Skin Cancer Detection: Computer Vision and Deep Learning Techniques

Aditya Bajaj (abajaj225@berkeley.edu) *Gerrit Lensink (gerrit.lensink@berkeley.edu)*
Rohit Srinivas (rkshatriya@berkeley.edu) *Ruby Han (rubyhan@berkeley.edu)*

GitHub Repository: https://github.com/abajaj25/MNIST-Skin-Cancer-with-Jetson/tree/main

1. Abstract

Our aim is to introduce an automated system which will be able to detect skin lesion type in dermoscopic images in this paper. The original challenge was defined by International Skin Imaging Collaboration (ISIC) with the purpose of improving melanoma detection. We utilized computer vision and deep learning modeling techniques through transfer learning by taking into account the imbalanced data in each class of the dataset. We attempted various networks with different width and depth and the best neural networks were DenseNet and EfficientNet models. Our best accuracy is 0.783 with EfficientNetB7 using ISIC 2018 test data.

2. Introduction

Skin cancer is one of the most dangerous and commonly diagnosed cancers in the United States [1]. Skin cancer is caused by damaged DNA in skin cells that reproduce abnormally, generating skin genetic defects or mutations [2]. When detected at early stages, it is one of the most curable cancers. To combat the rising mortality rate of skin cancer cases, mechanisms for early detection are required for disease diagnosis in its earliest stages. Proper treatment can produce a five-year survival rate over 98% [3]. If cancer progresses to the lymphatics system or beyond, the survival rate drops to as low as 16% [3].

Scientific researchers have developed various early detection computer vision techniques to recognize the different types of skin lesions and provide a supporting tool for skin cancer diagnosis. Skin lesion parameters such as shape, size, symmetry and color shades are used to identify skin cancer and differentiate between benign vs malignant skin cancer (melanoma).

International Skin Imaging Collaboration (ISIC) [4] defines some of the obstacles for developing automated methods which could help in improving melanoma diagnosis. One of the tasks for the ISIC 2018 challenge was skin lesion classification of dermoscopic level images in a given dataset.

The model inputs are skin images and the outputs are seven skin lesion classifications. The general flow in skin cancer diagnosis is obtaining the images, cleaning the data and classifying it as depicted in Figure 1.

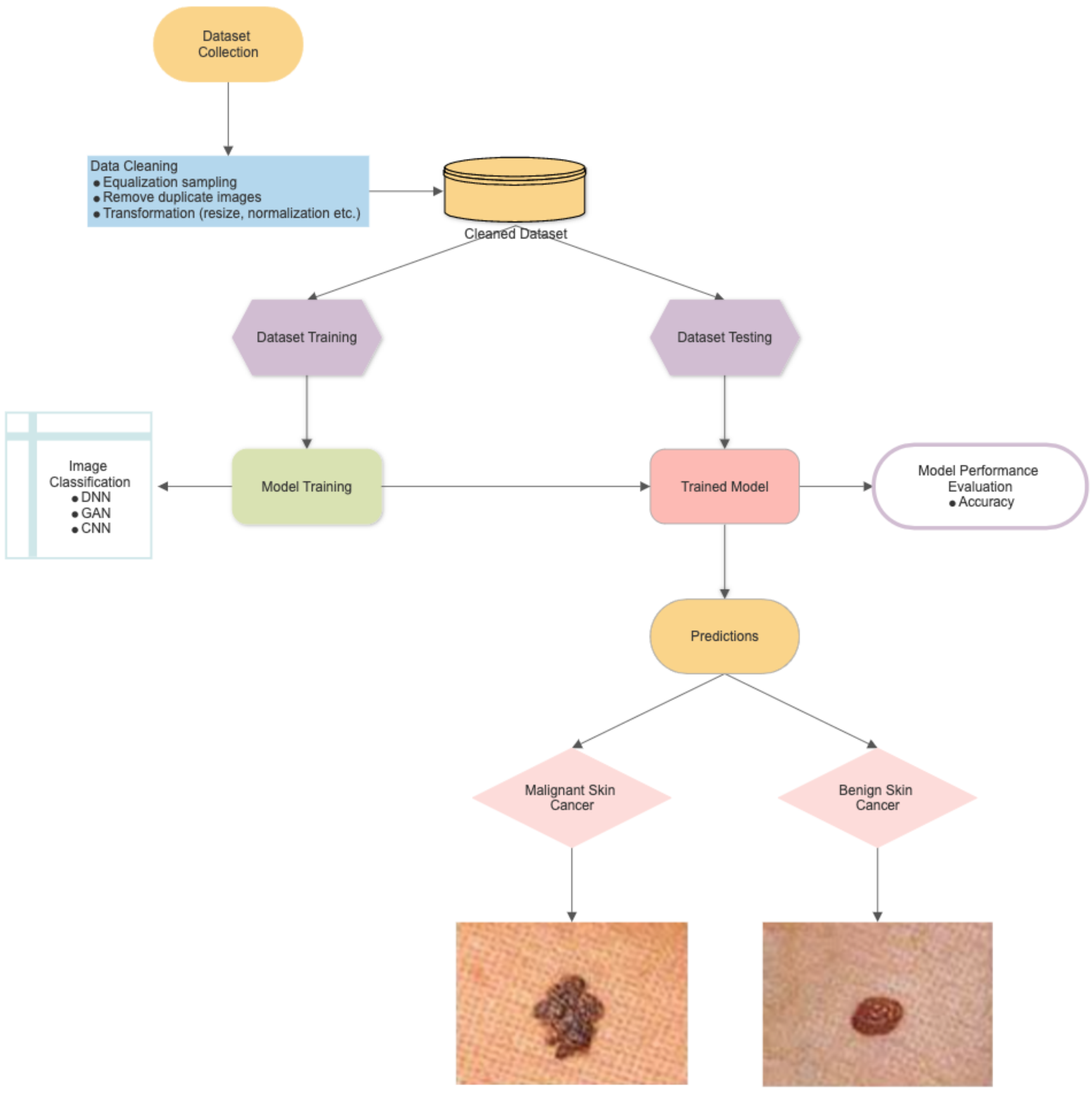


Figure 1. The modelling process of skin cancer detection. DNN = Deep Neural Network; GAN = Generative Adversarial Neural Network; CNN = Convolutional Neural Network; RNN = Recurrent Neural Network

This paper presents a systematic review for skin cancer detection using computer vision and deep learning techniques. Reputable published research papers with relevant skin cancer topics were reviewed and research findings are presented in trends, tables, flow chart frameworks, and image depictions for deeper understanding.

3. Edge to Cloud Infrastructure

Our work seeks to expand on pre-existing skin cancer detection models by creating an end-to-end pipeline which receives user photos as input on an edge device, and returns lesion type from a model deployed in the cloud.

We create this system by passing user images from a Jetson Nano 4GB to our lesion-classification model, trained and deployed in the cloud (Figure 2).

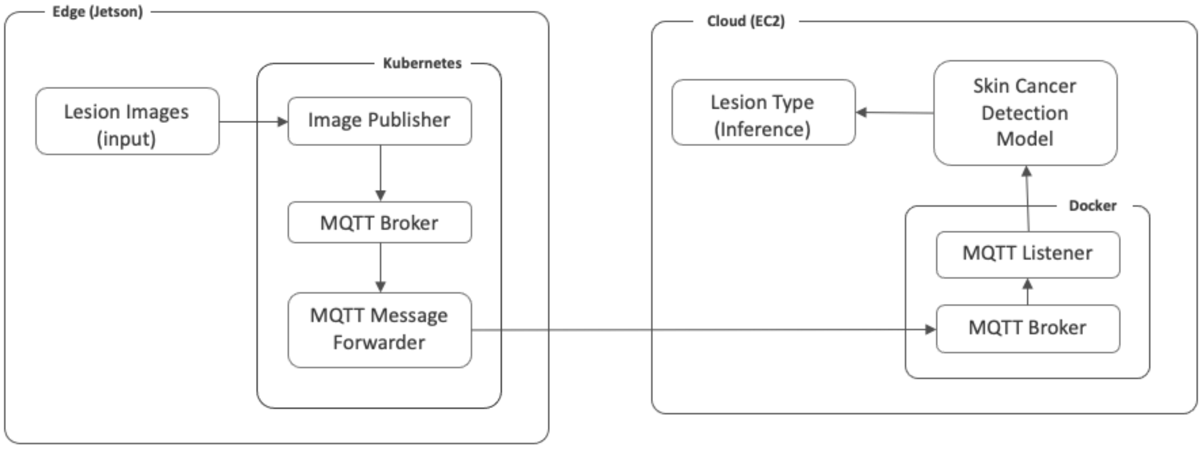


Figure 2. Pipeline infrastructure. Photos are read in on Jetson Nano 4GB, and streamed across MQTT to the cloud for inference.

The Edge

Our edge device is an ubuntu-configured NVIDIA Jetson Nano 4GB, which is set up to mimic a lightweight device that a common user might have access to. Our application is configured to receive user lesion photos as a folder of .jpg images. Once input is uploaded, lesion photos are forwarded through multiple kubernetes deployments which are used to orchestrate Docker containers and python processes. Each of these deployments serves different portions of the pipeline, like formatting and publishing images, or forwarding data from the edge to the cloud. Data is transferred across Mosquitto Networks.

Networking with Mosquitto

Mosquitto is a lightweight network protocol that allows streams of data across TCP/IP. In our pipeline, Mosquitto is employed to enable quick and dependable transfer of images within, as well as across, the edge and cloud. Mosquitto communicates data across a topic, which can be accessed across different devices by enabling forwarding and listening clusters. Each of these clusters have specific ports and host IP's for publishing to and subscribing to topics.

Mosquitto supports three qualities of service (QOS's) which serve specific purposes. QOS 0 will send a message once and only once - and is the lightest-weight deployment. This QOS depends on a stable network, and may require users to re-send data in case of failure. QOS 2 is the most exact, and will send the data as many times as necessary to ensure the sender has received the packet exactly once. QOS 1 will send the data as many times as is necessary to ensure the sender has received the message at least once, which may result in duplications of data. For the purposes of our pipeline, we employ QOS 0 as it is the least costly in terms of setup, and is adequate for the purposes of our application.

The Cloud

Our Model is trained and deployed on a single g4dn.2xlarge instance, provisioned on Amazon Web Services. Deploying the model on the cloud rather than the edge complicates the overall pipeline and necessitates stream of information across multiple devices, but provides space for larger and more accurate models. Cloud deployment also allows the application to be served across relatively weak or small edge devices.

Once images are forwarded as binary messages from the edge device, they are received in a MQTT Docker cluster, decoded in a Docker MQTT listener cluster, and temporarily staged within the instance. Once all of the messages have been received, inference is run using the deployed model. The user will receive output with the predicted classification of their lesion type (Actinic keratoses and intraepithelial carcinoma, Basal cell carcinoma, Benign keratosis-like lesions, Dermatofibroma, Melanoma, Melanocytic nevi, Vascular lesions) as well as a classification of benign vs. malignant.

Pipeline Limitations

Without a custom-built user interface, use of the application is limited to a user uploading images to our Jetson Nano, spinning up kubernetes clusters on the edge device, and then receiving output in the cloud following deployment of docker clusters.

With additional resources, the student group would like to focus on lesion-detection software that would enable real-time classification of lesions via webcam, similar to frameworks like YOLO. Additionally, in an ideal state inference would be passed back from the Cloud to the Edge directly. Instead, our current infrastructure would be able to support inference output on a web server or similar deployment - hosted in the cloud. The addition of docker clusters and more processes on the cloud side would also likely necessitate kubernetes.

Instructions for pipeline setup are included in the git repository [5].

4. Dataset

Data

The HAM10000 dataset consists of 10,000 dermoscopic lesion images [4]. The seven classes of skin lesion diseases are listed below, with brief definitions [6].

- **Benign keratosis-like lesions** (bkl): generally from keratin growth on skin. Usually difficult to differentiate from melanoma, so commonly biopsied
- **Dermatofibroma** (df): inflammatory reactions to minimal trauma
- **Melanocytic nevi** (nv): abnormal mass of tissue that forms from cell growth, commonly symmetric in color and structure
- **Vascular lesions** (vasc): benign tumors that result from overgrowth of capillaries
- **Actinic keratoses and intraepithelial carcinoma** (akiec): non-invasive variants of squamous cell carcinoma
- **Basal cell carcinoma** (bcc): variant of epithelial skin cancer that will grow destructively if untreated
- **Melanoma** (mel): malignant neoplasm that can be invasive or non-invasive.

The first four types of skin lesions are considered “benign” - non-cancerous skin growths [7], whereas actinic keratoses, basal cell carcinoma, and melanoma are considered “malignant” or cancerous skin growths.

During construction of the dataset, diagnoses for each lesion were obtained in one of four ways [6]. The most trusted methodology - histopathology - was used to confirm over 50% of the lesions in the dataset. This method involves molecular observation and diagnosis of excised lesions and is only accepted as ground truth in cases without any ambiguity. The second-most-common diagnosis methodology was follow-up, where specialists monitored lesions using digital dermatoscopy. In these cases - a lesion would be classified as nevi following no changes to the abnormality over three visits, or one and a half years. Confocal and consensus methodology accounts for less than 10% of all lesion images, where labels are assigned based on visual examination, only in cases assumed to be benign.

In addition to the 10,000 images, metadata were also included for each lesion, including age, sex, and lesion localization. During model development, our group focused training solely on images. This decision was based on the desire to provide an application that would accurately infer skin cancer types without requiring additional sensitive information from users.

Within the 10,000 observations, roughly 2,500 images were originally believed to be duplicates. Upon further investigation, these images were identified as the same lesion, with photos taken with different resolution, angles, or transformed in some other way. Because these images contained unique and additional information, they were not removed from the dataset. To ensure validation accuracy was not boosted by these partial duplicates, each lesion_id associated with multiple images was quarantined, and assigned to the train dataset during the train/test split.

Figure 7 highlights major class imbalance across the dataset, which provides risk to accurate classification of lesions. Techniques like oversampling were used to reduce this risk, and are discussed further in the Final Model section.

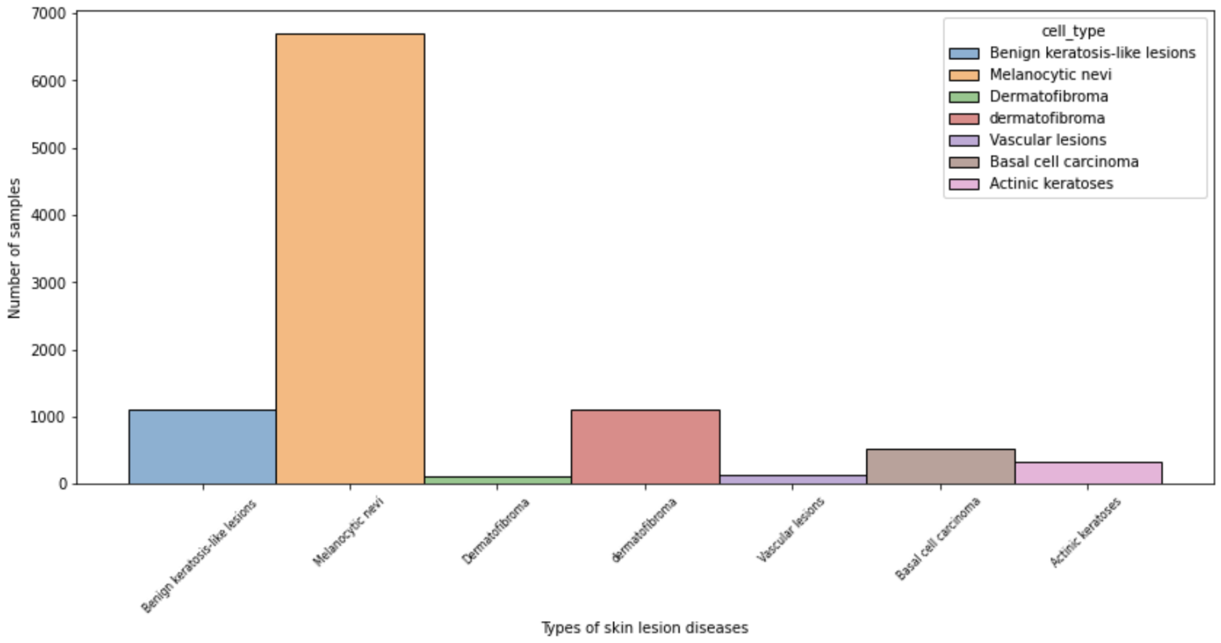
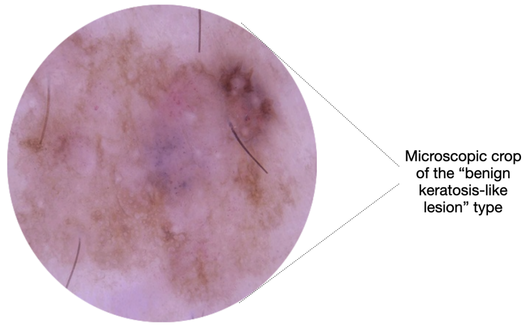


Figure 3. Histogram of all classes in the overall images from ISIC 2018 dataset before the train-test split

Data Augmentation, Transformation, and Processing

Our dataset consists of 8000 images used for training and around 2000 images for validation from all seven categories. All lesion images in the dataset are pre-centered at 800x600 pixels. The data in the dataset consists of images at multiple resolutions, as well as a substantial class imbalance making this a challenging dataset to classify correctly. For training and testing, we needed specific masks that show the presence or absence of clinically-meaningful skin lesion patterns, medically referred to as pigment networks, negative networks, streaks, cysts, and globules.

Augmentation (enlarging focus areas)



As part of data augmentation, we have developed a custom code to crop the images to bring focus to the specific patterns usually seen in benign or malignant skin lesion types. Shown in Figure 4 is one such sample image cropped from the original ISIC dataset to enlarge the hidden information to be easily visible to the ML algorithm.

Figure 4. Microscopic crop of the "benign keratosis-like" lesion type image from the original ISIC 2018 dataset.

Synthetic data augmentation using GAN

DCGAN (Deep Convolutional Generative Adversarial Network) is one of the most employed GAN architectures. Since most classes have far fewer images than the largest class (Figure 7), in addition to cropping techniques, we have utilized GAN to generate synthetic images entirely from noise, with no connection to the sampled actual image, except the discriminator guiding the generator to reproduce the correct image.

A GAN works by training two networks simultaneously. The first, Generator network, produces synthetic images (typically by learning the underlying data distribution). The second network is referred to as the Discriminator, with the objective of estimating the probability that an input sample is synthetic (i.e., it came from a generator) or real.

Though the output image lacks cohesion and cannot be compared at a pixel level, it can still offer information to compare the sharpness and presence of malignancy markers and their fine-grained details. Furthermore, the augmented images improved the class balance ratio, resulting in increases to F1-score, precision, and recall metrics (Table 1).

The following is a example Basal Cell Carcinoma skin lesion generated by GAN from complete noise:

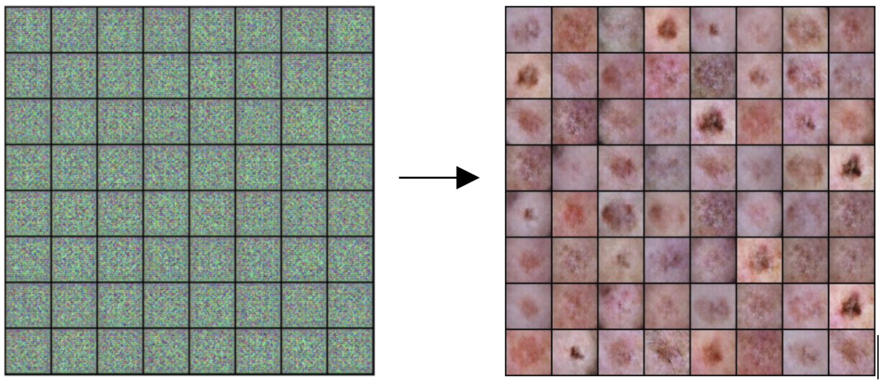


Figure 5: GAN generated “basal cell carcinoma” skin lesion shown from a Pytorch tensor object

Class	Precision	Recall	F1
AKIEC	92%	86%	89%
BCC	89%	91%	90%
BKL	71%	75%	73%
DF	98%	92%	95%
NV	95%	88%	91%
VASC	94%	98%	96%
MEL	54%	78%	64%

Table 1: Precision, Recall, and F1-Score for all the seven classes. Results obtained post GAN based augmentation.

5. Training

Convolutional neural networks (CNNs) have superior performance in image classification [8]. CNNs have three main layers, where the convolutional layers extract information from the image. The earlier convolutional layers focus on simple features such as colors and edges, while the later layers recognize the larger elements or shapes.

As a part of classifying the HAM10000 Dataset, we took pre-trained models ResNet, Densenet, VGG, & EfficientNet, and fine tuned them in PyTorch. While training, a high validation accuracy was observed, but the f1 score for the minority classes was quite low.

Our baseline model was established by using the Densnet121 model with Adam optimizer and Cross Entropy Loss. The accuracy on the test dataset was 65%. The baseline was run for 10 epochs. Our original code base was borrowed from Xinrui Zhuang’s kaggle submission [9], and modified in both model architecture and hyperparameters in order to increase accuracy. Zhuang’s submission was selected as our baseline due to their use of Pytorch, and a strong similarity to other neural network frameworks utilized throughout the course.

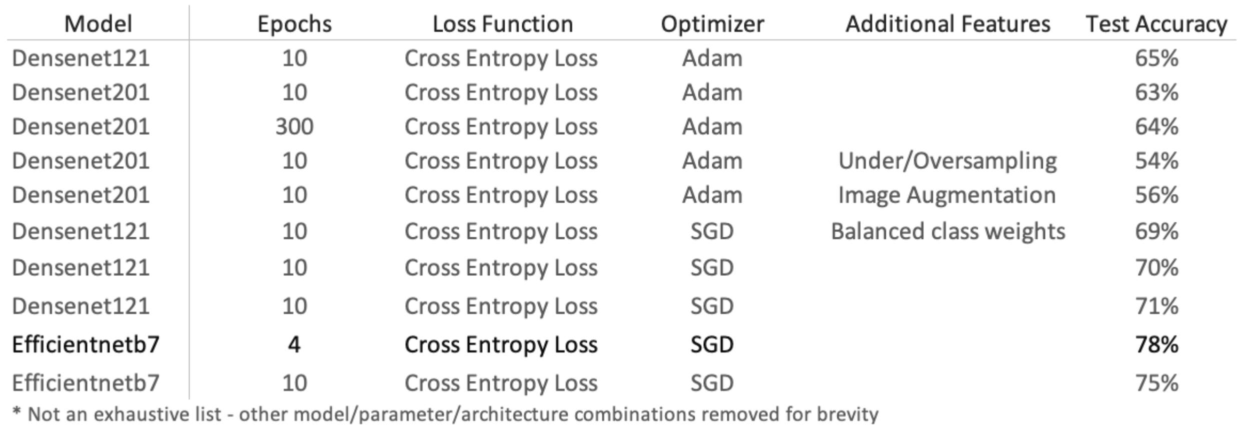
6. Tuning

While Densenet121 provided a good starting point and baseline, the results were poor for the minority classes. Even with a validation accuracy of around 90%, the test accuracy was 65%. This led to an exploratory effort to tune the baseline model.

Tuning the model consisted of:

1. Trying different models: ResNet, VGG, Densenet, & EfficientNet
2. Trying different loss functions: CrossEntropyLoss, FocalLoss
3. Trying different optimizers: Adam, SGD
4. Tried running for higher epochs: This varied from run to run and the highest was 300.
5. Tried random (over & under sampling) for classes
6. Image augmentation

The below table summarizes the results achieved from each of the runs:



Model	Epochs	Loss Function	Optimizer	Additional Features	Test Accuracy
Densenet121	10	Cross Entropy Loss	Adam		65%
Densenet201	10	Cross Entropy Loss	Adam		63%
Densenet201	300	Cross Entropy Loss	Adam		64%
Densenet201	10	Cross Entropy Loss	Adam	Under/Oversampling	54%
Densenet201	10	Cross Entropy Loss	Adam	Image Augmentation	56%
Densenet121	10	Cross Entropy Loss	SGD	Balanced class weights	69%
Densenet121	10	Cross Entropy Loss	SGD		70%
Densenet121	10	Cross Entropy Loss	SGD		71%
Efficientnetb7	**4**	**Cross Entropy Loss**	**SGD**		**78%**
Efficientnetb7	10	Cross Entropy Loss	SGD		75%

* Not an exhaustive list - other model/parameter/architecture combinations removed for brevity

Table 2: Test Accuracy across different Model, Loss function, Optimizer, and Hyperparameter combination trials

7. Final Model

Based on overall accuracy as well as balance of F1 score across all seven classes, we chose to deploy the EfficientNetB7 model trained over 4 epochs as our final inference base. Each F1 score over the lesion types was in the range of .8 or higher, which is noticeably an improvement from our baseline, where F1s ranged from .5 to .96 (Table 3).

Densenet is a lighter model compared to efficientnet which learns by connecting the output of each layer to every other layer in a feedforward fashion. For each layer the feature maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs to all subsequent layers. This resulted in a high model accuracy and perhaps could have been the best result if a balanced dataset was available.

EfficientNetB7 is a much heavier model than densenet or other tested architectures, but provided the biggest return in accuracy based on its ability to dynamically scale upward in three different dimensions [10]. Unlike other model architectures, EfficientNet was designed to scale depth, resolution, and channels in a way that “balances all dimensions of the network”. This scaling technique reduces the risk of vanishing gradients, as well as saturation of layer activation.

Combining EfficientNetB7's scaling capabilities with higher image resolution and oversampling techniques resulted in an accuracy of 78.3% for the deployed model. Additionally the F1 score for the minority classes improved on average by 10%. Figure 6 shows the class size before and after oversampling.

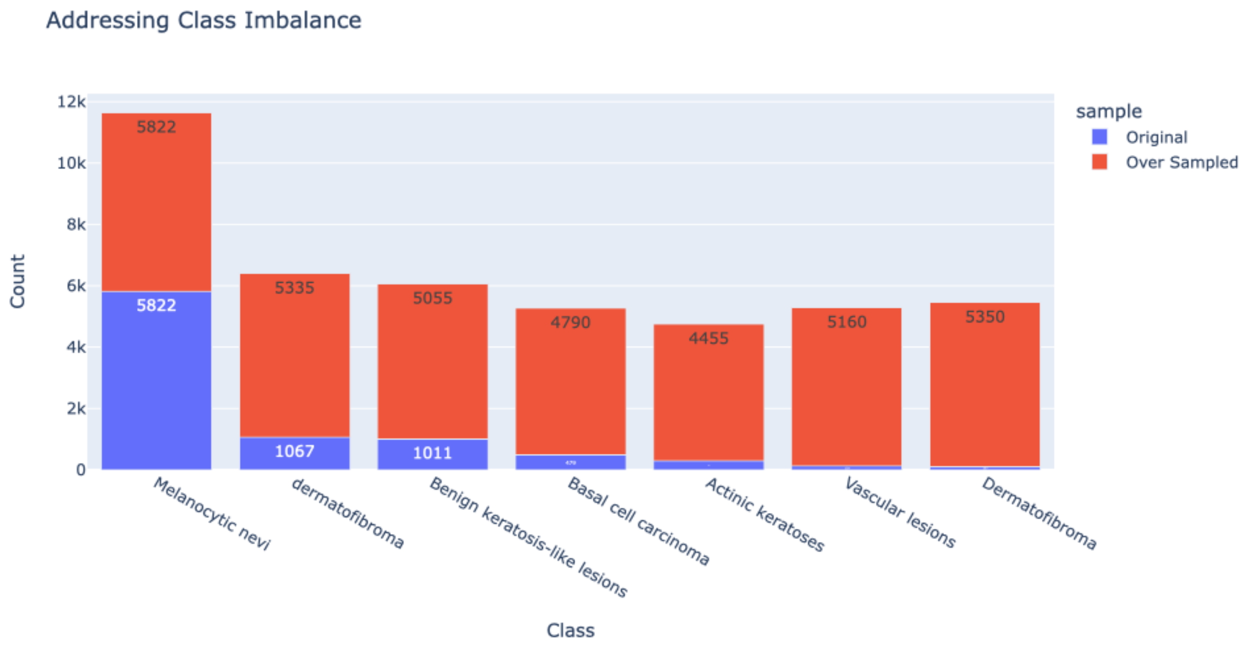


Figure 6: Class imbalance fixed by oversampling

8. Results

Our final application is able to read in images from an edge device, transmit to the cloud, and provide users with lesion classification within one second. Across 10 randomly selected lesion images from the internet, our model properly classified lesion type 4 times, and cancerous/non-cancerous 7 times (Figure 7).

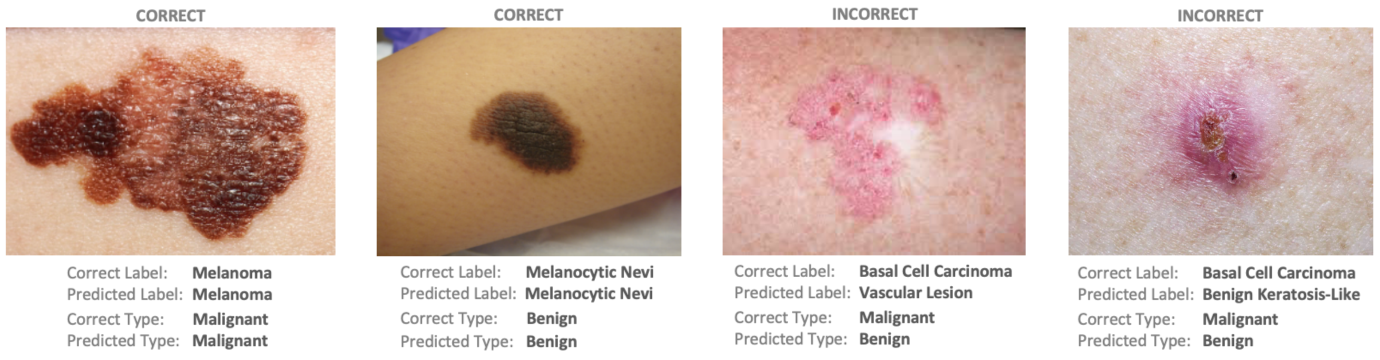

CORRECT
Correct Label: Melanoma
Predicted Label: Melanoma
Correct Type: Malignant
Predicted Type: Malignant
CORRECT
Correct Label: Melanocytic Nevi
Predicted Label: Melanocytic Nevi
Correct Type: Benign
Predicted Type: Benign
INCORRECT
Correct Label: Basal Cell Carcinoma
Predicted Label: Vascular Lesion
Correct Type: Malignant
Predicted Type: Benign
INCORRECT
Correct Label: Basal Cell Carcinoma
Predicted Label: Benign Keratosis-Like
Correct Type: Malignant
Predicted Type: Benign

Figure 7: Examples of two correctly classified, and two incorrectly classified lesions

Table 3 (below) shows the precision, recall and F-1 scores per class between the baseline model and the deployed model. The accuracy improved by 13% and the F1 Score for the minority classes improved by ~10%. Examples of correctly and incorrectly classified lesions are included in Figure 7.

	Precision		Recall		F1		Accuracy		
Class	Base	Final	Base	Final	Base	Final	Base	Final	Support
AKIEC	65%	**88%**	67%	**77%**	66%	**82%**			30
BCC	76%	**77%**	91%	**94%**	83%	**85%**			35
BKL	74%	**82%**	73%	**64%**	74%	**72%**			88
DF	71%	**70%**	62%	**88%**	67%	**78%**	65%	**78%**	8
NV	98%	**97%**	93%	**97%**	95%	**97%**			883
VASC	90%	**75%**	69%	**92%**	78%	**83%**			13
MEL	32%	**47%**	59%	**57%**	41%	**51%**			46

Table 3: Precision, Recall, and F1-Score for all the seven classes

9. Next steps

No model is perfect and there is always more work which can be done to improve the system. There were multiple ideas which would be interesting to try in the upcoming phases for this project. The GAN images were not used for training and it would be interesting to observe their effect and compare it with oversampling techniques used. It might also be useful to explore external data sources for images and use them to train. Another improvement would be to prune the EfficientNet model & make it lighter. Finally, in this end to end pipeline everytime the model predicts incorrectly, a trained professional such as a Dermatologist should have a user interface available to correct the model for continued training.

10. References

[1] https://www.cdc.gov/media/releases/2014/p1110-skin-cancer.html
[2] https://www.webmd.com/melanoma-skin-cancer/melanoma-guide/causes-skin-cancer#1
[3] https://www.cancer.net/cancer-types/melanoma/statistics
[4] https://challenge.isic-archive.com/landing/2018/
[5] https://github.com/abajaj25/MNIST-Skin-Cancer-with-Jetson/tree/main
[6] https://www.nature.com/articles/sdata2018161
[7] https://dermnetnz.org/topics/benign-skin-lesions
[8] https://www.ibm.com/cloud/learn/convolutional-neural-networks
[9] https://www.kaggle.com/xinruizhuang/skin-lesion-classification-acc-90-pytorch
[10]
https://cv-tricks.com/network-architecture/efficientnets-the-free-lunch-of-2019-for-convolutional-neural-networks/