Classifying Skin Lesions from Images

Team members: Mohamed Elghetany, Mike Khor, Megan Martin
MIDS 207 Section 7

Problem Statement

According to the International Skin Imaging Collaboration (ISIC) [1], “melanoma [...] is responsible for 75% of skin cancer deaths, despite being the least common skin cancer.” In 2020, 57,043 people died from melanoma worldwide [2]. Early and accurate diagnosis of melanoma can allow patients to start treatment (surgery, radiation therapy, chemotherapy) early, and can save lives.Identifying cancerous skin lesions relies heavily on visual indicators such as size, shape, color, and uniformity. Images of suspicious skin lesions are often utilized as part of the diagnostic evaluation to determine if biopsies are recommended.

An image-based multiclass classification model for predicting the presence of melanoma and other skin cancers can be useful for experts (dermatologists) and even non-experts (other physicians, oncologists). For non-experts, a model could help to flag patients with suspicious lesions for referral that would otherwise be overlooked. For experts, a trained model can be used as a confirmation tool or a way to quickly identify malignant vs. benign lesions, reducing the need for skin biopsies of false positive lesions.

Objective

Train a model to classify different types of skin lesions, including cancerous and benign lesions, from dermoscopic images.

Datasets

The International Skin Imaging Collaboration (ISIC) is an international effort to improve melanoma diagnosis [3]. The ISCI archive contains a collection of high-quality dermascopic images for a variety of skin lesions. Dermascopy eliminates surface reflection of the skin, resulting in enhanced visualization of deeper levels of skin compared to standard photography. Most images have metadata for the patient, which includes age, gender, and general location of the lesion on the body.

For this project, the aggregate ISIC 2019 image dataset was utilized [4, 5, 6]. This dataset contains demoscopic images for eight lesion types. Three of these eight classes are cancerous lesions: basal cell carcinoma (BCC), melanoma (MEL), squamous cell carcinoma (SCC) while the remaining 5 classes are benign: melanocytic nevi (NV), benign keratosis (BKL), actinic keratosis (AK), vascular lesion (VASC), dermatofibroma (DF). A total of 25,331 images with ground truth classification for these eight lesion types are included in the available train dataset. Along with images, a CSV containing metadata for each image is available which include patient age, sex, and general anatomic site.

Exploratory Data Analysis:

Although the total image dataset is quite large, it represents highly unbalanced classes. The largest class NV which represents benign moles is ~54 times larger than the smallest class dermatofibroma (DF):

MEL	NV	BCC	AK	BKL	DF	VASC	SCC
4522.0	12875.0	3323.0	867.0	2624.0	239.0	253.0	628.0

Images are sourced from various methods of dermoscopy, which results in images of various sizes.

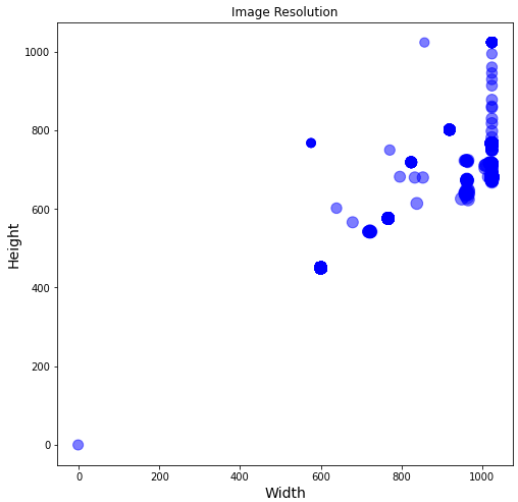


Figure 1. Distribution of image sizes.

Image size will be standardized for use in models. Image size distribution is displayed in **Figure 1** and represents images ranging from (600 x 450; 10,014 images) to (1024 x 1024; 12,410 images).

In addition to a distribution of sizes, as these images are sourced from real-world patients, images may have rulers, pen markings around the lesion, and microscopic "field of view" resulting in a circle around the image. Examples images in each class are shown in **Figure 2**.

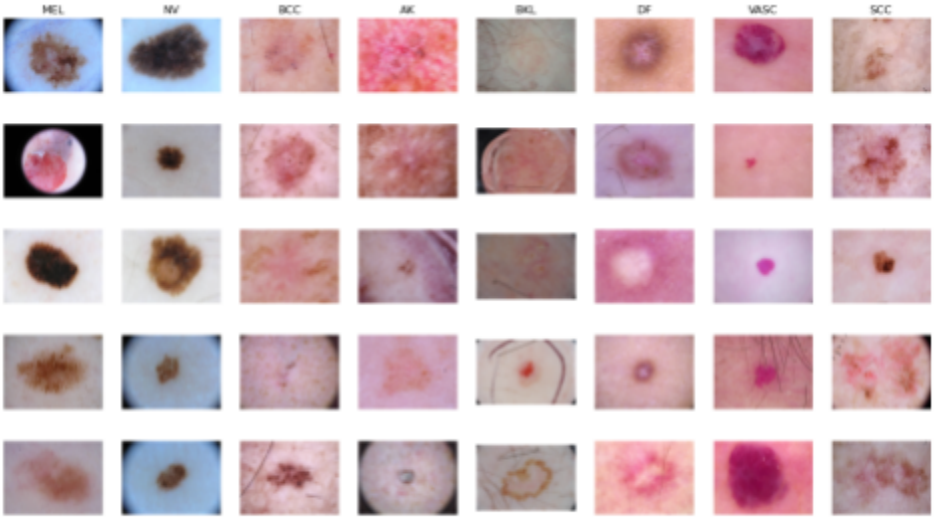


Figure 2. Five examples from each class of images.

Metadata consists of image ID, approximate age (which has been binned by 5 years already), general anatomic site, lesion ID, and patient sex. For purposes of analysis, lesion ID was not utilized. Note that there was a large number of missing values for each metadata field in the train set as shown in **Figure 3**. A total of 21,311 images had complete metadata fields.

Figure 3. Count of missing metadata

image	0
age_approx	437
anatom_site_general	2631
lesion_id	2084
sex	384

As part of the EDA process, we verified that there were no duplicate image IDs and no duplicate lesion IDs in the dataset.

Methodology

After the EDA process, a number of challenges were identified with the variability observed in the image dataset. The approach to mitigate these challenges are described below.

Image Cropping and resizing:

A number of dermoscopic images (6,226 out of 25,331; 25% of all images) contain a large portion of black region around a circular view of the skin lesion, which can add unwanted noise when training our model. To mitigate this, a cropping method was utilized on the training image set. For each image, a calculated RGB value difference between regions in the corners vs. the center of the image was utilized to identify images with black regions needing to be cropped (so that we don't apply cropping algorithm to images that don't have black outer circle). A masking method was then utilized to identify the area to be cropped, with a final trimming to result in a cropped image displaying the majority of the lesion. This process is displayed in **Figure 4**. After cropping, images were resized to 224 x 224 for uniformity.

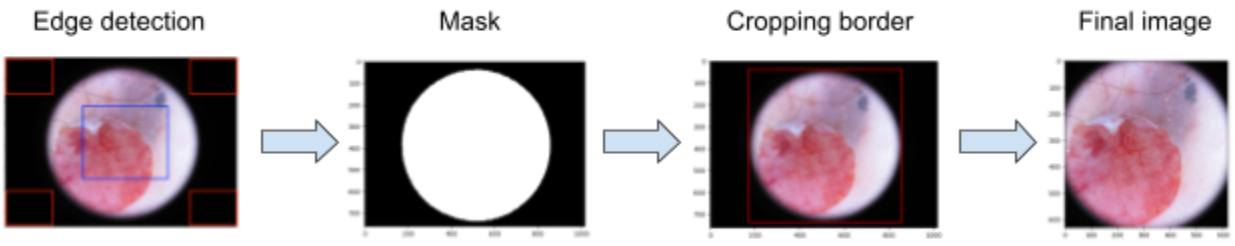


Figure 4. Cropping algorithm steps.

Class imbalance:
As discussed above, the eight different skin lesion classes were highly imbalanced. In order to improve the model fairness, a method to downsample or augment was utilized to bring each class image count to 2,000. For the majority classes (>2,000 images in training set), undersampling was utilized to bring the images down to 2,000. For minority classes (<2,000 images in training set), randomized augmentation was utilized to bring up each class to 2,000. Augmentation methods included random flip up, down, left and right, random rotation, random saturation, random contrast, and random brightness changes as seen in **Figure 5**. The smallest class (DF) required an 8.4 fold increase in images. We will describe subsequent mentions of this type of downsampled/augmented dataset as "*augmented-balanced*".

80% of images were subset for training and validation purposes while 20% was utilized for testing purposes. Of the former 80%, 8% is set aside for validation, and 72% is used for training which is the only set that gets the "augmented-balanced" procedure.

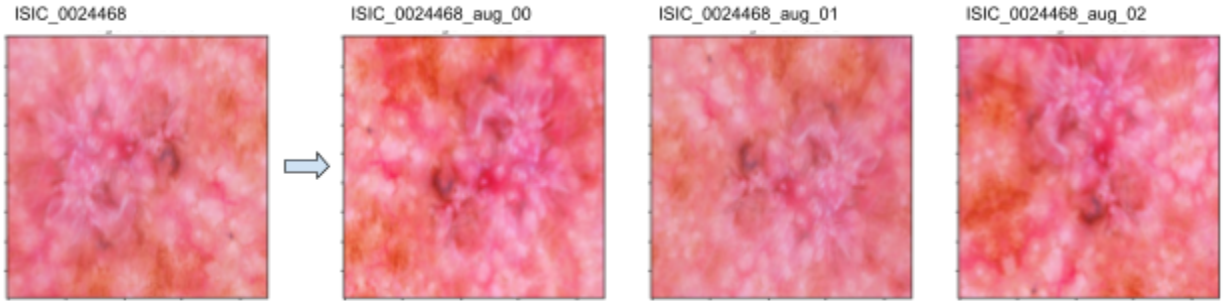


Figure 5. Example of an image (class: AK) augmented 3 times.

Models
Models were selected for multiclass classification. During model evaluation, a smaller image dataset was utilized representing 250 images per class (2,000 images total, augmented- balanced) to reduce computational burden during training. For well-performing models, training utilizing the 16,000 augmented-balanced dataset was performed as well. The following standard models were tested on the small dataset:

1. Logistic regression
2. K-nearest neighbors (KNN)
3. Random Forest
4. Convolutional neural network (CNN)

A simple CNN setup is made as an initial proof of effectiveness. Input image is rescaled to 224 x 224 x 3, then piped into 2 sets of convolutional layers (2-D convolution with 16 filters, kernel size of 3, ReLU activation, with padding; max pooling with pool size of 2, stride of 2; dropout of 0.1, batch normalization). After convolution layers, the output is flattened, then piped into 2 sets of dense layers (a dense layer with 10 units each, ReLU activation; dropout on 0.1, batch normalization). The output of dense layers is sent to a final classification dense layer (8 units for the number of classes; softmax activation).

After demonstrating performance of the CNN model, additional models were built utilizing the larger dataset:
5. CNN + hyperparameter tuning using Optuna:
Instead of using common hyperparameter tuning methods like scikit-learn's grid search (computationally

expensive) and randomized search, we opted for using Optuna, which is an open-source hyperparameter optimization package that is framework agnostic, and works great with Tensorflow/Keras deep learning structure [8]. Optuna has a clean API that manages studies and trials, and automatically randomizes hyperparameters with a custom pruning method (stop epochs) to save computation on training (we did not use their pruning feature). However, the most relevant advantage to CNN is that it allows for the optimization of individual layers' hyperparameters, while dynamically optimizing for the number of layers. Our optimization of our CNN model involves 8 types of hyperparameters, as highlighted in **Figure 6.**

Figure 6. Hyperparameter tuning of CNN structure

1. ***A*** sets of: (options: 2, 3)
   - Conv2D with:
     - ***B*** filters (options: 5 to 16)
     - ***C*** kernel size (options: 3 to 5)
     - ***D*** setting of padding/no padding (options: True/False)
     - activation: relu
   - MaxPooling
     - ***E*** pool size (options: 2, 3)
     - ***F*** stride (options: 2, 3)
   - Dropout: 0.1
2. Flatten
3. ***G*** layers of Dense (options: 0 to 3)
   - ***H*** units (options: 5 to 20)
   - Activation: relu
   - Dropout: 0.1
4. One output layer of Dense
   - output units: 8
   - Activation: softmax

6. CNN + transfer learning using using EfficientNet:

EfficientNet is a family of models created by Google AI scientists that efficiently scales up CNNs for image classification using compound scaling (simultaneously scale up width, depth, and resolution of model) [9]. When trained on ImageNet, the developers demonstrated higher accuracy in EfficientNet models over well-known model structures like ResNet and DenseNet families (when comparing with similar model sizes).

There are a total of 8 model types, numbered from B0 to B7, with increasing number of parameters. Tensorflow-keras API contains methods for instantiating EfficientNet models, and additionally gives access to model weights pre-trained by its developers on ImageNet. This gives a good opportunity for transfer learning, where we skip expensive training (traditionally from randomly initialized weights). We used EfficientNet's ImageNet weights (convolution filters) for our skin lesion images as a way to extract features (Conv2D/MaxPooling outputs) in our images, before passing the outputs into a traditional set of dense layers. As recommended by Keras [12], transfer learning training is split into a normal training part, and a fine-tuning part. During normal training, the transferred model and weights are frozen, and a normal learning rate is applied. During fine-tuning, the weights are unfrozen (made trainable) and a very low learning rate is applied, so that pretrained weights can adapt to our new dataset without being destroyed.

Each EfficientNet model has a different number of parameters (B0: ~4 million; up to B7: ~65 million). Although the developers demonstrated higher accuracy with more parameters, we tested all 8 models on our skin lesion data to compare accuracies before committing to a particular model for expensive fine-tuning. During this comparison, we kept the post-EfficientNet dense layers constant (2 dense layers, 10 units each, with dropout and batch normalization) to keep models comparable.

7. CNN + Optuna + EfficientNet

Here, we combine transfer learning with hyperparameter tuning. Using knowledge on which EfficientNet model type (B0 through B7) has the best performance (highest AUC-ROC), we continue to optimize the dense layers between EfficientNet convolutions and output classification layer via the Optuna hyperparameter tuning package.

During hyperparameter tuning trials, the EfficientNet weights are frozen to prevent the destruction of weights during dense layer training (which uses large learning rates). Once the best dense layer structure is identified, we continue onto fine-tuning of EfficientNet (see *Standards* section).

8. CNN + XGBoost

XGBoost (“Extreme Gradient Boosting”) is a popular library developed by Distributed Machine Learning Common that supports parallel tree boosting and efficient gradient descent optimization using GPU computation [10]. However, supplying raw image pixel-based data to XGBoost might not provide informative features. A way to mitigate this issue for image classification is to use CNN as a “feature extractor,” which then provides XGBoost contextualized data [11]. We attempted a combined CNN + XGBoost model using the best CNN model trained up to this point, and then replacing the CNN model’s output layer (softmax classifier) with XGBoost.

Additional models utilizing the metadata will be utilized:

9. CNN + EfficientNet + metadata (combining image data and structured metadata in a model)

We attempt to improve our best CNN (EfficientNet) model by contextualizing image data with metadata (age, sex, anatomical site). The metadata input is first encoded into categorical data (age gets binned into groups of 5 years; NA data gets its own category; input shape: 31), and then passed through several dense layers. The best trained CNN is transferred over (without top dense layers) and its weights are frozen. The outputs from both sequences are concatenated, and then more dense layers are added before the final classification layer. An example of this is shown in **Figure 7**.

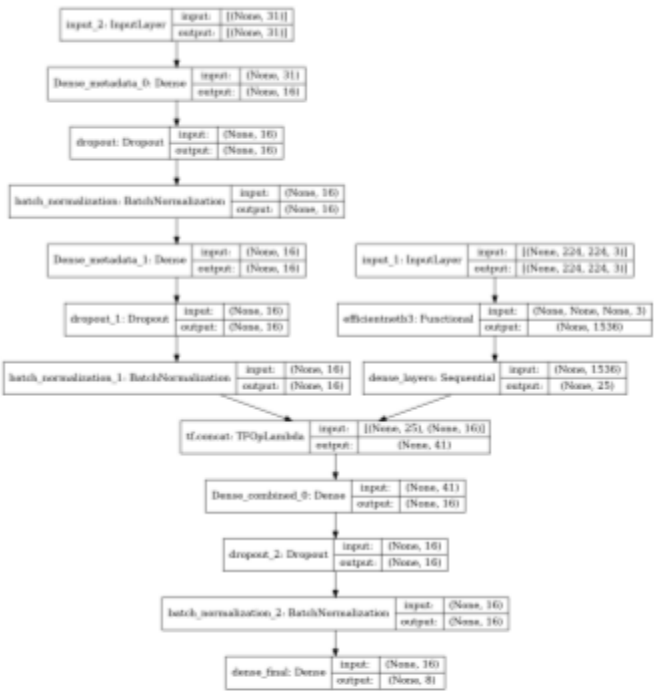


Figure 7. Example structure of model using metadata input (normal dense layers; left) and image data input (CNN; right), which gets concatenated and put through a dense layer before classification.

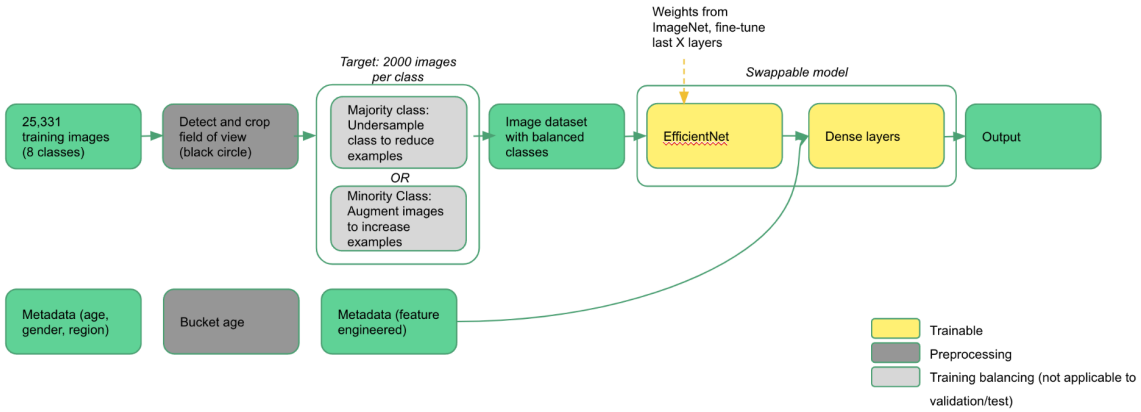
10. Metadata clustering using K-means and K-prototypes

Our metadata contains a mix of numerical and categorical data. The sample space for categorical data is discrete and doesn't have a natural origin, making standard k-means difficult. Applying standard K-Means on the only numerical feature (Age) won’t provide useful insights so an alternate clustering method that can include categorical data will also be utilized. .

K-Means + Hot-Encoding Categorical data: We transformed our categorical features into numerical using One-Hot-Encoding, then we used Standard K-Means

K-Prototype is an algorithm which combines k-means and k-modes. We chose k-prototypes because it can handle both numerical and categorical features simultaneously.

Block Diagram



What is considered Success/Failure?

Model performance for the ISIC image dataset is well-documented as challenging. Success will be a model that performs significantly better than the baseline logistic regression. We are not benchmarking our trained model against dermatologists (which would require data on dermatologist predictions using the same images). Furthermore, dermatologists achieve 86.6% sensitivity (high true positive identification) [7], which has proven difficult to meet or outperform from models alone.

Evaluation Parameters

Accuracy, recall, precision, and ROC AUC score will be used to evaluate each model. Based on similar models published, performance expectations are moderate. For the purposes of this project, an accuracy score greater than 55%, recall score greater than 60%, a precision score greater than 50%, and an ROC AUC score greater than 60% will be deemed successful.

Model Results

1. Logistic regression

Experiment: Training data was loaded for 250 images for each class (augmented-balanced; augmentation on DF and VASC) and scaled. Logistic regression was performed utilizing TensorFlow/Keras with one flatten layer, one dense layer with softmax activation, and 100 epochs. Adam optimizer was utilized, with categorical_crossentropy loss specified.

Results/Graphs: Validation Loss and Accuracy was unstable throughout the 100 Epochs. Overall, performance metrics did not meet success thresholds as seen in Figure 8.

Logistic Regression Classification Report

	precision	recall	f1-score	support
MEL	0.464	0.153	0.230	895
NV	0.708	0.788	0.746	2592
BCC	0.000	0.000	0.000	684
AK	0.000	0.000	0.000	160
BKL	0.243	0.054	0.088	502
DF	0.000	0.000	0.000	44
VASC	0.500	0.170	0.254	47
SCC	0.064	0.782	0.118	142
accuracy			0.459	5066
macro avg	0.247	0.243	0.180	5066
weighted avg	0.475	0.459	0.437	5066

Accuracy: 0.4591
ROC AUC: 0.5760

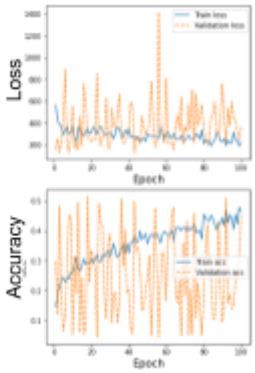
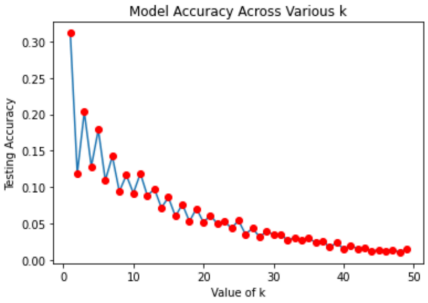


Figure 8. Logistic Regression Results

2. K-nearest neighbors (KNN)

Experiment: Training data was loaded for 250 images for each class (augmented-balanced). The KNeighborsClassifier was utilized from scikit learn. Optimization of K was tested by fitting the model on the training data across K= 1 to K = 49. Based on accuracy, K = 3 was utilized (see **Figure 9**).

Figure 9. KNN results

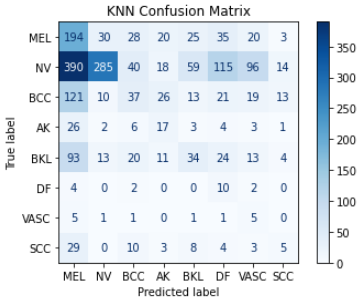



```
KNN Classification Report
              precision    recall  f1-score   support

         MEL       0.540     0.076     0.133       355
          NV       0.836     0.280     0.420      1017
         BCC       0.257     0.142     0.183       260
          AK       0.179     0.274     0.217        62
         BKL       0.238     0.160     0.192       212
          DF       0.047     0.556     0.086        18
        VASC       0.031     0.357     0.057        14
         SCC       0.125     0.081     0.098        62

   micro avg       0.354     0.210     0.263      2000
   macro avg       0.282     0.241     0.173      2000
weighted avg       0.590     0.210     0.292      2000
 samples avg       0.210     0.210     0.210      2000

Accuracy:  0.21
ROC AUC:  0.5933327707205982
```



Results/Graphs: Performance of KNN on the test images was poor, despite the attempt to optimize K. Based on the confusion matrix, this model is inaccurately predicting benign NV as melanoma. For clinical use, this is a poor performing model as this could lead to a lot of false positives and unnecessary patient biopsies.

3. Random Forest

Experiment: Training data was loaded for 250 images for each class (augmented-balanced; augmentation on DF and VASC). RandomForestClassifier was utilized from scikit learn. The default number of trees (n_estimators) in the forest is 100. Based on performance at 100 n_estimators, additional fine-tuning was performed using the validation set to identify n_estimators = 3 as the most reasonable performance as seen in **Figure 10**. Other than adjusting the n_estimators parameter, default parameters from sklearn RandomForestClassifier class were utilized.

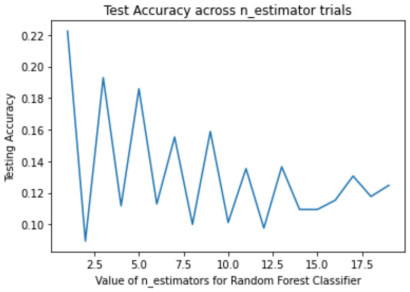


Figure 10. Random forest n_estimators tuning

Results/Graphs: Performance of the Random Forest model on the images was poor, despite attempts to select better-performing n_estimators as seen in **Figure 11**. Additionally, the model is incorrectly predicting benign moles class (NV) as melanoma. For clinical use, this is a poor performing model as this could lead to a lot of false positives and unnecessary patient biopsies.

```
Random Forest Classification Report
              precision    recall  f1-score   support

         MEL       0.336     0.164     0.221       895
          NV       0.832     0.281     0.420      2592
         BCC       0.295     0.094     0.142       684
          AK       0.132     0.169     0.148       160
         BKL       0.138     0.076     0.098       502
          DF       0.043     0.205     0.071        44
        VASC       0.062     0.319     0.105        47
         SCC       0.107     0.141     0.122       142

   micro avg       0.396     0.207     0.272      5066
   macro avg       0.243     0.181     0.166      5066
weighted avg       0.547     0.207     0.293      5066
 samples avg       0.207     0.207     0.207      5066

Accuracy:  0.20706671930517173
ROC AUC:  0.5672968987988957
```

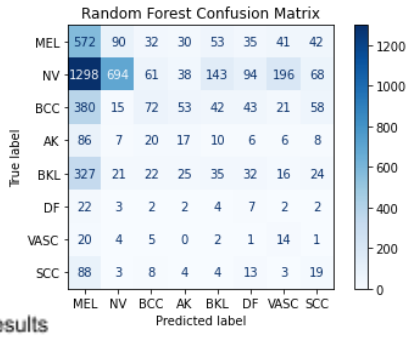


Figure 11. Random forest results

4. Convolutional neural network (CNN)

Experiment: Training data consists of 2000 images (250 per class, augmented-balanced; batch size: 32). Training is configured with Adam optimizer (1e-3 initial learning rate) to minimize categorical crossentropy loss. The model is trained for 100 epochs.

Results/Graphs: The simple CNN model performs slightly worse than random forest

```
CNN Classification Report
              precision    recall  f1-score   support

         MEL       0.187     0.254     0.215       895
          NV       0.513     0.185     0.272      2592
         BCC       0.133     0.325     0.189       684
          AK       0.007     0.006     0.007       160
         BKL       0.137     0.040     0.062       502
          DF       0.000     0.000     0.000        44
        VASC       0.006     0.106     0.011        47
         SCC       0.000     0.000     0.000       142

    accuracy                          0.189      5066
   macro avg       0.123     0.114     0.094      5066
weighted avg       0.327     0.189     0.209      5066

Accuracy: 0.1885
ROC AUC: 0.4760
```

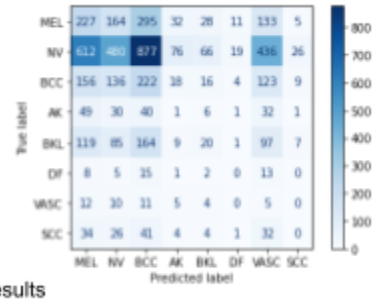


Figure 12. CNN results

(weighted average scores) in accuracy, precision, recall, ROC-AUC as seen in **Figure 12**. When analyzing the confusion matrix, although the images (all classes) are less likely to be misclassified as MEL, it appears that images have a tendency to be classified as NV, AK, or SCC, regardless of its true label.

Note that a plot of all training history of CNN-related models are available in the *Comparison of models* section.

5. CNN + hyperparameter tuning using Optuna

Experiment: The Optuna package is first tried using a small dataset of 2000 augmented-balanced images, before moving on to a larger augmented-balanced dataset of 16,000 images. An Optuna *study* is set up to run 100 *trials*, where each trial configures a different configuration of a CNN model (8 hyperparameters; see **Figure 13** for details). Similar to simple CNN, training is performed with categorical crossentropy loss, and Adam optimizer. Additionally, an early stopping callback is used during training, which stops training when a specified performance metric does not improve (we chose AUC-ROC on validation data as our metric to track). Even with max epoch set at 100, most Optuna trials stop early (some stop 20 epochs in), which saves computation time.

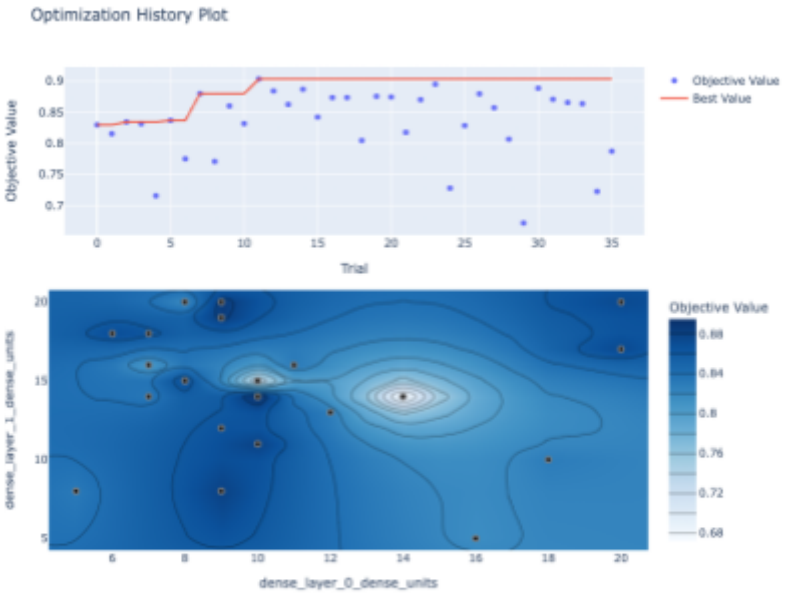


Figure 13. (top) history of AUC-ROC over the number of hyperparameter tuning trials, and the best value up until that point in time; (bottom) contour plot of average AUC-ROC plotted with number of units in the first two dense layers.

Results/Graphs: The Optuna study (training on large dataset of 16,000 images) is manually stopped at 37 trials (instead of 100) because it's taking too long to train. For the dense layers after convolution, the objective function (AUC-ROC) gives a complex response to number of dense units, with no clear optimal configuration (Figure X). Ultimately, the final hyperparameter-tuned model has the following properties: 3 convolutional layers - {layer 0: 11 filters, kernel size 3, pool size 3, strides 3; layer 1: 14 filters, kernel size 5, pool size 3, strides 2; layer 2: 11 filters, kernel size 4, pool size 3, strides 3}, 1 dense layer (20 units). The combination of using more training data and hyperparameter tuning drastically improved accuracy and recall from the previous simple CNN model (model #4). There is strangely a tendency for the model to misclassify images as BKL as seen in **Figure 14**.

CNN + Optuna Classification Report

	precision	recall	f1-score	support
MEL	0.286	0.126	0.157	895
NV	0.517	0.535	0.526	2592
BCC	0.157	0.028	0.047	684
AK	0.025	0.006	0.010	160
BKL	0.092	0.293	0.140	502
DF	0.000	0.000	0.000	44
VASC	0.000	0.000	0.000	47
SCC	0.000	0.000	0.000	142
accuracy			0.329	5066
macro avg	0.125	0.124	0.110	5066
weighted avg	0.332	0.329	0.317	5066

Accuracy: 0.3293
ROC AUC: 0.4918

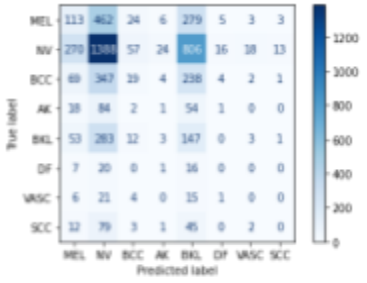


Figure 14. Result metrics for CNN + hyperparameter tuning using Optuna

6. CNN + transfer learning using using EfficientNet

Experiment: Understanding of EfficientNet (abbreviated as *EN*) is first limited to EN-B0, which has the fewest amount of parameters and fastest to train. Initial training on EN-B0 is compiled with Adam with a regular learning rate of 1e-3, but with all parameters in transferred EN-B0 (ImageNet weights)

frozen/made untrainable. Once training is done (early stopping on AUC-ROC; 48 epochs), fine-tuning is performed with the last 1,995,320 parameters (out of 4,049,571 parameters) in EN-B0 made trainable (arbitrary choice inspired by example notebook provided in live session). Fine-tuning training is compiled with Adam with a low learning rate of 1e-5 to prevent destruction of pretrained weights. With an early stopping callback (that was never invoked), the training is completed in 200 epochs.

Results/Graphs: The unoptimized EN-B0 model performs slightly poorer than a hyperparameter-tuned CNN (model #5), although the AUC-ROC is comparable. The model has tendencies to mislabel images as BCC and BKL as seen in **Figure 15.**

CNN + EfficientNet Classification Report

	precision	recall	f1-score	support
MEL	0.204	0.147	0.171	895
NV	0.511	0.438	0.467	2592
BCC	0.136	0.151	0.143	684
AK	0.026	0.025	0.025	168
BKL	0.092	0.145	0.112	582
DF	0.006	0.023	0.009	44
VASC	0.000	0.000	0.000	47
SCC	0.025	0.049	0.033	142
accuracy			0.283	5066
macro avg	0.125	0.121	0.128	5066
weighted avg	0.326	0.283	0.301	5066

Accuracy: 0.2831
ROC AUC: 0.4891

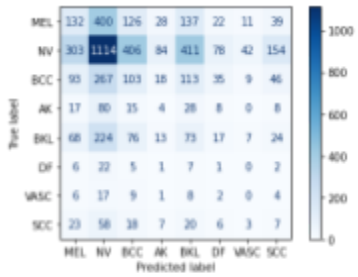


Figure 15. Result metrics for CNN + transfer learning using using EfficientNet

7. CNN + Optuna + EfficientNet

Experiment: By comparing the eight EN baseline models, EN-B3 is identified as the best performing model as seen in **Figure 16**. With that, an Optuna hyperparameter tuning study is performed with 9 trials (more is ideal, but expensive) to optimize the dense layers that occur after the EN-B3 layers. The best hyperparameter set is 1 dense layer with 25 units (performed better than models with more layers; up to 4 layers were tested). After normal training on dense layers (66 epochs), fine-tuning training on EN-B3 is done (155 epochs).

Results/Graphs: The optimized EN-B3 model did not perform much better than non-Optuna EN-B0 (ROC-AUC is slightly better; accuracy, precision recall are slightly worse). The tendency of misclassification towards BCC and BKL are also observed in the confusion matrix as seen in **Figure 17**.

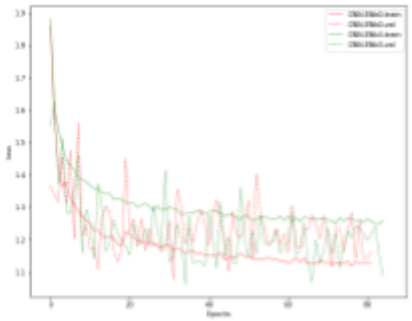


Figure 16. Training history of loss shows that the last validation loss for EN-B3 is lower than EN-B0, although both models' validation metrics are noisy.

CNN+Optuna+EfficientNet Classification Report

	precision	recall	f1-score	support
MEL	0.171	0.187	0.179	895
NV	0.500	0.371	0.426	2592
BCC	0.118	0.140	0.128	684
AK	0.024	0.031	0.027	168
BKL	0.092	0.155	0.116	582
DF	0.000	0.000	0.000	44
VASC	0.000	0.000	0.000	47
SCC	0.052	0.063	0.057	142
accuracy			0.260	5066
macro avg	0.128	0.118	0.117	5066
weighted avg	0.314	0.260	0.281	5066

Accuracy: 0.2598
ROC AUC: 0.4937

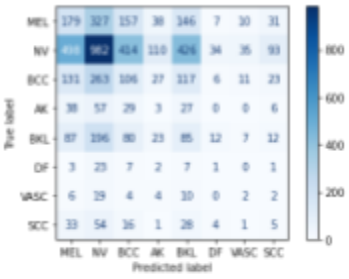


Figure 17. Result metrics for CNN + Optuna + EfficientNet

8. CNN + XGBoost

Experiment: The EN-B3 model (#7) was loaded, and the outputs of the EN and dense layer is used for inputs for XGBoost model. Because the EN + dense layer output size is small (25 units), each boost round in XGBoost does not take much time. A simple hyperparameter tuning using Optuna was done to find a roughly optimal model – booster: gblinear (generalized linear model), L2 regularization term on weights (lambda): 4.37e-5, L1 regularization term on weights (alpha): 8.17e-5.

Results/Graphs: On test data, the CNN + XGBoost (gblinear) model has good comparable recision and AUC-ROC, but less optimal accuracy and recall (see **Figures 18 and 19**). Slightly poor performance can be partially explained by the overfitting of data. One possible way to mitigate this could be to augment

image data going into CNN (EN-B3) layers, but the implementation (coding) is too complex for us to take on for this project.

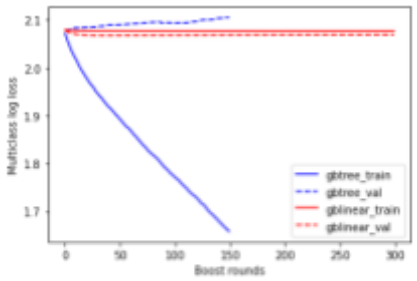


Figure 18. gbtree rounds are not generalizable; gblinear boosting rounds don't overfit, but does not improve multiclass log loss.

```
Classification Report
              precision    recall  f1-score   support

         MEL       0.165     0.059     0.087       895
          NV       0.500     0.333     0.400      2592
         BCC       0.209     0.034     0.058       684
          AK       0.047     0.050     0.049       160
         BKL       0.128     0.044     0.065       582
          DF       0.009     0.091     0.016        44
        VASC       0.014     0.170     0.025        47
         SCC       0.038     0.324     0.855       142

    accuracy                          0.203      5066
   macro avg       0.138     0.138     0.094      5066
weighted avg       0.328     0.203     0.238      5066

Accuracy: 0.2027
ROC AUC: 0.5863
```

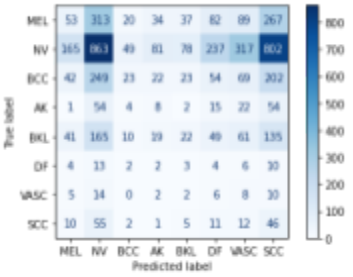


Figure 19. Result metrics for CNN + XGBoost

9. CNN + EfficientNet + metadata (combining image data and structured metadata in a model)

Experiment: The metadata-assisted model went through several iterations of structure (e.g.: which layer in EN-B3 should be used as concatenation input? How many convolution layers to pass metadata input through before concatenation? Number of dense layers after concatenation?). The final structure chosen is the one outlined in **Figure 20**. Training is compiled using Adam optimizer with a low learning rate (1e-5).

```
Classification Report
              precision    recall  f1-score   support

         MEL       0.248     0.333     0.284       895
          NV       0.371     0.035     0.065      2592
         BCC       0.146     0.069     0.093       684
          AK       0.035     0.381     0.064       160
         BKL       0.087     0.010     0.018       582
          DF       0.014     0.045     0.022        44
        VASC       0.000     0.000     0.000        47
         SCC       0.045     0.028     0.035       142

    accuracy                          0.100      5066
   macro avg       0.108     0.113     0.071      5066
weighted avg       0.257     0.100     0.100      5066

Accuracy: 0.1005
ROC AUC: 0.5219
```

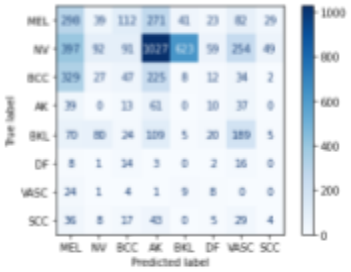


Figure 20. Result metrics for CNN + EfficientNet + Metadata

Results/Graphs: Training history shows that while crossentropy loss is decreasing over epochs, accuracy, recall, and AUC-ROC decrease with more epochs (for both training and validation data). The root cause of degradation is unknown, but model setup/coding mistake could be a factor (this is a complex functional model).

10. K-Prototypes

Experiment: The metadata features are patient age (Numerical), sex (Categorical), and general anatomic site (Categorical). After removal of missing data, the final dataset consisted of 21,311 rows × 3 columns. The Elbow Method was used to decide on the number of clusters (K). We found 3 is the optimal number as seen in **Figure 21**.

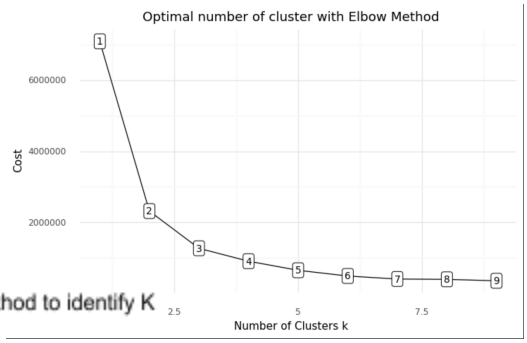


Figure 21. Elbow method to identify K

Results/Graphs:

K-prototypes produced 3 clusters around Age 74, 52 and 32. **Figure 22** represents a scatter plot between Age and Site for K=3. Ultimately, the performance of the K prototypes model was uninformative. Utility may be increased by incorporating a dimensionality reduction method to embed the data into two dimensions.

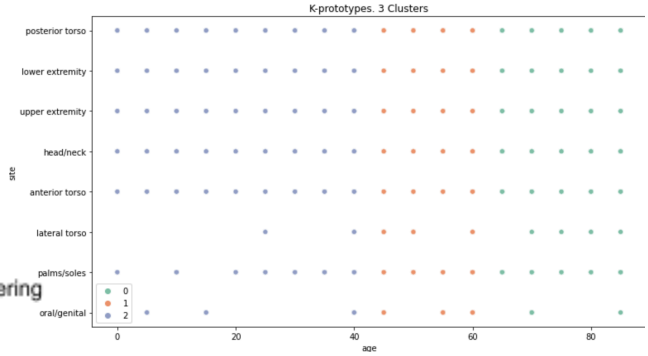


Figure 22. K-prototypes clustering

Comparison of models and Conclusions

Overall, the performance of the models on test data was poor. Based on Tensorflow training performance metrics, only one model met our performance threshold which was the CNN + EfficientNet(B3)+ Optuna model. Comparison of the train vs. validation performance metrics for various CNN models are described in **Figure 23**.

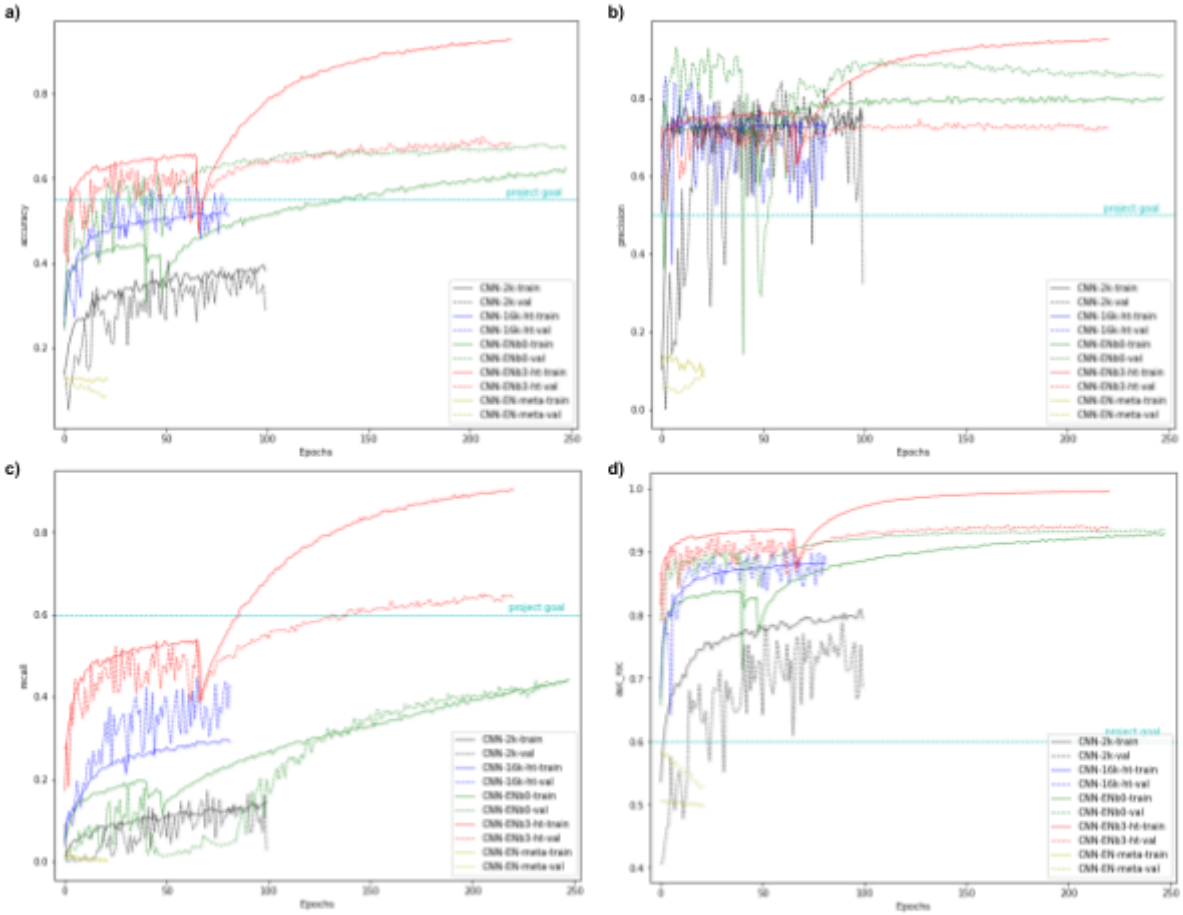


Figure 23. CNN models' training history (training set in solid line, validation set in dashed line), evaluation metrics (categorical): a) accuracy, b) precision, c) recall, d) AUC (ROC). Project goals are horizontal lines in cyan. The model abbreviations are – *CNN-2k*: small dataset; *CNN-16k-ht*: big dataset Optuna-tuned; *CNN-ENb0*: EfficientNetB0; *CNN-ENb3-ht*: EfficientNetB3 with Optuna hyperparameter-tuning; *CNN-EN-meta*: EfficientNet with metadata.

Despite promising training and validation performance, all models had poor performance on the test dataset (see **Figure 24**). Eight of the models had a high rate of misclassification of melanoma. From a clinical utility perspective, this is concerning as this would result in false negatives of an aggressive cancer. Out of all of the models, the standard CNN and logistic

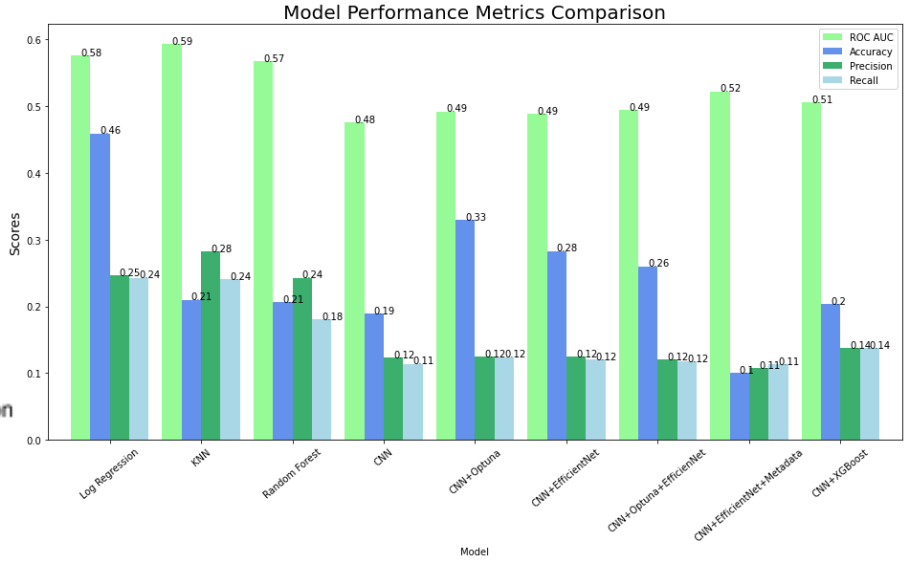


Figure 25. Overall performance comparison

regression had the best recall for the melanoma class, although it was only around 25%, falling far short of our 50% recall threshold.

Standards

Jupyter lab was utilized via Amazon SageMaker. The Python 3 image (Tensorflow 2.6 Python 3.8 GPU optimized) was utilized with EC2 instances of different sizes depending upon the model computational needs. The following packages and versions were utilized: Tensorflow 2.6.2, keras 2.6.0, scikit-learn 1.1.3, numpy 1.19.5, matplotlib 3.5.0, and pandas 1.2.5. Each model's performance was evaluated using the Sklearn.metrics library, with macro average selected for precision and recall. Performance for CNN models utilized the Tensorflow metrics and scikit-learn metrics for more direct comparisons. Note however that there is a difference between the tensorflow metrics and scikit-learn metrics, which is not well understood by this project group at this time.

Constraints

Due to the large, high-resolution image dataset, computational burden was a challenge. To address this, a ml.g4dn.xlarge EC2 instance was utilized. However, instance memory still posed a challenge and smaller test datasets were needed for certain algorithms (KNN). The computational burden influenced our approach to utilizing a smaller balanced dataset (2000 images) on certain models.

Strengths and Limitations of the Study

One of the strengths of this was that it utilized a large real-world dataset. Being a highly imbalanced dataset, we performed augmentation which improved the model's fairness. Additionally, we attempted to improve the model performance by implementing cropping to remove the dark fields. Further strengths of this study are the number of attempted models, utilization of hyperparameter tuning, and utilization of transfer learning.

The limitations of this study include reduced generalizability. Many of the images represent lesions from white patients, reducing generalizability across other races. Additional limitations result from smaller training sets due to computational limitations and reduced model optimization/tuning due to time constraints.

Future Work

Performance of the models could be improved by additional evaluation of the impact of non-lesion features in the images. For example, does the presence of rulers, marks, or hair occur more often in one skin lesion class compared to another? Principal component analysis can be used to understand which variables are correlated and to reduce dimensionality. Additional analysis can be done to determine if adjusting the hue or saturation of the image improves performance across multiple races and skin tones. Also, exploring different clustering techniques for metadata.

References

[1] International Skin Imaging Collaboration (ISIC) on Kaggle, 2020: https://www.kaggle.com/competitions/siim-isic-melanoma-classification/overview

[2] Cancer.net, February 2022. *Melanoma: Statistics*. https://www.cancer.net/cancer-types/melanoma/statistics#:~:text=It%20is%20estimated%20that%207%2C650,people%20worldwide%20died%20from%20melanoma.

[3] ISIC Archive : https://www.isic-archive.com/

[4] Tschandl P., Rosendahl C. & Kittler H. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci. Data 5, 180161 doi.10.1038/sdata.2018.161 (2018)

[5] Noel C. F. Codella, David Gutman, M. Emre Celebi, Brian Helba, Michael A. Marchetti, Stephen W. Dusza, Aadi Kalloo, Konstantinos Liopyris, Nabin Mishra, Harald Kittler, Allan Halpern: "Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), Hosted by the International Skin Imaging Collaboration (ISIC)", 2017; arXiv: 1710.05006.

[6] Marc Combalia, Noel C. F. Codella, Veronica Rotemberg, Brian Helba, Veronica Vilaplana, Ofer Reiter, Allan C. Halpern, Susana Puig, Josep Malvehy: "BCN20000: Dermoscopic Lesions in the Wild", 2019; arXiv:1908.02288.

[7] Haenssle, H. A., et al. (2018). Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists. *Annals of oncology : official journal of the European Society for Medical Oncology*, *29*(8), 1836–1842. https://doi.org/10.1093/annonc/mdy166

[8] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta,and Masanori Koyama. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. In KDD. https://optuna.org/

[9] Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv. https://doi.org/10.48550/arXiv.1905.11946

[10] dmlc XGBoost read-the-docs website (2022). https://xgboost.readthedocs.io/en/stable/index.html

[11] Ren, X., Guo, H., Li, S., Wang, S., Li, J. (2017). A Novel Image Classification Method with CNN-XGBoost Model. In: Kraetzer, C., Shi, YQ., Dittmann, J., Kim, H. (eds) Digital Forensics and Watermarking. IWDW 2017. Lecture Notes in Computer Science(), vol 10431. Springer, Cham. https://doi.org/10.1007/978-3-319-64185-0_28

[12] Chollet, F (2020) Keras, Transfer learning & fine-tuning. https://keras.io/guides/transfer_learning/