

## **Flower Classification Using Deep Learning Methods on TPUs**

*(A deep learning approach in search of a cost-effective and efficient model  
for classification of 102 types of flower images on TPUs)*

**Author: Mudiha Wazirali**  
**Mentor: Dr. Mhoon**

**University of Houston - Downtown**

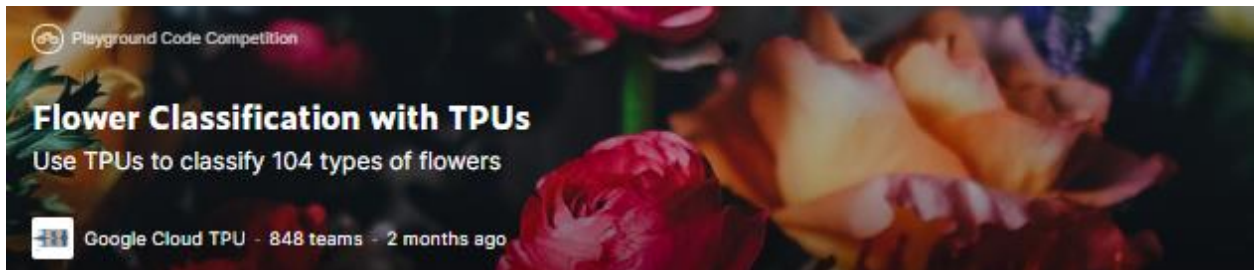
## Abstract

There are over 5,000 species of mammals, 10,000 species of birds, 30,000 species of fish, and over 400,000 different types of flowers. TPUs are powerful hardware accelerators specialized in deep learning tasks. They were developed (and first used) by Google to process large image databases, such as extracting all the text from Street View. The latest Tensorflow release (TF 2.1) was focused on TPUs and they're now supported both through the Keras high-level API and at a lower level, in models using a custom training loop. The point of TPUs is to make modeling efficient and cost-effective.

Through the optimization process and analysis of the efficient use of TPUs, it was found that the Inception\_V3 pretrained model is a model that meets both criteria better than other models. The project explores its characteristics to find that it delivers a consistent 94% accuracy.

## Motivation

The motivation for this project is based on an interest in improving understanding and skills of deep learning methods on the cloud. Furthermore, I wanted to improve the ability to adapt to different Python libraries and utilize them accurately and efficiently. In the long term, my goal is to develop confidence and quickness in skills implementation by taking part in Kaggle competitions. Thus, this project was inspired by the Kaggle competition: <https://www.kaggle.com/c/flower-classification-with-tpus>



## Objective

To build an efficient and cost-effective machine learning model on TPUs that identifies the 104 types of flowers in a dataset of images.

## Data Exploration

There are over 400,000 Types of Flowers in the wild. The dataset used in the project includes 104 Types of Flowers. Each sample is a TFRecord. The TFRecord format is a container format frequently used in Tensorflow to group and share files for optimal training performance. Each file contains Id, Label (the class of each element in training data), and img (pixels in array form).

```
CLASSES = [  
    'pink primrose', 'hard-leaved pocket orchid', 'canterbury bells', 'sweet pea',  
    'wild geranium', 'tiger lily', 'moon orchid', 'bird of paradise', 'monkshood',  
    'globe thistle', 'snapdragon', 'colt's foot', 'king protea', 'spear thistle',  
    'yellow iris', 'globe-flower', 'purple coneflower', 'peruvian lily',  
    'balloon flower', 'giant white arum lily', 'fire lily', 'pincushion flower',  
    'fritillary', 'red ginger', 'grape hyacinth', 'corn poppy',  
    'prince of wales feathers', 'stemless gentian', 'artichoke', 'sweet william',  
    'carnation', 'garden phlox', 'love in the mist', 'cosmos', 'alpine sea holly',  
    'ruby-lipped cattleya', 'cape flower', 'great masterwort', 'siam tulip',  
    'lenten rose', 'barberton daisy', 'daffodil', 'sword lily', 'poinsettia',  
    'bolero deep blue', 'wallflower', 'marigold', 'buttercup', 'daisy',  
    'common dandelion', 'petunia', 'wild pansy', 'primula', 'sunflower',  
    'lilac hibiscus', 'bishop of llandaff', 'gaura', 'geranium', 'orange dahlia',  
    'pink-yellow dahlia', 'cautleya spicata', 'japanese anemone', 'black-eyed susan',  
    'silverbush', 'californian poppy', 'osteospermum', 'spring crocus', 'iris',  
    'windflower', 'tree poppy', 'gazania', 'azalea', 'water lily', 'rose',  
    'thorn apple', 'morning glory', 'passion flower', 'lotus', 'toad lily',  
    'anthurium', 'frangipani', 'clematis', 'hibiscus', 'columbine', 'desert-rose',  
    'tree mallow', 'magnolia', 'cyclamen', 'watercress', 'canna lily',  
    'hippeastrum', 'bee balm', 'pink quill', 'foxglove', 'bougainvillea',  
    'camellia', 'mallow', 'mexican petunia', 'bromelia', 'blanket flower',  
    'trumpet creeper', 'blackberry lily', 'common tulip', 'wild rose']
```

The dataset used for the project may be found at: <https://www.kaggle.com/c/flower-classification-with-tpus/data>

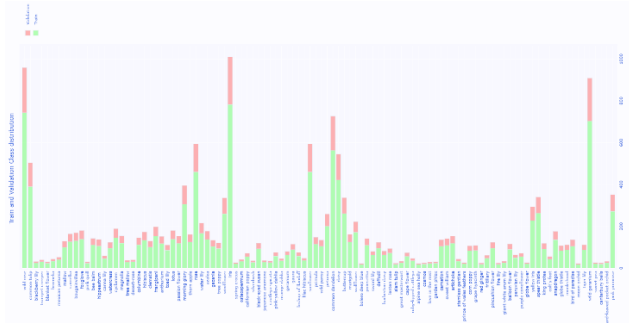
### Train and Validation Distribution and Class Imbalance:

Training Data: There are 12,753 samples in the training data.

Validation Data: There are 3,712 samples in the validation data.

This is an approximate 80/20 percent split between the data for training and validation purposes.

However, there is a class imbalance. Due to the fact the deep learning models are pretrained on ImageNet which has already trained on over ten million images, and because the dataset is not small, there will not be a need to augment the data by making multiples of the images. Furthermore, TPUs make it so that images are naturally augmented in certain ways (discussed later). Therefore, improving the learning of the data and reducing the need for a perfectly balanced dataset.



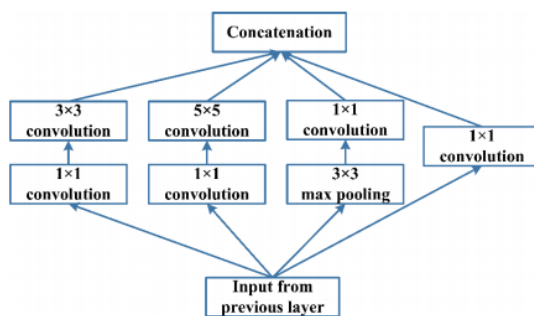
## Image Shapes and Labels:

The input shape used for the modeling is 512x512. The reason for this is to use TPU efficiently. Since there are eight core processors on one TPU, the bigger the matrix the more the TPU cores will be used. Otherwise, they will sit there under- utilized. It also helps the models learn the data better by dividing each sample into smaller pieces to learn their make -up.



## Base Model Inception\_V3 Architecture

Inception\_V3 is convolutional neural network that has 159 layers is used as the base model for the project. It uses batch normalization extensively to stabilize the modeling. Furthermore, it does not take extensive time to run the dataset given. This helps in being able to see how parameter optimization and other tweaks to the model affect predictive behavior much quicker. This is necessary since Kaggle only allows 3 hours per session and 30 hours per week for free TPU use. The Inception\_V3 includes something called inception blocks. Each of the inception blocks consist of the data being split, transformed, and the results are then merged for the output of the inception block.



Basic architecture of the inception block showing the split, transform, and merge concept.

The Inception\_V3 contains total parameters of 22,015,880 with 21,981,448 trainable parameters. The number of trainable parameters will play a role in making Inception\_V3 a strong contender to be used on TPU in an efficient manner.

```
Model: "sequential"
-----
Layer (type)                Output Shape              Param #
-----
inception_v3 (Model)        (None, 14, 14, 2048)     21802784
-----
global_average_pooling2d (Gl (None, 2048)              0
-----
dense (Dense)               (None, 104)              213096
-----
Total params: 22,015,880
Trainable params: 21,981,448
Non-trainable params: 34,432
-----
```

## Model Run on CPU, GPU, & TPU Without Using TPU Capabilities

**Inception\_V3 : Run on CPU w/ TPU OFF**: The preprocessing part of the data for the first layer took a long time. In a snapshot of the CPU usage at 11 minutes, it showed 395% of it being used. This would mean that if the data was being used from the computer directly instead of the cloud, the computer would have crashed. Only because the data was in a bucket near the processing unit, the process did not immediately shut down due to low memory allocation and instead after thirty minutes when the first layer's preprocessing had not even been completed, the run was canceled manually. The number of steps per epoch were noted to be 797.

Session      Disk  
11m          20MB  
9 hours      Max 4.9GB

CPU

CPU          RAM  
395.00%      14.5GB  
Max 16GB

Dataset: 12753 training images, 3712 validation images.  
Steps per epoch: 797

**Inception V3 : Run on GPU w/ TPU OFF:** While the CPU was used up to 161% and even though there was a cumulative use of 81% of the GPU, the run crashed due to low memory allocation within ten minutes and into three minutes of the first epoch run. Once again, the number of steps per epoch were calculated to be 797.

Session 6m  
9 hours

Disk 20MB  
Max 4.9GB

CPU

CPU 161.00%  
RAM 13GB  
Max 13GB

GPU

GPU 81.00%  
GPU Memory 15.2GB  
Max 15.9GB

Epoch 1/22

323/797 [=====>.....] - ETA: 3:02 - loss: 2.8992 - sparse\_categorical\_accuracy: 0.3456

**! Your notebook tried to allocate more memory than is available. It has restarted.**

**Inception V3 : Run on CPU w/ TPU ON but not utilized:** The last test that was tried was having the TPU started but the model was not run on it. The main purpose was to see that if more memory was allocated but the eight core processors on one TPU were not utilized, would the model still run and how long would it take? It was noted, that the run was still time consuming, but it did not crash due to a large memory allocation. First, this time the number of steps per epoch was 99. That is 12 percent of the steps per epoch compared to only CPU or GPU runs. Altogether over the whole run, it only had 1.5% of total steps for whole model compared to only CPU or GPU runs. Still, at a total run time of two hours and forty-four minutes, this modeling with the TPU off is not a good use of its power as will be demonstrated later. Last, the validation loss and accuracy loss stability show promise though. Therefore, Inception\_V3 will continue to be used on TPU next.

Dataset: 12753 training images, 3712 validation images  
Steps per epoch: 99

f1 score: 0.795, precision: 0.855, recall: 0.771

Session 2h:44m  
3 hours

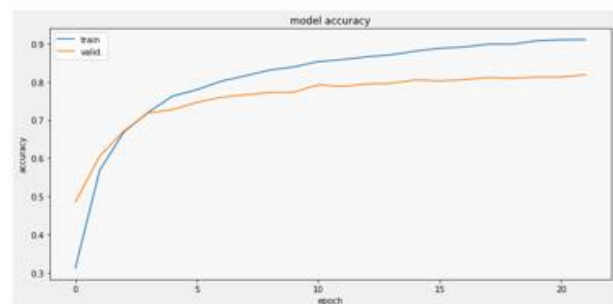
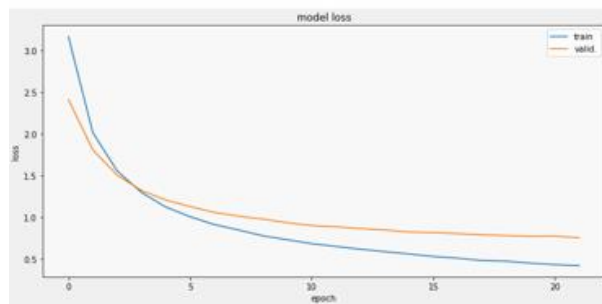
Disk 20MB  
Max 4.9GB

CPU

CPU 0.00%  
RAM 1.7GB  
Max 16GB

TPU

MXU 0.00%  
Idle Time --





## Benefits of Using Tensor Processing Units (TPU)

TPUs were created by Google for the explicit use in deep learning modeling:

- CPU : general purpose computational processor
- GPU : accelerate rendering of graphics
- TPU : accelerate deep learning tasks
- Multiplying and taking slices from arrays takes a lot of CPU clock cycles and memory
- TPUs were designed to relieve these specific workloads.
- First generation of TPUs: have targeted inference
  - Used an already trained model
- Later versions allow adjustments to layers within the model during the training phase
- TPU is 15 times to 30 times faster than contemporary GPUs and CPUs

## TPU Monitor on Kaggle

During the process of modeling, special attention is paid to the TPU monitor. Instead of having to add lines of code to read the efficiency of the TPU usage by the run of the model, Kaggle provides a user interface.

- Performance monitor appears when TPU gauge clicked
- The MXU percentage indicates how efficiently the TPU compute hardware is utilized → higher is better
- The "Idle Time" percentage measures how often the TPU is sitting idle waiting for data
  - Optimizing data pipeline is needed to make this as low as possible
- Measurements:
  - Refreshed approximately every 10 seconds
  - Only appear when the TPU is running a computation

## Inception\_V3 On TPU That is Utilized

**Inception\_V3 On TPU that is utilized (128 batches, 22 epochs):** When run on TPU, the model run time is an efficient eighteen minutes. The batch size is calculated as  $16 * \text{strategy\_number\_replicas}$ . The later part of the calculation was one for previous runs because only one processor was being used. However, since one TPU is composed of eight processors, the batch\_size will be  $16 * 8 = 128$  batches. The run initial was done on twelve epochs. It was not clear if the validation accuracy would improve significantly if more epoch were run, so ten more epochs were added. The accuracy and loss graphs gave a clearer picture of the trend with this number of epochs. More exploration of epochs is done later to see how

changing it might affect the model results. The results with the TPU being used show a validation accuracy of 94.34% and F1 score of 94.4%. This means that the Inception\_V3 is a great model to work with for this dataset. It also uses the MXU at around 20% and 37% of the time the TPU is sitting idle. While we would like the idle time to be lower, it is normal for the TPU to wait for the data to be preprocessed in the CPU.



## Comparison of Manually Optimized Batch\_Size

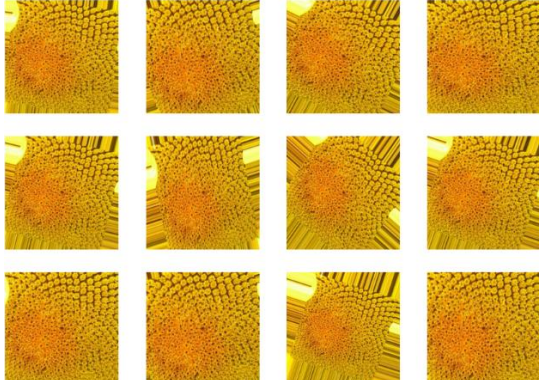
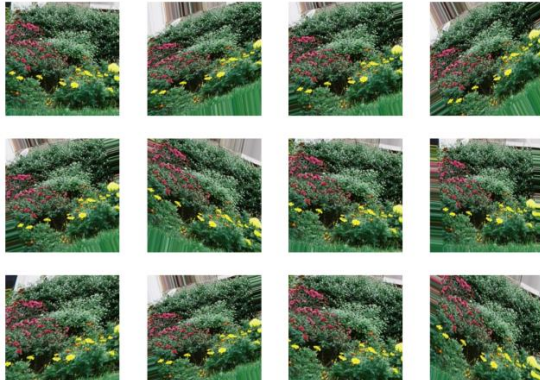
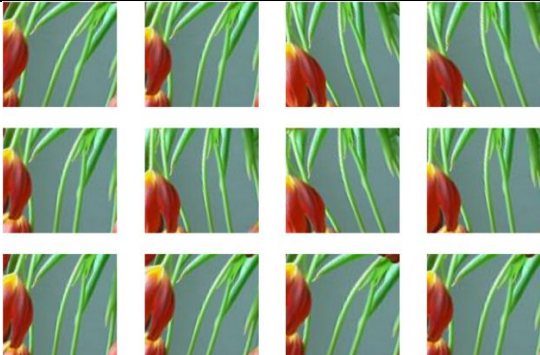
Once it was found that using TPU for running the Inception\_V3 gave strong results, the next steps was to optimize the model's different parameters to improve the results and aim for a cost effective and efficient model. First, the batch\_size was reduced. This only reduced the accuracy, increased loss, and reduced the F1 score. This demonstrated that using less of the TPU's eight processors is not an efficient use of its power. It also increased the model run time. Next, the batch\_sizes were increased. Overall, the accuracies decreased, as well as, the F1 scores. The model run times also did not decrease significantly enough to warrant a change in batch\_size from 128.

COMPARISON OF INCEPTION_V3 MODEL RUN WITH VARIOUS BATCHES PER EPOCH					
BATCH_SIZE	STEPS_PER_EPOCH	VAL_SPARSE_CATEGORICAL_ACCURACY	VAL_LOSS	F1 SCORE	MODEL RUN TIME
64	199	0.9391	0.3075	0.930	21 MINUTES
128	99	0.9434	0.2564	0.944	18 MINUTES
256	49	0.9405	0.2495	0.937	12 MINUTES
512	24	0.9367	0.2628	0.933	13 MINUTES
1028	12	0.9321	0.2757	0.927	69 MINUTES



## Image Augmentation Utilization (Transformation – Rotation + Shrink + Zoom+ Shift + Saturation)

With the use of different image augmentation tools that transform the images randomly, the model results are more stable due to the effort it takes to learn the images from various perspectives. Some of the augmentation, such as the left to right and vice versa transformation, already happens during data preprocessing on the CPU. It is part of the prefetch code line when the training data being sent into the input pipeline.

Transformation	Image
Rotation	
Shrink	
Zoom	

Shift	
Saturation	
Transformation – Rotation + Shrink + Zoom+ Shift + Saturation	

The results of augmentation improved the model in accuracy. The model run also improved. While the F1 decreased and the validation loss increased, there was not a significant difference. Due to the improvement in the model run time and its accuracy improvement, the augmented model was further explored and optimized.

COMPARISON OF INCEPTION_V3 MODEL RUN WITH VERSUS WITHOUT AUGMENATION					
AUGMENTED	STEPS_PER_EPOCH	VAL_SPARSE_CATEGORICAL_ACCURACY	VAL_LOSS	F1 SCORE	MODEL RUN TIME
YES	99	0.9442	0.2642	0.936	16 MINUTES
NO	99	0.9434	0.2564	0.944	18 MINUTES

## Inception\_V3 Model Trainable Parameters Comparison

In comparing whether to use fine-tuning or transferring learning for the trainable parameters in Inception\_V3, it was found adjusting the weights of the pre-trained model by fine tuning gave better results. When using the fine-tuning, the model had 21, 981, 440 trainable parameters. It made use of most of the total parameters in the model. In the transfer learning model, only 213, 096 parameters were trainable, because only the softmax layer was trained. Even when the number of epochs was increased to one hundred to see if the model needed more time to improve its learning of the data and hence its accuracy, it still did not perform at the level a fine-tuned one, which was the model used thus far. The Inception\_V3 model performed poorly on the given dataset with transfer learning. Thus, the pretrained\_model.trainable was turned on when running the model.

### FINE TUNING

```

Model: "sequential"
=====
Layer (type)                Output Shape              Param #
=====
inception_v3 (Model)         (None, 14, 14, 2048)     21802784
global_average_pooling2d (G1 (None, 2048)      0
dense (Dense)                (None, 100)              213096
=====
Total params: 22,015,880
Trainable params: 21,981,440
Non-trainable params: 34,440
=====
  
```

### TRANSFER LEARNING

```

Model: "sequential"
=====
Layer (type)                Output Shape              Param #
=====
inception_v3 (Model)         (None, 14, 14, 2048)     21802784
global_average_pooling2d (G1 (None, 2048)      0
dense (Dense)                (None, 100)              213096
=====
Total params: 22,015,880
Trainable params: 213,096
Non-trainable params: 21,802,784
=====
  
```

- Pretrained\_model.trainable = True
- Adjusts the weights and biases of the pre-trained model → initial values adjusted in training
- Works well on big data

- Pretrained\_model.trainable = False
- Freezes the weights and biases of the pre-trained model → only trains softmax layer
- Works well on small data

## COMPARISON OF INCEPTION\_V3 MODEL RUN WITH VERSUS WITHOUT TRANSFER LEARNING

TRANSFER LEARNING	NUMBER OF EPOCHS	VAL_SPARSE_CATEGORICAL_ACCURACY	VAL_LOSS	F1 SCORE
NO	22	0.9442	0.2642	0.936
YES	22	0.6775	1.4767	0.560
YES	100	0.8314	0.6653	0.820

## Comparison of Manually Optimized Epochs

To see if increasing the number of epoch would improve accuracy, several number of epochs were checked up to one hundred. The decision based on efficiency and results was that twenty-two epochs was an optimal number for modeling purposes. It was reasonably quick, and the accuracy was minutely better.

COMPARISON OF INCEPTION_V3 MODEL RUN WITH FINE TUNING AT 22 VERSUS 100 EPOCHS				
NUMBER OF EPOCHS	VAL_SPARSE_CATEGORICAL_ACCURACY	VAL_LOSS	F1 SCORE	MODEL RUN TIME
22	0.9442	0.2642	0.936	16 MINUTES
100	0.9432	0.2796	0.939	64 MINUTES

## Inception\_V7 Comparison with DenseNet201 & EfficientNetB7

First, when comparing the efficient use of the TPU, it was clear Inception\_V3 used the MXU the most at 20%. Its ratio to the idle time TPU was waiting to run on the model was 54% MXU to TPU Idle Time. This is versus about 45% for both DenseNet201 and EfficientNetB7.

### Efficiency Comparison on TPU

Inception_V3		DenseNet201		EfficientNetB7	
Session 18m 3 hours	Disk 20MB Max 4.9GB	Session 29m 3 hours	Disk 20MB Max 4.9GB	Session 51m 3 hours	Disk 20MB Max 4.9GB
CPU		CPU		CPU	
CPU 6.00%	RAM 1.9GB Max 16GB	CPU 15.00%	RAM 3.1GB Max 16GB	CPU 12.00%	RAM 4.2GB Max 16GB
TPU		TPU		TPU	
MXU 20.00%	Idle Time 37.00%	MXU 15.00%	Idle Time 33.00%	MXU 11.00%	Idle Time 24.00%

The model run time was also significantly less than EfficientNetB7 by 64.61%. The CPU usage was also used less in Inception\_V3. This efficiency strength of the Inception\_V3 had to be kept in mind when seeing the results of the models. EfficientNetB7 had the highest validation accuracy at 95.96% and F1 score of 95.7%.

Both Inception\_V3 and DenseNet201 had lower number of trainable parameters by about two thirds less and their number of layers were significantly less. While the greater number of layers do not necessarily mean an improved accuracy, these three models implemented significant amounts of batch\_normalization to their layers. With the greater amount of layers, it could be this made a difference for the accuracy in EfficientNetB7 model.

COST EFFECTIVE COMPARISON OF MODELS (128 BATCHES, 22 EPOCHS, FINE TUNING)						
MODEL NAME	LAYERS	VAL_SPARSE_CATEGORICAL_ACCURACY	VAL_LOSS	F1 SCORE	TRAINABLE PARAMETERS	MODEL RUN TIME
INCEPTION_V3	159	0.9442	0.2642	0.936	21,981,448	16 MINUTES
DENSENET201	201	0.9518	0.2249	0.952	18,292,712	29 MINUTES
EFFICIENTNETB7	813	0.9596	0.2119	0.957	64,053,304	51 MINUTES

However, another down side to the EfficientNetB7 besides its lower use of MXU, was the time it took for the data to be preprocessed before the first epoch ran was 14 minutes versus 3 minutes for Inception\_V3. Altogether, the EfficientNetB7 run time was 51 minutes versus Inception\_V3 at 18 minutes. These results showed that EfficientNet B7 took 2.8% times longer to result in a 1.54 % greater validation accuracy. However, the F1 score of the EfficientNetB7 makes a case for being patient using it to model if one has the time and ability to use TPU effectively at the cost it brings with it.

**EfficientNetB7 Model:** The time it took for the data to be preprocessed before the first epoch ran was 14 minutes versus 3 minutes for Inception\_V3.

Epoch 1/22 99/99 [=====] - ETA: 0s - loss: 1.9099 - sparse_categorical_accuracy: 0.9442		Session 14m 3 hours	Disk 20MB Max 4.9GB
CPU			

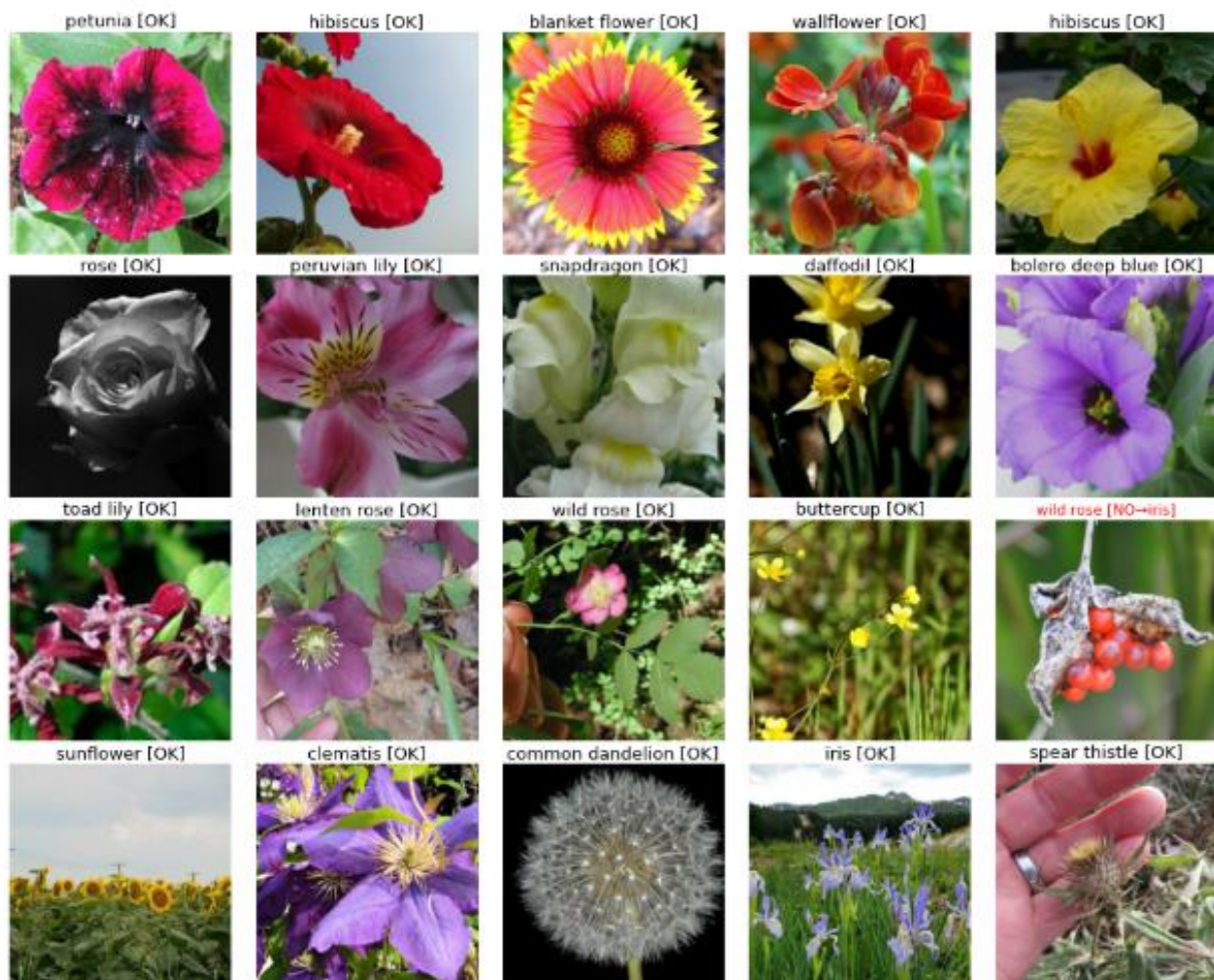


## Conclusion: TPU Limitations & Challenges Affects Ultimate Model Selection

Remember, in Kaggle, one only has 30 hours per week and 3 hours per session to run a model on TPU. In Google Cloud, the cost of each hour of TPU use is \$4.50. Thus, the Inception\_V3 serves our purpose to model the prediction for the given data. It is not only efficient for the given data, but it also cost effective.

- Google charges \$4.50 per hour(s)
- Kaggle allows up to 30 hours per week of TPU use and 3 hours per session
- Difficulty in setting up buckets for data set up on Google Cloud
- Unnecessary for models which are not deep learning, but crashes if large amount of data or memory is needed if not on TPU.

## Best Cost-Effective Model Inception\_V3 Prediction





## Future Exploration

For future exploration, I would like to further fine tune the Inception\_V3 model parameters with such hyperparameter optimizations such SGD, Stochastic Gradient Descent or doing custom loops. In addition, I want to learn PyTorch using XLA to improve modeling efficiency and accuracy. Also, I would like to compete actively in Kaggle competitions to increase confidence and grow collaboration ability. Last, I found that I need to improve the Google Cloud skills necessary to be able to effectively run models on TPU.

## Appreciation of Support

First, I want to thank Dr. Mhoon for her vast amount of knowledge sharing in class with a kindness that surpasses any experience of mine thus far. She empowers her students to believe in themselves and through the learning process in her class, they gain confidence in their abilities without realizing how much she is making them practice the concepts in a practical manner. Dr. Mhoon is a strong role model for any student but especially for those of us who don't see many women in the technology field who are strong and visible yet with a quiet strength like hers. She answers every student's questions with complete patience. It allows everybody to feel heard and respected. When you feel overwhelmed, it takes one conversation from Dr. Mhoon to make it all right.

Second, Ms. Griffin has always been a constant source quick support. It seems she has the answer to everything and is gracious in answering the multitude number of questions we send her with grace. She has helped me personally make some important decisions on this journey towards the completion of my Master's.

Dr. Shastri and Dr. Soibam teach classes that are constantly challenging. They support their students tirelessly and don't expect less than what they can deliver. Besides, the practical work, their resources are deep, and one cannot help but be thankful for the challenge in their classes when one works on a comprehensive project such as this. They build in you a desire to learn and grow.

Dr. King and Dr. Nguyen, both love what they teach. I asked them both many mathematical questions as they explained statistics and how its use in modeling. Dr. King takes a new batch of students to the program and ingrains in them a long term understanding of the statistics that is critical to understanding modeling and its results. Dr. Nguyen, with his humble demeanor shares knowledge and common-sense advice during the class. If you listen to it, it helps to make your modeling a lot less complicated and more intuitive based on of course sound statistical knowledge.

## Resources

Asifullah Khan, Anabia Sohail, Umme Zahoora , and Aqsa Saeed Qureshi; A Survey of the Recent Architectures of Deep Convolutional Neural Networks. Published in Artificial Intelligence Review, DOI: <https://doi.org/10.1007/s10462-020-09825-6> 1 arXiv:1901.06032 [cs.CV]

Mahbub Hussain, Jordan J. Bird, Diego R. Faria; A Study on CNN Transfer Learning for Image Classification. Part of the Advances in Intelligent Systems and Computingbook series (AISC, volume 840). Published on 11 August 2018

Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, Kurt Keutzer; Computer Vision and Pattern Recognition (cs.CV). Published online 14 Sep 2017

Codementor:

- <https://www.codementor.io/@sheena/python-path-virtualenv-import-for-beginners-du107r3o1>

GitHub:

- <https://github.com/tensorflow/tensorflow/releases/tag/v2.2.0-rc4>

Google Cloud Learning Resources:

- <https://cloud.google.com/tpu/docs/inception-v3-advanced>
- <https://cloud.google.com/tpu/docs/tpus>
- <https://codelabs.developers.google.com/codelabs/keras-flowers-tpu/#5>
- <https://cloud.google.com/tpu/docs/more-models>

Kaggle:

- Learn With Me: Getting Started with Tensor Processing Units (TPUs) | Kaggle: <https://www.youtube.com/watch?v=1pdwRQ1DQfY&feature=youtu.be>
- <https://www.kaggle.com/docs/tpu>

Medium:

- <https://medium.com/sciforce/understanding-tensor-processing-units-10ff41f50e78>
- <https://medium.com/@antonpaquin/whats-inside-a-tpu-c013eb51973e>

Tensorflow:

- [https://www.tensorflow.org/guide/keras/transfer\\_learning](https://www.tensorflow.org/guide/keras/transfer_learning)