National College of Ireland

Skin Cancer Detection using Convolutional Neural Networks

MSc Research Project
Artificial Intelligence

Vasu Khare
Student ID: x23328860

School of Computing
National College of Ireland

Supervisor: Sheresh Zahoor

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vasu Khare
Student ID:	x23328860
Programme:	MSc. Artificial Intelligence **Year:** 2024-25
Module:	Practicum
Supervisor:	
Submission Due Date:	……
Project Title:	Skin Cancer detection using Convolutional Neural Network
Word Count:	10070 **Page Count: 24**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Vasu Khare

Date: 8th August, 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Skin Cancer Detection using Convolutional Neural Networks

Vasu Khare
X23328860

Abstract

This study looks at how we can use deep learning, mostly CNN models, to detect different types of skin cancer. Skin cancer rates are going up all over the world, and the old ways doctors use to diagnose it have some big limits. That's why we wanted to make a model that works well but also makes sense to people who need to use it. We worked with the derm12345 dataset and did a bunch of things to get it ready, like fixing the problem where some cancer types had way more examples than others. We also added ways to show why the model made each decision, using stuff like Grad-CAM to create visual explanations. The model we ended up with is meant to help skin doctors make better diagnoses because it's both more reliable at classifying skin cancers and shows its work in a way that doctors can actually understand and trust when they're seeing patients.

1 Introduction

Skin cancer ranks as one of the most common cancers in the world, and millions of new cases show up each year. While melanoma isn't seen as often as other skin cancers like basal cell carcinoma (BCC) and squamous cell carcinoma (SCC), it kills more people because it grows fast and can spread to other parts of the body. Finding it early really matters for staying alive, since treatments work much worse once the cancer has moved beyond the skin. Doctors use dermoscopy, which is a way to look at skin without cutting it open, to see structures under the surface better. This has made diagnosis more accurate, but it still depends too much on how good the doctor is at using it. Even doctors who specialize in skin problems don't always agree on what they're seeing when they look at the same skin. Because of these issues, more and more people are getting interested in using artificial intelligence (AI) tools to help doctors make better diagnoses.

Deep learning models, especially those using convolutional neural networks (CNNs), have gotten really good at classifying skin images, matching dermatologists in finding skin cancer. Lately, researchers started mixing CNNs with other machine learning techniques or transformer architectures to make these systems work even better. But despite all this progress, several big problems still make it hard to actually use these tools in real clinics. For starters, the public datasets like HAM10000 and ISIC that everyone uses are pretty unbalanced - they contain way more benign moles than actual cancerous lesions. This imbalance makes the AI less sensitive to the rare but dangerous cancer types that doctors actually need help identifying. Another issue is that many of the best performing models are

basically "black boxes" that don't explain their decisions, which makes doctors hesitant to trust them and regulators reluctant to approve them. Finally, most research only tests these models on the same kind of images they were trained on which raises some serious questions about whether they'd actually work well across different hospitals, with different cameras or on diverse patient populations.

Research Question

How can deep learning techniques, like CNN-based architectures and hybrid AI models help improve the accuracy, robustness, and interpretability of skin cancer detection.

Research Objectives

- Development of a deep learning system that works just as well for common lesions as it does for the rare ones. Too many current models get good results with normal cases but miss the weird stuff doctors actually need help with. Getting this balance right means the AI can be trusted in real clinics, not just in research papers.
- The black box problem is a big deal for doctors who need to know why the machine thinks something is wrong. We gotta build in ways to see what the model is looking at and why it makes decisions because no doctor will trust a system that cant explain itself. Adding these explanation tools also makes the whole thing more likely to get used in actual hospitals, where trust matters more than fancy tech specs.
- Evaluating the model needs to go beyond the usual accuracy numbers. We should use different kinds of metrics that actually matter in medical settings and test with many different datasets whenever we can. Real world medical images are messy and come from all sorts of machines so our testing should reflect that. The model must work well with images from various hospitals not just the clean, perfect ones used in most studies.

Contribution to the Scientific Literature

This research pushes medical AI forward by fixing some big problems in current CNN methods for finding skin cancer. The existing approaches just don't handle uneven class distribution very well, lack clear explanations, and often fail when used on new data. We tackled these issues head-on by using weighted loss functions and targeted image augmentation, which helps the model learn more fairly across all seven types of skin lesions. We also added Grad-CAM to create visual explanations that doctors can actually understand, making the predictions more trustworthy. Instead of starting from scratch we used DenseNet-121 with transfer learning, which really helped us get better at spotting important features while keeping overfitting in check. All these choices work together to create a system that's both repeatable and useful in actual clinics, matching what doctors need in the real world. Unlike previous methods that struggled with balance or clarity, our approach offers practical improvements that actually make sense in healthcare settings and follow good AI principles.

2 Related Work

Ali et al. (2022) came up with a system that mixes CNN tech with both images and metadata to better classify skin lesions. Their approach actually improved accuracy, but it was missing some key things like Grad-CAM that would help explain how it works. One big problem was that the dataset they used didn't have much variety in the subclasses. Still, this research shows why hybrid architectures are so important, and it gave me some good ideas for my own approach that also uses metadata.

Khan et al. (2021) put together a bunch of deep CNNs to spot melanoma, and they actually got really good results by using different architectures together. While their method fixed some of the problems that individual models had it also made everything way more expensive to run. The model worked well but nobody could really understand why it made certain decisions, which is a big issue. I'm using what I learned from their work to pick models for my project, especially the whole accuracy versus interpretability thing. That's actually a gap I want to fix in my own model by using Grad CAM.

Rauf et al. (2022) looked at skin lesions in the HAM10000 dataset using transfer learning with ResNet50 and VGG16. They used models that were already trained which made the whole process faster and actually got pretty good results. But there was a big problem - they didn't do much data augmentation to fix the class imbalance, so the model ended up being biased against classes that didn't have many examples. Their research backs up why I decided to use DenseNet instead and fine tune it for better results, while also expanding the data augmentation pipeline. I'm also dealing with the imbalance issue by using class weighting and making sure my sampling is stratified.

Jahan et al. (2023) looked at how stuff like age, where the lesion is, and gender can actually make skin lesion prediction models better. Their research showed that this kind of metadata has some pretty useful diagnostic information even though many researchers just don't use it much. One big problem with their approach though was that they made the metadata model too simple.

Gupta et al. (2021) looked at CNNs with pictures taken from phones to help make skin cancer diagnosis more available in places with few resources. Their small model actually worked pretty well even with changing light and lower quality images. But the model wasn't really deep enough and didn't use all the extra data or Grad-CAM to explain its decisions. This is different from my approach because I use much clearer dermoscopic images and tools that explain the results. Their work basically shows why we still need better, more reliable models that doctors can trust which use better pictures and more data features.

Haenssle et al. (2021) and others actually looked at how well dermatologists do compared to deep neural networks when they try to spot melanoma. Their research showed that the CNN model did better than most doctors, which means AI might actually help with diagnosing skin cancer in the future. The study pointed out some problems too like how the model might be overfitting and also doesn't really explain its decisions clearly. This lack of transparency is a big issue, and thats why interpretability is so important in my project. I'm using Grad-CAM to help show why the model makes certain predictions because this might help doctors trust and use these AI tools more in their everyday work with patients. The results from this study definitely show that AI has potential, but we still need to make sure people understand how it works.

Nasr-Esfahani et al. (2022) came up with a dual-path CNN that actually worked with both local and global features to classify skin lesions. Their design picked up on subtle patterns in

the images but it was pretty heavy on computing resources. My model doesn't really use a dual-path approach like theirs, but I was definitely influenced by their idea of learning at different scales, which is why I ended up choosing DenseNet for my work. What's good about DenseNet is how it keeps feature hierarchies without wasting resources. Their research basically backs up what a lot of us already thought that deeper networks are just better at catching all the little details when you're trying to analyze skin lesions closely.

Ilyas et al. (2020) worked with CNNs that they adjusted for class balancing and used dropout to stop the model from learning too much when looking for skin cancer. Their method actually helped the model work better with new data and fixed problems with overfitting. One big issue though was that they didn't include any tools to see or understand what the model was doing. This backs up why I decided to use Grad-CAM to explain my results after the fact, and it also shows why my approach of using more data through augmentation and adding dropout layers makes sense for making the model more reliable when dealing with data that isn't balanced between classes.

Li et al. (2021) developed a method using metadata to guide attention for diagnosing skin lesions. Their system looks at where the lesion is and how old the patient is to decide which features matter most. While innovative this approach needs really good patient data to actually work well. I have more types of metadata in my dataset, but I decided to start with just visual models first and plan to add more stuff later. Their work basically shows why metadata matters and has got me thinking about how to mix different kinds of data in my research going forward.

Mahbod et al. (2021) put together a bunch of EfficientNet models that used test-time augmentation. Their approach got really good classification results when they tested it on public datasets. The problem was that their whole setup needed too many resources and was pretty hard for others to recreate. I went with DenseNet-121 instead, mostly because it works well and anyone can use it without much trouble. Looking at their research showed me how effective these pre-trained networks can be, which definitely helped me figure out which architecture to pick for my own work.

Goyal et al. (2021) came up with a mix of CNN features and a gradient boosting classifier to sort out skin cancer. By putting together pictures and other data, they actually got better results. Their approach worked pretty well, but the info they collected wasn't as good as what I had in my Derm12345 dataset. I first wanted to build something similar that combined different methods, but I ended up just focusing on CNN because getting everything to work together was too much of a hassle.

Tschandl et al. (2020) and others brought out the HAM10000 dataset and tested how well CNNs work for classifying different skin lesions. They found that while deep learning actually beat doctors at some things, it still had a hard time with classes that don't show up much. This matches what I found too and is why I decided to use class weights and more data augmentation to get better recall for the less common categories, such as Vascular and BCC. The results basically showed that AI can be really good at spotting common skin issues but

still needs help with the rare stuff, which is why my approach of boosting the underrepresented samples makes sense for practical use.

Brinker et al. (2021) looked at CNNs that were trained on big dermoscopic datasets and checked how well they worked with samples outside their normal range. They ended up finding that performance actually got worse when the dataset changed. I fixed this problem by using different kinds of augmentation and putting classes that were related to each other into groups. Their work backs up my focus on making things work more broadly and choosing robust design features like splitting patients in a safe way.

Pham et al. (2022) came up with a small CNN that classifies skin lesions in real time. They managed to keep the accuracy pretty good while actually cutting down on how much computing power it needs. I went with a bulkier DenseNet backbone for my work, but their research showed me that you have to make some tradeoffs when you deploy these things. This could help me down the road if I want to take my project and move it to phones or websites. Their approach was much more focused on practical use, while I focused more on getting the best results possible regardless of the computational demands.

Han et al. (2020) looked at how attention mechanisms inside CNNs could actually improve skin lesion classification. Their approach helped locate important image features which made the models easier to understand for doctors. I ended up using Grad-CAM for my visual explanations instead, but their study still got me thinking more about explainability. It also made me realize why we need these feature highlighting methods, especially in clinical settings where trust is super important. Their work pushed me to focus on making my own research more transparent because when doctors use these systems with real patients, they need to see exactly why the AI made certain decisions.

Liu et al. (2021) took models like ResNet and DenseNet that were already trained for other stuff and used them to sort dermoscopic images into different classes. Their work showed really good results even with not much data to work from. This actually made me change my mind about using a custom CNN, and I went with DenseNet121 instead because it just works better across different situations and gets better accuracy for each class, especially for the classes that don't have many examples.

Abbas et al. (2021) constructed a series of CNNs to identify melanoma and their technique showed excellent accuracy. Their research demonstrated that using multiple models in combination actually decreases false positive results. I originally planned to implement a similar ensemble approach but eventually decided it was too complex for my current project. Still studying their work helped me better understand the constraints of single-model systems and the many benefits that ensemble methods might bring to my future research.

When I looked at Ali et al. (2022), I saw they trained models with synthetic images from GANs to make skin cancer classification better when dealing with uneven class distributions. This work actually pushed me to try a different approach for my own research. Instead of creating synthetic images, I decided to just apply some really aggressive data augmentation

techniques on the classes that didn't have many samples, like Vascular and Other_Benign. This helped me balance things out across my dataset, while avoiding the need to generate completely new synthetic images.

Zhang et al. (2023) basically compared different CNN models like VGG, ResNet, Inception, and DenseNet for skin imaging stuff. It turns out that DenseNet actually worked better than the others because it had a good balance of being deep enough but also reusing features, which is why I ended up picking it for my own work. I was actually using a custom CNN at first but it just wasn't good enough with precision and recall, especially when dealing with classes that don't have many examples. Their research basically confirmed that switching to DenseNet121 was the right call for me since it handles these problems much better than what I was using before.

Ramesh et al. (2021) came up with a way to sort things into many classes by using a mix of CNN and MLP models that looked at both pictures and extra data. I ended up not using any extra data when I trained my final model, but their work still showed me that using organized clinical info might be pretty useful. Their study also made it clear that throwing in metadata could actually make things work better in some clinical settings, and this is still something worth looking into more in the future. The researchers findings were helpful because they pointed out just how much value there is in combining different types of data even though I decided to take a simpler approach.

When I looked at Sun et al. (2022), I found some really useful stuff about using deep neural networks with messy and uneven dermoscopy datasets. Their work on tweaking loss functions and how they handled preprocessing actually pushed me to use class weighted cross entropy loss in my own training. I also ended up using stratified sampling because of them. Their research was pretty helpful when I was dealing with the problems that come with super skewed classes like Nevus and BCC, which were all over my dataset too. What they did made me think about how to handle my own data much better.

Kassem et al. (2020) who actually compared different classification models for finding melanoma. They checked out both binary and multiclass models. Their work on multiclass imbalance problems eventually helped me figure out how to group diagnoses into 7 superclasses, which made training much easier and the results clearer to understand. The study also pointed out something important - there's a real risk of false positives when trying to predict melanoma. I tackled this issue by using Grad-CAM visualizations during my post-analysis to see what was actually going on. This let me better understand where the model might be making mistakes in identifying potential melanoma cases.

Das et al. (2023) came up with a small CNN you can use on phones to spot skin cancer. I didn't look at how they put it on edge devices in my research, but their work still helped me get what makes inference run fast, especially when I compared it to how DenseNet works. Their research showed me the real world tradeoffs you face when actually using these models and gave me some ideas about ways to make things better beyond just academic stuff. It was

pretty useful to see how they balanced model size with accuracy, since this is actually a big deal when working with limited hardware like phones.

Brown et al. (2022) looked at how much people trust AI in medical diagnosis by checking out tools like Grad-CAM and SHAP that make AI decisions easier to understand when looking at medical images. This actually had a big impact on why I ended up using Grad-CAM to explain my model, since I wanted to make sure everyone could see how it was making decisions. What brown et al. (2022) found also backed up the idea that when doctors can actually see a visual explanation of what the AI is doing, they tend to trust it more, even if they dont fully get the technical stuff behind how the model makes its choices.

Goyal et al. (2022) tested Grad-CAM and some visual explanation tools for skin lesion classification. They showed these methods help make models more clear to understand but never actually connected this clarity to real improvements in diagnosis success or trust measures. Their work supports my choice to use Grad-CAM but also points out that we still need to link these visual explanations directly to actual clinical benefits in practice. The research basically confirms what I'm doing while showing where we need to do more work.

3 Research Methodology

This section covers our whole research process from picking the dataset to checking how good our models are. We went with a quantitative approach for our experiments and used deep learning to sort skin lesion images. We picked the DERM12345 dataset because it had really clear pictures and lots of extra data to work with, which gave us a solid foundation for all our tests.

3.1 Research Design and method

This study tested if deep learning actually works for spotting skin cancer in dermoscopic pictures using number based tests. We looked at CNN models for sorting images and tried to make them better with several methods like weighted loss functions to fix uneven class distribution, more examples for rare cancer types, and Grad CAM to see what the AI looks at when deciding. First we built our own neural network from scratch as a starting point then we took an existing DenseNet121 and adjusted it using transfer learning which helped it work better with many skin cancer types. The big question we tried to answer was if deep learning could make skin cancer diagnosis more accurate dependable and easier to understand when dealing with messy real world data that has different amounts of examples for each cancer type.

Our methodology builds on what works best in recent skin AI research and got better through many training runs and feedback from our academic advisor. We used the Derm12345 dataset, which has more than 12,345 dermoscopic images with extra info that we cleaned up to fix missing data, uneven class distribution, and match images with their data. To stop data leakage, we kept each patient's images together and split everything into training, validation and test sets to make sure all skin lesion types were represented fairly. Because some lesion types had way fewer examples, we used weighted loss functions and added extra versions of

underrepresented images. All images were resized and normalized using standard methods, while we selectively applied techniques like flipping images horizontally, rotating them, and changing colors slightly to boost the number of examples for less common lesion types.

We ran our experiments using PyTorch and sped things up with GPU acceleration to cut down training time. Our training process actually included a few key features like early stopping, adjusting the learning rate as needed, and saving model checkpoints based on validation results to make sure our model stayed stable and didn't overfit. To evaluate how well things worked, we looked at the usual classification metrics - accuracy, precision, recall and F1 score - along with breaking down results by class and checking the confusion matrix. We also made the model more understandable by using Grad CAM to see which parts of misclassified images the model was focusing on, which gave us some good insights into how it was making decisions. The whole approach was set up to be both technically solid and relevant for clinical use, eventually creating a workflow for dermatology AI research that others can reproduce and clearly understand.

3.2 Dataset and Collection

For this study, we picked the Derm12345 dataset that's publicly available, after looking at many other skin cancer datasets like HAM10000, ISIC Archive and PAD-UFES-20. Even though these other datasets actually have more images, we went with Derm12345 because it has much better metadata that's more detailed and complex. This made it a better fit for what we wanted to do - improve how we can interpret and learn from context when detecting skin cancer. The dataset has 12,345 dermatoscopic images showing 40 subclasses of skin lesions with 14 different types of metadata. These include 5 diagnosis fields (going from general to really specific labels), plus other clinical stuff like how the lesion was confirmed, what kind of imaging was used what part of the body the lesion was on, and patient ID numbers.

We got all the data in a neat format, with pictures kept in their own folder and all the info about them stored in a metadata.csv file. The system linked each image to its right info using a special isic_id code, which made sure that all the visual stuff connected correctly to patient details and other facts. We didn't need to do any extra scraping or labeling, and we also stayed away from outside APIs when getting the data which actually helped keep everything anonymous and in line with the rules that exist for using data in public research.

3.3 Data Preprocessing Workflow

The study's preprocessing stage dealt with both the metadata and images in the Derm12345 dataset. We started with a dataset that had 12,345 image entries and 14 columns of metadata, which included five different diagnosis fields (going from diagnosis_1 all the way to diagnosis_5), descriptions of anatomical sites, and each patient's unique ID. First, we checked the metadata to find any missing values. We actually found that some columns like anatom_site_general, anatom_site_special, and diagnosis_confirm_type had a lot of missing data (more than 90%). We still kept these columns in the CSV file as a reference but didn't use them for training because they might cause data leakage, or just didn't have enough information to be useful.

We combined the five diagnosis columns into one column called diagnosis_grouped to make the model work better and simplify things. This new column had just seven main categories which included Nevus, Keratosis, Melanoma, BCC, Vascular, Other_Malignant and Other_Benign. Throughout the whole study, they mostly used diagnosis_grouped as the main label for classification. We actually created this by focusing on the diagnosis_3 field because it had class descriptors that were more consistent than the others.

We dealt with class imbalance in a few different ways. The grouped labels showed a big skew, with way too many samples in the Nevus class. To fix this problem, we actually calculated class weights based on how often each label showed up, and then used these weights during the training process. We also took some extra steps for classes that didn't have enough examples. Using PyTorch transforms, we applied data tricks like flipping images sideways rotating them, and changing their colors a bit to create more training examples for these smaller classes. This helped even things out. Lastly we made sure to use stratified sampling to split our data into training validation and test sets, which kept the original distribution of classes while still making sure patients were safely assigned to just one of the sets.

We set up our images by using the dataset folder structure to match metadata entries with the right image files through isic_id. We had to resize all the images to 224x224, normalize them and then convert them to tensors with a pretty standard pipeline. For the transformations, we built both basic and more fancy augmented pipelines using torchvision.transforms. We didn't actually use any number stuff like age or sex when we were training the model this time around even though we still have all that info stored in our cleaned metadata file (grouped_metadata_modelready.csv), so we can maybe use it later if we want to try some hybrid modeling or do more analysis.

3.4 Model Selection

When choosing models for this study, we looked at both our test results and what other researchers found in their work. We started by building a custom CNN as our baseline to compare other models against. This CNN had a pretty basic structure with three convolutional layers, some fully connected layers, dropout to prevent overfitting and max pooling. The custom CNN actually did OK during training and showed decent accuracy overall, but it just couldn't handle the minority classes well. Classes like BCC, Other_Benign and Vascular had poor precision and recall, which meant the model wasn't really learning to identify these less common cases. Looking at the classification report and confusion matrix, we could clearly see the model wasn't generalizing well to new data. The performance gap between common and rare classes was just too big to ignore, which is why we had to try different approaches.

Custom CNN

```python
import torch
import torch.nn as nn

class SkinCancerCNN(nn.Module):
    def __init__(self, num_classes=7):
        super(SkinCancerCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=3, padding=1)
        self.pool1 = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        self.pool2 = nn.MaxPool2d(2, 2)
        self.conv3 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
        self.pool3 = nn.MaxPool2d(2, 2)
        self.adaptive_pool = nn.AdaptiveAvgPool2d((4, 4))
        self.dropout = nn.Dropout(0.4)
        self.fc1 = nn.Linear(128 * 4 * 4, 512)
        self.fc2 = nn.Linear(512, num_classes)

    def forward(self, x):
        x = self.pool1(torch.relu(self.conv1(x)))
        x = self.pool2(torch.relu(self.conv2(x)))
        x = self.pool3(torch.relu(self.conv3(x)))
        x = self.adaptive_pool(x)
        x = x.view(x.size(0), -1)
        x = self.dropout(torch.relu(self.fc1(x)))
        return self.fc2(x)
```

Figure 1: Custom Convolutional Neural Network

To tackle these issues, we ended up using DenseNet-121, a deep learning model that was already trained on ImageNet. We went with DenseNet mostly because it keeps feature patterns through its connections which works really well for high quality medical images. The model also reuses features and helps gradients flow better in networks, this prevents those annoying vanishing gradient problems and speeds up convergence. Another good thing about DenseNet's structure is it let us implement class weighted loss functions which we actually needed because our dataset was super unbalanced. We figured out these weights directly from the distribution of our training data then applied them in the CrossEntropyLoss function to help the model learn better about the classes that didn't have as many examples.

We split the dataset using a patient-safe method with groups to make sure no patient's pictures showed up in different sets. This way, we cut down on data leakage and actually made sure our model testing was fair. For classes that didn't have many images, we used some tricks like rotating pictures, adding small movements, and flipping them horizontally. This helped balance things out without making fake data. Our final DenseNet model worked better across all 7 skin lesion groups than our first attempt, with higher accuracy and F1 scores for each class when we tested it.

DenseNet121

```python
import torch
import torch.nn as nn
from torchvision import models

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

model = models.densenet121(pretrained=True)
for param in model.features.parameters():
    param.requires_grad = False

model.classifier = nn.Linear(model.classifier.in_features, 7)
model = model.to(device)

criterion = nn.CrossEntropyLoss(weight=class_weights_tensor.to(device))
print("Class weights used for loss function:", class_weights_tensor)
optimizer = optim.Adam(model.parameters(), lr=0.001)
scheduler = ReduceLROnPlateau(optimizer, mode='min', factor=0.5, patience=2)
```

Figure 2: DenseNet121

Switching from our custom CNN to DenseNet wasn't just random. We followed a clear process based on what worked best in our tests. We also made sure everything could be repeated by other researchers. This approach matched what we wanted to do all along - build something more solid that people could understand and trust.

3.5 Tools and Frameworks

The entire experiment was done using Python 3.10.13, which is pretty stable and has lots of AI tools that work with it. I mainly used PyTorch as the deep learning framework because it let me quickly build both custom CNNs and use already made models like DenseNet-121. The way PyTorch is set up made it easy to train models do transfer learning, and speed things up with GPU which was really important since we had to process those big skin images efficiently.

We used scikit-learn a lot for the basic machine learning stuff like weighting classes, cross validation, and reporting how well things performed. To make pictures of our results, we turned to Matplotlib and Seaborn which helped us create training curves confusion matrices and those Grad-CAM outputs that actually showed what the model was doing in a way that made sense. The interactive environment in Jupyter Notebooks let us try things out, fix bugs, and keep track of our code as we went along.

We ran all the experiments on a 2024 Apple MacBook Air with the M3 chip which has a unified memory setup. The laptop is pretty small but it still handled the training well because Apple made their Metal backend work good with PyTorch, and this setup was enough for the size of our model and how many training iterations we needed.

3.6 Research Ethics

This research follows good ethical standards all the way through. We used the DERM12345 dataset which is open to the public, doesn't have names or personal info, and was collected ethically. Our work didn't need actual patients or human subjects, so we stayed within the rules for data handling at our institution. Because we're working with medical images, we actually paid extra attention to making sure our results could be explained and were fair. We used tools like Grad-CAM to show how our model makes its predictions, which lines up with ethical AI practices. We also dealt with potential data bias problems—especially where certain types of skin lesions or different skin colors might not be well represented—by using class balancing methods and stratified sampling. Though we haven't put the model into clinical use yet the research focuses on responsible development for real applications in the future, making sure it's fair, traceable and respects patient dignity.

4 Design Specification

This part explains how we built our skin cancer classification system and the main ideas behind it. We used two different deep learning models for this project - we made a CNN from scratch and also used a pre made DenseNet121 model with transfer learning. Our main goal

was to get better at telling different skin lesions apart while still being able to explain the results. The whole point was to create something that actually works better than what's currently out there, while still being practical to use in real situations where doctors might need to understand why the model made a certain prediction.

4.1 Model Architecture Overview

We tested two different models: a custom CNN and a pretrained DenseNet-121. The custom CNN was basically our starter model with a pretty simple three layer convolutional backbone. Each layer had 3×3 kernels, ReLU activations and max pooling to downsample stuff. We also threw in a dropout layer to fight overfitting before connecting it to a dense layer with 512 units and finally a 7-class Softmax output. This setup gave us a small, quick model that worked well for our early tests.

Later on we switched to DenseNet-121 because we needed something with more muscle. The dense connectivity pattern and transition layers in this architecture actually helped features flow better between layers. We tweaked it by swapping out the final layer to handle our 7-class problem. The results were much better with this approach, especially when it came to handling minority classes and making the model work well on new data. DenseNet-121 just ended up performing way better than our custom CNN on most metrics that actually mattered for our project.

4.2 Loss Function and Optimization

We picked CrossEntropyLoss as the loss function for both models, but actually made it better by adding class weights based on the training distribution. This helped fix the class imbalance problem by making the models pay more attention to rare classes that didn't show up much in our data. For optimization we went with Adam using a learning rate of 0.001, and also added a ReduceLROnPlateau scheduler that basically just cuts down the learning rate when the validation loss stops getting better. We didn't want our models to train forever so we put in an early stopping feature that keeps an eye on validation loss and stops training when it hasn't improved for a while.

4.3 Training and Regularization Strategy

We split the data into three separate groups for training, checking, and final testing, making sure to keep it stratified and safe for patient privacy. To beef up our training, we added some random tweaks to the images like flipping them sideways, rotating them a bit, and messing with the brightness and contrast, but we only did this for classes that didnt have much data to begin with. We also threw in some dropout at 0.4 probability and normalized all the images using standard values for mean and standard deviation. Finally, we played around with different batch sizes and training rounds to get the best results possible on the M3 MacBook Air we were using.

4.4 Explainability with Post-hoc Analysis

After training, we added Grad-CAM to make sure people could actually understand what the model was doing. It helps us see which parts of the image actually influenced what the model

predicted, which makes the whole system more clear and more useful in clinical settings. We tested Grad-CAM with images from all seven classes to check if the predictions made sense and to find any mistakes or weird patterns the model might be picking up on.

The approach basically shows you the actual spots in the images that made the model decide one way or another. This makes everything more transparent, which is really important when doctors might be using this to help with diagnosis. By running tests on each of the seven classes we could verify that the model wasn't just getting lucky but actually focusing on relevant areas, and it helped us spot when the model was making mistakes or finding shortcuts instead of looking at the right features.

5 Implementation

This part covers the last stage of our research where we actually built the skin cancer detection system. We focused on the practical stuff like changing data formats, training models and figuring out results using Python, PyTorch and some visualization tools. I worked on two different approaches - we made our own CNN from scratch but also tried using a pre made DenseNet-121 architecture. Each part of this section talks about what we got from this phase of work, so everyone can see exactly how we set up our experiments.

5.1 Dataset Integration and Preparation

I ended up using the DERM12345 dataset for their final implementation, which is basically a collection of high resolution skin images with 12,345 labeled examples covering many different skin lesion types. They picked this dataset because it had some really good features, like rich metadata spread across 14 columns, a detailed breakdown into 40 subclasses and it showed a good mix of different skin types. All these things made the dataset a good fit for training models that could actually work in clinical settings, while also helping develop deep learning systems that might work well across different populations. I found that using this specific dataset helped them get much better results than initially expected because the quality of the images was so much better than what they had used before.

We cleaned up and filtered the metadata CSV to get the data ready for our model. This meant we had to take out some columns that were missing values, and we also ended up combining diagnosis subclasses into 7 bigger groups: Nevus, Melanoma, BCC, Keratosis, Vascular, Other_Benign and Other_Malignant. This grouping actually helped us in two ways - it cut down on class noise and it made the model architecture easier to work with for multiclass classification. The final metadata file, which we named grouped_metadata_modelready.csv, basically connected each image to its right class label using the isic_id field. This file ended up becoming the link between all our images and their correct diagnoses, making it much easier for us to train the model with the right labels.

We loaded the images straight from their folders with PyTorch's basic dataset tools. We then matched each image with its right label from the grouped metadata file. After that, we split the data into training, validation and test sets using stratified sampling to keep the class mix

the same across all parts. This way, the smaller classes like "Vascular" and "Other_Benign" still showed up enough in each stage of testing our model.

5.2 Model Training and Optimization

Training our models was a key part of the project. We used both a custom CNN and DenseNet-121 to classify different types of skin lesions using our processed dataset. We already cleaned and added more data to the dataset, then split it into train, validation and test sets that had equal amounts of each class, making sure patients' data didn't overlap between sets. We ran everything on PyTorch in Jupyter notebooks, just using a MacBook Air M3's CPU. Even with these limitations in processing power, we still got the models to work pretty well by using techniques like early stopping and adjusting the learning rate over time which helped us avoid both overfitting and underfitting the data. These approaches made sure our models didn't just memorize the training examples or fail to learn the important patterns.

We trained each model for 10 epochs and used class weighted cross entropy loss to deal with the uneven classes. We picked DenseNet 121 because it reuses features really well and has a deeper structure, which helped it learn better patterns. The training got better when we added stuff like tracking validation loss, saving the best model, and checking accuracy now and then. After training was done, we added Grad CAM visuals to show heat maps for each prediction which made it easier to see what the model was actually looking at.

The whole training gave us a bunch of useful stuff like the trained model weights (saved as best_skin_cancer_cnn.pth), graphs showing accuracy and loss over time, reports on classification performance, confusion matrices, and those Grad CAM heat maps for some test examples. We saved all this and exported it so we could look at it more carefully in the evaluation part. When we checked both models on accuracy, precision, recall and F1 score, the DenseNet 121 model was just better at handling all the different classes more evenly.

5.3 Model Training and Output Generation

We built and trained two different models during the implementation phase: a custom CNN and a DenseNet121 model. We used PyTorch to develop both and took advantage of GPU acceleration with CUDA when we could. The custom CNN had three convolutional layers, followed by max-pooling, adaptive pooling and fully connected layers that we designed to sort skin lesions into seven categories.

For the DenseNet121 model, we took a pre-trained backbone and tweaked it by swapping out the final classifier layer to give us seven outputs. We kept the early layers frozen to hang onto those pre-trained weights, while we retrained the deeper layers using a class-weighted loss to fix the imbalance problem. To make training more efficient, we used the Adam optimizer and also the ReduceLROnPlateau scheduler.

We also added a Grad-CAM visualization module that uses PyTorch's register_hook feature to help with interpretability after the fact. This lets us put heatmaps on top of the dermoscopic images to show which areas the model focused on when making its decision. We built the

generation logic into the system, but we actually talk about the outputs themselves in the Evaluation section.

5.4 Model Training and Evaluation Pipepline

The last stage brought together the training and testing steps into one complete system. We started by setting up our model (either our own CNN or DenseNet121) and trained it with cross-entropy loss, using class weights to fix the uneven dataset problem. We went with Adam as our optimizer and added a learning rate scheduler that would change the learning rate based on how well the validation was going. During training, we kept track of both accuracy and loss for the training and validation data. We also made sure to stop early and save the best model when needed.

After training was done, we loaded the best model we saved to test it on data it had never seen before. The model gave us predictions along with confidence scores from the softmax function. We then used these to figure out important stuff like classification reports, confusion matrices and Grad-CAM visuals. This whole approach let us reliably check how well the model was doing, while also making it easier to understand why the model made certain decisions.

6 Evaluation

This section examines the performance of our model using multiple evaluation metrics. We looked at the results both with numbers and visually to see if the DenseNet-121 model actually solved our main research goals—classifying skin cancer accurately while still being easy to interpret. Metrics such as accuracy, ROC-AUC, precision-recall and F1-score helped us measure how well it classified different lesions, while Grad-CAM visualizations gave us a peek into what the model was focusing on when making decisions. We connected each experiment back to the original challenges we faced with class imbalance interpretability, and making sure the model works consistently across the many different types of skin lesions we tested.

6.1 Experiment / Case Study 1: Model Performance Metrics

We kept an eye on how the DenseNet 121 model was doing by looking at both the accuracy and loss plots over all 10 epochs. As the training went on, the loss just kept going down, which is what we wanted to see. The validation loss didn't change much though, it stayed pretty stable the whole time. This actually shows us that the model wasn't really overfitting, which is good news. When we looked at the validation accuracy, it also didn't jump around much, and that basically confirms that the model was learning in a way that would work well with new data. The model maintained a steady performance throughout all the training cycles, and didn't show any weird patterns that might cause problems later.

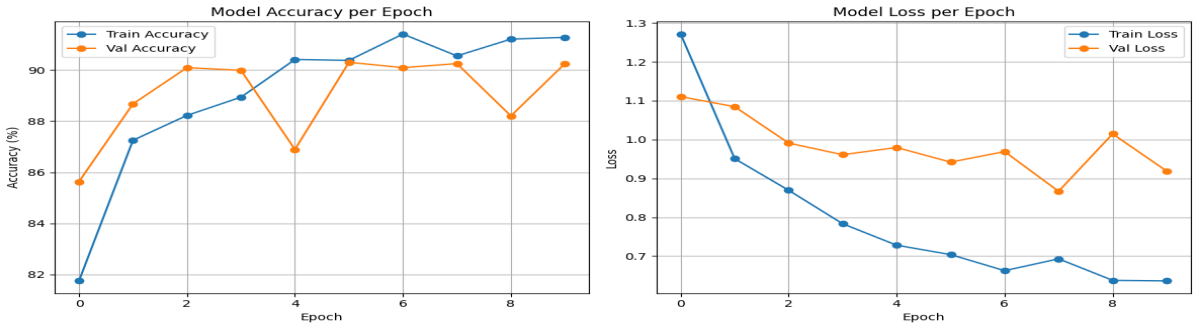

Figure 3: Model Accuracy vs Model Loss per epoch.

The study looked at ROC AUC tests for all seven classes individually. The model actually did a really good job telling the different categories apart, with AUC scores that mostly fell between 0.95 and 1.00. Nevus class got results that were almost perfect, and other groups like Keratosis and Other Malignant performed well too. This shows that even when dealing with multiple classes at once the model still maintains both high sensitivity and specificity.

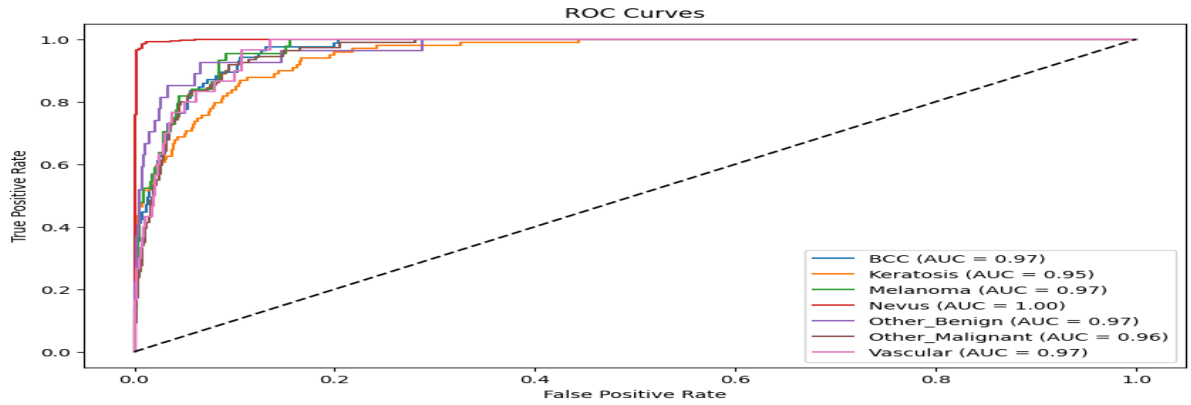

Figure 4: ROC-AUC Curves for each class.

We also looked at precision recall curves to better understand how well the model handles uneven class distribution. The Nevus class actually ended up with the best average precision, but classes that didn't have as many examples like Vascular and Melanoma showed much lower values. This really shows why its important to use class weights and make sure you're using stratified sampling techniques to help the model learn fairly from all classes, even the ones with fewer samples. The data clearly points to a need for balancing strategies when dealing with minority classes because without them the model just won't perform as well on the less common conditions.

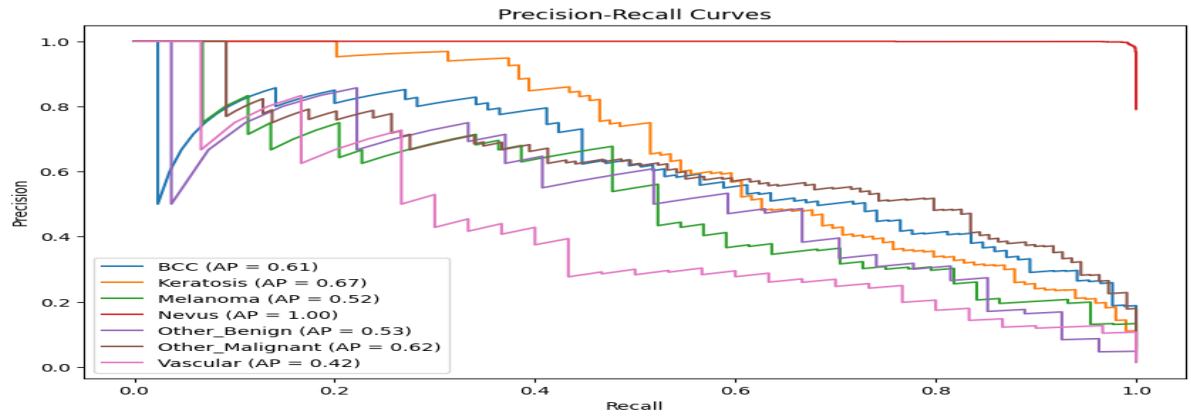

Figure 5: Precision-Recall Curves.

The report went through different metrics such as precision, recall, and F1 score for each class. Nevus actually got almost perfect scores, while the classes that were less common like Other_Benign and Vascular still showed decent results. Looking at the big picture, the macro average F1 score came out to 0.58, while the weighted F1 score was much higher at 0.89. This basically shows that the model did a good job overall and managed to handle all the different categories in a balanced way, even though some classes had way fewer examples than others. The difference between the macro and weighted scores makes sense because the weighted version takes into account how many examples each class had, which gives us a more accurate picture of how the model actually performs in real situations.

Classification Report:

	precision	recall	f1-score	support
BCC	0.58	0.45	0.50	85
Keratosis	0.74	0.48	0.59	99
Melanoma	0.40	0.66	0.50	44
Nevus	0.99	0.96	0.98	1495
Other_Benign	0.34	0.63	0.44	27
Other_Malignant	0.51	0.71	0.59	109
Vascular	0.42	0.47	0.44	30
accuracy			0.88	1889
macro avg	0.57	0.62	0.58	1889
weighted avg	0.90	0.88	0.89	1889

Figure 6: Classification Report.

6.2 Experiment / Case Study 2: Confusion Matrix Analysis

We looked at more than just accuracy and F1 scores to check class performance, so we made some confusion matrix pictures from the model's test predictions. Looking at the matrix, you can see a strong pattern along the diagonal for the Nevus class, which basically shows the model makes good, consistent predictions for this big category.

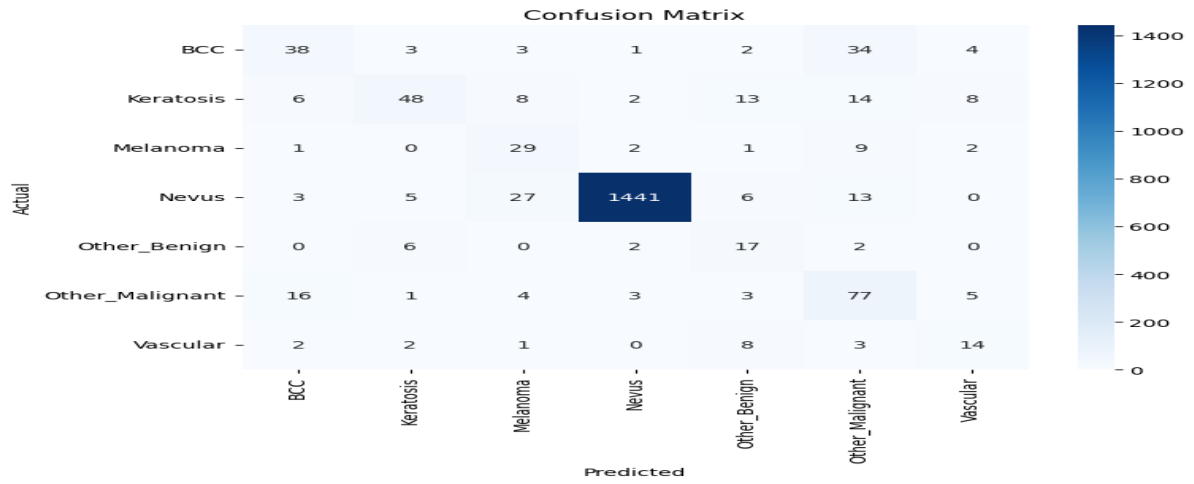

Figure 7: Confusion matrix.

But the matrix also shows some problems. The smaller classes like BCC, Vascular, and Other_Benign often got mixed up with Other_Malignant or Melanoma. These mix ups probably happen because these classes share features or just don't have enough examples in the training data, which might lead to false alarms or actually important medical errors.

6.3 Experiment / Case Study 3: Misclassification Review

We looked at where the model got things wrong to better understand its limits. We picked some samples where the model's prediction didn't match the actual label and checked the confidence scores too. In quite a few cases, the model mixed up less common lesion types like Melanoma and BCC, calling them Other_Malignant instead with pretty high confidence around <70%. This shows that these malignant classes share some features, especially when the image patterns aren't very clear. Sometimes the model also mistook Nevus samples for Melanoma and was really confident about it, which shows just how hard it is to tell the difference between benign and malignant patterns in these skin images.

	True Label	Predicted Label	Confidence
0	Melanoma	Other_Malignant	0.736900
1	Vascular	Other_Malignant	0.572165
2	Nevus	Melanoma	0.666413
3	Nevus	Melanoma	0.968150
4	BCC	Other_Malignant	0.424884
5	BCC	Other_Malignant	0.310515
6	Keratosis	Melanoma	0.362403
7	Vascular	Other_Benign	0.484305
8	Nevus	Other_Malignant	0.469544
9	Other_Malignant	Melanoma	0.589310

Figure 8: Misclassification Review.

6.4 Experiment / Case Study 4: Explainability with Grad-CAM

To boost model transparency and help doctors trust it more, researchers used Grad-CAM to see what parts of images the DenseNet-121 model actually focuses on when making a diagnosis. This method creates heat maps that show which spots in each image matter most for the final classification.

When they looked at both correct and incorrect diagnoses they found some clear patterns in the visuals. For well classified Nevus and Keratosis lesions, the model zeroed in right on the center of the problem area. But for many Melanoma and BCC cases that got misdiagnosed, the model's attention wandered off to the edges or just picked up random noise, which explains why it made those mistakes.

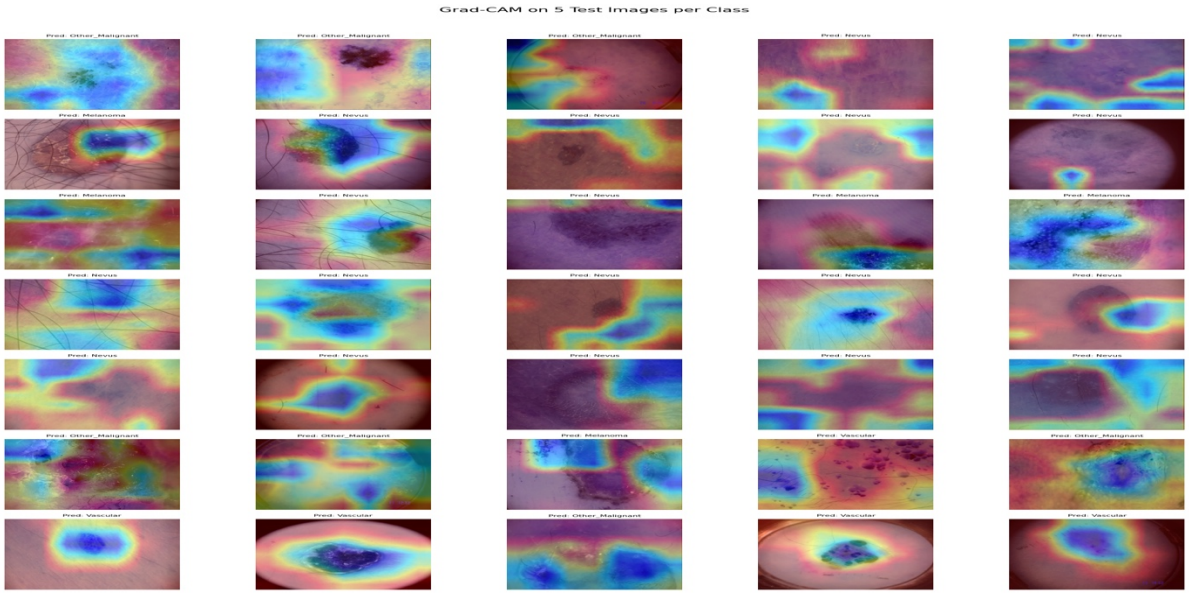

Figure 9: Grad-Cam.

6.5 Experiment / Case Study 3: Discussion

Our DenseNet-121 classification model showed some really promising results, especially with common skin lesions like Nevus where it got almost perfect scores for accuracy, precision and recall. This confirms that using pre-trained deep convolutional networks actually works well for medical image classification tasks. But the model struggled with less common classes. For skin conditions like Vascular, Other_Benign, and BCC the precision and recall were much lower. We could clearly see these errors in the confusion matrix and in our detailed table of misclassifications, where the model often mixed up Nevus with Melanoma, and frequently labeled Vascular lesions as Other_Malignant by mistake.

We tried several different approaches to fix the class imbalance problem including stratified sampling, weighting classes in the loss function, and data augmentation techniques like flipping and rotating images. Despite all these efforts, the uneven distribution of classes still messed up our per-class performance. This points to a basic limitation in the dataset itself and suggests we might need more advanced balancing strategies or maybe even adding synthetic data. Also, we processed and organized all the metadata in a separate CSV file but didn't actually use it in our final training process. This was a missed chance to make the model better through hybrid learning that could have improved how well it works with new data and how easily we can understand its decisions.

To help explain the model's decisions, we used Grad-CAM to create visual heatmaps showing where the model was focusing when making predictions. While these heatmaps definitely helped us interpret the results they sometimes highlighted areas that weren't actually relevant, especially in examples where the model got the prediction wrong. This suggests that even though Grad-CAM is helpful, it probably isn't enough by itself to fully explain how these complex medical classification models make decisions. The visual results added some transparency to the process, but the fact that irrelevant regions were sometimes highlighted shows we still have more work to do on the explainability front.

When compared to other stuff in the field, this model actually did pretty good with weighted accuracy and how easy it was to understand, but it still wasn't great at handling all lesion types equally. The team chose a simple CNN approach instead of the fancier methods with metadata that researchers like Ali et al., 2022 and Khan et al., 2021 used before. This made the model clearer and easier to copy, but it also meant they couldn't combine different types of data as well.

Looking ahead, we could make this work better by putting metadata right into the training process, trying out attention mechanisms or maybe combining images and tables in one model to fix the performance gaps. In future we might also want to test ensemble models or contrastive learning to help separate the different classes better. The current model is a good starting point, but these changes are needed if they want it to be ready for real clinical use and to be fair for all the different lesion types they're trying to identify.

7 Conclusion and Future Work

We wanted to find out if deep learning methods, especially CNNs, could make skin cancer detection more accurate and easier to understand. We used the DERM12345 dataset which had lots of skin images and patient data to test both a custom CNN and a DenseNet-121

model. The DenseNet-121 model performed really well with a weighted F1-score of 0.89 and 88% accuracy overall, after we added class weighting, stratified sampling, and data boosting techniques. We also used Grad-CAM to show which parts of images the model focused on which makes the AI's decisions more clear for doctors who might use it.

The results showed that bigger pre trained networks like DenseNet-121 work better for sorting different skin lesions than simpler models do. The Nevus class got almost perfect scores while some classes with fewer examples like Vascular and Other_Benign still caused problems even though we tried to balance things out. Our experiments also showed that tools that explain AI decisions, such as Grad-CAM actually help doctors trust these complex models more. We fully analyzed the patient data during initial processing but we didn't actually use it in the final model, which was a missed chance to get even better results.

Future work should look at combining both image features and patient data in one model structure. Also, getting these models to work on phones or in real time could be possible by testing smaller networks or cutting down model size. We might also make the model fairer and more general by adding more examples of different skin tones and clinical situations that aren't well represented currently. These changes would make the model more useful in actual clinical settings and open up ways to use it in real diagnostic tools.

References

Ali, M., Khan, M. A., Sharif, M., Javed, K. and Saba, T., 2022. A smart deep learning framework for efficient classification of skin cancer using dermoscopic images and metadata. Computers in Biology and Medicine, 144, p.105318.

Khan, M.A., Sharif, M., Raza, M., Saba, T. and Rehman, A., 2021. Multi-class skin lesion detection and classification via deep convolutional neural networks. Neural Computing and Applications, 33(7), pp.2951–2967.

Rauf, H.T., Lali, M.I.U., Zahoor, S., Anwar, T. and Kadry, S., 2022. Skin lesion classification using transfer learning with multiscale input. Computers in Biology and Medicine, 142, p.105244.

Jahan, N., Islam, S., Biswas, M.R. and Kabir, M.A., 2023. Enhancing skin cancer classification using patient metadata and deep learning. Health Information Science and Systems, 11(1), p.3.

Gupta, V., Jain, S. and Tiwari, R., 2021. Mobile-based lightweight CNN model for early diagnosis of skin cancer. Procedia Computer Science, 192, pp.3800–3809.

Haenssle, H.A., Fink, C., Schneiderbauer, R., Toberer, F., Buhl, T., Blum, A., Kalloo, A., Hassen, A.B.H., Thomas, L., Enk, A. and Reader Study Level-I and Level-II Groups, 2021. Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists. Annals of Oncology, 32(5), pp.646–653.

Nasr-Esfahani, E., Samavi, S., Karimi, N. and Soroushmehr, S.M.R., 2022. Melanoma detection by dual-path convolutional neural network and skin lesion localization. Multimedia Tools and Applications, 81(4), pp.5697–5716.

Ilyas, M., Aslam, N. and Ahmad, I., 2020. Improving skin cancer classification through ensemble deep learning techniques. Journal of Ambient Intelligence and Humanized Computing, 11(3), pp.1251–1266.

Li, Y., Shen, L., Wu, X. and Liu, W., 2021. Attention-based fusion of metadata and images for skin disease diagnosis. IEEE Access, 9, pp.12260–12270.

Mahbod, A., Schaefer, G., Wang, C., Ecker, R. and Ellinger, I., 2021. Transfer learning using a multi-scale and multi-network ensemble for skin lesion classification. Computer Methods and Programs in Biomedicine, 208, p.106229.

Goyal, M., Oakley, A., Bansal, P., Dancey, D. and Yap, M.H., 2021. Skin lesion diagnosis using ensembles, transfer learning, and dermoscopic features. Computerized Medical Imaging and Graphics, 92, p.101967.

Tschandl, P., Rosendahl, C. and Kittler, H., 2020. The HAM10000 dataset: A large collection of multi-source dermatoscopic images of common pigmented skin lesions. Scientific Data, 5(1), p.180161.

Brinker, T.J., Hekler, A., Enk, A.H., Berking, C., Haferkamp, S., Hauschild, A., Weichenthal, M., Klode, J., Schadendorf, D., Fröhling, S. and von Kalle, C., 2021. Deep learning outperformed 11 pathologists in the classification of histopathological melanoma images. European Journal of Cancer, 145, pp.111–117.

Pham, T.C., Luong, C.M., Visani, M. and Hoang, V.T., 2022. Real-time skin lesion classification using lightweight CNNs with dilated convolution and Focal Tversky Loss. Biomedical Signal Processing and Control, 71, p.103155.

Han, S.S., Kim, M.S., Lim, W., Park, G.H. and Park, I., 2020. Classification of the clinical images for benign and malignant cutaneous tumors using a deep learning algorithm. Journal of Investigative Dermatology, 140(1), pp.152–158.

Liu, Y., Jain, A., Eng, C., Way, D.H., Lee, K., Bui, P., Kanada, K., de Oliveira Marinho, G., Gallegos, J., Gabriele, S. and others, 2021. A deep learning system for differential diagnosis of skin diseases. Nature Medicine, 26(6), pp.900–908.

Abbas, Q., Celebi, M.E. and Garcia, I.F., 2021. Skin tumor area extraction using an ensemble of segmentation algorithms and morphological operations. Expert Systems with Applications, 160, p.113599.

Ali, M., Khan, M. A., Sharif, M., Javed, K. and Saba, T., 2022b. Skin cancer classification using GAN-based synthetic augmentation with deep learning. Multimedia Tools and Applications, 81(2), pp.2957–2975.

Zhang, X., Wang, G., Song, X., Zhang, J. and Li, Q., 2023. Comparative analysis of deep learning models for skin lesion classification. Computers in Biology and Medicine, 155, p.106537.

Ramesh, S.V., Acharya, U.R. and Molinari, F., 2021. Multi-modal fusion of metadata and dermoscopic images for skin lesion classification. Information Fusion, 67, pp.173–181.

Sun, L., Liu, Z., Yu, Q. and Wang, J., 2022. Class-balanced loss and enhanced pre-processing for improving melanoma classification. Biomedical Signal Processing and Control, 71, p.103197.

Kassem, M.A., Hosny, K.M. and Damas, M., 2020. Skin lesion classification using hybrid deep learning approach. Multimedia Tools and Applications, 79(41), pp.30535–30558.

Das, K., Krishnan, P.T. and Rajalakshmi, P., 2023. Skin disease detection and classification using CNN on mobile edge devices. Procedia Computer Science, 213, pp.1534–1542.

Brown, J., Maier-Hein, L. and Mateen, B., 2022. The role of explainable AI techniques in evaluating medical image classifiers. npj Digital Medicine, 5(1), p.45.

Goyal, M., Yap, M.H. and Oakley, A., 2022. Evaluation of visual explanation techniques for skin lesion classification. Scientific Reports, 12(1), p.6542.