UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CNN and Transformers-based Diagnosis of Dermatological Diseases

Diploma Thesis

Karagkounis Theodoros

Supervisor: Dimitrios Katsaros

Septemeber 2025

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CNN and Transformers-based Diagnosis of Dermatological Diseases

Diploma Thesis

Karagkounis Theodoros

Supervisor: Dimitrios Katsaros

Septemeber 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Διάγνωση Δερματολογικών Παθήσεων με Συνελικτικά Νευρωνικά Δίκτυα και Μετασχηματιστές

Διπλωματική Εργασία

Καραγκούνης Θεόδωρος

Επιβλέπων: Δημήτριος Κατσαρός

September 2025

Approved by the Examination Committee:

Supervisor **Dimitrios Katsaros**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Dimitrios Rafailidis**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Georgios Thanos**
Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

First of all I want to thank my family for the support they gave me during my academic career. Thanks to my parents I have the opportunity to study at the school where I am today.

Additionally, I would like to express my gratitude to Associate Professor Dimitrios Katsaros, my supervisor, for his leadership and the time he spent giving me insightful counsel and directions on how to finish my thesis. His assistance has greatly influenced how my work has turned out. Additionally, I would like to express my gratitude to Georgios Thanos of the Laboratory Teaching Staff and Associate Professor Dimitrios Rafailidis for their contributions to my thesis review committee.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Karagkounis Theodoros

Diploma Thesis

CNN and Transformers-based Diagnosis of Dermatological Diseases

Karagkounis Theodoros

Abstract

Early and accurate diagnosis of dermatological diseases is a critical factor for the prevention and effective treatment of skin cancer. In this paper, we study the application of modern deep learning techniques for image classification of the HAM10000 dataset, which includes skin images from various disease categories. Two main approaches are compared: the widely known Convolutional Neural Networks (CNNs), which are well established in the field of image classification, and Vision Transformers (ViTs), a more recent architecture that exploits attention mechanisms for image processing. The models were trained and evaluated using common performance metrics such as accuracy. The results highlight the comparative advantages of each approach and suggest that Transformers can achieve competitive or even superior performance compared to Convolutional Neural Networks , especially in scenarios where classes do not have the same amount of data and the morphology of the images is very complex. This work is an example of how new technologies such as deep learning ,can contribute to the field of medicine. Finally it demonstrates that with the development of artificial intelligence , the diagnosis for any patient will be made with the help of an algorithm. //

Keywords: Convolutional Neural Networks,Vision Transformers,Classification,Accuracy

Διπλωματική Εργασία

Διάγνωση Δερματολογικών Παθήσεων με Συνελικτικά Νευρωνικά Δίκτυα και Μετασχηματιστές

Καραγκούνης Θεόδωρος

Περίληψη

Η έγκαιρη και ακριβής διάγνωση δερματολογικών παθήσεων αποτελεί κρίσιμο παράγοντα για την πρόληψη και αποτελεσματική αντιμετώπιση του καρκίνου του δέρματος. Στην παρούσα εργασία μελετάται η εφαρμογή σύγχρονων τεχνικών βαθιάς μάθησης για την ταξινόμηση εικόνων του HAM10000 dataset, το οποίο περιλαμβάνει εικόνες από το δέρμα από διάφορες κατηγορίες ασθενειών. Συγκρίνονται δύο κύριες προσεγγίσεις: τα ευρέως γνωστά Συνελικτικά Νευρωνικά Δίκτυα (CNNs), τα οποία έχουν εδραιωθεί στον τομέα της ταξινόμησης των εικόνων και οι Μετασχηματιστές Όρασης (ViTs), μια πιο πρόσφατη αρχιτεκτονική που αξιοποιεί μηχανισμούς προσοχής για την επεξεργασία εικόνων. Τα μοντέλα εκπαιδεύτηκαν και αξιολογήθηκαν με χρήση κοινών μετρικών απόδοσης όπως η ακρίβεια. Τα αποτελέσματα αναδεικνύουν τα συγκριτικά πλεονεκτήματα κάθε προσέγγισης και υποδεικνύουν ότι οι Μετασχηματιστές μπορούν να επιτύχουν ανταγωνιστικές ή και ανώτερες επιδόσεις σε σύγκριση με τα Συνελικτικά Νευρωνικά Δίκτυα , ιδιαίτερα σε σενάρια που οι κλάσεις δεν έχουν τον ίδιο αριθμό δεδομένων και η μορφολογία των εικόνων είναι πολύ σύνθετη. Η εργασία αυτή αποτελεί παράδειγμα πως οι νέες τεχνολογιές όπως η βαθιά μάθηση ,μπορούν να συνεισφέρουν στον τομέα της ιατρικής. Τέλος αποδεικνύει πως με την εξέλιξη της τεχνητης νοημοσυνης , η διαγνωση για οποιαδηποτε ασθενεια θα γινεται με την βοηθεια ενός αλγορίθμου.

Λέξεις-κλειδιά: Συνελικτικά Νευρωνικά Δίκτυα,Μετασχηματιστές Όρασης,Ταξινόμηση,Ακρίβεια

Table of contents

List of figures

List of tables

Abbreviations

AI	Artificial Intelligence
CNN	Convolutional Neural Network
ViT	Vision Transformer
ML	Machine Learning
ReLu	Rectidied Linear Unit
MHAN	Multi-head Attention Network
MLP	Multi-Layer Perceptrons
GELU	Gaussian Error Linear Unit

Chapter 1

Introduction

1.1 AI and Medicine

Artificial intelligence in medicine [1] is defined as the use of machine learning models to search medical data to find information and patterns that help improve health outcomes and patient experience. In artificial intelligence, computers learn through experience, using large amounts of data. AI is therefore a building block of modern medicine, as there is a wealth of data in the healthcare industry. In addition, the various AI algorithms and applications created for the purpose of supporting healthcare professionals have multiple benefits in research and clinical trials. A key application of AI in medicine is Disease Detection and Diagnosis. Machine learning models can be used to observe the vital signs of critically ill patients

1.2 Aim and purpose of the work

The aim of this thesis is the study, implementation and evaluation of deep learning models for the classification of skin images in order to improve the accuracy of automatic diagnosis of medical conditions. In particular, two state-of-the-art approaches to image processing are investigated: convolutional neural networks [2] (CNNs) and vision transformers [3] (ViTs). In addition, a custom convolutional neural network was designed and trained from scratch on the dataset, in order to compare its performance with that of a pretrained CNN model. This comparison highlights how pretrained architectures converge faster and achieve higher accuracy, while models trained from scratch generally require more epochs and careful tuning to approach competitive performance.

The main objectives of the work include:

- Understanding the theoretical principles underlying CNNs and Vision Trans-formers.

- The processing and analysis of the HAM10000 dataset [4], which contains dermoscopic images from seven different categories of skin diseases.
- The implementation and training of different deep learning [5] models for the multiclass classification [6] problem of the HAM10000 dataset, including:
  - A pre-trained ResNet-50 model [7].
  - A constructed CNN model designed from scratch for the needs of the work.
  - Google Vision Transformer (ViT-B/16), based on a 16×16 patch size, utilizing pre-trained weights.
- Comparing the performance of the models using appropriate metrics (accuracy, precision, recall, F1-score).
- To draw conclusions about the suitability of each architecture for medical image diagnosis applications.
- To suggest possible improvements or extensions for future research.

This work seeks to enhance the understanding of how modern artificial intelligence models, such as Neural Networks and Transformers , can be effectively applied to medical image classification. The aim is for AI to act as an aid to the doctor, not as a replacement.

1.3 Thesis Structure

This thesis is divided into six chapters. Chapter 2 gives an overview of convolutional neural networks `(CNNs)` and Vision Transformers `(ViTs)`, explaining what they are and how they function. Chapter 3 talks about the dataset used in this project, highlighting why it's important and how the quality and balance of the data influence the performance of the models. In Chapter 4, the three models used in the study are described in detail, including their structure, whether they were trained using pre-existing data or from scratch, and the datasets they were originally trained on. Chapter 5 covers optimization methods like the type of loss function, gradient descent, and learning rate, all of which are important during the training process. Chapter 6 explains the evaluation methods used to measure model performance, including test accuracy, classification reports with visual results for each model, and ideas for future improvements.

Chapter 2

Theoretical Background

2.1 CNN

Tens or hundreds of layers make up a convolutional neural network, and each one learns to identify distinct aspects of an image. Each training image is subjected to various resolutions of filters, and the output of each convolved image serves as the input for the subsequent layer. The filters can begin as very basic features like edges and brightness and progress in complexity to qualities that give the object a unique identity.

A CNN is composed of an input layer, an output layer, and many hidden layers in between.

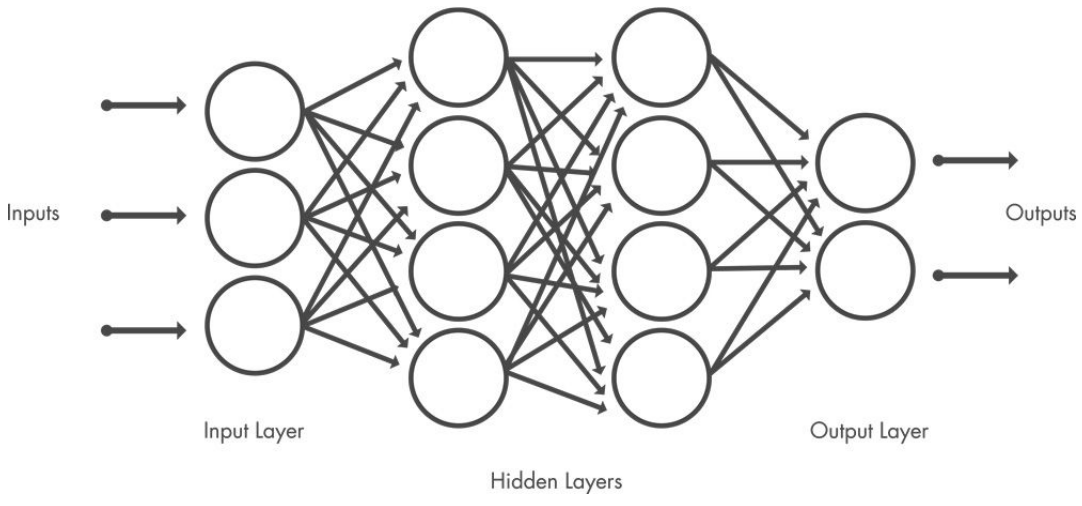


Figure 2.1: Neural Network

[8]

These layers perform operations that modify the data with the purpose of learning features specific to the data. Three of the most common layers are convolution [9], activation or

ReLU [10], and pooling [11].

- Convolution activates specific features in the input images by passing them through a series of convolutional filters.

- Rectified linear unit (ReLU) allows for faster and more effective training by mapping negative values to zero and maintaining positive values. This is sometimes referred to as activation, because only the activated features are carried forward into the next layer.

- Pooling simplifies the output by performing nonlinear downsampling, reducing the number of parameters that the network needs to learn.

Each layer learns to recognize distinct features as these procedures are repeated across tens or hundreds of levels.

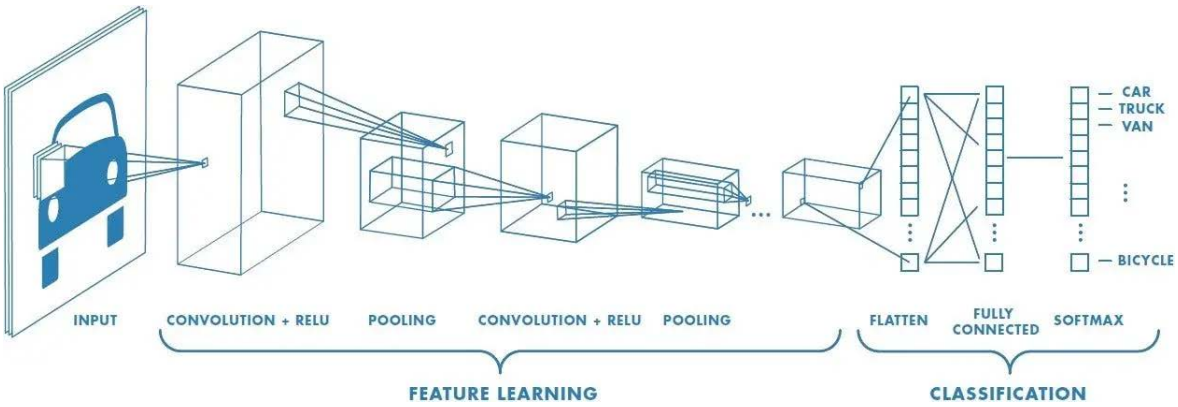


Figure 2.2: CNN

[12]

A CNN differs from a traditional neural network in that each hidden neuron in a layer has the same weights and bias values [2]. This indicates that the same feature, like an edge or blob, is being detected by every hidden neuron in various areas of the image. As a result, the network can tolerate objects in an image being translated. A network that has been taught to identify autos, for instance, will be able to do so in any image where the car is present.

Classification Layer [2]: The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

2.2 ViT

Vision Transformer (ViT) came out as a competitive option to CNNs that are currently state-of-the-art in computer vision and widely used for different image recognition tasks. ViT models outperform the current state-of-the-art CNNs by almost four times in terms of computational efficiency and accuracy [3]. Vision transformers are widely used in common image recognition tasks as action identification, object detection, picture segmentation, and classification even in more exotic applications [13].

Let us take a closer look at the architecture of the vision transformer.

- 1. Divide a picture into patches.
- 2. Make the patches flat.
- 3. From the flattened patches, create lower-dimensional linear embeddings.
- 4. Include embeddings for positions
- 5. Provide the sequence to a conventional transformer encoder as an input.
- 6. Use picture labels to pretrain the model (completely supervised on a large dataset).
- 7. Focus on picture categorization using the downstream dataset.

Image patches are the sequence tokens (like words). Also the ViT encoder is made up of several blocks, each of which has three main processing components:

- Layer Norm
- Multi-head Attention Network (MHAN)
- Multi-Layer Perceptrons (MLP)

Layer Norm [14] maintains the training process on course. by allowing the model to adjust to the differences between the training images. The network in charge of creating attention maps from the provided embedded visual tokens is called a Multi-head Attention Network (MHAN). The network can concentrate on the most important areas of the image, like the object or objects, with the aid of these attention maps. Attention maps share the same notion as saliency maps and alpha-matting, which are found in the traditional computer vision literature.

The GELU [15] is the final component of the two-layer classification network known as MLP [16]. The transformer uses the last MLP block, commonly known as the MLP head, as an output. Classification labels can be obtained by applying softmax to this output (i.e., if the application is Image Classification).

2.2.1 How do Vision Transformers work?

At first the input image is is converted to square patches. For example suppose we have an image of size 32 x 32 pixels with 3 color channels (RGB).

Step 1: Split the image into patches

Let's say we use a patch size of 16 × 16 pixels. That means:

- The image is split into 32/16= 2 patches per dimension
- So in total: 2×2=4 patches

Each patch has shape: 16 × 16 × 3 = 768 values .

Step 2: Flatten each patch

Each 16×16×3 patch is flattened into a 1D vector of 768 values. This vector contains all pixel values (for R, G, and B) from the patch placed in a single line.

Step 3: Each vector becomes a token Now you have:

- 4 vectors (one for each patch)
- Each vector is 768-dimensional These vectors are treated as tokens — just like words in a sentence for a language model.

Step 4 : Add Positional Encodings

One problem with Transformers is that the sequence order is not enforced naturally since data is passed in at a shot instead of timestep-wise. To combat this, the Visi-on Transformer uses Positional Encodings/Embeddings that establish a certain or-der in the inputs.

The positional embedding matrix Epos is randomly generated and added to the concate-nated matrix containing the learnable class embedding and patch projec-tions.

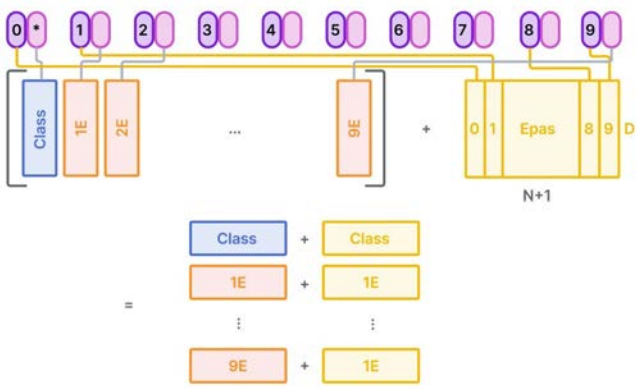


Figure 2.3: Positional embeddings in ViT

[17]

Throughout the Transformer, D is the fixed latent vector size. Before sending the input vectors to the encoder, we squish them to this. These positional embeddings and patch projections come together to create a larger matrix that will shortly be passed through the Transformer Encoder.

Use Transformer layers to process the picture token sequence. The Transformer encoder processes each patch, which is now represented as a token. Each patch learns about other patches in the image with the aid of the self-attention mechanism. For instance, the patch that displays an automobile's wheel will pick up information from the patch that displays the vehicle's body.

ViT employs a unique categorization token after processing the picture tokens. To achieve the final classification, this token combines the global data from every patch and runs it through a basic neural network (e.g., identifying the item in the image).

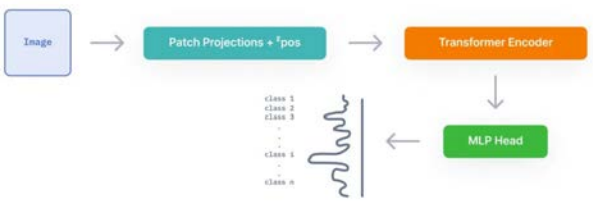


Figure 2.4: ViT Encoder

[17]

Finally, the whole diagram of ViT looks like this

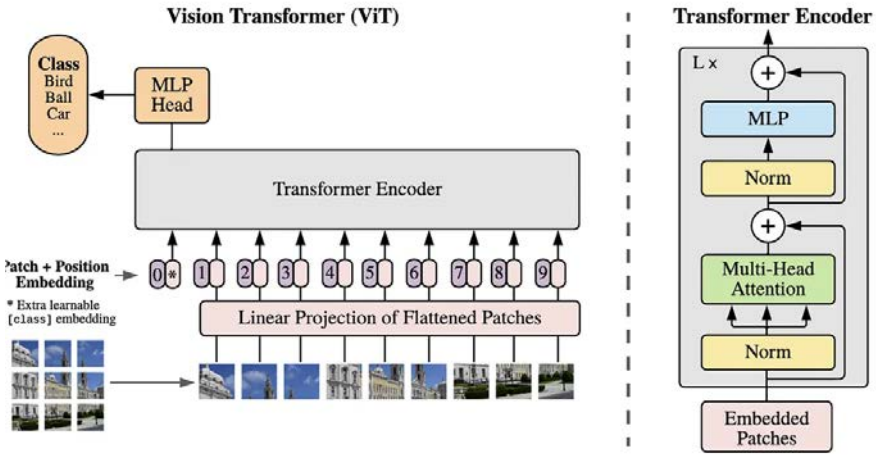


Figure 2.5: ViT

[18]

2.3 Differences between CNN and VIT

For image processing applications, Vision Transformers (ViTs) have become a potent substitute for Convolutional Neural Networks (CNNs), upending CNN dominance [19]. Knowing how these two architectures differ from one another is essential for AI and computer vision practitioners and students. The main distinctions, benefits, drawbacks, and uses of Vision Transformers and CNNs are examined in this blog post.

Feature Extraction :

- Vision Transformer (ViT): ViTs model the links between pictures by treating them as a series of patches and using self-attention techniques. As a result, ViTs are better able to capture global context and long-range dependencies.
- CNN: CNNs use convolutional layers to scan the image with filters and identify local patterns such as textures and edges. CNNs progressively develop a global understanding of the image as more layers are added.

Global Context:

- Vi-si-on Transformer (ViT): ViTs are able to represent global interactions between patches in a single layer because they employ self-attention throughout the image.
- CNNs need several layers in order to construct hierarchical features, ranging from global features that represent complete objects to local features like edges.

Positional Encoding:

- Vi-si-on Transformer (ViT): ViTs use explicit positional encodings to track the location of each patch in the image because the Transformer design is indifferent to the tokens' position.
- CNN: CNNs indirectly maintain positional information by encoding spatial hierarchies through their convolutional structure.

Performance on Small Datasets:

- Vision Transformer (ViT): Unless pre-trained on big datasets or in conjunction with data augmentation approaches, ViTs have a tendency to overfit on small datasets.
- CNN: Because CNNs rely on local features and have fewer learnable parameters than ViTs, they typically perform better on small datasets

Applications of Vision Transformers and CNNs: Vision Transformers (ViT):

- Image Classification: When trained on massive datasets such as ImageNet, Vision Transformers have demonstrated state-of-the-art performance in image classification tasks.
- Object Detection: When modified for object detection tasks, ViT-based models such as DETR [20] have demonstrated competitive performance in comparison to CNN-based models.
- Medical Image Segmentation: Vision Transformers are being used for fine-grained tasks such as tumor segmentation in medical imaging.

Convolutional Neural Networks (CNNs):

- Image Classification: CNNs have been the go-to architecture for image classification for years, with models like ResNet and Inception leading the way.
- Face Recognition: CNNs are widely used in facial recognition systems due to their ability to detect and recognize features across varying scales.
- Object Detection and Segmentation: CNN-based models are widely used for object detection and image segmentation tasks.

Chapter 3

Dataset

3.1 The Crucial Role of Datasets in Neural Network Performance

In the realm of artificial intelligence and machine learning, neural networks have emerged as powerful tools capable of solving complex problems across diverse fields—from image recognition and natural language processing to medical diagnostics and autonomous driving. At the heart of every successful neural network lies one critical component: the dataset. The quality, quantity, and relevance of the data used to train a neural network are often more influential than the architecture or the algorithms themselves. A well-curated dataset enables a model to learn meaningful patterns, generalize effectively to unseen data, and make accurate predictions. Conversely, poor or biased data can lead to overfitting, underperformance, and even ethical concerns. Therefore, understanding the importance of datasets is not just a technical necessity, but a foundational aspect of developing robust and trustworthy neural network models.

3.2 Data that were used in Project

For this project, we used the HAM10000 [4] dataset, a large and publicly available collection of dermatoscopic images of pigmented skin lesions. The dataset was specifically designed to support the development and evaluation of machine learning models in dermatology, with an emphasis on the classification of skin cancer and other types of skin lesions.

Cases include a representative collection of all important diagnostic categories in the

realm of pigmented lesions: Actinic keratoses and intraepithelial carcinoma / Bowen's disease (akiec), basal cell carcinoma (bcc), benign keratosis-like lesions (solar lentigines, seborrheic keratoses, and lichen planus-like keratoses, bkl), dermatofibroma (df), melanoma (mel), melanocytic nevi (nv), and vascular lesions (angiomas, angiokeratomas, pyogenic granulomas, and hemorrhage, vasc).

The images are stored in two files: HAM10000_images_part_1 and part_2. I downloaded the dataset from Kaggle. The dataset also contains a metadata.csv file, in which every sample includes 7 columns. The names of the columns and their meanings are:

- **image_id**: The image_id in the metadata file corresponds directly to the filenames of the image files stored in the HAM10000 dataset, across the two image folders, with the extension .jpg.
- **lesion_id**: Multiple images can correspond to the same lesion (i.e., the same skin spot, taken from different angles or at different times).
- **dx**: Diagnosis label (ground truth), one of 7 categories: akiec, bcc, bkl, df, nv, mel, vasc. This column is very important because each diagnosis label represents a different class.
- **dx_type**: Type of labeling method (e.g., histo for histopathology, consensus, or follow_up).
- **age**: Age of the patient.
- **sex**: Gender of the patient (male, female).
- **localization**: Anatomical site of the lesion (e.g., back, lower extremity, face).

.

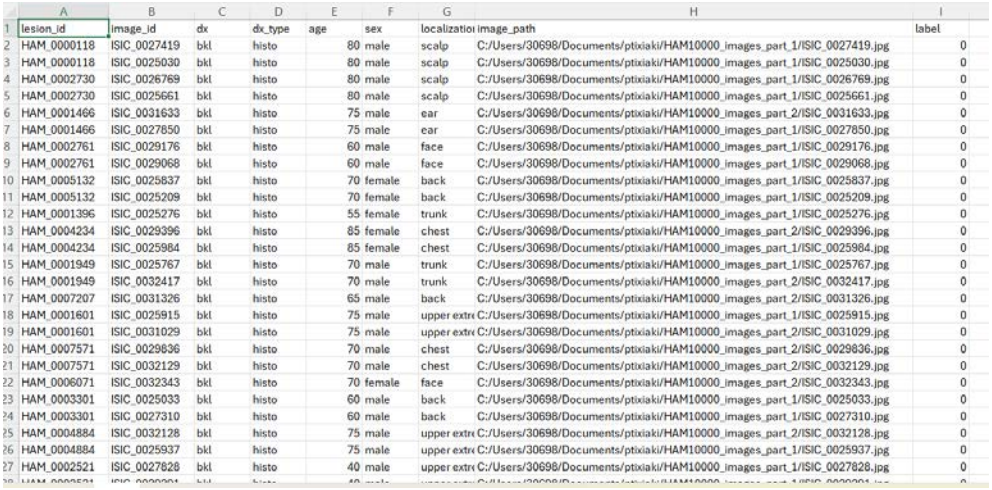
3.3 Preproccesing of the data

To facilitate efficient image loading and ensure reliable linkage between the metadata and the corresponding image files, it was important to enrich the metadata with a new column containing the absolute file path for each image. Since the images in the HAM10000 dataset are distributed across two separate directories (HAM10000_images_part_1/ and

HAM10000_images_part_2/), a Python script was developed to search both folders for each image based on its image_id.

The script iterated through the metadata file, matched each image_id to the corresponding .jpg file in either folder, and, upon locating it, inserted the full path into a new column called image_path. After this preprocessing step, the dataset was split into 80% for training and 20% for testing. From the training portion, 12.5% was further set aside for validation, ensuring that each subset preserved the original class distribution.

Below is an example of how the metadata file looks after adding the image_path column.



	A	B	C	D	E	F	G	H	I
1	lesion_id	image_id	dx	dx_type	age	sex	localizatio	image_path	label
2	HAM_0000118	ISIC_0027419	bkl	histo	80	male	scalp	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0027419.jpg	0
3	HAM_0000118	ISIC_0025030	bkl	histo	80	male	scalp	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0025030.jpg	0
4	HAM_0002730	ISIC_0026769	bkl	histo	80	male	scalp	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0026769.jpg	0
5	HAM_0002730	ISIC_0025661	bkl	histo	80	male	scalp	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0025661.jpg	0
6	HAM_0001466	ISIC_0031633	bkl	histo	75	male	ear	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_2/ISIC_0031633.jpg	0
7	HAM_0001466	ISIC_0027850	bkl	histo	75	male	ear	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0027850.jpg	0
8	HAM_0002761	ISIC_0029176	bkl	histo	60	male	face	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0029176.jpg	0
9	HAM_0002761	ISIC_0029068	bkl	histo	60	male	face	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0029068.jpg	0
10	HAM_0005132	ISIC_0025837	bkl	histo	70	female	back	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0025837.jpg	0
11	HAM_0005132	ISIC_0025209	bkl	histo	70	female	back	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0025209.jpg	0
12	HAM_0001396	ISIC_0025276	bkl	histo	55	female	trunk	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0025276.jpg	0
13	HAM_0004234	ISIC_0029396	bkl	histo	85	female	chest	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_2/ISIC_0029396.jpg	0
14	HAM_0004234	ISIC_0025984	bkl	histo	85	female	chest	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0025984.jpg	0
15	HAM_0001949	ISIC_0025767	bkl	histo	70	male	trunk	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0025767.jpg	0
16	HAM_0001949	ISIC_0032417	bkl	histo	70	male	trunk	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_2/ISIC_0032417.jpg	0
17	HAM_0007207	ISIC_0031326	bkl	histo	65	male	back	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_2/ISIC_0031326.jpg	0
18	HAM_0001601	ISIC_0025915	bkl	histo	75	male	upper extre	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0025915.jpg	0
19	HAM_0001601	ISIC_0031029	bkl	histo	75	male	upper extre	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_2/ISIC_0031029.jpg	0
20	HAM_0007571	ISIC_0029836	bkl	histo	70	male	chest	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_2/ISIC_0029836.jpg	0
21	HAM_0007571	ISIC_0032129	bkl	histo	70	male	chest	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_2/ISIC_0032129.jpg	0
22	HAM_0006071	ISIC_0032343	bkl	histo	70	female	face	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_2/ISIC_0032343.jpg	0
23	HAM_0003301	ISIC_0025033	bkl	histo	60	male	back	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0025033.jpg	0
24	HAM_0003301	ISIC_0027310	bkl	histo	60	male	back	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0027310.jpg	0
25	HAM_0004884	ISIC_0032128	bkl	histo	75	male	upper extre	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_2/ISIC_0032128.jpg	0
26	HAM_0004884	ISIC_0025937	bkl	histo	75	male	upper extre	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0025937.jpg	0
27	HAM_0002521	ISIC_0027828	bkl	histo	40	male	upper extre	C:/Users/30698/Documents/ptixiaki/HAM10000_images_part_1/ISIC_0027828.jpg	0

Figure 3.1: Data Excel File

In the context of our multi-class image classification project using the HAM10000 dataset, the two most important columns from the metadata are image_path and dx. The image_path column contains the full path to the image file, which is essential for loading the image data during training and inference. The dx column is equally important, as it serves as the target label for classification and indicates the diagnosed skin condition for each image. In addition, the lesion_id column plays a crucial role, as it ensures that images originating from the same lesion are not distributed across different splits (train, validation, test). This prevents data leakage, which would otherwise allow the model to "see" highly similar images of the same lesion in both training and testing phases, artificially inflating performance. For example, if two images with different image_id values but the same lesion_id (IMG_001, IMG_002 from lesion L123) were placed in separate splits, the model could unfairly benefit from this overlap.

3.3.1 Label Encoding for Classification

In order to prepare the dataset for training deep learning models, I created a new column in the metadata that maps each diagnostic category (`dx`) to a corresponding integer label. This process, known as label encoding, is essential because neural networks operate more efficiently with numerical values rather than string-based class names. Alternatively, the dataset could have been preprocessed into a binary classification scheme, where class `0` represents benign diseases and class `1` represents cancerous cases. However, this simplification was not preferred, as the goal was to preserve the full multi-class structure of the dataset in order to evaluate model performance across all diagnostic categories.

The following mapping was applied:

- `bkl` $\rightarrow$ 0
- `nv` $\rightarrow$ 1
- `df` $\rightarrow$ 2
- `mel` $\rightarrow$ 3
- `vasc` $\rightarrow$ 4
- `bcc` $\rightarrow$ 5
- `akiec` $\rightarrow$ 6

This mapping was stored in a new column named `label`, which was then used throughout the training pipeline for model input and loss calculation. By converting categorical values to integers, it becomes possible to apply functions like cross-entropy loss, which require class indices as input rather than string labels.

3.3.2 Addressing Class Imbalance in the HAM10000 Dataset with Augmentation and Oversampling

Before passing the image data to the training models, it became evident that the dataset used for this project is highly imbalanced, as shown in Figure X. The majority of the samples belong to class 1 (corresponding to the diagnosis `nv` – melanocytic nevi), while other classes such as `df`, `vasc`, and `akiec` have significantly fewer samples. Such an imbalance can result in biased models with disproportionately high performance on the majority class and poor sensitivity to minority classes. This is particularly problematic in medical image analysis, where accurate identification of rare conditions is crucial.

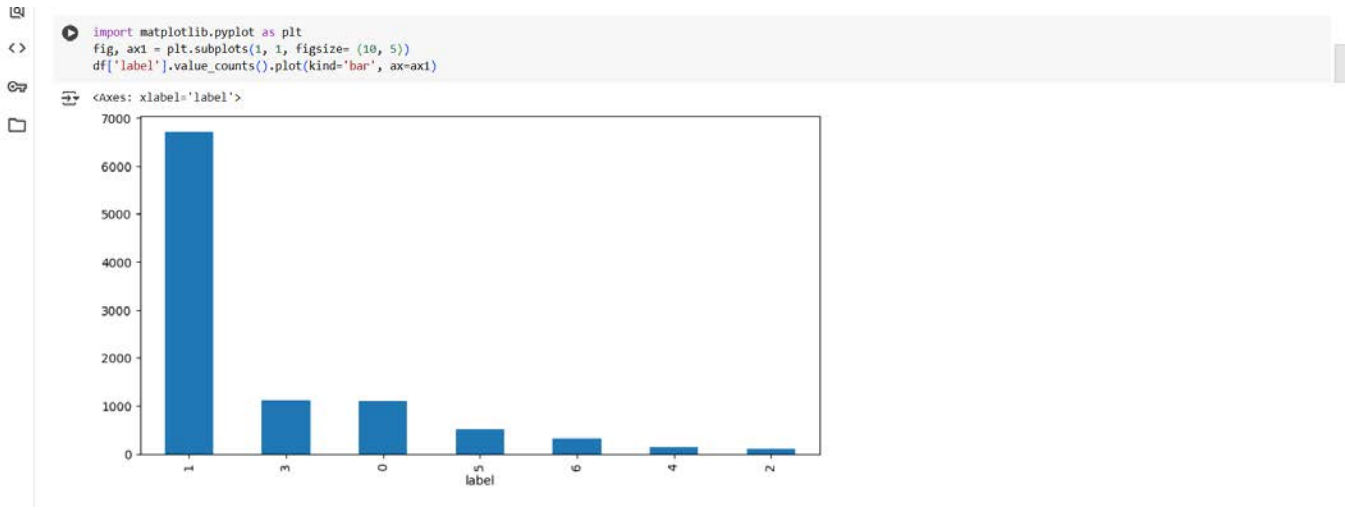


Figure 3.2: Bar Plot

The bar plot 3.2 clearly illustrates the extent of this imbalance, with an overwhelming proportion of the data falling under the `nv` class. Without addressing this issue, the model would likely underperform on underrepresented classes, limiting its real-world clinical usefulness.

To mitigate this issue, I applied a combination of data augmentation and oversampling. A custom Python [21] pipeline was developed to generate synthetic examples for minority classes using a variety of augmentation techniques. These included Gaussian blurring [22], perspective distortion, color jitter, rotation, affine transformation, and random flipping. The intensity of augmentations was adjusted depending on the severity of underrepresentation:

- For severely underrepresented classes (e.g., classes 2, 3, and 6), a more advanced augmentation pipeline was applied.
- For the remaining classes, a simpler yet effective augmentation strategy was used.

The augmentation process generated new synthetic samples and appended them to the training set, while maintaining the correct image-label association by preserving the integer label in the newly added rows.

This oversampling approach helped rebalance the dataset and significantly improved the model's ability to learn from all classes more equitably, especially the rare but clinically important ones.

Chapter 4

Models Architectures

4.1 Models that were used

Three distinct deep learning models `ResNet50` [7], one Vision Transformer more especially, `Google's ViT-Patch16` [23], and finally one CNN from —were applied in this study. A popular convolutional neural network, `ResNet50` is renowned for using residual connections to aid in the training of extremely complex designs. A more recent method is the Vision Transformer `(ViT-Patch16)`, which uses transformer-based structures that were first created for natural language processing and treats photos as collections of patches. Finally, a bespoke CNN was used to investigate a customized solution for the `HAM100000` dataset that balances complexity and performance.

4.2 ResNet-50

The *ResNet-50* [7] architecture is a member of the ResNet (Residual Network) series, which was presented in 2015 by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun from Microsoft Research Asia. *ResNets* were created to address the degradation issue that may arise during the training of deep neural networks. They presented the idea of residual learning via shortcut (or skip) connections.

The *ResNet-50* model features 50 layers and has emerged as one of the most popular deep learning frameworks because of its effective blend of depth, precision, and computational performance. It was first trained and assessed on the extensive *ImageNet* [24] dataset. Since its debut, ResNet-50 has become a standard backbone for tasks involving image classification

and is widely utilized and refined in multiple computer vision applications.

4.2.1 ResNet and Residual Blocks

The main issue that ResNet addressed was the *degradation problem* [25]in deep neural networks. As networks grow deeper, their accuracy initially improves but eventually saturates and then rapidly declines. This decline is not primarily due to overfitting, but rather to difficulties in the training process itself.

ResNet mitigates this issue through the introduction of *Residual Blocks* [25], which incorporate skip (or shortcut) connections that allow information to bypass certain layers. These skip connections help preserve gradient flow during backpropagation, effectively alleviating the vanishing gradient problem and enabling the successful training of very deep networks.

The specific residual block implemented in ResNet-50 is known as the *Bottleneck Residual Block*. This block follows the architecture:

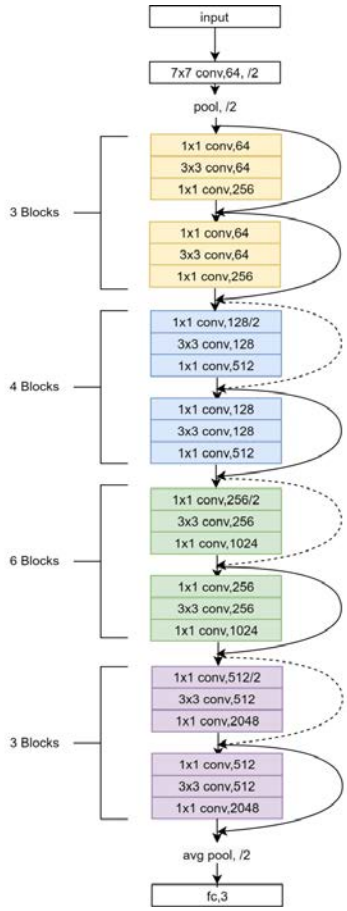


Figure 4.1: ResNet

[26]

Below is an overview of the elements present in the bottleneck residual block used in ResNet-50:

- **ReLU Activation:** The ReLU (Rectified Linear Unit) activation function is applied after every convolutional and batch normalization layer. ReLU allows only positive values to pass through, introducing non-linearity into the network, which is essential for learning complex patterns in the data.
- **Bottleneck Convolution Layers:** The block consists of three convolutional layers, each followed by batch normalization and a ReLU activation function.
  - The *first* convolutional layer typically uses a 1×1 filter to reduce the number of channels in the input, thereby compressing the data and improving computational efficiency while retaining most of the essential information.
  - The *second* convolutional layer uses a 3×3 filter to extract spatial features from the data.
  - The *third* convolutional layer again uses a 1×1 filter to restore the original channel count, preparing the output to be added to the shortcut connection.
- **Skip Connection:** A defining feature of residual blocks, the skip (or shortcut) connection allows the original input to bypass the convolutional layers and be added directly to their output. This mechanism ensures that essential information is preserved and transmitted through the network, even if the intermediate convolutional layers struggle to learn useful features.

Bottleneck residual blocks in ResNet-50 effectively address the vanishing gradient problem by combining deep feature extraction with identity mappings, and by incorporating dimensionality reduction through bottleneck layers. This design enables the successful training of much deeper networks and contributes to ResNet's high performance in image classification tasks.

4.2.2 ResNet-50 Conclusion

Residual Networks marked a significant breakthrough that reshaped training methodologies for deep convolutional neural networks, particularly in the field of computer vision. This

innovative architecture, distinguished by the use of skip connections and residual blocks, not only transformed how such networks are trained but also paved the way for the development of more advanced and efficient models. With its 50 bottleneck residual blocks, ResNet-50 has exhibited outstanding capabilities in addressing challenges such as the vanishing gradient problem, thereby enabling the successful training of very deep neural networks.

4.3 ViT-Patch16

The ViT-Patch16 transformer model divides each input image into non-overlapping 16×16 pixel patches. These patches are then linearly embedded and processed as a sequence by a standard transformer encoder. This model was introduced in the influential paper *"An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale"* by Dosovitskiy et al.

The version used in this study was pre-trained on the large-scale ImageNet-21k dataset [27], which contains approximately 14 million images and 21,843 classes, and subsequently fine-tuned on ImageNet-2012 [24], comprising 1 million images and 1,000 classes. Both training stages were conducted using an input resolution of 224×224 pixels.

This pre-training and fine-tuning strategy enables the Vision Transformer to achieve competitive performance in image classification tasks by effectively leveraging global attention mechanisms, distinguishing it from traditional convolutional approaches.

In ViT, each input image is divided into fixed-size patches of 16×16 pixels, which are then linearly embedded and treated as input tokens to the transformer. A special classification token, denoted as `[CLS]`, is prepended to the sequence of embedded patches and is used for the classification task. Absolute positional embeddings are added to the patch embeddings before passing the sequence through multiple layers of the transformer encoder.

Through pretraining, the ViT learns rich internal representations of visual data that are transferable to downstream tasks. For example, when working with a labeled image dataset, a simple linear classifier can be added on top of the pretrained encoder. Typically, this classifier is applied to the final hidden state corresponding to the `[CLS]` token, which is considered to encapsulate a global representation of the entire image.

4.4 Custom Convolutional Neural Network for HAM10000

In addition to advanced models such as ResNet50 and ViT-Patch16, this study incorporates a bespoke Convolutional Neural Network (CNN), developed and trained from scratch for the HAM10000 dataset.

Unlike pre-trained models, the custom CNN starts with randomly initialized weights and learns features directly from the data during training. This approach enables precise control over the model's architecture and complexity, allowing it to be tailored specifically to the characteristics of the HAM10000 dataset.

While it does not benefit from the extensive knowledge embedded in pre-trained networks, the custom CNN serves as a lightweight, efficient, and interpretable baseline for comparative analysis.

4.4.1 Architecture

The adapted CNN I design, branded `SimpleCNN`, consolidates its architecture around four convolutional stacks, culminating in a class label head. Each stack embeds a 3×3 convolutional layer that pads its input, ensuring that intermediary spatial dimensions remain preserved. Immediately after, batch normalization is deployed and routed to a ReLU nonlinearity, thus consistently amplifying the training throughput. To counteract the threat of overfitting, dropout is judiciously inserted only in the first and the third stacks.

Between the inference stacks, max-pooling layers—minimal 2×2 masked windows—process the results of the second, third, and fourth stacks, permitting the common progressive shrinking of feature volumes. Filter counts within stacks scale from 32, rising through transparent increments to 64, 128, and finally 256, preparing the model to capture increasingly abstract and invariant representations.

Once the convolutional feature extractors finish, their compressed panorama is flattened and routed to a classification section. This section first employs a dense assembly of 512 neurons, followed by a ReLU activation and a dropout layer with $p = 0.3$. The features are then passed to a second dense layer with 128 neurons, again followed by ReLU and an equivalent dropout threshold ($p = 0.3$). Representations are afterward delivered to a concluding dense layer that projects them onto seven output units, corresponding to the target labels.

Table 4.1: SimpleCNN Architecture

Block	**Details**
Input	$3 \times H \times W$ RGB image
Block 1	Conv2d: $3 \to 32$, 3×3, padding=1 BatchNorm2d, ReLU Dropout2d: $p = 0.25$
Block 2	Conv2d: $32 \to 64$, 3×3, padding=1 BatchNorm2d, ReLU MaxPool2d: 2×2
Block 3	Conv2d: $64 \to 128$, 3×3, padding=1 BatchNorm2d, ReLU Dropout2d: $p = 0.25$ MaxPool2d: 2×2
Block 4	Conv2d: $128 \to 256$, 3×3, padding=1 BatchNorm2d, ReLU MaxPool2d: 2×2
Classifier	Flatten Linear: $256 \times 28 \times 28 \to 512$, ReLU Dropout: $p = 0.3$ Linear: $512 \to 128$, ReLU Dropout: $p = 0.3$ Linear: $128 \to 7$

Chapter 5

Training Configuration and Optimization

5.1 Optimization in Deep Learning

Evolving Techniques on Optimization of Deep Learning Modules

The process of training a model can be framed as solving an optimization problem [28]. The model attempts to adjust its parameters so that the loss function captures the gap between the predictions it makes and the actual labels in the dataset. The primary method that addresses this problem is called *gradient descent* [29], where the parameters of the model are iteratively updated in the opposite direction of the gradient of the loss function.

The general formulation of the update rule is given by:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t) \tag{5.1}$$

In the above equation, θ_t represents the model parameters at time t, η is the learning rate, while $\nabla_\theta \mathcal{L}(\theta_t)$ denotes the gradient of the loss function with respect to the parameters.

Over time, many optimizers have been developed to increase the speed and stability of convergence of the loss function toward a global minimum. Popular implementations include **Stochastic Gradient Descent (SGD)**, **SGD with Momentum**, **RMSProp**, and **Adam**. Each of these methods introduces mechanisms to address specific challenges, such as instability, slow convergence, sensitivity to the choice of learning rate, and the presence of noisy gradients.

5.2 Learning Rate

The learning rate [30] is an important hyperparameter that determines the magnitude of each update performed during gradient descent [28]. It regulates the extent to which the model's parameters are adjusted at each training iteration. When the learning rate is excessively high, the training may destabilize and not reach convergence. Conversely, if it is excessively low, the model could require an extremely extended period to learn or might become trapped before achieving an optimal solution.

5.2.1 The Importance of the Learning Rate

The learning rate fundamentally determines the "pace" of learning. In an appropriate environment, the model can consistently progress toward the optimal solution, steering clear of overshooting the goal and failing to make significant advancements. An apt comparison is descending a slope: go too slowly and you will hardly progress, but hurry too much and you will trip. The optimal learning rate establishes a secure and efficient rate of decline.

5.3 Focal Loss as the Objective Function

Loss functions specify what the model aims to improve. In this research, focal loss [31] was utilized in place of the traditional cross-entropy loss [32]. Focal loss is especially beneficial for imbalanced datasets, as it minimizes the influence of easily classified examples and emphasizes more challenging, misclassified ones.

The equation for focal loss is:

$$FL(p_t) = -\alpha(1 - p_t)^{\gamma} \log(p_t)$$

Here, p_t is the predicted probability of the correct class, α is a balancing factor between classes, and γ adjusts how much focus is placed on difficult cases.

In practice, the process works as follows:

- Focal loss produces the error signal that shows how far off the predictions are.
- Gradient descent uses this signal to determine the direction of improvement.
- The optimizer then updates the model's parameters accordingly.

5.4 Adam Optimizer

Adam Optimizer

While gradient descent provides the foundation, specific algorithms are used in practice to make optimization more efficient and stable. For this work, the *Adam optimizer* was chosen.

Adam (**Adaptive Moment Estimation**) improves upon basic stochastic gradient descent (SGD) by maintaining *per-parameter adaptive learning rates*. Unlike plain SGD, which applies the same global learning rate to all parameters, Adam adapts the step size for each parameter based on the **first and second moments** of the gradients.

Formally, given a parameter θ_t at iteration t and its gradient $g_t = \nabla_\theta \mathcal{L}(\theta_t)$, Adam computes the moving averages:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \qquad \text{(first moment, gradient mean)} \tag{5.2}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \qquad \text{(second moment, gradient variance)} \tag{5.3}$$

Since m_t and v_t are initialized at zero, bias-corrected estimates are applied:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \tag{5.4}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{5.5}$$

The parameter update rule then becomes:

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{5.6}$$

where α is the global learning rate, β_1 and β_2 control the exponential decay of the moving averages (commonly $\beta_1 = 0.9$, $\beta_2 = 0.999$), and ϵ is a small constant (typically 10^{-8}) to ensure numerical stability.

Comparison with Other Optimizers

- **SGD:** Uses a single global learning rate and often struggles with sparse or noisy gradients. Requires careful learning rate tuning and decay scheduling.
- **Momentum SGD:** Incorporates a moving average of gradients to accelerate convergence, but still applies the same learning rate to all parameters.

- **RMSProp:** Adapts the learning rate by dividing by the root of an exponential average of squared gradients, stabilizing updates but lacking momentum.
- **Adam:** Combines the advantages of **Momentum** (through m_t) and **RMSProp** (through v_t), providing robustness to noisy gradients and better handling of imbalanced data.

Why Adam was used in this study

Because the classification tasks in this work involve imbalanced data and potentially noisy gradient updates (due to focal loss emphasizing hard-to-classify examples), Adam provides an excellent balance of fast convergence and stability. Its adaptive mechanism automatically scales learning rates for rare classes and difficult samples, reducing the need for extensive manual tuning.

Chapter 6

Experiments and Results

The experiments' outcomes are presented in this chapter. Five key metrics were used to evaluate each model: *test accuracy*, *precision*, *recall*, *F1-score*, and *support*. These were chosen because they offer a comprehensive view of performance, especially in situations involving multi-class classification where class imbalance could be problematic.

Loss curves were used to monitor the training process in addition to the final outcomes. These graphs make it easier to spot possible underfitting or overfitting by showing how training and validation loss vary over the epochs.

Together, the quantitative metrics and loss curves provide a numerical and visual representation of the methods' efficacy. In the sections that follow, I present each experiment's results and examine the most important takeaways.

6.1 Evaluation Metrics

Before analyzing the results, it is important to describe the evaluation metrics that were used. In classification problems, especially when dealing with class imbalance, relying only on accuracy may not give a complete picture of model performance. For this reason, additional metrics such as precision, recall, F1-score, and support were also considered.

Accuracy

Accuracy [33] measures the proportion of correctly classified samples out of the total number of samples. It is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP = true positives, TN = true negatives, FP = false positives, and FN = false negatives.

Precision

Precision [33] (also called Positive Predictive Value) indicates how many of the samples predicted as positive are actually positive:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall

Recall [33](also called Sensitivity or True Positive Rate) measures how many of the actual positive samples were correctly identified by the model:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-score

The F1-score [33] is the harmonic mean of precision and recall. It provides a balance between the two, especially when there is an uneven class distribution:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Support

Support [34]refers to the number of true instances for each class in the dataset. It is not a performance metric itself, but it helps to interpret the other metrics by showing how many samples each class contributed to the evaluation.

Loss Function, Training Loss and Validation Loss

For this project, as I mentioned in the previous section,the **Focal Loss** [31] was used instead of the standard cross-entropy loss. Focal Loss is particularly useful in cases of class imbalance, as it reduces the relative loss contribution from easy-to-classify examples and focuses more on hard, misclassified ones.

The **training loss** [35] is the loss computed on the training set after each epoch. It shows how well the model is fitting the data it has already seen. A decreasing training loss usually indicates that the model is learning the patterns in the training data.

The **validation loss** [35] is the loss calculated on a separate validation set that is not used for parameter updates. It reflects how well the model generalizes to unseen data. If both training and validation loss decrease, the model is learning appropriately. However, if training loss continues to decrease while validation loss starts to increase, this is often a sign of *overfitting*.

6.2 Vision Transformer (ViT) Results

The Vision Transformer (ViT) underwent training with multiple experimental configurations in order to determine the most effective setup for the HAM10000 dataset. Different batch sizes (16, 32, and 64) were tested, with the best results achieved when using a batch size of **32**. The learning rate was explored in the range of 10^{-5} to 10^{-4}, and after many trials, the most effective value was found to be 7×10^{-5}.

Regarding the number of epochs, the model was initially trained for up to 40 epochs with early stopping applied. Signs of overfitting appeared after the first few epochs, and the best validation performance was observed at **epoch 7** (Figure 6.2). Early stopping was therefore triggered at epoch 14.

The final evaluation metrics on the test set are presented in Figure 6.1. The ViT achieved an **overall test accuracy of 80.93%**, with a **balanced accuracy of 68.47%** and a **macro F1-score of 0.68**. Looking more closely at the classification report, the larger classes (such as class 1) achieved very high recall and precision, which strongly influenced the overall weighted score. However, minority classes such as classes 4 and 6 displayed weaker recall values, meaning the model struggled to detect them consistently. This behavior is common with imbalanced datasets such as HAM10000, even when augmentation and oversampling are applied. Still, precision remained relatively high for these smaller classes, suggesting that when the model did predict them, the predictions were often correct.

Examining the loss curves in Figure 6.2, the training loss decreased steadily, approaching zero, which indicates that the model learned the training data patterns efficiently. In contrast, the validation loss plateaued early and remained relatively stable across epochs, showing little improvement. This stability suggests that the model generalized reasonably well without falling into severe overfitting.

At the same time, the lack of significant improvement in validation loss implies that the

model may have reached its performance ceiling on this dataset. Further improvements would likely require either expanding the dataset or applying more advanced balancing strategies.

Overall, the results are strong: the model achieved more than 80% test accuracy, with a weighted F1-score close to 0.81. However, recall for the under-represented classes remains unsatisfactory, which highlights the ongoing challenge of class imbalance in medical imaging datasets.

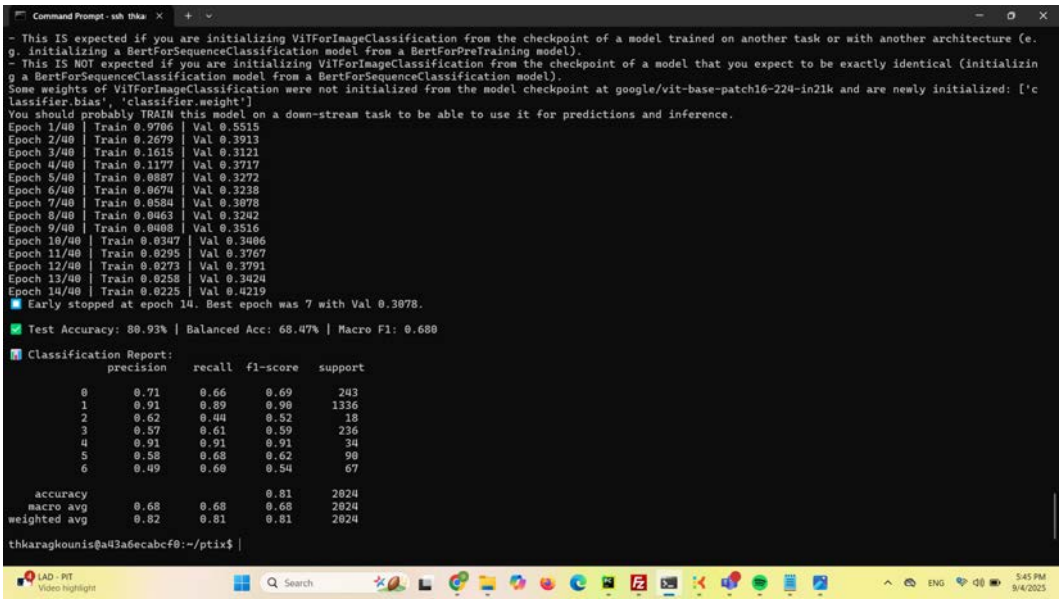


Figure 6.1: Classification report and test results of Vit-16.

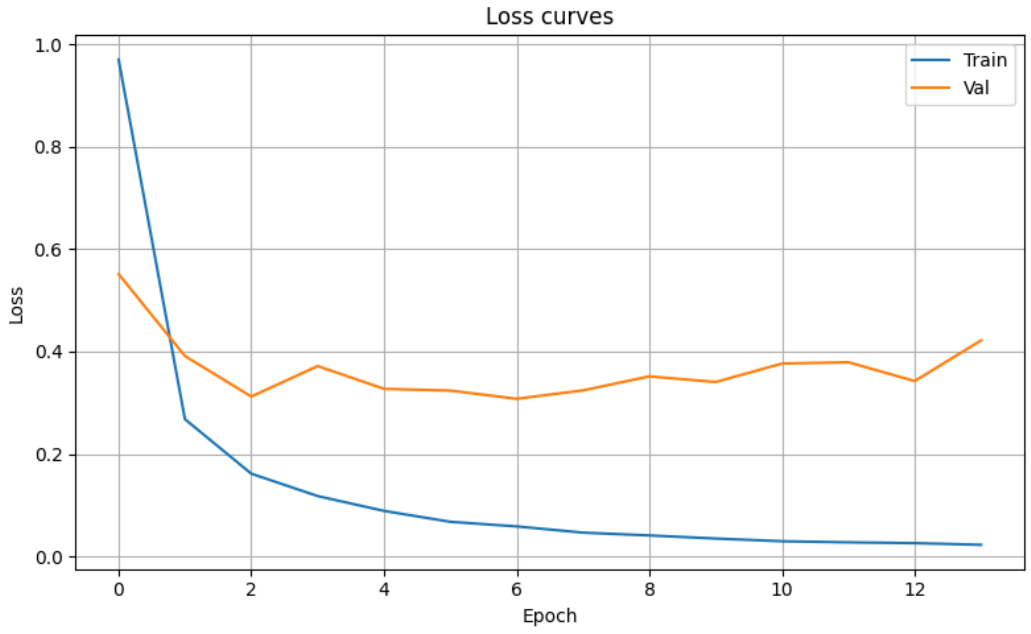


Figure 6.2: Training and validation loss curves of Vit-16.

6.3 ResNet-50 Results

For the evaluation of ResNet-50, several training setups were tested in order to determine the most effective configuration. After experimentation, the best results were achieved with a batch size of **32** and a learning rate of 2×10^{-4}. The model was trained for up to 40 epochs with early stopping enabled, which terminated training at **epoch 14**, with the best validation performance observed at **epoch 7**.

The classification results are shown in Figure 6.3. ResNet-50 achieved a **test accuracy of 79.55%**, a **balanced accuracy of 65.20%**, and a **macro F1-score of 0.64**, with a weighted F1-score of **0.80**. As expected, the largest class (class 1) achieved the highest precision and recall values, which strongly influenced the overall weighted metrics. However, smaller classes such as class 2 and class 6 suffered from low recall, highlighting the same imbalance issue observed in the Vision Transformer experiments. Interestingly, class 4 showed relatively better balance in precision and recall, suggesting that the model was able to capture its discriminative features more effectively.

The training and validation loss curves are presented in Figure 6.4. The training loss decreased steadily, approaching very low values, while the validation loss remained consistently higher and displayed more fluctuation. This gap indicates overfitting, particularly as training progressed, even though augmentation and regularization strategies were employed.

In summary, ResNet-50 produced competitive results, with an accuracy close to 80% and a weighted F1-score of 0.80. Similar to the Vision Transformer, the model handled majority classes well but continued to struggle with underrepresented classes, reflecting the inherent imbalance challenges of the HAM10000 dataset.

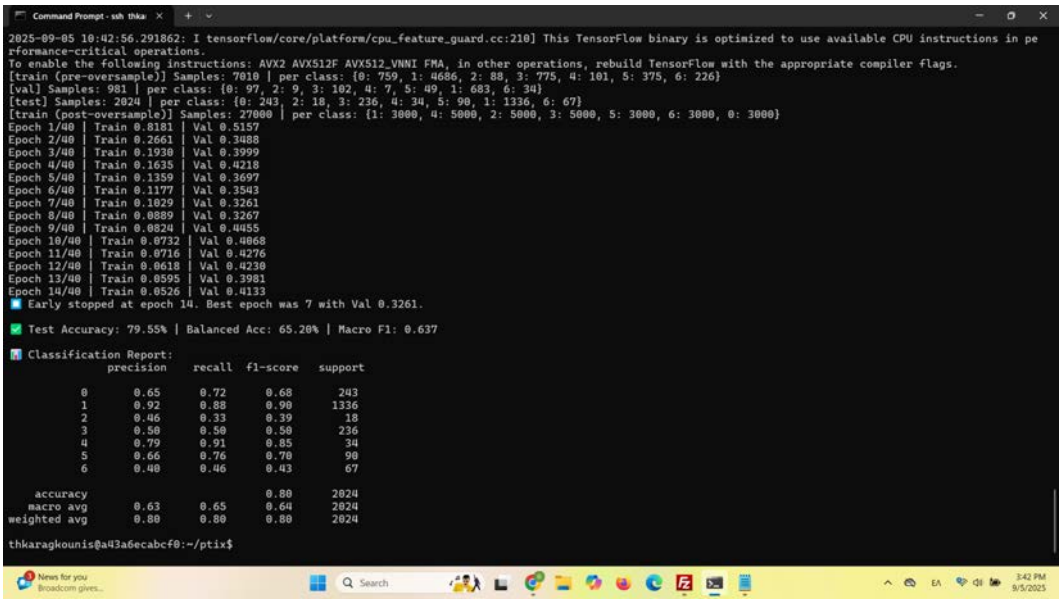


Figure 6.3: Classification report and test results of ResNet-50.

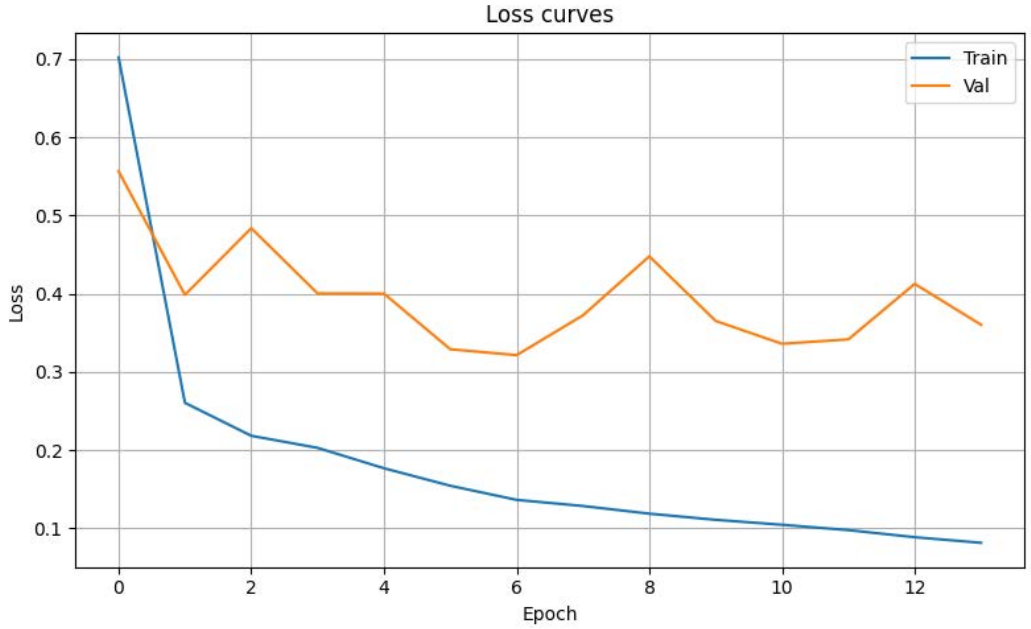


Figure 6.4: Training and validation loss curves of ResNet-50.

6.4 Custom CNN Results

Using pretrained models such as ResNet-50 and Vision Transformer as references, a distinctive convolutional neural network (CNN) was developed and trained from scratch to set a baseline for comparison. The architecture comprised four convolutional layers featuring

batch normalization, ReLU activations, dropout for regularization, and max pooling for spatial reduction. Following these convolutional layers, a fully connected classifier was introduced, consisting of two hidden layers with dropout, leading to a final output layer with seven neurons corresponding to the classes of HAM10000. The model was trained with a batch size of **32** and a learning rate of 1×10^{-3}, for up to 200 epochs with early stopping implemented. Training concluded at epoch 159, with the best validation performance observed at epoch 134. The custom CNN achieved a **test accuracy of 65.42%**, a **balanced accuracy of 62.31%**, and a **macro F1-score of 0.53**, while the weighted F1-score remained considerably lower than the pretrained models, which is expected given its relatively shallow architecture and the absence of prior knowledge from large-scale datasets.

Examining the metrics for each category reveals a comparable trend to the other models: the majority class (class 1) achieved the highest recall and precision, while the minority classes, such as class 2 and class 0, showed reduced recall values, indicating that the model struggled to generalize effectively for the less represented categories. The classification report is given in the Figure 6.5. Furthermore, the training and validation loss curves (Figure 6.6) illustrate that the custom CNN required significantly more epochs to achieve convergence compared to the pretrained models. The training loss consistently decreased during the process, whereas the validation loss exhibited greater variability but eventually stabilized.

In summary, the custom CNN served as a useful baseline experiment. Achieving a test accuracy of 65% and a macro F1-score of 0.53, it highlighted the challenges of training models from scratch on an imbalanced medical imaging dataset. Compared with the pretrained ResNet-50 and Vision Transformer, which delivered substantially higher accuracy and F1-scores, the results of the custom CNN underscore the clear benefits of transfer learning and the importance of leveraging prior knowledge from large-scale datasets.

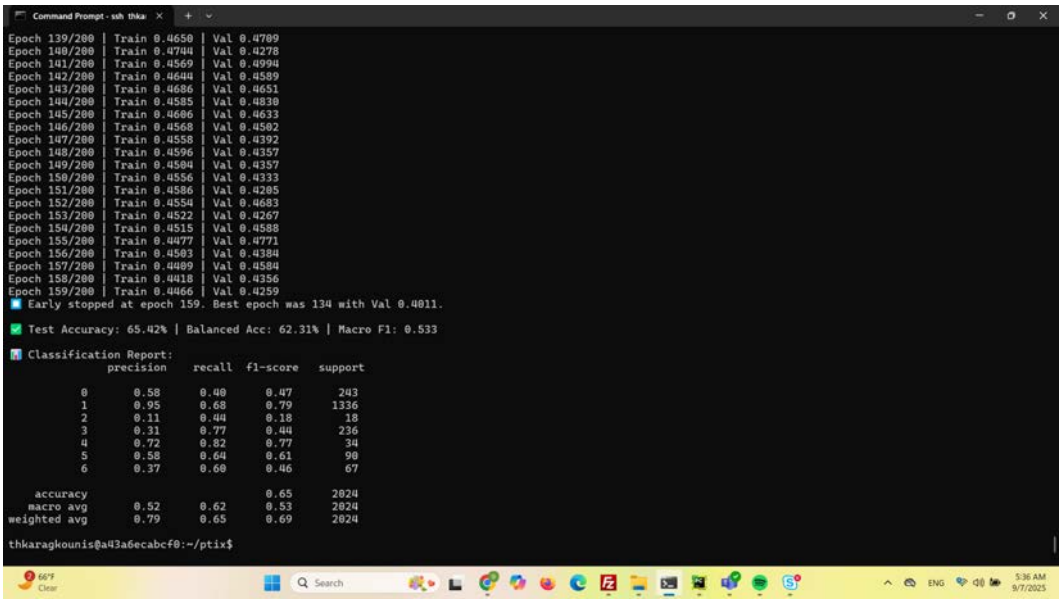


Figure 6.5: Classification report and test results of the custom CNN.

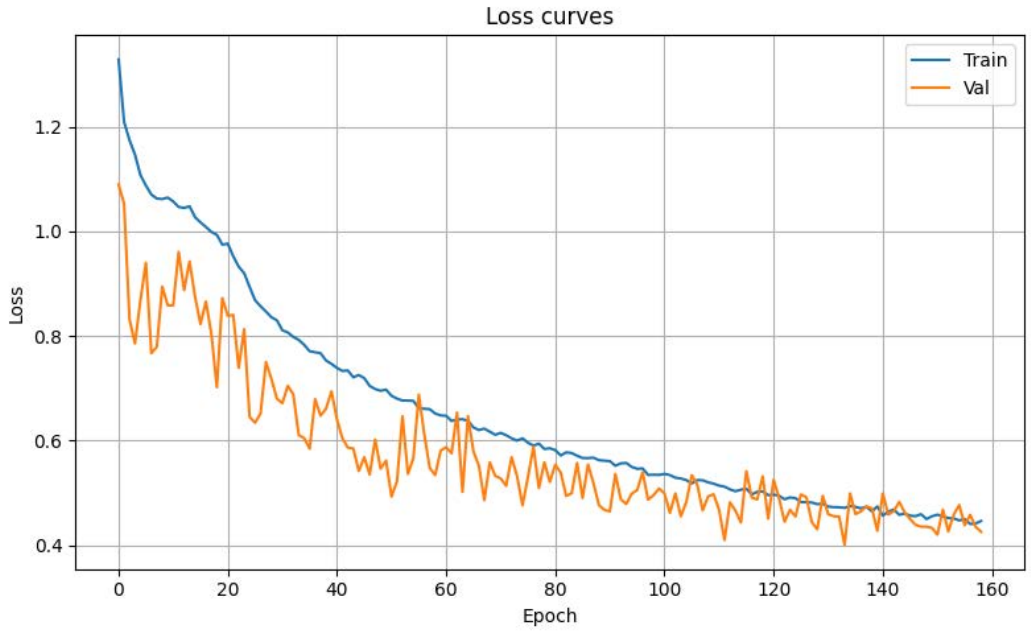


Figure 6.6: Training and validation loss curves of the custom CNN.

6.5 Conclusion

In conclusion, the comparison of the three models—the Vision Transformer, ResNet-50, and the specially designed CNN built from the ground up—highlights both the benefits of modern pretrained architectures and the limitations of building networks without prior infor-

mation. With about **81%** test accuracy and over **68%** balanced accuracy, the Vision Transformer achieved the best overall performance. ResNet-50 came in second with almost **80%** accuracy and **65%** balanced accuracy. In contrast, the custom CNN required substantially more epochs to converge, yet only managed to attain **65%** accuracy and a balanced accuracy of **62%**. Given that pretrained feature extractors provide a strong foundation for medical image analysis, this gap emphasizes the advantages of transfer learning.

Looking more closely at the results for each class, two specific groups—class 2 and class 4—had trouble performing well in all the models tested. For class 2, the problem is mainly due to the very small number of test examples. This makes it difficult for the models to learn useful patterns and leads to unreliable precision and recall. Class 4, although larger, still underperformed, which suggests that its images share visual similarities with other classes and are harder to distinguish from more common categories. These cases highlight two key challenges: data scarcity and inter-class similarity.

Another major issue affecting all experiments is the imbalance in the HAM10000 test set. The most common class dominates the evaluation, so the models achieve high overall accuracy and weighted F1-scores even when performance on the rarer classes remains unsatisfactory. This imbalance hides the fact that minority classes, which are highly relevant in real clinical scenarios, are not being detected well. Although training strategies such as data augmentation, class weighting, and oversampling were applied, they cannot compensate for the skewed distribution of the test data itself.

In general, the experiments show that models like ResNet-50 and the Vision Transformer work better and faster for classifying skin lesions, while models built from scratch face significant challenges. At the same time, the persistent difficulties with underrepresented classes and an imbalanced test set indicate that future improvements will require not only more sophisticated models but also more balanced and representative datasets.

6.6 Limitations and Future Work

Overall, the results of this project show that deep learning models are effective for classifying skin lesions, but several important challenges remain. The HAM10000 dataset is highly imbalanced, with the `nv` class containing approximately 6,500 images, while categories such as class 2 and class 4 each have fewer than 1,000 images. This imbalance made it easier for

the models to achieve strong performance on the majority class, but much harder to handle the minority classes consistently. In addition, the experiments were limited by GPU resources. Larger batch sizes such as 64 could not be used due to memory constraints, and more advanced architectures like `google/vit-base-patch16-324` could not be tested for the same reason. To address these issues, future work should focus on expanding the dataset with more samples from underrepresented classes and employing more powerful GPU hardware. With these improvements, it would be possible to explore larger transformer models, more sophisticated training strategies, and ultimately achieve more balanced and reliable results in all categories. Finally, one could also explore federated learning approaches [36, 37], as well as techniques for model reduction e.g., [38] in order to make the models suitable for running over lightweight devices and transmitted over the Web [39].

Bibliography

[1] Coursera. Machine learning in health care, 2023. Accessed: 26 August 2025.

[2] GeeksforGeeks. Introduction to convolutional neural network, 2023. Accessed: 26 August 2025.

[3] Viso.ai. Vision transformer (vit) explained, 2023. Accessed: 26 August 2025.

[4] Philipp Tschandl. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions, 2018. Accessed: 26 August 2025.

[5] IBM. Deep learning, 2023. Accessed: 26 August 2025.

[6] Google Developers. Machine learning crash course: Multiclass classification, 2023. Accessed: 26 August 2025.

[7] Viso.ai. Resnet: Residual neural network explained, 2023. Accessed: 26 August 2025.

[8] Aditya Kumar Singh. Architecture of artificial neural network, 2021. Accessed: 2025-09-10.

[9] GeeksforGeeks. Kernels and filters in convolutional neural networks, 2023. Accessed: 26 August 2025.

[10] GeeksforGeeks. Relu activation function in deep learning, 2023. Accessed: 26 August 2025.

[11] GeeksforGeeks. Introduction to pooling layer in convolutional neural networks, 2023. Accessed: 26 August 2025.

[12] Unknown or Author Name. A guide to convolutional neural networks — the eli5 way, 2023. Accessed: 2025-09-10.

[13] D. Katsaros, N. Zacharia, S.-D. Kravariti, and D. Papakostas. Modern deep learning approaches for galaxy classification. In *International Astronomical Union Symposium 397 on Exploring the Universe with Artificial Intelligence (UniversAI)*, 2025.

[14] GeeksforGeeks. What is layer normalization?, 2023. Accessed: 26 August 2025.

[15] Nachi Keta. Deep learning: Gelu (gaussian error linear unit) activation function, 2020. Accessed: 26 August 2025.

[16] GeeksforGeeks. Multi-layer perceptron learning in tensorflow, 2023. Accessed: 26 August 2025.

[17] Deval Shah. Vision transformer: What it is & how it works [2024 guide], December 2022. Accessed: 2025-09-10.

[18] Gaudenz Boesch. Vision transformers (vit) in image recognition, November 2023. Accessed: 2025-09-10.

[19] Hassaan Idrees. Vision transformer vs cnn: A comparison of two image processing giants, 2023. Accessed: 26 August 2025.

[20] LearnOpenCV. Detr: Overview and inference, 2023. Accessed: 26 August 2025.

[21] Python Software Foundation. Python programming language – official website, 2025. Accessed: 26 August 2025.

[22] ScienceDirect. Gaussian blur – engineering topic overview, 2025. Accessed: 26 August 2025.

[23] D.-K. Nguyen, M. Assran, U. Jain, M.R. Oswald, C.G.M. Snoek, and X. Chen. An image is worth more than 16x16 patches: Exploring transformers on individual pixels. *arXiv preprint arXiv:2406.09415*, 2024.

[24] TensorFlow Datasets. Imagenet-2012 dataset, 2025. Accessed: 26 August 2025.

[25] cvinvolution. Residual neural network and the degradation problem, 2020. Accessed: 26 August 2025.

[26] Norah Ahmed Al-Humaidan and Master Prince. An illustration of ResNet-50 layers architecture. Figure 1 in ‘‘A Classification of Arab Ethnicity Based on Face Image

Using Deep Learning Approach'', March 2021. Available on ResearchGate. Accessed: 2025-09-10.

[27] T. Ridnik, E. Ben-Baruch, A. Noy, and L. Zelnik-Manor. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021.

[28] E2E Networks. Optimization in deep learning: Learn with examples, 2023. Accessed: 26 August 2025.

[29] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[30] ScienceDirect. Learning rate – computer science topic overview, 2025. Accessed: 26 August 2025.

[31] Rohith Gandhi. Focal loss — a better alternative for cross entropy, 2018. Accessed: 26 August 2025.

[32] Yutao Zhong Anqi Mao, Mehryar Mohri. Cross-entropy loss functions: Theoretical analysis and applications. *arXiv preprint arXiv:2304.07288*, 2023.

[33] Piyush Kashyap. Understanding precision, recall, and f1-score metrics, 2021. Accessed: 26 August 2025.

[34] Nirajan Acharya. Understanding precision, recall, f1-score, and support in machine learning evaluation, 2022. Accessed: 26 August 2025.

[35] GeeksforGeeks. Training and validation loss in deep learning, 2023. Accessed: 26 August 2025.

[36] E. Fragkou, E. Chini, M. Papadopoulou, D. Papakostas, D. Katsaros, and S. Dustdar. Distributed federated deep learning in clustered Internet of Things wireless networks with data similarity-based client participation. *IEEE Internet Computing magazine*, 28(6):53–61, 2024.

[37] E. Fragkou and D. Katsaros. Joint survey in decentralized federated learning and tinyml: A brief introduction to swarm learning. *Future Internet*, 16(11), 2024.

[38] E. Fragkou, M. Koultouki, and D. Katsaros. Model reduction of feed forward neural networks for resource-constrained devices. *Applied Intelligence*, 53:14102–14127, 2023.

[39] D. Katsaros and Y. Manolopoulos. Broadcast program generation for Webcasting. *Data and Knowledge Engineering*, 49(1):1–21, 2004.