TECHNICAL UNIVERSITY OF CRETE

DIPLOMA THESIS

Neural Network Architectures for Skin Cancer Detection

Author:

ALEXANDROS PAPADOPOULOS

Thesis Committee:

PROF. VASILEIOS DIGALAKIS
PROF. MICHAIL LAGOUDAKIS
PROF. EVANGELOS KALOGERAKIS

School of Electrical and Computer Engineering

October 2025

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σύγκριση Αρχιτεκτονικών Νευρωνικών Δικτύων για τη Διάγνωση Καρκίνου του Δέρματος

Συγγραφέας:
ΑΛΕΞΑΝΔΡΟΣ ΠΑΠΑΔΟΠΟΥΛΟΣ

Εξεταστική Επιτροπή:
ΚΑΘ. ΒΑΣΙΛΕΙΟΣ ΔΙΓΑΛΑΚΗΣ
ΚΑΘ. ΜΙΧΑΗΛ ΛΑΓΟΥΔΑΚΗΣ
ΚΑΘ. ΕΥΑΓΓΕΛΟΣ ΚΑΛΟΓΕΡΑΚΗΣ

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Οκτώβριος 2025

Abstract

Deep learning has transformed medical imaging, particularly for automated disease diagnosis, with Convolutional Neural Networks (CNNs) traditionally dominating image analysis. The recent rise of Transformer architectures, including Vision Transformers (ViTs), offers a powerful new paradigm for image recognition. This thesis provides a systematic comparison of custom designed CNN and ViT architectures built from scratch for dermatological image classification. It explores the impact of fundamental design choices, such as network depth and regularization techniques, to maximize predictive accuracy, while considering computational constraints. These models are benchmarked on the HAM10000 dataset, a collection of over 10,000 dermatoscopic images across seven skin lesion categories, using a comprehensive suite of metrics including validation accuracy, loss, Macro F1-score, and parameter count. This study offers critical, evidence-based insights into the architectural trade-offs inherent to each approach, serving as a methodical guide for developing effective, specialized models for challenging medical imaging tasks.

Περίληψη

Η βαθιά μάθηση έχει μεταμορφώσει την ιατρική απεικόνιση, ιδίως στον τομέα της αυτοματοποιημένης διάγνωσης ασθενειών, με τα Συνελικτικά Νευρωνικά Δίκτυα (CNNs) να κυριαρχούν παραδοσιακά στην ανάλυση εικόνας. Η πρόσφατη άνοδος των αρχιτεκτονικών τύπου Μετασχηματιστή, συμπεριλαμβανομένων των Vision Transformers (ViTs), εισάγει ένα ισχυρό νέο παράδειγμα για την αναγνώριση εικόνων. Η παρούσα διατριβή παρουσιάζει μια συστηματική σύγκριση προσαρμοσμένων αρχιτεκτονικών CNN και ViT, οι οποίες έχουν αναπτυχθεί εξ ολοκλήρου από την αρχή για την ταξινόμηση δερματολογικών εικόνων. Εξετάζεται ο αντίκτυπος βασικών επιλογών σχεδιασμού, όπως το βάθος του δικτύου και οι τεχνικές κανονικοποίησης, με στόχο τη μεγιστοποίηση της προγνωστικής ακρίβειας, λαμβάνοντας παράλληλα υπόψη τους υπολογιστικούς περιορισμούς. Τα μοντέλα αξιολογούνται στο σύνολο δεδομένων HAM10000, μια συλλογή από περισσότερες από 10.000 δερματοσκοπικές εικόνες επτά κατηγοριών δερματικών βλαβών, χρησιμοποιώντας ένα ολοκληρωμένο σύνολο μετρικών που περιλαμβάνει την ακρίβεια επικύρωσης, την απώλεια, τη βαθμολογία Macro F1 και τον αριθμό παραμέτρων. Η μελέτη παρέχει κρίσιμες, τεκμηριωμένες γνώσεις σχετικά με τους αρχιτεκτονικούς συμβιβασμούς που είναι εγγενείς σε κάθε προσέγγιση, λειτουργώντας ως μεθοδικός οδηγός για την ανάπτυξη αποτελεσματικών και εξειδικευμένων μοντέλων για απαιτητικές εφαρμογές ιατρικής απεικόνισης.

Acknowledgements

I am deeply grateful to my supervisor, Professor Vasileios Digalakis. His guidance was essential at every stage of this work, and his insightful feedback helped shape this thesis into what it is today.

I would also like to thank the members of my examination committee, Professor Michail Lagoudakis and Professor Evangelos Kalogerakis.

To my grandmother Stavroula, thank you for your endless love and encouragement. Your belief in me has always been my greatest source of strength.

Finally, to my family, thank you for your constant support and patience. I couldn't have done this without you.

Ευχαριστίες

Θα ήθελα να εκφράσω τις βαθύτατες ευχαριστίες μου στον επιβλέποντα καθηγητή μου, κ. Βασίλειο Διγαλάκη. Η καθοδήγησή του υπήρξε καθοριστική σε κάθε στάδιο αυτής της εργασίας, και τα εύστοχα σχόλιά του συνέβαλαν καθοριστικά στη διαμόρφωση της παρούσας διατριβής.

Επίσης, ευχαριστώ θερμά τα μέλη της εξεταστικής μου επιτροπής, τον καθηγητή κ. Μιχαήλ Λαγουδάκη και τον καθηγητή κ. Ευάγγελο Καλογεράκη.

Στη γιαγιά μου, Σταυρούλα, σε ευχαριστώ για την ατελείωτη αγάπη και την ενθάρρυνσή σου. Η πίστη σου σε εμένα ήταν πάντα η μεγαλύτερη πηγή δύναμής μου.

Τέλος, στην οικογένειά μου, σας ευχαριστώ για τη συνεχή υποστήριξη και την υπομονή σας. Δεν θα μπορούσα να τα είχα καταφέρει χωρίς εσάς.

Nomenclature

Abbreviation	Term
AI	Artificial Intelligence
CAD	Computer Aided Diagnosis
CNN	Convolutional Neural Network
FFN	Feed Forward Network
GPU	Graphics Processing Unit
HAM10000	Human Against Machine with 10000 training images
MLP	Multi Layer Perceptron
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
ViT	Vision Transformer
ANNs	Artificial Neural Networks
Adam	Adaptive Moment Estimation
AKIEC	Actinic Keratosis
BCC	Basal Cell Carcinoma
BKL	Benign Keratosis
DF	Dermatofibroma
MEL	Melanoma
NV	Melanocytic Nevi
VASC	Vascular Lesions
GELU	Gaussian Error Linear Unit
AdamW	Adam with Weight Decay Decoupling
BatchNorm	Batch Normalization
NLP	Natural Language Processing
CLS	Classification (token)
CPU	Central Processing Unit
RAM	Random Access Memory
PR	Precision-Recall
t-SNE	t-Distributed Stochastic Neighbor Embedding
AUC-PR	Area Under the PR Curve
Grad-CAM	Gradient-weighted Class Activation Mapping

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Skin cancer is one of the most common forms of malignancy worldwide, with melanoma being the most dangerous type due to its high potential for metastasis. Early and accurate diagnosis is critical for improving patient prognosis and survival rates. Traditionally, the diagnosis of skin lesions relies on visual inspection by a dermatologist, often aided by dermoscopy, a noninvasive imaging technique that enhances the visibility of subsurface skin structures. While this process is effective, it remains subjective and dependent on the expertise of the clinician.

In recent years, the field of artificial intelligence (AI), particularly deep learning, has shown significant potential to revolutionize medical diagnostics. Deep learning models, when trained on large datasets of medical images, can learn to identify complex patterns and classify images with performance often matching or even exceeding that of human experts. This capability has led to a surge of interest in developing computer aided diagnosis (CAD) systems to assist dermatologists. These systems aim to make screenings faster, more consistent, more accessible, and more objective. CNNs have dominated this space for nearly a decade, but recent advancements in ViTs offer a compelling alternative architecture.

1.2 Problem Statement

The potential of deep learning in dermatology is very clear, but the optimal architectural choice for skin lesion classification remains an open question. Many existing studies rely on large, pre trained models that are computationally expensive and may not suit the specific nuances of dermatoscopic images. There is a pressing need to move beyond ready to use solutions and explore custom designed architectures tailored for a specific challenge.

This thesis addressed this gap by conducting a rigorous comparative study of several custom built deep learning models. The core problem is to determine the relative strengths and weaknesses of different architectural philosophies, namely, convolutional based and attention based approaches, when designed specifically for the task of classifying skin lesion types in the HAM10000 dataset.

1.3 Research Aims and Objectives

The primary aim of this research is to design, implement, evaluate, and compare a series of novel CNN and ViT models to provide a clear, evidence-based comparison of their effectiveness for dermatological image classification. To achieve this aim, four concrete objectives have been established. First, several distinct CNN architectures will be designed and implemented, exploring the impact of network depth and regularization strategies. Second, various ViT architectures will be developed, investigating the effect of different patch embedding strategies and encoder depths. Third, all models will be systematically trained and evaluated on the HAM10000 dataset under a consistent experimental framework. Finally, a comparative analysis will be conducted using performance metrics such as validation accuracy, and training time to illustrate architectural trade-offs.

1.4 Contributions of the Thesis

This thesis introduces a collection of novel, lightweight deep learning models specifically optimized for skin lesion classification. It presents a rigorous, multifaceted benchmarking study of these custom architectures on the publicly available HAM10000 dataset, enabling a direct comparison of their performance and computational efficiency. Finally, it delivers actionable, evidence-based insights into the architectural trade-offs between different CNN and ViT models for medical imaging, providing practical guidance for future research and development in this domain.

1.5 Thesis Outline

The remainder of this thesis is structured as follows:

Chapter 2 provides a review of the relevant literature, covering the clinical background of dermatoscopy and skin lesions, a description of the HAM10000 dataset, and a detailed overview of the core concepts behind CNNs and ViTs.

Chapter 3 details the research methodology. This includes the specific preprocessing steps applied to the dataset, the architectural breakdown of the custom designed CNN and ViT models, the complete experimental setup including the software and hardware environments, and the evaluation metrics used to assess model performance.

Chapter 4 presents the empirical results of the experiments. It provides a comprehensive comparison of all developed models, showcasing their performance through final evaluation metrics and training curves.

Chapter 5 a discussion and interpretation of the results. It analyzes the performance of each model, provides a comparative perspective on architectural trade-offs, contextualizes the findings by comparing them against existing literature, and acknowledges the limitations of this study.

Chapter 6 concludes the thesis. It summarizes the key findings of the research, reiterates the contributions made, and suggests potential directions for future work.

Chapter 2

Background and Literature Review

2.1 Skin Lesions and Dermatoscopy

The diagnosis of skin lesions is a cornerstone of dermatology. While the human eye is a powerful tool, its capabilities are limited to the surface of the skin. Many key diagnostic features of skin lesions lie beneath the stratum corneum, the outermost layer of the epidermis. The natural reflection and scattering of light from this layer obscure these deeper structures, posing a challenge for accurate visual assessment.

Dermatoscopy (also known as dermoscopy or epiluminescence microscopy) is a noninvasive diagnostic technique developed to overcome this limitation [1]. It utilizes a specialized handheld instrument called dermatoscope, which combines high magnification with a liquid interface or cross polarized lighting. This system minimizes surface light scatter and renders the subsurface structures of the skin visible, from the epidermis down to the reticular dermis. This enhanced view allows clinicians to identify specific patterns, colors, and vascular structures that are not visible to the naked eye, significantly improving diagnostic accuracy [2].

The HAM10000 dataset, includes seven important diagnostic categories of skin lesions. An overview of each is provided below [3].

Actinic Keratosis (AKIEC) These are pre-cancerous lesions caused by long term sun exposure. They typically appear as rough, scaly, or crusty patches on sun exposed areas like the head and hands. They have the potential to progress into squamous cell carcinoma if left untreated.

Basal Cell Carcinoma (BCC) This is the most common form of skin cancer. It arises in the basal cells of the epidermis and often appears as a slightly transparent, pearly bump, though it can also present as a flat brown scar-like lesion. BCCs are slow growing and rarely metastasize, but require treatment to prevent local tissue damage.

Benign Keratosis (BKL) This is a broad category of benign growths that can sometimes mimic melanoma. It primarily includes seborrheic keratosis (a common, waxy looking growth), sun spots, and lichenoid keratoses. They are non cancerous and generally do not require treatment unless they become irritated.

Dermatofibroma (DF) A common and benign cutaneous nodule, a dermatofibroma is a firm bump that develops in the dermis layer of the skin. They are typically small,

brown or reddish, and may dimple inwards when pinched. Their exact cause is unknown, but they can sometimes arise after minor trauma like an insect bite.

Melanoma (MEL) The most serious type of skin cancer, melanoma develops in the melanocytes, the cells that produce melanin. It is highly aggressive and has a strong potential to metastasize to other parts of the body if not detected early. Clinically, it often presents as an asymmetrical lesion with an irregular border, varied color, a diameter larger than 6 mm, and may evolve in size or shape over time.

Melanocytic Nevi (NV) Commonly known as moles, these are benign growths of melanocytes. They are extremely common and are typically small, symmetrical, well defined, and uniform in color. They are diagnostically important as certain types can be precursors to or resemble melanoma.

Vascular Lesions (VASC) This category includes a range of conditions related to blood vessels, such as angiomas and angiokeratomas. They can appear as small red or purple spots or bumps and are generally benign.

2.2 The HAM10000 Dataset

A significant challenge in training robust deep learning models for medical imaging is the availability of diverse and well annotated datasets. To address this issue in the context of dermatology, the HAM10000 (Human Against Machine with 10000 training images) dataset was developed. It has since become a standard benchmark for developing and evaluating machine learning algorithms for skin lesion classification.

The dataset consists of 10,015 dermatoscopic images collected from different populations over a 20 year period from clinics in Australia and Austria. This multi source approach ensures a high degree of diversity in the data, capturing variations in skin types and imaging conditions. A key strength of HAM10000 is the quality of its ground truth annotations. Over 50% of the lesions in the dataset have been confirmed by histopathology (pathological examination of a biopsy), which is the gold standard for diagnosis. The ground truth for the remaining images was established through either expert consensus among dermatologists or by confocal microscopy, a high resolution imaging technique.

The dataset is highly imbalanced, which reflects the real world prevalence of different skin lesions. The distribution across the seven classes is heavily skewed towards Melanocytic Nevi, which make up over two thirds of the images. This class imbalance presents a significant challenge for model training and is a key characteristic that must be addressed in the methodology of any study using this data.

2.3 Neural Network Fundamentals

Artificial Neural Networks (ANNs) are computational models inspired by the structure and function of biological neural networks. Modern deep learning is a recent phenomenon, but its conceptual origins date back to the mid 20th century. The first mathematical model of a neuron was proposed by McCulloch and Pitts in 1943 [4], describing a simple

computational unit that could make logical decisions. This was followed by the development of the Perceptron by Frank Rosenblatt in the late 1950s [5], which introduced a learning rule that allowed the network to adjust its weights based on examples.

A significant obstacle for early neural network development was the Perceptron's inability to learn non-linearly separable patterns [6]. This limitation led to a period of reduced interest in the field. The breakthrough that reignited progress was the adoption of the backpropagation algorithm in the 1980s, enabling the training of multi layered networks and laying the groundwork for modern deep learning [7].

2.3.1 The Artificial Neuron

The fundamental unit of an ANN is the artificial neuron. It receives one or more inputs, computes a weighted sum of these inputs, adds a learnable constant known as a bias, and then passes this result through a non linear activation function. Mathematically, the output y of a single neuron is given by:

$$y = f\left(\sum_{i=1}^{n}(w_i x_i) + b\right) \tag{2.1}$$

where x_i are the inputs, w_i are their corresponding weights, b is the bias, and f is the activation function. The weights and biases are the parameters of the network that are learned during training. The weights determine the strength of the influence of each input on the output, while the bias acts as a tunable threshold, allowing the activation function to be shifted to better fit the data.

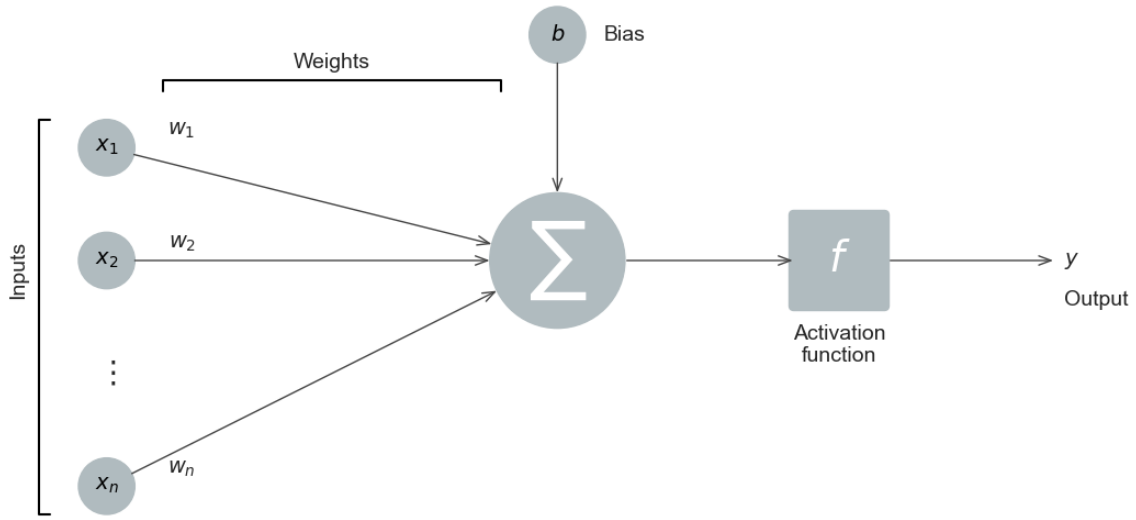


Figure 2.1: Diagram of a single artificial neuron.

2.3.2 Network Architecture

Artificial neurons are organized into a series of layers to form a complete network. The most common arrangement, known as a Multi-Layer Perceptron (MLP), is a sequence

of layers beginning with an input layer, followed by one or more hidden layers, and concluding with an output layer. Information flows through the network sequentially from one layer to the next in a process called forward propagation.

The input layer is the entry point of the network. It does not perform any computation, its role is simply to receive the raw data and pass it to the first hidden layer. The number of neurons in this layer corresponds directly to the number of features in the input data. For image classification, this typically means one neuron for each pixel value in an image.

The hidden layers are the computational engine of the network. In a standard MLP, these are fully connected layers, every neuron in a hidden layer receives input from every neuron in the preceding layer. Within these layers, the network applies successive transformations to the data, learning to extract complex and abstract features. The number of hidden layers determines the network's depth and the number of neurons in each layer defines its width. The combination of depth and width dictates the overall capacity of the network. A network with multiple hidden layers is called a deep neural network.

The output layer is the final stage, responsible for producing the model's prediction. For a multi-class classification problem, the output layer typically has one neuron for each class. These neurons produce raw output scores, or logits, which are then converted into a final prediction by an appropriate activation function, as discussed in the following section.

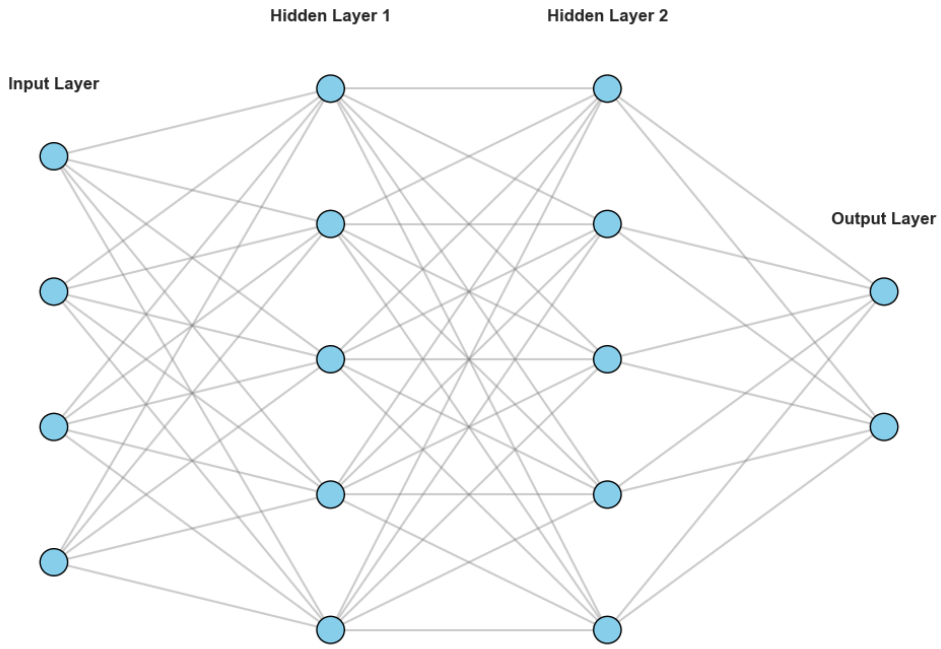


Figure 2.2: Architecture of a Multi-Layer Perceptron (MLP).

2.3.3 Activation Functions

Activation functions play a fundamental role in neural networks by introducing nonlinear transformations, which enable the learning of complex patterns. Without them, a deep network would be functionally equal to a single layer linear model, severely limiting its capacity.

The choice of activation function typically differs between the hidden layers and the output layer. For hidden layers, a variety of functions can be used. Historically, functions like the sigmoid

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.2}$$

and the hyperbolic tangent (tanh)

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2.3}$$

were common. Recently, their use has declined as they are prone to the vanishing gradient problem, which can stall network training [8]. The modern standard is the Rectified Linear Unit (ReLU), widely adopted due to its computational efficiency and its effectiveness at mitigating vanishing gradients. It is defined as:

$$\mathrm{ReLU}(z) = \max(0, z) \tag{2.4}$$

where z denotes the input. ReLU has its own drawbacks. For instance, it can suffer from the "dying ReLU" problem, where neurons become permanently inactive if their input is always negative [9]. To address this limitation, several variants have been proposed. One popular and effective alternative, particularly in Transformer architecture, is the Gaussian Error Linear Unit (GELU) [10]. GELU provides a smoother activation curve and is defined as:

$$\mathrm{GELU}(z) = z \cdot \Phi(z) \tag{2.5}$$

where $\Phi(z)$ is the standard Gaussian cumulative distribution function. A common approximation used in practice is:

$$\mathrm{GELU}(z) \approx 0.5z \left(1 + \tanh\left[\sqrt{2/\pi}(z + 0.044715z^{3})\right]\right) \tag{2.6}$$

For the output layer, the activation function is chosen based on the task. Unlike the element wise functions used in hidden layers, multi-class classification requires a function that operates on the entire output vector. The Softmax function is used for this purpose. It takes the raw, unnormalized scores (logits) from the final layer and converts them into a probability distribution over the M possible classes. Given a vector of logits z, the probability for the j-th class is:

$$\text{Softmax (z)}_{j} = \frac{e^{z_j}}{\sum_{k=1}^{M} e^{z_k}} \tag{2.7}$$

The result is a vector of probabilities where each element is between 0 and 1, and all elements sum to 1, representing the model's confidence for each class.

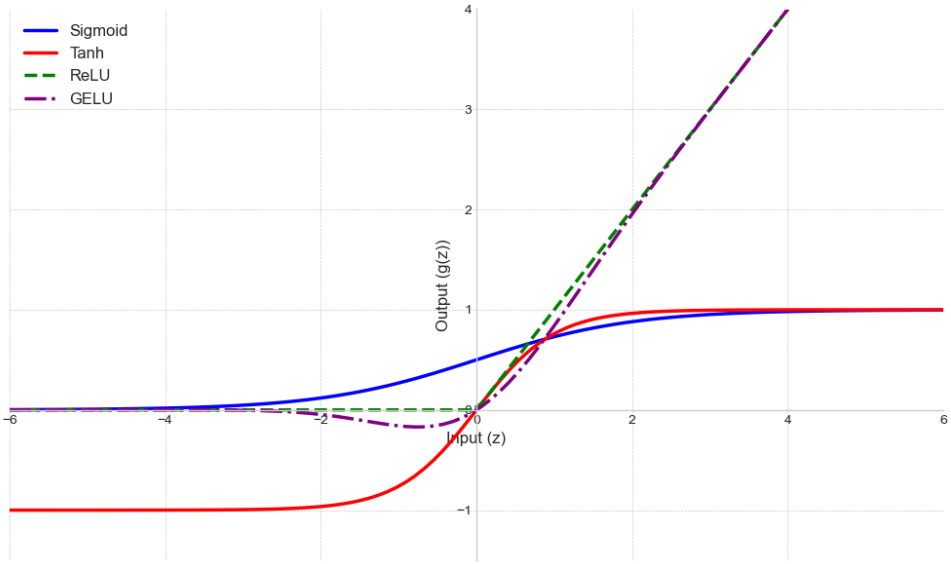


Figure 2.3: Comparison of common hidden layer activation functions.

2.3.4 The Learning Process

The learning process of a neural network can be seen as an optimization task. During training, the network iteratively updates its internal parameters, the weights and biases, in order to reduce the error between its predictions and the actual targets. This error is measured using a loss function. To minimize the loss, the network relies on a method called backpropagation, which efficiently computes the gradient of the loss with respect to each parameter. These gradients are then used by an optimization algorithm to adjust the parameters in the direction that minimizes the loss, gradually improving the model's performance over time [11].

2.3.4.1 Loss Functions

A loss function, $\mathcal{L}(\hat{y}, y)$, measures the discrepancy between the model's predicted output, $\hat{y}$, and the true label, y. The value of the loss function is what the network aims to minimize during training.

For a multi-class classification problem, the standard loss function is Categorical Cross Entropy. This function is designed to work directly with the probability distribution produced by the Softmax output layer. It evaluates how well the predicted probability distribution $\hat{y}$ matches the true class label, which is typically represented as a one-hot encoded vector (e.g., [0, 0, 1, 0, 0, 0, 0] for the third class). For a single example, the loss is given by:

$$\mathcal{L}(\hat{y}, y) = -\sum_{c=1}^{M} y_c \log(\hat{y}_c) \tag{2.8}$$

This loss is minimized when the predicted probability for the correct class approaches 1. It penalizes the model heavily for being confident in the wrong prediction, making it an effective objective for training classifiers.

2.3.4.2 Backpropagation

Backpropagation is the key algorithm that makes the training of neural networks efficient. It computes the gradient of the loss function with respect to each parameter of the network by applying the chain rule of calculus. Conceptually, the gradient is a vector of partial derivatives that points in the direction of the steepest ascent of the loss and by moving in the opposite direction, the model can effectively minimize the loss. This process allows the model to determine how each weight and bias should be adjusted to decrease the error.

The process consists of two main passes through the network:

Forward pass The input data is passed through the network's layers to compute the final output $\hat{y}$ and the value of the loss function, $\mathcal{L}(\hat{y}, y)$.

Backward pass The algorithm steps backward from the loss, propagating an error signal from the output layer to the input layer.

For a single layer with input x, weights W, and bias b, the layer's output before activation is $z = Wx + b$. The backpropagation algorithm computes the gradient of the loss $\mathcal{L}$ with respect to this layer's parameters. The gradient for the weights W and bias b are given by the chain rule:

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial W} = \delta x^T \tag{2.9}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial b} = \delta \tag{2.10}$$

Here, $\delta = \frac{\partial \mathcal{L}}{\partial z}$ is the error signal for that layer, which is propagated backward from the subsequent layer.

2.3.4.3 Optimization Algorithms

The goal of training is to find the set of parameters, θ, that minimizes the total loss. This is achieved through an iterative process guided by the gradient of the loss function, $\nabla_\theta \mathcal{L}$, which is calculated efficiently using the backpropagation algorithm.

The foundational optimization algorithm is Stochastic Gradient Descent (SGD). It updates the network's parameters by taking a small step in the opposite direction of the gradient, which is calculated on a mini batch of training data. The update rule is:

$$\theta_t = \theta_{t-1} - \eta g_t \tag{2.11}$$

where θ_t represents the model's parameters at iteration t, η is the learning rate that controls the step size, and g_t is the gradient of the loss with respect to the parameters. SGD's main limitations are that its update direction can be very noisy and it uses a single, fixed learning rate for all parameters. To address these issues, more advanced optimizers that adapt the learning rate for each parameter are now commonly used.

Adaptive Moment Estimation (Adam) is one of the most popular of these optimizers [12]. It computes adaptive learning rates for each parameter by maintaining two moving averages:

m_t The exponentially decaying average of past gradients (first moment).

v_t The exponentially decaying average of past squared gradients (second moment).

At each iteration t, with gradient g_t, the moments are updated as:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.12}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.13}$$

where β_1 and β_2 are hyperparameters that control the decay rates of these moving averages. These estimates are then bias corrected:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.14}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.15}$$

The final parameter update rule is:

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.16}$$

where η is the learning rate, $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected moment estimates, and ϵ is a small constant for numerical stability.

AdamW (Adam with Weight Decay Decoupling) is an improved version of Adam that enhances regularization in deep learning [13]. In standard Adam, L2 regularization (weight decay) is coupled with the gradient updates, which can distort the adaptive moment estimates and lead to poor regularization and convergence. AdamW fixes this by separating weight decay from the gradient updates. It applies the regularization directly to the parameters:

$$\theta_t = (1 - \eta\lambda)\theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.17}$$

The $(1 - \eta\lambda)\theta_{t-1}$ term adds pure L2 regularization, separate from the adaptive learning rate based on $\hat{m}_t$ and $\hat{v}_t$.

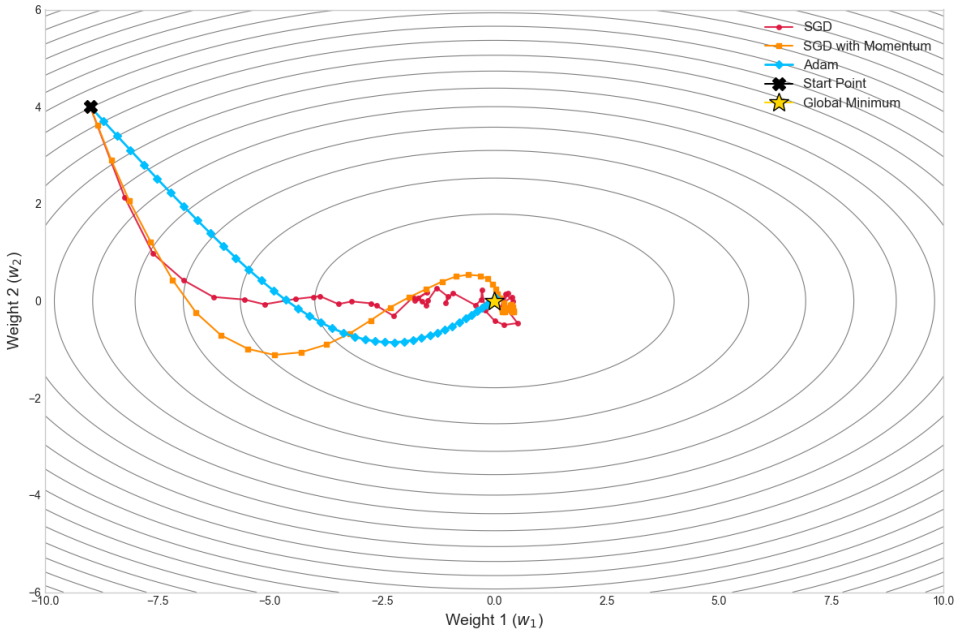


Figure 2.4: Optimization trajectories of different algorithms on a loss surface.

2.3.4.4 Batch Normalization

Training very deep neural networks can be challenging because the distribution of each layer's inputs changes during training as the parameters of the previous layers change. This phenomenon, known as internal covariate shift, can slow down training and require careful parameter initialization.

Batch Normalization (BatchNorm) is a technique designed to address this problem [14]. It normalizes the input to an activation layer by subtracting the mini batch mean and dividing by the mini batch standard deviation. For a given activation x_i in a mini-batch $\mathcal{B}$, the normalized activation $\hat{x}_i$ is:

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \tag{2.18}$$

where $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}^2$ are the mean and variance over the mini batch, and ϵ is a small constant. This normalization is followed by a learnable transformation:

$$y_i = \gamma \hat{x}_i + \beta \tag{2.19}$$

where γ (scale) and β (shift) are learnable parameters. This allows the network to learn the optimal scale and mean for the activations and can revert to the original distribution if needed.

BatchNorm has several major benefits. It allows for much higher learning rates, significantly accelerates training convergence, acts as a form of regularization, and makes the network less sensitive to weight initialization.

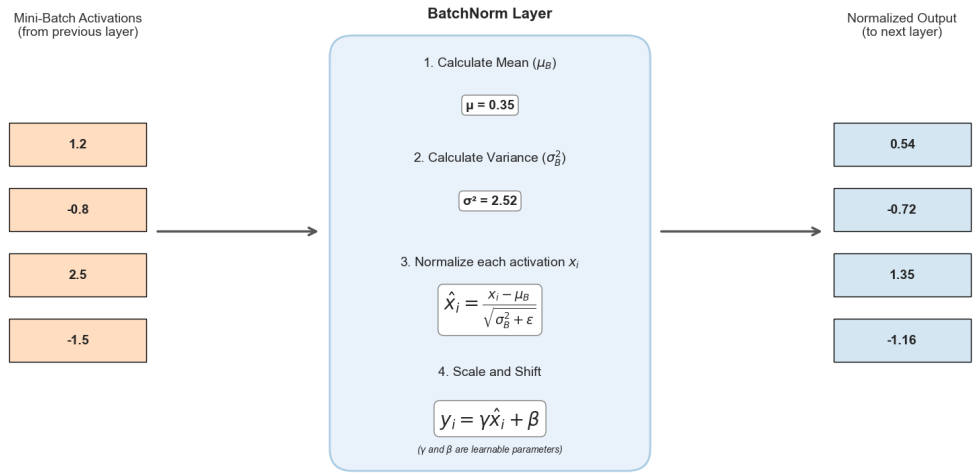


Figure 2.5: The Batch Normalization process within a neural network layer.

2.3.4.5 Dropout

Deep neural networks with a large number of parameters are highly flexible and this makes them prone to overfitting. Overfitting occurs when a model learns the training data too

well, including its noise, and fails to generalize to new, unseen data. To address this, a powerful and widely used regularization technique called Dropout was developed [15].

The core idea of Dropout is simple. During each forward pass of the training phase, the outputs of a random fraction of neurons in a given layer are set to zero. The probability of a neuron being dropped, known as the dropout rate p, is a hyperparameter typically set between 0.1 and 0.5. This means that for each training sample, the network is forced to operate with a different, thinned architecture, preventing neurons from becoming overly reliant on each other.

Dropout is only active during training. For inference, it is disabled, and all neurons are used. In modern frameworks like PyTorch, an efficient technique called inverted dropout is implemented. Instead of scaling the outputs down during inference, the outputs of the neurons that are not dropped during training are scaled up by a factor of $1/(1-p)$. This ensures the expected value of the outputs is consistent, while leaving the network unchanged and more efficient during inference.

The primary effect of this process is that it prevents complex adaptations among neurons. This encourages the network to learn more robust features that are useful in connection with many different random subsets of other neurons, improving the model's generalization performance.

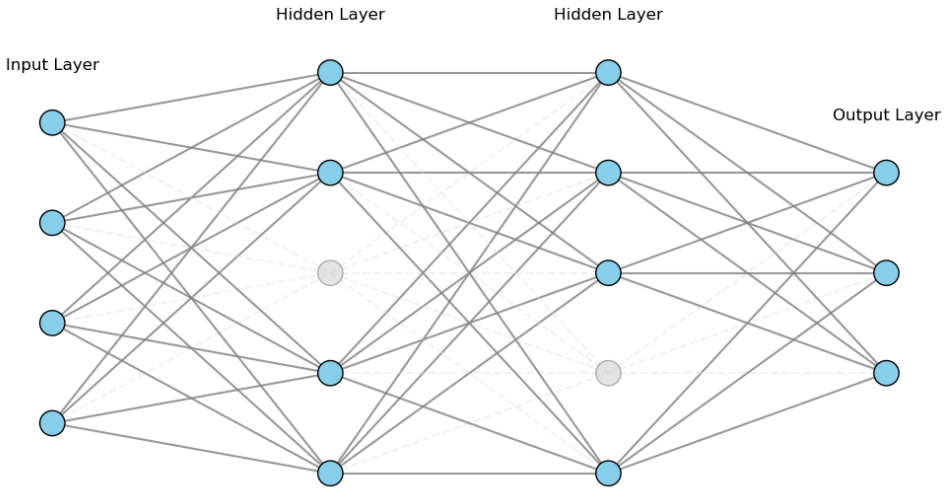


Figure 2.6: Illustration of the Dropout technique.

2.3.4.6 Learning Rate Schedulers

Learning rate η is a really important hyperparameter in training. Initially, a larger learning rate can help the model make rapid progress and converge quickly towards a good region of the loss landscape. As the model gets closer to a minimum, a smaller learning rate is needed to make smaller adjustments and avoid overshooting the optimal set of parameters. Learning rate schedulers are used to automate this process of dynamically adjusting the learning rate over time [16].

A learning rate scheduler dynamically adjusts the learning rate during training according to a predefined strategy. Two common approaches are Step Decay and Cosine Annealing. Step Decay reduces the learning rate by a multiplicative factor, for example by 0.1, at specific, pre defined epochs. This creates a learning rate that remains constant for a period and then drops sharply at set intervals. Cosine Annealing is a popular

method that smoothly decays the learning rate from an initial high value to a minimum value, often close to zero, following the curve of a cosine function. This gradual reduction can help models settle into broader and more robust minima.

2.4 Convolutional Neural Networks (CNNs)

Using standard fully connected networks for raw images is impractical. This approach requires flattening an image into a single long vector, a process that destroys the image's 2D structure and loses crucial spatial information. It also creates a huge number of parameters, which makes the network very slow to train and highly prone to overfitting. This is a major concern when working with high resolution medical images like those in the HAM10000 dataset.

CNNs were designed to solve these exact problems. Inspired by the processing in the human visual cortex [17], their architecture is built to work directly with the grid structure of an image. CNNs use small, localized filters that share weights as they move across the image. This design preserves spatial information and dramatically reduces the total parameter count, making them a popular choice for computer vision tasks.

2.4.1 The Convolutional Layer

The foundational building block of a CNN is the convolutional layer. It applies a small sliding window, called a kernel or filter, typically a matrix of learnable weights, across the input image. At each spatial position, the kernel computes a dot product with the overlapping region of the input, producing a feature map. This feature map captures spatial patterns such as edges, textures, and shapes. These localized features are essential for detecting dermatological patterns and lesion boundaries. The key to this layer's efficiency is parameter sharing, where the same kernel is applied across the entire image. This technique drastically reduces the total number of parameters and creates translation invariance, which allows the network to detect a feature regardless of its position. Other settings like stride and padding provide control over the output feature map's size.

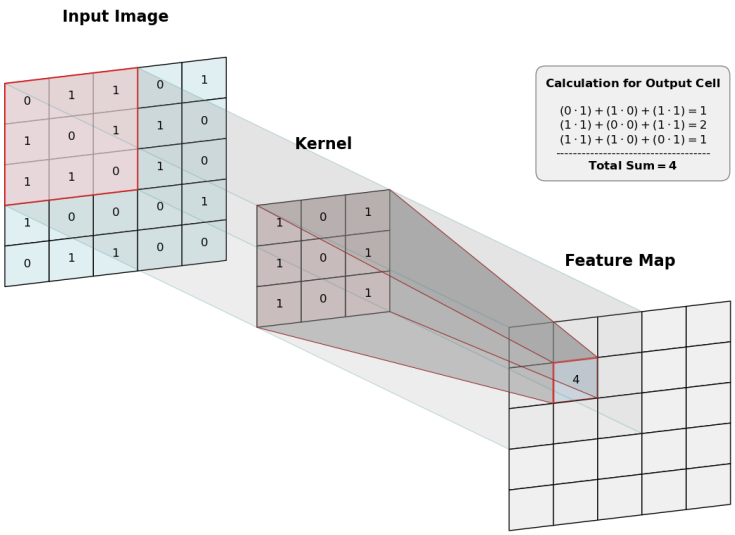


Figure 2.7: Illustration of a convolutional layer operation.

2.4.2 The Pooling Layer

A pooling layer is often used after a convolutional layer to downsample, or shrink, its feature maps. This process has two main benefits. First, it reduces the number of parameters and computations for later layers. Second, it makes the model more robust to small shifts or rotations in the input, a useful property for handling variations in dermatoscopic images. This robustness is known as local spatial invariance.

The two most common types of pooling are max-pooling and average-pooling. Max-pooling selects the strongest signal (the maximum value) from a small window of the feature map, preserving prominent details like sharp lesion edges. In contrast, average-pooling calculates the average value within the window, which provides a smoother feature representation. By summarizing information, both methods also help prevent the model from overfitting to noise in the training data.

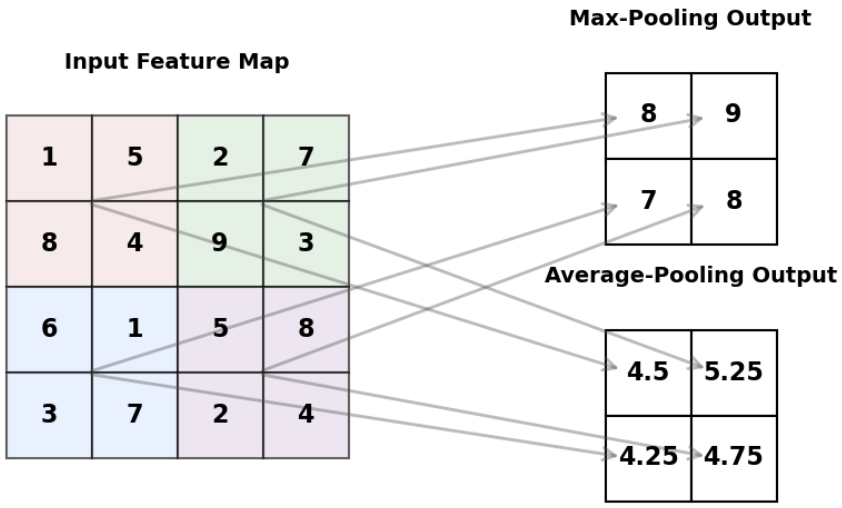


Figure 2.8: Demonstration of max-pooling and average-pooling operations.

2.4.3 Overall CNN Architecture

A complete CNN is built by stacking several blocks. Each block typically contains a convolutional layer, a ReLU activation function, and a pooling layer. This stacked design creates a feature hierarchy. Early layers in the network learn simple features like lines and color gradients, which are useful for finding lesion outlines. Deeper layers then combine these simple patterns to recognize more complex things, such as specific textures and shapes.

After the last feature extraction block, the final 2D feature maps are flattened into a single, one-dimensional vector. This vector is then fed into a standard fully connected network, often called the classifier head. This part of the network makes the final decision, using a Softmax activation to output the class probabilities.

2.4.4 Landmark Architectures

Several landmark models have shaped the evolution of modern CNNs. The deep learning revolution was effectively launched by AlexNet, which won the 2012 ImageNet competition by a large margin [18]. Later, VGGNet demonstrated the power of very deep networks by stacking simple, repeating 3×3 convolutional blocks [19]. A more fundamental breakthrough came with ResNet, which introduced residual connections [20]. This technique solved a major problem with vanishing gradients, making it possible to effectively train networks with hundreds or even thousands of layers. The core ideas from these landmark models are still central to computer vision and continue to influence the design of new architectures in specialized fields like medical imaging.

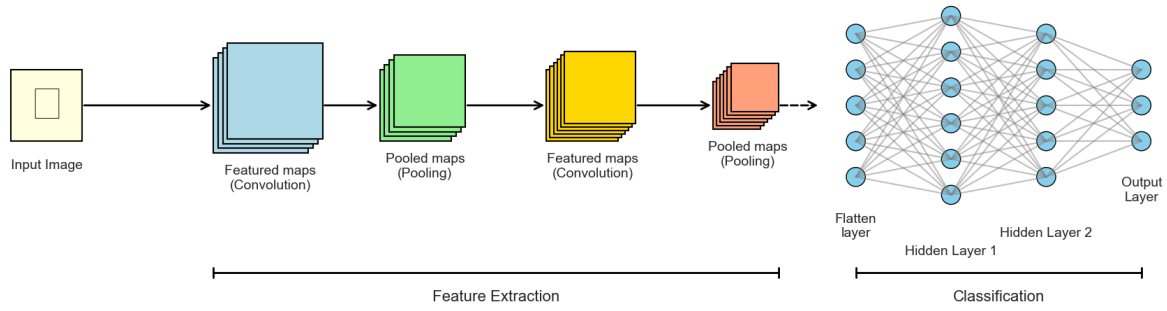


Figure 2.9: A Typical CNN Architecture.

2.5 Vision Transformers (ViTs)

While the CNNs discussed in the previous section excel at learning from local features, their reliance on localized kernels can make it difficult to model long range, global dependencies within an image. This characteristic has motivated the exploration of alternative architectures that are not constrained by local receptive fields.

A powerful new approach emerged from the field of natural language processing (NLP) with the introduction of the Transformer [21]. The Transformer's core innovation, the self-attention mechanism, allowed it to process an entire sequence of text at once, weighing the influence of every word on every other word. Its state of the art performance prompted researchers to adapt this powerful architecture for vision, leading to the development of the Vision Transformer (ViT) for image classification [22].

2.5.1 From Image to Sequence: Patch Embeddings

The primary challenge in applying a Transformer to an image is that self-attention mechanism expects a sequence of tokens, not just a 2D grid of pixels. ViT model solves this with a simple strategy: it divides the input image into a sequence of fixed sized patches. For example, a 224×224 image can be broken into 196 non overlapping patches, each with a size of 16×16. Each patch is then flattened into a vector. This vector is then mapped to a different dimensional space using a learnable linear projection, creating a

patch embedding. This embedding is a dense vector representation that captures the visual features of the patch in a format that the Transformer encoder can process [22].

To retain spatial information, which is lost during the flattening process, learnable positional embeddings are added to each patch embedding. This gives the model information about the original location of each patch. Finally, a special learnable class token (CLS) is prepended to the sequence. The final output corresponding to this token serves as the aggregate image representation for the classification head.

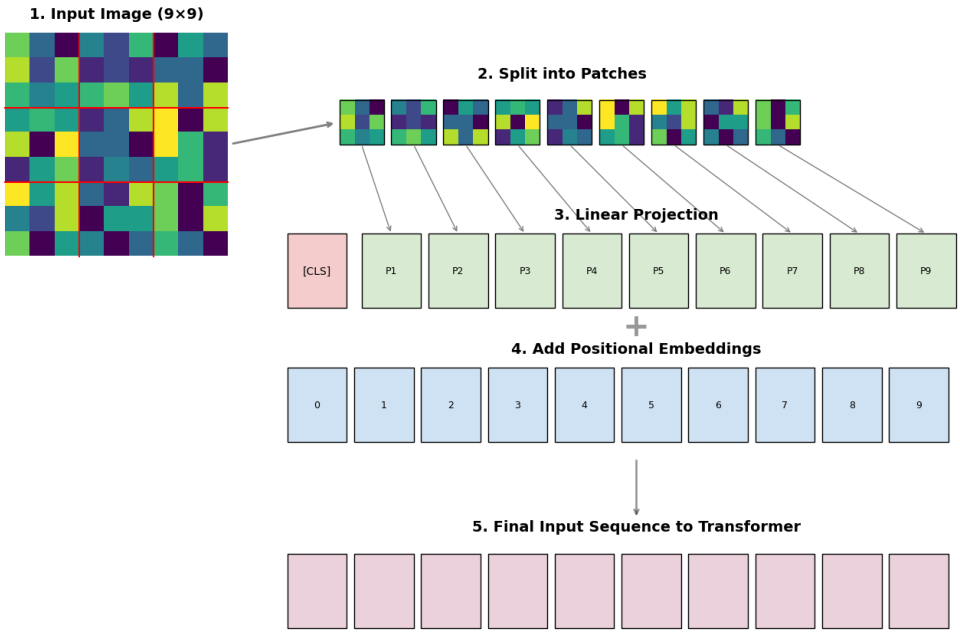


Figure 2.10: The ViT input process.

2.5.2 The Transformer Encoder

The core of the ViT is the Transformer Encoder, which is built from a stack of identical layers. Each layer in the stack improves upon the features learned by the layer before it. A layer has two main parts: first, a Multi-Head Self-Attention mechanism computes pairwise interactions between all patches, enabling the model to integrate global contextual information across the entire image. Second, a Feed-Forward Network then processes each patch's information by itself. ViT's ability to see the whole image right from the start is a key difference from CNNs and is what allows them to understand how distant parts of an image relate to each other [21].

2.5.2.1 The Self-Attention Mechanism

The self-attention mechanism works on the principle of relating different positions of a single sequence to compute a representation of that sequence [21]. To do this, it first creates three vectors from each input patch embedding by multiplying the embedding by three distinct, learnable weight matrices (W^Q, W^K, and W^V). This produces a Query (Q), a Key (K), and a Value (V) vector for each patch. The Query represents the current

patch's request for information. The Keys from all other patches act as labels for the information they hold, and the Values contain the actual content to be shared.

The process, known as Scaled Dot-Product Attention, uses the Query and Key vectors to compute a set of attention scores. These scores are scaled and converted into probabilities using a softmax function. The final output is a weighted sum of all Value vectors, where the weights are the probabilities just calculated. This operation is captured in a single equation:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.20}$$

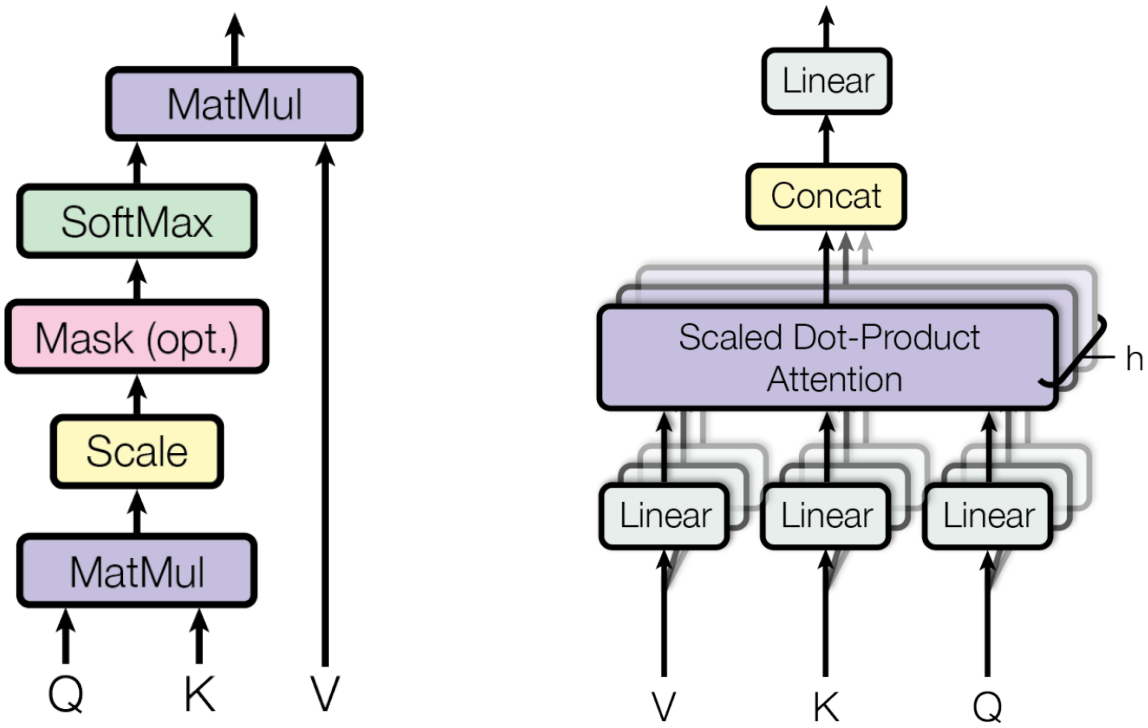
where d_k is the dimension of the key vectors [21].

Instead of performing this attention calculation once, the Transformer uses a Multi-Head Self-Attention mechanism. This approach was found to be more powerful as it allows the model to jointly attend to information from different representation subspaces. The queries, keys, and values are first projected into multiple heads using different, learnable linear projections. The attention mechanism is then applied to each head in parallel. The outputs of these heads are then concatenated and passed through a final linear projection to produce the output. This is formally defined as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.21}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.22}$$

where the projections are parameter matrices W_i^Q, W_i^K, W_i^V, and W^O [21].



(a) Scaled Dot-Product Attention. **(b)** Multi-Head Attention mechanism.

Figure 2.11: The components of the self-attention mechanism used in Transformers.

2.5.2.2 Feed-Forward Network

The second major component in an encoder block is a simple, fully connected feed-forward network (FFN). This network is applied independently to each patch representation after the self-attention step. It typically consists of two linear layers with a non-linear activation function, like GELU, in between. The purpose of the FFN is to provide further processing of each patch's representation, adding capacity to the model and transforming the features learned by the attention mechanism. Both the self-attention and FFN sub-layers have residual connections around them followed by layer normalization, which helps stabilize the training of these deep architectures.

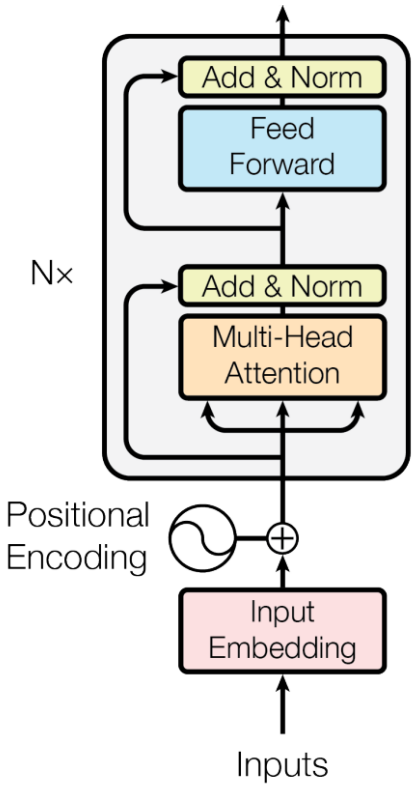


Figure 2.12: Architecture of a standard Transformer Encoder block.

2.5.3 Overall ViT Architecture

A complete ViT is built by stacking multiple Transformer Encoder blocks one after another. The sequence of patch embeddings is passed through this entire stack, and at each layer, the model refines its understanding of the global relationships between all the image patches. This process allows the model to build progressively more complex and abstract representations.

After the sequence has been processed by the final encoder block, the representation associated with the special CLS token is taken. This single vector acts as the broad representation of the entire image, containing all the information needed for classification. This vector is then fed into a standard fully connected network, which makes the final decision using a Softmax activation to output the class probabilities.

Chapter 3

Methodology

3.1 Dataset and Preprocessing

The foundation of any deep learning project is the data it is using and the process used to prepare that data. This section details the HAM10000 dataset used for all experiments, the cross-validation strategy employed to ensure robust evaluation, and the specific preprocessing and data augmentation pipelines.

3.1.1 The HAM10000 Dataset

The HAM10000 dataset used in this study contains 10,015 dermatoscopic images of pigmented skin lesions in JPEG format, each with a resolution of 600×450 pixels. These images are distributed across seven diagnostic categories, with representative examples shown in Figure 3.1.

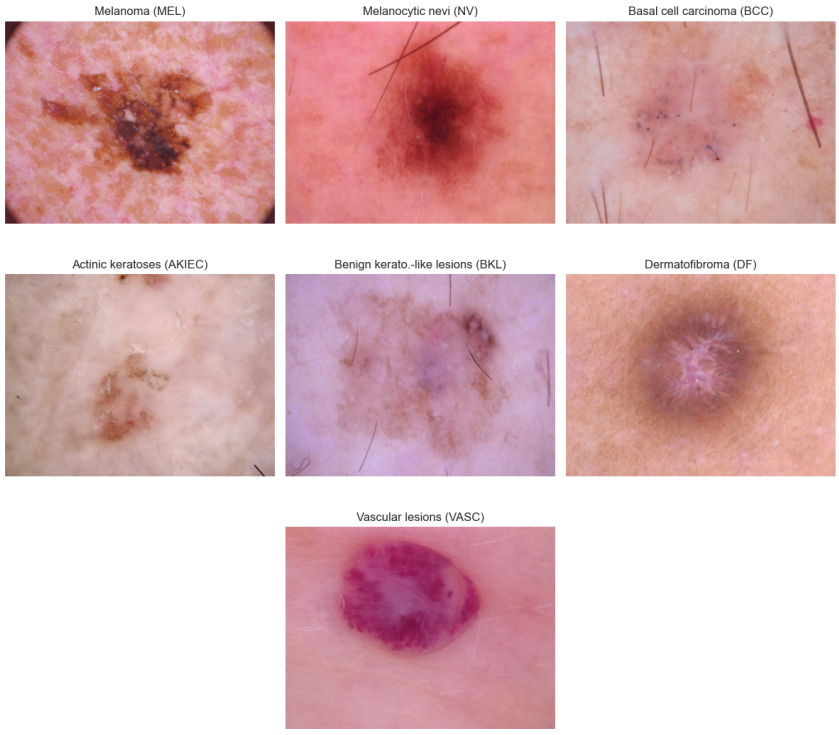


Figure 3.1: Example dermatoscopic images from the HAM10000 dataset.

The overall distribution of these classes is as follows:

Melanocytic nevi (NV) 6,705 images

Melanoma (MEL) 1,113 images

Benign keratosis-like lesions (BKL) 1,099 images

Basal cell carcinoma (BCC) 514 images

Actinic keratoses (AKIEC) 327 images

Vascular lesions (VASC) 142 images

Dermatofibroma (DF) 115 images

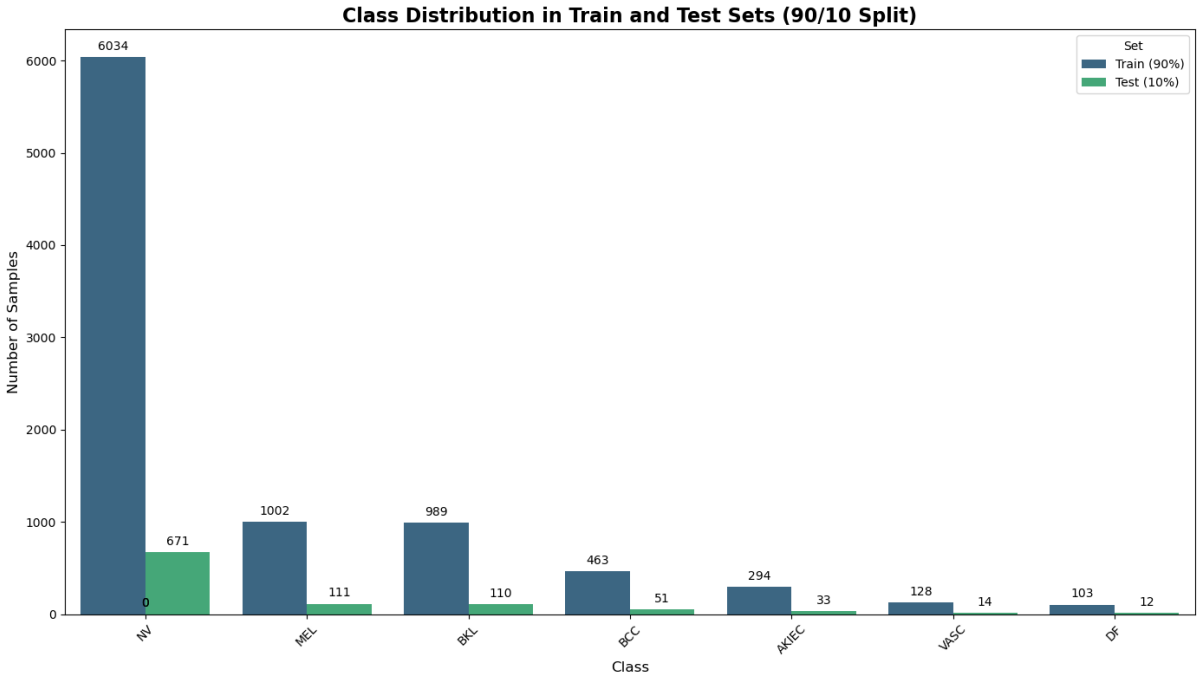


Figure 3.2: Class distribution of the HAM10000 dataset.

3.1.2 Validation Strategy: Stratified K-Fold Cross-Validation

To ensure a robust and reliable evaluation of model performance, a 10-fold stratified cross-validation strategy was employed. In this procedure, the entire dataset is partitioned into 10 equally sized folds. For each of the 10 iterations, one fold is held out as the test set, while the remaining nine folds are used for training, as illustrated in Figure 3.3. The "stratified" nature of the split ensures that each fold maintains the same class distribution as the overall dataset, which is critical for handling the class imbalance.

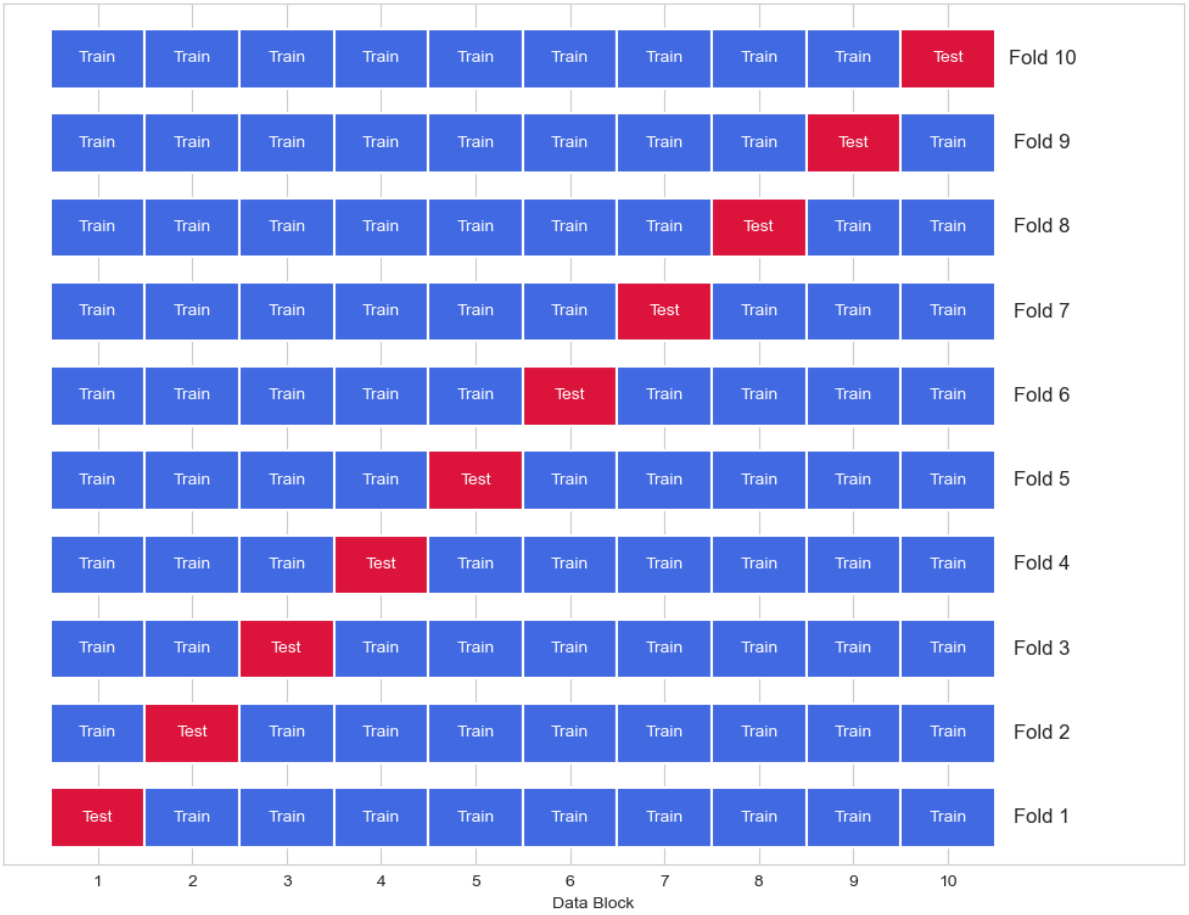


Figure 3.3: Illustration of the 10-fold cross-validation process. The dataset is partitioned into 10 folds, and training is repeated 10 times, with each fold serving as the test set exactly once.

This approach provides two key benefits. First, it ensures that every image in the dataset is used in the test set exactly once, making maximum use of the available data for validation. Second, it provides a more stable and reliable estimate of a model's true performance compared to a single train-test split. The final performance metrics reported in this thesis are the average and standard deviation of the results obtained across all 10 folds. To guarantee a fair and direct comparison between all experiments, a fixed random seed (42) was used for the fold creation, ensuring that every model was trained and tested on the exact same data splits.

3.1.3 Data Augmentation and Preprocessing

Data preprocessing and augmentation are critical steps for training deep learning models. Preprocessing standardizes the input images, while augmentation artificially expands the training set to help the model generalize better and reduce overfitting.

For all models in this study, input images were resized to 224×224 pixels and normalized using the standard ImageNet [23] mean and standard deviation ($\mu = [0.485, 0.456, 0.406]$, $\sigma = [0.229, 0.224, 0.225]$). The validation and test sets were never augmented. Images from these sets were only resized and normalized.

Two distinct augmentation strategies were used for the training data, as illustrated in Figure 3.4.

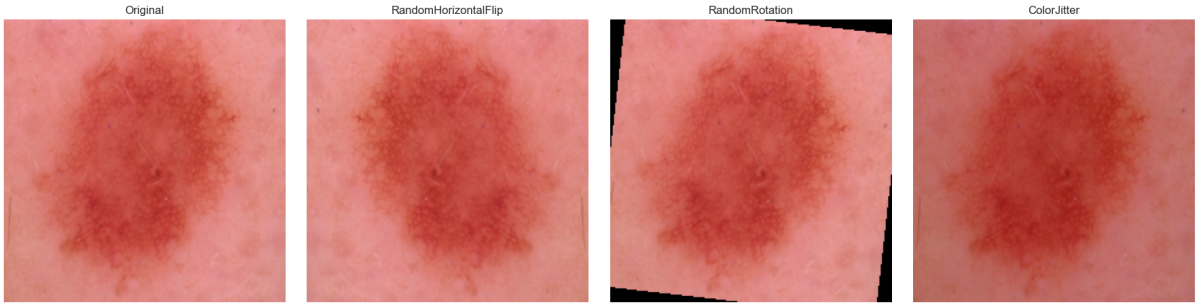


Figure 3.4: An example of data augmentation. The original image (top left) is shown with several augmented versions generated by applying random transformations.

Standard Augmentation The primary strategy, applied to all CNN models and the first four ViT models, consisted of a set of light augmentations: random horizontal flips and random rotations of up to 10 degrees. This standard pipeline was designed to increase data variance without drastically altering the images.

Advanced Augmentation For ViT Model 5, a more advanced strategy combining Mixup [24] and Cutmix [25] was used. As illustrated in Figure 3.5, these techniques create new training samples by combining two existing images and their labels, which acts as a powerful form of regularization.

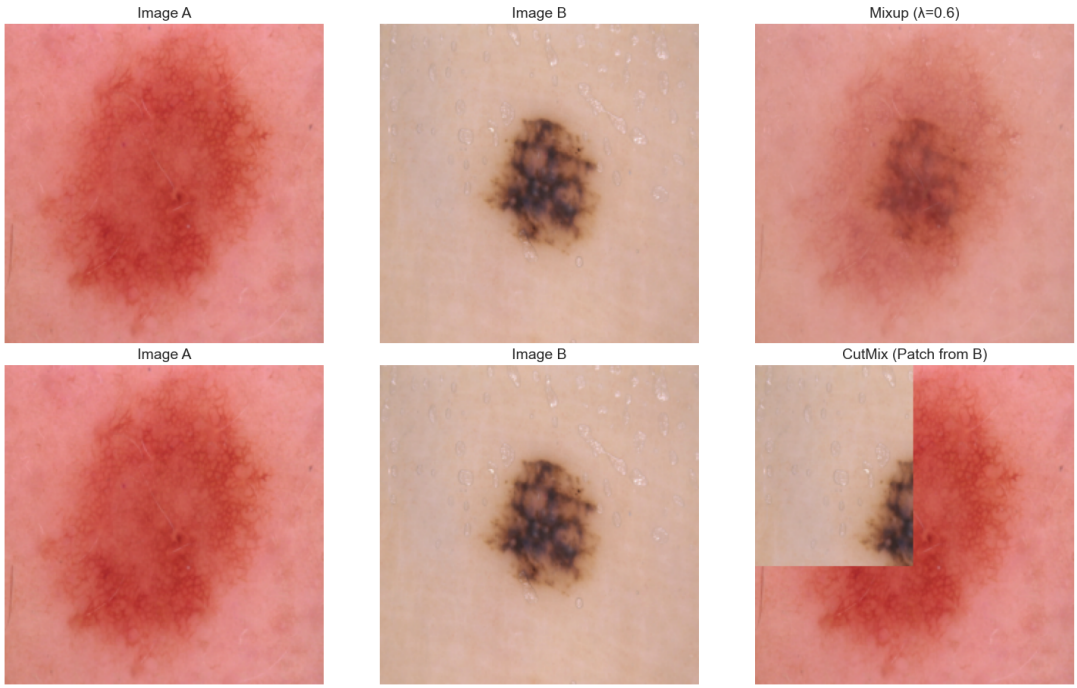


Figure 3.5: Illustration of Mixup and CutMix augmentation. Mixup (top row) blends two images, while CutMix (bottom row) pastes a patch from one image onto another.

3.2 CNN Models

This section details the four custom CNN architectures designed and evaluated in this study. The models were developed with a strategy of incremental complexity, starting

with a simple baseline and progressively incorporating advanced techniques such as Batch Normalization, Dropout, and residual connections. All models utilize the ReLU activation function within their hidden layers and are designed for an input resolution of 224×224. This approach enables a systematic analysis of architectural choices, including network depth, regularization, and skip connections.

Model 1 (Baseline) This shallow CNN serves as a performance baseline, consisting of two convolutional blocks, each with 3×3 filters (16 and 32 channels, respectively), stride 1, and 2×2 max pooling, followed by a classifier with a single hidden layer (128 units). Due to a large fully connected layer ($32 \times 56 \times 56$ to 128), it has approximately 12.85 million parameters.

Model 2 (Regularized) This deeper network includes three convolutional blocks with 3×3 filters (32, 64, and 128 channels), each followed by Batch Normalization, ReLU, Dropout (0.2-0.3), and 2×2 max pooling (except for the final adaptive average pooling to 7×7). The classifier has one hidden layer (512 units) with 0.2 Dropout, reducing the parameter count to approximately 3.31 million.

Model 3 (Deeper Network) Extending Model 2, this model adds a fourth convolutional block (256 channels) with similar regularization (BatchNorm, Dropout 0.2-0.3) and adaptive average pooling to 7×7. The classifier includes two hidden layers (512 and 256 units) with Dropout (0.4 and 0.3), increasing the parameter count to approximately 6.95 million.

Model 4 (Residual Network) This ResNet-style architecture includes a stem (7×7 convolution with 64 channels, stride 2, followed by 3×3 max pooling) and four residual layers, each with two ResidualBlocks (channel sizes 64, 128, 256, 512). Each ResidualBlock uses 3×3 convolutions with BatchNorm and ReLU, and shortcut connections. The classifier uses global average pooling and a single hidden layer (256 units) with 0.5 Dropout, totaling approximately 11.31 million parameters.

A summary of the key architectural differences is provided in Table 3.1.

Table 3.1: Summary of Custom CNN Model Architectures. 'Final Channels' refers to the number of filters in the last convolutional layer. Note: Dropout in Model 4 is applied only in the classifier.

Characteristic	**Model 1**	**Model 2**	**Model 3**	**Model 4**
Style	Sequential	Regularized	Deeper	ResNet
Extractor Depth	2 Conv Blocks	3 Conv Blocks	4 Conv Blocks	Stem + 4 Res Layers
Final Channels	32	128	256	512
Regularization	None	BN, Dropout	BN, Dropout	BN, Dropout (classifier)
Classifier Depth	1 Hidden Layer	1 Hidden Layer	2 Hidden Layers	1 Hidden Layer
Classifier Units	128	512	512, 256	256
Total Parameters	∼12.85 M	∼3.31 M	∼6.95 M	∼11.31 M

The architecture of CNN Model 1 is visualized in Figure 3.6. The diagram was generated using the NN-SVG tool [1].

[1] Available at: https://alexlenail.me/NN-SVG/

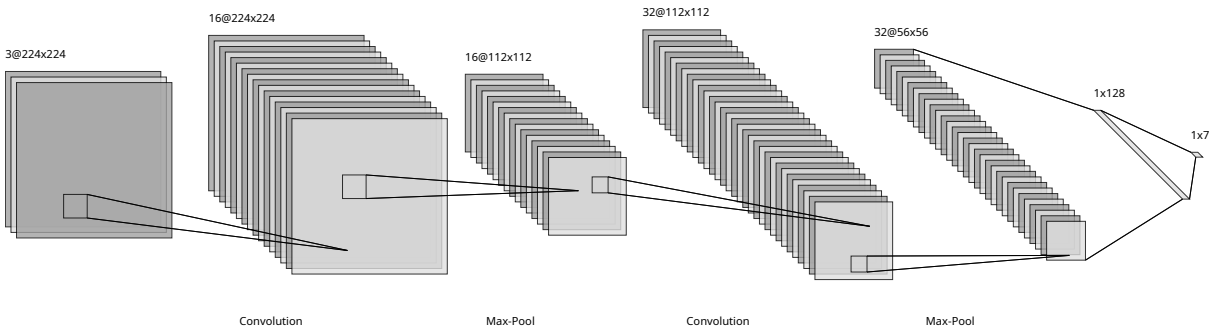


Figure 3.6: The architecture of CNN Model 1, the baseline sequential model.

3.3 Vision Transformer Models

All models are built upon a single, flexible base implementation, allowing for a systematic investigation into the effects of key hyperparameters. To ensure a fair comparison and isolate the impact of these changes, the classifier head is identical across all configurations, consisting of three linear layers (256→512→256→7) with ReLU activations and Dropout (0.4, 0.3). Furthermore, all Transformer Encoder blocks utilize the GELU activation function. The five experimental configurations are referred to as ViT Model 1 through ViT Model 5.

ViT Model 1 (Baseline) This computationally light baseline uses a patch size of 32×32 (50 tokens, including CLS token) and a shallow Transformer Encoder with 3 layers, totaling 3,040,263 parameters.

ViT Model 2 (Smaller Patches) This model uses a smaller patch size of 16×16 (197 tokens), providing a more fine-grained view of the 224×224 input image. It maintains 3 encoder layers, with 2,488,071 parameters.

ViT Model 3 (Deeper Network) Building on Model 2, this configuration increases the Transformer Encoder depth to 5 layers, enhancing model capacity with 3,804,935 parameters.

ViT Model 4 (Wider Network) This model explores the impact of model width. It uses the 3-layer depth of Model 2 but doubles the embedding dimension to 512 and the number of attention heads to 8, significantly increasing the size to 8.77 million parameters.

ViT Model 5 (Advanced Augmentation) This configuration is architecturally identical to ViT Model 2, utilizing a 16×16 patch size and a 3-layer Transformer Encoder. It is distinguished not by its architecture, but by its training methodology, which incorporates the powerful regularization techniques of Mixup and Cutmix augmentation.

A summary of the key specifications for these configurations is provided in Table 3.2.

Table 3.2: Summary of Custom ViT Model Architectures. 'Num Patches' includes the CLS token.

Characteristic	ViT Model 1	ViT Model 2	ViT Model 3	ViT Model 4
Image Size	224×224	224×224	224×224	224×224
Patch Size	32×32	16×16	16×16	16×16
Num Patches	50	197	197	197
Encoder Depth	3 Layers	3 Layers	5 Layers	3 Layers
Embedding Dim.	256	256	256	512
Attention Heads	4	4	4	8
Total Parameters	∼3.04 M	∼2.49 M	∼3.80 M	∼8.77 M

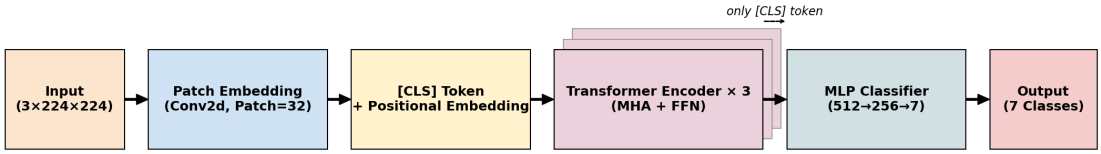


Figure 3.7: The architecture of ViT Model 1, the baseline Transformer model.

3.4 Experimental Setup

This section provides a detailed overview of the experimental environment and the training protocols used to evaluate all custom models. A consistent and well documented setup is essential for ensuring the reproducibility of the results presented in this thesis.

3.4.1 Software and Hardware

All experiments were implemented in Python (version 3.10.18) [26] using the PyTorch (version 2.7.1+rocm6.3) deep learning framework [27]. The experimental workflow relied on several key libraries: Pandas [28] and NumPy [29] were used for data manipulation. Scikit-learn [30] provided utilities for model evaluation and cross-validation. Pillow and OpenCV were used for image processing. Visualization of results was performed using Matplotlib and Seaborn.

The hardware configuration consisted of:

CPU AMD Ryzen 9 7900

GPU AMD RX 6600 (8GB VRAM)

RAM 32 GB DDR5 (6400 MT/s)

Operating System Ubuntu 24.04.2

3.4.2 Training Parameters

A consistent set of training parameters was established to ensure a fair comparison across the different model architectures, with specific adjustments made to accommodate the different model families and training techniques.

For the CNN models, all training runs used the Adam optimizer with a learning rate of 10^{-3} and the Categorical Cross-Entropy loss function. A batch size of 32 was used. To prevent overfitting, an early stopping mechanism was implemented to halt training if the validation loss did not improve by at least 0.001 for 10 consecutive epochs. The baseline Model 1 was trained for a maximum of 100 epochs, while the more complex Models 2, 3, and 4 were trained for up to 200 epochs.

For the ViT models, a more advanced training protocol was employed. All ViT models were trained using the AdamW optimizer, which provides improved weight decay, with a learning rate of 3×10^{-4} and a weight decay value of 0.05. A Cosine Annealing scheduler was used to smoothly decay the learning rate throughout training to a minimum of 10^{-6}. The standard ViT models (1-4) were trained for a maximum of 200 epochs, with an early stopping patience of 10 epochs. The fifth ViT model, which utilized Mixup and Cutmix augmentation, was trained for a longer duration of 300 epochs with an increased patience of 20 epochs to fully leverage this advanced regularization.

Table 3.3: Summary of Training Hyperparameters.

Hyperparameter	**CNN Models**	**ViT Models (Standard)**	**ViT Model (Mixup)**
Optimizer	Adam	AdamW	AdamW
Learning Rate	10^{-3}	3×10^{-4}	3×10^{-4}
Weight Decay	N/A	0.05	0.05
LR Scheduler	None	Cosine Annealing	Cosine Annealing
Loss Function	Cross-Entropy	Cross-Entropy	Cross-Entropy
Batch Size	32	32	32
Max Epochs	100-200	200	300
Early Stopping	Patience=10	Patience=10	Patience=20

3.5 Evaluation Metrics

To provide a comprehensive comparison of the custom model architectures, a combination of quantitative performance metrics and qualitative visualization techniques was used. The quantitative metrics measure the predictive performance and computational efficiency of each model, while the qualitative methods offer insights into their internal decision-making processes.

3.5.1 Quantitative Performance Metrics

The primary evaluation of the models was based on a set of key performance indicators, which were tracked during the 10-fold cross-validation process. The final reported score for each metric is the average over the 10 folds.

3.5.1.1 Classification Performance Metrics

Several metrics were used to assess classification performance. While Accuracy served as the primary metric for model saving during training, a more detailed evaluation was conducted using metrics derived from the confusion matrix. Given the dataset's class imbalance, the Macro F1 Score was a key indicator, as it provides a balanced measure between Precision and Recall. These core metrics are defined as:

$$\text{Precision} = \frac{T_p}{T_p + F_p} \tag{3.1}$$

$$\text{Recall (Sensitivity)} = \frac{T_p}{T_p + F_n} \tag{3.2}$$

$$\text{Specificity} = \frac{T_n}{T_n + F_p} \tag{3.3}$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.4}$$

Here, T_p, F_p, T_n, and F_n represent True Positives, False Positives, True Negatives, and False Negatives, respectively.

Precision Precision quantifies the accuracy of positive predictions. It is defined as the ratio of correctly identified positive instances (True Positives) to the total number of instances predicted as positive (the sum of True Positives and False Positives). A high precision score indicates a low false positive rate, signifying that the model is reliable in its positive classifications.

Recall (Sensitivity) Recall, also known as Sensitivity or the True Positive Rate, measures the model's ability to identify all actual positive instances within the dataset. It is the ratio of correctly identified positive instances to the total number of actual positive instances (the sum of True Positives and False Negatives). High recall is critical in applications where failing to detect a positive case has significant consequences.

Specificity Specificity, or the True Negative Rate, measures the model's ability to correctly identify all actual negative instances. It is calculated as the ratio of correctly identified negative instances (True Negatives) to the total number of actual negative instances (the sum of True Negatives and False Positives). A high specificity indicates that the model is effective at avoiding the misclassification of negative samples.

F1 Score The F1 Score is the harmonic mean of Precision and Recall, providing a single metric that balances both measures. Unlike the arithmetic mean, the harmonic mean penalizes extreme values, ensuring that both precision and recall must be reasonably high for the F1 Score to be high. This makes it particularly useful for imbalanced datasets where one metric alone may give a misleading impression of performance.

The macro average computes the F1 score for each class and then finds their unweighted mean, treating all classes equally regardless of their size in the dataset.

3.5.1.2 Validation Loss

The Categorical Cross-Entropy loss on the validation set was monitored during training. This metric was used by the early stopping mechanism to determine when a model had converged.

3.5.1.3 Training Time & Epoch Count

The training time and the number of epochs to converge were recorded for each fold as practical measures of computational efficiency.

3.5.2 Model Interpretability and Visualization

Beyond quantitative metrics, several visualization techniques were employed to better understand the features learned by the models and the reasoning behind their predictions.

3.5.2.1 Feature and Filter Visualization

For the CNN models, feature maps from convolutional layers were visualized to observe the patterns (e.g., edges, textures) that the network had learned to detect at different depths. For the ViT models, the learned representations within the initial patch embedding layer were visualized.

3.5.2.2 Attention Map Visualization

For the ViT models, attention maps from the self-attention layers were visualized to illustrate how the special classification token ([CLS]) attended to each image patch. Since the [CLS] token's final representation is used for classification, this visualization revealed which regions of the image were most influential in the model's final decision.

3.5.2.3 t-SNE Visualization

t-Distributed Stochastic Neighbor Embedding (t-SNE) was applied to visualize the high dimensional feature space learned by the models. Feature vectors for the test images were projected into a 2D space to examine whether the models had learned to group images of the same class into distinct clusters [31].

3.5.2.4 Precision-Recall (PR) Curves

PR curves were generated to illustrate the trade-off between precision and recall for each class across all decision thresholds. This visualization is particularly valuable for imbalanced datasets, as it reveals a model's performance on minority classes (e.g., Melanoma) more effectively than aggregate metrics such as accuracy. The Area Under the PR Curve (AUC-PR) was computed to provide a single score summarizing this performance.

3.5.2.5 Grad-CAM Heatmaps

Gradient-weighted Class Activation Mapping (Grad-CAM) was used to produce heatmaps highlighting the most influential regions in an input image for a given prediction. This technique helped verify that the models were focusing on relevant pathological areas of skin lesions rather than on confounding artifacts [32].

Chapter 4

Results

This chapter presents the empirical results of the experiments detailed in the Methodology. The performance of the custom designed CNN and ViT models is evaluated and compared using the metrics defined in the previous chapter.

4.1 Performance of CNN Models

This section presents the performance of the four custom CNN models evaluated on the HAM10000 dataset using 10-fold cross-validation. The primary performance indicators are summarized in Table 4.1, which reports the average metrics calculated from the best-performing epoch of each fold. The learning curves, in contrast, show the performance averaged at each specific epoch across all folds. It is important to note that this difference in aggregation can lead to apparent discrepancies between the final reported scores in the tables and the visual trends in the plots. Table 4.2 details the training characteristics of each model.

Table 4.1: Summary of Performance Metrics for Custom CNN Models. Values are reported as mean ± standard deviation across 10 folds.

Model	Accuracy (%)	Macro F1 (%)	Precision (%)	Recall (%)	Specificity (%)
Model 1	76.09 ± 0.7	49.74 ± 2.2	54.12 ± 3.6	47.70 ± 2.2	92.67 ± 0.5
Model 2	78.14 ± 1.2	51.93 ± 3.7	59.52 ± 4.3	48.87 ± 4.3	93.20 ± 0.6
Model 3	78.69 ± 0.7	52.20 ± 3.1	60.12 ± 6.4	49.09 ± 2.7	93.46 ± 0.4
Model 4	79.25 ± 0.7	56.61 ± 3.5	62.69 ± 5.9	54.92 ± 3.8	94.13 ± 0.4

Table 4.2: Summary of Training Characteristics for Custom CNN Models.

Model	Avg. Fold Time (s)	Avg. Epochs
Model 1	410 ± 88	23.5 ± 5.7
Model 2	1748 ± 475	60.6 ± 16.6
Model 3	2753 ± 685	88.2 ± 23.3
Model 4	1427 ± 207	50.5 ± 7.4

The baseline **Model 1** achieved an accuracy of 76.09%, but its modest macro F1-score of 49.74% was constrained by a low recall of 47.70%, indicating difficulty in correctly

identifying true positives across all classes. Its learning curves are presented in Figure 4.1.

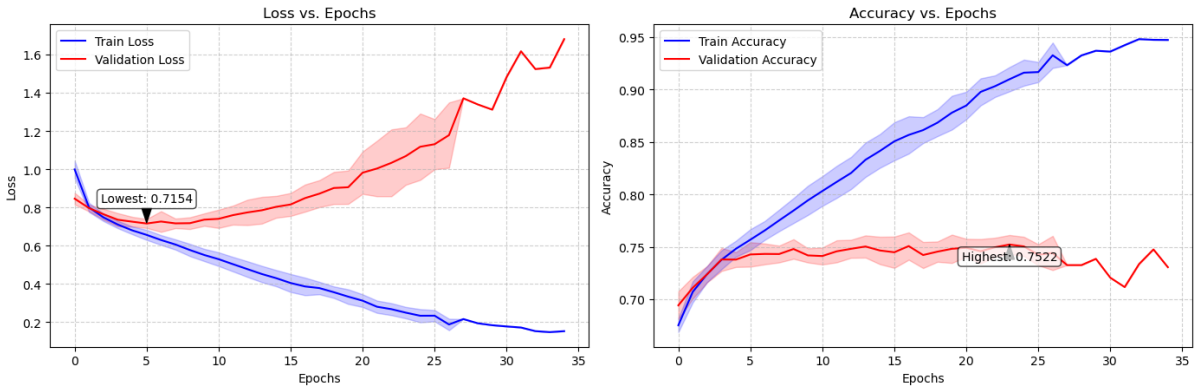


Figure 4.1: Average learning curves for CNN Model 1 across 10 folds. The shaded area represents the standard deviation between folds.

Model 2, which introduced regularization, showed a clear improvement in accuracy (78.14%) and precision (59.52%). However, as shown in its learning curves (Figure 4.2), this came at the cost of longer training times. Its recall remained low at 48.87%, explaining the only marginal increase in the F1-score to 51.93%. This suggests that while its positive predictions were more likely to be correct, it still failed to identify many true cases.

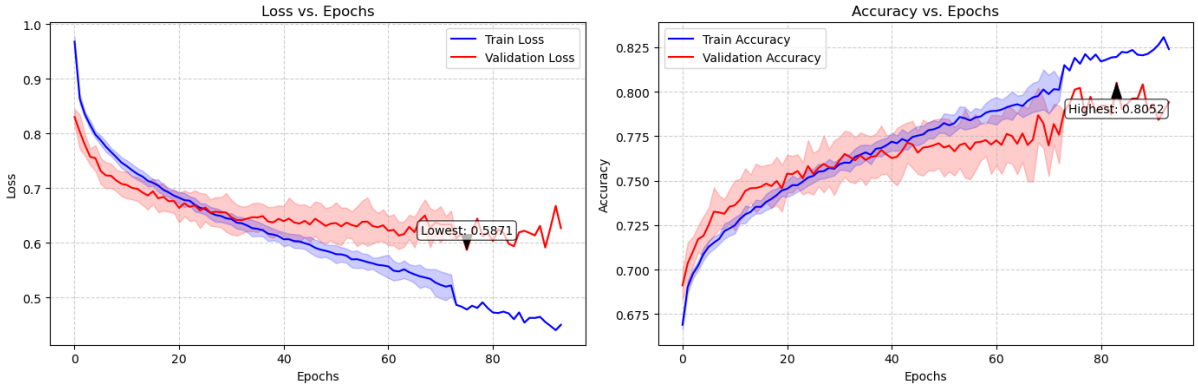


Figure 4.2: Average learning curves for CNN Model 2 across 10 folds. The shaded area represents the standard deviation between folds.

Model 3 further increased network depth, achieving an accuracy of 78.69% and a precision of 60.12%. This marginal gain over Model 2 required the longest training time, as seen in Figure 4.3, and showed almost no improvement in recall (49.09%), yielding a nearly identical F1-score of 52.20%.

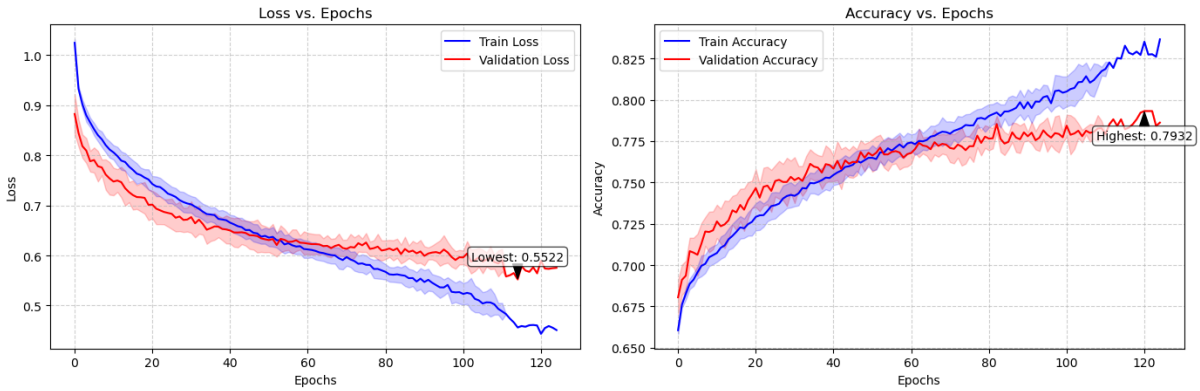


Figure 4.3: Average learning curves for CNN Model 3 across 10 folds. The shaded area represents the standard deviation between folds.

The ResNet-style architecture of **Model 4** yielded the best performance across all metrics. It achieved the highest accuracy (79.25%) and demonstrated a substantial leap in diagnostic capability. Both its precision (62.69%) and, most critically, its recall (54.92%) were the highest of all models. This marked improvement in recall, the model's ability to find all true positives, is the primary driver for its superior F1-score of 56.61% and indicates it was far more effective at learning the features of minority classes.

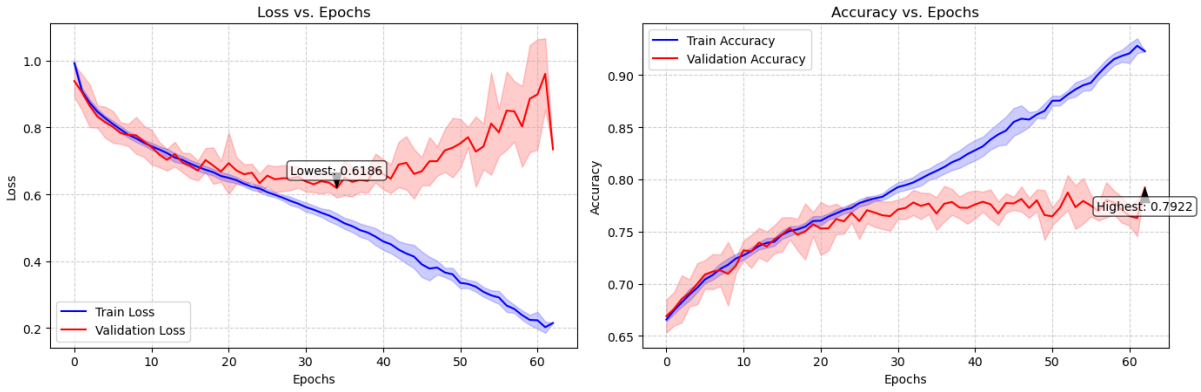


Figure 4.4: Average learning curves for CNN Model 4 across 10 folds. The shaded area represents the standard deviation between folds.

Across all models, the **specificity** remained consistently high (92−94%), showing that all architectures were proficient at correctly identifying true negatives. This is a valuable trait, as it minimizes false alarms for healthy patients. However, the progression from Model 1 to Model 4 clearly shows that the ResNet architecture was necessary to achieve a better balance between precision and recall, leading to a more clinically useful model. This superior performance is visualized in the comparative learning curves (Figure 4.5) and the cleaner class separation in its confusion matrix (Figure 4.6d).

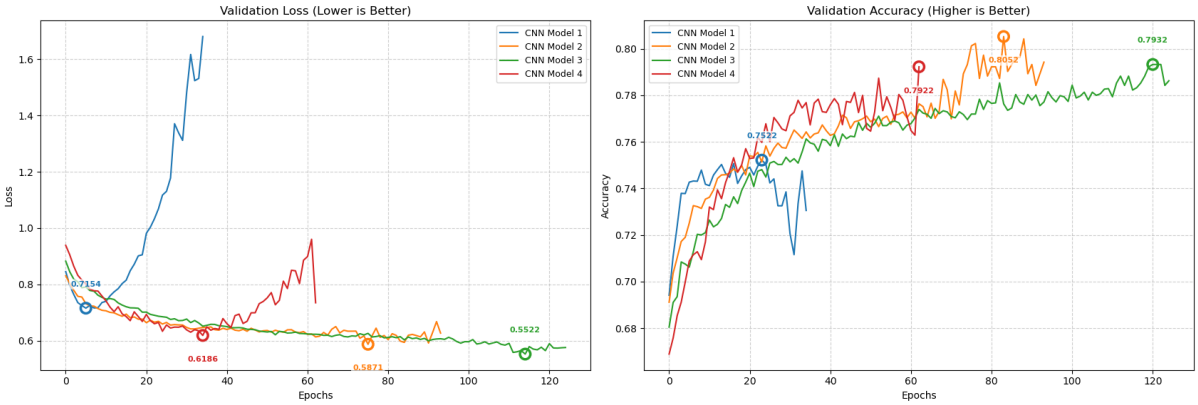
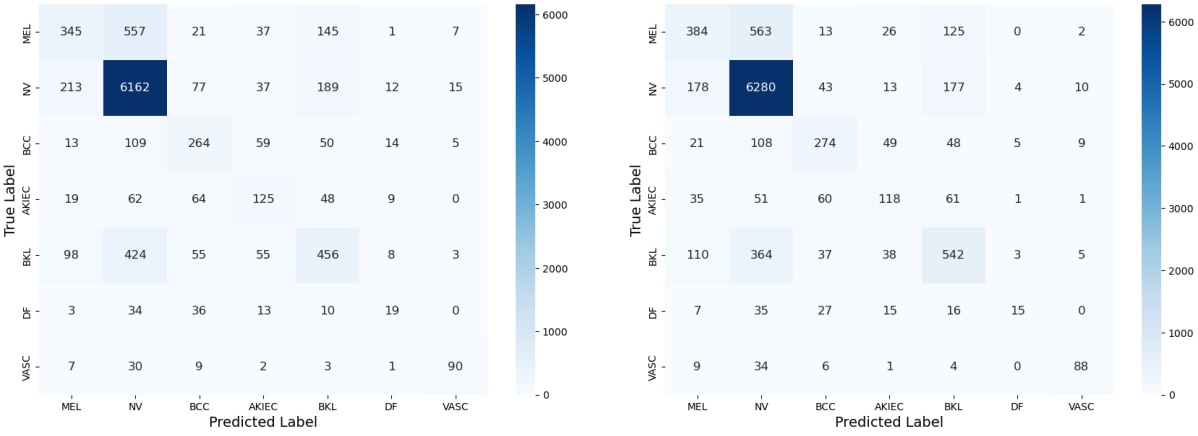
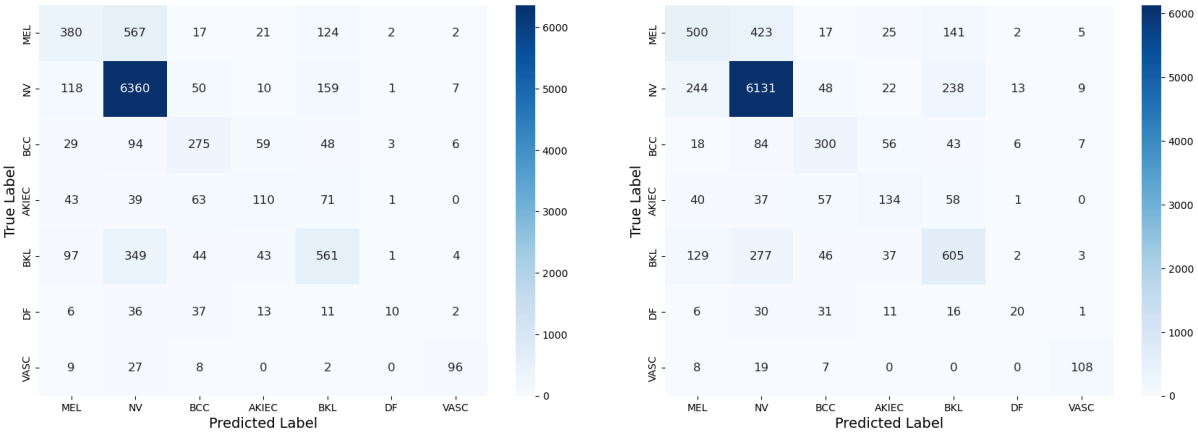


Figure 4.5: Comparison of average validation learning curves for all four custom CNN models across 10 folds. The left plot shows loss versus epochs, and the right plot shows accuracy.



(a) CNN Model 1 (Baseline)

(b) CNN Model 2 (Regularized)



(c) CNN Model 3 (Deeper)

(d) CNN Model 4 (ResNet)

Figure 4.6: Normalized confusion matrices for the four custom CNN models.

4.1.1 Analysis of the Best Performing CNN

To better understand the behavior of the best performing CNN architecture, Model 4, several interpretability analyses were conducted. This deep dive uses t-SNE to visualize

the learned feature space, a PR curve to assess performance on all classes, and Grad-CAM with error analysis to provide insight into the model's decision making process.

First, a t-SNE plot was generated to visualize the feature space learned by the model's classifier head, as shown in Figure 4.7. The plot shows a clear tendency for the model to group images of the same class into distinct clusters, indicating that it has learned a meaningful feature representation for separating the different lesion types.

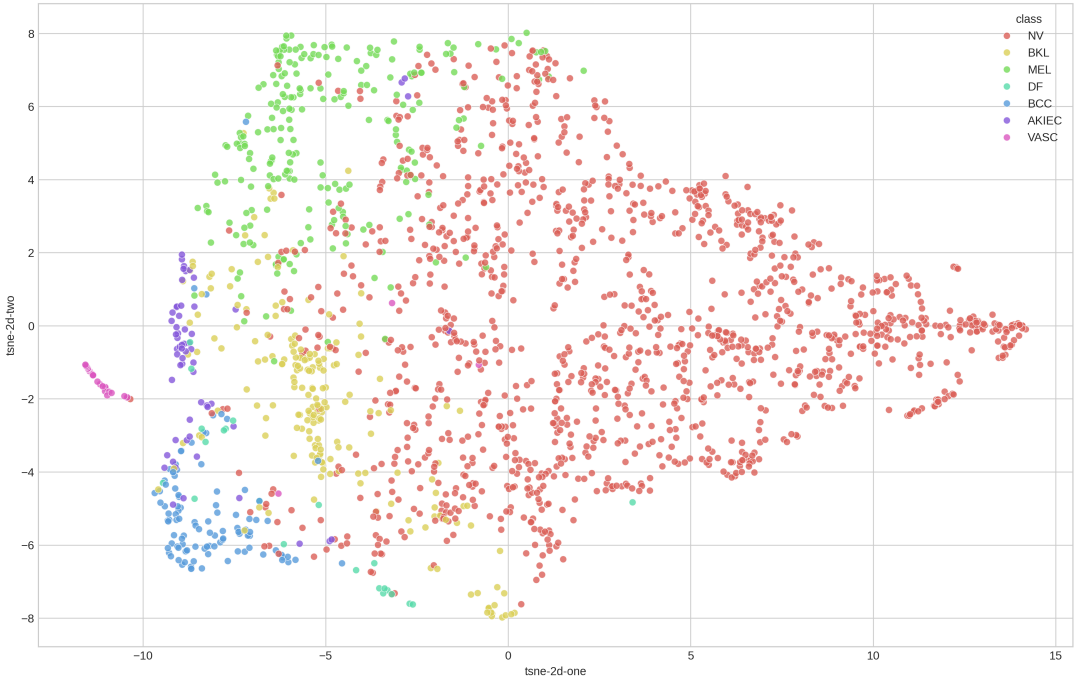


Figure 4.7: t-SNE visualization of the feature space learned by CNN Model 4 on the test set. Each point represents an image, colored by its true class.

To analyze the model's performance on each distinct class, a comprehensive PR curve was generated, as shown in Figure 4.8. While the micro-average Area Under the Curve (AUC-PR) is a high 0.871, this metric is heavily skewed by the model's excellent performance on the majority class, Melanocytic nevi (NV), which achieved a near-perfect AUC-PR of 0.962. A more nuanced picture emerges from the individual classes. The model shows strong performance on Vascular lesions (VASC) with an AUC-PR of 0.811, but struggles significantly with the critical Melanoma (MEL) class (0.516). Performance is lowest on Dermatofibroma (DF), with an AUC-PR of just 0.266, indicating that this class is frequently misidentified. This detailed view confirms the class-specific challenges hidden by overall accuracy metrics.

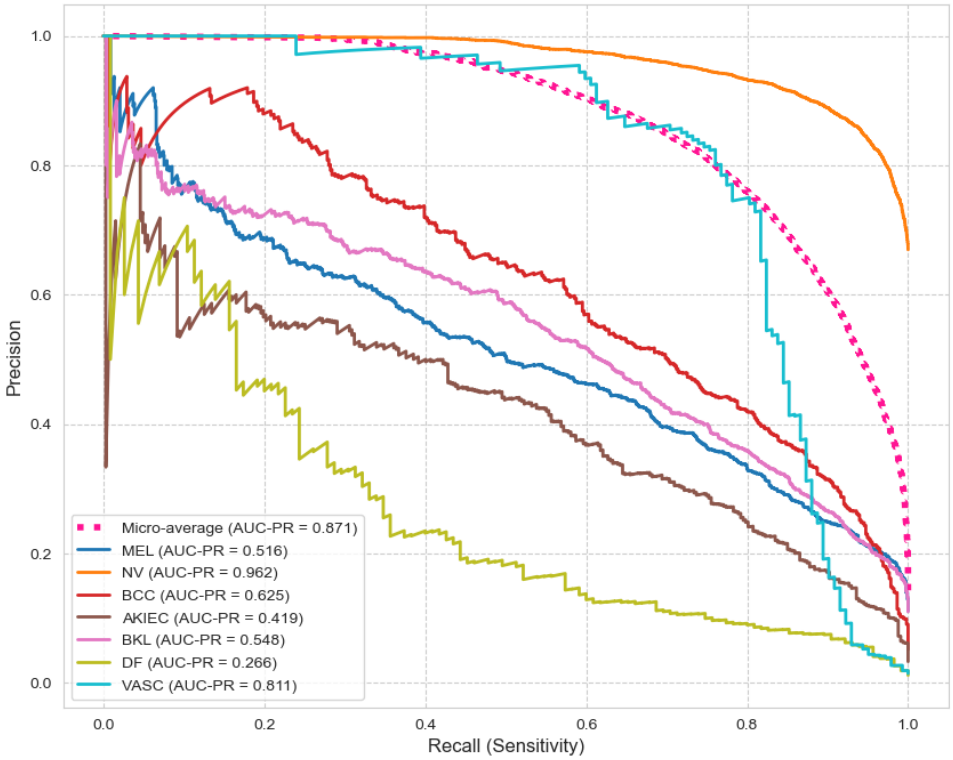


Figure 4.8: Aggregated PR curves for each class for CNN Model 4.

Next, Grad-CAM was used to visualize the model's focus on individual predictions. Figure 4.9 shows an example of a correct prediction, where the model accurately focuses on the lesion itself. In contrast, Figure 4.10 reveals how the model's attention can be misplaced for an incorrect prediction, offering clues into its specific failure modes.

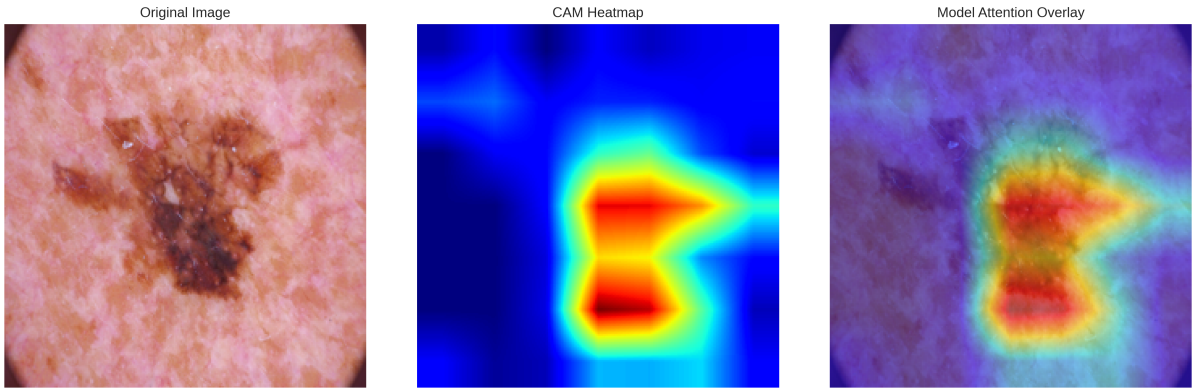


Figure 4.9: Grad-CAM heatmap illustrating a correct prediction by CNN Model 4.

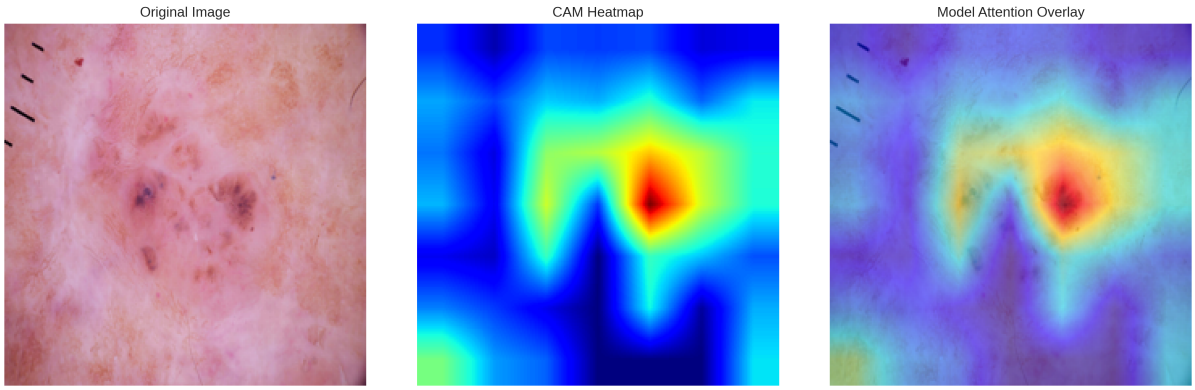


Figure 4.10: Grad-CAM heatmap illustrating an incorrect prediction by CNN Model 4.

To broaden this error analysis, the images with the highest classification loss across all validation folds were identified (Figure 4.11). A recurring theme in these difficult cases is the misclassification of various lesion types as Melanocytic nevi (NV), the dataset's majority class. This provides a qualitative illustration of the model's systematic bias.

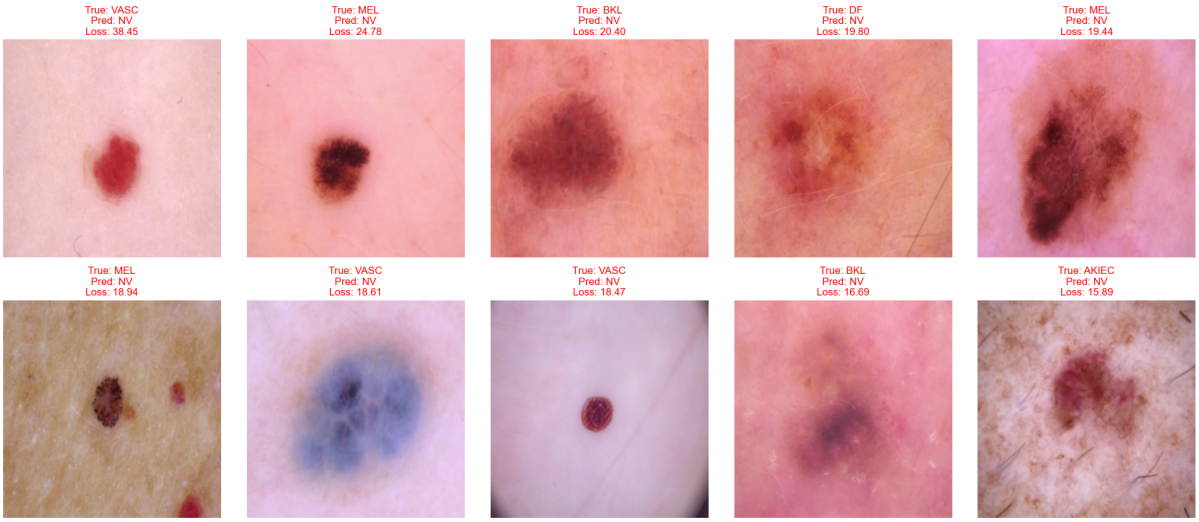


Figure 4.11: Top 10 most difficult images for CNN Model 4, aggregated across all folds and sorted by highest loss.

Finally, to inspect the patterns learned inside the network, the feature maps from the model's convolutional layers were visualized. Figure 4.12 shows that an early layer captures simple features, while Figure 4.13 shows that a deeper layer detects more abstract representations.

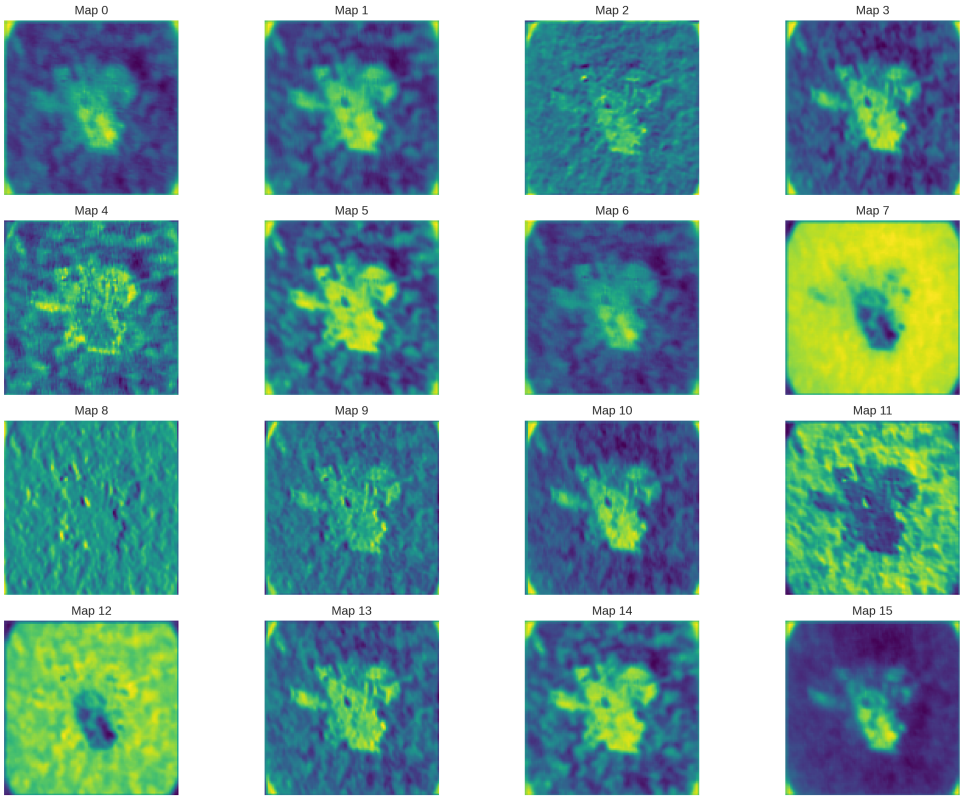


Figure 4.12: Feature map visualizations from an early convolutional layer of CNN Model 4.

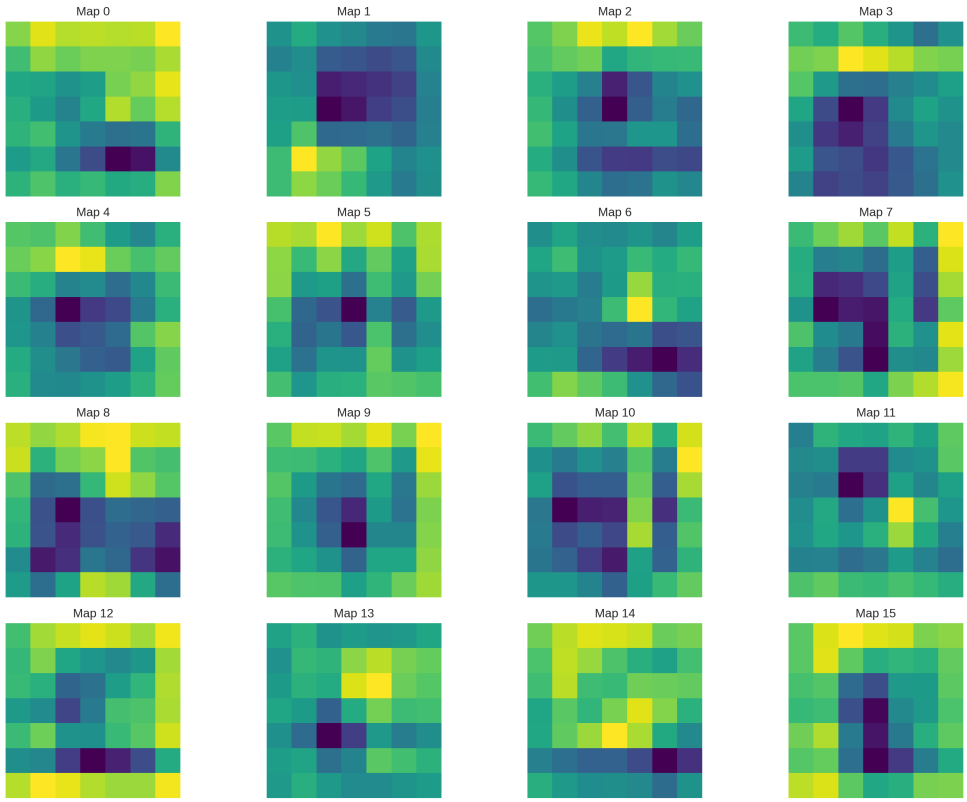


Figure 4.13: Feature map visualizations from a deep convolutional layer of CNN Model 4.

4.2 Performance of ViT Models

This section details the performance of the five custom ViT model configurations evaluated using 10-fold cross-validation. A comprehensive summary of the performance metrics is provided in Table 4.3, which reports the average metrics calculated from the best-performing epoch of each fold. The learning curves, in contrast, show the performance averaged at each specific epoch across all folds. It is important to note that this difference in aggregation can lead to apparent discrepancies between the final reported scores in the tables and the visual trends in the plots. Table 4.4 details the training characteristics of each model.

Table 4.3: Summary of Performance Metrics for Custom ViT Models. Values are reported as mean ± standard deviation across 10 folds.

Model	Accuracy (%)	Macro F1 (%)	Precision (%)	Recall (%)	Specificity (%)
Model 1	75.65 ± 1.10	47.36 ± 3.4	53.76 ± 6.4	45.47 ± 3.8	92.72 ± 0.5
Model 2	78.24 ± 1.13	51.90 ± 4.2	56.64 ± 4.0	50.68 ± 4.4	93.66 ± 0.8
Model 3	78.15 ± 1.42	53.87 ± 5.8	58.40 ± 7.4	52.12 ± 5.3	93.79 ± 0.7
Model 4	73.00 ± 1.07	33.92 ± 3.6	45.24 ± 7.0	31.59 ± 3.0	89.82 ± 0.8
Model 5	80.11 ± 1.70	57.47 ± 6.3	62.68 ± 6.9	55.45 ± 5.6	94.01 ± 0.7

Table 4.4: Summary of Training Characteristics for Custom ViT Models.

Model	Avg. Fold Time (s)	Avg. Epochs
Model 1	412 ± 130	44.7 ± 14.8
Model 2	742 ± 313	43.2 ± 8.3
Model 3	1256 ± 869	49.1 ± 25.8
Model 4	1096 ± 293	31.1 ± 7.8
Model 5	1816 ± 700	84.5 ± 27.6

The baseline **ViT Model 1**, using a large 32×32 patch size, achieved an accuracy of 75.65%. Its diagnostic capability was limited, reflected by a macro recall of only 45.47%, resulting in a modest F1-score of 47.36%.

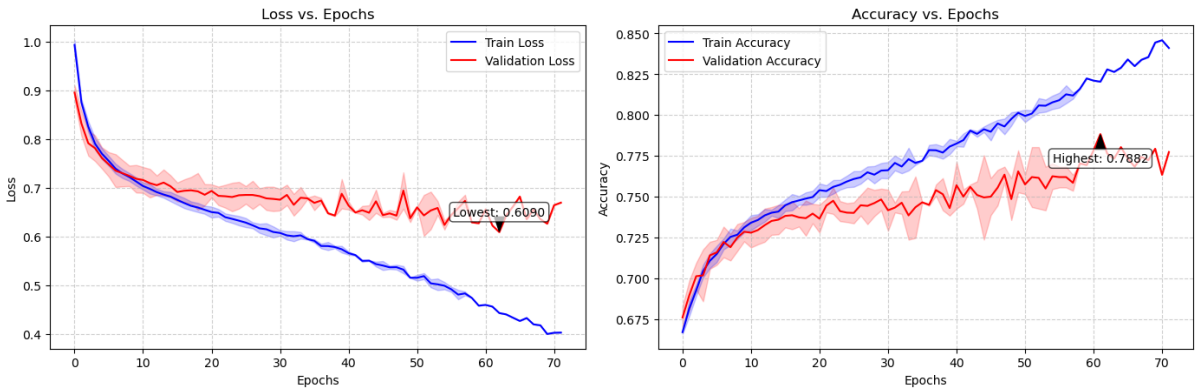


Figure 4.14: Average learning curves for ViT Model 1 across 10 folds. The shaded area represents the standard deviation between folds.

By reducing the patch size to 16×16, **ViT Model 2** saw a significant performance increase. Accuracy rose to 78.24%, and more importantly, recall jumped to 50.68%. This improved ability to identify true positives directly translated to a much healthier F1-score of 51.90%, highlighting the benefit of a more fine-grained input sequence for feature extraction.

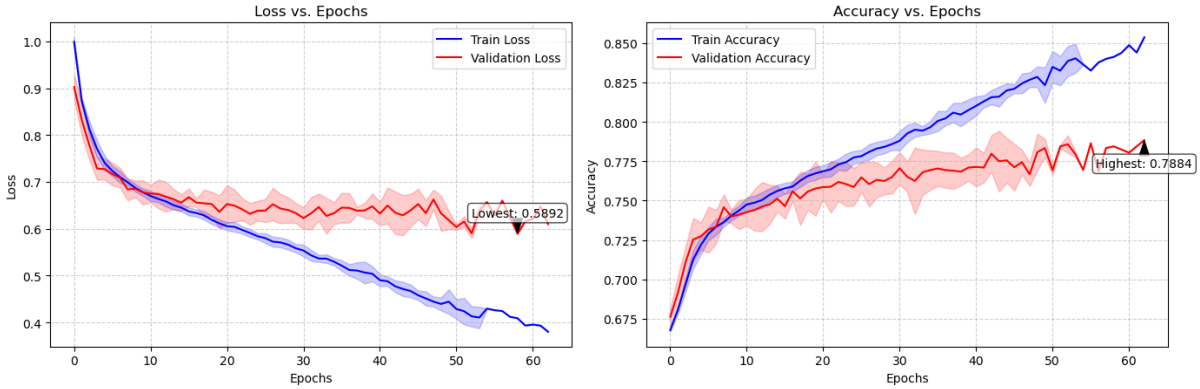


Figure 4.15: Average learning curves for ViT Model 2 across 10 folds. The shaded area represents the standard deviation between folds.

ViT Model 3 explored increasing network depth. While accuracy remained stable at 78.15%, this model achieved the second highest recall (52.12%) and precision (58.40%) of the architectural variants, leading to a further improved F1-score of 53.87%. This suggests that a deeper network was better able to model the relationships between the larger number of patches.

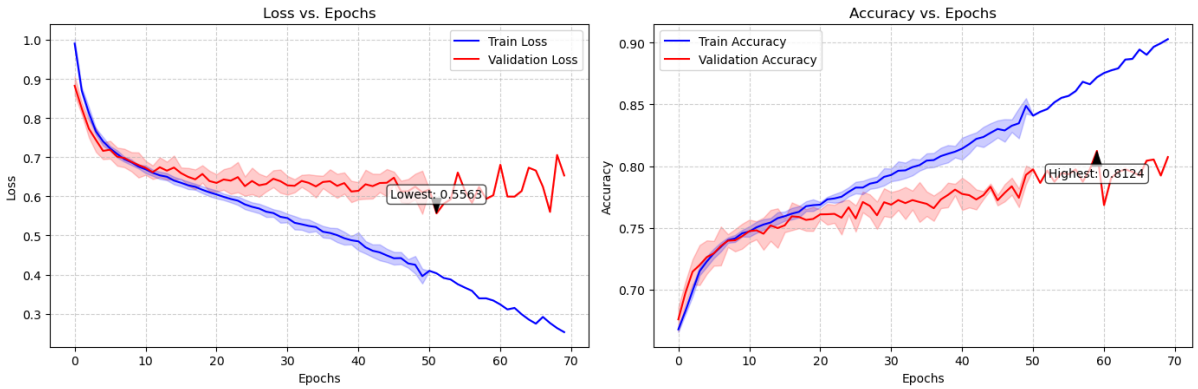


Figure 4.16: Average learning curves for ViT Model 3 across 10 folds. The shaded area represents the standard deviation between folds.

In contrast, **ViT Model 4**, the wider architecture, suffered a catastrophic drop in performance. Its accuracy fell to 73.00%, and its recall collapsed to 31.59%, meaning it failed to identify more than two thirds of the true positives in the minority classes. This led to the lowest F1-score of all models (33.92%) and a notable drop in specificity to 89.82%, indicating that naively increasing the embedding dimension without sufficient data or regularization was highly detrimental.

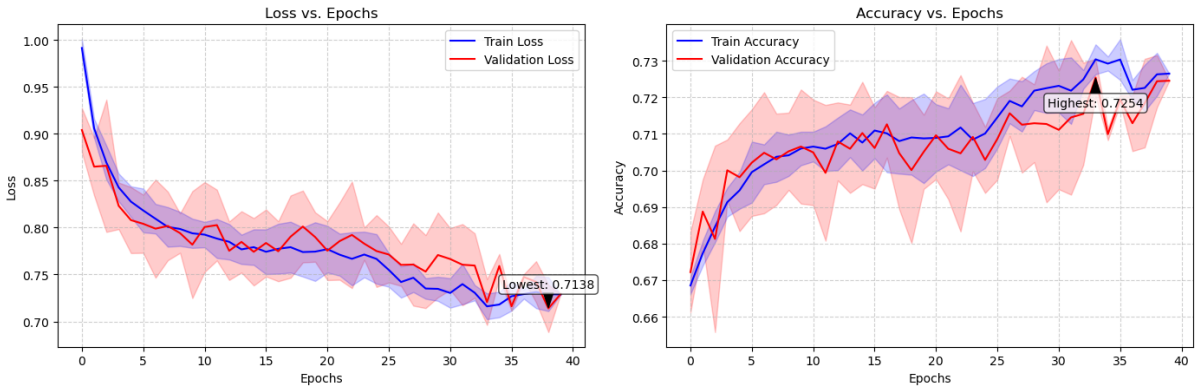


Figure 4.17: Average learning curves for ViT Model 4 across 10 folds. The shaded area represents the standard deviation between folds.

The best performing ViT was **ViT Model 5**, which used the same architecture as Model 2 but with Mixup and Cutmix augmentation. It achieved the highest scores across the board: accuracy (80.11%), precision (62.68%), recall (55.45%), specificity (94.01%), and F1-score (57.47%). This demonstrates conclusively that advanced data augmentation was the single most impactful strategy for improving ViT performance in this study, enabling the model to achieve a superior balance of identifying true positives while minimizing false alarms.

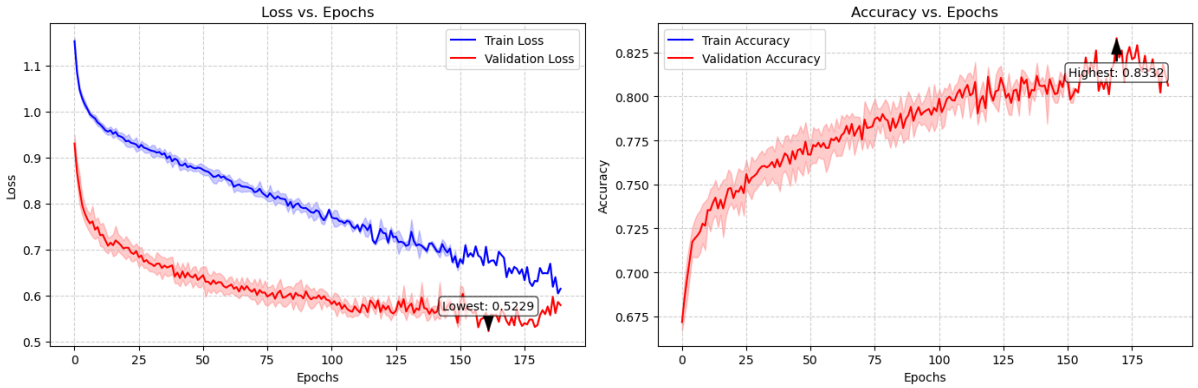


Figure 4.18: Average learning curves for ViT Model 5 across 10 folds. The shaded area represents the standard deviation between folds. Note: The training accuracy curve is intentionally omitted. Because this model was trained with Mixup and Cutmix, which create "soft" labels by combining two images, a standard accuracy metric is not meaningful during the training phase.

Again we see that specificity remained consistently high ($92-94\%$), with the exception of model 4. This shows that all models were proficient at identifying correctly true negatives. The progression from Model 1 to 5 highlights two critical factors for ViT success on this task, the use of 16×16 patch to capture more detailed features, and the application of strong data augmentation techniques (Cutmix, Mixup). A direct visual comparison of the learning dynamics for all five ViT configurations is shown in Figure 4.19, and a detailed breakdown of the classification performance for each model is provided by their respective confusion matrices in Figure 4.20.

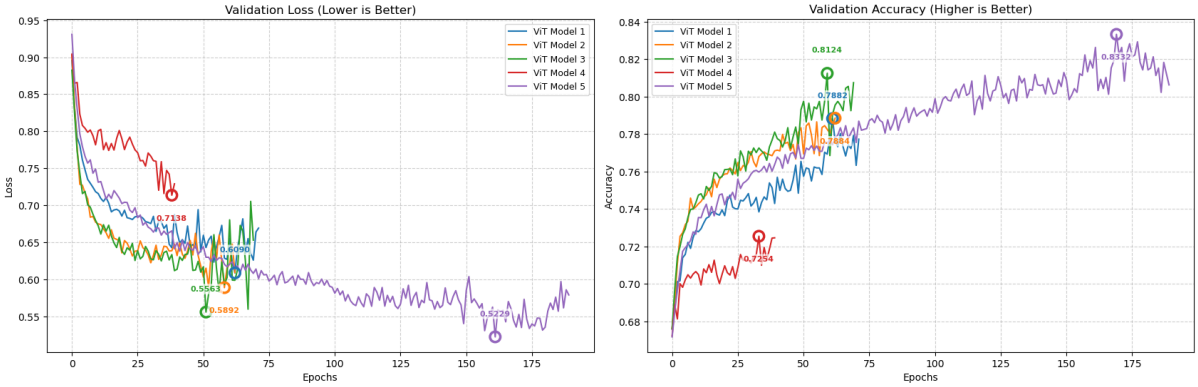
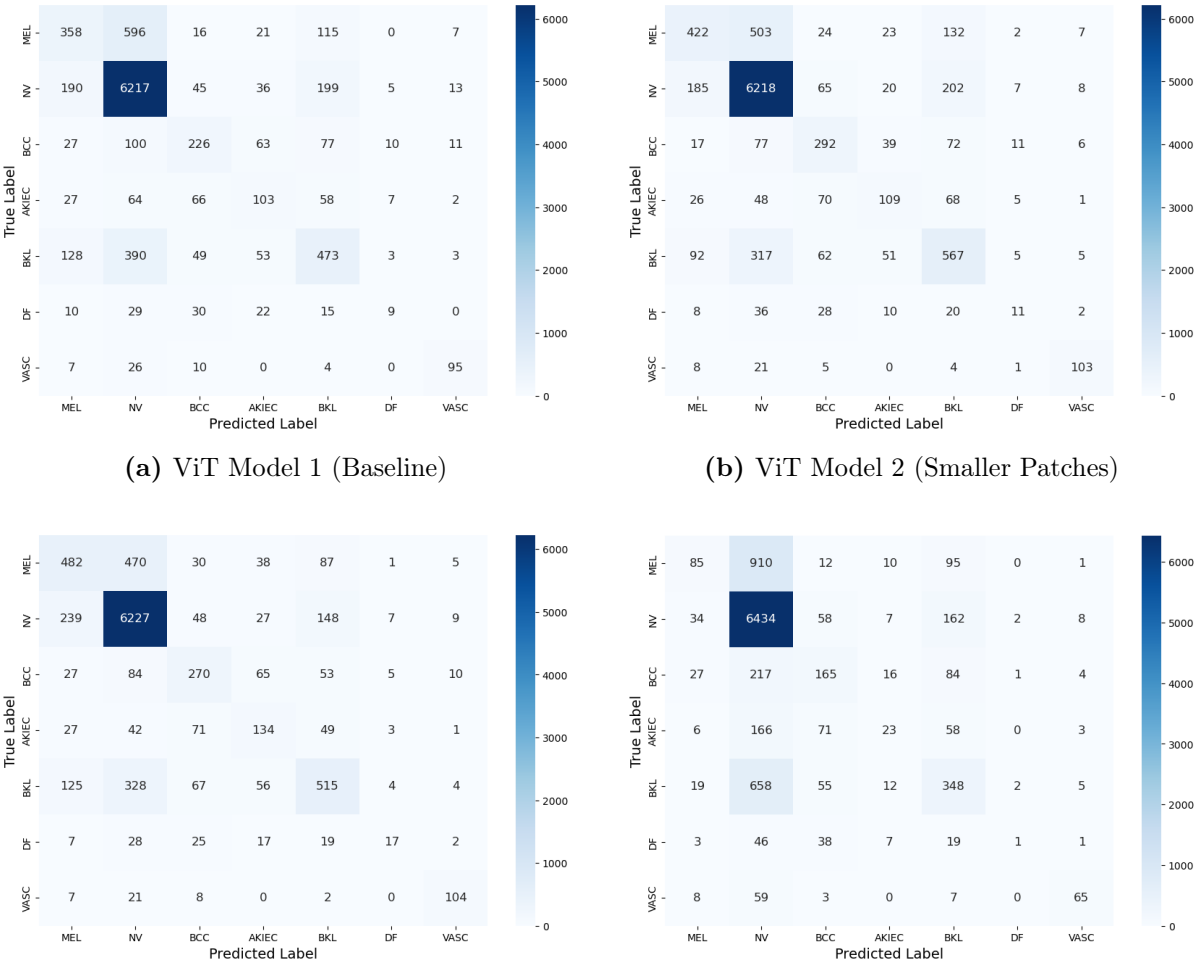


Figure 4.19: Comparison of average validation learning curves for all five custom ViT models across 10 folds.



(a) ViT Model 1 (Baseline)

(b) ViT Model 2 (Smaller Patches)

(c) ViT Model 3 (Deeper)

(d) ViT Model 4 (Wider)

Figure 4.20: Normalized confusion matrices for the four ViT architectural variants (Models 1–4).

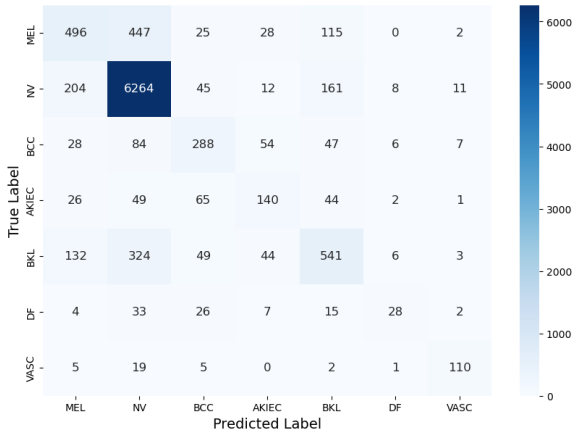


Figure 4.21: Normalized confusion matrix for the best performing ViT Model 5, which used the architecture of Model 2 with Mixup and Cutmix augmentation.

4.2.1 Analysis of the Best Performing ViT

To gain a deeper understanding of the behavior of the best performing ViT architecture, Model 5, a series of interpretability analyses were conducted. This analysis uses t-SNE, per-class PR curves, Grad-CAM, error analysis, and visualizations of the model's internal mechanisms.

First, a t-SNE plot was generated to visualize the feature space from the model's [CLS] token, as shown in Figure 4.22. The plot shows that the model learns to form reasonably distinct clusters for the different lesion types, though with some overlap, indicating a well-learned but challenging feature space.

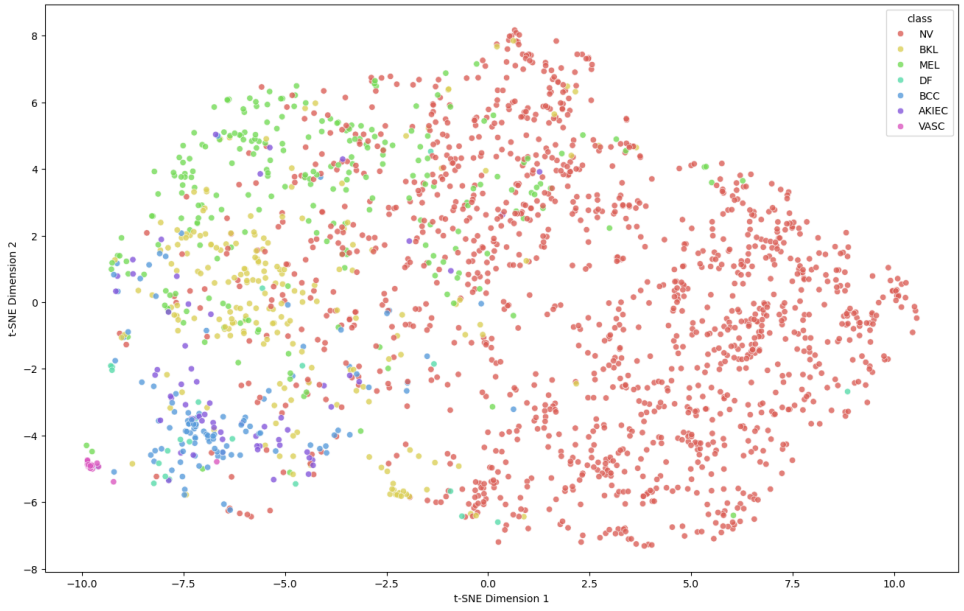


Figure 4.22: t-SNE visualization of the feature space learned by ViT Model 5 on the test set.

A detailed, per-class analysis of the model's performance is provided by the PR curves in Figure 4.23. The model achieves a high micro-average AUC-PR of 0.880, largely

driven by its excellent performance on the majority Melanocytic nevi (NV) class (AUC-PR = 0.966). Performance on other classes varies significantly. The model is strong at identifying Vascular lesions (VASC) with an AUC-PR of 0.822, but finds Melanoma (MEL) more challenging (AUC-PR = 0.537). Similar to the best CNN, its weakest performance is on Dermatofibroma (DF), with an AUC-PR of just 0.368.

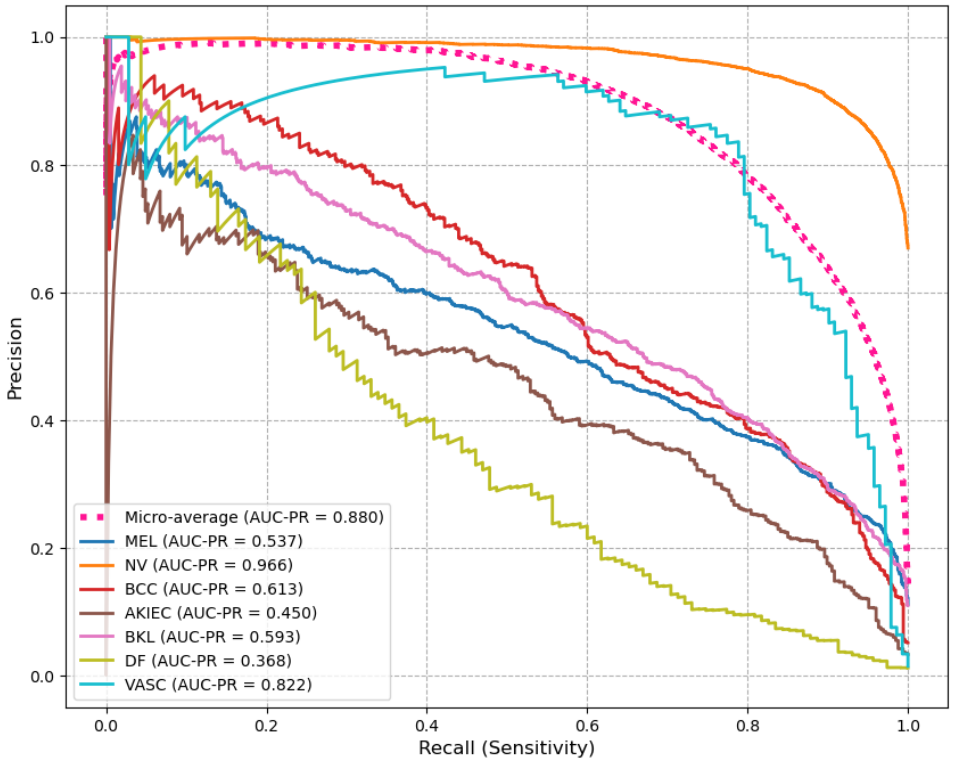


Figure 4.23: Aggregated PR curves for each class for ViT Model 5.

Next, Grad-CAM heatmaps were produced to highlight which image patches were most influential. Figure 4.24 shows that for a correct prediction, attention is appropriately focused. Figure 4.25 shows a failure case where a melanoma (MEL) was misclassified, with the model's attention being more diffuse and around the lesion.

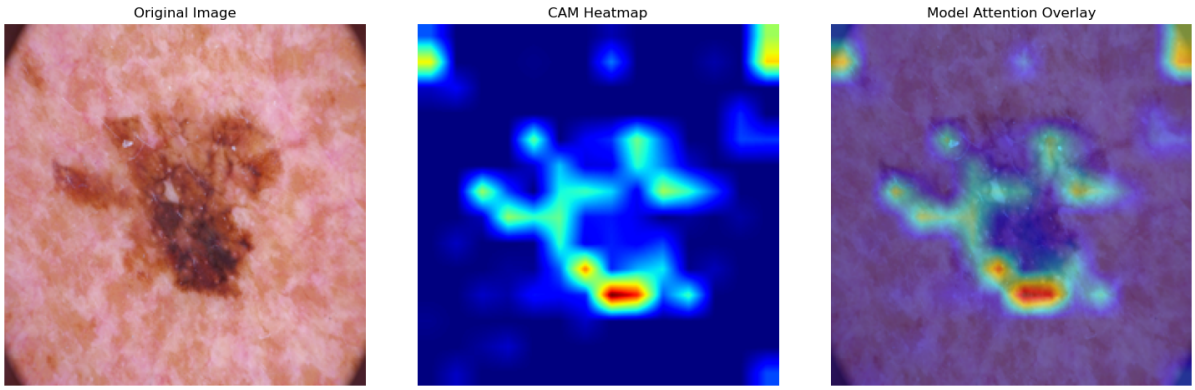


Figure 4.24: Grad-CAM heatmap for ViT Model 5, showing a correct prediction.

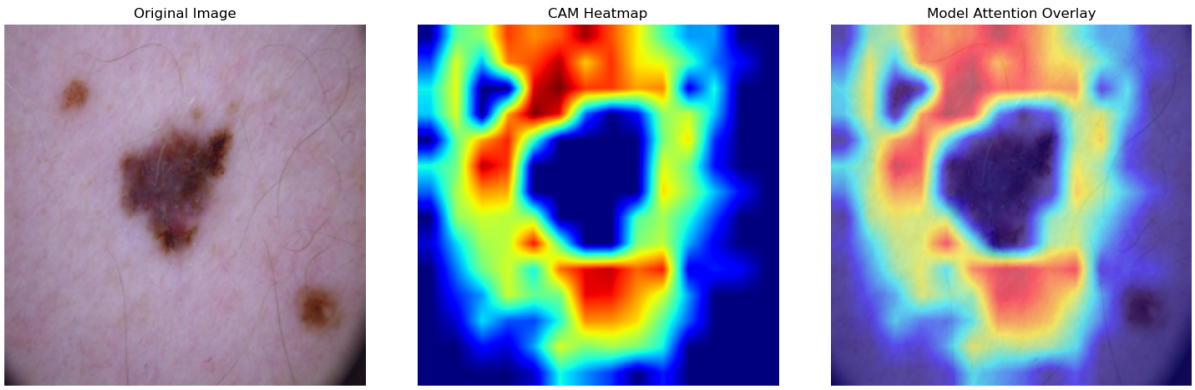


Figure 4.25: Grad-CAM heatmap for ViT Model 5, showing an incorrect prediction.

An analysis of the images with the highest classification loss (Figure 4.26) reveals the model's specific failure modes. Unlike the best CNN which primarily defaulted to the majority class, the ViT model exhibits a more varied pattern of errors, for example, misclassifying a Benign keratosis-like lesion (BKL) as a Vascular lesion (VASC). This suggests that while still imperfect, its learned representations may be more nuanced.

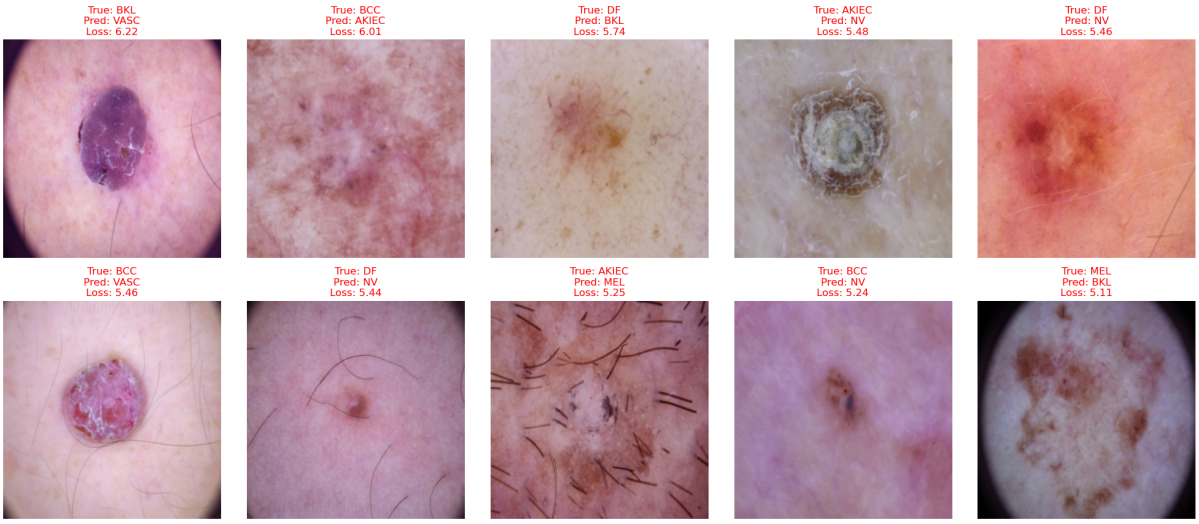


Figure 4.26: Top 10 most difficult images for ViT Model 5, aggregated across all folds and sorted by highest loss.

Finally, to look inside the ViT architecture, its internal mechanisms were visualized. The patch embedding filters (Figure 4.27) show the fundamental low-level patterns the model learns. The self-attention maps (Figures 4.28–4.29) show how the model weighs the importance of different image patches, with attention becoming more focused in deeper layers.

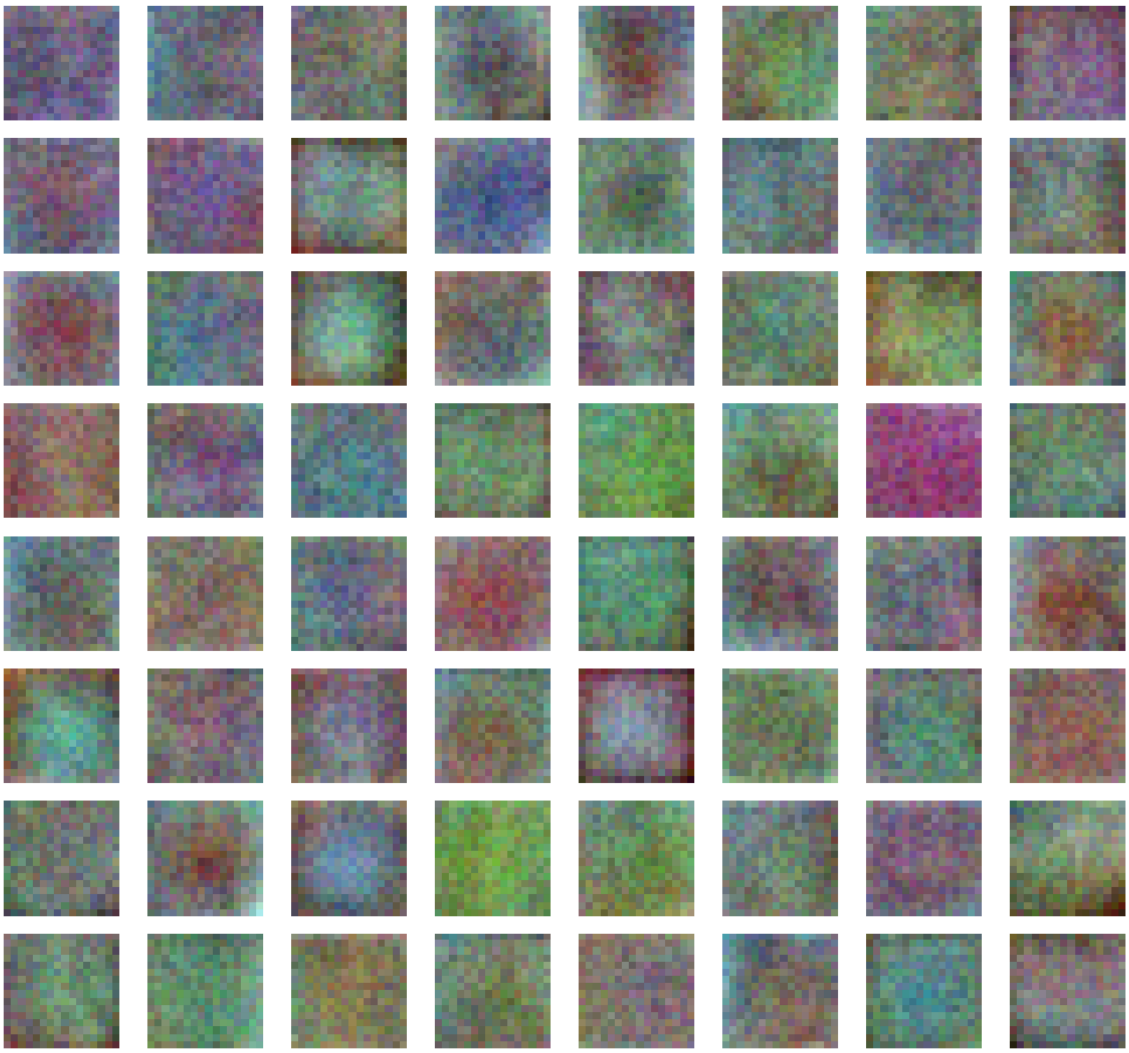


Figure 4.27: Visualization of the first-layer patch embedding filters learned by ViT Model 5.

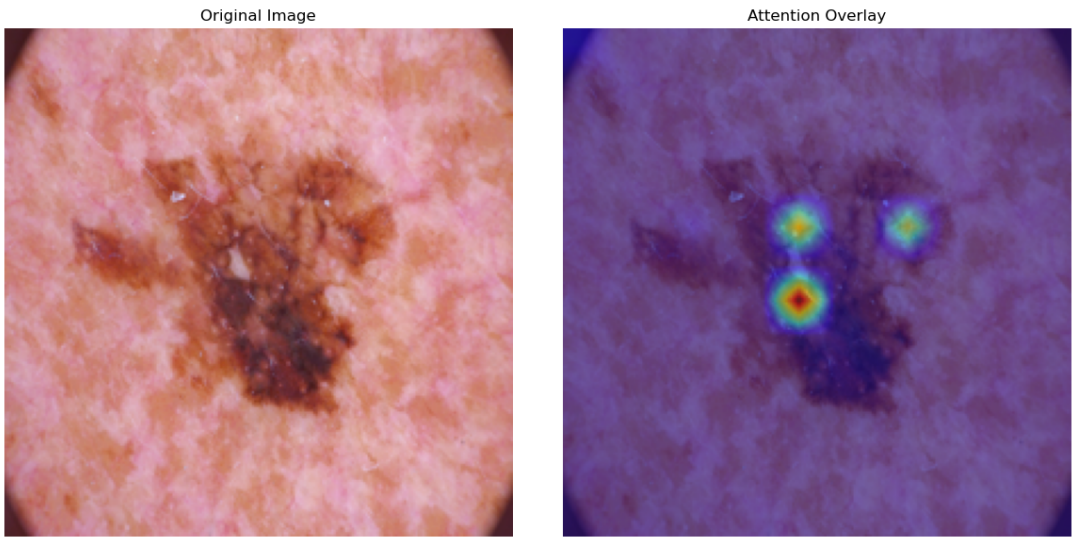


Figure 4.28: Visualization of the [CLS] token's attention map from an early encoder layer of ViT Model 5.

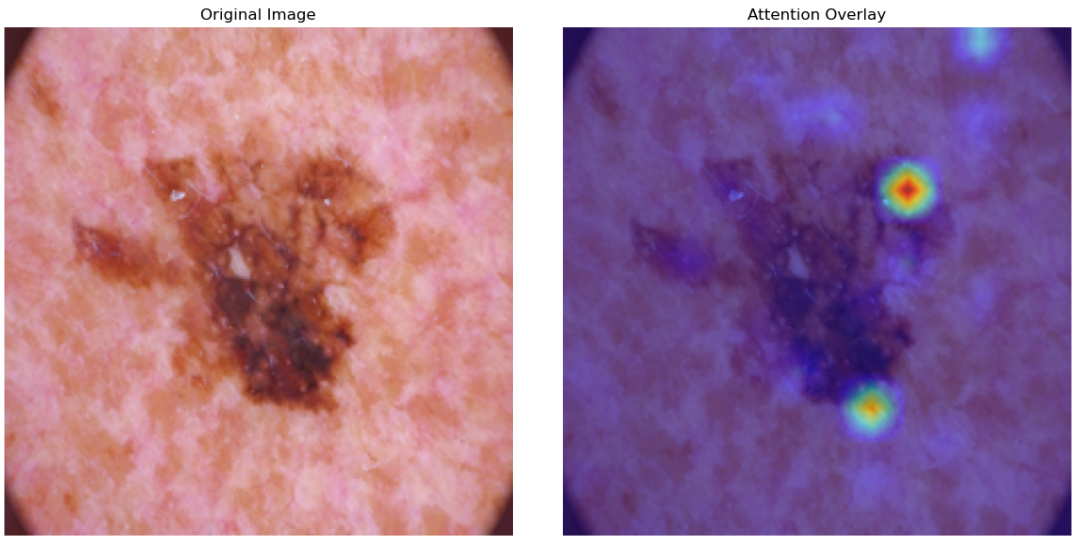


Figure 4.29: Visualization of the [CLS] token's attention map from the final encoder layer of ViT Model 5.

4.3 Comparative Analysis

This section provides a direct comparison between the best performing models from each architectural family: the ResNet-style **CNN Model 4** and the **ViT Model 5** trained with Mixup augmentation. The comparison focuses on predictive performance and computational efficiency to determine the ultimate trade-offs between the two approaches.

A summary of the final metrics for these two champion models is presented in Table 4.5. While ViT Model 5 achieved the highest overall validation accuracy, a closer look at the detailed metrics reveals a very close competition. The Macro Precision scores for both models were nearly identical, and there was only a small difference in their Macro Recall and Specificity, indicating that both architectures converged to a similarly high level of performance. In terms of efficiency, an interesting trade-off emerges: while the champion ViT model has significantly fewer parameters than the champion CNN, its training time per fold was longer due to the more extensive training schedule required by its advanced augmentation.

Table 4.5: Performance Comparison of the Best CNN and ViT Models.

Metric	**CNN Model 4 (ResNet)**	**ViT Model 5 (Mixup)**
Avg. Val. Accuracy (%)	79.25 ± 0.7	80.11 ± 1.7
Avg. Macro F1 (%)	56.61 ± 3.5	57.47 ± 6.3
Precision (%)	62.69 ± 5.9	62.68 ± 6.9
Recall (%)	54.92 ± 3.8	55.45 ± 5.6
Specificity (%)	94.13 ± 0.4	94.01 ± 0.7
Avg. Fold Time (s)	1427 ± 207	1816 ± 700
Parameters	∼11.31 M	∼2.49 M

A visual comparison of the learning dynamics is shown in Figure 4.30, which plots the validation accuracy and loss curves for the two best models. This plot illustrates

the convergence behavior of each architecture, highlighting the path each model took to reach its final performance.

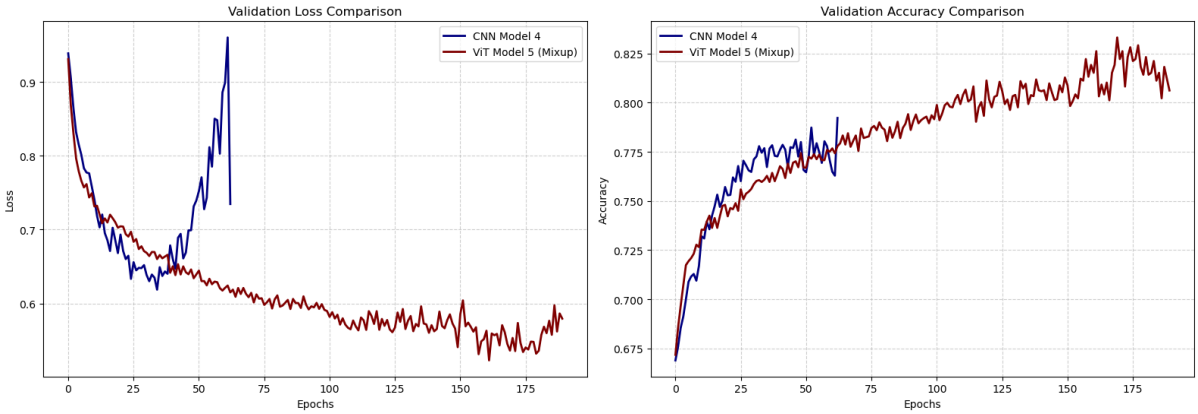


Figure 4.30: Comparison of average validation learning curves for the best performing CNN (Model 4) and ViT (Model 5). The left plot shows loss versus epochs, and the right plot shows accuracy.

Chapter 5

Discussion

This chapter provides an interpretation of the empirical results presented in Chapter 4. The performance of the custom CNN and ViT models is analyzed to draw conclusions about their relative strengths, weaknesses, and the impact of different architectural designs. The findings are then situated within the context of the existing literature, and the limitations of this study are acknowledged.

5.1 Interpretation of CNN Performance

This section provides a detailed analysis of the performance of the four custom CNN models, interpreting the quantitative results and qualitative visualizations presented in Chapter 4. The goal is to understand the impact of specific architectural choices on performance and efficiency.

The baseline **Model 1** was the most computationally efficient but also the worst-performing of the four CNNs. It achieved an average validation accuracy of 76.09% and a Macro F1 score of 49.74%. Its learning curves (Figure 4.1) reveal that the model began to overfit almost immediately, causing the early stopping mechanism to halt training in an average of just 23.5 epochs. This led to the fastest average training time of 410 seconds per fold. While fast, this rapid convergence is a symptom of the model's inability to learn generalizable features due to its lack of regularization.

The model's confusion matrix (Figure 4.6a) breaks down these performance failures. The extremely low Macro Recall of 47.70% confirms that the model failed to identify positive cases of the minority classes. This is most critical for Melanoma (MEL), where more true cases were misclassified as benign nevi (NV) than were correctly identified. In contrast, the model's high Specificity of 92.67% is a direct result of its bias towards the majority class, by defaulting to the nevus prediction, it correctly identifies a large number of true negatives for all other classes. This combination of low recall and high specificity is characteristic of a model that has not learned the features of the rare classes.

The introduction of Batch Normalization and Dropout in **Model 2** led to a significant improvement in generalization. The model achieved an average validation accuracy of 78.14% and a Macro F1 score of 51.93%. This performance increase came at the cost of a longer training time, converging in an average of 60.6 epochs with a fold time of 1748 seconds. The learning curves (Figure 4.2) confirm the effectiveness of the regularization. Unlike the baseline, the validation loss tracks the training loss closely, which indicates that overfitting was successfully mitigated.

The confusion matrix (Figure 4.6b) provides a more nuanced view of the performance.

The model's Macro Precision increased notably to 59.52%, suggesting its positive predictions were more reliable than the baseline's. However, the Macro Recall remained very low at 48.87%, showing almost no improvement. This indicates that while regularization helped the model make fewer incorrect positive predictions, it was still not significantly better at finding all true cases of the minority classes. The strong bias towards predicting Melanocytic nevi (NV) remains the primary failure.

Model 3 was designed to test the effect of increased depth by adding a fourth convolutional block and a deeper classifier. This change resulted in only a marginal performance increase, with an average validation accuracy of 78.69% and a Macro F1 score of 52.20%. However, this small gain came at a significant computational cost. Model 3 had the longest training time of all CNNs, averaging 2753 seconds per fold, and required the most epochs to converge. The learning curves in Figure 4.3 show a stable training process, similar to Model 2, indicating that the regularization remained effective even with the added depth.

The confusion matrix (Figure 4.6c) and detailed metrics confirm that this extra capacity did not lead to a breakthrough in classification. The Macro Precision (60.12%) and Macro Recall (49.09%) were nearly identical to those of the shallower Model 2. This shows that despite its greater complexity, the deeper model was no better at reliably identifying positive predictions or finding all instances of the minority classes. The result is a clear case of diminishing returns, where simply making the sequential network deeper was not an efficient strategy for improving performance on this dataset.

The architectural shift to a ResNet-style design in **Model 4** yielded the best performance among all the CNNs. It achieved the highest average validation accuracy of 79.25% and, more importantly, the highest Macro F1 score of 56.61%. This significant improvement in the F1 score, particularly the jump in Macro Recall to 54.92%, suggests that the residual architecture is substantially better at handling the dataset's class imbalance. Furthermore, Model 4 was remarkably efficient, converging in an average of 50.5 epochs and requiring significantly less training time than the deeper sequential Model 3. This superior performance and efficiency can be attributed to the residual connections, which facilitate better gradient flow.

The confusion matrix for Model 4 (Figure 4.6d) provides clear evidence of its improved discriminative power. Compared to the other models, it correctly identified a much larger number of Melanoma (MEL) cases and significantly reduced the number of MEL cases misclassified as benign nevi (NV). This indicates that the ResNet-style architecture was more successful at learning the distinguishing features of the minority classes.

The qualitative visualizations provide further validation of Model 4's success. The t-SNE plot (Figure 4.7) shows that images from the same class tend to form distinct clusters, which confirms that the model learned to create separable, meaningful representations of the different lesion types. The Grad-CAM heatmaps (Figure 4.9) confirm that the model's predictions are based on relevant visual evidence by showing that its attention is properly focused on the pathological areas of the lesion. Finally, the feature maps (Figure 4.12) illustrate a healthy hierarchical learning process, confirming that the early layers detect simple features like edges, while deeper layers learn more complex and abstract textures.

5.2 Interpretation of ViT Performance

The baseline **ViT Model 1**, characterized by its large 32×32 patch size, served as the starting point for the Transformer experiments. It achieved an average validation accuracy of 75.65% and a Macro F1 score of 47.36%. While its overall training time of 412 seconds per fold was comparable to the CNN baseline, it required nearly twice as many epochs (44.7) to converge, indicating a different learning dynamic. The learning curves in Figure 4.14 show that while a gap develops between the training and validation accuracy, the validation loss remains stable, suggesting the model suffers from some overfitting but less severely than its unregularized CNN counterpart.

An analysis of the confusion matrix (Figure 4.20a) and detailed metrics reveals the model's primary limitation. Its Macro Recall of just 45.47% was the lowest of all models tested, confirming that it was largely unable to identify the minority classes. This is particularly evident in the misclassification of a very high number of true Melanoma (MEL) cases as benign nevi. In contrast, its high Specificity of 92.72% is a direct result of this bias towards the majority class. This combination of the poor recall and high specificity shows that the broader view from the large patch size was insufficient for learning discriminative features.

ViT Model 2 was designed to test the impact of a more fine-grained input by reducing the patch size from 32×32 to a more standard 16×16. This change led to a significant performance improvement, with the average validation accuracy increasing to 78.24% and the Macro F1 score rising to 51.90%. This suggests that the longer input sequence of 197 tokens allowed the model to learn more detailed representations. This performance gain came with an increased computational cost. The training time per fold was longer at 742 seconds, which is expected as the self-attention mechanism's complexity scales with sequence length. The learning curves in Figure 4.15 show a stable training process without significant overfitting.

The confusion matrix (Figure 4.20b) provides quantitative evidence for this improvement. The model's Macro Recall saw a substantial increase from the baseline to 50.68%, which is reflected in the higher number of correctly identified Melanoma (MEL) and Benign keratosis-like lesions (BKL). This shows that the more detailed view from smaller patches was critical for improving the model's sensitivity to the minority classes. While the bias towards predicting Melanocytic nevi (NV) remains the primary failure mode, the overall improvement in both Precision (56.64%) and Recall establishes the smaller patch size as a superior architectural choice.

ViT Model 3 investigated the effect of increasing network depth by using a 5 layer encoder. This change led to a nuanced outcome in performance. While the average validation accuracy slightly decreased to 78.15%, the Macro F1 score saw a notable increase to 53.87%. This suggests that the deeper model, while not improving overall accuracy, was more effective at learning features for the underrepresented minority classes. This trade-off came at a high computational cost, as the average training time per fold increased significantly to 1256 seconds. The learning curves in Figure 4.16 show that training remained stable, though with a higher variance in validation performance compared to the shallower model.

The confusion matrix (Figure 4.20c) and detailed metrics provide a clearer picture of this trade-off. The increase in the F1 score was driven by an improvement in Macro Recall to 52.12%, reflected in the model's substantially improved ability to correctly identify Melanoma (MEL). However, this gain was accompanied by a slight decrease

in performance on other classes like Benign keratosis-like lesions (BKL). This indicates that the added depth allowed the model to learn more complex features necessary for Melanoma detection, but it did not uniformly improve performance across all classes and was computationally expensive.

ViT Model 4 was designed to explore the impact of a wider architecture by doubling the embedding dimension to 512 and the number of attention heads to 8. This change proved to be detrimental, resulting in a severe drop in performance. The model achieved the lowest average validation accuracy of all ViTs at 73.00% and a Macro F1 score of just 33.92%. The learning curves in Figure 4.17 clearly show that the model began to overfit almost immediately. A large gap forms between the training and validation curves, and the validation performance stagnates at a low level. This caused the early stopping mechanism to halt training very early, in an average of only 31.1 epochs.

The confusion matrix (Figure 4.20d) and detailed metrics confirm a near total failure to learn the minority classes. The model produced the worst Macro Precision (45.24%) and Macro Recall (31.59%) by a wide margin. This indicates that the model was both unable to find the majority of minority class cases and that its predictions for these classes were highly unreliable. The number of correctly identified Melanoma (MEL) lesions is extremely low, with the vast majority being misclassified as the benign Melanocytic nevi (NV). This demonstrates that the model's significantly increased capacity, without a corresponding increase in data or stronger regularization, made it unable to generalize.

ViT Model 5 was the overall best performing model in this study. This experiment used the architecture of Model 2 but added a powerful regularization strategy in the form of Mixup and Cutmix augmentation. This change pushed the model to the highest average validation accuracy of 80.11% and the top scores across all metrics, including a Macro F1 of 57.47%, Precision of 62.68%, and Recall of 55.45%. The learning curves in Figure 4.18 demonstrate the profound effect of this regularization. The model was able to train for a much longer schedule of 84.5 epochs on average without overfitting, allowing it to converge to a better solution. This superior performance, however, required the longest training time of all experiments.

The confusion matrix for Model 5 (Figure 4.21) highlights its strong and balanced classification performance. The key to its success was achieving the highest Macro Recall of all models, indicating it was the most effective at identifying true positive cases of the difficult minority classes. This improved sensitivity did not come at the cost of precision, as the model also achieved the highest Macro Precision and Specificity. The results strongly suggest that the architectural efficiency of the smaller patch ViT, when combined with an advanced data augmentation strategy, provides the most effective pathway to high performance on this dataset.

The qualitative visualizations provide further insight into this model's success. The t-SNE plot (Figure 4.22) shows that the model learned to group the classes into reasonably distinct clusters in its feature space. The Grad-CAM heatmaps (Figures 4.24–4.25) confirm that the model's predictions are based on relevant features within the lesion. Furthermore, the attention maps (Figures 4.28–4.29) show the model's focus evolving from a diffuse, grid-like pattern in the early layers to a more semantically meaningful focus in the final layer.

5.3 CNN vs. ViT A Comparative Perspective

The results from the two model families provide a clear basis for a comparative analysis. The best performing models from each category, the ResNet-style CNN Model 4 and the regularized ViT Model 5, both achieved strong results. However, the ViT model ultimately obtained a slight edge, securing the highest validation accuracy and Macro F1 score of all models tested. This suggests that for the task of skin lesion classification on this dataset, the architectural principles of the ViT offer a small but measurable performance advantage.

The success of the champion CNN can be attributed to its strong inductive biases. The ResNet architecture, with its convolutional layers and residual connections, is inherently optimized for processing spatial hierarchies in images. This allows it to learn powerful, localized features efficiently from a moderately sized dataset like HAM10000. In contrast, the ViT's strength lies in the self-attention mechanism's ability to capture global, long-range dependencies from the very first layer. The ViT's superior performance suggests that this capacity to model the entire image contextually was particularly beneficial for distinguishing the complex and varied patterns found in dermatoscopic images.

However, the results also highlight the different learning requirements of the two architectures. The best CNN performed very well with only standard data augmentation. The best ViT, on the other hand, only surpassed the CNN after being trained with an advanced augmentation strategy combining Mixup and Cutmix. This finding aligns with existing literature suggesting that ViTs, having weaker inductive biases, are more data-hungry than CNNs. The success of Model 5 demonstrates that powerful regularization techniques can effectively compensate for this, allowing the ViT to generalize well even without pre-training on a massive external dataset.

Finally, the comparison reveals an interesting trade-off between model size and training efficiency. The champion ViT model was significantly smaller than the champion CNN, with roughly one fifth of the parameters. Despite this, its training time was longer, a consequence of both the computational demands of self-attention and the longer training schedule required by the Mixup augmentation.

5.4 Comparison with Existing Literature

To scientifically contextualize the findings of this thesis, it is essential to compare the performance of the custom developed models against benchmarks established in recent literature. This section analyzes the results of this study in relation to contemporary works that also utilized the HAM10000 dataset, focusing on differences in methodology, model architecture, and experimental constraints.

5.4.1 Benchmark Performance on 7-Class Classification

A recent study provides a relevant and robust benchmark on the full 7-class HAM10000 classification task [33]. That work evaluated several standard pre trained CNNs and a multimodal ViT architecture, ALBEF. Their key results are summarized in Table 5.1.

Table 5.1: Performance metrics of models on the 7-class HAM10000 task.

Model	Accuracy	Precision	Recall	Specificity	F1-Score
Inception-V3	86.53%	82.02%	76.08%	95.70%	78.38%
ResNet50	85.03%	78.15%	71.29%	96.46%	72.92%
DenseNet121	88.62%	85.05%	82.49%	96.85%	83.50%
ALBEF (Image-Only)	91.32%	89.59%	85.24%	97.48%	86.93%
ALBEF (Multimodal)	**94.11%**	**90.73%**	**90.19%**	**98.33%**	**90.33%**

The results highlight two critical points. First, the standard fine-tuned models (e.g., DenseNet121 at 88.62% accuracy and 83.50% F1-Score) outperform the best custom models from this thesis (ViT Model 5 at 80.11% accuracy and 57.47% F1-Score). This performance differential is likely attributable to the use of heavily optimized architectures pre trained on large-scale datasets.

Second, the study's highest performance was achieved not by an image only model, but by the multimodal ALBEF model. By fusing the image data with just three additional pieces of patient metadata, age, sex, and lesion location, the model's performance improved significantly across all metrics, boosting the F1-Score from 86.93% to 90.33%. This finding strongly indicates that valuable diagnostic information exists in clinical data that is complementary to the visual features of the lesion.

In this context, the contribution of the present study is not to establish a new state-of-the-art benchmark, but rather to provide a systematic and controlled comparison of architectural principles under a consistent, image-only training paradigm without large-scale pre-training. The 80.11% accuracy achieved here is a strong result under these constraints and serves to validate the architectural lessons learned regarding regularization, residual connections, and ViT patch size.

5.4.2 Alternative Methodologies: Ensemble Learning on a Simplified Task

Another relevant study explores the effectiveness of ensemble learning, though on a simplified version of the classification task [34]. It is important to note that this work addresses a binary classification problem (Melanoma vs. Benign keratosis-like lesions) rather than the full 7-class task. The results, summarized in Table 5.2, are therefore a benchmark for methodology rather than directly comparable performance.

Table 5.2: Performance of models on a binary HAM10000 task (Melanoma vs. BKL)

Model	Test Accuracy
DenseNet201	95.86%
Xception	95.17%
InceptionResNetV2	95.17%
DenseNet121	94.48%
SCC-NET (Ensemble)	**97.93%**

This research demonstrates that ensembling several pre trained models into a single predictive unit can yield exceptionally high performance. This suggests that a promising direction for future work would be to create an ensemble from the best performing models

in this thesis, such as the ResNet-based CNN Model 4 and the augmented ViT Model 5, to potentially improve predictive accuracy.

In summary, the comparison with existing literature successfully positions the contributions of this thesis. The results confirm that the architectural principles investigated are sound. The performance gap relative to the state-of-the-art clearly illuminates the impact of three powerful techniques: fine tuning heavily pre trained models, incorporating multimodal data, and using ensemble methods.

5.5 Limitations of the Study

While this thesis provides a systematic comparison of custom CNN and ViT architectures, it is important to acknowledge several limitations that frame the context of the results and offer avenues for future work.

Dataset Constraints The HAM10000 dataset, while a standard benchmark, is still relatively small for training deep learning models entirely from scratch. Also, its severe class imbalance poses a significant challenge. Although this was addressed with a stratified validation strategy and robust metrics, the imbalance likely constrained the performance of all models on the minority classes.

Exclusion of Transfer Learning A deliberate choice was made to train all models from scratch to perform a fair comparison of the custom architectures' ability to learn from the data directly. This is a limitation in terms of absolute performance, as leveraging weights pre trained on large datasets like ImageNet would almost certainly have resulted in higher final accuracy scores, as shown by the literature comparison.

Unimodal Approach The models in this study were unimodal, relying exclusively on image data. This is a significant limitation, as discussed in the benchmark analysis (Section 5.4.1), where incorporating patient metadata like age, sex, and lesion location was shown to boost classification accuracy by several percentage points.

Individual Model Evaluation This work evaluated each model individually and did not explore ensemble methods. As shown in the analysis of alternative methodologies (Section 5.4.2), combining multiple distinct architectures into an ensemble can yield state-of-the-art results, often outperforming any single model. This represents a clear avenue for future performance improvement.

Scope of Architectures This study explored a specific and limited set of custom designed architectures to investigate the impact of fundamental design choices like depth, width, and patch size. A vast universe of other high performing architectures, such as EfficientNets [35], Swin Transformers [36], or Mamba-based [37] architectures, was not included in this comparison.

Limited Hyperparameter Optimization The hyperparameters for each model, such as learning rate and dropout values, were selected based on common practices and limited manual tuning. The study did not employ systematic, automated hyperparameter optimization techniques like grid search or Bayesian optimization. While the consistent use

of these parameters ensured a fair comparison between architectures, it is likely that the performance of each individual model could be further improved with a more exhaustive tuning process.

Computational Resources The experiments were conducted on a single consumer grade GPU. This constrained the practical upper limit on model size and batch size and, as mentioned above, limited the extent of hyperparameter tuning that could be performed. More extensive computational resources could have allowed for the exploration of larger models or more exhaustive training strategies.

Chapter 6

Conclusion and Future Work

This thesis presented a systematic design and comparative analysis of custom built CNNs and ViTs for the task of skin lesion classification on the HAM10000 dataset. This final chapter summarizes the key findings of the study and proposes potential directions for future research.

6.1 Summary of Findings

The investigation into custom CNN architectures demonstrated the critical importance of modern design principles. While a simple baseline model suffered from severe overfitting, the introduction of regularization techniques and, most notably, residual connections in **CNN Model 4** led to a robust and efficient model that achieved a validation accuracy of 79.25%. The experiments showed that simply increasing the depth of a sequential network yielded diminishing returns, whereas adopting a more advanced ResNet-style architecture was a much more effective strategy.

The exploration of ViT models highlighted their sensitivity to key hyperparameters and their reliance on strong regularization. A smaller patch size of 16×16 was found to be significantly more effective than a larger one. The best performance was not achieved by the deepest or widest architecture, but by applying an advanced augmentation strategy of Mixup and Cutmix to an efficient baseline. This model, **ViT Model 5**, achieved the highest performance in the entire study, with a validation accuracy of 80.11%.

In a direct comparison, the best performing ViT model showed a slight performance advantage over the best performing CNN. However, this came at the cost of a significantly longer and more complex training time. The study concludes that while both paradigms are highly capable, ResNet-style CNNs provide a powerful and efficient solution from scratch, whereas ViTs, when paired with powerful regularization, can unlock a small but measurable increase in predictive accuracy.

6.2 Potential Improvements and Future Directions

Building on the findings and limitations of this study, several promising directions for future research can be identified. The insights gained from the architectural comparisons provide a strong foundation for developing even more effective models for skin lesion classification.

Incorporate Transfer Learning The most direct path to improving predictive performance would be to fine tune models that have been pre trained on large scale datasets like ImageNet. Adapting established architectures such as ResNet50, EfficientNet, or a standard ViT would likely yield a significant boost in all performance metrics, establishing a more competitive state-of-the-art baseline.

Multimodal Data Integration The comparison with existing literature highlighted the substantial performance gains achieved by incorporating patient metadata. A key direction for future work is to extend the best performing models in this study into multimodal architectures that can process both the image data and the tabular metadata (age, sex, lesion location) available in the HAM10000 dataset.

Ensemble Methods As suggested by the literature, creating an ensemble of models is a powerful technique for improving robustness and accuracy. A promising future step would be to combine the predictions of the best performing models from this thesis, such as the ResNet-style **CNN Model 4** and the augmented **ViT Model 5**. Their different architectural approaches may have learned complementary features, leading to a more robust final prediction.

Advanced Architectural Exploration Future work could go beyond the foundational designs tested here. This includes exploring hybrid architectures (e.g., using a CNN as a feature extractor for a ViT), systematically studying the impact of scaling depth, width, and resolution in a principled manner, and investigating novel architectures beyond standard transformers, such as Mamba-based State Space Models.

Data Centric and Imbalance Strategies The performance of all models was constrained by the dataset's class imbalance. Future work should explore more advanced data centric solutions. This could involve applying sophisticated augmentation strategies like AutoAugment [38], or implementing targeted training techniques such as class weighted loss functions or focal loss [39] to give more importance to the underrepresented classes. Furthermore, validating the final models on external datasets would be a critical step to ensure their generalizability.

Exhaustive Hyperparameter Optimization A more exhaustive hyperparameter search, using automated methods like Bayesian optimization or grid search, could further optimize the performance of the custom models developed in this work and ensure that each architecture is performing at its full potential.

Bibliography

[1] Giuseppe Argenziano, H. Peter Soyer, Sergio Chimenti, Renato Talamini, Rosamaria Corona, Francesco Sera, Michael Binder, Lorenzo Cerroni, Gaetano De Rosa, Gerardo Ferrara, Rainer Hofmann-Wellenhof, Michael Landthaler, Scott W. Menzies, Hubert Pehamberger, Domenico Piccolo, Harold S. Rabinovitz, Roman Schiffner, Stefania Staibano, Wilhelm Stolz, Igor Bartenjev, Andreas Blum, Ralph Braun, Horacio Cabo, Paolo Carli, Vincenzo De Giorgi, Matthew G. Fleming, James M. Grichnik, Caron M. Grin, Allan C. Halpern, Robert Johr, Brian Katz, Robert O. Kenet, Harald Kittler, Jürgen Kreusch, Josep Malvehy, Giampiero Mazzocchetti, Margaret Oliviero, Fezal Özdemir, Ketty Peris, Roberto Perotti, Ana Perusquia, Maria Antonietta Pizzichetta, Susana Puig, Babar Rao, Pietro Rubegni, Toshiaki Saida, Massimiliano Scalvenzi, Stefania Seidenari, Ignazio Stanganelli, Masaru Tanaka, Karin Westerhoff, Ingrid H. Wolf, Otto Braun-Falco, Helmut Kerl, Takeji Nishikawa, Klaus Wolff, and Alfred W. Kopf. Dermoscopy of pigmented skin lesions: results of a consensus meeting via the internet. *Journal of the American Academy of Dermatology*, 48(5):679–693, 2003.

[2] Harald Kittler, Ashfaq A. Marghoob, Giuseppe Argenziano, Cristina Carrera, Clara Curiel-Lewandrowski, Rainer Hofmann-Wellenhof, Josep Malvehy, Scott Menzies, Susana Puig, Harold Rabinovitz, Wilhelm Stolz, Toshiaki Saida, H. Peter Soyer, Eliot Siegel, William V. Stoecker, Alon Scope, Masaru Tanaka, Luc Thomas, Philipp Tschandl, Iris Zalaudek, and Allan Halpern. Standardization of terminology in dermoscopy/dermatoscopy: Results of the third consensus conference of the international society of dermoscopy. *Journal of the American Academy of dermatology*, 74(6):1093–1106, 2016.

[3] P. Tschandl, C. Rosendahl, and H. Kittler. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific Data*, 5:180161, 2018.

[4] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[5] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[6] Marvin Minsky and Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1969.

[7] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[10] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[13] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[16] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016.

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[22] 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. *CoRR*, abs/2010.11929, 2020.

[23] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[24] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017.

[25] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *CoRR*, abs/1905.04899, 2019.

[26] Python Software Foundation. Python language reference, version 3.10.18. http://www.python.org, 2024.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pages 8026–8037, 2019.

[28] The pandas development team. pandas-dev/pandas: Pandas, 2020.

[29] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[30] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

[31] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[32] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.

[33] Abdulmateen Adebiyi, Nader Abdalnabi, Emily Hoffman Smith, Jesse Hirner, Eduardo J. Simoes, Mirna Becevic, and Praveen Rao. Accurate skin lesion classification using multimodal learning on the ham10000 dataset. *medRxiv*, 2024.

[34] Yogendra Sharma, Tushar Manger, Amar Sanyasi, Janiela Bhutia, Anushka Pradhan, and Smriti Koirala. Skin cancer classification and comparison from HAM10000 dataset images using ensemble of convolutional neural networks. *International Journal of Scientific Research & Engineering Trends*, 10(3):511–519, 2024.

[35] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.

[36] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030, 2021.

[37] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[38] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018.

[39] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[40] Aman Chadha. Transformers. *Distilled AI*, 2020. https://aman.ai.

[41] Brandon Rohrer. Transformers from scratch. https://e2eml.school/transformers.html, 2021.

[42] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. *An Introduction to Statistical Learning: with Applications in Python*. Springer, 2023.