Dermnet: Classificação de Doenças de Pele

Daniel Valadares de Souza Felix[1], **Gustavo Silvestre Almeida Conceição**[1], **João Vitor Lima de Melo**[1], **Larissa Valadares Silqueira**[1], **Leonardo Barbosa Brandão**[1]

[1]Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)

Resumo. *Este artigo apresenta o desenvolvimento do Dermnet, um sistema baseado em aprendizado de máquina para a classificação de doenças de pele a partir de imagens. Utilizando a rede neural ResNet50, o modelo foi treinado e validado em um conjunto de dados de imagens dermatológicas. Os resultados indicaram a presença de overfitting e uma acurácia de 56,7% no teste final. O sistema foi implementado com Flask para o back-end e HTML, CSS e JavaScript para o front-end, sendo hospedado na nuvem através da plataforma PythonAnywhere. O Dermnet se mostrou promissor como ferramenta de suporte para diagnósticos preliminares em ambientes acadêmicos e hospitalares, com futuras iterações focadas em aumentar a precisão e a usabilidade do sistema.*

1. Introdução

A análise de imagens dermatológicas desempenha um papel crucial na detecção precoce e no tratamento de doenças de pele. No entanto, a complexidade e a variabilidade da condição torna a análise manual dessas imagens suscetível a erros. Diante desse desafio, surge a necessidade de um sistema automatizado que possa auxiliar profissionais de saúde e pesquisadores na classificação precisa de diversas doenças de pele. Este projeto, utiliza técnicas de processamento de imagens e aprendizado de máquina para desenvolver uma solução eficiente e confiável para classificar as doenças de pele disponíveis em Dermnet, uma base de dados composta por imagens que representam 23 tipos de doenças dermatológicas. [Goel 2020]

1.1. Contextualização

A aplicação de técnicas de processamento de imagens e aprendizado de máquina na medicina tem se mostrado extremamente promissora, especialmente no campo da dermatologia. A detecção precoce de doenças de pele é fundamental para um tratamento eficaz, e as tecnologias de aprendizado de máquina podem acelerar e aprimorar significativamente esse processo. Redes neurais convolucionais (CNNs), como a ResNet50, têm sido amplamente utilizadas para a classificação de imagens médicas, demonstrando alta acurácia em diversos estudos.

1.2. Problema

A análise manual de imagens dermatológicas é um processo complexo e suscetível a erros, que depende da experiência e do julgamento subjetivo dos profissionais de saúde. Para superar essas limitações, é necessário desenvolver um sistema automatizado capaz de classificar diversas doenças de pele com alta precisão. Tal sistema não só reduzirá o tempo necessário para um diagnóstico, mas também minimizará a possibilidade de erros.

1.3. Objetivos

O objetivo principal deste projeto é desenvolver um sistema capaz de classificar diversas doenças de pele a partir de imagens dermatológicas. Este sistema deve fornecer suporte confiável para diagnósticos preliminares, auxiliando tanto no ambiente acadêmico quanto no hospitalar. Especificamente, o projeto visa:

- Implementar um classificador de doenças de pele utilizando a arquitetura ResNet50.
- Garantir que o sistema seja acessível através de uma interface web intuitiva, desenvolvida com tecnologias modernas de front-end.
- Integrar o sistema com uma infraestrutura em nuvem para garantir escalabilidade e disponibilidade.
- Avaliar o desempenho do sistema utilizando métricas de acurácia e eficiência, com foco em minimizar falsos negativos (doenças que não foram classificadas corretamente).

1.4. Justificativa

A automação na análise de imagens dermatológicas traz inúmeros benefícios, incluindo a redução do tempo de diagnóstico e o aumento da precisão. Um sistema automatizado pode processar grandes volumes de imagens de forma rápida e consistente, algo que seria impraticável manualmente. Além disso, ao fornecer suporte confiável para diagnósticos preliminares, este projeto tem o potencial de contribuir significativamente para a triagem médica e a educação na área de dermatologia.

2. Revisão da Literatura

Nesta seção, serão discutidos os artigos que contribuíram para o desenvolvimento do projeto Dermnet, oferendo orientações para o uso de redes neurais convolucionais na classificação de doenças dermatológicas. Além disso, são destacadas a importância de otimizar o desempenho do hardware, considerando o uso de GPU no projeto proposto, e também, os benefícios da computação em nuvem, já que o sistema será hospedado a partir da utilização do PythonAnywhere.

O artigo de Schielein investiga a detecção de outliers na dermatologia, avaliando o desempenho de diferentes redes neurais convolucionais (CNNs) na classificação binária de doenças inflamatórias da pele. O estudo destaca a importância de selecionar dados profissionais para treinamento e discute as variações de desempenho entre diferentes arquiteturas de CNN. [Schielein et al. 2023]

Assim como o projeto Dermnet, essa abordagem utiliza redes neurais convolucionais para a classificação de doenças de pele. Mas enquanto Schielein aborda especificamente a detecção de doenças inflamatórias e a importância da seleção de dados de treinamento, a proposta do Dermnet visa a classificação de uma ampla gama de doenças de pele.

O estudo de Shi avalia diferentes métodos de colorização de imagens infravermelhas em sistemas embarcados de baixa potência da NVIDIA Jetson. O desempenho de 11 métodos de colorização de imagens NIR foi testado em três configurações diferentes de placas NVIDIA Jetson, com o método Pix2Pix mostrando o melhor desempenho. [Shi et al. 2023]

Embora o foco deste artigo seja diferente do sistema desenvolvido, ambos compartilham o uso de hardware acelerado pela NVIDIA. Enquanto Shi se concentram na colorização de imagens infravermelhas, o Dermnet utiliza GPU para acelerar o treinamento e a inferência do modelo de classificação de doenças de pele. Ambos os projetos demonstram a importância de otimizar o desempenho em sistemas de hardware específicos para aplicações de processamento de imagens.

Fahmi et al. desenvolveram um sistema em nuvem inteligente com um servidor de processamento de imagens para o diagnóstico de doenças cerebrais, direcionado a hospitais com recursos limitados. O sistema utiliza a biblioteca ITK e serviços web para analisar imagens médicas em tempo real, fornecendo diagnósticos automáticos. [Fahmi et al. 2017]

Sua semelhança com o projeto é a utilização de infraestrutura em nuvem para processar imagens médicas. Ambos os sistemas visam melhorar o diagnóstico automatizado, a partir do uso da computação em nuvem na área da saúde, sendo que a diferença entre ambos é apenas o objetivo da classificação.

Essas comparações mostram como o projeto Dermnet se alinha com as tendências e tecnologias atuais na área de processamento de imagens médicas e aprendizado de máquina, ao mesmo tempo em que aborda desafios específicos na classificação de doenças de pele.

3. Metodologia

Essa seção descreve em detalhes os processos e técnicas utilizados para desenvolver o sistema de classificação de doenças de pele, incluindo a implementação da rede neural, a construção do sistema front-end e back-end, seu processo de hospedagem e a avaliação de desempenho da rede neural.

3.1. Rede Neural

O treinamento e teste da base de dados foram realizados utilizando o Google Colab. A rede neural implementada foi a ResNet50, uma arquitetura de rede neural convolucional profunda, composta por 50 camadas. Esta arquitetura é conhecida por sua capacidade de melhorar a precisão ao lidar com redes neurais profundas, devido à sua estrutura de ”residual learning”. A execução foi feita em GPU, utilizando uma Tesla T4 com 2560 núcleos CUDA.

Devido ao desbalanceamento inicial dos dados, o conjunto de classes foi reduzido para 10, escolhendo-se as classes com mais imagens disponíveis. Foi estipulado um limite máximo de 800 imagens por classe para balancear a rede, resultando em um total de 7457 imagens. A rede foi treinada em 18 épocas, com 80% das imagens para treino e o restante para validação. O teste foi realizado com 2728 imagens, e os resultados serão discutidos na próxima seção.

3.2. Implementação do Sistema

O desenvolvimento do front-end foi realizado no Visual Studio Code (VSCode), a partir das tecnologias HTML, CSS e JavaScript, para construir a interface do usuário. A implementação foi feita adotando um design responsivo, garantindo que a interface se

adapte adequadamente a diferentes tamanhos de tela e dispositivos, proporcionando uma experiência consistente e intuitiva em nos mais diversos dispositivos. A Figura 1 mostra a página inicial do website, ilustrando a interface desenvolvida.

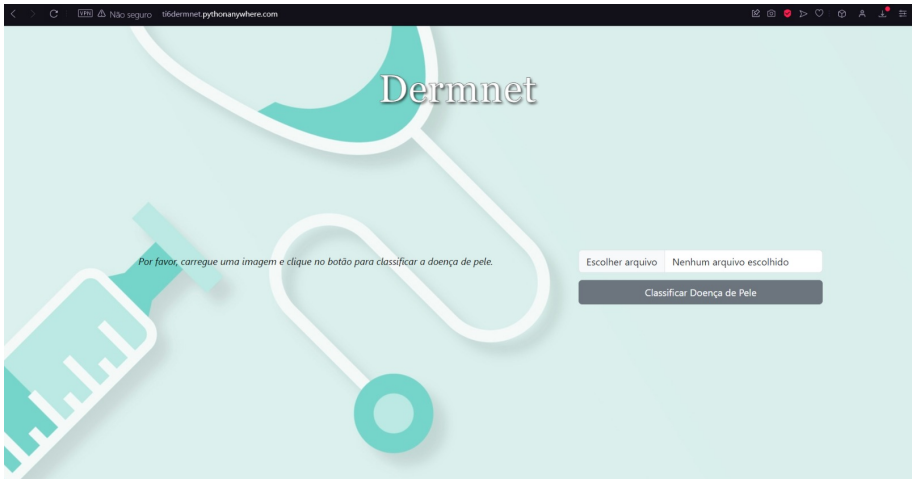

Figura 1. Interface do Sistema

Para o desenvolvimento do back-end, foi utilizado o framework Flask, que é um microframework para Python, conhecido por sua simplicidade e flexibilidade. Ele facilita a criação de aplicativos web, permitindo a integração fácil com diversas bibliotecas e ferramentas.

No sistema desenvolvido, o Flask atua como o intermediário entre o front-end e a rede neural ResNet50. Quando o usuário envia uma imagem através da interface do usuário, o Flask processa a requisição. A imagem é então pré-processada para garantir que esteja no formato correto para ser analisada pela rede neural. Em seguida, a rede neural classifica a imagem e retorna a classe à qual a imagem pertence. O resultado é então exibido ao usuário na interface do website.

3.3. Avaliação de Desempenho

A rede neural do sistema foi executada utilizando a GPU Tesla T4 da NVIDIA, que possui 2560 CUDA Cores distribuídos em 40 SMs. Essa GPU foi disponibilizada pelo Google Colab, e a execução do treinamento durou 377 segundos. No entanto, como não se pode manipular o número de threads utilizadas pela GPU no Colab, foi necessário buscar outra abordagem para avaliar o desempenho.

Os testes de desempenho foram realizados com a base de dados utilizando uma CPU: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, com 4 núcleos e 8 threads. A escolha dessa CPU permitiu a configuração da quantidade de threads utilizadas durante o treinamento da rede neural, utilizando as funções *set_intra_op_parallelism_threads* e *set_inter_op_parallelism_threads*.

Para verificar a escalabilidade forte, que é a capacidade de um sistema lidar com um aumento de carga mantendo a quantidade de dados constante, o treinamento foi realizado com as 10 classes escolhidas para implementação do sistema, ao longo de 12 épocas, com tamanho de batch igual a 64. A quantidade de threads foi variada em 1, 2, 4 e 8, e os resultados obtidos foram os seguintes:

Threads	Tempo de Execução	Ganho	Eficiência
1	1827 segs		
2	1179 segs	1,55	38,7%
4	941 segs	1,94	48,5%
8	853 segs	2,14	53,5%

Tabela 1. Dados da Escalabilidade Forte

Esses resultados indicam uma melhora na eficiência e na capacidade de processamento à medida que o número de threads aumenta. A Figura 2 exibe os gráficos de ganho e eficiência obtidos.

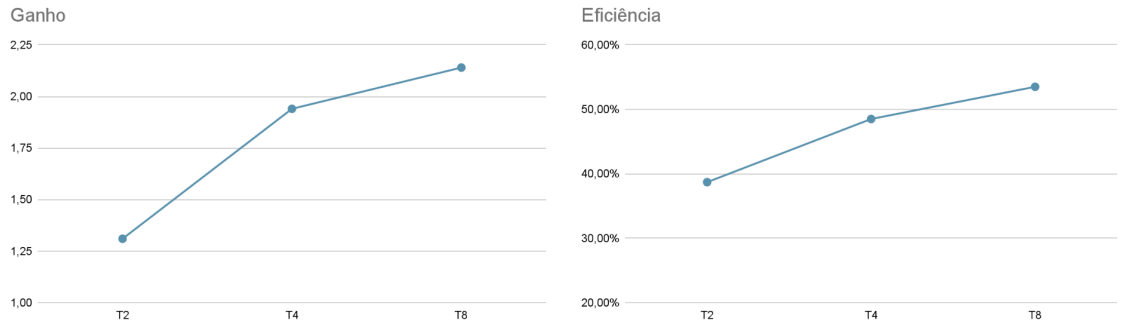

Figura 2. Speed Up da Escalabilidade Forte

No caso da escalabilidade fraca, que é a capacidade do sistema de lidar com aumentos de carga ao adicionar mais recursos (como mais classes ou dados), a configuração base da rede neural foi alterada para que a RAM pudesse suportar a nova quantidade de dados a serem processados. Os treinos foram realizados em 20 épocas com batch de tamanho 32, variando a quantidade de classes e threads:

Classes	Threads	Tempo de Execução	Ganho	Eficiência
10	1	2747 segs		
12	2	2093 segs	1,31	34,3%
14	4	1903 segs	1,44	36,0%
18	8	1863 segs	1,47	36,8%

Tabela 2. Dados da Escalabilidade Fraca

Esses resultados mostram que o sistema consegue manter um desempenho razoável mesmo com o aumento da quantidade de dados. Em comparação com os resultados obtidos na escalabilidade forte, tem-se pouca evolução do ganho e a eficiência, devido à complexidade adicional de processamento. A Figura 3 exibe os gráficos de ganho e eficiência para a configuração detalhada.

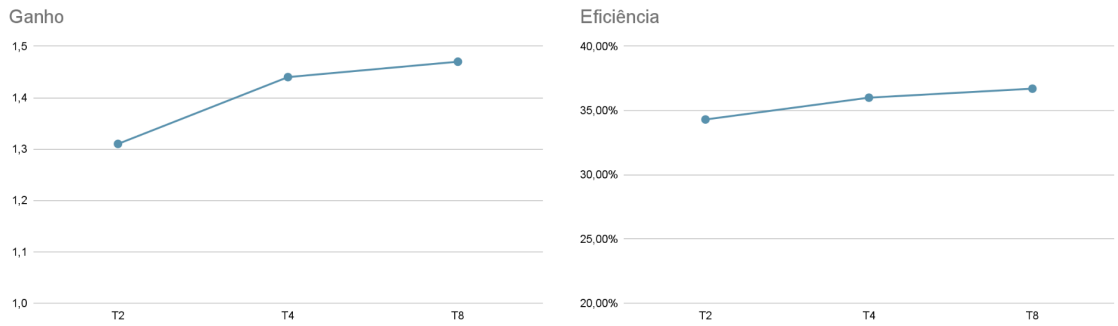

Figura 3. Speed Up da Escalabilidade Fraca

3.4. Hospedagem em Nuvem

A implementação do sistema foi realizada no PythonAnywhere, uma plataforma que oferece um ambiente Python pré-configurado e pronto para uso. PythonAnywhere permite que desenvolvedores hospedem e executem aplicativos web utilizando servidores WSGI, que gerenciam a comunicação entre a aplicação e a web.

Para este projeto, o código foi hospedado em um diretório próprio no PythonAnywhere, utilizando o servidor WSGI para aplicação com Flask. O microframework Flask foi utilizado para o gerenciamento de rotas e requisições HTTP, permitindo uma integração eficiente e rápida com a rede neural. A configuração do ambiente incluiu a definição de regras de entrada para permitir acesso via HTTP e HTTPS, a substituição da instância pela g4dn.xlarge para maior capacidade de processamento, e a verificação do funcionamento local do Flask via terminal.

A aplicação foi entregue através do Gunicorn, um servidor WSGI de produção, rodando na porta 8000. Um arquivo .service foi criado para especificar as configurações do Gunicorn, e um upstream foi adicionado para transmitir a aplicação localmente para a web. A definição do ProxyPass do upstream eliminou a necessidade de especificar a porta para acessar a aplicação. A configuração final do WSGI garantiu o gerenciamento eficiente da aplicação, permitindo um acesso universal e eficiente aos serviços oferecidos pelo sistema. A Figura 4 exibe a estrutura dos arquivos no PythonAnywhere, ilustrando como o ambiente foi organizado para hospedar o sistema de forma eficiente e escalável.

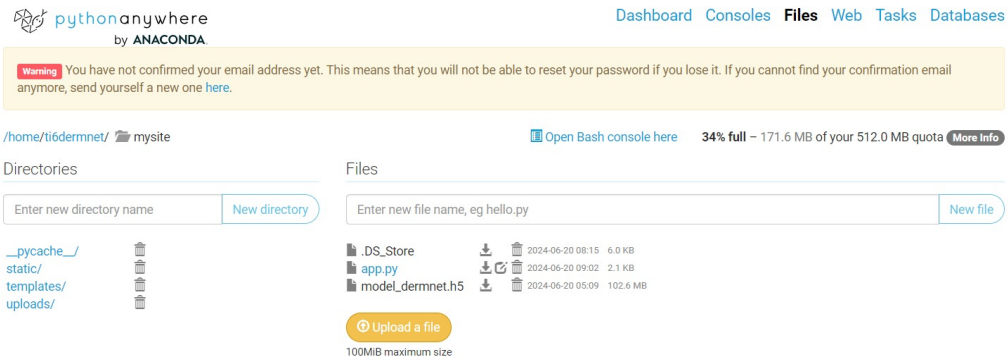

Figura 4. Estrutura de Arquivos no PythonAnywhere

4. Resultados

Os resultados obtidos a partir da implementação e treinamento da rede neural ResNet50 apresentaram variações significativas entre os conjuntos de treino e validação, indicando a presença de overfitting. A Figura 5 ilustra as curvas de perda e acurácia tanto para o conjunto de treino quanto para o de validação ao longo das 18 épocas de treinamento. A acurácia média do conjunto de treino foi de 84%, enquanto a acurácia do conjunto de validação foi significativamente menor, alcançando apenas 55%.

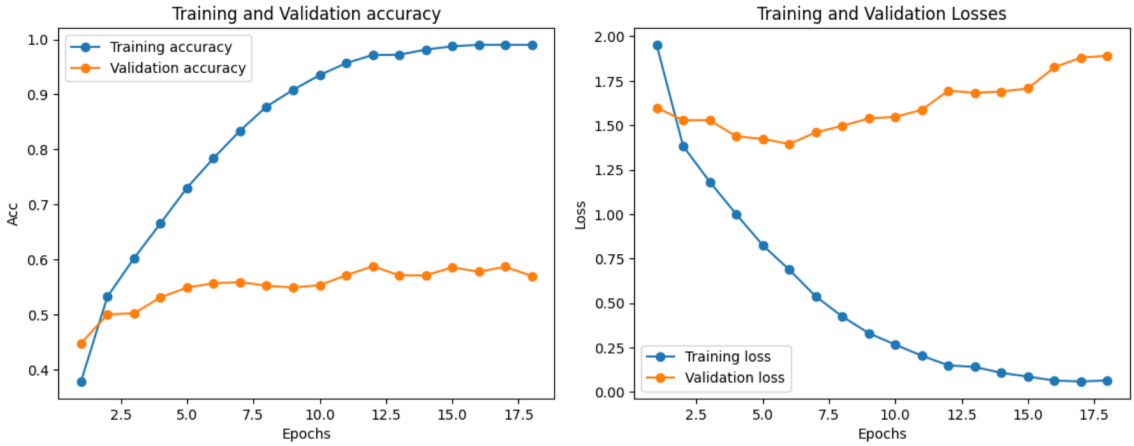

Figura 5. Curvas de Perda e Acurácia da Rede Neural

A elevada acurácia no conjunto de treino sugere que a rede neural foi capaz de se ajustar bem aos dados fornecidos para essa etapa. No entanto, a discrepância acentuada em relação à acurácia do conjunto de validação é um indicativo claro de overfitting. Esse fenômeno ocorre quando o modelo se ajusta demais aos dados de treino, incluindo ruídos e padrões específicos desses dados, mas falha em generalizar para novos dados não vistos anteriormente. A grande quantidade de dados disponíveis para o treino, em comparação com o menor volume de dados destinados à validação, pode ter contribuído para esse comportamento, uma vez que a rede teve mais oportunidades de memorizar os exemplos de treino do que de aprender padrões generalizáveis.

Na Figura 6, aprentase-se a matriz de confusão do modelo após a realização do teste, que obteve uma acurácia de 56,7%. Esse valor está em consonância com a acurácia obtida durante a validação, reforçando a hipótese de que o modelo sofre de overfitting.

A matriz de confusão revela como as diferentes classes foram preditas pelo modelo, destacando áreas de alto desempenho e outras onde o modelo apresentou dificuldades. Analisando a matriz, pode-se observar que certas classes são mais frequentemente confundidas entre si, o que sugere a necessidade de um maior balanceamento dos dados ou a utilização de técnicas adicionais de regularização para melhorar a generalização do modelo.

5. Conclusões

Neste projeto, foi desenvolvido um sistema denominado Dermnet, capaz de classificar doenças de pele a partir de imagens utilizando técnicas de aprendizado de máquina e computação distribuída. Os principais objetivos eram fornecer suporte confiável para

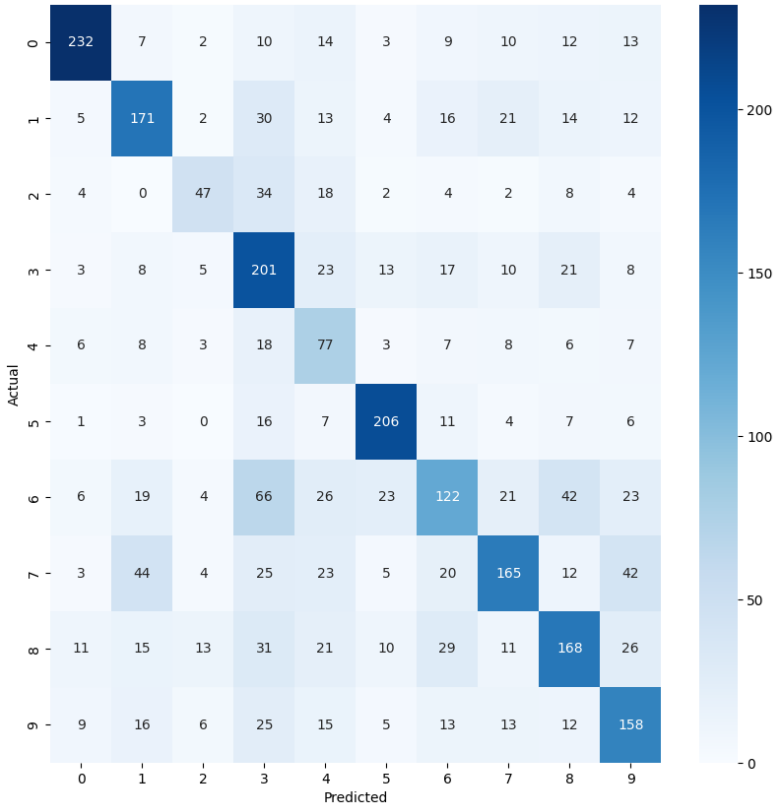

Figura 6. Matriz de Confusão

diagnósticos preliminares no ambiente acadêmico e hospitalar, e desenvolver uma ferramenta automatizada para auxiliar na detecção precoce de doenças de pele, contribuindo para um tratamento mais eficaz.

Considerando os objetivos iniciais, o sistema desenvolvido mostrou-se promissor, mas ainda há espaço para aprimoramentos. Entre as possíveis melhorias estão a ampliação do conjunto de dados de validação, a aplicação de técnicas de data augmentation, e a exploração de diferentes arquiteturas de redes neurais ou parâmetros de treinamento. Além disso, a otimização da infraestrutura de hospedagem e a melhoria da interface do usuário podem contribuir para uma experiência mais robusta e eficaz.

Referências

Fahmi, F. et al. (2017). Smart cloud system with image processing server in diagnosing brain diseases dedicated for hospitals with limited resources. *Technology and health care*, 25(3):607–610.

Goel, S. (2020). Dermnet.

Schielein, M. C. et al. (2023). Outlier detection in dermatology: Performance of different convolutional neural networks for binary classification of inflammatory skin diseases. *JEADV*, 37(5):1071–1079.

Shi, S. et al. (2023). A comparative analysis of near-infrared image colorization methods for low-power nvidia jetson embedded systems. *Frontiers in neurorobotics*, 17:1143032.