Week 2 - Ungraded Lab: A journey through Data

Welcome to the ungraded lab for week 2 of Machine Learning Engineering for Production. **The paradigm behind Deep Learning is now facing a shift from model-centric to data-centric.** In this lab you will see how data intricacies affect the outcome of your models. To show you how far it will take you to apply data changes without addressing the model, you will be using a single model throughout: a simple Convolutional Neural Network (CNN). While training this model the journey will take you to address common problems: class imbalance and overfitting. As you navigate these issues, the lab will walk you through useful diagnosis tools and methods to mitigate these common problems.

IMPORTANT NOTES BEFORE STARTING THE LAB

Once opened in Colab, click on the "Connect" button on the upper right side corner of the screen to connect to a runtime to run this lab.

NOTE 1:

For this lab you get the option to either train the models yourself (this takes around 20 minutes with GPU enabled for each model) or to use pretrained versions which are already provided. There are a total of 3 CNNs that require training and although some parameters have been tuned to provide a faster training time (such as `steps_per_epoch` and `validation_steps` which have been heavily lowered) this may result in a long time spent running this lab rather than thinking about what you observe.

To speed things up we have provided saved pre-trained versions of each model along with their respective training history. We recommend you use these pre-trained versions to save time. However we also consider that training a model is an important learning experience especially if you haven't done this before. **If you want to perform this training by yourself, the code for replicating the training is provided as well. In this case the GPU is absolutely necessary, so be sure that it is enabled.**

To make sure your runtime is GPU you can go to Runtime -> Change runtime type -> Select GPU from the menu and then press SAVE

- Note: Restarting the runtime may be required.
- Colab will tell you if restarting is necessary -- you can do this from Runtime -> Restart Runtime option in the dropdown.

If you decide to use the pretrained versions make sure you are not using a GPU as it is not required and may prevent other users from getting access to one. To check this, go to Runtime -> Change runtime type -> Select None from the menu and then press SAVE.

NOTE 2:

Colab **does not** guarantee access to a GPU. This depends on the availability of these resources. However **it is not very common to be denied GPU access**. If this happens to you, you can still run this lab without training the models yourself. If you really want to do the training but are denied a GPU, try switching the runtime to a GPU after a couple of hours.

To know more about Colab's policies check out this FAQ.

Let's get started!

```
import os
import shutil
import random
import zipfile
import tarfile
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf
import matplotlib.pyplot as plt

# To ignore some warnings about Image metadata that Pillow prints out
import warnings
warnings.filterwarnings("ignore")
```

Before you move on, download the two datasets used in the lab, as well as the pretrained models and histories:

```
# Download datasets
```

```
# Download datasets

# Cats and dogs
!wget https://storage.googleapis.com/mlep-public/course_1/week2/kagglecatsanddogs_3367a.zip

# Caltech birds
!wget https://storage.googleapis.com/mlep-public/course_1/week2/CUB_200_2011.tar

# Download pretrained models and training histories
!wget -q -P /content/model-balanced/ https://storage.googleapis.com/mlep-public/course_1/week2/model-balanced/saved_mode
!wget -q -P /content/model-balanced/variables/ https://storage.googleapis.com/mlep-public/course_1/week2/model-balanced/
!wget -q -P /content/model-balanced/variables/ https://storage.googleapis.com/mlep-public/course_1/week2/model-balanced/
!wget -q -P /content/history-balanced/ https://storage.googleapis.com/mlep-public/course_1/week2/history-balanced/histor

!wget -q -P /content/model-imbalanced/ https://storage.googleapis.com/mlep-public/course_1/week2/model-imbalanced/saved_
!wget -q -P /content/model-imbalanced/variables/ https://storage.googleapis.com/mlep-public/course_1/week2/model-imbalar
!wget -q -P /content/model-imbalanced/variables/ https://storage.googleapis.com/mlep-public/course_1/week2/model-imbalar
!wget -q -P /content/history-imbalanced/ https://storage.googleapis.com/mlep-public/course_1/week2/history-imbalanced/hi

!wget -q -P /content/model-augmented/ https://storage.googleapis.com/mlep-public/course_1/week2/model-augmented/saved_mo
!wget -q -P /content/model-augmented/variables/ https://storage.googleapis.com/mlep-public/course_1/week2/model-augmente
!wget -q -P /content/model-augmented/variables/ https://storage.googleapis.com/mlep-public/course_1/week2/model-augmente
!wget -q -P /content/history-augmented/ https://storage.googleapis.com/mlep-public/course_1/week2/history-augmented/hist
```

```
--2021-06-01 09:04:41--  https://storage.googleapis.com/mlep-public/course_1/week2/kagglecatsanddogs_3367a.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 142.250.101.128, 142.250.141.128, 2607:f8b0:4023:c06::
Connecting to storage.googleapis.com (storage.googleapis.com)|142.250.101.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 824894548 (787M) [application/zip]
Saving to: 'kagglecatsanddogs_3367a.zip'

kagglecatsanddogs_3 100%[===================>] 786.68M   129MB/s    in 6.2s

2021-06-01 09:04:48 (127 MB/s) - 'kagglecatsanddogs_3367a.zip' saved [824894548/824894548]

--2021-06-01 09:04:48--  https://storage.googleapis.com/mlep-public/course_1/week2/CUB_200_2011.tar
Resolving storage.googleapis.com (storage.googleapis.com)... 142.250.141.128, 142.250.101.128, 2607:f8b0:4023:c0b::
Connecting to storage.googleapis.com (storage.googleapis.com)|142.250.141.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1259479040 (1.2G) [application/x-tar]
Saving to: 'CUB_200_2011.tar'

CUB_200_2011.tar    100%[===================>]   1.17G  36.2MB/s    in 13s

2021-06-01 09:05:01 (94.5 MB/s) - 'CUB_200_2011.tar' saved [1259479040/1259479040]
```

A story of data

To guide you through this lab we have prepared a narrative that simulates a real life scenario:

Suppose you have been tasked to create a model that classifies images of cats, dogs and birds. For this you settle on a simple CNN architecture, since CNN's are known to perform well for image classification. You are probably familiar with two widely used datasets: `cats vs dogs`, and `caltech birds`. As a side note both datasets are available through `Tensforflow Datasets (TFDS)`. However, you decide NOT to use `TFDS` since the lab requires you to modify the data and combine the two datasets into one.

Combining the datasets

```
with zipfile.ZipFile(cats_and_dogs_zip, 'r') as my_zip:
  my_zip.extractall(base_dir)


with tarfile.open(caltech_birds_tar, 'r') as my_tar:
  my_tar.extractall(base_dir)
```

For the cats and dogs images no further preprocessing is needed as all exemplars of a single class are located in one directory: `PetImages\Cat` and `PetImages\Dog` respectively. Let's check how many images are available for each category:

```
base_dogs_dir = os.path.join(base_dir, 'PetImages/Dog')
base_cats_dir = os.path.join(base_dir,'PetImages/Cat')

print(f"There are {len(os.listdir(base_dogs_dir))} images of dogs")
print(f"There are {len(os.listdir(base_cats_dir))} images of cats")
```

```
There are 12501 images of dogs
There are 12501 images of cats
```

The Bird images dataset organization is quite different. This dataset is commonly used to classify species of birds so there is a directory for each species. Let's treat all species of birds as a single class. This requires moving all bird images to a single directory (`PetImages/Bird` will be used for consistency). This can be done by running the next cell:

```
raw_birds_dir = '/tmp/data/CUB_200_2011/images'
base_birds_dir = os.path.join(base_dir,'PetImages/Bird')
os.mkdir(base_birds_dir)

for subdir in os.listdir(raw_birds_dir):
  subdir_path = os.path.join(raw_birds_dir, subdir)
  for image in os.listdir(subdir_path):
    shutil.move(os.path.join(subdir_path, image), os.path.join(base_birds_dir))

print(f"There are {len(os.listdir(base_birds_dir))} images of birds")
```

```
There are 11788 images of birds
```

It turns out that there is a similar number of images for each class you are trying to predict! Nice!

Sample dog image:

Sample bird image:

```
# Move the remaining images to the eval dir
move_to_destination(base_cats_dir, os.path.join(base_dir, 'eval/cats'), 1)
move_to_destination(base_dogs_dir, os.path.join(base_dir, 'eval/dogs'), 1)
move_to_destination(base_birds_dir, os.path.join(base_dir, 'eval/birds'), 1)
```

Something important to mention is that as it currently stands your dataset has some issues that will prevent model training and evaluation. Mainly:

1. Some images are corrupted and have zero bytes