaluation also has the upside that people can't just take the labels from the publicly available source, perhaps modify the predictions a bit so it isn't immediately obvious they just took the results, and then submit them.

This leads to the next point: how and where the evaluation takes place. There are two common options. One is that the participants of a challenge submit their results in an easy-to-manipulate format like .csv or .json, as is the case for VDD and WILDS. Another one is that the participants upload code, which is then run on the infrastructure that's provided. Such code submissions with training and evaluation on infrastructure which was provided was done for the AutoML Decathlon (Pavao et al., 2023) for example. Other considerations tie into the previous points. If a benchmark were to use publicly available datasets, then an organizer might shuffle the datasets in order to prevent people from training on the test split, on which the evaluations will be made. Such shuffling is often done via random seeds, which can be reverse-engineered. So even if the training set is provided as a shuffled dataset it would be possible to find out the labels for the test set, without even needing to make a submission, as the datasets can be downloaded and tinkered with locally on an malicious actor's device. Providing a shuffled publicly available dataset also entails having to provide a custom solution for downloading that dataset. An upside of using publicly available datasets to organizers low on infrastructure might be not having to store the datasets for their challenge and not having to offer a download option. Instead they can just provide a link to the source where the datasets which are part of the challenge had been made available initially.

[11] `https://zenodo.org/records/10519652`

3.2 Website

Technical Implementation: There are three key components from a development standpoint which are vital to the website presented for this thesis: the realization of the user interface, back-end and the database that's being used. These resemble the cornerstones on which the bespoke website crafted for the BIG challenge is built on. For the user interface, React[12] (rea) was chosen. React is an open-source, Javascript-based library for creating user interfaces based on reusable, nestable components. For the back end, flask (fla)[13] was utilized. Flask is a lightweight, Python-based web application framework which provides functionality for HTTP requests as well as routing. The choice of Flask for the back end is based on the the required data processing capabilities of a website for hosting the BIG challenge. This framework is based on Python which offers the possibility to utilize the pandas library for easily manipulating and analyzing data. MySQL[14] (mys) was chosen for the implementation of the database. It's an open-source relational database management system that has been around for decades and enjoys great popularity. Widenius and Axmark (2002) even claim it's the most popular open source SQL database. These web technologies were largely chosen because they fulfill the task-specific requirements. Further reasons include extensive documentation which is often concomitant with great popularity that is given for all three mentioned web technologies.
Content: A contribution from section 1.2 is the development of a website on which a challenge based on the BIG benchmark could be hosted. As elaborated in 1.3.2, there exists a similar bespoke website that hosts a challenge for the WILDS benchmark. Building on that, several core components which constitute a well-structured website for the purposes of hosting such a challenge are identified. This is a non-exhaustive list, as with different benchmarks and challenge-setups different requirements will be in place. However, for most challenges, those core components will serve a good purpose. More importantly, the website for the possible accompanying challenge for the BIG benchmark will make use of this structure. Firstly, a landing page with an overview over what the website and thus challenge offers should be implemented. Secondly, a description of the challenge should be provided. Thirdly, an overview over the datasets which are used for the benchmark is useful. Then a submission page is absolutely needed to do justice to the idea of a bespoke website for hosting a challenge. Moreover, a leaderboard is essential for a challenge and serves as a ranking for researchers to compare the performance of their methods on the level playing field that the benchmark provides. Furthermore, appealing to the academic and scientific nature of such a benchmark and challenge, additional resources should be provided, links to related papers, perhaps a paper detailing the challenge. Lastly, some contact information of the organizers, something like a site notice.
Having identified those crucial constituents, the website developed to host a challenge for the BIG benchmark and possible challenge is made up of the following

[12] `https://react.dev/`
[13] `https://flask.palletsprojects.com/en/stable/`
[14] `https://dev.mysql.com/`

pages: a home page giving the user a quick glance over the challenge and offering navigation elements, a page detailing the challenge, a page containing a summary of the datasets being used for the BIG benchmark, a leaderboard page, a page with information about the evaluation metrics that are used, a page with links to research related to the BIG benchmark and lastly, an About Us page containing relevant contact information.

Submissions: The functionality of making submissions is realized via an upload form when logged in. An account for a team can be created by visiting the register page via clicking a button. A login page also is implemented, from which the user is routed to his profile page. There, he can upload a singular .csv-file and specify some metadata regarding the upload. The .csv-file must contain the predictions for all datasets, including and especially the output probabilities as they're used to calculate the AUC. Based on this .csv-file containing the prediction results on the test sets, the calculation of metrics is realized by the back end. If a submission surpasses a number of best-ranking submissions, it places itself in the leaderboard at the rank corresponding to its ranking for a given classification performance.

Upon having made a submission, some plots based on the submission can be downloaded as well. These include different visualizations of the performance determined by making use of the submitted .csv-file. Providing these plots caters to the aim of facilitating research by making available uniform visualizations of the results.

Cheating prevention: For this thesis it's not possible to withhold any datasets, as there were no non-public datasets available that could have been withheld. The practice of shuffling the datasets can be reverse-engineered, as outlined before. Besides, no infrastructure was available to conduct training on. The main approach of the BIG challenge to combat cheating is prompting participants to submit a link to a peer-reviewed study detailing their training. This makes it possible for submissions to be tested in terms of reproducibility. This approach would also be beneficial for more discourse among the participants of the challenge. Submissions which include a link to a peer-reviewed study will be highlighted in the leaderboard. A submission limit of 10 submissions per day and 100 total submissions per team is also enforced.

Establishing a baseline: The results of the reevaluation of the models trained by Doerrich et al. (2024) will be used as a baseline for the BIG benchmark. Additionally, the mm-PT based models detailed in this thesis also serve to establish a baseline.

3.3 Model Training

The experiments conducted for this thesis in large are split into two parts. First, the training of models using the mm-PT training paradigm described in algorithm 1. Especially the comparison of two architectures, one representative of CNNs and one representative of ViTs is considered here. Second, a reevaluation of the models presented in Doerrich et al. (2024) with regards to the evaluation metrics is conducted and considered in comparison with the models trained on mm-PT. For this, the very models given in the paper were taken and their performance on the three evaluation metrics AUC, BALACC, CO of the BIG benchmark was determined. These two

distinct parts are brought together in an evaluation across training paradigms, with performance comparisons between the mm-PT trained models and the reevaluated models from Doerrich et al. (2024). Those latter models have been trained and evaluated on a single dataset each, representing a contrast to the principle of CNN.

For the mm-PT, bespoke code was created to map the concept of mm-PT. The core functionality of the code corresponding to the mm-PT paradigm is largely based on simply randomly sampling a dataset out of the collection of MedMNIST+ datasets and training the model on the corresponding task. A batch is sampled from those batches which haven't been seen yet during the epoch and is then used to train the model on the task. For this to be as plain of an implementation as it is inside the training loop, a proper model architecture is needed. This is given by using a backbone according to what model one is using, so for ResNet18 a ResNet18 backbone is used. This backbone consists of all layers except for the last classification layer. Using mm-PT as implemented for this thesis, there isn't simply one classification layer, but multiple ones. This multitude of classification heads and with that classification tasks follows the backbone. However, this isn't done in a sequential manner where the heads follow sequentially upon each other, but instead the model is adjusted to use just the backbone and one classification head per batch. This classification head then must correspond to the classification task of the dataset which the batch is sampled from. For updating the parameters of the model, only the backbone as well as the classification head corresponding to the sampled dataset and batch are updated. The other classification heads are simply frozen.

Training setup: In general, all training runs were configured to be as comparable as possible to the training by Doerrich et al. (2024). Just as in their paper, models were downloaded from the Pytorch Image Models (timm)[15] (hug) library. However, the training runs for this thesis did not involve a training on multiple random seeds. One of the random seeds of the paper by Doerrich et al. (2024) was chosen to conduct the training with. It's important to note that only for the training of the ResNet18 and ViT-B/16 model timm was used. The reevaluation isn't a training per se and for the end-to-end and linear probing only required loading the already trained models trained in Doerrich et al. (2024). An overview over the hyperparameters the authors used in the training of the models is given in Table 3. For better comparability, the mm-PT training used the same training configuration, meaning the same hyperparameters and general training setup, as described in the paper. The model training was set to run for 100 epochs at most. Early stopping was utilized with ten epochs and the validation loss as the early stopping criterion. Further variables include that like Doerrich et al. (2024) a Cosine Annealing Learning Rate Scheduler was used. The images were zero-padded to be of size 224×224. Training was performed on a single NVIDIA RTX™A5000 GPU. A CUBLAS environment variable was set upon executing the training script, looking like this.

```
CUBLAS_WORKSPACE_CONFIG=:16:8 python file_name.py --config_file="/pat
h/to/file_name"
```

Setting the CUBLAS environment variable enables deterministic algorithms and

[15] `https://huggingface.co/timm`

thus facilitates reproducibility of results. The loss functions were chosen as chosen by Doerrich et al. (2024) with the Cross-Entropy loss applied to the logits being used for dataset. For binary and multi-class classification, as well as the ordinal regression task. For the multi-label classification however, the Binary Cross-Entropy with logits was used as the loss function.

Weighting: For some models trained for this thesis a weighting was used. This weighting weighted the loss of the smallest dataset with 1 and the other datasets with $\frac{length\ smallest\ dataset}{length\ dataset}$ such that they'd receive a weight smaller than 1, calculated using the length of the smallest dataset in the numerator.

Maximum # epochs	**Early Stopping**	**Learning Rate**
100	10	0.0001
Batch Size	**Seed**	**Optimizer**
64	9930641	AdamW

Table 3: Training Configuration

4 Experiments

4.1 Datasets

4.1.1 MedMNIST v2

MedMNIST v2 comprises 12 datasets for 2D as well as 6 datasets for 3D biomedical imaging. (Yang et al., 2023). It is described by the authors to provide a diverse, standardized and lightweight collection of datasets. Thus there is a wide range of data modalities as well as classification tasks offered in MedMNIST v2. Binary classification, multi-class classification, multi-label classification as well as ordinal linear regression tasks are all part of MedMNIST v2. Cubic spline interpolation was used in resizing all the images to 28x28 pixels, in the case of the 2D images, or 28×28×28, for the 3D images. The authors highlight that this small image size is suited to provide a lightweight collection of datasets for machine learning algorithms. MedMNIST v2 extends its predecessor MedMNIST v1 by adding two datasets for 2D images, hence bringing the total of 2D image datasets from 10 to 12 (Yang et al., 2021). These additional 2D datasets add to the diversity of the domain of images, previously in MedMNIST v1 the images were all in the medical realm while these two additions also include datasets from the bioimaging domain.

4.1.2 MedMNIST+

MedMNIST+ (Yang et al., 2024) extends the MedMNIST v2 datasets by providing a range of resolutions for the images contained in MedMNIST v2. MedMNIST+ does use the 12 datasets in 2D from the biomedical domain that MedMNIST v2 provides but also offers these in four different solutions. So every image from the 2D MedMNIST datasets is provided in the resolutions 28×28, 64×64, 128×128 as well as 224×224. It also extends the 3D images to 28×28×28 and 64×64×64 each. Obtaining these different image sizes for the same pictures for the 12 biomedical 2D image datasets differed from dataset to dataset and included resizing the source images or center-cropping them. There is evidence that with higher resolutions there is the chance of better performance for deep learning models in the biomedical realm (Sabottke and Spieler, 2020), thus MedMNIST+ presents the chance to not just possibly achieve superior performance using the higher resolutions of e.g. 224×224 compared to the 28×28 of MedMNIST v2, but also to compare the performance of models among varying resolutions, further examining the effects of different resolutions on model performance. While Sabottke and Spieler (2020) highlight the superior performance of models trained on higher resolutions, they also acknowledge that training on images with higher resolution in the past had yielded inferior results, possibly due to overfitting stemming from the higher number of features leading to a higher number of parameters being optimized. The authors mention the counter-intuitive observation that in the past models trained on images with lower resolutions fared better than models trained on higher resolutions. The reason they give for this is that before the implementation of residual blocks as in

Name	Data Modality	Task (# Classes)	# Samples	# Training / Validation / Test
PathMNIST	Colon Pathology	MC (9)	107,180	89,996 / 10,004 / 7,180
ChestMNIST	Chest X-Ray	ML (14) BC (2)	112,120	78,468 / 11,219 / 22,433
DermaMNIST	Dermatoscope	MC (7)	10,015	7,007 / 1,003 / 2,005
OCTMNIST	Retinal OCT	MC (4)	109,309	97,477 / 10,832 / 1,000
Pneumonia MNIST	Chest X-Ray	BC (2)	5,856	4,708 / 524 / 624
RetinaMNIST	Fundus Camera	OR (5)	1,600	1,080 / 120 / 400
BreastMNIST	Breast Ultrasound	BC (2)	780	546 / 78 / 156
BloodMNIST	Blood Cell Microscope	MC (8)	17,092	11,959 / 1,712 / 3,421
TissueMNIST	Kidney Cortex Microscope	MC (8)	236,386	165,466 / 23,640 / 47,280
OrganAMNIST	Abdominal CT	MC (11)	58,850	34,581 / 6,491 / 17,778
OrganCMNIST	Abdominal CT	MC (11)	23,660	13,000 / 2,392 / 8,268
OrganSMNIST	Abdominal CT	MC (11)	25,221	13,940 / 2,452 / 8,829

Table 4: MedMNIST2D Dataset Information, adapted from Yang et al. (2023)

the popular ResNet architecture, the models employing deeper architectures faced a higher training loss due to the increased number of model parameters reducing the tractability of optimization.
For the BIG benchmark, only the 2D images of the MedMNIST+ collection of datasets were used.

4.1.3 PathMNIST

In total, 107,180 images of non-overlapping hematoxylin-eosin-stained tissue slides make up the PathMNIST dataset. These histological slides were used by Kather et al. (2019) to build a system aimed at predicting survival from colorectal cancer histology slides. The colored source images were of dimensions $3\times224\times224$. Tissue samples come from the National Center for Tumor diseases in Heidelberg, Germany and the University Medical Center Mannheim in Mannheim, Germany. Hand-delineating single-tissue regions in 86 colorectal cancer tissue slides resulted in more than 100,000 HE image patches which made their way into the final dataset. The

remaining 7,180 images stem from 25 colorectal cancer patients out of the DACHS study (Kather et al., 2019). These colorectal cancer histology slides contain tissue from 9 classes in total, them being adipose tissue, background, debris, lymphocytes, mucus, smooth muscle, normal colon mucuosa, cancer-associated stroma and colorectal adenocarcinoma epithelium.

4.1.4 ChestMNIST

The ChestMNIST dataset (Wang et al., 2017) builds on the NIH ChestXRay dataset and contains frontal-view chest X-Ray images which in Machine Learning are often used to classify thoracic diseases captured in these images or even conduct spatial localization. With its 14 different classes, applying ML methods to ChestMNIST resembles a multi-class classification problem. The possible classifications encompass Atelectasis, Cardiomegaly, Effusion, Infiltration, Mass, Nodule, Pneumonia, Pneumothorax, Consolidation, Edema, Emphysema, Fibrosis, PT and Hernia. In total, ChestMNIST encompasses 112,120 images from 30,805 unique patients. The source images were available in a resolution of 1024×1024 pixels.

4.1.5 DermaMNIST

The HAM10000 challenge (Tschandl et al., 2018) resembles the image basis for DermaMNIST (Yang et al., 2023). In this dataset 10,015 dermatoscopic images are provided from multiple sources, capturing common pigmented skin lesions. There are 7 different diseases captured in the DermaMNIST images, resembling a multi-class classification problem. These 7 different diseases consist of: actinic keratoses, basal cell carcinoma, benign keratosis, dermatofibroma, melanocytic nevi, melanoma, vascular skin lesions. Of these diagnoses for the diseases, Tschandl et al. (2018) state that over 50% have been confirmed by pathology, whereas the rest had been given based on follow-up, expert consensus or lastly in-vivo confocal microscopy. The images were acquired over 20 years from different populations on two different sites, namely Medical University of Vienna, Austria, and the skin cancer practice of Cliff Rosendahl in Queensland, Australia. The Medical University of Vienna began storing images even before digital cameras were widely available and hence the images and metadata existed in various formats such as diapositives which then had to be digitized. Some other, digitally available images had to be extracted from Powerpoint slides or Excel files. Images were generally cropped to center the skin lesions, however, for some images different magnifications or angles were used for the same skin lesions.

4.1.6 OctMNIST

OctMNIST (Kermany et al., 2018) relies on optical coherence tomography images of the retina to examine age-related macular degeneration as well as diabetic macular edema. Initially, 207,130 of those were obtained, however after an image quality

review only about half of those imagges made it into the final dataset, leaving OctMNIST with 109,312 images in total. OctMNIST encompasses 4 different diagnosis categories, namely choroidal neovascularization, diabetic macular edema, drusen and normal. This presents a multi-class classification problem. Source images were of resolutions (384–1,536) × (277–512). Center-cropping was employed for resizing these images. Retrospective cohorts of adult patients were used to obtain the images from, with the time span over which these images were acquired ranging from July 1, 2013 to March 1, 2017. The Shiley Eye Institute of the University of California San Diego, the California Retinal Research Foundation, Medical Center Ophthalmology Associates, the Shanghai First People's Hospital the and Beijing Tongren Eye Center all provided data.

4.1.7 PneumoniaMNIST

The PneumoniaMNIST dataset originally was used by Kermany et al. (2018) to research the generalizability across multiple imaging modalities, having trained their model on the optical coherence tomography images of the OctMNIST dataset. PneumoniaMNIST deals with pediatric pneumonia, so it contains chest X-rays of children of which some had fallen ill with pneunomia. 5,856 of these chest X-ray images of just as many patients make up the PneumoniaMNIST dataset. The classification task is a binary one, with classes being divided in just pneumonia versus normal. The source images were of dimensions (384–2,916) × (127–2,713) and got center-cropped to be part of MedMNIST v2 (Yang et al., 2023).

4.1.8 RetinaMNIST

The RetinaMNIST dataset stems from the DeepDRiD challenge (Liu et al., 2022) and is meant to explore the application of machine learning systems for auto-screening systems in diabetic retinopathy. Diabetic retinopathy stands out as the disease that's caused most frequently by diabetes. Diagnoses are given based on retinal fundus images. Liu et al. (2022) explain how "five levels of DR are distinguished, based on the International Clinical DR (ICDR) classification scale: (1) no apparent retinopathy (grade 0), (2) mild NPDR (grade 1), (3) moderate NPDR (grade 2), (4) severe NPDR (grade 3), and (5) PDR (grade 4)". This leads to RetinaMNIST deviating from the commonly seen binary or multi-class classification and instead offering the chance to perform ordinal regression. This ordinal regression is split into the 5 labels (levels) of diabetic retinopathy as just listed. The entire dataset encompasses 1,600 retinopathy images. Source images were of dimensions 3 × 1,736 × 1,824 and got center-cropped.

4.1.9 BreastMNIST

BreastMNIST (Al-Dhabyani et al., 2020) encompasses 780 images of breast ultrasound scans. The images have been acquired from 600 women over the course of

2018, with the patients having been between the ages of 25 and 75. The source images were acquired at the Baheya Hospital for Early Detection & Treatment of Women's Cancer in Cairo, Egypt using the LOGIQ E9 ultrasound and LOGIQ E9 Agile ultrasound system. The source images were of resolution 500 × 500 pixels. Ultrasound images are inherently grayscale. The data originally was stored in DICOM format and subsequently converted to the PNG format. Originally, the data encompassed 3 classes (normal, benign and malignant), however BreastMNIST poses a binary classification task with the labels normal and benign being combined to form the positive class and the label malignant used to represent the negative class.

4.1.10 BloodMNIST

BloodMNIST (Acevedo et al., 2020) is a dataset made up of microscopic peripheral blood cell images. These hematological images were captured at the Core Laboratory of the Hospital Clinic of Barcelona, using the analyzer CellaVision DM96. Collectively, 17,092 images make up BloodMNIST. Eight different classes are found in the image data: neutrophils, eosinophils, basophils, lymphocytes, monocytes, immature granulocytes (promyelocytes, myelocytes, and metamyelocytes), erythroblasts and platelets or thrombocytes. It's important to note that the individuals which contributed to this dataset did not have any sort of infection, hematologic or oncologic disease and did not undergo pharmacologic treatment, thus providing a blood sample which is not affected by the aforementioned factors. Originally, images were stored in the .jpg-format and were of dimensions 360 × 363, in RGB color space. These images were acquired over a 4-year period from 2015 to 2019.

4.1.11 TissueMNIST

TissueMNIST (Woloshuk et al., 2021) consists of 236,386 images of human kidney tissue. These images were acquired with an upright Leica SP8 Confocal Microscope. Source images came in resolutions of 32×32×7, where 7 is the amount of slices. The images were grayscale and had 8 possible classes resembling epithelial cells from the proximal tubules (PT), thick ascending limbs (TAL), distal convoluted tubules (DCT), and collecting duct (CD), and other cells such as leukocytes (LEUK), podocytes (PODO) and endothelial cells (ENDO) in glomeruli (G) or in the peritubular (P) space. 2D maximum projections were obtained by taking the maximum pixel value along the axial-axis of each pixel (Yang et al., 2023).

4.1.12 OrganMNIST

OrganMNIST describes a family of datasets all originating from the Liver Tumor Segmentation Benchmark (Bilic et al., 2023). The datasets are split into OrganAMNIST, OrganCMNIST and OrganSMNIST, where the letters A, C and S denote the axial, coronal and sagittal view (Yang et al., 2023). This is because these 2D

datasets are sourced from 3D CT images from the Liver Tumor Segmentation Benchmark (LiTS). Yang et al. (2021) describe how they used bounding-box annotations of 11 body organs from another study to obtain the organ labels. They then mention how Hounsfield-Units of the 3D images are transformed into grayscale with an abdominal window. Subsequently, they cropped the 2D images from the center slices of the 3D bounding boxes in the axial, coronal and sagittal views, hence the distinction into OrganAMNIST, OrganCMNIST and OrganSMNIST. The images in the OrganMNIST dataset only differ in the views, so it's the same liver lesions in the datasets but as noted, captured in different planes. Bilic et al. (2023) mention the technical challenges of liver segmentation being varied contrast agents, variations in contrast enhancement due to different injection timing and different acquisition parameters such as resolution, mAs and kVp exposure, reconstruction kernels. Seven clinical institutions contributed to the datasets. These seven institutions are Rechts der Isar Hospital, the Technical University of Munich in Germany, Radboud University Medical Center in the Netherlands, Polytechnique Montréal and CHUM Research Center in Canada, Sheba Medical Center in Israel, the Hebrew University of Jerusalem in Israel, Hadassah University Medical Center in Israel and IRCAD in France. Furthermore, the diversity of the data is emphasized by Bilic et al. (2023), bringing up how images of a diverse set of liver tumor diseases make up the dataset, how the tumors had varying lesion-to-background ratios and how different CT scanners and acquisition protocols, including imaging artifacts (e.g. metal artifacts) are features of the datasets. All three OrganMNIST datasets resemble a multi-class classification problem with 11 classes, however they differ in the number of samples they contain. OrganAMNIST contains 58,850 samples, of which 34,581 are in the training set, 6,491 are in the validation set and 17,778 are in the test set. OrganCMNIST contains 23,660 samples, of which 13,000 are in the training set, 2,392 are in the validation set and 8,268 are in the test set. Finally, OrganSMNIST contains 25,221 samples, of which 13,940 are in the training set, 2,452 are in the validation set and 8,829 are in the test set.

4.2 Experiment 1: comparison of the performance of CNNs and ViTs

Goal: investigate which architecture outperforms the other on the BIG benchmark metrics or if they perform similarly.
Experiment description: experiment 1 is mostly focused on comparing how CNNs perform compared against ViTs using the mm-PT training paradigm by Woerner et al. (2024). For this, both a ResNet18 and a ViT-B/16 model have been trained using mm-PT. Training was conducted over all 12 datasets available in MedMNIST+ with all their four available resolutions. One training run per resolution was performed.
Results: the results are shown in table 5. The best performing model per evaluation metric and resolution is highlighted in bold respectively. One quick glance determines the obvious winner — the model trained with the ViT-B/16 architecture

and no weighting of the loss outperforms the other models across all resolutions and all evaluation metrics, without exception. Even the ViT-B/16 with the weighted loss which is worse than the ViT-B/16 in every regard still outperforms both ResNet18 models across all resolutions and all evaluation metrics, without exception.

Architectures	**AUC**		**BALACC**		**Co κ**	
	28×28	64×64	28×28	64×64	28×28	64×64
ResNet 18	0.7774	0.8154	0.3617	0.3992	0.1870	0.2167
ResNet 18 weighted	0.8092	0.8573	0.4016	0.4571	0.2624	0.3164
ViT-B/16	**0.8986**	**0.9133**	**0.6510**	**0.7000**	**0.5653**	**0.6188**
ViT-B/16 weighted	0.8603	0.8878	0.5720	0.6385	0.4725	0.5531
Architectures	**AUC**		**BALACC**		**Co κ**	
	128×128	224×224	128×128	224×224	128×128	224×224
ResNet 18	0.8388	0.7776	0.4643	0.3890	0.3099	0.1736
ResNet 18 weighted	0.8257	0.7718	0.4369	0.3759	0.3163	0.1709
ViT-B/16	**0.9254**	**0.9314**	**0.7305**	**0.7545**	**0.6355**	**0.6882**
ViT-B/16 weighted	0.8975	0.8963	0.6660	0.6647	0.5735	0.5785

Table 5: Performance of mm-PT end-to-end trained models on resolutions 28×28, 64×64, 128×128, and 224×224, evaluated on AUC, BALACC, and Co κ.

Aside on Training: Training was set up to run for a maximum of 100 epochs as shown in table 3. Also an early stopping criterion of 10 epochs was given, so after 10 epochs throughout which the validation loss didn't improve, early stopping would trigger and the best model would be saved. For the ResNet18 models, this happened in the majority of cases after 11 or 12 total epochs, meaning that the best model was found in either epoch 1 or 2. After at most 2 epochs, the validation loss didn't improve anymore, causing the training to terminate 10 epochs later. The best-performing ResNet18 model, the one trained on images of resolution 128×128 using a weighted loss, effectively reached its best performing model in epoch 2 of training. A side-by-side comparison of the training and validation loss for the best ResNet18 and the best ViT-B/16 is shown in Fig. 15.

Figure 15: Comparison of losses during training

The validation loss for the ResNet18 almost immediately goes upwards, however it does so in a gradual manner, not exploding, not solely going straight up but also putting in a few lower highs which never reach below the loss of epoch one or two. Thus the validation loss diverges from the training loss as it pertains to its trajectory, the training loss continues on a path downwards, declining epoch after epoch. Fig. 21 contains a dataset-wise overview over the loss curves of the best-performing ResNet18 model trained using the mm-PT method. Three possible behaviors of a curve can be seen. For four datasets, namely the three organmnist-based datasets and pneunomiamnist, the validation loss declines and continues to decline, perhaps finding a plateau as close to or past 10 epochs. The second behavior is a type of meandering where the loss generally fluctuates in a range, never breaking out of it, as can be seen for bloodmnist, breastmnist, chestmnist, dermamnist and retinamnist. The third and last behavior is an overall rise in the validation loss as epochs increase, as is the case for octmnist, pathmnist and tissuemnist. These three losses also are bigger than some of the other losses by an order of magnitude.
On the other hand, the losses of the best-performing ViT-B/16 follow a commonly observed trajectory during training, with both the training loss and the validation loss initially declining for multiple epochs until they level off and the validation loss fails to fall below the previous best validation loss for 10 epochs. The dataset-wise overview over the loss curves for the best-performing ViT-B/16 is given in Fig. 22. All curves show a clearly distinguishable decline in validation loss in accordance with the training loss. The chestmnist curve also doesn't appear to meander or stay in a range, the different size of the training and validation loss however make it difficult to discern visually. This represents an outlier, whereas the validation loss for other datasets stayed closer to the training loss, the validation loss for chestmnist is about 75% larger than the training loss without narrowing the gap with further training.

4.3 Experiment 2: effects of resolution on classification performance

Goal: investigate how using higher resolutions affects classification performance in the models used for training.
Experiment description: Consulting table 5 again, the model performance across resolutions can be examined. This is one of the unique selling points of the MedMNIST+ collection of datasets — to provide the datasets in four different resolutions, 28×28, 64×64, 128×128 as well as 224×224.
Results: As stated in section 4.2, experiment 1, the ViT-B/16 architecture with no weighting of the loss outperforms the other models across all resolutions and all evaluation metrics, without exception. This best-performing model always achieves gains in the performance metrics when going from a lower to a higher resolution. This holds for this model for all resolutions and all performance metrics. These gains in AUC achieved by using higher resolutions for ViT-B/16 are smaller in size compared to those in BALACC and CO. The weighted ViT-B/16 is very similar, although gains in the performance metrics can be seen when going from 28×28 to 64×64 and to 128×128, however not when going from resolutions 128×128 to 224×224. There, the model trained on the 128×128 resolution performs better ever so slightly, with improvements mostly being in the second or third digit. The unweighted ResNet also achieves better performance over the same resolutions, going from 28×28 to 64×64 and to 128×128 and then failing to do better when increasing the resolution from 128×128 to 224×224. The worse performance of the higher resolution compared to the model trained on 128×128 images is more pronounced for this ResNet model than it was for the ViT that exhibited this trait. A drop in the AUC by around 0.06 can be observed, a drop of around 9 percent in BALACC and 0.13 in CO. The same relationship exists among the models trained on different resolutions with a weighted ResNet, including a drop in performance from models trained on images of resolution 128×128 to models trained on images of resolution 224×224 of similar magnitude.

4.4 Experiment 3: effects of weighting on classification performance

Goal: investigate how using the weighting described in section 3.3 affects classification performance in the models used for training.
Experiment description: Models using ResNet18 and ViT-B/16 were also trained with the weighting described in section 3.3. This weighting was intended to prevent overfitting on the larger datasets, so their loss was weighted less than the loss of the smaller datasets. A comparison then can be drawn between two models trained on the same architecture, but one with a weighted loss and one without. Only the weighting is different for the models in table 5. This experiment focuses especially on the best performing ViT and ResNet18 models respectively. These were trained on different resolutions, with ViT notching its top performance on images of size

224×224 and ResNet18 performing best on images of size 128×128.
Results: the unweighted version of the ViT-B/16 model outperformed the weighted version throughout the entire benchmark, all resolutions, all metrics. For the ResNet, this isn't so easy. For resolutions 28×28 to 64×64 the weighted version outperforms the unweighted version across all metrics. For the higher resolutions, 128×128 to 224×224, this flips and suddenly the weighted model underperforms the unweighted one. This underperformance then too subsists for all metrics. Fig. 16 shows that the ViT model using the weighted loss on smaller datasets like DermaMNIST, PneumoniaMNIST, BreastMNIST or RetinaMNIST performs worse than the model without weights. The weighted version only edges out the unweighted version on bloodmnist with a 0.9989 AUC versus a 0.99742 AUC.

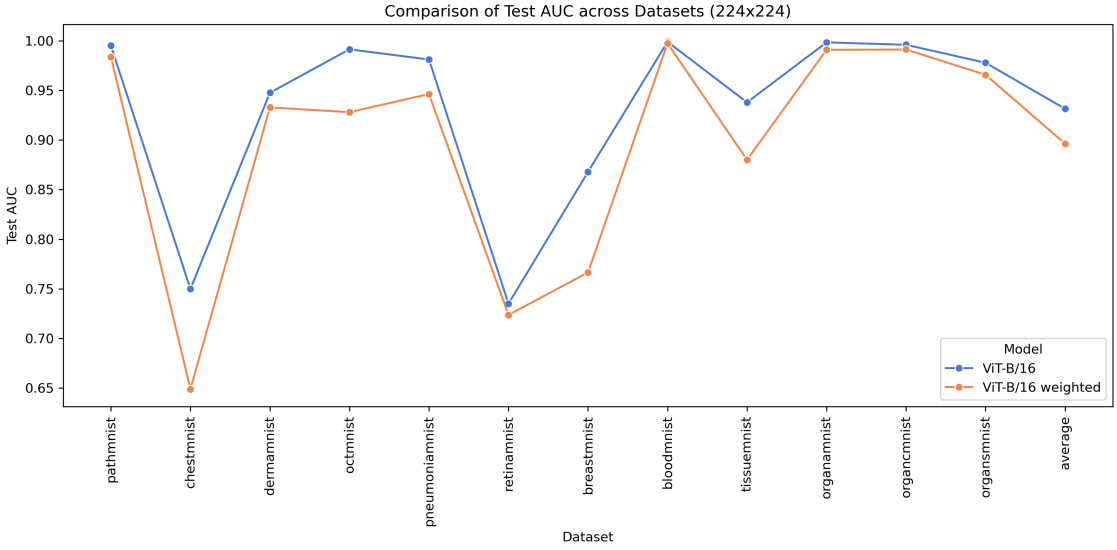


Figure 16: Comparison weighted ViT vs unweighted ViT

For the ResNet models the picture is a different one. While the weighted loss seemed to not help the weighted ViT-B/16 to perform better on smaller datasets at all, this changes for the weighted ResNet18. There is much less of a difference between the weighted and the unweighted ResNet18, with the weighted version even surpassing the unweighted on pneumoniamnist by more than just a slim margin. The weighted ResNet also outperformed the unweighted one on the dermamnist dataset, the retinamnist dataset, the breastmnist dataset, the organcmnist dataset as well as the organsmnist dataset, as can be seen in Fig. 17.

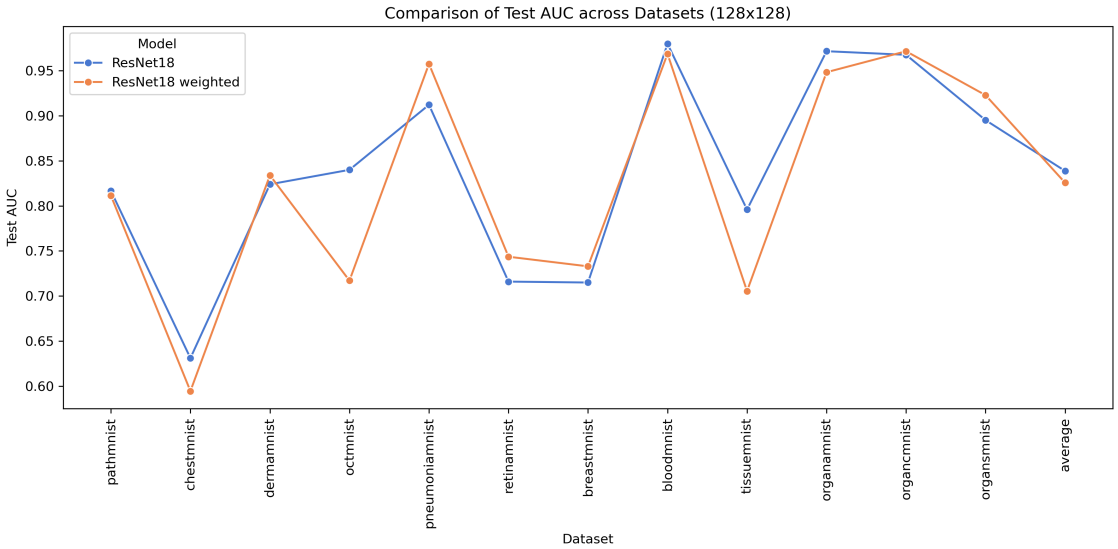


Figure 17: Comparison weighted ResNet18 vs unweighted ResNet18

4.5 Experiment 4: comparison of individually trained models vs mm-PT models

Goal: investigate how a mm-PT trained model fares against models trained on each dataset individually.
Experiment description: A reevaluation of the models presented by Doerrich et al. (2024) is made and compared to the results of the newly trained models presented in section 4.2. The reevaluation is focused on the end-to-end trained models of Doerrich et al. (2024), now offering an overview over the metrics of the BIG benchmark as well. This offers a direct comparison of the mm-PT trained versus the averaged performance of the models trained on each dataset individually.
Results: The mm-PT ViT-B/16 achieves performances comparable to the other ViT-based architectures, surpassing the performance of CLIP across all resolutions and metrics except for the 128×128 Cohen's Kappa. It also beats out SAM and EVA-02 for the AUC across all resoolutions. Generally, also taking a look at the other performanc metrics, the mm-PT ViT-B/16 tends to beat two or three other ViT-architectures. Except for the EfficientNet-B4, which is the worst performing CNN of the models not trained with mm-PT, the mm-PT ViT-B/16 can't quite reach the performance of the CNN-based models, although for some resolutions of AUC and BALACC it's not more than 0.01 or 0.02 away sometimes. The mm-PT ResNet-18 on the other hand underperforms all other models, no matter the architecture, resolution or metric, by a wide margin.

Architectures	AUC		BALACC		Co κ	
	28×28	64×64	28×28	64×64	28×28	64×64
VGG16	0.9266 ± 0.0041	0.9424 ± 0.0028	0.7263 ± 0.0142	0.7598 ± 0.0151	0.6560 ± 0.0156	0.6995 ± 0.0213
AlexNet	0.9114 ± 0.0043	0.9269 ± 0.0034	0.6875 ± 0.0129	0.7375 ± 0.0099	0.6124 ± 0.0145	0.6685 ± 0.0140
ResNet-18	0.9092 ± 0.0028	0.9249 ± 0.0050	0.6805 ± 0.0082	0.7253 ± 0.0101	0.6099 ± 0.0145	0.6620 ± 0.0120
DenseNet-121	0.9175 ± 0.0055	0.9359 ± 0.0023	0.6995 ± 0.0139	0.7448 ± 0.0106	0.6181 ± 0.0194	0.6793 ± 0.0162
EfficientNet-B4	0.8701 ± 0.0084	0.9008 ± 0.0064	0.6120 ± 0.0178	0.6788 ± 0.0165	0.5121 ± 0.0261	0.5898 ± 0.0230
ViT-B/16	0.9054 ± 0.0047	0.9253 ± 0.0069	0.6727 ± 0.0210	0.7324 ± 0.0205	0.5898 ± 0.0230	0.6583 ± 0.0229
CLIP ViT-B/16	0.8922 ± 0.0111	0.9091 ± 0.0051	0.6498 ± 0.0200	0.6990 ± 0.0172	0.5600 ± 0.0303	0.6117 ± 0.0185
EVA-02 ViT-B/16	0.8891 ± 0.0101	0.9053 ± 0.0054	0.6578 ± 0.0164	0.6912 ± 0.0118	0.5717 ± 0.0221	0.5940 ± 0.0138
DINO ViT-B/16	0.9102 ± 0.0048	0.9194 ± 0.0032	0.6781 ± 0.0159	0.7177 ± 0.0139	0.5991 ± 0.0232	0.6398 ± 0.0207
SAM ViT-B/16	0.8901 ± 0.0115	0.9079 ± 0.0101	0.6703 ± 0.0136	0.7096 ± 0.0223	0.5917 ± 0.0153	0.6238 ± 0.0357
mm-PT ViT-B/16	0.8986	0.9133	0.6510	0.7000	0.5653	0.6188
mm-PT ResNet-18	0.7774	0.8154	0.3617	0.3992	0.1870	0.2167
Architectures	**AUC**		**BALACC**		**Co κ**	
	128×128	224×224	128×128	224×224	128×128	224×224
VGG16	0.9516 ± 0.0027	0.9530 ± 0.0022	0.7786 ± 0.0141	0.7812 ± 0.0150	0.7231 ± 0.0172	0.7253 ± 0.0159
AlexNet	0.9429 ± 0.0030	0.9490 ± 0.0023	0.7621 ± 0.0117	0.7724 ± 0.0096	0.7005 ± 0.0125	0.7103 ± 0.0112
ResNet-18	0.9391 ± 0.0027	0.9451 ± 0.0024	0.7565 ± 0.0101	0.7665 ± 0.0079	0.7025 ± 0.0126	0.7150 ± 0.0099
DenseNet-121	0.9457 ± 0.0021	0.9503 ± 0.0023	0.7806 ± 0.0090	0.7869 ± 0.0101	0.7252 ± 0.0092	0.7318 ± 0.0119
EfficientNet-B4	0.9189 ± 0.0039	0.9164 ± 0.0073	0.7151 ± 0.0126	0.7202 ± 0.0134	0.6363 ± 0.0178	0.6401 ± 0.0184
ViT-B/16	0.9325 ± 0.0034	0.9408 ± 0.0038	0.7610 ± 0.0171	0.7716 ± 0.0190	0.6838 ± 0.0195	0.7040 ± 0.0213
CLIP ViT-B/16	0.9151 ± 0.0031	0.9183 ± 0.0058	0.7187 ± 0.0151	0.7212 ± 0.0142	0.6358 ± 0.0187	0.6447 ± 0.0223
EVA-02 ViT-B/16	0.9119 ± 0.0122	0.9260 ± 0.0094	0.7110 ± 0.0290	0.7509 ± 0.0181	0.6058 ± 0.0545	0.6785 ± 0.0245
DINO ViT-B/16	0.9291 ± 0.0036	0.9390 ± 0.0073	0.7513 ± 0.0109	0.7640 ± 0.0180	0.6753 ± 0.0172	0.6889 ± 0.0204
SAM ViT-B/16	0.9194 ± 0.0108	0.9191 ± 0.0055	0.7459 ± 0.0170	0.7432 ± 0.0157	0.6753 ± 0.0212	0.6665 ± 0.0208
mm-PT ViT-B/16	0.9254	0.9314	0.7305	0.7545	0.6355	0.6882
mm-PT ResNet-18	0.8388	0.7776	0.4643	0.3890	0.3099	0.1736

Table 6: Performance of individually trained models (end-to-end) and mm-PT models.

5 Discussion

5.1 Experiment 1

Discussion Reasons for this clear outperformance of the ViT architecture could be sought in the biases and characteristics inherent to CNNs and ViTs. One explanation might be that CNNs apply attention much more locally as compared to ViTs, as described in 2.1.3. This might especially offer an advantage due to the mm-PT training paradigm being employed. Regarding that paradigm it could be beneficial to encode into the model parameters a less local understanding, as if the model attempts to learn the fine intricacies. Perhaps a more global understanding of an image helps especially if there are multiple domains. On the other hand, there is also an argument to be made that ViTs might capture much more detail due to their parameter count which for ViT-B/16 is a multiple of that of a ResNet18. For this refer to table 1.
Outlook Additional experiments could examine the first argument about the more global attention of ViTs. How ViTs encode the images into feature space and whether those images belonging to different datasets are particularly distinctly separated might be an approach for further research. Then for the second possibility, it might be possible to build a comparable architecture to the ResNet-18 architecture which however includes many more parameters, to investigate the effect of parameter count on model performance. As addressed in section 2.2, this is not as simple as just for instance using a VGG16 model which has a more than ten times higher parameter count than a ResNet18, as the differences in architecture might distort results. Thus adapting the ResNet18 architecture to have more parameters or extending another architecture with the residual blocks of ResNets seem to be more reasonable approaches for future research.

5.2 On training

Discussion The training that effectively lasted for just one or two epochs for ResNet18 resembles an unusual case. Part of this might be explained by how the training was conducted. A single optimizer was used for all the heads combined. This is connected to the use of a single Learning Rate Scheduler. If a sample of some dataset 1 for instance is seen using mm-PT and then many samples of dataset 2 and lastly a sample of dataset 1 again, the learning rate might be unfit to adjust the classification head for dataset 1. **Outlook** One approach to improve training could be using a different optimizer for each classification head. Leaving out the Learning Rate Scheduler entirely might improve the course of training as well.

5.3 Experiment 2

Discussion Doerrich et al. (2024) found that in their experiments a higher input resolution tends to yield better classification performance. However, a plateau is

reached at resolution 128×128. The results of experiment 2 corroborate their notion. It is however noteworthy that the model performing best, the ViT-B/16 stands out when examining the idea that a plateau is reached at resolution 128×128. This can't be said to be the case at all, as the improvement in AUC still is 0,006 and the improvement in BALACC is more than two percent, while CO increased by more than 0.05. Taking a closer look at their results, it can be seen that for the ViT-B/16 architecture the same phenomenon can be observed as in experiment 2. The ViT-B/16-based model achieves incremental improvements in classification performance across all metrics and all resolutions, without fail. The gain in performance does abate going from 128×128 to 224×224 as compared to going from 28×28 to 64×64, however there still is a gain. This suggests that ViTs can benefit from using higher resolutions and that they might be capable of making use of the finer information contained in higher resolutions. The gain in performance metric might abate after some point but to reach a true plateau, maybe even higher resolutions are needed.
Outlook Further research could be focused on whether there exists such a boundary, after which an increase in resolution yields no further improvement in classification performance or even a degradation of classification performance. Also qualitative assessments of the information extracted by ViTs as compared to CNNs could be made, so what structures or parts of the image did the model pay attention to. This could offer the chance to explore whether ViTs capture more fine-grained details as the resolution increases as opposed to CNNs or whether something else is responsible for their continued performance improvements.

5.4 Experiment 3

Discussion The weighting did not achieve what it was intended to do with the ViT models. Weighting the smallest datasets more almost caused the opposite effect to what was intended, as some of the smallest datasets register a noticeable drop in classification performance as compared to the larger datasets. However, the underperformance isn't exclusive to only small datasets so there seems to be a more general decline in classification performance due to weighting the ViT model. This seems counter-intuitive as such a weighting in theory would have had the potential to help the model perform better on the smaller datasets at ideally a minimal cost to classification performance on the bigger datasets. However, the especially poor performances on chestmnist, retinamnist and tissuemnist might have to do with the tasks themselves and less the weighting. ChestMNIST is the only multi-label binary classification task and compared to some plain binary classification tasks more more demanding, with 14 possible labels. RetinaMNIST has a unique standing as the only Ordinal Linear Regression Task. For this baseline, it was treated as a multi-class classification task, not using also not using a loss suited for Ordinal Linear Regression tasks. It thus could have been expected that a model trained on the union of the MedMNIST+ datasets would perform especially poor on the RetinaMNIST dataset, which is what happened for both the ResNet as well as the ViT architecture. The outstanding underperformance of ChestMNIST, being worst of all datasets for

both versions of ResNet, weighted and unweighted, and worst for weighted version of ViT might be explained by the complexity of the underlying classification task, as previously explained.

Outlook Given that for the ViT architecture weighting the loss based on the smallest datasets didn't achieve the desired result of the model performing better on those smaller datasets, one other approach would to still build on the idea of weighting the loss but leaving out the datasets with special tasks. Given how the models presented in this paper weren't fully equipped to handle a Ordinal Logistic Regression task, the model could be retrained the same way but without considering the Ordinal Logistic Regression task in the weighting. Meaning this one doesn't get weighted more as compared to other datasets. As a consequence of the implementation that was presented as lacking the proper loss function for Ordinal Logistic regression, it would be easy to address this issue. Simply a proper implementation of this classification task added to the existing code has the potential to improve classification performance and eliminate an outlier that so far is pulling down the performance of the model. Going back to the idea of a weighting, another approach might be to examine which datasets have simpler tasks compared to other datasets and simply weighting the loss of the simpler tasks less. So giving a binary classification on a large dataset a smaller weight than a dataset with a more complex task or less images available could yield an improvement in classification performance. Moreover, addressing the poor performance of ChestMNIST might be as simple as including another multi-label classification task. Both RetinaMNIST and ChestMNIST seem a bit out of place, due to the lack of datasets with similar tasks in MedMNIST+. Extending the collection of datasets with datasets that offer the chance to train on more diverse classification task might improve the performance on datasets which don't have similar classification tasks in the MedMNIST+ collection of datasets.

5.5 Experiment 4

Discussion Two clear results can be taken from the comparison of the mm-PT end-to-end trained models and the individually trained models. Firstly, the results suggest that the mm-PT training paradigm seems to perform competitively when training ViTs. The mm-PT ViT-B/16 achieved competitive results among its ViT peers, although it didn't rank best on any training configuration. Secondly, the data gathered for mm-PT ResNet-18 suggests a stark underperformance, making it appear unfit to consider as a model for mm-PT. **Outlook** The results gathered in this thesis suggest that further research is needed to determine the generalization capability of both ViTs and CNNs as related to both the biomedical domain as well as the mm-PT training paradigm. Training more ViTs using the mm-PT training paradigm, seeking the comparison to its individually trained models might uncover more insights into whether some ViT architectures perform particularly well with the mm-PT training paradigm. It would be interesting to see whether one of the other ViT models could outperform all of the individually trained ViTs from table 6. On

the CNN side further research is needed to determine the suitability of CNNs for the mm-PT approach. The results presented in table 6 seems discouraging when looking how the ResNet got outperformed by a wide margin. An conjecture that could be made based on poor performance of the mm-PT ResNet-18 is that CNNs could be less suited for such generalization tasks. After all, they even outperform most ViTs when trained on the datasets individually. However, tackling the unusual training behavior discussed in section 4.2 might yield a more competitive performance of the mm-PT ResNet-18 architecture. Considering the context of the BIG benchmark and possible challenge, no fine-tuning has really taken place in order to stay as close and comparable as possible to Doerrich et al. (2024). So the models presented in table 6 all could perform vastly different with adjusted hyperparameters. This has the potential to yield exciting new results in the future. Further research could also be focused on the influence of the parameter count of a model as it relates to its generalization capability as well as the different biases inherent to CNNs and ViTs.

6 Conclusion

The application of Deep Learning to the medical domain is continually increasing. New datasets emerge, new architectures spring up and ever more researchers with a non-medical background come into the field. As a result, the possibilities of testing out novel approaches on newly available data rise continuously. One such possibility is the MedMNIST+ collection of datasets, which offers an opportunity to examine the generalization capability of different architectures across different medical domains.

The Biomedical Image Generalization (BIG) benchmark is introduced in order to capitalize on the possibilities that the MedMNIST+ collection of datasets offers. The proposed benchmark offers a chance to investigate the generalization capabilities of models on biomedical image data. Moreover, many exciting comparisons can be made as it relates to resolution, training on multiple datasets at once and evaluating the trained models based on different evaluation metrics.
There is the opportunity of hosting a challenge based on the BIG benchmark, building upon the bespoke website detailed in this thesis. A baseline has been contributed in part based on a reevaluation of already trained models and in part based on the models for which the training was detailed. Cheating prevention will remain a topic of discussion, as there doesn't seem to be a silver bullet to combat cheating when laying down the rules for a competition centered around the BIG benchmark. The proposed benchmark relies in part on the peer-reviewing process to deal thwart cheating. Submission limits are also a cost-effective solution that is implemented to deter malicious actors.
The ViT-based model trained using mm-PT seemed especially suited for generalization on the varying biomedical domains of the BIG benchmark, especially as compared to its CNN counterpart the ResNet-18 using mm-PT. However, there might be ways to improve the seemingly underwhelming performance of the mm-PT trained CNN in further research. The mm-PT trained ViT-B/16 model achieved competitive results compared against its ViT peers which were trained on the individuals datasets, with their performance metrics having been averaged. Some of the individually trained ViT peers were even outperformed, with the mm-PT trained model presented placing somewhere in the middle of the field among the other ViTs. It remains a topic for further research how different architectures affect a model's generalization capability across medical domains. This could focus on both the inherent properties and biases of the competing architectures as well as simple properties like the parameter count of a model.
In conclusion, the bespoke website presented for the newly introduced BIG benchmark offers the possibility to leverage the unique possibilities provided by the MedMNIST+ collection of datasets. Some evidence has been gathered that a ViT-based model trained for generalization can perform at a similar level compared to other ViT-based models which were individually trained on the datasets. With the BIG benchmark, future research can investigate the generalization capabilities of models further, fine-tuning the baseline that this thesis has contributed to.

A Appendix

A.1 VDD Dataset Overview

Name	# of Images	# of categories	RGB	Citation
FGVC-Aircraft Benchmark	10,000	100	yes	Maji et al. (2013)
CIFAR100	60,000	100	yes	Krizhevsky et al. (2009)
Daimler Mono Pedestrian Classification	50,000	2	no	Munder and Gavrila (2006)
Describable Texture Dataset	5,640	47	yes	Cimpoi et al. (2014)
German Traffic Sign Recognition	51,840	43	yes	Stallkamp et al. (2012)
Flowers102	8,189	102	yes	Nilsback and Zisserman (2008)
ILSVRC12	1,200,000	1000	yes	Russakovsky et al. (2015)
Omniglot	32,460	1623	no	Lake et al. (2015)
The Street View House Numbers	over 600,000	10	yes	Netzer et al. (2011)
UCF101	see textual description	101	yes	Soomro et al. (2012)

Table 7: Overview of the datasets used in the Visual Domain Decathlon

A.2 MSD Dataset Overview

Phase	Task	Modality	Source	# Cases (Train/Test)
Development Phase	Brain	mp-MRI	BraTS 2016 & 2017 (Bakas et al., 2018)	750 4D volumes (484/266)
	Heart	MRI	LASC (Gomez et al., 2015)	30 3D volumes (20/10)
	Hippo-campus	MRI	Vanderbilt University Medical Center, Nashville, US	394 3D volumes (263/131)
	Liver	CT	LiTS (Bilic et al., 2023)	210 3D volumes (131/79)
	Lung	CT	Lung and lung cancer data can	96 3D volumes (64/32)
	Pancreas	CT	Memorial Sloan Kettering Cancer Center, New York, US	420 3D volumes (282/139)
	Prostate	mp-MRI	Radboud University Medical Center, Nijmegen, The Netherlands	48 4D volumes (32/16)
Mystery Phase	Colon	CT	Memorial Sloan Kettering Cancer Center, New York, US	190 3D volumes (126/64)
	Hepatic Vessels	CT	Memorial Sloan Kettering Cancer Center, New York, US	443 3D volumes (303/140)
	Spleen	CT	Memorial Sloan Kettering Cancer Center, New York, US	61 3D volumes (41/20)

Table 8: Overview of datasets used in the Medical Segmentation Decathlon

A.3 WILDS Dataset Overview

Dataset	Metric	Setting
iWILDCAM2020-WILDS	Macro F1	Domain gen.
CAMELYON17-WILDS	Average acc	Domain gen.
RxRx1-WILDS	Average acc	Domain gen.
OGB-MOLPCBA	Average AP	Domain gen.
GLOBALWHEAT-WILDS	Average domain acc	Domain gen.
CIVILCOMMENTS-WILDS	Worst-group acc	Subpop. shift
FMoW-WILDS	Worst-region acc	Hybrid
POVERTYMAP-WILDS	Worst-U/R Pearson R	Hybrid
AMAZON-WILDS	10th percentile acc	Hybrid
PY150-WILDS	Method/class acc	Hybrid

A.4 Resnet Architecture Visualization

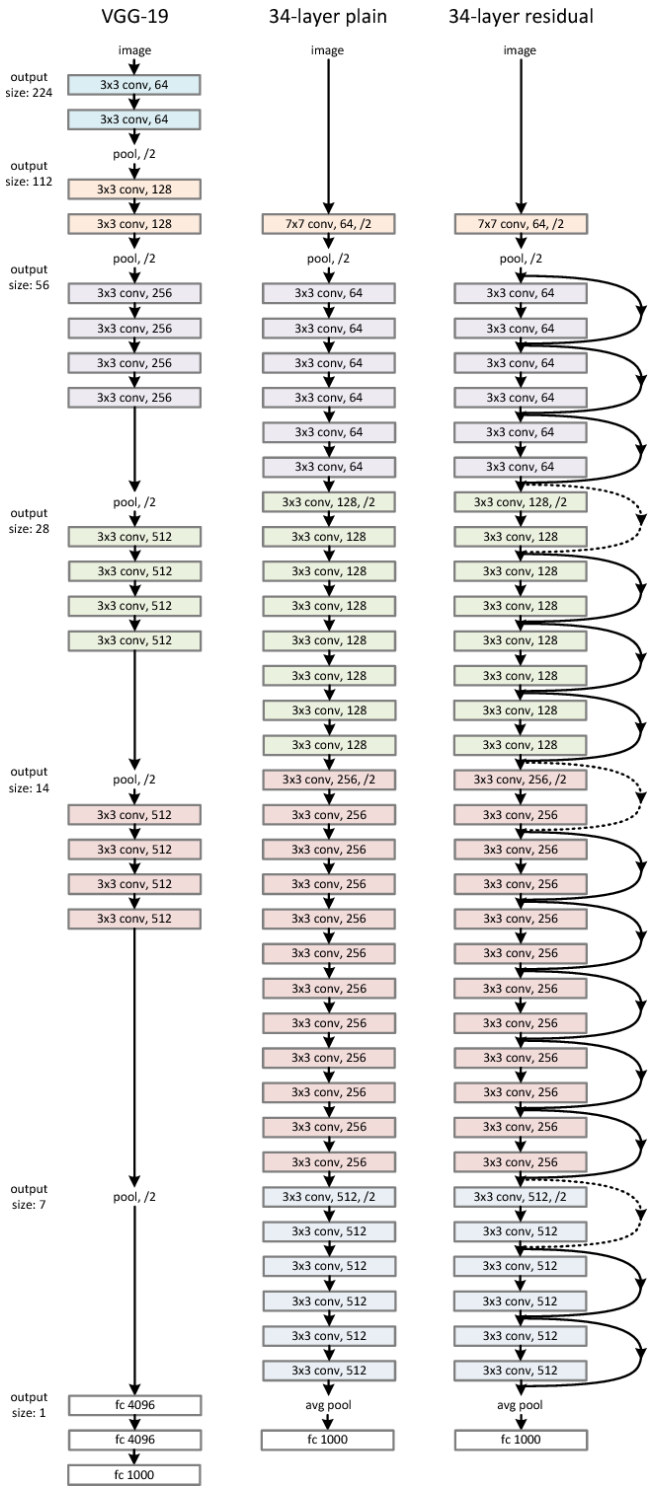


Figure 18: Example network architectures, reprinted from He et al. (2016)

A.5 DenseNet Architecture Specification

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

Figure 19: DenseNet architectures for ImageNet (Huang et al., 2017)

A.6 EfficientNet Compound Scaling Method

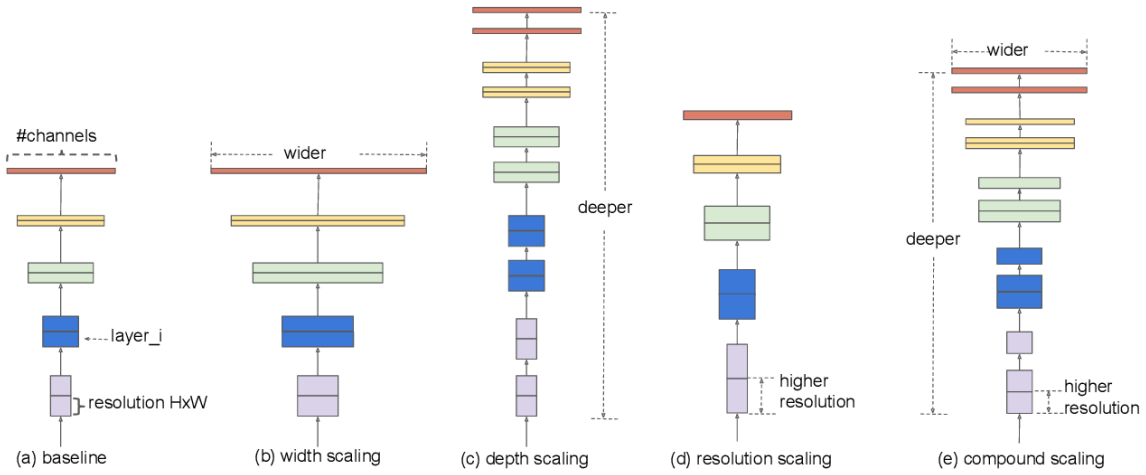


Figure 20: Model scaling, reprinted from Tan (2019)

A.7 Pseudocode for the mm-PT training paradigm

Algorithm 1 Multi-domain Multi-task Pre-training

Require: Set T of tasks, Datasets D_t for tasks $t \in T$, Model f with parameters θ, Classification heads l_t for $t \in T$

while training has not converged **do**
  $t \leftarrow$ Sample a task t from T with probability $\frac{|D_t|}{\sum_{j \in T} |D_j|}$
  $B \leftarrow$ Sample a batch from dataset D_t
  $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}_t \left(l_t \left(f(B; \theta) \right) \right)$
end while
return θ

A.8 Dataset-wise loss curves for training of ResNet18 (128×128)

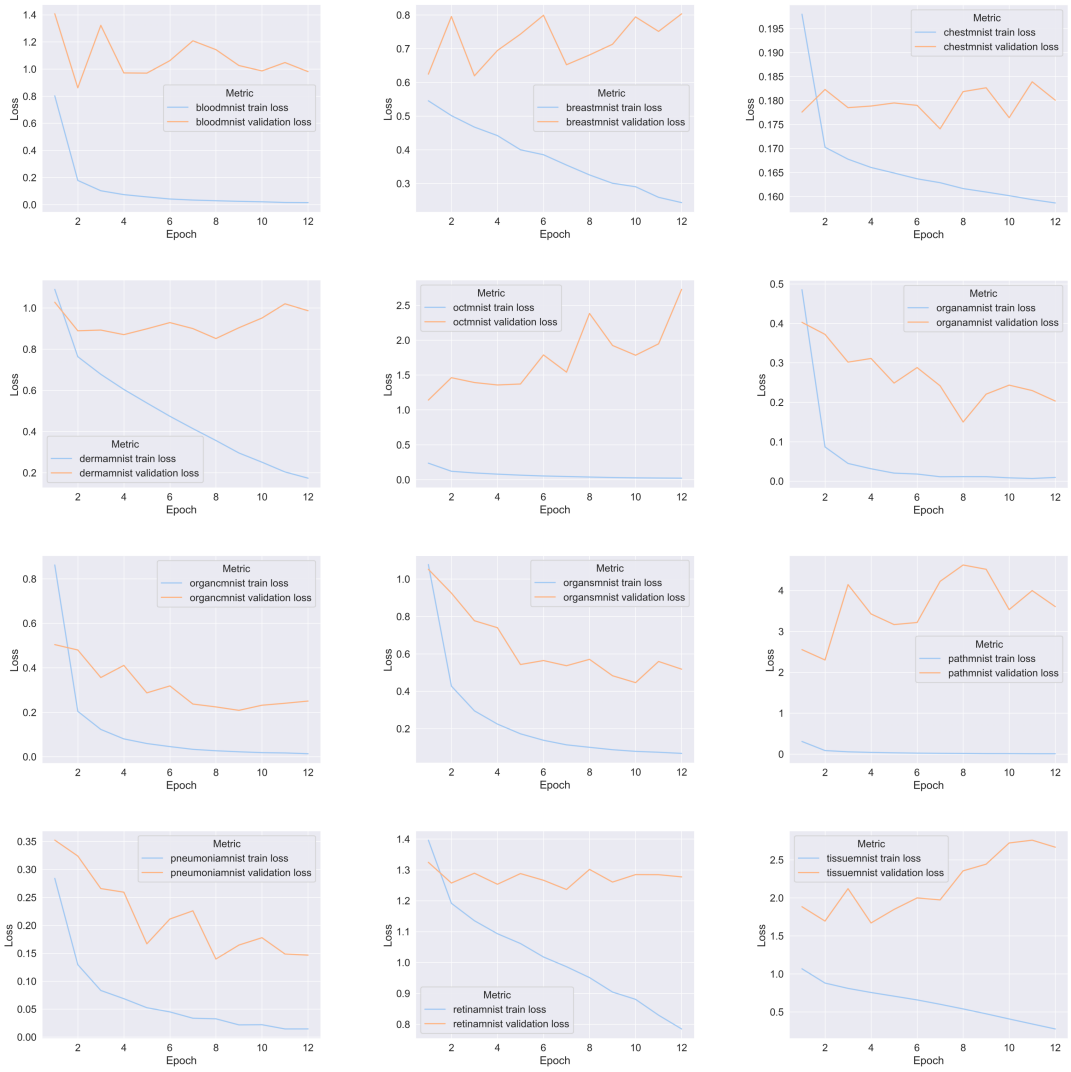


Figure 21: Overview of loss curves for ResNet18 (128×128)

A.9 Dataset-wise loss curves for training of ViT-B/16 (224×224)

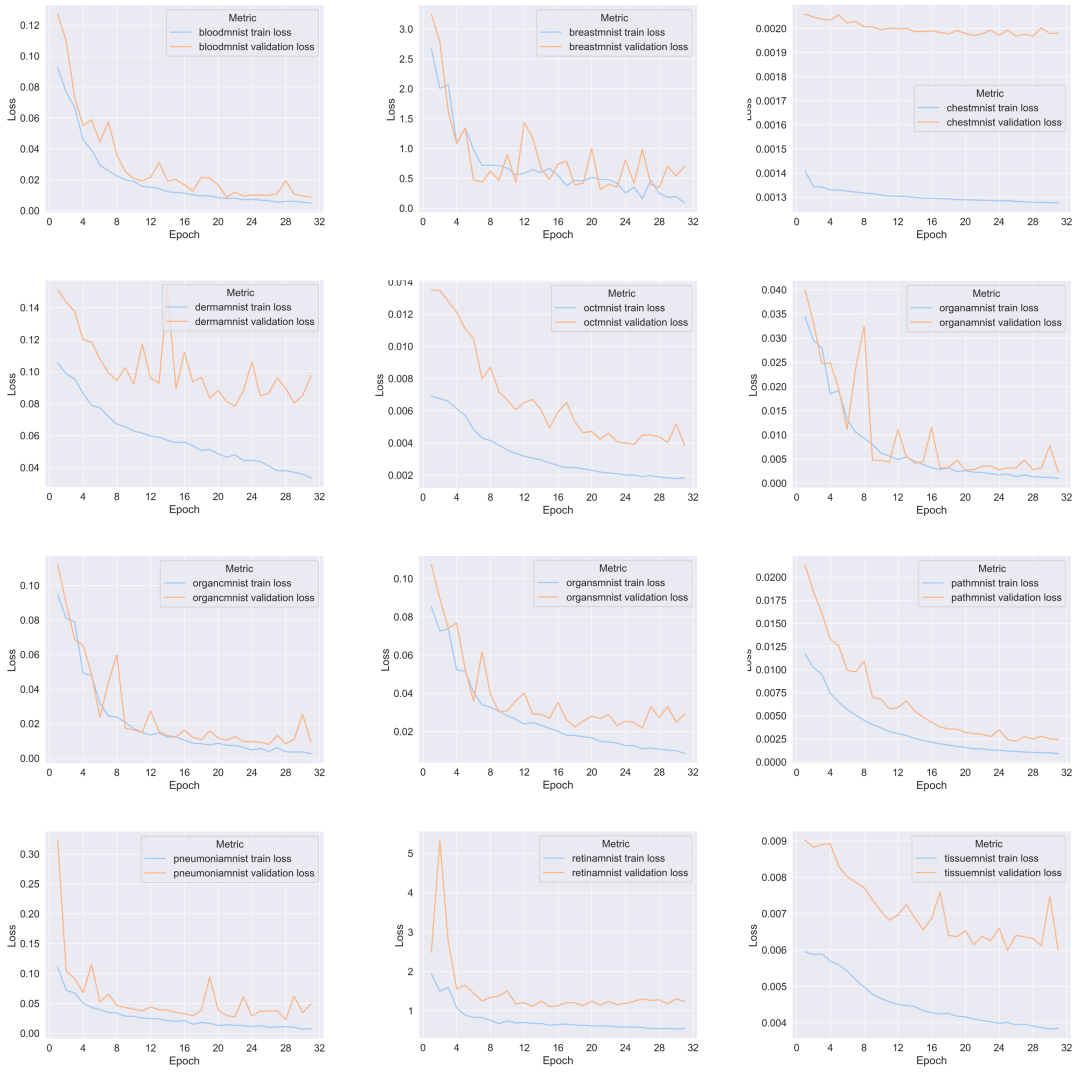


Figure 22: Overview of loss curves for ViT-B/16 (224×224)

Bibliography

CodaLab - Competition — zeus.robots.ox.ac.uk. https://zeus.robots.ox.ac.uk/competitions/competitions/9#results. [Accessed 12-10-2024].

Visual Decathlon Challenge — robots.ox.ac.uk. https://www.robots.ox.ac.uk/~vgg/decathlon/#res. [Accessed 12-10-2024].

Welcome to The Cancer Imaging Archive - The Cancer Imaging Archive (TCIA) — cancerimagingarchive.net. https://www.cancerimagingarchive.net/.

Welcome to Flask &x2014; Flask Documentation (3.0.x) — flask.palletsprojects.com. https://flask.palletsprojects.com/en/stable/. [Accessed 27-10-2024].

Grand Challenge — grand-challenge.org. https://grand-challenge.org/. [Accessed 18-10-2024].

timm (PyTorch Image Models) — huggingface.co. https://huggingface.co/timm. [Accessed 28-10-2024].

Kaggle: Your Machine Learning and Data Science Community — kaggle.com. https://www.kaggle.com/. [Accessed 12-10-2024].

https://medicaldecathlon.com/results/. [Accessed 12-10-2024].

MySQL :: Developer Zone — dev.mysql.com. https://dev.mysql.com/. [Accessed 27-10-2024].

React — react.dev. https://react.dev/. [Accessed 27-10-2024].

3.4. Metrics and scoring: quantifying the quality of predictions — scikit-learn.org. https://scikit-learn.org/stable/modules/model_evaluation.html. [Accessed 14-10-2024].

Leaderboard submissions — wilds.stanford.edu. https://wilds.stanford.edu/submit/, a. [Accessed 11-10-2024].

WILDS — wilds.stanford.edu. https://wilds.stanford.edu/, b. [Accessed 27-10-2024].

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, 2020. ISSN 2352-3409. doi: https://doi.org/10.1016/j.dib.2020.105474. URL https://www.sciencedirect.com/science/article/pii/S2352340920303681.

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

Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

Michela Antonelli, Annika Reinke, Spyridon Bakas, Keyvan Farahani, Annette Kopp-Schneider, Bennett A Landman, Geert Litjens, Bjoern Menze, Olaf Ronneberger, Ronald M Summers, et al. The medical segmentation decathlon. *Nature communications*, 13(1):4128, 2022.

André Araujo, Wade Norris, and Jack Sim. Computing receptive fields of convolutional neural networks. *Distill*, 4(11):e21, 2019.

Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

Patrick Bilic, Patrick Christ, Hongwei Bran Li, Eugene Vorontsov, Avi Ben-Cohen, Georgios Kaissis, Adi Szeskin, Colin Jacobs, Gabriel Efrain Humpire Mamani, Gabriel Chartrand, et al. The liver tumor segmentation benchmark (lits). *Medical Image Analysis*, 84:102680, 2023.

Andrea Bocchieri, Lorenzo V Mugnai, Enzo Pascale, Quentin Changeat, and Giovanna Tinetti. Detecting molecules in ariel low resolution transmission spectra. *Experimental Astronomy*, 56(2):605–644, 2023.

Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. *arXiv preprint arXiv:1911.03584*, 2019.

Hal Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

Sebastian Doerrich, Francesco Di Salvo, Julius Brockmann, and Christian Ledig. Rethinking model prototyping through the medmnist+ dataset collection. *arXiv preprint arXiv:2404.15786*, 2024.

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. *ArXiv*, abs/2010.11929, 2020. URL `https://api.semanticscholar.org/CorpusID:225039882`.

Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva: Exploring the limits of masked visual representation learning at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19358–19369, 2023.

Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva-02: A visual representation for neon genesis. *Image and Vision Computing*, 149:105171, 2024.

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*, pages 770–778, 2016.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Mohammad Hossin and Md Nasir Sulaiman. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1, 2015.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

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*, pages 4700–4708, 2017.

Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

Jakob Nikolas Kather, Johannes Krisam, Pornpimol Charoentong, Tom Luedde, Esther Herpel, Cleo-Aron Weis, Timo Gaiser, Alexander Marx, Nektarios A Valous, Dyke Ferber, et al. Predicting survival from colorectal cancer histology slides using deep learning: A retrospective multicenter study. *PLoS medicine*, 16(1): e1002730, 2019.

Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan,

et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *cell*, 172(5):1122–1131, 2018.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*, pages 5637–5664. PMLR, 2021a.

Pang Wei Koh, Shiori Sagawa, et al. WILDS: A benchmark of in-the-wild distribution shifts. `https://github.com/p-lambda/wilds`, 2021b. Version 2.0.0.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266): 1332–1338, 2015.

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

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

Ruhan Liu, Xiangning Wang, Qiang Wu, Ling Dai, Xi Fang, Tao Yan, Jaemin Son, Shiqi Tang, Jiang Li, Zijian Gao, et al. Deepdrid: Diabetic retinopathy—grading and image quality estimation challenge. *Patterns*, 3(6), 2022.

Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on mri. *Zeitschrift für Medizinische Physik*, 29 (2):102–127, 2019.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Mary L McHugh. Interrater reliability: the kappa statistic. *Biochemia medica*, 22 (3):276–282, 2012.

Stefan Munder and Dariu M Gavrila. An experimental study on pedestrian classification. *IEEE transactions on pattern analysis and machine intelligence*, 28(11): 1863–1868, 2006.

Johannes C. Myburgh, Coenraad Mouton, and Marelie H. Davel. Tracking translation invariance in cnns. In Aurona Gerber, editor, *Artificial Intelligence Research*, pages 282–295, Cham, 2020. Springer International Publishing.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 4. Granada, 2011.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE, 2008.

Adrien Pavao, Isabelle Guyon, Anne-Catherine Letournel, Dinh-Tuan Tran, Xavier Baro, Hugo Jair Escalante, Sergio Escalera, Tyler Thomas, and Zhen Xu. Codalab competitions: An open source platform to organize scientific challenges. *Journal of Machine Learning Research*, 24(198):1–6, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in neural information processing systems*, 34:12116–12128, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.

Nicholas Roberts, Samuel Guo, Cong Xu, Ameet Talwalkar, David Lander, Lvfang Tao, Linhang Cai, Shuaicheng Niu, Jianyu Heng, Hongyang Qin, et al. Automl decathlon: Diverse tasks, modern methods, and efficiency at scale. In *NeurIPS 2022 Competition Track*, pages 151–170. PMLR, 2023.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

Carl F Sabottke and Bradley M Spieler. The effect of image resolution on deep learning in radiography. *Radiology: Artificial Intelligence*, 2(1):e190015, 2020.

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*, pages 4510–4520, 2018.

Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19(1):221–248, 2017.

Bibo Shi, Rui Hou, Maciej A. Mazurowski, Lars J. Grimm, Yinhao Ren, Jeffrey R. Marks, Lorraine M. King, Carlo C. Maley, E. Shelley Hwang, and Joseph Y. Lo. Learning better deep features for the prediction of occult invasive disease in ductal carcinoma in situ through transfer learning. In *Medical Imaging*, 2018. URL `https://api.semanticscholar.org/CorpusID:3638594`.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Satya P Singh, Lipo Wang, Sukrit Gupta, Haveesh Goli, Parasuraman Padmanabhan, and Balázs Gulyás. 3d deep learning on medical images: a review. *Sensors*, 20(18):5097, 2020.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. A dataset of 101 human action classes from videos in the wild. *Center for Research in Computer Vision*, 2(11):1–7, 2012.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

J Su, Y Lu, S Pan, A Murtadha, B Wen, and Y Liu Roformer. Enhanced transformer with rotary position embedding., 2021. *DOI: https://doi. org/10.1016/j. neucom*, 2023.

Mingxing Tan. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

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):1–9, 2018.

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)*, pages 3462–3471, 2017. doi: 10.1109/CVPR.2017.369.

Michael Widenius and David Axmark. *MySQL reference manual: documentation from the source*. " O'Reilly Media, Inc.", 2002.

Stefano Woerner, Arthur Jaques, and Christian F Baumgartner. A comprehensive and easy-to-use multi-domain multi-task medical imaging meta-dataset (medimeta). *arXiv preprint arXiv:2404.16000*, 2024.

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, et al. In situ classification of cell types in human kidney tissue using 3d nuclear staining. *Cytometry Part A*, 99(7):707–721, 2021.

Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9:611–629, 2018.

Jiancheng Yang, Rui Shi, and Bingbing Ni. Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis. In *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pages 191–195. IEEE, 2021.

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.

Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. [medmnist+] 18x standardized datasets for 2d and 3d biomedical image classification with multiple size options: 28 (mnist-like), 64, 128, and 224, 2024.

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.

Bamberg, 28.10.2024
Place, Date

M. Bachmeier
Signature