Petals to the Metal

Flower Classification

“The Machine Learning Mavericks”

Team: Cindy Chen & Aiden Gray
Mentor: Tarek Radi

Background:

Objective: Classifying flowers into 104 flower types on Intel's Developer Cloud (using CPUs), Locally (using CPUs) and on Kaggle (using TPUs)

Data input format: Binary TFRecord format (includes picture and labels)

Datasets

1. **Training dataset:** 12,753 Training images predict the outcome you design your model to predict.
   Datasets size: 192 * 192, 512 * 512
2. **Validation dataset:** 3,712 validation images w / labels intended to calculate the model's performance.
3. **Test dataset**: 7,382 unlabeled test images to predict flower classification.

Workflow Strategy

01

Loading datasets

- Import libraries
- Distribution Strategy (detect TPUs, if any)
- Create data pipeline

02

Data Preprocessing

- Scaling the data
- Data augmentation

03

Explore data

- Train the model
- Evaluate the model
- Analyze the confusion matrix

04

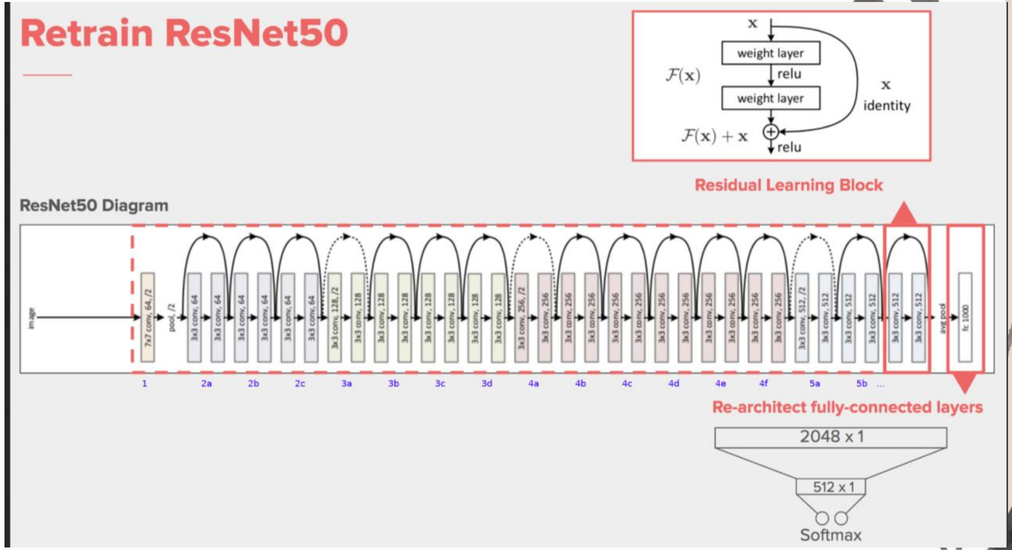
Visual validation

- Display batch of images
- Print predictions

ResNet50 Model

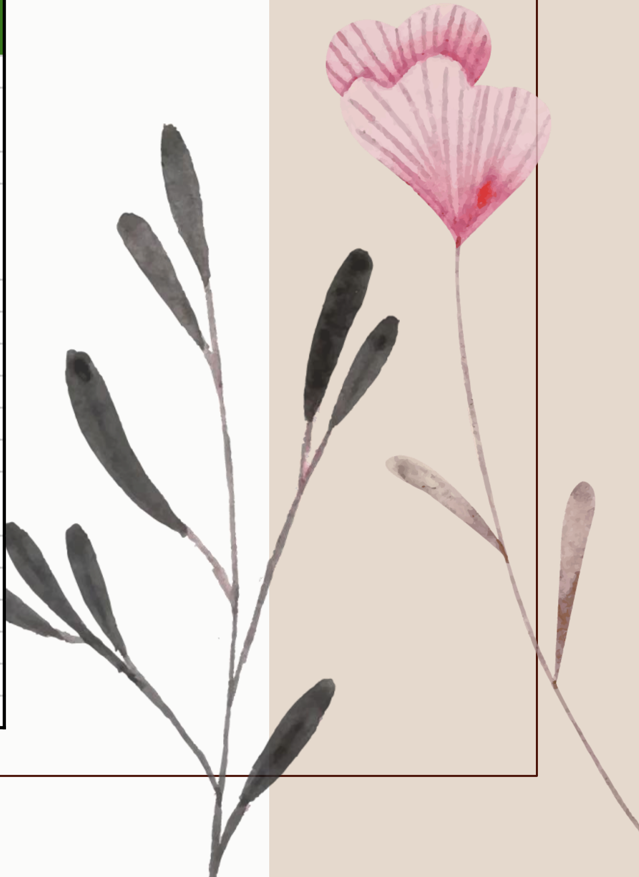
TensorFlow Keras is a high-level neural network API for building and training deep learning models.
ResNet-50 is a convolutional neural network that is 50 layers deep and it can load a pre-trained version of the network trained on more than a million images from the ImageNet database.
ResNet 50V2 is considered to be a more accurate and robust model than the original ResNet50.

General Parameters used:
EPOCHS = 90
BATCH_SIZE = 128
NUM_TRAINING_IMAGES = 12753
STEPS_PER_EPOCH = 10



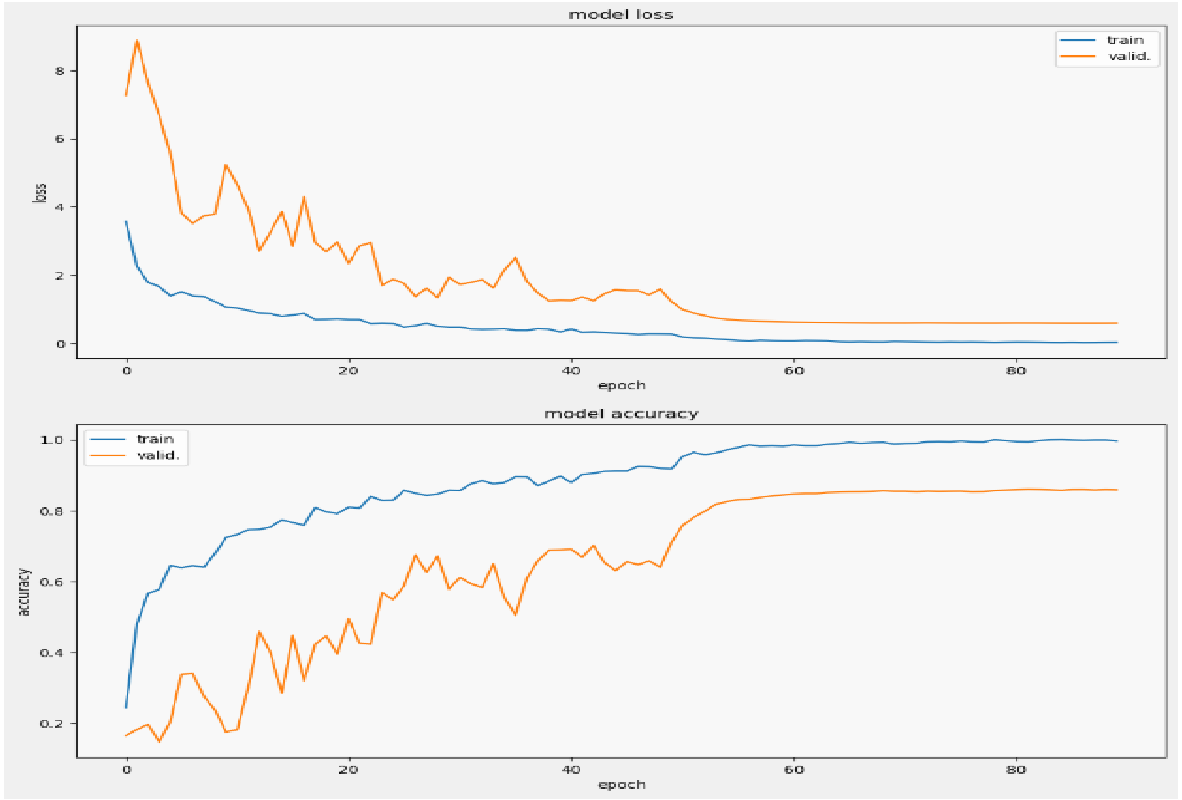
Intel DevCloud

Parameters	Checkout Point 1		Checkout Point 2		Final Presentation	
Owner	Cindy	Cindy	Cindy	Cindy	Cindy	Cindy
Execution Location	DevCloud	DevCloud	DevCloud	DevCloud	DevCloud	DevCloud
Model	ResNet50	ResNet50	ResNet50	ResNet50V2	ResNet50V2	ResNet50V2
Callbacks	EarlyStopping	LearningRate Scheduler	ReduceLROnPlateau	ReduceLROnPlateau_callback	ReduceLROnPlateau_callback	ReduceLROnPlateau_callback
Epochs	90	90	90	90	90	90
	X	X	X	X	X	X
	-	-	-	X	X	X
Image prep	-	-	-	X	X	X
	-	-	-	X	X	X
	-	-	-	X	-	X
Steps per epoch	10	10	10	10	10	10
Image size	192x192	192x192	192x192	192x192	192x192	192x192
BATCH SIZE	16 * 8	16 * 8	16 * 8	16 * 8	16 * 8	16 * 8
Optimizer	Nadam	Nadam	Nadam	RMSprop	Nadam	Nadam
Recall	NA	0.625	0.686	0.832	0.836	0.84
Precision	NA	0.764	0.833	0.861	0.872	0.869
F1 score	NA	0.625	0.717	0.841	0.848	0.849

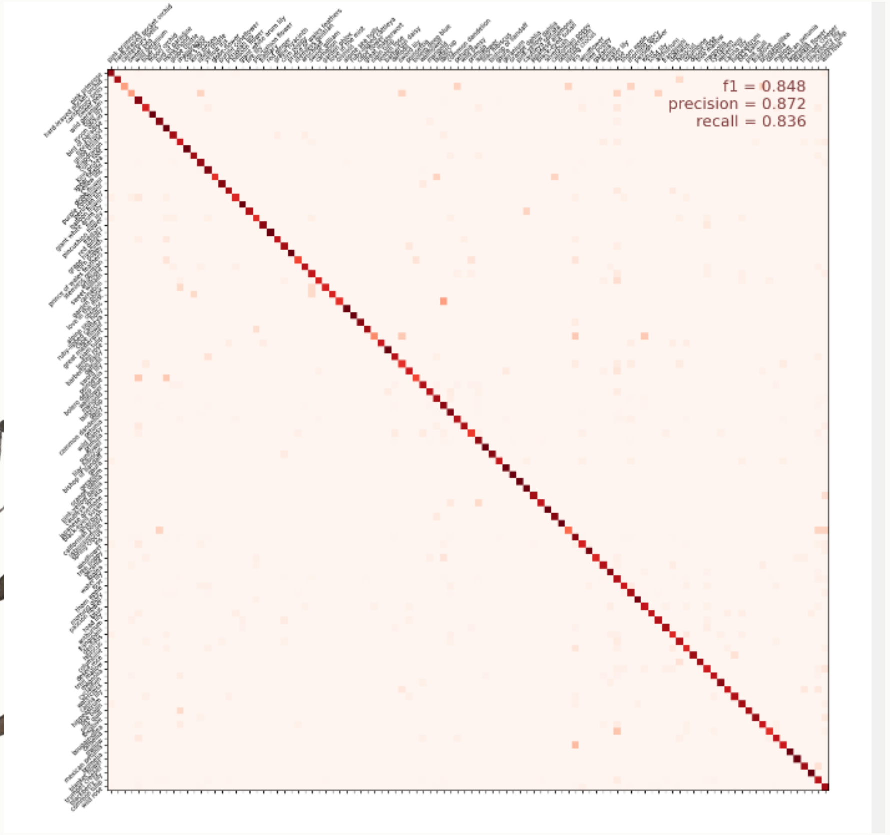


ResNet 50 V2 Model loss & Model accuracy

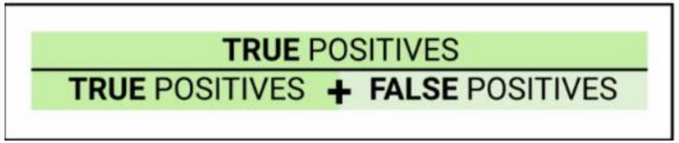
(Intel DevCloud)



ResNet 50 V2 Confusion Matrix

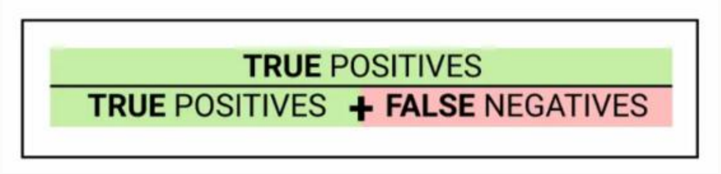


- Precision = **0.872**



$$\frac{\textbf{TRUE POSITIVES}}{\textbf{TRUE POSITIVES} + \textbf{FALSE POSITIVES}}$$

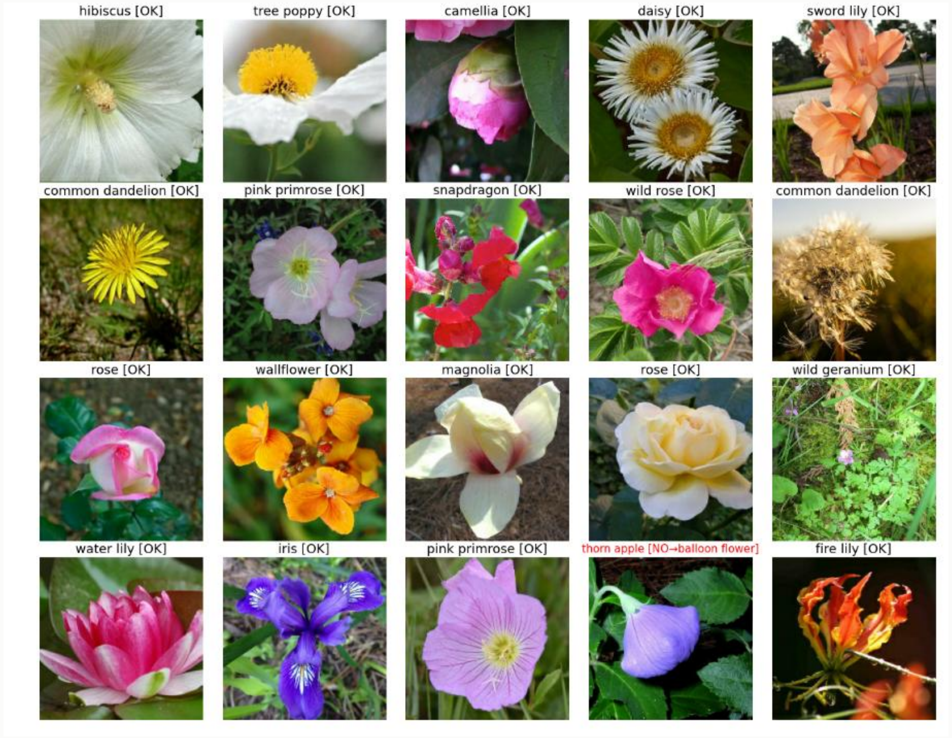
- Recall = **0.836**



$$\frac{\textbf{TRUE POSITIVES}}{\textbf{TRUE POSITIVES} + \textbf{FALSE NEGATIVES}}$$

- F1 – Score = **0.848**
  Balances the Precision and Recall score:
  F1 = 2 * (Precision * Recall) / (Precision + Recall)

Visual Validation



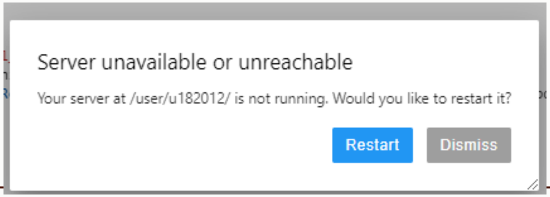
Limitations:

No TPU only CPU

The model takes a long time to run [~3 hours] and accuracy was not as high as we hoped

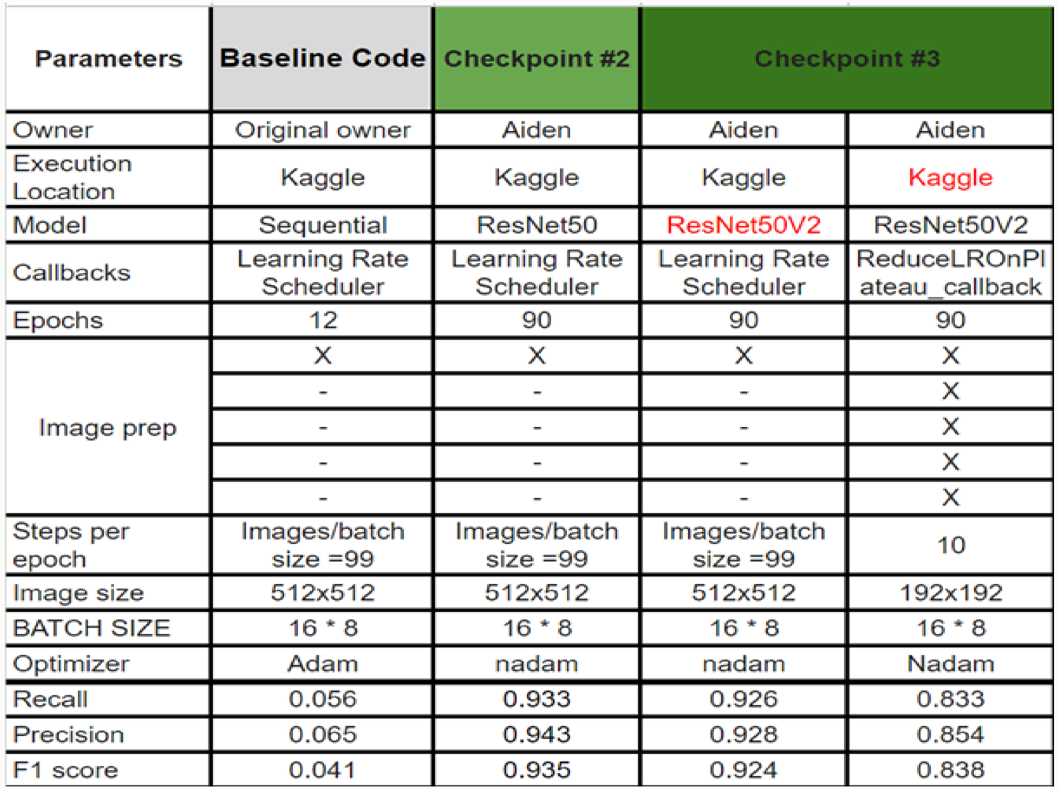
Sever stop running after certain hours

Epochs = 90 is the highest epoch

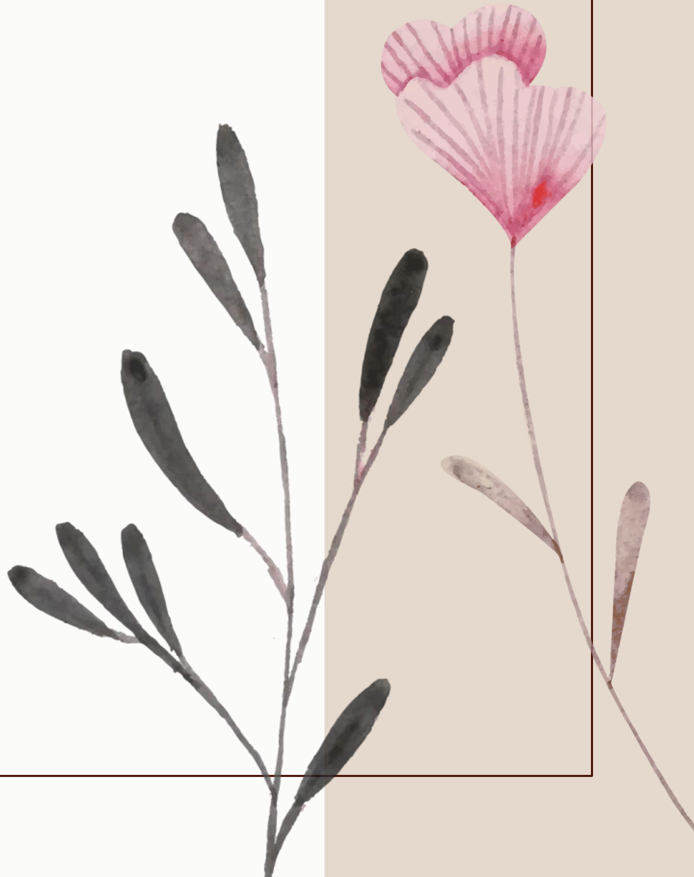

Server unavailable or unreachable

Your server at /user/u182012/ is not running. Would you like to restart it?

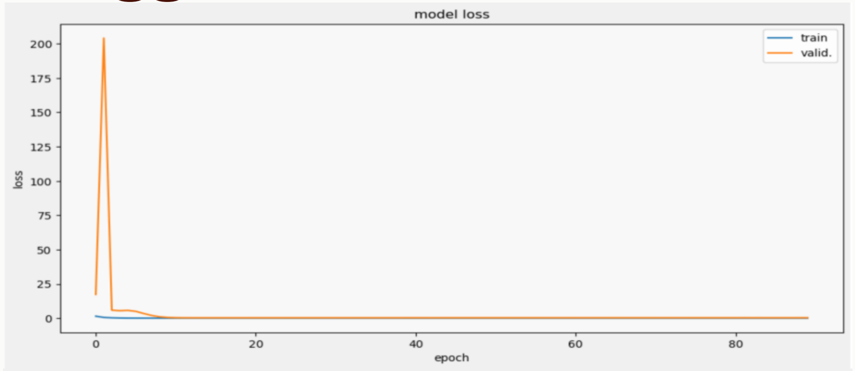
Kaggle

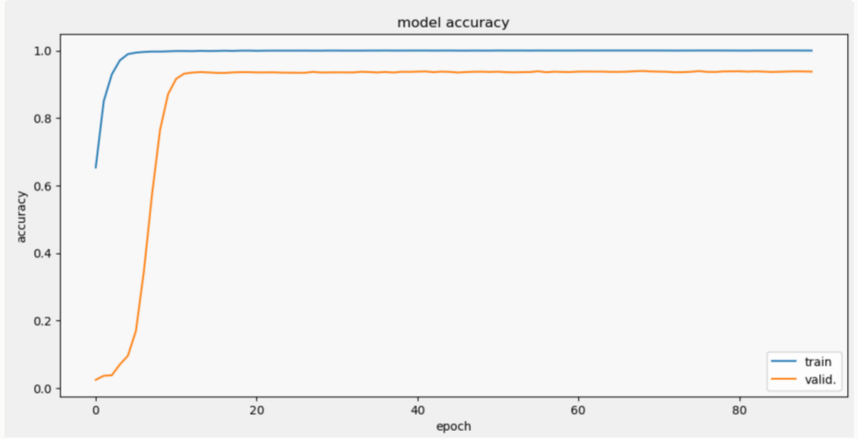


Parameters	Baseline Code	Checkpoint #2	Checkpoint #3	
Owner	Original owner	Aiden	Aiden	Aiden
Execution Location	Kaggle	Kaggle	Kaggle	Kaggle
Model	Sequential	ResNet50	ResNet50V2	ResNet50V2
Callbacks	Learning Rate Scheduler	Learning Rate Scheduler	Learning Rate Scheduler	ReduceLROnPlateau_callback
Epochs	12	90	90	90
Image prep	X	X	X	X
	-	-	-	X
	-	-	-	X
	-	-	-	X
	-	-	-	X
Steps per epoch	Images/batch size =99	Images/batch size =99	Images/batch size =99	10
Image size	512x512	512x512	512x512	192x192
BATCH SIZE	16 * 8	16 * 8	16 * 8	16 * 8
Optimizer	Adam	nadam	nadam	Nadam
Recall	0.056	0.933	0.926	0.833
Precision	0.065	0.943	0.928	0.854
F1 score	0.041	0.935	0.924	0.838



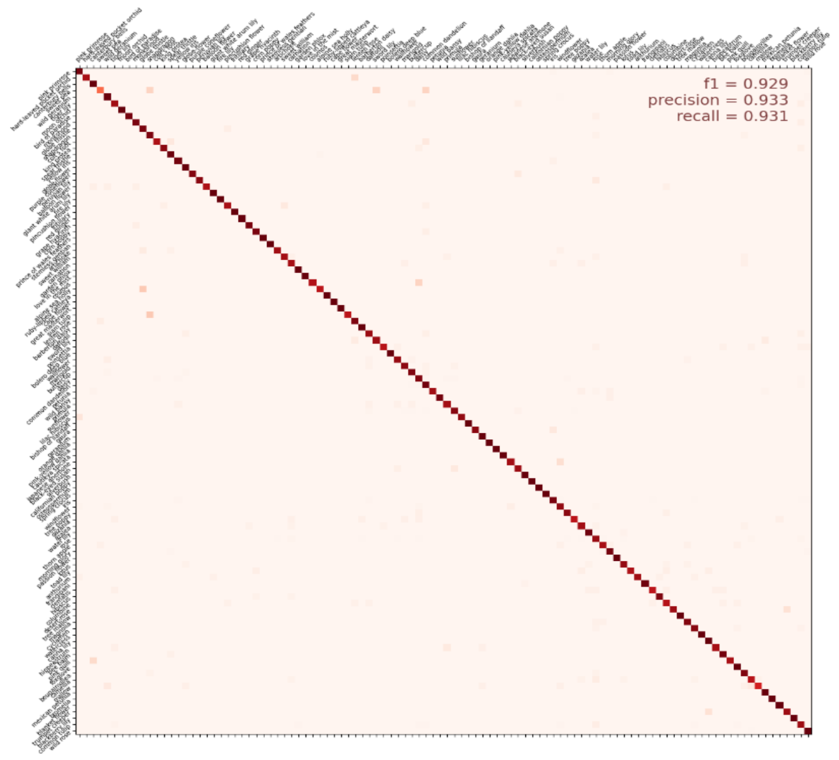
ResNet 50 Model loss & Model accuracy (Kaggle)



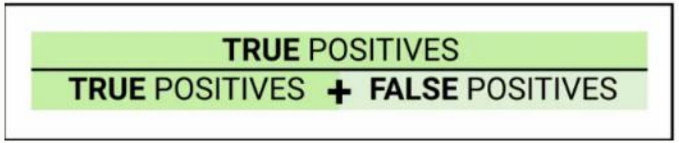


Epoches: 90
Batch Size: 99
Callback: LearningRateScheduler
Image Size: 512 x 512
TPUs

ResNet 50 Confusion Matrix (Kaggle)

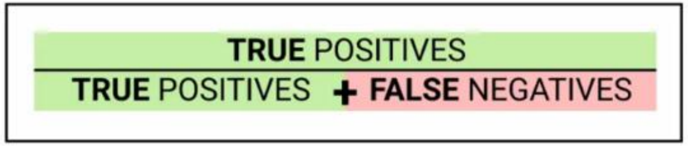


- Precision = **0.933**



$$\frac{\textbf{TRUE POSITIVES}}{\textbf{TRUE POSITIVES} + \textbf{FALSE POSITIVES}}$$

- Recall = **0.931**



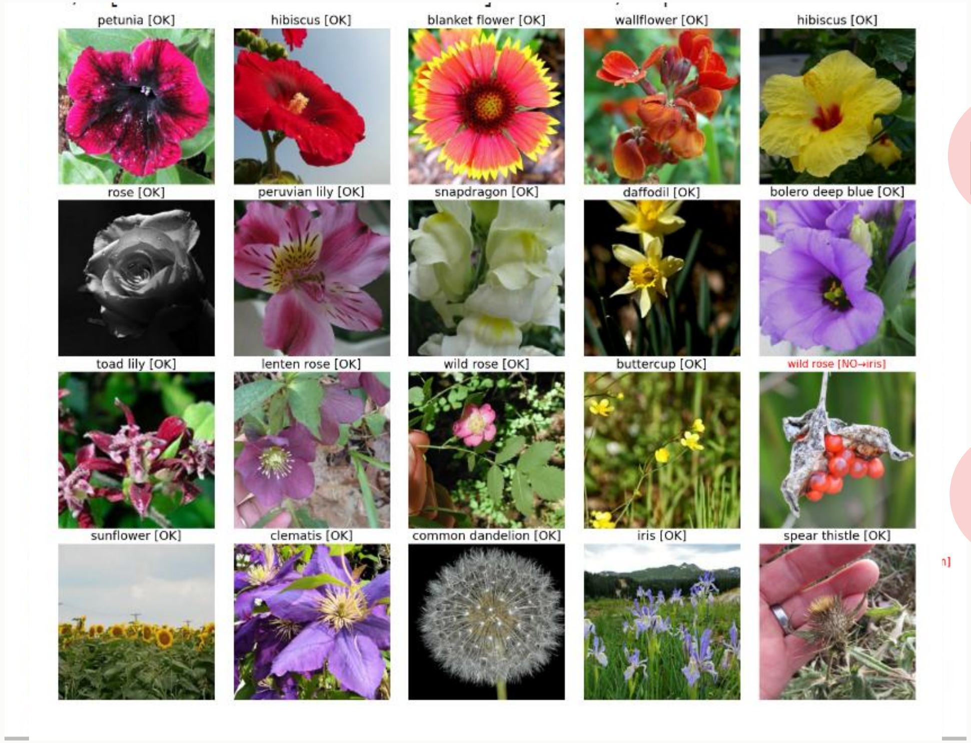
$$\frac{\textbf{TRUE POSITIVES}}{\textbf{TRUE POSITIVES} + \textbf{FALSE NEGATIVES}}$$

- F1 – Score = **0.929**

Balances the Precision and Recall score:

F1 = 2 * (Precision * Recall) / (Precision + Recall)

Visual Validation



Limitations:

TPU

The model takes 45 minutes to run and accuracy is higher than through Intel DevCloud

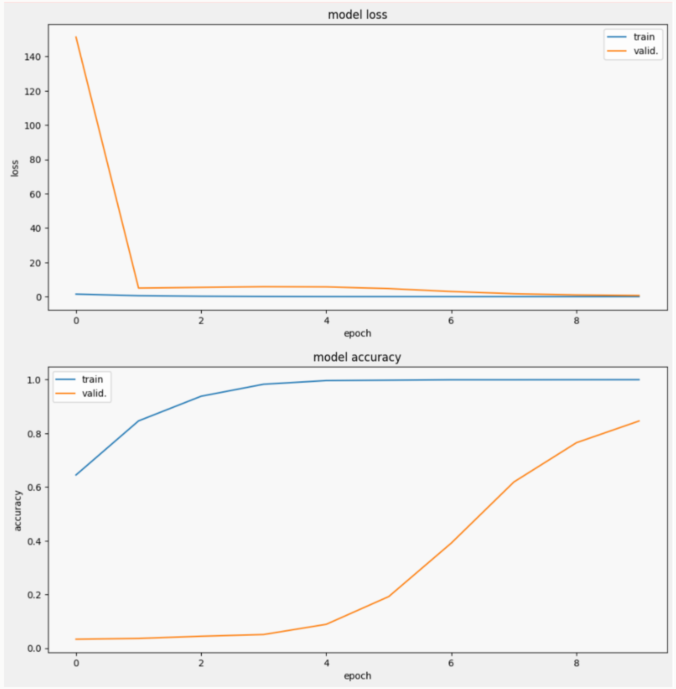
Server has not stopped yet

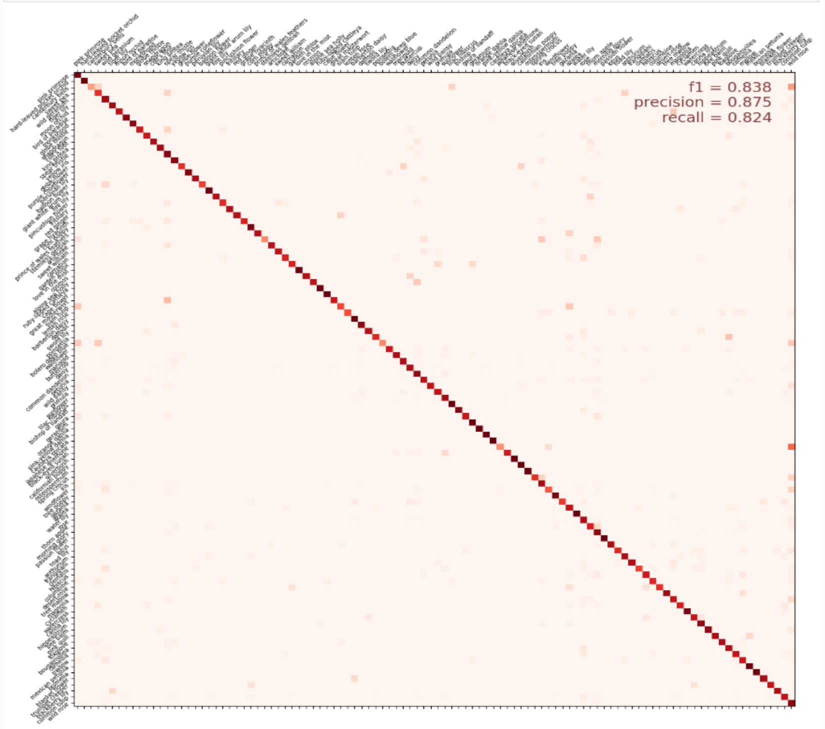
Epochs = 90 is the highest tried

Local Server Experimentation

- Model: Resnet50
- Callback: Learning Rate Scheduler
- Epochs: 10
- Image Size: 192x192

ResNet 50 Model loss & Model accuracy & Confusion Matrix (Local)





Limitations of local servers

1. It cannot handle large image sizes
   - 512 x 512
2. It takes too long per epoch to run the code multiple times
   - 13 min per epoch

Learning Journey - Aiden

Start

MO:
None

Progress

Now

MO:
Read documentation
Study others' codes
Write code
Test and repeat

Learning Journey Cindy

Start

MO:

Python Beginner

Don’t know ML

Challenge

Progress

MO:

Research

Build ML model from the beginning

Don’t give up!!

Now

MO:

Build ML

Improve model accuracy

Growth mindset

Thank You

Appendix:

ReduceLROnPlateau callback:

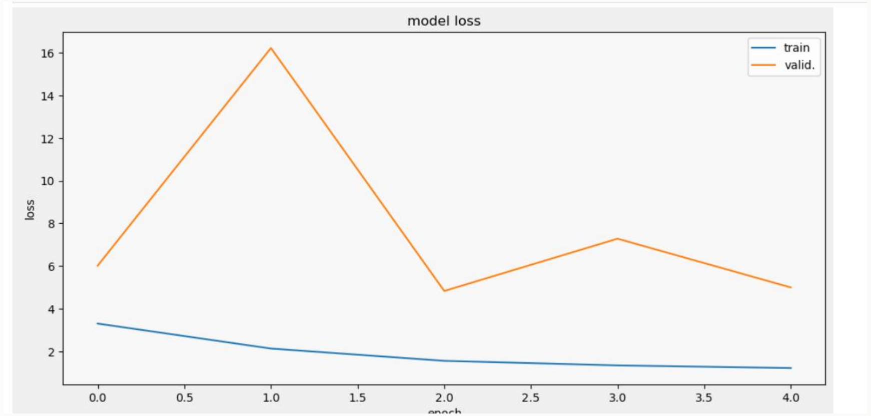
```
ReduceLROnPlateau_callback = 
tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=10, 
verbose=0, mode='auto', min_delta=0.0001)
```

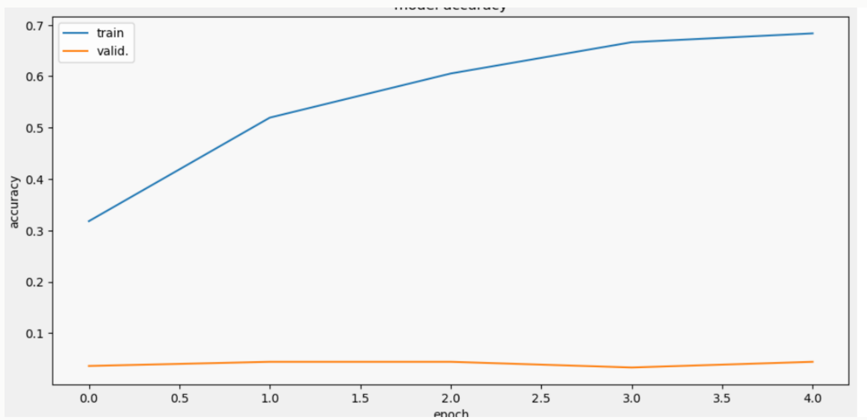
The above code snippet says that if the val_loss has not improved (the lower the better) for 10 epochs then it will change the learning rate to its 1/10 th value.

Appendix:

Earlystop callback:

```
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0, patience = 2, mode='auto')
```

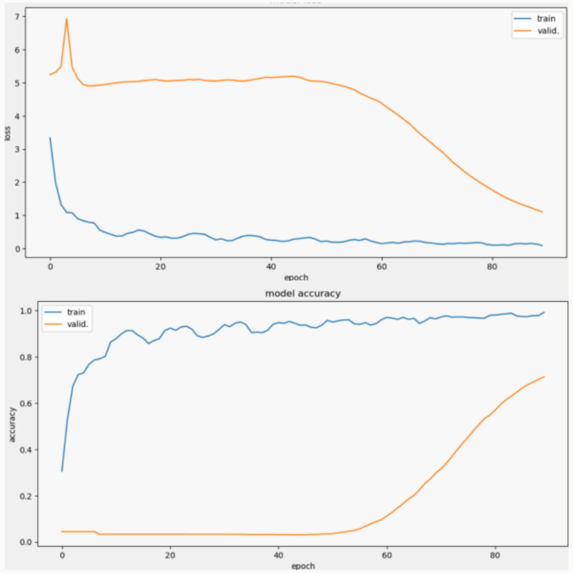


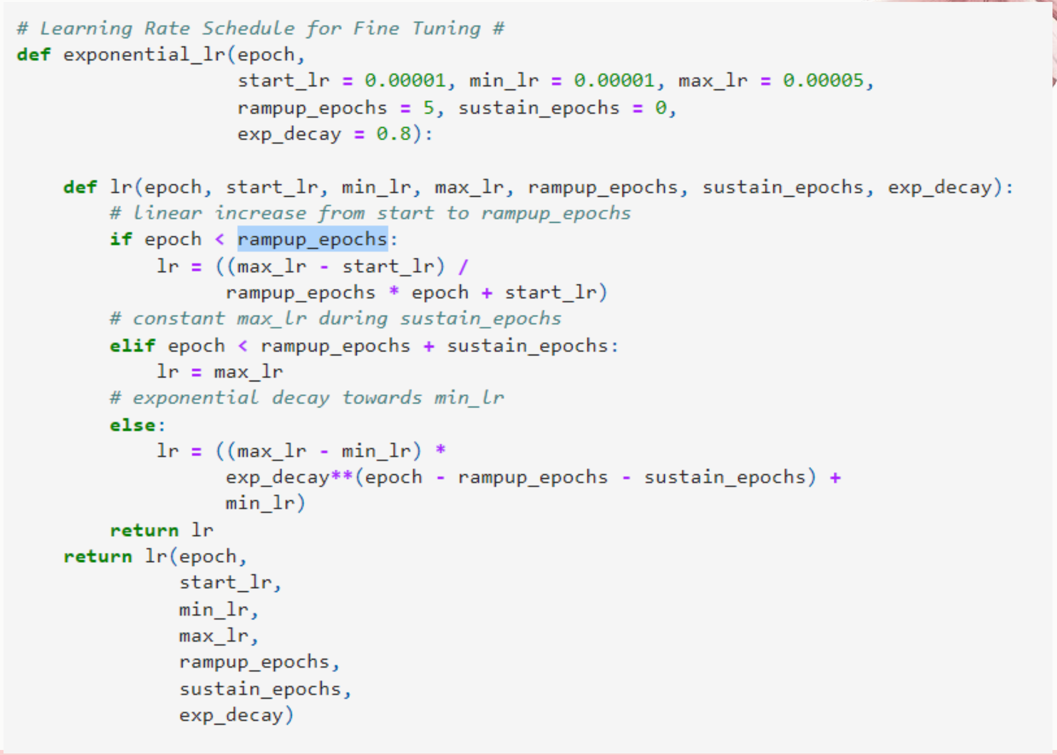


Appendix:

lr callback:

lr_callback = tf.keras.callbacks.LearningRateScheduler (exponential_lr, verbose=True)





```
# Learning Rate Schedule for Fine Tuning #
def exponential_lr(epoch,
                   start_lr = 0.00001, min_lr = 0.00001, max_lr = 0.00005,
                   rampup_epochs = 5, sustain_epochs = 0,
                   exp_decay = 0.8):

    def lr(epoch, start_lr, min_lr, max_lr, rampup_epochs, sustain_epochs, exp_decay):
        # linear increase from start to rampup_epochs
        if epoch < rampup_epochs:
            lr = ((max_lr - start_lr) /
                  rampup_epochs * epoch + start_lr)
        # constant max_lr during sustain_epochs
        elif epoch < rampup_epochs + sustain_epochs:
            lr = max_lr
        # exponential decay towards min_lr
        else:
            lr = ((max_lr - min_lr) *
                  exp_decay**(epoch - rampup_epochs - sustain_epochs) +
                  min_lr)
        return lr
    return lr(epoch,
              start_lr,
              min_lr,
              max_lr,
              rampup_epochs,
              sustain_epochs,
              exp_decay)
```