Machine Learning Bootcamp Launchpad

Sayak Paul | Deep Learning Associate at PyImageSearch

December 12 - 13, 2019

Bengaluru, India

Launchpad

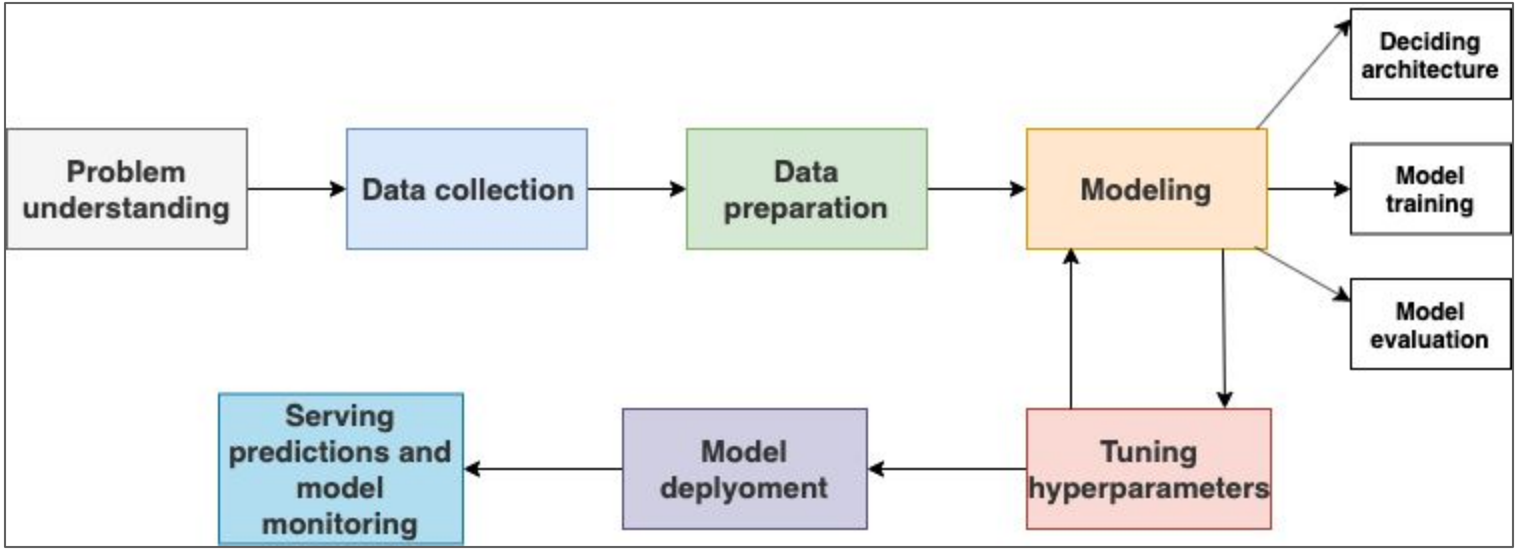
Agenda

- Implement the *typical* steps of a machine learning project from *data collection* to *model serving*.
    - Use a combination of TensorFlow (2.0) and several GCP services (spoiler: AI Platform, Google Cloud Storage).
    - Go back and forth between discussions and hands-on examples.

Acknowledgements

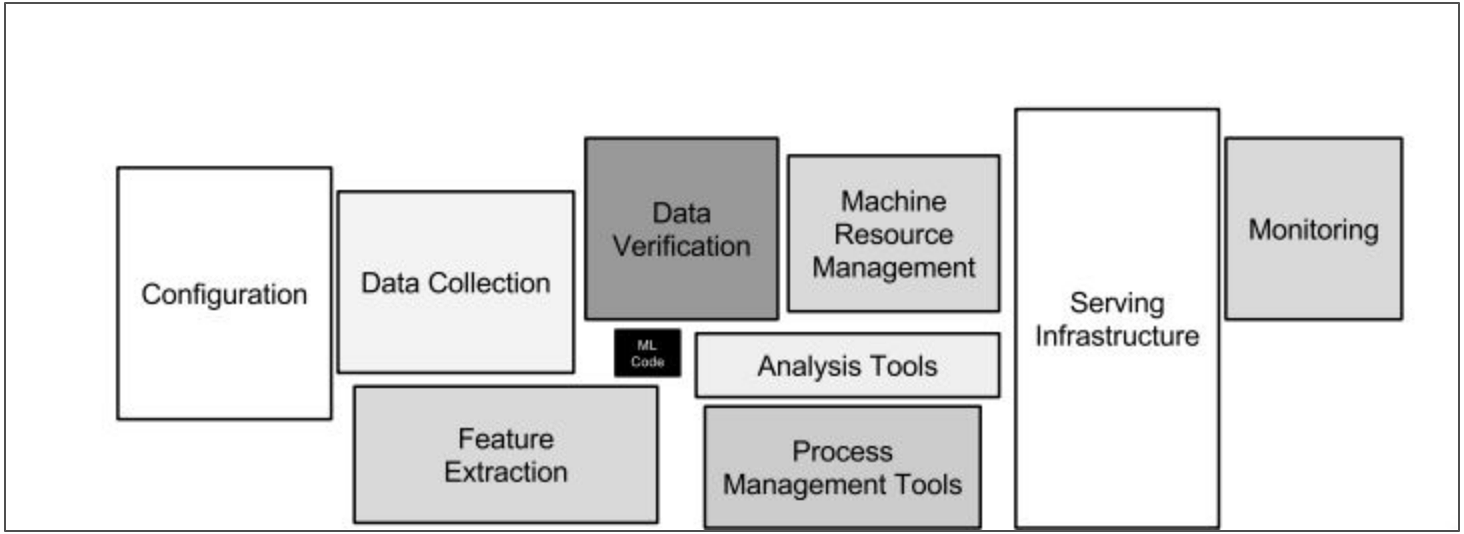
- The entire team at PyImageSearch
- Martin Görner
- Yufeng Guo
- ML-GDE team for granting me GCP Credits to aid this Bootcamp

A typical ML workflow



*Model optimization excluded

A typical ML workflow (much more ordered)



Source: [Hidden Technical Debt in Machine Learning Systems]

Worth spending some time on **problem framing in machine learning**!

https://developers.google.com/machine-learning/problem-framing/

Our problem for today and tomorrow

Given an image of a flower the task is to predict the category to which it is most likely gonna belong to.

Necessary files

- Notebooks: http://bit.ly/mlb-code-sayak
- Deck: http://bit.ly/mlb-sayak

Our dataset: Flowers-17

Sample images and labels

Environment setup

- Create a Notebook instance via the AI Platform.

Notebook instances BETA   NEW INSTANCE   REFRESH   START   STOP   RESET   DELETE

Filter table

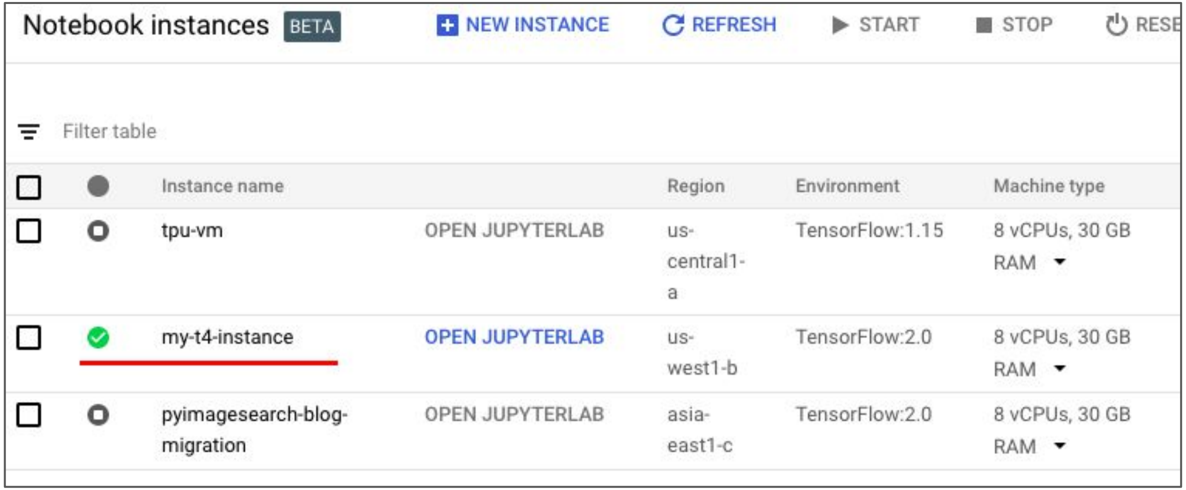
		Instance name		Region	Environment	Machine type	GPUs
		tpu-vm	OPEN JUPYTERLAB	us-central1-a	TensorFlow:1.15	8 vCPUs, 30 GB RAM	None
		my-t4-instance	OPEN JUPYTERLAB	us-west1-b	TensorFlow:2.0	8 vCPUs, 30 GB RAM	NVIDIA Tesla T4 x 1
		pyimagesearch-blog-migration	OPEN JUPYTERLAB	asia-east1-c	TensorFlow:2.0	8 vCPUs, 30 GB RAM	NVIDIA Tesla V100 x 1

Environment setup

- Create a Notebook instance via the AI Platform.
- Customize your instance to have the following:
    - TensorFlow 2.0 environment
    - Tesla T4 / Tesla V100 GPU (I will tell the reason later)
    - RAM according to your choice

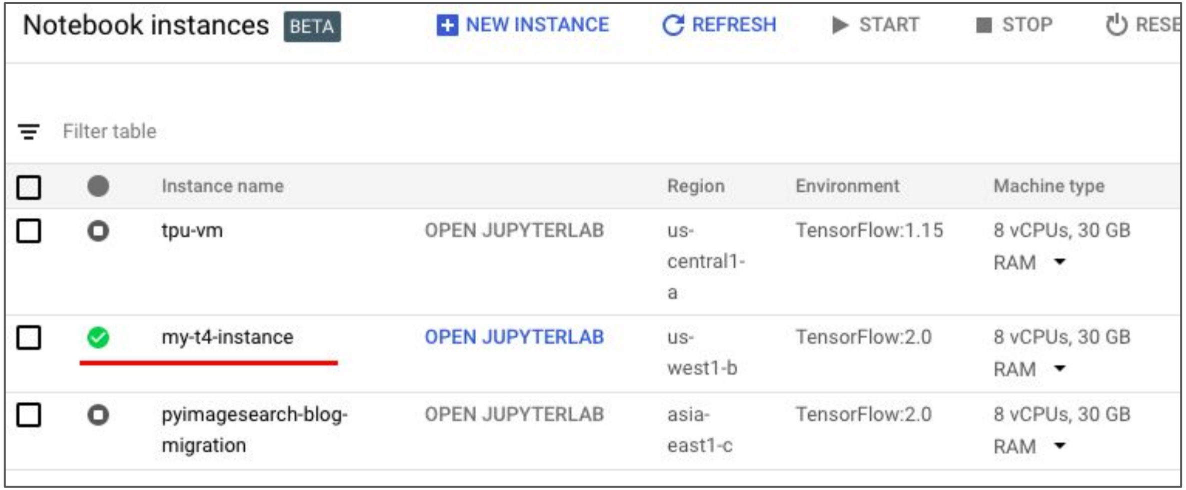
Environment setup

- Create a Notebook instance via the AI Platform.
- Customize your instance.
- Your notebook should be up and running at this stage.



Environment setup

- Create a Notebook instance via the AI Platform.
- Customize your instance.
- Your notebook should be up and running at this stage.



Click on **OPEN JUPYTERLAB!**

Using Weights and Biases for experiments

- Set up W&B
  - `sudo pip3 install wandb`
  - Might need to set paths (instructions here)
- Login to your W&B account
  - `wandb login`
- Initialize W&B
  - `wandb.init()`
- Supply `WandbCallback` to your model.

Using Weights and Biases for experiments

```python
# imports
from wandb.keras import WandbCallback
import wandb

# login to your wandb account
!wandb login

# initialize wandb
wandb.init("project-name", "run-name")

# define your model
model = ...

# compile your model
model.compile(...)

# train your model with Wandbcallback
model.fit(...,
         callbacks=[WandbCallback()])
```

1. Data collection

Data collection

- Collect the Flowers-17 dataset.
- Visualize the images and the labels.
- Create train and test sets.

Data collection

- Collect the Flowers-17 dataset
- Visualize the images and the labels
- Create train and test sets

Notebook: **01_Data_Collection.ipynb**

2. Data preparation

Data preparation

- Data preprocessing
  - Scaling image pixel values
  - Encoding the labels
  - Serializing the pixel values and encoded labels (optional)

Data preparation

- Data preprocessing
  - Scaling image pixel values
  - Encoding the labels
  - Serializing the pixel values and encoded labels (optional)

Notebook: **02_1_Data_Preparation.ipynb**

Data preparation

- Data preprocessing
- Data input pipeline
  - Data augmentation with `ImageDataGenerator`
  - Measuring the performance of `ImageDataGenerator`

Notebook: **[02_1_Data_Preparation.ipynb]**

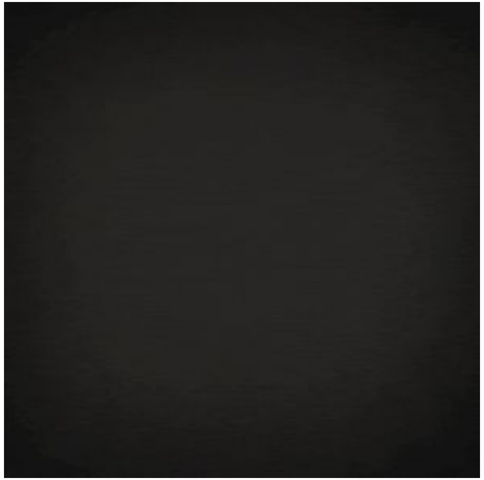
Data preparation

- Data preprocessing
- Data input pipeline
  - Data augmentation with `ImageDataGenerator`
  - Measuring the performance of `ImageDataGenerator`
  - Using `tf.data` to speed things up
  - Building a data input pipeline with `tf.data`

Data preparation

- Data preprocessing
- Data input pipeline
  - Data augmentation with `ImageDataGenerator`
  - Measuring the performance of `ImageDataGenerator`
  - Using `tf.data` to speed things up
  - Building a data input pipeline with `tf.data`

Notebook: [02_2_Data_Preparation.ipynb]()



3. Model building and training

Model building and training

- Starting with a shallow convnet
- Analyze model training and model performance

Model building and training

- Starting with a shallow convnet
- Analyze model training and model performance

Notebook: **03_1_Modeling.ipynb**

Model building and training

- Starting with a shallow convnet
- Analyze model training and model performance
- Doing better with transfer learning (VGG16)
- Analyze model training and model performance

Model building and training

- Starting with a shallow convnet
- Analyze model training and model performance
- Doing better with transfer learning (VGG16)
- Analyze model training and model performance

Notebook: **03_2_Modeling.ipynb**

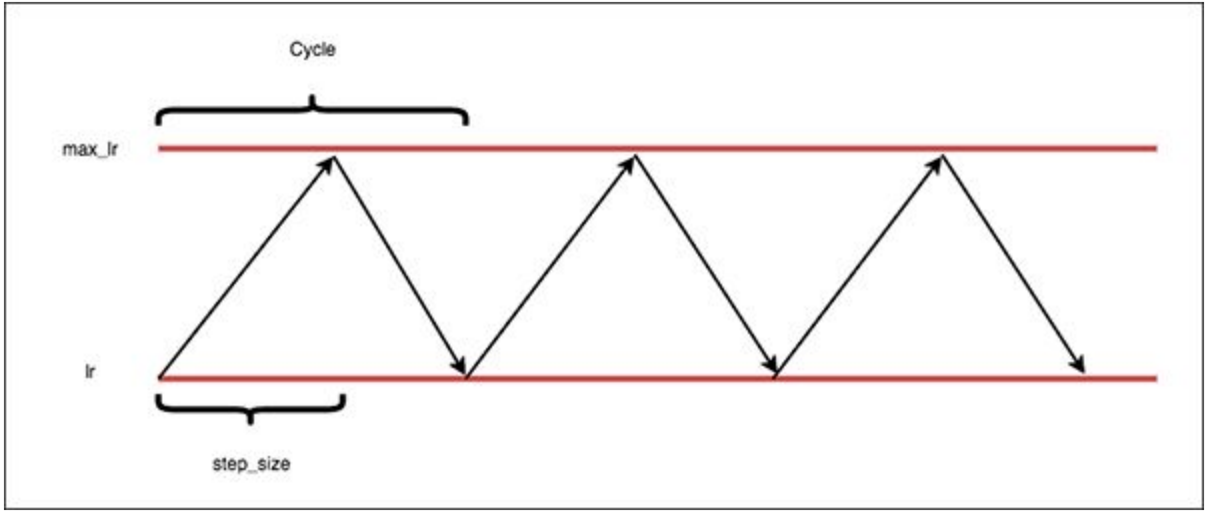
4. Hacking into model training

Hacking into model training

- Hyperparameter tuning
    - Using the *cyclical learning rate* to train our model better
        - Learning rate finder
        - CLR callback

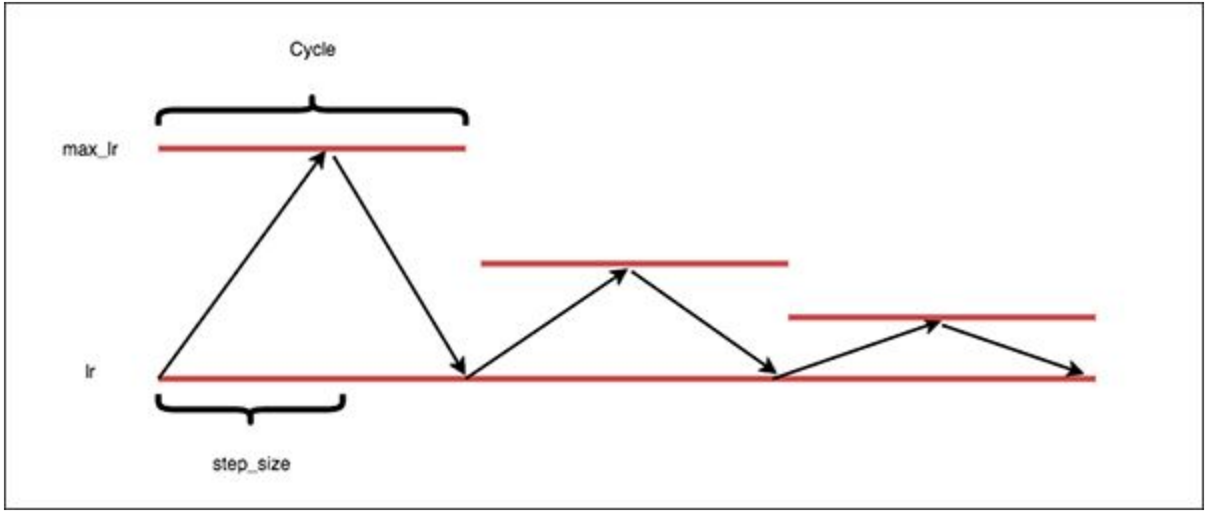
Hacking into model training

- Hyperparameter tuning
    - Using the *cyclical learning rate* to train our model better



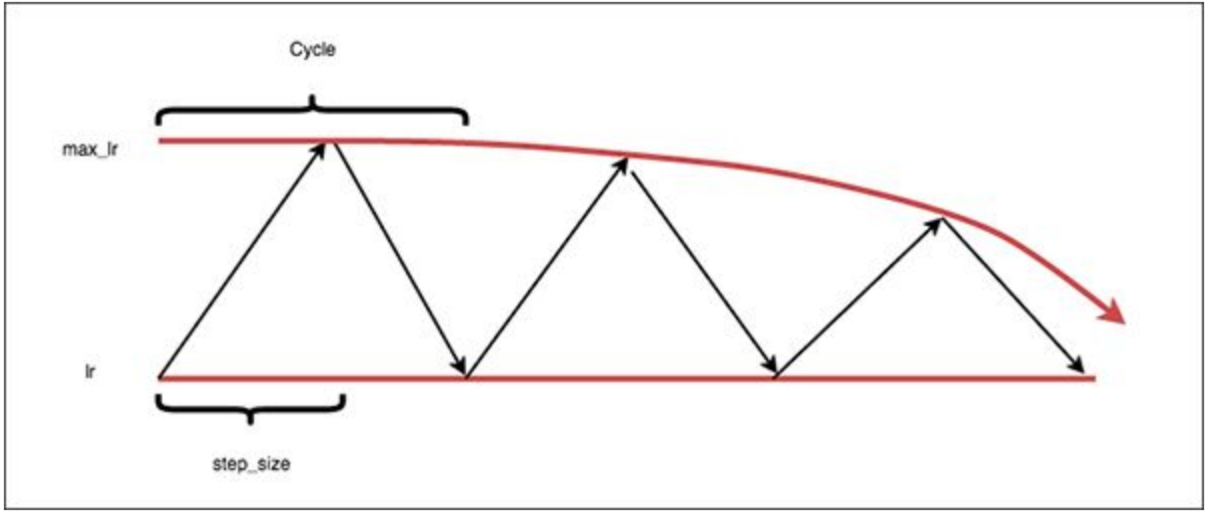
Hacking into model training

- Hyperparameter tuning
    - Using the *cyclical learning rate* to train our model better



Hacking into model training

- Hyperparameter tuning
    - Using the *cyclical learning rate* to train our model better



Hacking into model training

- Hyperparameter tuning
  - Using the *cyclical learning rate* to train our model better
    - Learning rate finder
    - CLR callback

Notebook: **04_1_Model_Hacking.ipynb**

Hacking into model training

- Hyperparameter tuning
- Using *mixed-precision training* to speed up the training process
  - Works with NVIDIA Volta and Turing generation of GPUs
  - Using a combination of float16 and float32 computations
  - Loss and gradient scaling to prevent numerical underflow
  - Computations happening on both Tensor cores (FP16) and CUDA cores (FP32)
  - Not every model can benefit from it

Hacking into model training

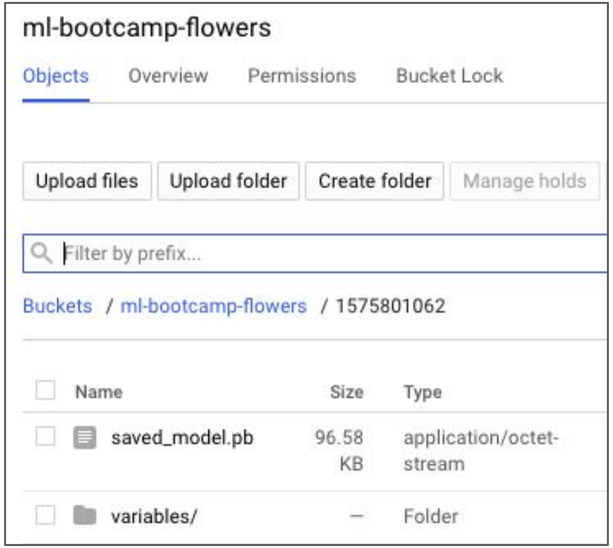
- Hyperparameter tuning
- Using *mixed-precision training* to speed up the training process

Notebook: [04_2_Model_Hacking.ipynb]

5. Model deployment

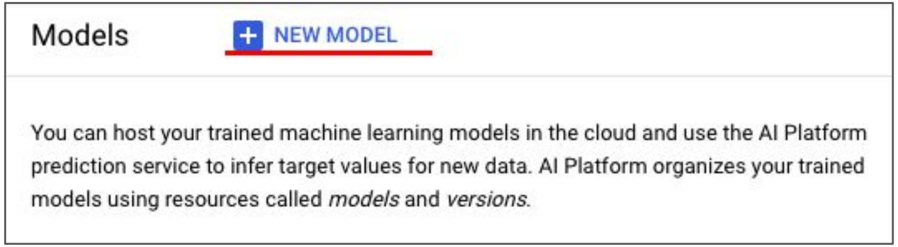
Model deployment

- Serializing the final model in `SavedModel` format (covered already)
- Uploading our model artifacts to [Google Cloud Storage]



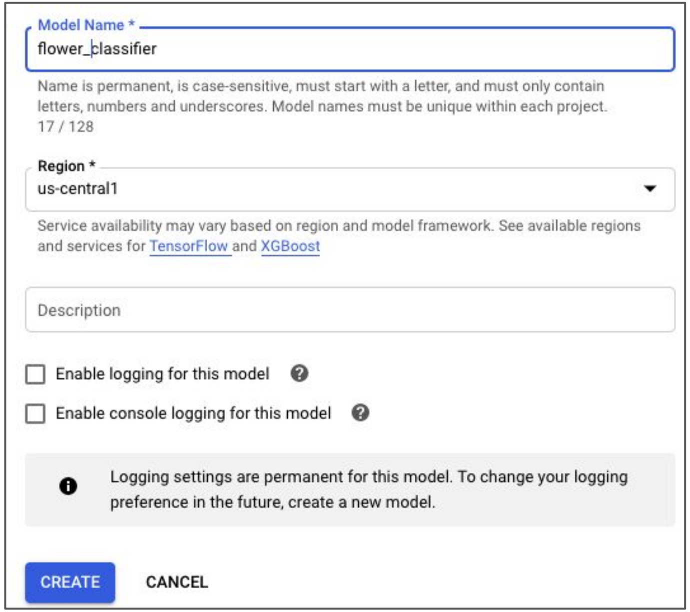
Model deployment

- Serializing the final model in `SavedModel` format (covered already)
- Uploading our model artifacts to GCS
- Kickstart an AI Platform model submission job via the AI Platform GUI
  - Go to Models and click on **NEW MODEL**



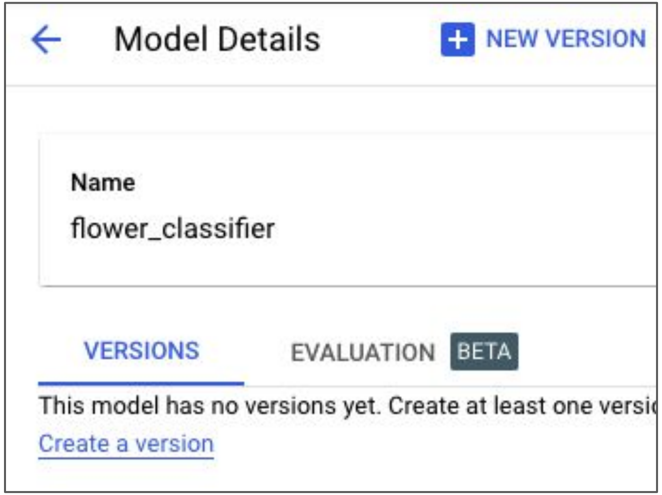
Model deployment

- Kickstart an AI Platform model submission job via the AI Platform GUI
  - Go to Models and click on **NEW MODEL**
  - Create the model



Model deployment

- Kickstart an AI Platform model submission job via the AI Platform GUI
  - Go to Models and click on **NEW MODEL**
  - Create the model
  - Create a version of the model

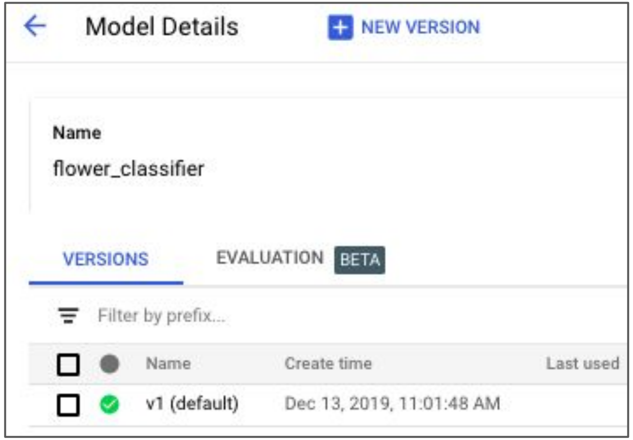


Model deployment

- Kickstart an AI Platform model submission job via the AI Platform GUI
  - Go to Models and click on **NEW MODEL**
  - Create the model
  - Create a version of the model
  - Specify the runtime and upload the artifacts

Model deployment

- Kickstart an AI Platform model submission job via the AI Platform GUI
  - Go to Models and click on **NEW MODEL**
  - Create the model
  - Create a version of the model
  - Specify the runtime and upload the artifacts



And voila!

6. Serving predictions

Serving predictions

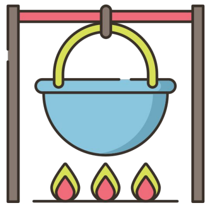
- Randomly selecting a set of images for testing
- Preparing the images for online prediction
- Using AI Platform’s predict jobs to perform inference

Serving predictions

- Randomly selecting a set of images for testing
- Preparing the images for online prediction
- Using AI Platform's predict jobs to perform inference

Notebook: [05_Serving_Predictions.ipynb]

Food for further executions



- Include preprocessing steps in the model itself.
- Use TensorRT to further optimize the inference time (GPU only).
- Use Keras-Tuner to tune other hyperparameters of a model.
- Use AI Platform for model training.
- Include type annotations in the utility functions.
- Include a set of useful test cases for each of the steps.
- Model interpretation with tools like TensorFlow Lucid.

See you next time

@RisingSayak

Thank you very much :)

Experts

pyimagesearch