Lambton College | In Toronto

Capstone Project Weekly Progress Report

Semester	Fall 2022
Course Code	AML 2404
Section	Section 2
Project Title	Skin Diseases Classification using Deep Learning
Group Name	G
Student names/Student IDs	Tomson George (C0857730) Praveen Mahaulpatha (C0860583) Thulana Abeywardana (C0861333) Jaskaran Singh Moti (C0860026)
Reporting Week	Week 5 (09 October 2022 to 16 October 2022)
Faculty Supervisor	William Pourmajidi

1. Tasks Outlined in Previous Weekly Progress Report

Task 01: Create a server and understand how to implement the model in the cloud and make it live

Responsible: (Tomson)

Using Elastic Beanstalk service, a server for deploying the application was created. Furthermore, using the code pipeline service by AWS, a connection between our GitHub repository and server was created to deploy a sample application.

Task 02: Initialization of MLOps

Responsible: (Praveen)

In this task, the key goal was to initiate the implementation of MLOps in the project. Research of the domain was carried out to understand the key concepts and applicable implementations for our specific project. Secondly, from the said concepts, some were implemented to an extent that will be further explained in the latter of this report.

Task 03: Improving Model performance of Model 1

Responsible: (Thulana)

Even though the model was trained for different hyperparameters, adding and dropping layers, significant progress has not been observed in model performances (overfitting). Therefore, data needs to be preprocessed again before being put into the data pipeline.

Task 04: Plotting the validation results using Plotly

Responsible: (Jaskaran)

Learned basics of graph representation using plotly. Learned to plot 2d graphs and how they can play an important role in data visualization.

2. **Progress Made in Reporting Week**

Task 1: Tomson: Implementing sample Flask project with and making it live on the server

1.1 Learn more about the Flask framework and how to make API calls.

Created a sample flask application to upload and save the image. It was running successfully on the local machine, so a server was created using Elastic Beanstalk. The code was uploaded from Github to the server and deployed. The application was successfully deployed and reached using a URL provided by AWS.

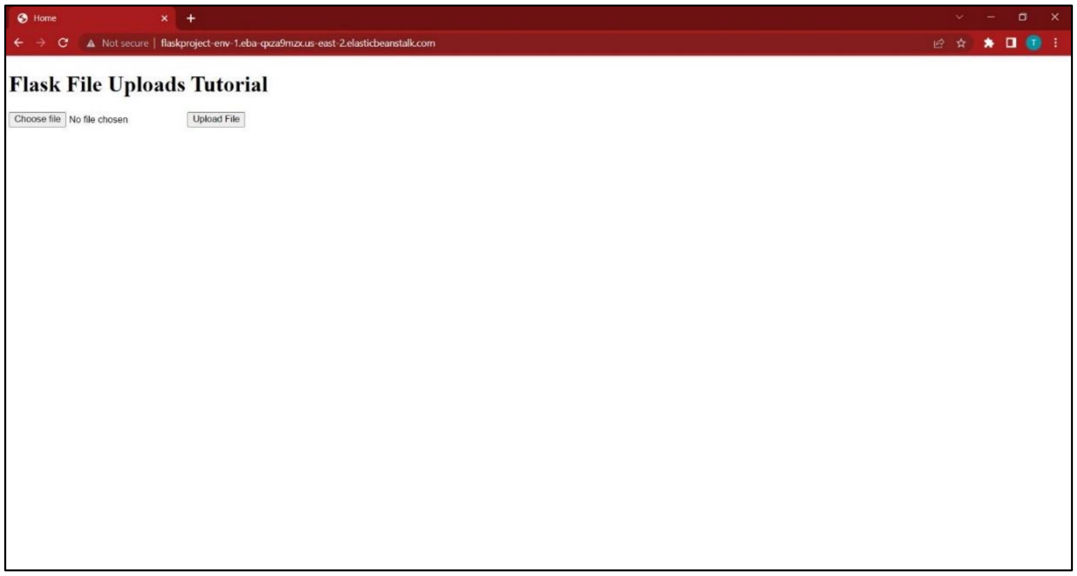

Home
Not secure | flaskproject-env-1.eba-qxza9mzx.us-east-2.elasticbeanstalk.com
Flask File Uploads Tutorial
Choose file
No file chosen
Upload File

Figure 1.1

1.2 Figure out what went wrong and fix the 502 error.

This problem was solved simply by renaming the app object variable name and python file name to the application.

Task 2: Praveen: Initialization of MLOps

Simply put, MLOps is a set of processes that target the deployment and maintenance of machine learning models in production to be reliable and efficient. Figure 2.1 represents the high-level representation of the MLOps lifecycle.

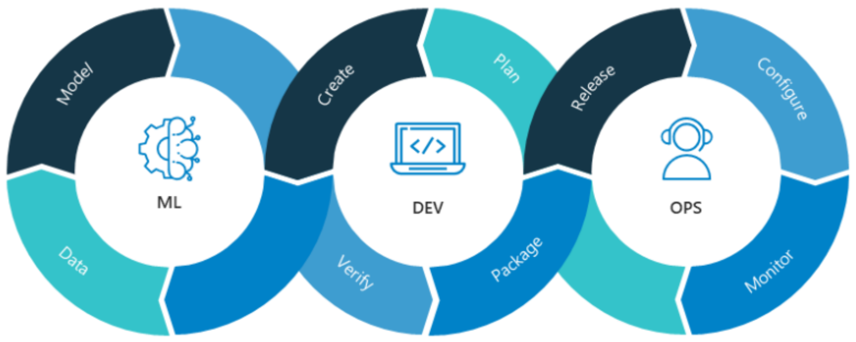


Figure 2.1

Also, identified key 2 concepts of MLOps that we try to implement in this project are:
1) Versioning of the Machine Learning Model
2) Automation of ML pipeline through CI/CD (Continuous Integration and Continuous Deployment)

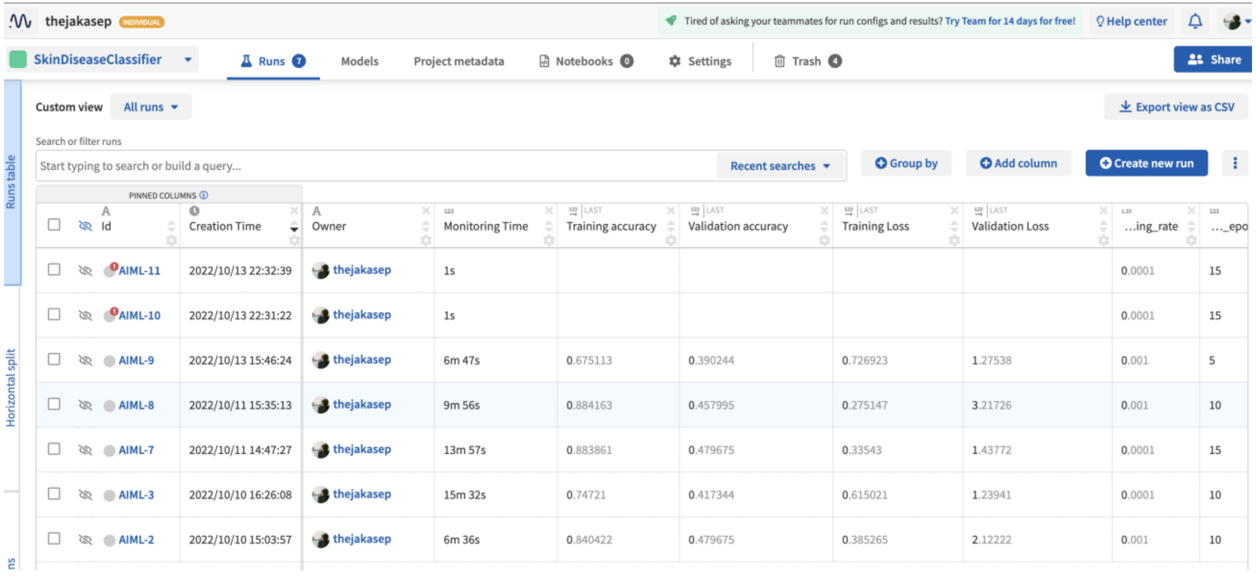
2.1 Versioning of Machine Learning Model

Machine learning models are run to generate an expected accuracy level on the prediction that we want. In order to do that, there are things that are fine-tuned. Some of the main fine-tuned points of a machine learning model are,

- Hyperparameters: Parameters that we explicitly define in the model. (the number of hidden Nodes and Layers, input features, Learning Rate, and Activation Function.)
- Parameters: The attributes of the training data that are learned during training by the Machine learning model(Generally weights and biases)
- Data: The input dataset that we use to train, validate and test the model

It is important to keep the versioning of a model with respect to the corresponding values of these parameters to regenerate the expected output at any given time.

For this matter, we used a platform named Neptune.ai to Log, store, query, display, organize, and compare all our model metadata. We have used an API to log the data from the python script, and desired parameters to be logged were explicitly defined. Figure 2.1.1 shows the logged hyperparameters of our model of each run after tuning some.



thejakasep INDIVIDUAL — Tired of asking your teammates for run configs and results? Try Team for 14 days for free!

SkinDiseaseClassifier — Runs 7 | Models | Project metadata | Notebooks 0 | Settings | Trash 4 | Share

Custom view: All runs — Export view as CSV

Search or filter runs: Start typing to search or build a query... — Recent searches | Group by | Add column | Create new run

Id	Creation Time	Owner	Monitoring Time	Training accuracy	Validation accuracy	Training Loss	Validation Loss	...ing_rate	..._epo
AIML-11	2022/10/13 22:32:39	thejakasep	1s					0.0001	15
AIML-10	2022/10/13 22:31:22	thejakasep	1s					0.0001	15
AIML-9	2022/10/13 15:46:24	thejakasep	6m 47s	0.675113	0.390244	0.726923	1.27538	0.001	5
AIML-8	2022/10/11 15:35:13	thejakasep	9m 56s	0.884163	0.457995	0.275147	3.21726	0.001	10
AIML-7	2022/10/11 14:47:27	thejakasep	13m 57s	0.883861	0.479675	0.33543	1.43772	0.0001	15
AIML-3	2022/10/10 16:26:08	thejakasep	15m 32s	0.74721	0.417344	0.615021	1.23941	0.0001	10
AIML-2	2022/10/10 15:03:57	thejakasep	6m 36s	0.840422	0.479675	0.385265	2.12222	0.001	10

Figure 2.1.1

2.2 Automation of ML pipeline through CI/CD

Similar to DevOps, it is a process to make frequent able changes by automating the development stages. In MLOps, these steps include not only the code but also the data and hyperparameters of the models as well. The model will be run through the continuous plan, code, build, test, release, deploy, operate and monitor processes as .represented in Figure 2.2.

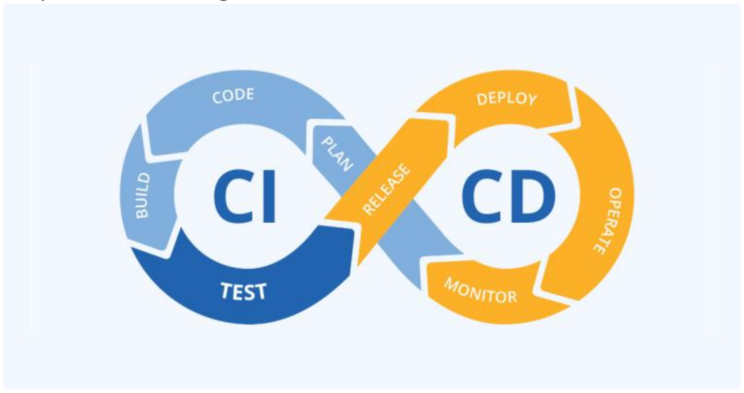
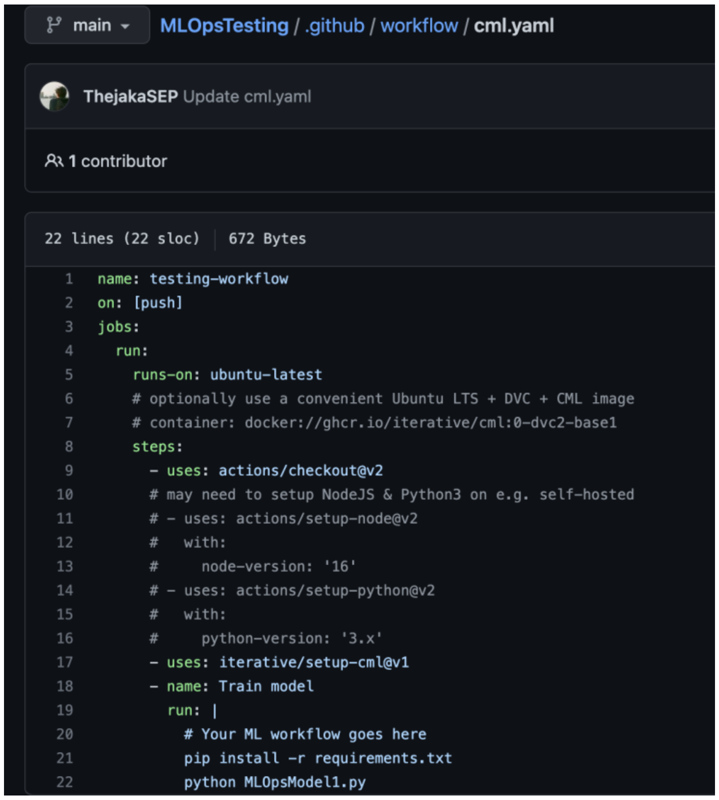


Figure 2.2.1

In this task, we chose to use Github 'Actions' to automate the CI/CD process. Our initial target was to set up an action to run the training and testing of the code once a git push or commit is done for the repository the python file is saved in. The action was defined as represented in Figure 2.2.2 in a workflow which is a configurable automated process that will run one or more jobs.



main / MLOpsTesting / .github / workflow / cml.yaml

ThejakaSEP Update cml.yaml

1 contributor

22 lines (22 sloc) | 672 Bytes

```yaml
name: testing-workflow
on: [push]
jobs:
  run:
    runs-on: ubuntu-latest
    # optionally use a convenient Ubuntu LTS + DVC + CML image
    # container: docker://ghcr.io/iterative/cml:0-dvc2-base1
    steps:
      - uses: actions/checkout@v2
      # may need to setup NodeJS & Python3 on e.g. self-hosted
      # - uses: actions/setup-node@v2
      #   with:
      #     node-version: '16'
      # - uses: actions/setup-python@v2
      #   with:
      #     python-version: '3.x'
      - uses: iterative/setup-cml@v1
      - name: Train model
        run: |
          # Your ML workflow goes here
          pip install -r requirements.txt
          python MLOpsModel1.py
```

Figure 2.2.2

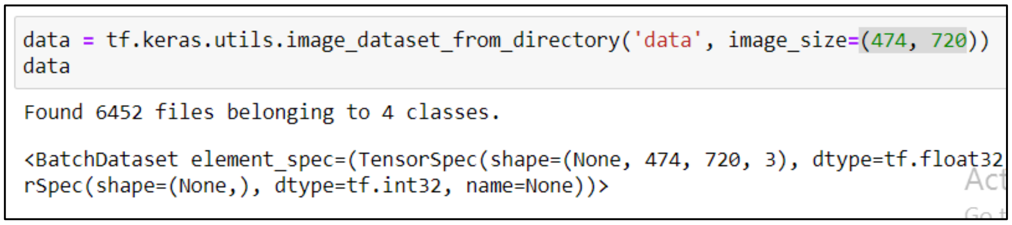
Also, a file named 'requirements' was created to be run along with the model, which contains the required python packages to be installed before the model run. It includes packages such as (TensorFlow, NumPy, pandas, OpenCV, neptuneai, etc). The action was run but encountered with error due to the code needing to access a local dataset repository which cannot be accessed through the workflow. For this matter, we need to find alternatives such as,

- Hosting the dataset through a server
- Having the dataset on google drive and access
- Create a toy dataset on Git (This will only validate the code but not the model accuracy)

Task 3 : Thulana: Improving Model performance – Model 02

Step 1:

Images of the dataset are in different sizes and most of them are 474 x 720. Therefore, the input size of the data pipeline was changed to 474 x 720 (width, height).



```python
data = tf.keras.utils.image_dataset_from_directory('data', image_size=(474, 720))
data
```

```
Found 6452 files belonging to 4 classes.

<BatchDataset element_spec=(TensorSpec(shape=(None, 474, 720, 3), dtype=tf.float32
rSpec(shape=(None,), dtype=tf.int32, name=None))>
```

Figure: 3.1

However model took a long time to train as the number of images and size of each image were high. To get rid of this issue, the number of images was taken down to 700 per class for testing purposes. The corresponding neural network structure and the main parameters are given in the below Table.

Layer	Kernels	Filter Size	Activation Function	Shape / Prob
Input Layer				(474,720,3)
Convo2D	32	3*3	“relu”	
MaxPool2D				
Flatten				
Dense				128
Dropout				0.5
Dense				256
Dropout				0.5
Dense				64
Dropout				0.5
Output Layer				4

Table: 3.1

Parameter	Value
Learning Rate	0.01
Epochs	20
Loss Function	SparseCategoricalCrossentropy (logits = False)

Table: 3.2

The following results were observed.

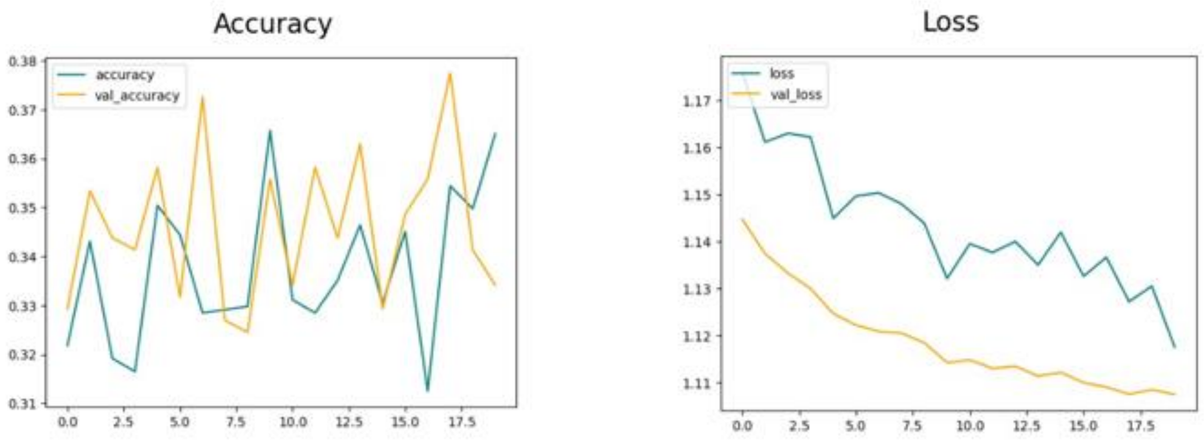
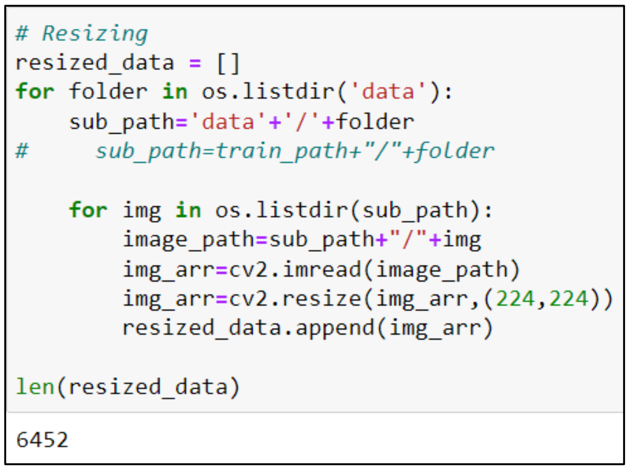


Figure: 3.2

However, the performance of the model has not been increased in terms of accuracy (for both training and validation data)

Step 2:

As mentioned above, "Step 1" did not provide a significant improvement in the model; the images were resized to 224 x 224 size.



```
# Resizing
resized_data = []
for folder in os.listdir('data'):
    sub_path='data'+'/'+folder
#     sub_path=train_path+"/"+folder

    for img in os.listdir(sub_path):
        image_path=sub_path+"/"+img
        img_arr=cv2.imread(image_path)
        img_arr=cv2.resize(img_arr,(224,224))
        resized_data.append(img_arr)

len(resized_data)
```

```
6452
```

Figure: 3.3

The resized images were put into an array. However, those images need to be prepared to feed to the data pipeline by saving those resized images and calling them to the data pipeline.

The model was trained for resized images as follows:

Layer	Kernels	Filter Size	Activation Function	Shape / Prob
Input Layer				(224, 224, 3)
Convo2D	32	3*3	"relu"	
MaxPool2D				
Flatten				
Dense				128
Dropout				0.5
Dense				256
Dropout				0.5
Dense				64
Dropout				0.5
Output Layer				4

Table: 3.3

Parameter	Value
Learning Rate	0.01
Epochs	10
Loss Function	SparseCategoricalCrossentropy (logits = False)

Table: 3.4

The following results were observed after resizing the images:

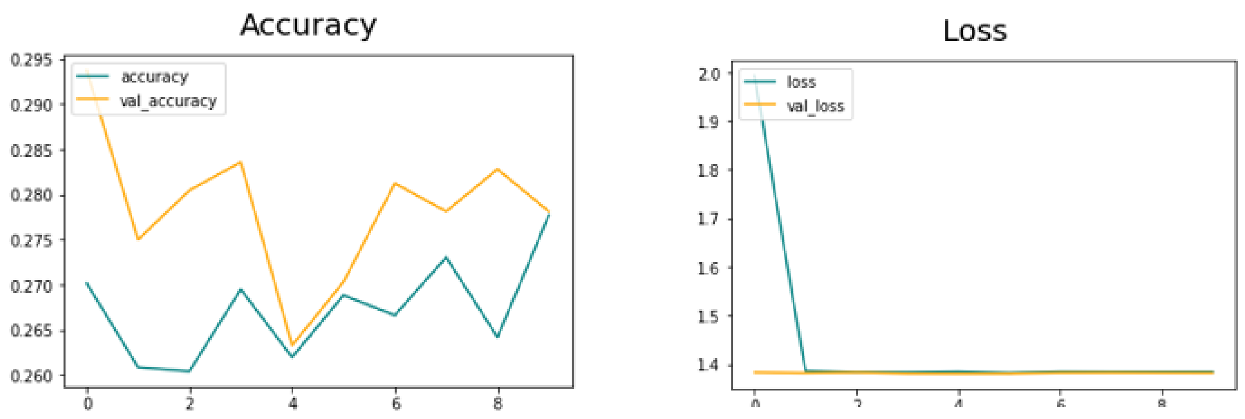
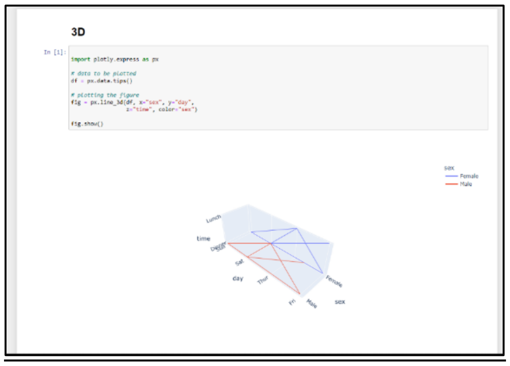


Figure: 3.4

Note: As per the above figure 3.4, improvement in model accuracy has not been observed even after resizing the images. Therefore, image augmentation will be performed in the next step.

Task 4: Jaskaran: Work on complex and 3D charts on plotly.

4.1 Learning basics of plotly - following basic 3D plots were learned for experiment purposes.



3D

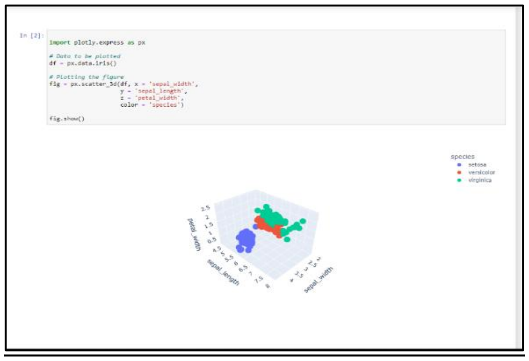
```
In [1]:
import plotly.express as px

# data to be plotted
df = px.data.tips()

# plotting the figure
fig = px.line_3d(df, x="sex", y="day",
                 z="time", color="sex")

fig.show()
```

Figure: 4.1



```
In [2]:
import plotly.express as px

# Data to be plotted
df = px.data.iris()

# Plotting the figure
fig = px.scatter_3d(df, x = 'sepal_width',
                    y = 'sepal_length',
                    z = 'petal_width',
                    color = 'species')

fig.show()
```

Figure: 4.2

Basic knowledge of plotting different types of charts has been learned. Generally worked on 3d plots and 2d plots for comparisons. Learned how to make plots interactive by the use of sliders.

3. Difficulties Encountered in Reporting Week

a. **Deployment on Cloud** - There was some error when trying to run the web application in the local system. But it was resolved with the help of Google.

b. **MLOps** -The workflow created cannot be run due to the dataset existing in the local machine.
Solutions:

- Hosting the dataset through a server
- Having the dataset on google drive and access
- Create a toy dataset on Git (This will only validate the code but not the model accuracy)

c. **Model Implementation** - As the dataset consists of a high number of images, the training and validation process takes a considerable time period (Model 2).

4. Tasks to Be Completed Next Week

Tasks	**Responsible**
Improve structure flask application in and creating web templates for uploading and displaying the image	Tomson
Debugging GitHub workflow to access dataset	Praveen
Image Augmentation	Thulana
Visualize the model performance using plotly	Jaskaran