Final Project

LesionCheckr

Group 7

An AI–Driven Application to Classify Skin Lesions

Andrew Mark Dale
S/N: 100491442

Ali Istanbullu
S/N: 100905755

Chinedu Omenkukwu
S/N: 100805353

Andre Dallaire
S/N: 100337151

AIDI 2004: AI in Enterprise Systems
Durham College

Project

Motivation:

We have chosen project #43 from the document. Skin cancer is an extremely common cancer. According to a number of studies, 9,500 Americans are diagnosed with skin cancer every single day [1]. Therefore, there exists room for an AI-driven solution to aid in the diagnosis of skin cancers. We have decided to create a phone application using Flutter. Developing our solution in Flutter allows us to create an application that can work across a number of different platforms (ours is optimized as a mobile application). The model was trained in a Google Colab notebook. The backend is a Flask application that is hosted on Heroku. We could have easily hosted the model on Azure; however, we have more experience with Flask which has streamlined our development.

Data:

Data source: https://www.kaggle.com/datasets/kmader/skin-cancer-mnist-ham10000

The data used to train our model was taken from Kaggle. Specifically, there is a competition named “Skin Cancer Mnist HAM10000” that has 10,015 large images. The model has the following classes: Actinic keratoses and intraepithelial carcinoma / Bowen’s disease (akiec), basal cell carcinoma (bcc), benign keratosis-like lesions (solar lentigines / seborrheic keratoses and lichen-planus like keratosis, bkl), dermatofibroma (df), melanoma (mel), melanocytic nevi (nv) and vascular lesions (angiomas, angiokeratomas, pyogenic granulomas and hemorrhage, vasc).

Development:

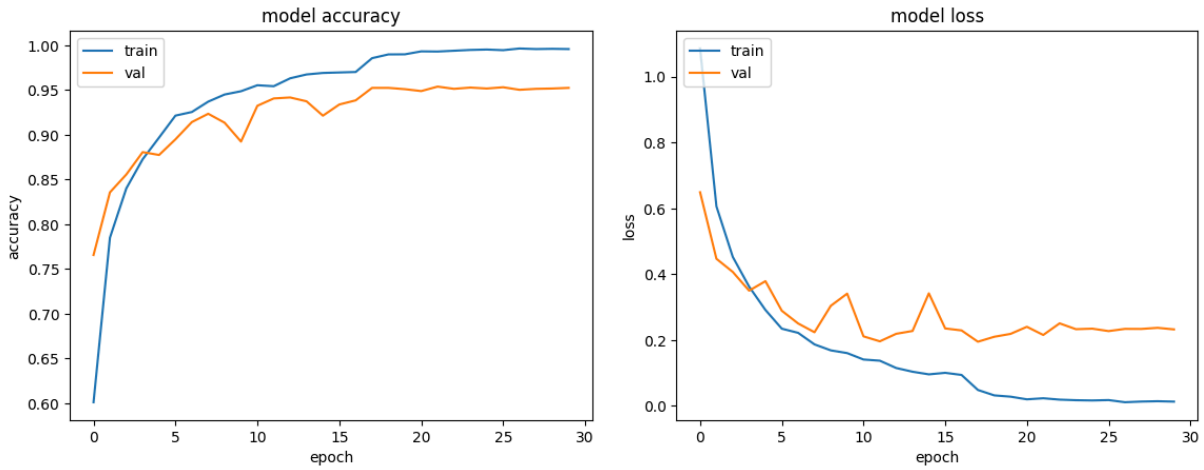
To train our model, it was necessary to use the Kaggle API to download the set (6 GB). The dataset is imbalanced, and we needed to balance the number of samples on which we are to train the model. Luckily, the dataset comes with a data dictionary that lists the image files and gives the diagnoses. Now, to balance the dataset, we used “resample” from Sklearn, which allowed us to even out the classes (the dataset is dominated by normal moles and melanoma). Once this was completed, we were left with 2000 images for each of the above classes.

The images were reshaped to (224,224,3), 224 pixels by 224 pixels by 3 channels, an RBG image. Now we have to change the images to Numpy Arrays and change the label column to categorical. We are now able to split the set into training images, testing images, training labels and testing labels.

Our model is a deep convolutional neural network that also makes use of transfer learning. Since our model is large, it necessitated the use of a GPU-accelerated environment to keep the runtime reasonable. The base model we used was EfficientNetB0, which is a large pre-trained network that is typically used for thousands of classes. We set the EfficientNet model to not be

trainable, leaving us with 4,050,531 non-trainable parameters. For the part of the model we developed ourselves, we decided to use a multitude of different layers. Beyond the EfficientNet model, we use groupings of Conv2D, MaxPooling and BatchNormalization. Rather than flattening the network prior to the fully connected layers, we instead use GlobalAveragePooling, which accomplishes the same task but removes a significant number of trainable parameters, making training time much faster, our model has 1,944,599 trainable parameters. The total number of parameters for the model is 5,995,130. To give the model some additional help, we reduce the learning rate on validation loss plateaus, to ensure a tighter convergence.

On the test data, our model accomplishes a 99% accuracy. We have the following charts that show the accuracy over epochs and loss of epochs. We have built a number of different models, as seen on the GitHub for our backend.

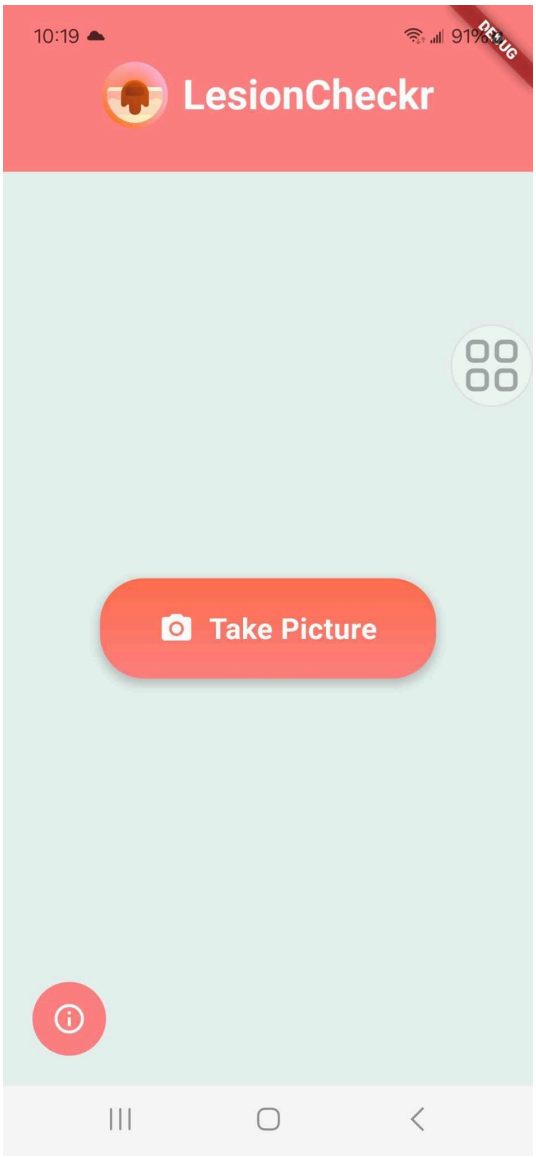


As can be seen from the above charts, our model has excellent accuracy. However, the model is only as good as the training data, and we would never suggest that this type of application could replace consulting with a physician – we have noted as much in the application, if the user clicks the information button, a little popup will say that they should consult with a physician if concerned along with giving a few suggestions.

Application:

Our application was developed using the Flutter framework and the Dart programming language. It is a very simple application. It starts on the homepage, which allows the user to take a photo, or hit the information button at the bottom to learn how to use the application as well as a little blurb about consulting a physician. Once the “Take Picture” button is clicked, the user is directed to their camera. Once they take the photo, the photo is redirected to the next page and presents them with a button to classify the image. Once that button is clicked, it sends the image to our backend for classification. The classification is then displayed above the image.

Here are some sample images from our application:



Home page

Information page

Classification page (this is an image of BCC)

Backend:

The backend is a simple Flask application. The Flutter application sends the image to the backend, the backend responds with the image classification. Very simple, and it works. Basically, we just load the model, and labels. We decode the image sent from the application from base64 and ensure that it is RGB. It is then resized to be passed to the model. The image numpy array is expanded along its first axis. Given this array, the model can now predict the type of skin lesion. The prediction is passed to a label dictionary, and the value is returned to the Flutter app. The backend is hosted at:
https://final-project-backend-group7-a195d5bde686.herokuapp.com/

Collaboration Plan and Work Division

We weren't sure if this was necessary, but we included it as it was required for the other assignments.

Meeting Schedule:

Our group communicated using Microsoft Teams and held informal meetings to discuss our project.

Work Hours Commitment:

This project required quite a bit more effort than anything that had come before it. Luckily, we were able to work collaboratively using Google Colab and on the Flutter application as necessary. We each worked a decent number of hours on the project.

Work Division:

Beyond all of us contributing to the final selection of a paper, delegated duties were as follows:

Andrew Mark Dale (S/N: 100491442): Developed model, Flutter application and the Flask backend.

Ali Istanbullu (S/N: 100905755): Ali aided in the development of our model. He has also aided in the development of this report.

Chinedu Omenkukwu (S/N: 100805353): Chinedu aided in the development of the Flutter application. He gave some insightful commentary on the model development.

Andre Dallaire (S/N: 100337151): Andre helped get the Flask backend hosted on Heroku and aided in the development of the model and Flutter application.

Conflict Resolution and Inequitable Work Distribution:

Conflicts will be settled through open dialogue. Each of us wants to do well in our classes; therefore, we should expect to freely discuss any issues we have with regards to assignments. We expect to have equitable work distribution; however, if the need arises, we will discuss delegating more duties as required. As with any group work, it is inevitable that some students will do more; this is acceptable, as long as it is not egregious.

References

[1] Skin Cancer Facts & Statistics - The Skin Cancer Foundation