DermaGram

Andrew Hearst, Shondell Baijoo, Steven Spear, Jaspal Singh
GitHub Link: https://github.com/DermaGram/DermaGram

When a dermatologist performs a screening on a patient, she has only her sight and experience to identify which parts of the skin may exhibit features commonly associated with cancer, namely melanoma. The five tell-tale signs may be easily remembered as the ABCDE's[1]:

- A for asymmetry
- B for border irregularity
- C for color
- D for diameter
- E for evolution

If any region on the patient's skin exhibits one of these characteristics, then the dermatologist will likely recommend a biopsy, in which case the affected region is surgically removed and delivered to a pathologist for review.

The purpose of DermaGram is to aid the dermatologist in making this important determination - whether or not a biopsy should be performed. It is important for several reasons:

- The cost of a biopsy may range from $100-$500.
- Failure to perform a biopsy on a skin lesion in its earliest stages leads to a lower survival rate[2]:
    - **Stage I:** The 5-year survival rate is around 92% to 97%.
    - **Stage II:** The 5-year survival rate is around 53% to 81%.
    - **Stage III:** The 5-year survival rate is around 40% to 78%.
    - **Stage IV:** The 5-year survival rate is about 15% to 20%.

DermaGram allows dermatologists to create a private account, upload an image of their patient's skin lesion, generate a probability as to whether or not the skin lesion should be biopsied, and store this information for future review. The application was developed in such a way that dermatologists and patients could trust that their images would remain safe and secure.

In order to build such an application, our team focused on training an image-recognition model to classify images, setting up a database to store user information, and developing a website to encapsulate the user experience. Below is a breakdown of each component of the application with more technical details.

Registration/Login

Users must create an account with a password and username that is at least 6 characters long. The password is saved using SHA256 encryption to a local MySql database.

[1] https://www.melanoma.org/understand-melanoma/diagnosing-melanoma/detection-screening/abcdes-melanoma

[2] https://www.cancer.org/cancer/melanoma-skin-cancer/detection-diagnosis-staging/survival-rates-for-melanoma-skin-cancer-by-stage.html

Image Storage/Retrieval

Once a login account is setup, a private album is created on Imgur to store the images. Imgur uses OAuth 2.0 authentication, which means all requests are encrypted and sent via HTTPS.[3] Everytime an album is accessed, a unique token and ID must be supplied in the request to verify the authenticity of the user before any data or images may be transmitted.

Analysis/Image Classification

DermaGram will process the image using an image-recognition classifier based on TensorFlow's neural network based ResNet model. The model will return three classifications: Biopsy, No Biopsy, and Unsure, as well as their associated probabilities, or confidence levels. The dermatologist may incorporate these values into their decision making process when debating whether or not a biopsy should be performed.

The following is a summary by each team member discussing their individual contribution.

Andrew Hearst

I originally came up with the idea for DermaGram when I visited my dermatologist in August of 2017. The idea that dermatologists lacked any tools beyond their own sight and experience to identify skin lesions and classify them as cancerous seemed unsatisfactory. My opinion was that this is an area of opportunity to introduce a solution using the latest image recognition technology. As such, I presented it as my idea for the Capstone project and was excited to have been one of four projects selected for the semester. Furthermore, it was a pleasure working with Shondell, Steven, and Jaspal, without whom this application would not be what it is today.

In September, I focused on collecting and curating pre-labeled images from a qualified source – the ISIC archives.[4] We resized the images using an online tool called Bulk Resize[5] and cropped out any features that may negatively affect the image recognition model, such as black outlines and hair.

In October, I installed a linux VM at home using Virtual Box and activated TensorFlow to train Inception-V3[6], which is one of the image recognition models. Since my computer was limited to four CPUs and no GPU, I only retrained the final layer of the neural network using a new set of categories.[7] I also documented the installation process for each operating system, the dependencies for each utility, and kept track of our ToDo list in GitHub.

In November, I created an account with Imgur and integrated the ImgurPython API[8] into our app to create albums, upload images, and download their metadata. Every time a user creates an account with DermaGram, we generate a unique Imgur album. The user's account information is stored in a local MySql database from which we retrieve the unique album id in order to authenticate the uploading and downloading of images to Imgur.

I also re-organized the screen code and python utilities so that we followed the design principles of keeping the code as modular as possible. For example, the main app.py script which

[3] https://apidocs.imgur.com/
[4] https://isic-archive.com/
[5] https://bulkresizephotos.com/
[6] https://www.tensorflow.org/tutorials/image_recognition
[7] https://www.tensorflow.org/tutorials/image_retraining
[8] https://github.com/Imgur/imgurpython

launches the website was simplified to generate three templates – home, login, and profile. I broke down the various components of each template into their own HTML pages, such as ‘image_table’, ‘analysis_results’, and ‘nav_bar’. I also defined the python-based classes for the different utilities, namely ‘imgur_utils’, ‘file_utils’, and ‘session_utils’.

In December, I implemented the code that connected Steven’s analysis button to Jaspal’s ResNet model, displayed the classifications and their probabilities in ‘analysis_results’, and stored these results as part of the image’s metadata on Imgur so that it could be shown in ‘image_table’.

Shondell Baijoo

In the beginning of the project, we all created sample TensorFlow Inception-V3 models using their introductory flower dataset. This gave us all a better understanding of how this model functioned. Once this was done, for the remainder of September, I worked on the back end by creating a database in MySql. I chose MySql because we had decided to create the website in Python Flask, and MySql has seamless functionality with Python. At this point, I simply created the database locally on my machine. I wrote a script for the database, so that my other team members could also make a local database on their machines.

Once the database was set up, I was able to create login and register functions that utilized the database for user information. The user interface form for registration was created using WTForms, a Python module. This was used as it includes validation of correct input. It allowed me to limit the number of characters for user input. This protects our site from buffer overflow by rejecting input that is too long. It also checks that the user enters what could be a valid email address (ie. the string ends in “@somewhere.com”). In addition, if the input is incorrect, it returns messages to the user outlining what they did wrong. This makes creating an account with our site user friendly. During registration, the user’s password would be hashed with a SHA256 implementation from the Python module passlib and stored into our database. SHA256 is a one-way secure encryption, developed by the NSA. This protects our user passwords from anyone who can see the contents of our database, for if someone is able to see the hashed password, they would still not be able to decipher what the plain text version is.

After a user is registered, they could then log in to the site. Login was done by querying the database using escape_string(), a function in the mysqldb module in Python. This can help protect against SQL injection because it scans the database for the entire input, including whitespace, as a string, rather than recognizing them as commands for the database. In other words, if I had input “username SELECT * FROM DG_User” into the username box, it will recognize that entire string as a username, and search the database for it, rather than recognizing ‘username’ and running the rest as a query. The login function also hashes the input password using SHA256 and confirms that it matches the query in the database. If there is any incorrect input such as no username, and incorrect username, or an incorrect password, the login function will return false, and using the Python module Flash, we display a message to the user, telling them the error.

While setting up login and register functions, I also worked on getting the website onto a web hosting server. I did research for servers that work well with Flask and Python. Eventually I came across PythonAnywhere, and created an account for us. I chose PythonAnywhere because the server is specifically built for Python based websites. They have Bash consoles and error logs on the server that you can use for set-up and debugging. They also give you the option to create a virtual environment, which allowed us to easily add additional modules to our site like TensorFlow and passlib.

Once the back-end work for login and register was complete, I focused on the front end. I went through many modifications of the login/register forms. Since they used different functions, it took time to make them look the same. I wound up using a CSS sheet for them both in order to keep the formatting consistent. Once that was done, using HTTP and CSS, I changed the UI of the entire site by giving it a coordinating color theme and background texture.

Steven Spear

When I first saw the presentation of the DermaGram idea in class, I was both impressed and intrigued, given my long standing interest in health and healing. If this interest could be combined with my Computer Science project, it would be a rewarding experience with the potential possibility of helping people. I was excited to learn that the DermaGram project had been chosen, and that I was going to be working along with Andrew, Shondell, and Jaspal. I could sense that these new team members were as deeply dedicated to this project as I was.

I quickly started learning and using new software and technologies and was having a lot of fun in the process. For instance, Shondell was experienced with Slack, which is a team collaboration tool, and we all started using it to communicate with one another about our new work-in-progress. I had some knowledge of Python and Flask, which is a Web Framework, as I had built a Blog application over the summer with it. I also started using GitHub, which I knew a little. I quickly came to learn that I needed training in the workings of GitHub, and Andrew, who was proficient, helped me quite a bit.

My first task was to use Flask to create a Website that had an upload button and present it as a completed weekly assignment. I was able to accomplish this. I felt that the class was well structured with the professor overseeing the project on a week-to-week basis and assigning tasks to be completed to each team member.

The next task I was involved with pertained to the API at Imgur. I was directed to download a program called Postman, which is an API development program. With this program, Imgur helps you to connect with their API in a secure fashion. I ran into an issue that took some time and energy to resolve because the developers at Imgur had not fixed a bug in the registration process. Once this bug was fixed, I was able to proceed.

I wrote Flask code that generated a Login function and then had issues linking it to the previous version of the Website I had built. The database the Login function used was SQLAlchemy, which comes packaged with Flask. Our team decided that we should use MySQL for the database in lieu of SQLAlchemy. I tried installing MySQL on Windows, which I normally run, but I had issues with that. It was at this point that I switched to using Linux, which I have installed on my system through a Virtual Machine to do the rest of the development for this project. I had some previous experience with Linux, but this gave me a lot more, and I learned very much.

My next task was to design an upload function using Flask code. I learned that I had to upload the image file from my system to a folder in the "/static" directory so the app could utilize it properly. This involved finding the File Path which Browsers do not allow due to security reasons.

As the semester came to a close I worked very hard to obtain a working link between a row in an HTML Table and an image. I studied Jquery, which is a JavaScript Library, to be able to implement this. I was able to add a button to the table row, which, when clicked on, showed the image associated with that row. The page that I implemented the JQuery Script on was a template generated by the Jinja2 Templating Engine, which is a part of Flask. Another page was added to this page with the same process and I was able to establish a connection between these two pages using JQuery.

At the beginning of the semester I had hoped to help more with the Machine Learning aspect of this project. As the semester progressed I found that the time constraints of meeting the deadlines of the app development did not allow me to study Machine Learning. I'm thankful for being introduced to Tensorflow, which is the program used in this project for Machine Learning applications such as neural networks. I plan to go over how Tensorflow was used in this project and study this aspect of Computer Science after the semester ends.

Jaspal Singh

Jaspal researched the various models that could be trained from AlexNet and TensorFlow, such as Inception-V3 and ResNet. His research led to a deeper understanding about the difference between transfer learning and fine-tuning. In order to improve the model's predictive capabilities, he also experimented with different parameters and actively cleaned the training data set. Thanks to his efforts we were able to integrate a trained image-recognition model into DermaGram.