Malaria Detection Using Convolutional Neural Network

Aparajita Sengupta

Applied Data Science Program
(Massachusetts Institute of Technology)

Capstone project

PROBLEM STATEMENT

Malaria is a devastating disease that killed 627,000 people in 2020 alone.

It is caused by infected female Anopheles mosquito bites that carry Plasmodium parasites.

Traditional method of visual inspection is costly, tedious and time-consuming.

KEY QUESTIONS

1. Can we develop a cost-effective, automated Deep Learning algorithm to identify the malaria-causing parasitized cells?

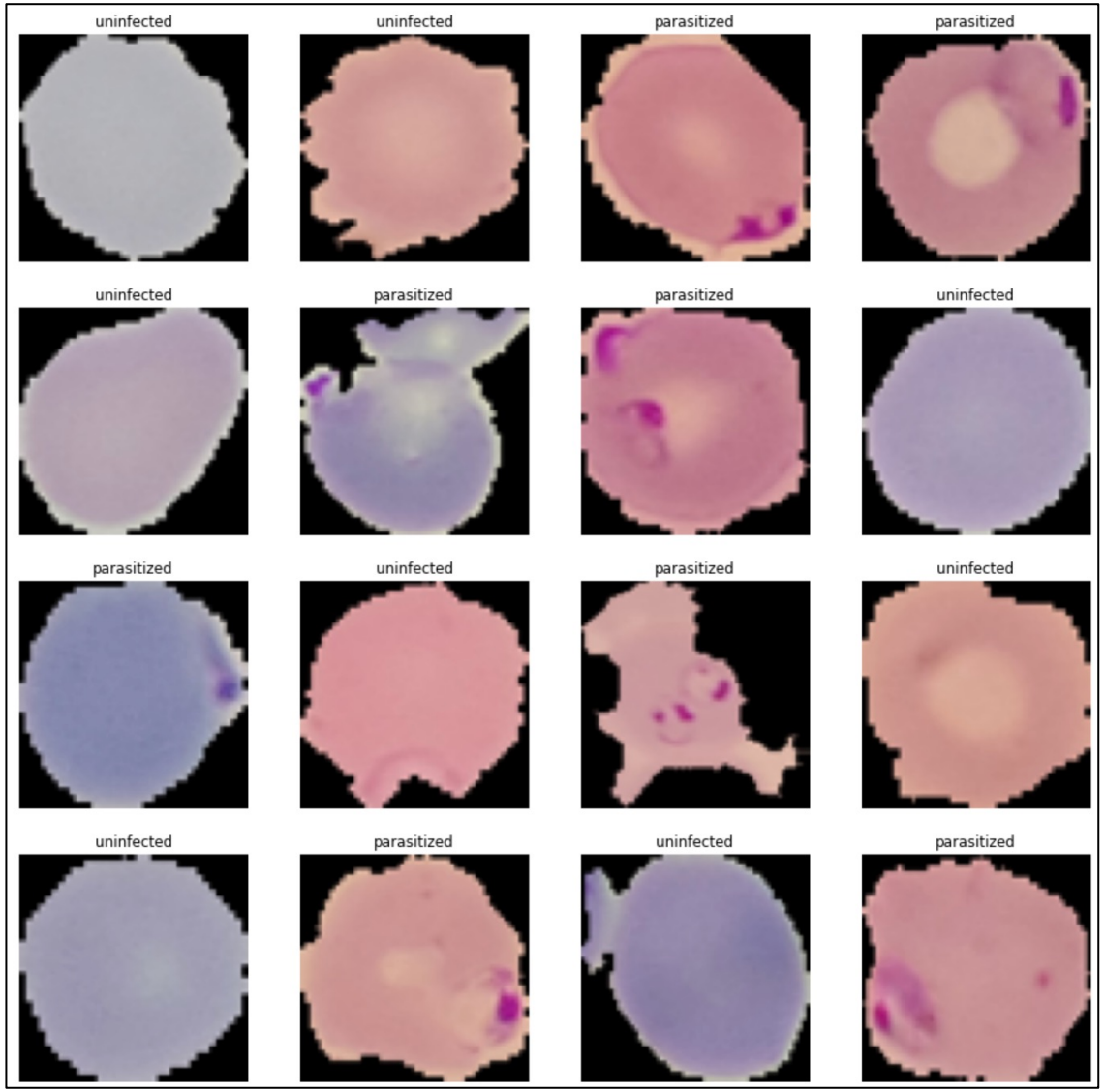
2. Can we do the above with a higher level of accuracy and ease than a manual detection of infected cells?

3. Will our Deep Learning algorithm be scalable for large data sets?

KEY PATTERNS IN THE DATA:

- Visual inspection shows that the parasitized cells have one or more **dark patches** on them, which are missing in the uninfected cells.

- These patches will be our most important identifying features to differentiate between parasitized and uninfected cells.



PROPOSED SOLUTION DESIGN

- Create a base Convolutional Neural Network (CNN) model.
- Add extra layers, change the number of epochs and activation function.
- Use augmented data, Gaussian smoothing, change RGB colors to HSV.
- Use Transfer Learning with added layers.

Why Convolutional Neural Network (CNN)?

- **CNN** is a Deep Learning algorithm that can classify images by using learnable weights and biases.
- CNN is a superior algorithm in terms of **predictability**, **computational time**, economy in **memory usage and cost**, as compared to a fully connected feed-forward neural network as is the case in Artificial Neural Network (ANN).

MODEL COMPARISON:

	ACCURACY	RECALL (parasitized)	PRECISION (parasitized)	time /epoch
MODEL 0	98.07%	98%	98%	~ 4-5 s
MODEL 1	98.11%	98%	98%	~ 5 s
MODEL 2	97.23%	95%	99%	~ 5 s
MODEL 3	98.42%	98%	99%	~ 23-25 s
MODEL 4	95%	96%	95%	-
MODEL 5	90.8%	82%	100%	-

- Note that **Recall** minimized false negatives for parasitized cells because we want to identify correctly as many infected cells as possible.
- False positives are better than false negatives in this case because the patient can undergo further diagnosis to figure out if they are infected.

FINAL MODEL SOLUTION

- ✓ I propose using the **base model** for the classification of infected and parasitized malaria blood cells.
  - ✓ The **accuracy** of the model with augmented images and the base model are comparable (within the error limit).
  - ✓ But the **time required** to run the model with augmented images is 4-6 times longer than that required for the base model.
- ✓ I also propose using an epoch of about 10, because the accuracy converges pretty quickly within the first few epochs.

EXECUTIVE BUSINESS SOLUTION

CNN must be used instead of visually inspecting the blood cells of the patients for malaria parasites in order to increase:

- ✓ Accuracy
- ✓ Cost-effectiveness (by eliminating the need of skilled personnel required for visual inspection).
- ✓ Ease of detection
- ✓ Scalability

All of the above leading to early detection of malaria and saving lives.

EXECUTIVE SUMMARY

- Most of the people who die of malaria are from South America or South Asia where the resources are scarce and there is socio-economic instability (CDC).
- Visually identifying the parasitized cells using microscopy is a costly, tedious and time-consuming process, and depends on the reliability of the laboratory performing the identification.
- The accuracy of rapid diagnostic test results interpreted visually is 82.1% (Kalinga et. al, 2017). Whereas accuracy of our base CNN model is 98.07%.
- In a country like United States, there are about two thousand cases of malaria diagnosed and reported each year. Hence, an average laboratorian does not perform this test regularly, and may not be too proficient.
- In order to reduce the time, cost, error and requirement for human supervision, an automated classification technique (CNN) can be implemented. This has a high accuracy as well as recall value. This can help with the early and accurate detection of malaria and hence save lives in both low-resource as well as high-resource but low occurrence countries.

RISKS & SOLUTIONS

- Risk1: Major **processing lag** in training the models when run on CPUs. For training large number of parameters over increased epochs, the code ran very slowly and/or crashed.
  - Solution: Using **GPU** (Graphics Processing Unit) one can increase the performance of the code. It is a specialized processor with dedicated memory for rendering graphics. I bought access to premium GPU from Google in order to run the codes successfully on Google Colab.
- Risk2: Callbacks are used for early stopping of the training process when the validation loss doesn't improve any longer. However, sometimes this value can get stuck in a **local minima**.
  - Solution: We can use the '**ReduceLROnPlateau**' to change the learning rate if the validation loss doesn't improve after the number of epochs as specified by 'patience'.
- Risk3: **Transfer learning**, although easier to use, do not yield good results for this particular task, even with added layers.
  - Solution: **Use CNN models built from scratch** and tune the hyperparameters to improve the performance.

Thank you!

APPENDIX

Model description in details

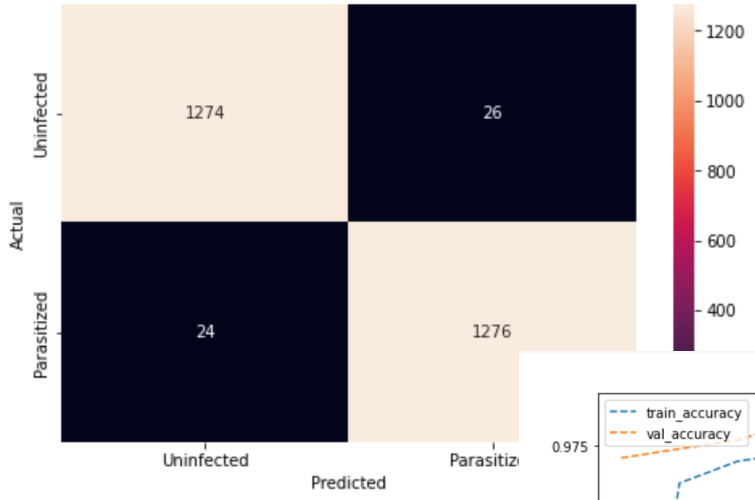
MODEL 0

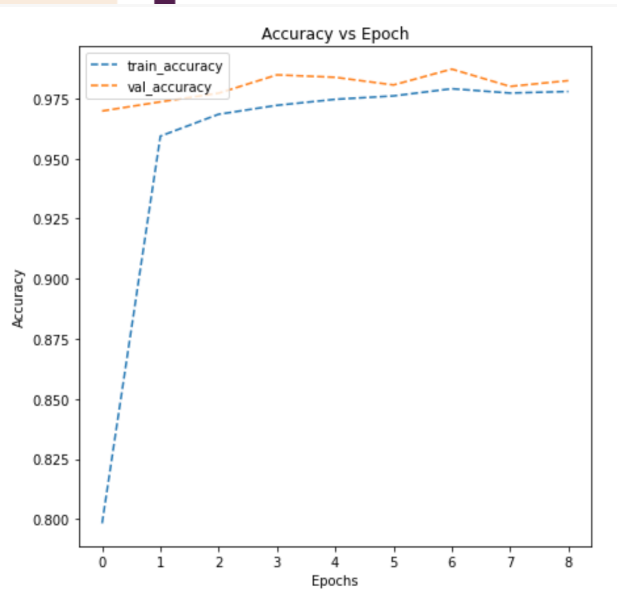
Base CNN model

- The base model contains
  - 3 convolution layers,
  - 3 maxpooling layers,
  - 4 dropout layers.
  - It uses ReLU as the activation function.
  - The final layer has an activation function of softmax which helps in classification problems.
  - The final dense layer has two neurons representing the two classes (uninfected and parasitized).
  - The epoch is 20.

Description of models

	precision	recall	f1-score	support
0	0.98	0.98	0.98	1300
1	0.98	0.98	0.98	1300
accuracy			0.98	2600
macro avg	0.98	0.98	0.98	2600
weighted avg	0.98	0.98	0.98	2600





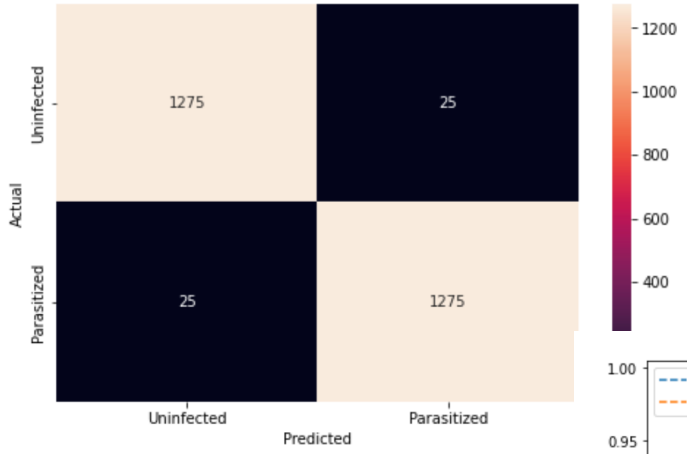
Description of models

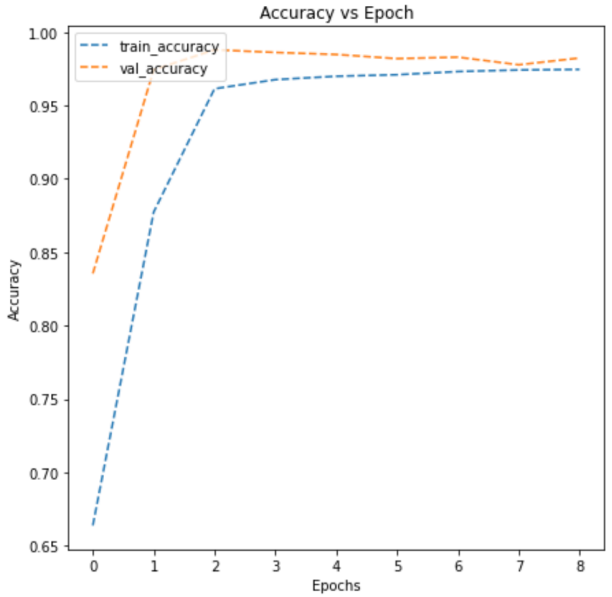
MODEL 1

Adding more layers

- This model contains
  - 4 convolution layers,
  - 4 maxpooling layers,
  - 5 dropout layers.
  - It uses **ReLU** as the activation function.
  - The epoch is 30.

	precision	recall	f1-score	support
0	0.98	0.98	0.98	1300
1	0.98	0.98	0.98	1300
accuracy			0.98	2600
macro avg	0.98	0.98	0.98	2600
weighted avg	0.98	0.98	0.98	2600





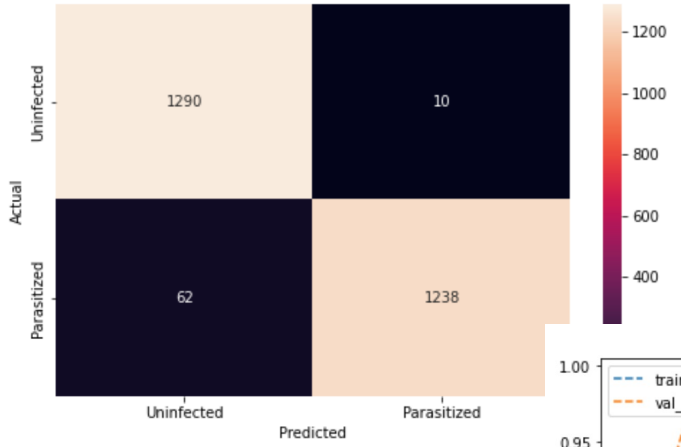
Description of models

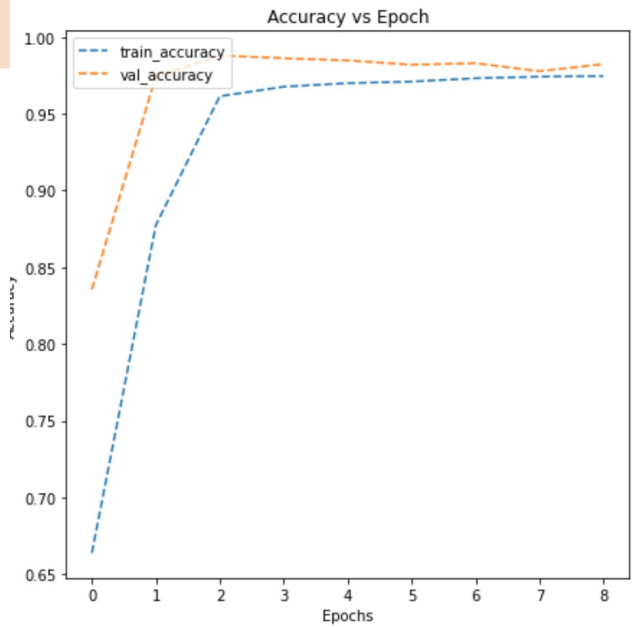
MODEL 2

Changing activation function and adding batch normalization

- This model contains same layers as the base model except
  - It uses **LeakyReLU** as the activation function.
  - And the epoch is **10**.

	precision	recall	f1-score	support
0	0.95	0.99	0.97	1300
1	0.99	0.95	0.97	1300
accuracy			0.97	2600
macro avg	0.97	0.97	0.97	2600
weighted avg	0.97	0.97	0.97	2600





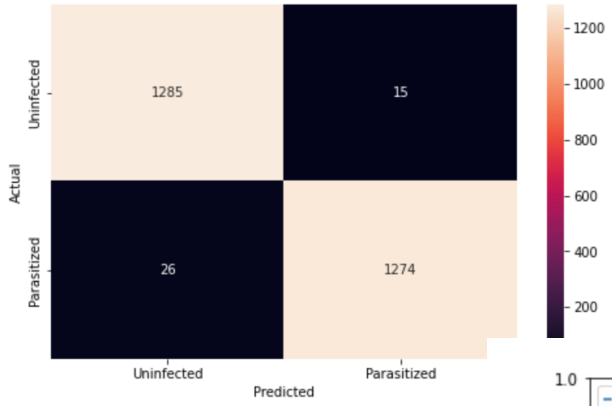
MODEL 3

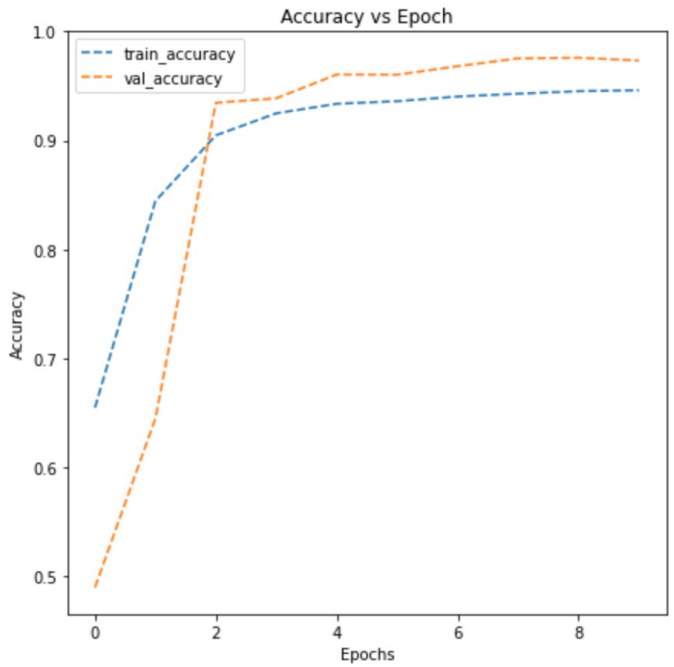
Description of models

Image Data Augmentation

- Data augmentation synthesizes new data from existing data.
- This model contains same layers as the base model except
  - It uses **LeakyReLU** as the activation function.
  - And the epoch is **10**.

	precision	recall	f1-score	support
0	0.98	0.99	0.98	1300
1	0.99	0.98	0.98	1300
accuracy			0.98	2600
macro avg	0.98	0.98	0.98	2600
weighted avg	0.98	0.98	0.98	2600





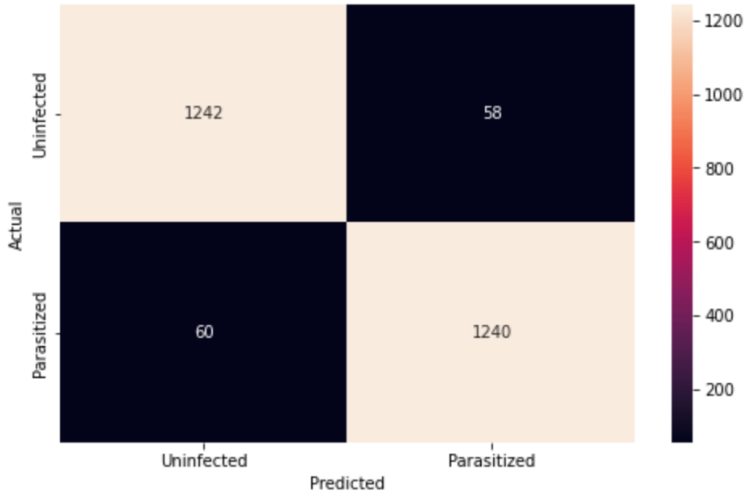
Description of models

MODEL 4

Transfer Learning: VGG-16

- Used **pre-trained model** that has been already created for a similar task.
- Flattened the output from the 3rd block of the VGG16 model.
- Added **two dense layers** with 128 and 64 neurons.
- Added **a dropout layer** and a **batch-normalization layer.**
- It uses **ReLU** as the activation function for all but the final dense layer.
- Epoch is **10**.
- Patience for callback: 5

	precision	recall	f1-score	support
0	0.95	0.96	0.95	1300
1	0.96	0.95	0.95	1300
accuracy			0.95	2600
macro avg	0.95	0.95	0.95	2600
weighted avg	0.95	0.95	0.95	2600



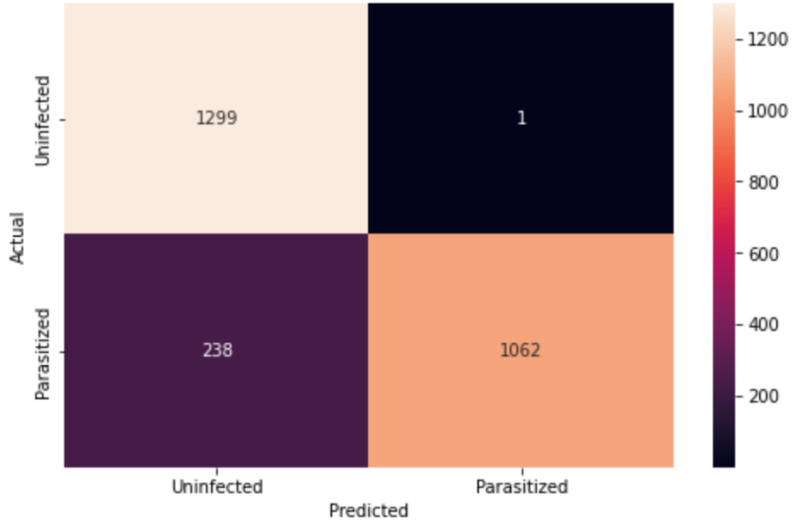
Description of models

MODEL 5

Transfer Learning: VGG-19

- Used **pre-trained model with 19 convolution layer** that has been already created for a similar task.

	precision	recall	f1-score	support
0	0.85	1.00	0.92	1300
1	1.00	0.82	0.90	1300
accuracy			0.91	2600
macro avg	0.92	0.91	0.91	2600
weighted avg	0.92	0.91	0.91	2600



Results and discussions in details

RESULT AND DISCUSSION

MODEL 0

Base CNN model

- ✓ Uses **ReLU** as activation function, and 20 epochs.
- ✓ Already a very **high accuracy** for the test dataset without any image augmentation and smoothing.

MODEL 1

Adding more layers

- ✓ **Increasing convolution layer →**
  - ✓ Increases complexity learnt by the model.
- ✓ **Increasing maxpooling layers →**
  - ✓ Sends only the important data to next layers.
  - ✓ Reduces the number of computations and hence the computational time.
  - ✓ Helpful for a large datasets.
- ✓ **Increasing dropout layer →**
  - ✓ prevents overfitting on the training data
  - ✓ masks the contribution of some features towards the next layer and leaves the rest of the features unmodified
- ✓ **Increasing number of epochs →**
  - ✓ to make sure we are not underfitting the base model.

Discussion

No improvement in the accuracy even after increasing the layers, number of epoch and patience. Hence there is no benefit in increasing the convolution layers and epochs.

RESULT AND DISCUSSION

MODEL 2

Changing activation function and adding batch normalization

- ✓ **Added batch-normalization→**
  - ✓ normalizes the contributions to a layer for every mini-batch.
  - ✓ This reduces the number of training epochs required to train deep neural networks.
- ✓ **Change the activation function →**
  - ✓ replaced ReLU activation function by leakyReLU to fix the dying ReLU problem.
  - ✓ It also helps train the neural network faster.

Discussion

No improvement in accuracy even after adding batch normalization and using leakyReLU as activation function.

RESULT AND DISCUSSION

MODEL 3

Image Data Augmentation

- ✓ For a complex task, we need more learnable parameters. For more learnable parameters we need more data.
- ✓ In order to deal with limited data in real life tasks, we use data augmentation. Data augmentation synthesizes new data from existing data.
- ✓ Three separate data sets have been created using image augmentation where we have used different sets of hyperparameters.
- ✓ These three data sets have been used on the base model, which is the best model so far in terms of accuracy.

Discussion

- Highest accuracy so far.
- Flipside: Run time is almost 4-6 times longer than the other models.

RESULT AND DISCUSSION

MODEL 4

Transfer Learning: VGG-16

- ✓ **Advantage of Transfer Learning:**
  - ✓ One advantage of transfer learning is that we can build on it and introduce additional layers according to our requirements for the task at hand. VGG-16 is **pre-trained model with 16 convolution layer**
- ✓ **Added extra layers**:
  - ✓ Dropout and batchnormalization.
- ✓ **Used ReLU as activation function.**
- ✓ **Reducing epochs:**
  - ✓ We have used 10 epochs for this model since the accuracy converges pretty quickly within a few epochs.

Discussion

The VGG-16 pre-trained model didn't perform as well as the other models for this task.

RESULT AND DISCUSSION

MODEL 5

Transfer Learning: VGG-19

- Used **pre-trained model with 19 convolution layer** that has been already created for a similar task.
- Rest of the layers and parameters are similar to that used in VGG-16.

Discussion

The VGG-19 pre-trained model also didn't perform as well as the other models for this task.

Although the precision is very good for this model, the recall and hence the accuracy is the lowest amongst all the models we tried.