

Machine Learning Bootcamp Launchpad

Sayak Paul | Deep Learning Associate at [PyImageSearch](#)

December 12 - 13, 2019

Bengaluru, India



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.



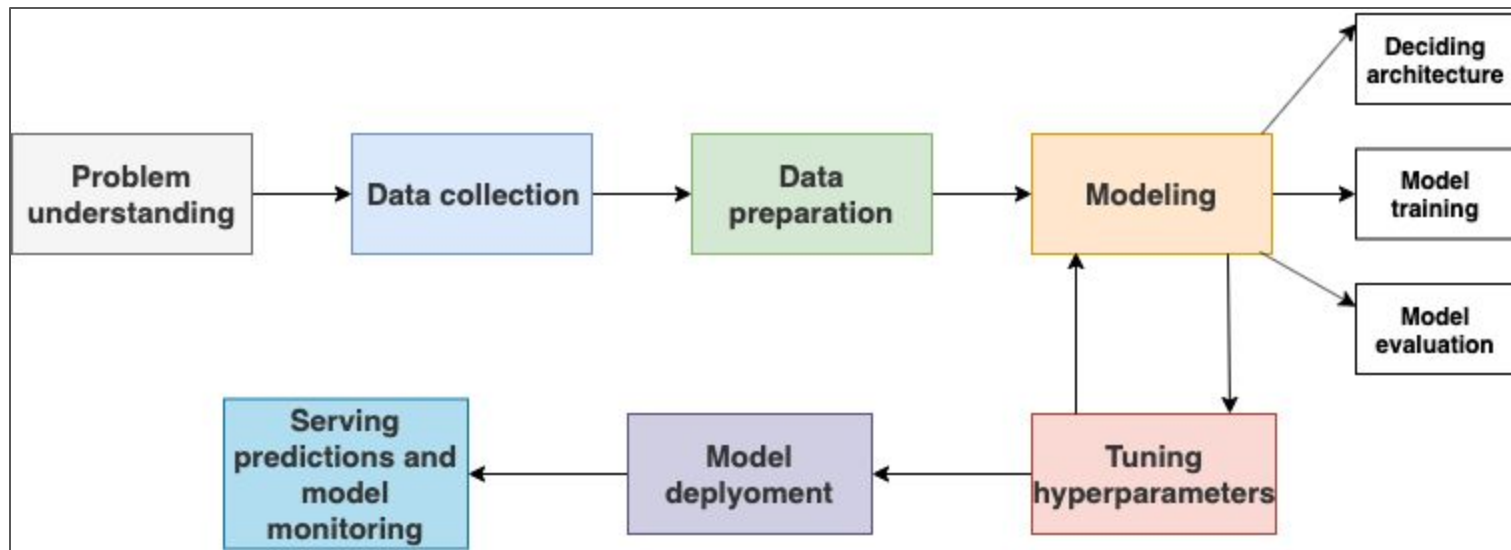
TensorFlow



Acknowledgements

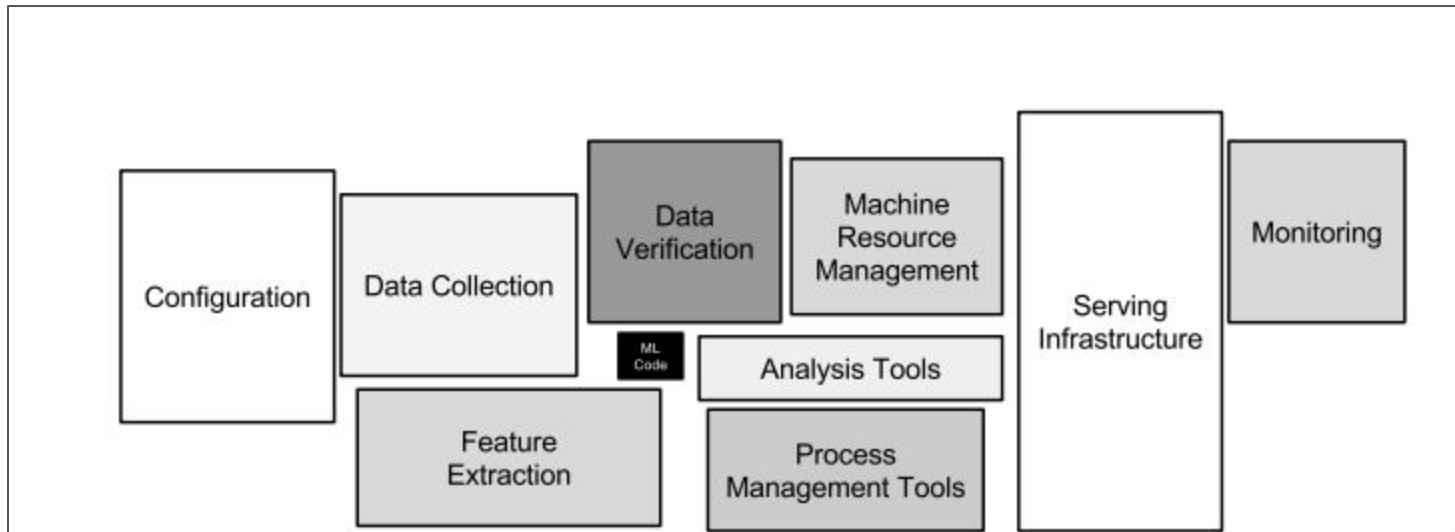
- The entire team at PyImageSearch
- [Martin Görner](#)
- [Yufeng Guo](#)
- ML-GDE team for granting me GCP Credits to aid this Bootcamp

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												
<input type="checkbox"/>	<input type="radio"/>	Instance name		Region	Environment	Machine type	GPUs					
<input type="checkbox"/>	<input type="radio"/>	tpu-vm	OPEN JUPYTERLAB	us-central1-a	TensorFlow:1.15	8 vCPUs, 30 GB RAM ▼	None ▼					
<input type="checkbox"/>	<input type="radio"/>	my-t4-instance	OPEN JUPYTERLAB	us-west1-b	TensorFlow:2.0	8 vCPUs, 30 GB RAM ▼	NVIDIA Tesla T4 x 1 ▼					
<input type="checkbox"/>	<input type="radio"/>	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.

Notebook instances BETA + NEW INSTANCE ↻ REFRESH ▶ START ■ STOP 🔄 RESE					
☰ Filter table					
<input type="checkbox"/>		Instance name	Region	Environment	Machine type
<input type="checkbox"/>		tpu-vm	us-central1-a	TensorFlow:1.15	8 vCPUs, 30 GB RAM ▼
<input type="checkbox"/>		my-t4-instance	us-west1-b	TensorFlow:2.0	8 vCPUs, 30 GB RAM ▼
<input type="checkbox"/>		pyimagesearch-blog-migration	asia-east1-c	TensorFlow:2.0	8 vCPUs, 30 GB RAM ▼

Environment setup

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


Notebook instances BETA + NEW INSTANCE ↻ REFRESH ▶ START ■ STOP 🔄 RESE					
☰ Filter table					
<input type="checkbox"/>		Instance name	Region	Environment	Machine type
<input type="checkbox"/>		tpu-vm	us-central1-a	TensorFlow:1.15	8 vCPUs, 30 GB RAM ▼
<input type="checkbox"/>		my-t4-instance	us-west1-b	TensorFlow:2.0	8 vCPUs, 30 GB RAM ▼
<input type="checkbox"/>		pyimagesearch-blog-migration	asia-east1-c	TensorFlow:2.0	8 vCPUs, 30 GB RAM ▼

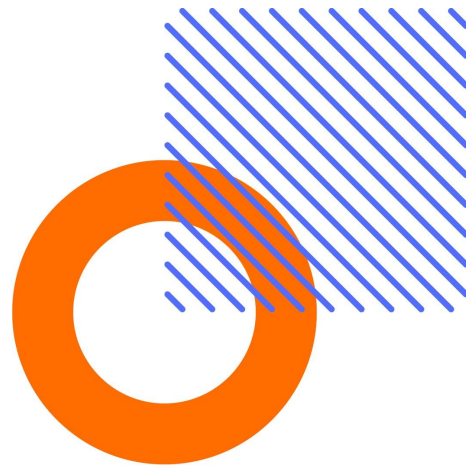
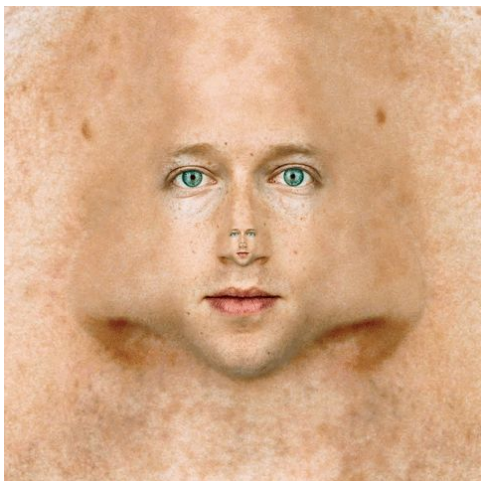
Click on **OPEN JUPYTERLAB!**

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

```
  
  
# 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

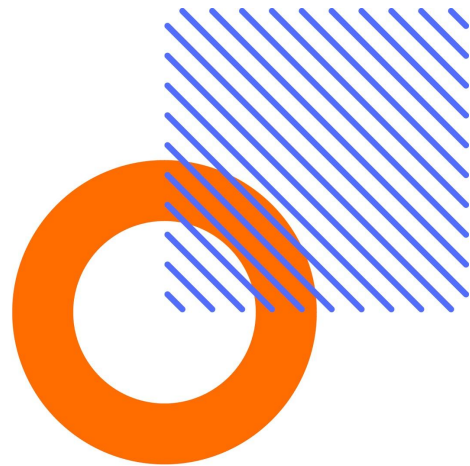
Data collection

- 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](#)

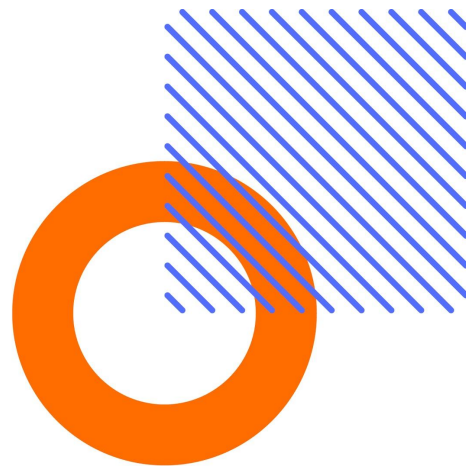
Data preparation

- 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

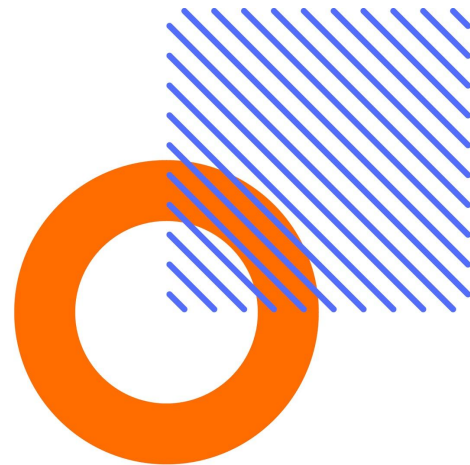
- Starting with a shallow convnet
- Analyze model training and model performance
- Doing better with transfer learning (VGG16)
- Analyze model training and model performance

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](#)

```
00252d0 32 d1 af 60 81 65 0d 58 3f a0 c0 74 08 03 1a
00252e0 3c 68 e8 05 88 07 84 06 05 2f af d8 2b 2a cc 61
00252f0 de 07 01 ad 78 89 62 4a d7 1e 37 18 bf 6a 5a 20
0025300 5f 77 19 df 69 a7 c5 06 29 c4 2c 5e ea 8a 28 26
0025310 ab a3 90 89 2f 73 12 f7 a9 4b 72 d2 41 8b e5 b1
0025320 53 d3 f2 1c b0 be ec ac 51 2c 3b c0 aa 74 24 39
0025330 54 dd 92 3c d0 06 35 a1 26 32 8e 92 b1 11 21 5f
0025340 43 01 bb 8b cb 77 f2 85 5e dc 71 9d 15 ae bf 28
0025350 e7 8a db ca f7 15 fb 08 99 df dd df 7d c2 57 77
0025360 96 8f 75 55 66 5f 52 7c 64 78 64 f3 06 02 73 ab
0025370 9d 0b c7 5a 81 01 33 65 8c 6c e2 e0 2a a7 38 06
0025380 e0 41 c9 29 72 b0 c7 84 0b ef 64 e2 4d 59 39 96
0025390 72 4b 1d 56 2c ba 37 ad 1e d9 6f 7f 82 5b 97 bb
00253a0 7d dc e6 3d 97 d5 2b c4 08 1f 87 1f d2 aa 1e c9
00253b0 7d 89 29 8b ec b6 fd 08 96 54 26 b5 49 87 d8 24
00253c0 dd 0d ad 42 0e 5c 21 b7 6e 5c 95 20 3e 68 ac 40
00253d0 e0 b7 1e 40 84 7d e4 bf eb 01 09 ae f5 3f 7b e4
00253e0 46 3e 7e be 3c bb bb bf bb f6 23 9a 8e 7a 1c 8f
00253f0 b2 62 1c 06 bf 4d 71 75 50 89 23 3f f5 ad 34 d3
0025400 a4 4a 04 57 89 54 3b a1 06 64 62 04 c9 47 0a 3e
0025410 3c a3 97 b5 2b 34 f0 d3 bb a1 fb ac 7a af dd df
0025420 71 37 2f 7b bb bc be 25 54 57 da 42 7b ca 42 29
0025430 73 bf 04 56 df 82 27 8a a0 23 aa 62 78 6a 0c b1
```



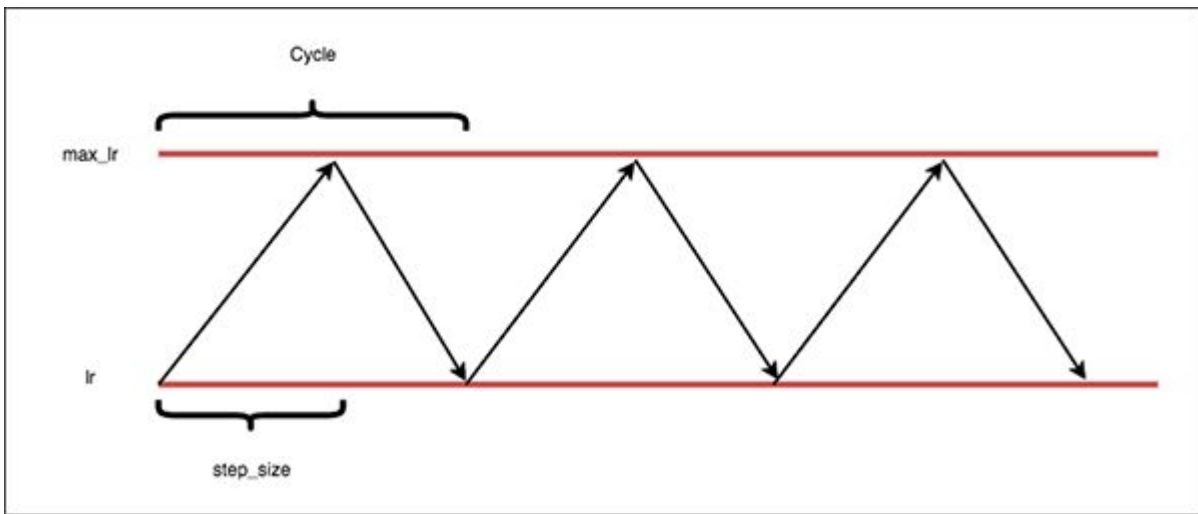
4. Hacking into model training

Hacking into model training

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

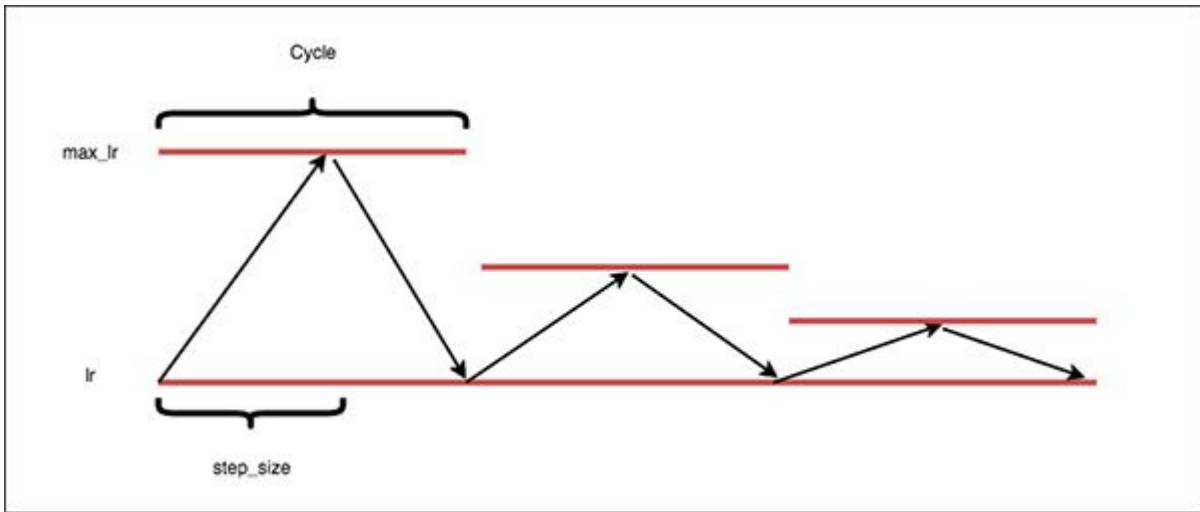
Hacking into model training

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



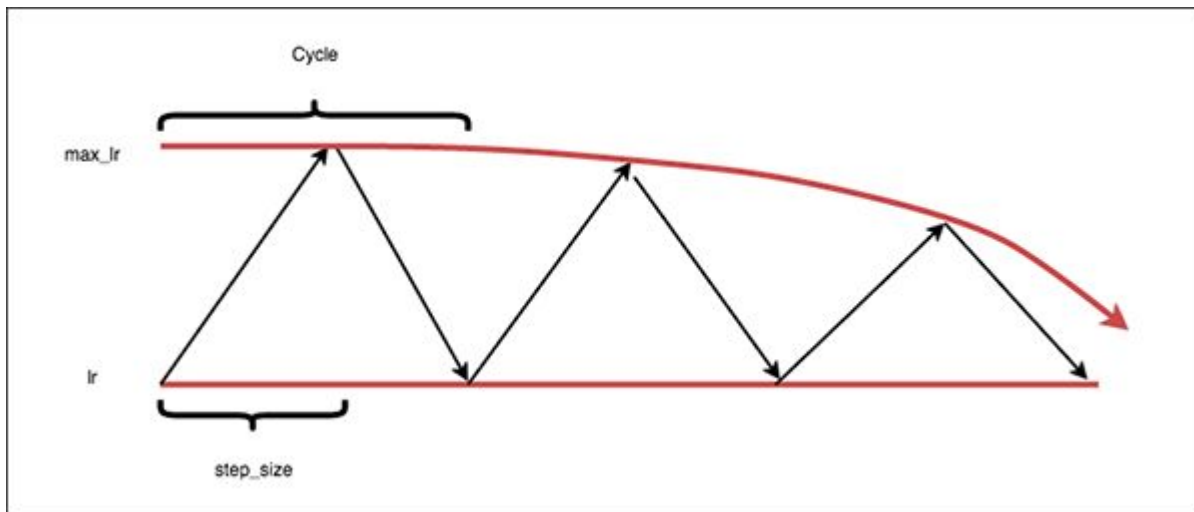
Hacking into model training

- 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](#)

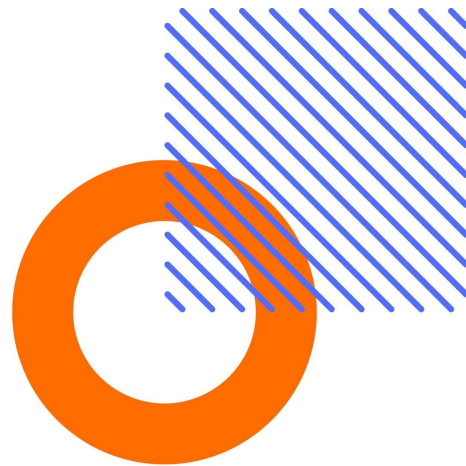
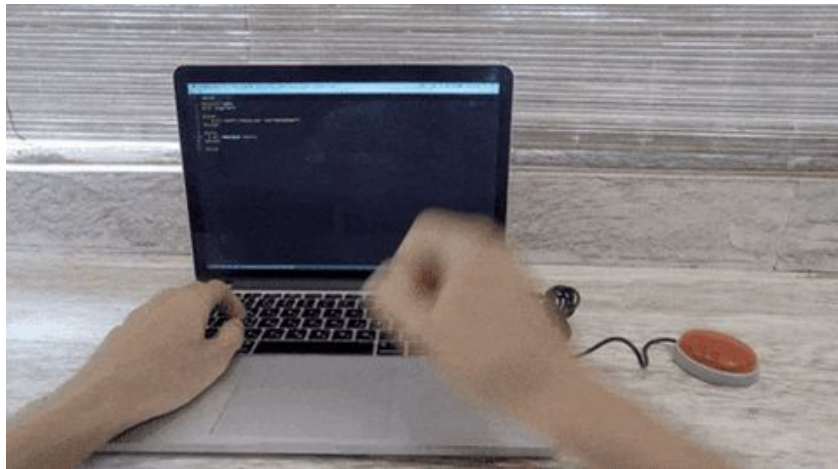
Hacking into model training

- 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

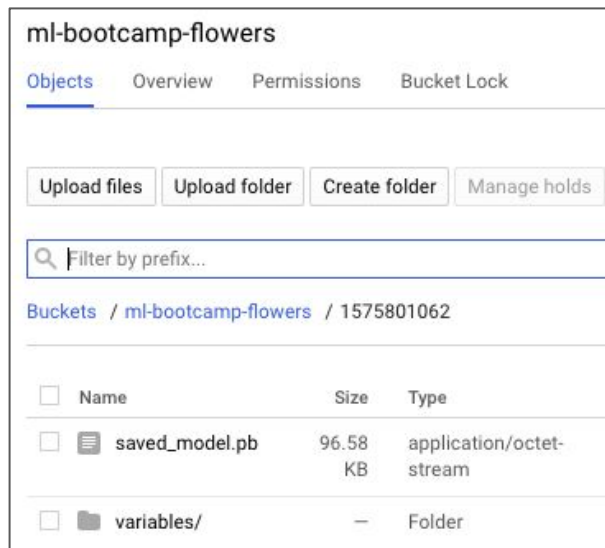
Notebook: [04_2_Model_Hacking.ipynb](#)



5. Model deployment

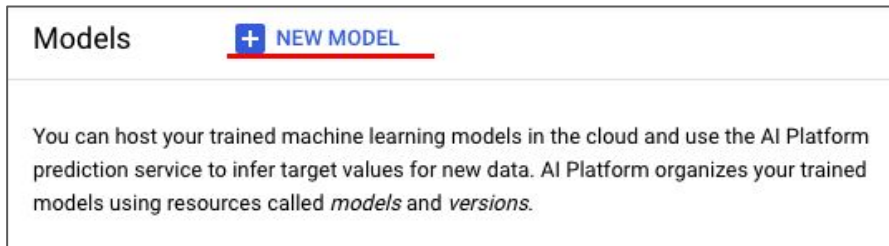
Model deployment

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



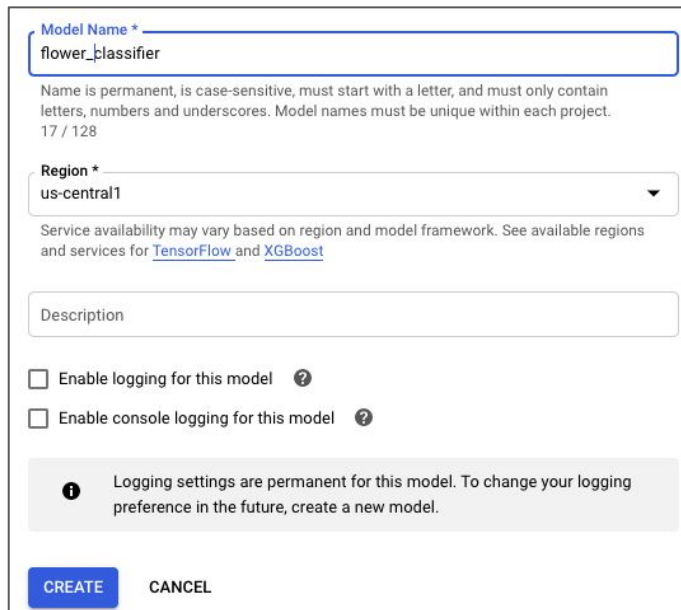
Model deployment

- 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



The screenshot shows the 'NEW MODEL' form in the AI Platform GUI. It includes a 'Model Name' field with the value 'flower_classifier', a 'Region' dropdown menu set to 'us-central1', and a 'Description' field. Below these fields are two checkboxes for logging: 'Enable logging for this model' and 'Enable console logging for this model', both of which are unchecked. At the bottom, there is a blue 'CREATE' button and a grey 'CANCEL' button. A grey informational box at the bottom states: 'Logging settings are permanent for this model. To change your logging preference in the future, create a new model.'

Model Name *
flower_classifier

Name is permanent, is case-sensitive, must start with a letter, and must only contain letters, numbers and underscores. Model names must be unique within each project.
17 / 128

Region *
us-central1

Service availability may vary based on region and model framework. See available regions and services for [TensorFlow](#) and [XGBoost](#)

Description

☐ Enable logging for this model ?

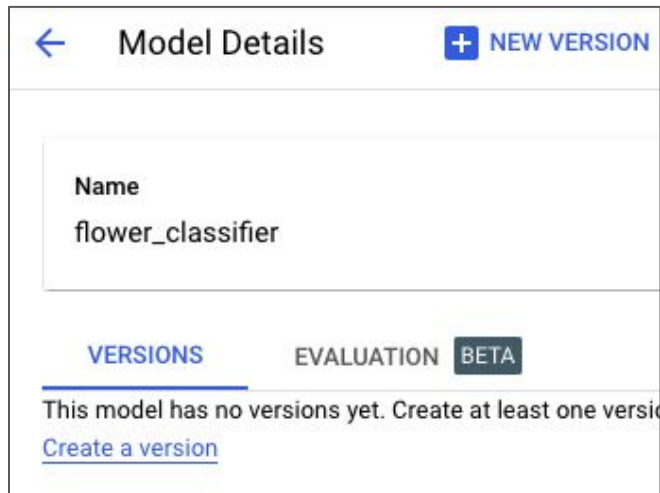
☐ Enable console logging for this model ?

i Logging settings are permanent for this model. To change your logging preference in the future, create a new model.

CREATE CANCEL

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



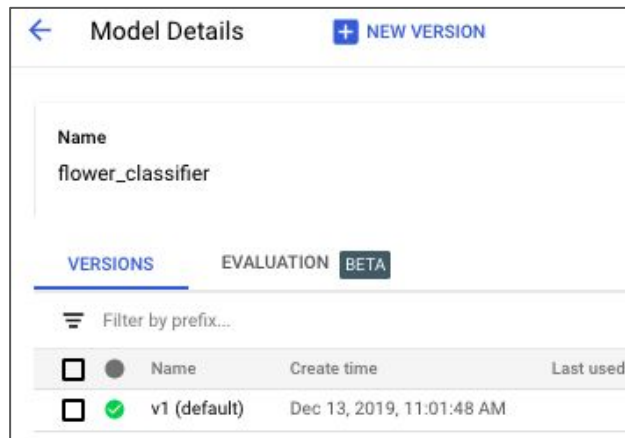
Model deployment

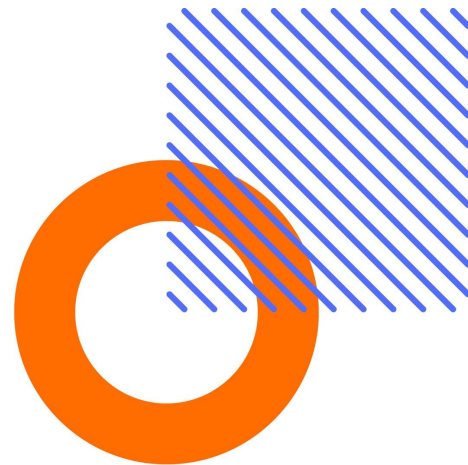
- Kickstart an AI Platform model submission job via the AI Platform GUI
 - Go to [Models](#) and click on **NEW MODEL**
 - Create the model
 - Create a version of the model
 - Specify the runtime and upload the artifacts

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](https://twitter.com/RisingSayak)

Thank you very much :)



Experts

