# ResNET50 Image Classification

**CV-02**:
Ritvik RAINA, Suryansh MANAV,
Xingjie FENG and Ngan Yi Ho

## 01 Introduction

A closer look at the problem and different approaches to solve it

## 02 The Model

ResNet50 & customized transfer learning architecture

## 03 Performance

Results and limitations of ResNet50

## 04 Future Development

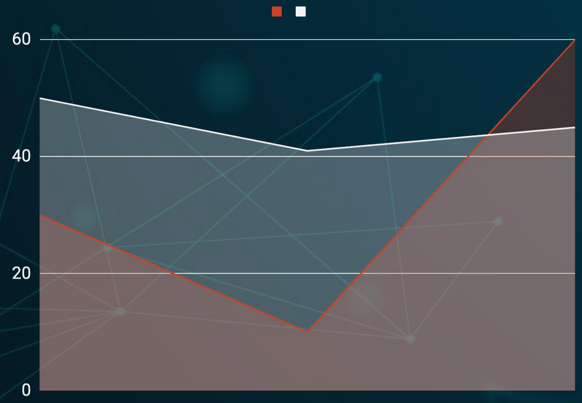
Possibilities after deployment and next steps of improvement

# Introduction 01

A closer look at the problem and different approaches to solve it

# Our Problem Statement

*"Build a Neural Network model which after being fed an image of a fruit can predict to which out the six classes, namely fresh apples, fresh oranges, fresh bananas, rotten apples, rotten oranges, and rotten bananas, it belongs."*




95% traits matches with fresh apple

4% traits matches with rotten apple

# How Does the Data Look Like?

Fresh looking mostly red apples which can be eaten

**Fresh Apples**

Apples with some kind of spots on them and probably cannot be eaten

**Rotten Apples**

Fresh and healthy looking oranges which can be eaten

**Fresh Oranges**

Oranges with dark spots and some fungus, cannot be eaten

**Rotten Oranges**

Fresh looking mostly yellow bananas which can be eaten

**Fresh Bananas**

Bananas with big spots mostly black, cannot be eaten

**Rotten Bananas**

# Initial Thinking and Approach

- After looking at the data for a while we realized that it is an image classification problem with six different labels and we need to build a multi-layer neural network in order to come up with an efficient solution.

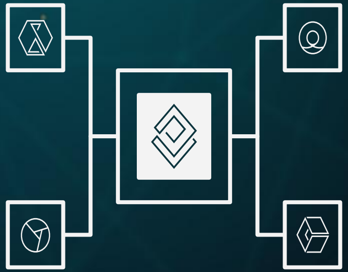
- The next big challenge was to choose and build the most efficient model and we started with 4 different approaches for initial testing.

- Things got complicated when all the models performed relatively good. Some models achieved very high accuracy but took hours for training, whereas some models were great at speed but lacked a bit at the accuracy.

- It was a trade-off situation between speed and accuracy and at last we settled down in between where both the speed and the accuracy were optimal enough.

**(1) 4-layer Conv2D with MaxPool + FC**

**(3) Hybrid Model of (1) & (2)**



**(2) 2-layer Conv2D with MaxPool & BN + FC (steps/epoch limited)**

**(4) Pre-trained ResNet50 + Dropout**

# How We Reached the Final Model?

## Loading Data

Collected the data in drive, mounted it to load in the notebook

## Data Analysis

Explored the data by loading some randomly selected images

## Model Selection

Listed out some helpful models and selected 4 of them for testing

## Model Training

Trained the selected models using different parameters to achieve high accuracy and less training time

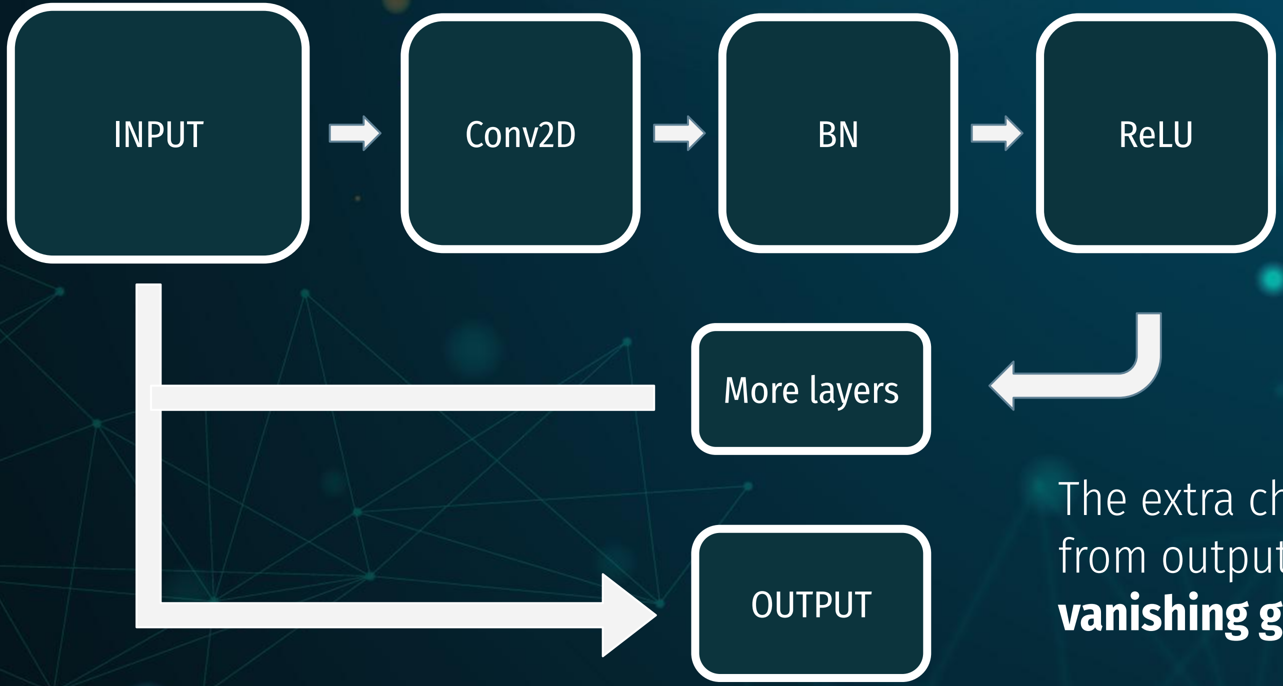
## Model Testing & Validation

Did some graph plotting, made confusion matrix and run some sample tests to choose the best model i.e. ResNET50

# The Model 02

ResNet50 & customized
transfer learning architecture

# Skipped Connections - Crux of ResNets



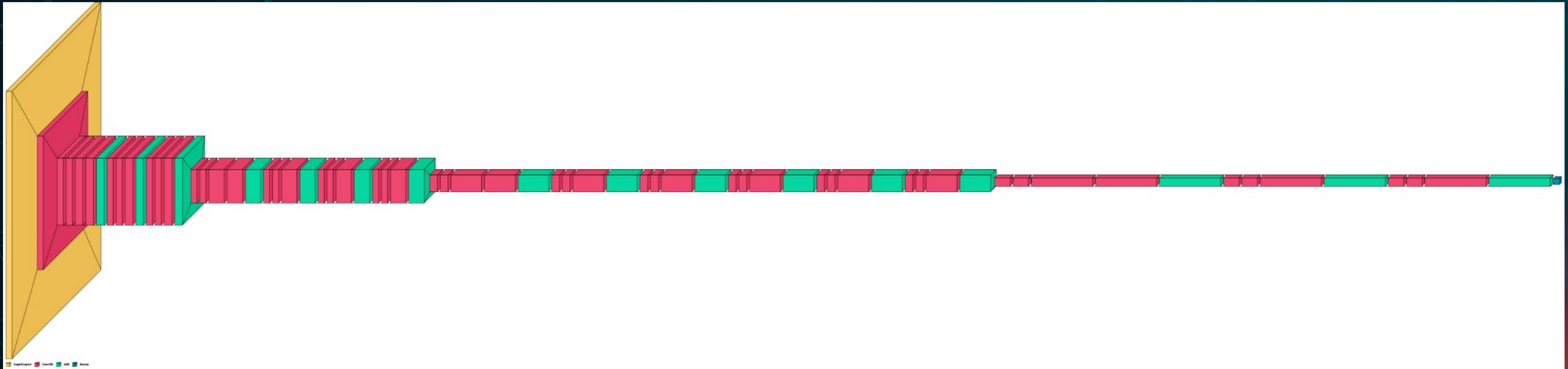

**Vanishing Gradient**:
If initial weights are small in magnitude, or if the model has too many layers, it might fail to update the top layers' weights.

The extra channel connecting input from output alleviates the **vanishing gradient** problem

# Entire ResNet50

**One red block** = Conv2D & Batch Normalization & ReLU

**One green block** = Output from the previous red block
+ Output from the previous green block (add a Conv2D with an identity kernel if dimension reduction is necessary)



(Max pooling, average pooling, flatten, & dropout layers are omitted in the chart)

# Data Loading & Preprocessing

We use **ImageDataGenerator** from tf.keras.preprocessing.image to perform the below functions.

**Train-Validate-Test Split**

Spare away 20% of images from the original training set to validate the model after each training epoch

**Data Augmentation**

Random zoom, random rotate, random shear, random;
Only conducted for the training and validation sets

**Batch Size**

All images are packed into batches with size 32

**Image Size**

224 by 224 pixels per image

# Model Rundown

Data preprocessing

⬇

ResNet50 (with top layers frozen)

⬇

Flatten & Dropout

⬇

Dense (for output)

Total params: 24,189,830
Trainable params: 24,136,710
Non-trainable params: 53,120

**Hyperparameters Tuning**

Decrease **steps per epoch to 50** for faster training;
Freeze the **top 2 layers** of ResNet50;
Set **dropout rate to 0.5** for the dropout layers;
Assign **RMSProp** as the optimizer with **learning rate 0.0001**

# Transfer Learning

Compared to the transfer learning algorithm instructed on Day 6 of DTT, our model is designed differently

## The Instructed Model

*Freezes all base model layers;

*Trains the surrounding layers;

*Unfreezes some layers but leaves many top layers frozen;

*Trains the base model

## Our Model

*Freezes a few top layers of the base model;

*Trains the entire model all at once

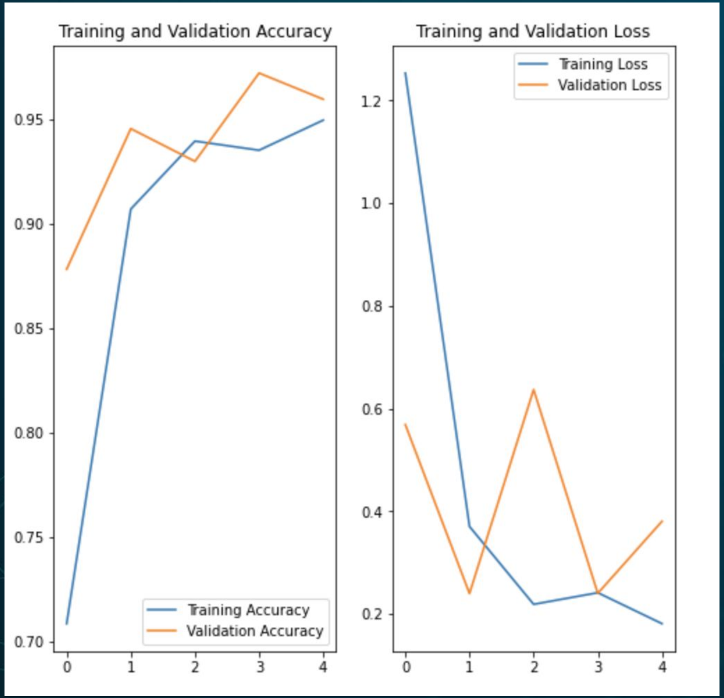
**Why?**

- More computational power with premium GPU
- Fast training due to limited steps per epoch
- ResNets with pre-trained weights are quick to fine-tune

# Performance 03

Evaluation and limitation of ResNet50

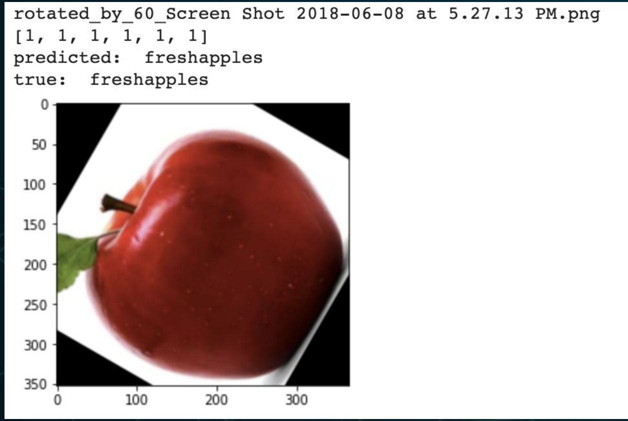
# Training and Test Accuracy

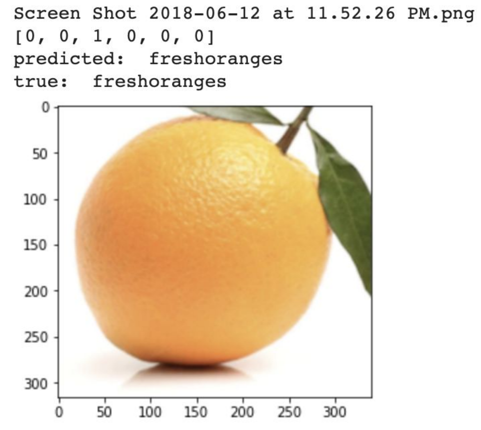


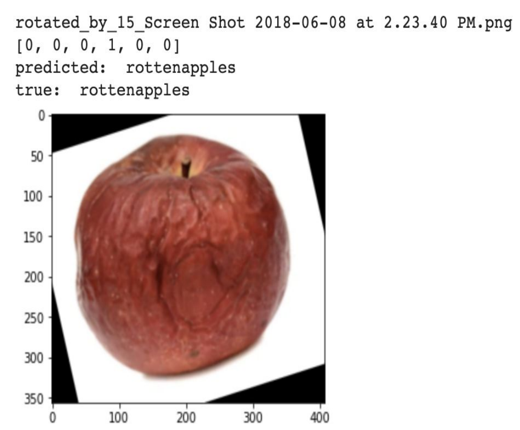

As the training and validation accuracies are close to each other, our model is not over-fitted just for the training data and works well with testing and out-of-the-world data.

On average the test accuracy is above 93%!!!

# Sample Predictions

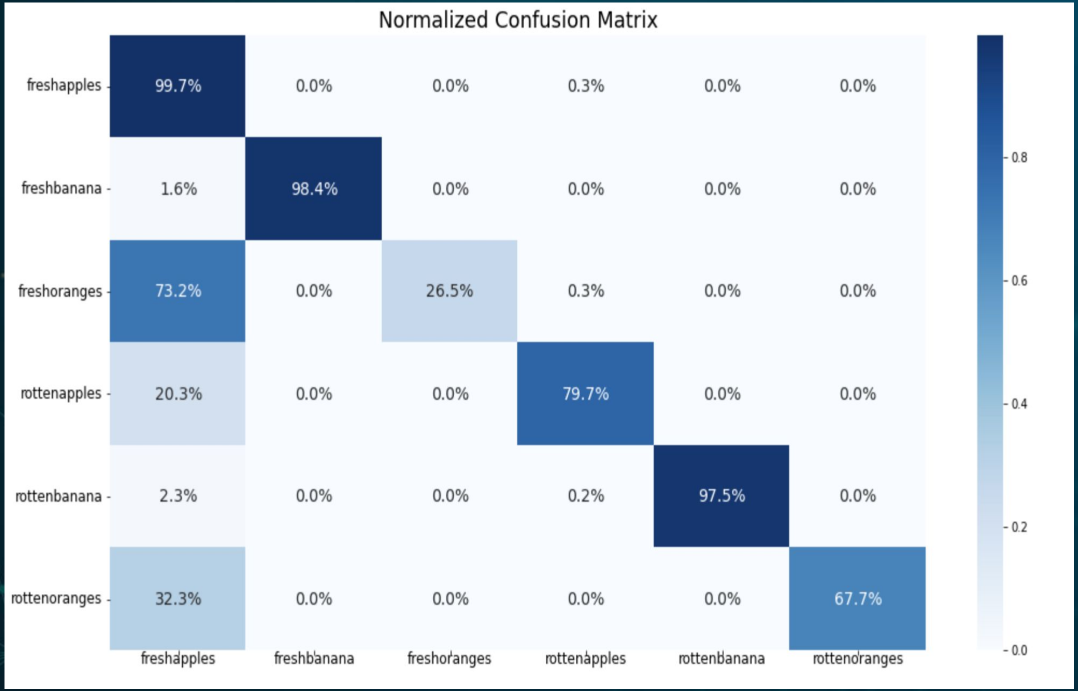



These are some sample predictions with their predicted and true labels.

# Confusion Matrix




By looking at the confusion matrix, we can say that our model was confused by apple and orange sometimes, but works well in identifying bananas.

# Limitations

- Our model may require a lot more data for classifying  6 classes

```
In train_set
freshapples :     1355
freshbanana :     1281
freshoranges :    1173
rottenapples :    1874
rottenbanana :    1780
rottenoranges :   1276
```

# Limitations

- Our model might be confused and mixed fruits that are red and green with fresh apples.

Also, green rotten areas of an orange would sometimes look like green leaves, resembling a fresh one. That’s why 32.3% of rotten oranges are ‘fresh’.

# Future Development

# 04

Possibilities after deployment and next steps of improvement

# WHAT DO WE WANT IN THE FUTURE?

- Less computing time i.e. fast training of the model
- Optimization of the model to make it more general
- Increased accuracy
- Increased flexibility and modularity

# HOW CAN WE ACHIEVE IT?

### HYPERPARAMETER TUNING

By tuning some simple parameters such as Batch Size, Kernel, etc. we successfully achieved the accuracy of around 95%, but we want to take it on another level by tuning more parameters in the future

### MERGING DIFFERENT MODELS

We look forward to try and test different transfer learning models until we settle down for the best

### AVOIDING OVERFITTING & FEEDING MORE GENERAL DATA

We do not want our model to work only with our data, rather we want it to be a general one which can work accurately for a wide range of data right after a simple training

# REFERENCES


- Understanding ResNet50 architecture. Aakash Kaushik.
  https://iq.opengenus.org/resnet50-architecture

- Understanding and Coding a ResNet in Keras. Priya Dwivedi.
  https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

- Study of Residual Networks for Image Recognition. Mohammad S Ebrahimi & Hossein K Abadi.
  https://arxiv.org/abs/1805.00325