AI Hands-On Workshop

Worksheet

April 17th, 2024 - 4:30 - 5PM EDT

This is the exercise worksheet and resource guide to the AI Hands-On: Beyond LLMs Workshop from Cisco Offensive Summit 2024.

Workshop Links

GitHub Repositories
Exercise Files: https://github.com/aexcode/cos2024
AI and Machine Learnign for Coders - Companion Code:
Google Colab: https://colab.research.google.com/

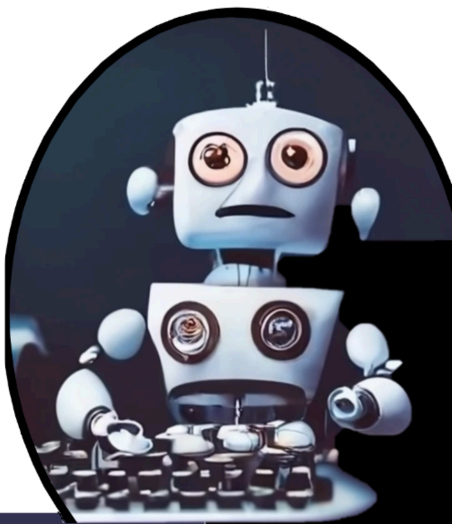
Infosec: "Try Harder"

This stuff **is** *hard*

How about:

" Work ***smarter****, not harder"*

-Allen Morgenstern, Industrial Engineer

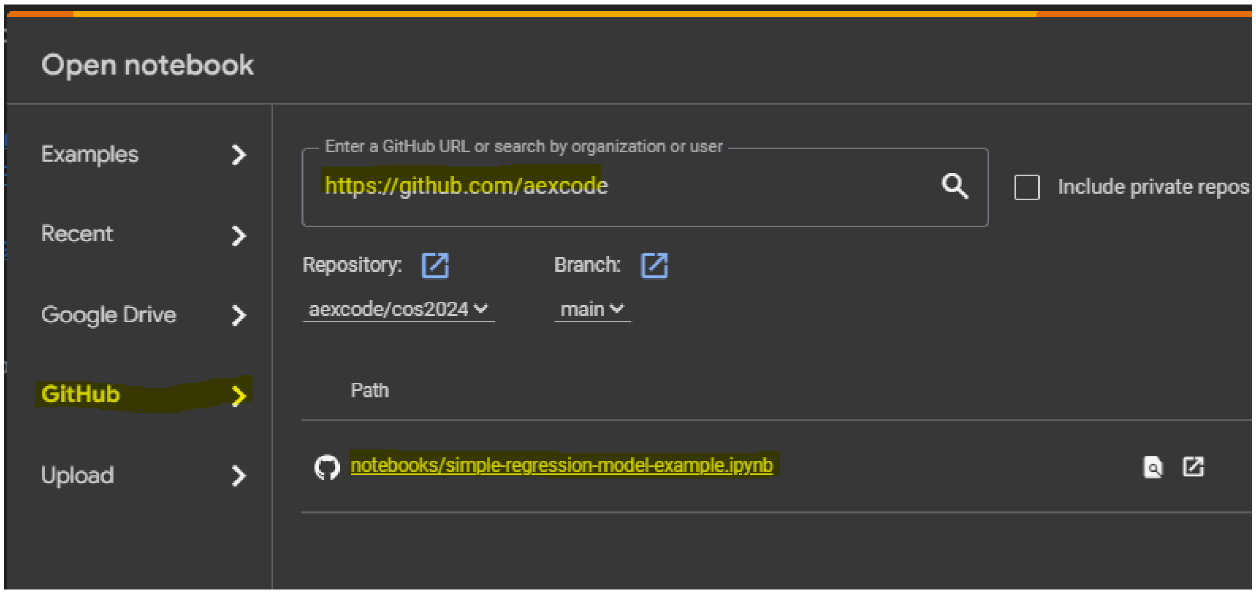


Exercises

Exercise 1: Simple linear regression using a neural network

Our first exercise is a simple linear regression model using a **feed forward, densely connected** single hidden layer neural network. More specifically, feed forward neural networks may also be called FNN.

First, download the Jupyter notebook from the Workshop GitHub repository:


Open notebook

Enter a GitHub URL or search by organization or user: https://github.com/aexcode

Include private repos

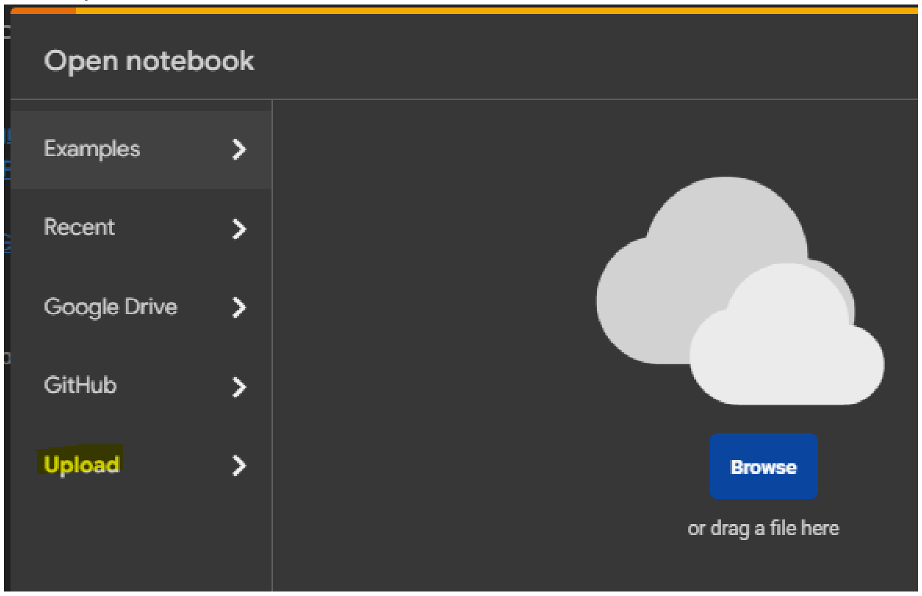
Repository: aexcode/cos2024

Branch: main

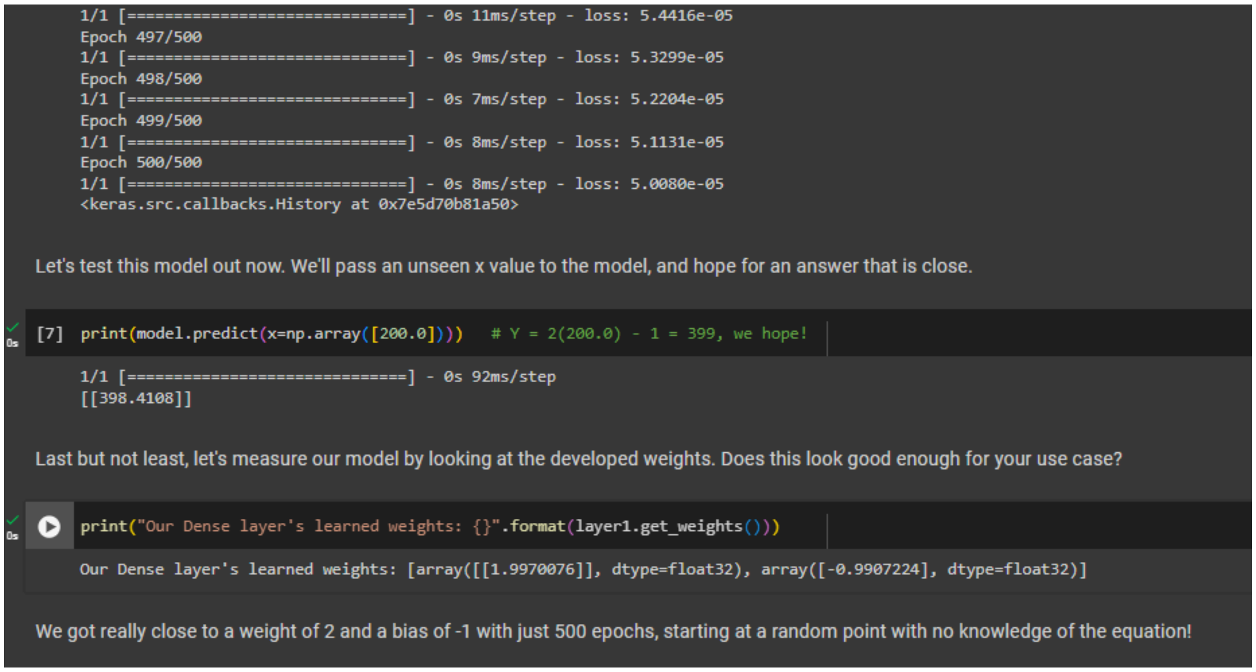
Path

notebooks/simple-regression-model-example.ipynb

Now upload to Colab:


Open notebook

Follow each cell, reading markdown cells and running code cells in order to understand how a simple linear regression model can help you understand how weights and biases work to optimize the training process of a neural network and ultimately create a successful model.



```
1/1 [==============================] - 0s 11ms/step - loss: 5.4416e-05
Epoch 497/500
1/1 [==============================] - 0s 9ms/step - loss: 5.3299e-05
Epoch 498/500
1/1 [==============================] - 0s 7ms/step - loss: 5.2204e-05
Epoch 499/500
1/1 [==============================] - 0s 8ms/step - loss: 5.1131e-05
Epoch 500/500
1/1 [==============================] - 0s 8ms/step - loss: 5.0080e-05
<keras.src.callbacks.History at 0x7e5d70b81a50>
```

Let's test this model out now. We'll pass an unseen x value to the model, and hope for an answer that is close.

```
[7] print(model.predict(x=np.array([200.0])))   # Y = 2(200.0) - 1 = 399, we hope!
```

```
1/1 [==============================] - 0s 92ms/step
[[398.4108]]
```

Last but not least, let's measure our model by looking at the developed weights. Does this look good enough for your use case?

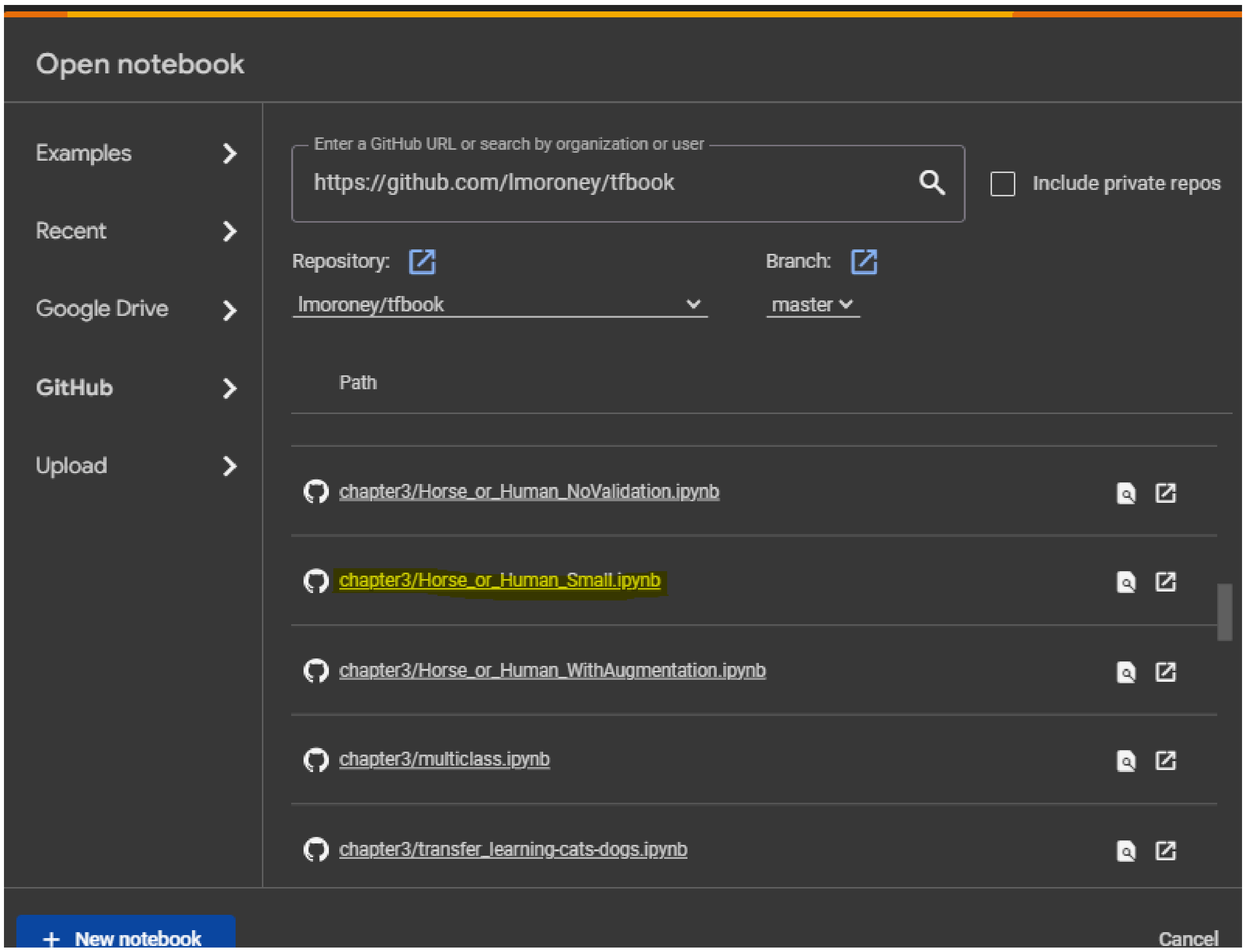
```
print("Our Dense layer's learned weights: {}".format(layer1.get_weights()))
```

```
Our Dense layer's learned weights: [array([[1.9970076]], dtype=float32), array([-0.9907224], dtype=float32)]
```

We got really close to a weight of 2 and a bias of -1 with just 500 epochs, starting at a random point with no knowledge of the equation!

Exercise 2: Convolutional Neural Network (CNN)

The next notebook is a basic convolutional neural network for classification in images. We'll be pulling from an open source repository that is paired with the book "AI and Machine Learning for Coders". You'll need to import the model and the dataset to begin.



Open notebook

Enter a GitHub URL or search by organization or user: https://github.com/lmoroney/tfbook

Include private repos

Repository: lmoroney/tfbook

Branch: master

Path

chapter3/Horse_or_Human_NoValidation.ipynb

chapter3/Horse_or_Human_Small.ipynb

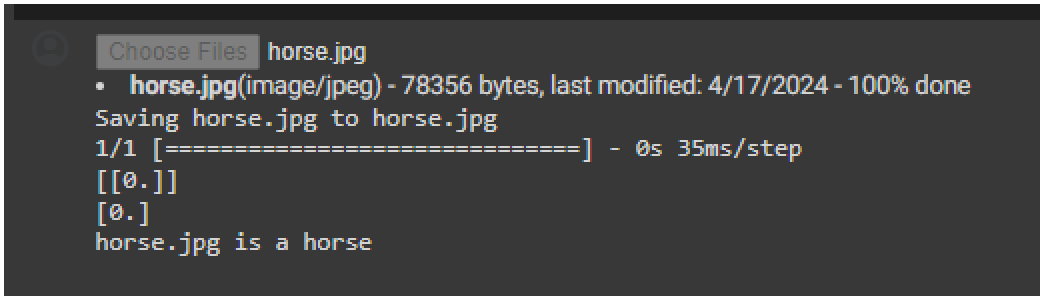
chapter3/Horse_or_Human_WithAugmentation.ipynb

chapter3/multiclass.ipynb

chapter3/transfer_learning-cats-dogs.ipynb

Once imported, follow the cells to observe the code in action and gain familiarity with Tensorflow, Keras, and importing outside datasets into Colab.This model will classify images as *horse*, or *human*.

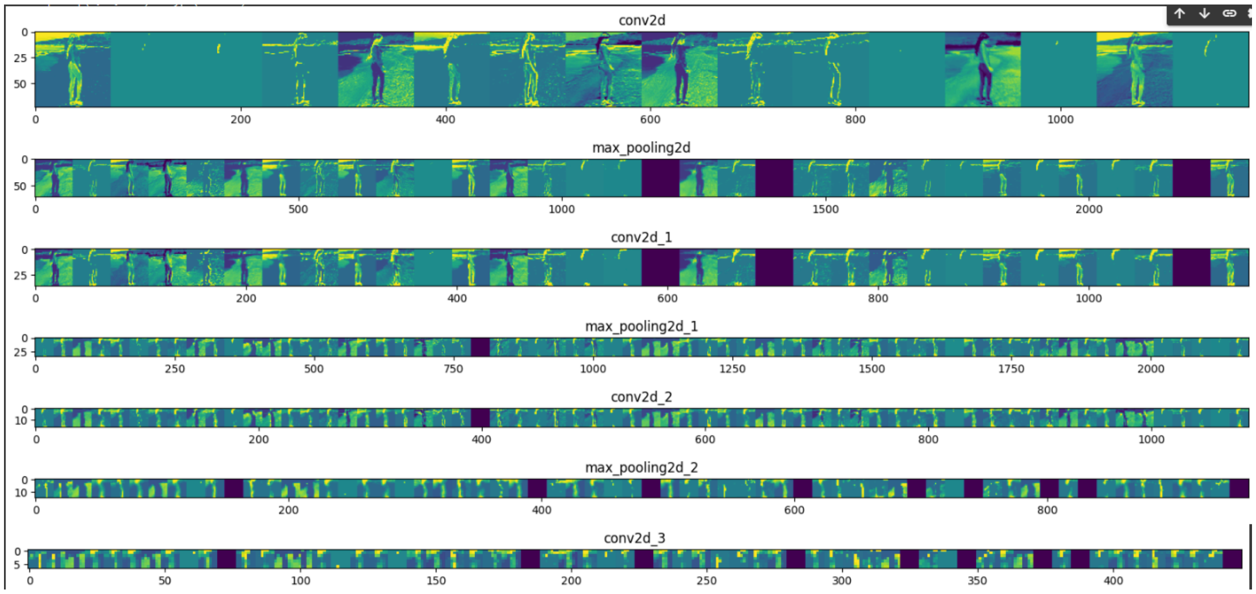
Once you reach the training section, make sure to try uploading a random image and see the model at work. Download a JPG from Google Images or provide your own human or horse picture, then upload.



```
Choose Files horse.jpg
• horse.jpg(image/jpeg) - 78356 bytes, last modified: 4/17/2024 - 100% done
Saving horse.jpg to horse.jpg
1/1 [==============================] - 0s 35ms/step
[[0.]]
[0.]
horse.jpg is a horse
```

If you find an example that fails, consider why that image may have escaped - is it cartoonish or unusual compared to the training set? Does it depict both classes? Are there glyphs or anomalies in the picture that might throw the simplistic model off track? Understanding what a good training dataset requires is an important aspect of creating good neural network designs. Datasets are a critical component of the engineering process, and require a rigorous examination of the problem you are trying to solve with your model.

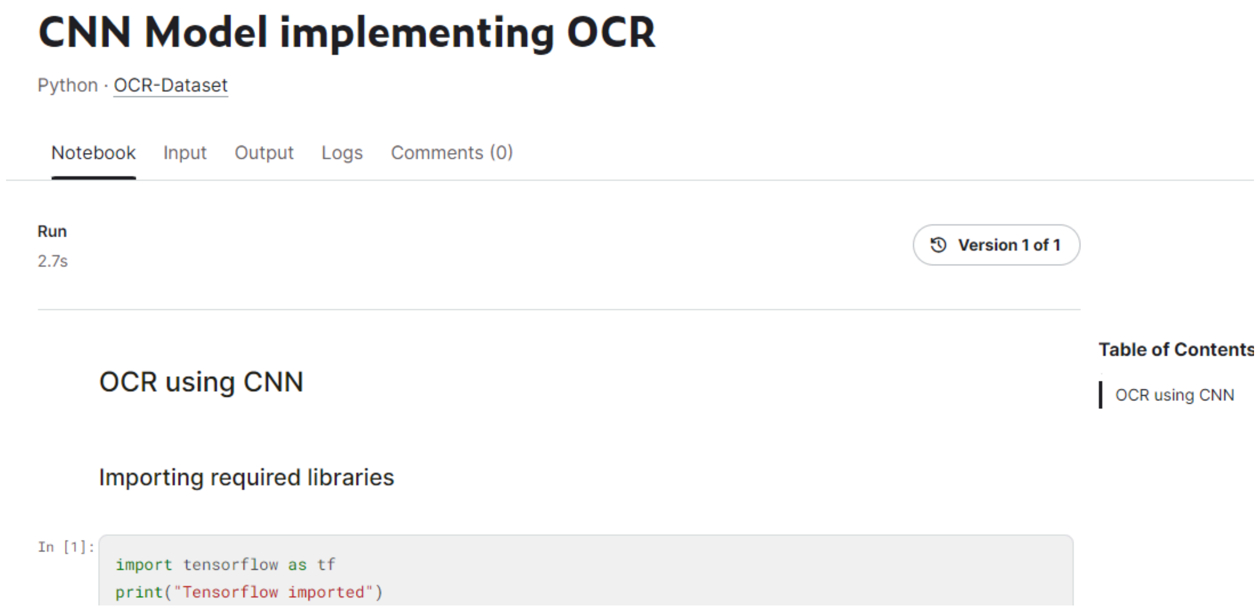
This notebook provides an excellent illustration of the actual actions of a convolutional network, with a display of the feature maps in each layer for better intuition on how these layers work.



Exercise 3: Using a Kaggle Model in Colab

We're going to download "CNN Model implementing OCR" from Kaggle, then correct it by using environment tools and documentation, and finally run it from Colab.

Our selected notebook:
https://www.kaggle.com/code/harieh/cnn-model-implementing-ocr/notebook



CNN Model implementing OCR

Python · OCR-Dataset

Notebook Input Output Logs Comments (0)

Run
2.7s

Version 1 of 1

Table of Contents
OCR using CNN

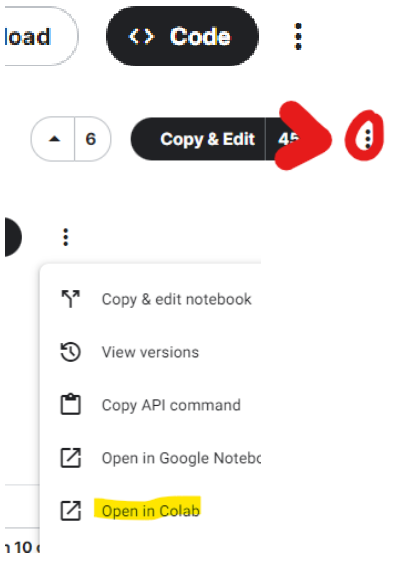
OCR using CNN

Importing required libraries

```
In [1]:
import tensorflow as tf
print("Tensorflow imported")
```

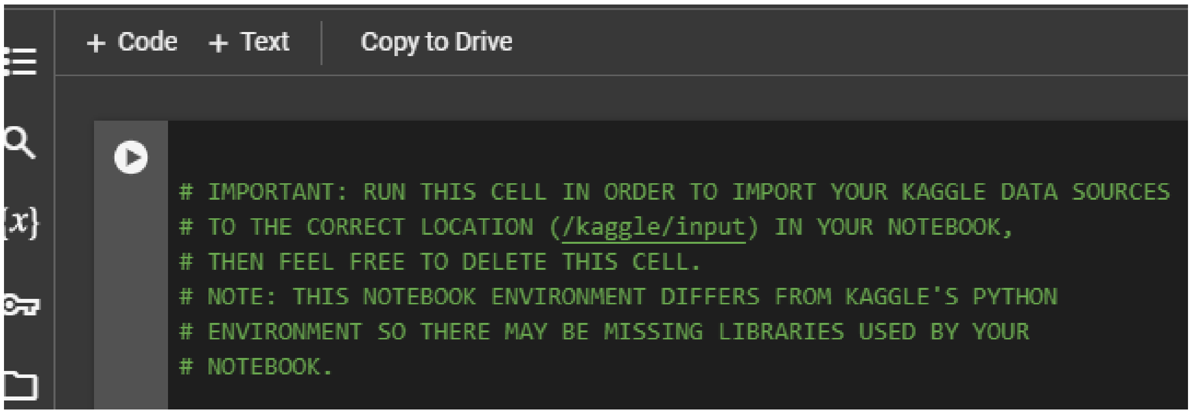
Use a Kaggle model in CoLab

- Open the model of interest and click **Code** → Select the notebook → **… menu** (next to **Copy & Edit**)
  -> **Open in Colab**



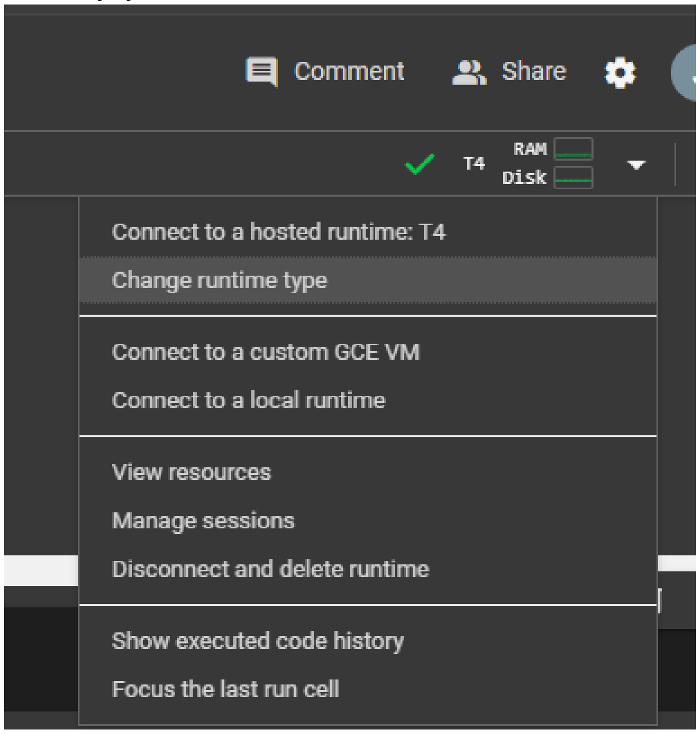
- You should now be brought to Colab with a copy of the selected notebook in your lab window.

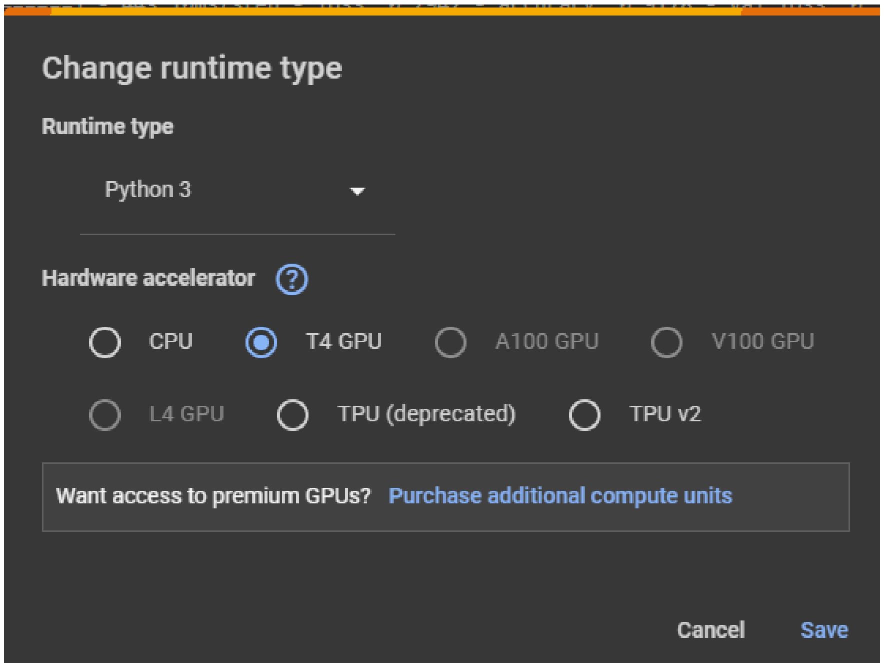
From Colab, ensure the model downloads a copy of the data needed - you should see this at the top of the model after importing from Kaggle; if so, you will only need to run this cell once for each time you start up a kernel and workspace.



```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.
```

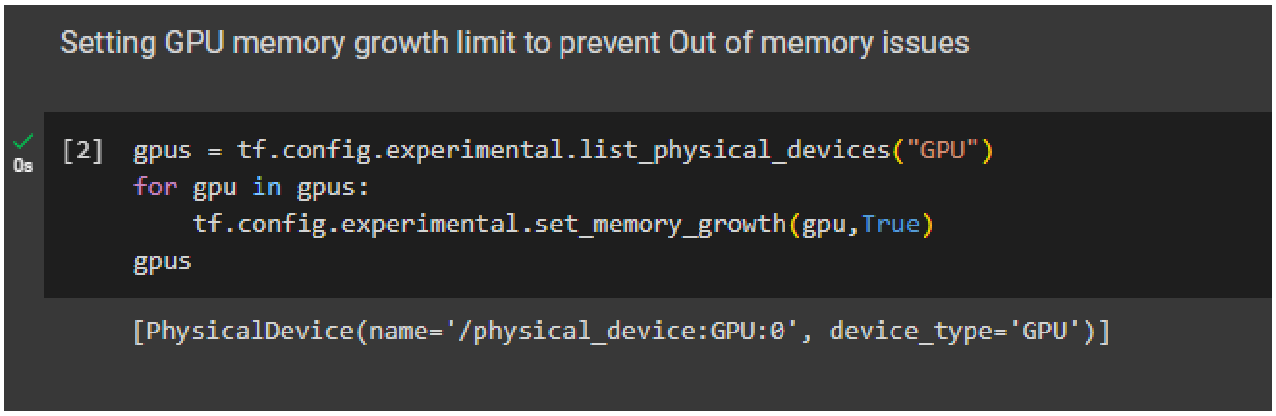
Next we'll change our running kernel from a CPU instance to a GPU instance. If you do this correctly, you'll see the GPU listed in a later section.


Comment
Share
T4
RAM
Disk
Connect to a hosted runtime: T4
Change runtime type
Connect to a custom GCE VM
Connect to a local runtime
View resources
Manage sessions
Disconnect and delete runtime
Show executed code history
Focus the last run cell



Hardware accelerator: ( ) CPU [✓ T4 GPU] ( ) A100 GPU ( ) V100 GPU ( ) L4 GPU ( ) TPU (deprecated) ( ) TPU v2

This will result in a listed GPU for the "Setting GPU Memory" section - double check if you have no GPUs listed between the square brackets!



Setting GPU memory growth limit to prevent Out of memory issues

```
[2] gpus = tf.config.experimental.list_physical_devices("GPU")
    for gpu in gpus:
        tf.config.experimental.set_memory_growth(gpu,True)
    gpus

    [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

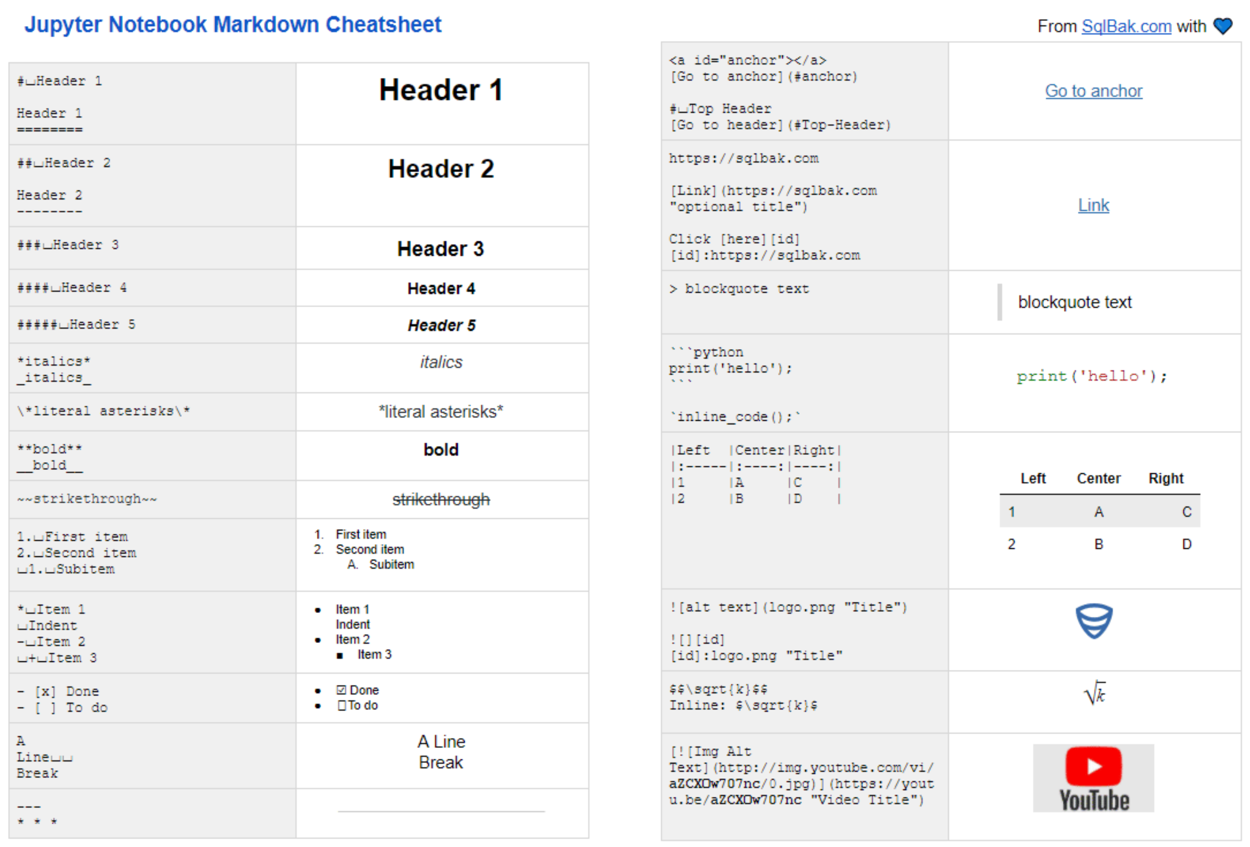
You'll encounter another error during the accuracy analysis using MatPlotLib, on the function plt.savefig. You can easily look up any of these functions to validate the expected inputs and outputs, eg. https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.savefig.html - the notebook function call appears to be trying to save the generated image to a nonexistent path. This may happen when importing from Kaggle directly, so you'll need to watch for errors like this, but they're easy to fix.

Checking the filesystem again, there is no path. We're going to create it so the figure can be saved. Follow the same process and you will be able to run the cell, saving the accuracy image.

Quick Tips

Jupyter notebook install (https://jupyter.org/install): pip3 install notebook
You can run notebooks locally even when offline using your own device.

How to Use Jupyter Notebooks (quick article on the basics):
https://www.codecademy.com/article/how-to-use-jupyter-notebooks



Jupyter Notebook Markdown Cheatsheet

From SqlBak.com with 💙

`#␣Header 1`
 	
`Header 1`
`========`
	Header 1

`##␣Header 2`
 	
`Header 2`
`--------`
	Header 2

`###␣Header 3`	**Header 3**
`####␣Header 4`	**Header 4**
`#####␣Header 5`	***Header 5***
`*italics*` `_italics_`	*italics*
`\*literal asterisks\*`	*literal asterisks*
`**bold**` `__bold__`	**bold**
`~~strikethrough~~`	~~strikethrough~~

`1.␣First item`
`2.␣Second item`
`␣1.␣Subitem`
	1. First item
	2. Second item
	A. Subitem

`*␣Item 1`
`␣Indent`
`-␣Item 2`
`␣+␣Item 3`
	• Item 1
	Indent
	• Item 2
	▪ Item 3

`- [x] Done`
`- [ ] To do`
	• ☑ Done
	• ☐ To do

`A`
`Line␣␣`
`Break`
	A Line
	Break

`---`
`* * *`	

`<a id="anchor"></a>`
`[Go to anchor](#anchor)`
 	
`#␣Top Header`
`[Go to header](#Top-Header)`
	Go to anchor

`https://sqlbak.com`
 	
`[Link](https://sqlbak.com "optional title")`
 	
`Click [here][id]`
`[id]:https://sqlbak.com`
	Link

`> blockquote text`	blockquote text

`
```
python `
`print('hello');`
`
```
 `
 	
`` `inline_code();` ``
	`print('hello');`

`|Left  |Center|Right|`
`|:-----|:----:|----:|`
`|1     |A     |C    |`
`|2     |B     |D    |`
	Left Center Right
	1 A C
	2 B D

`![alt text](logo.png "Title")`
 	
`![][id]`
`[id]:logo.png "Title"`	

`$$\sqrt{k}$$`
`Inline: $\sqrt{k}$`
	$\sqrt{k}$

`[![Img Alt Text](http://img.youtube.com/vi/aZCXOw707nc/0.jpg)](https://youtu.be/aZCXOw707nc "Video Title")`	YouTube

https://sqlbak.com/blog/wp-content/uploads/2020/12/jupyter_cheat_sheet3.png

You won't be able to import a Kaggle notebook into Colab unless you link your Google account and import code to Colab afterwards.

Resource Links

Types of Neural Networks

The Neural Zoo: https://www.asimovinstitute.org/neural-network-zoo/

The Neural Network Zoo Prequel: Cells and Layers:
https://www.asimovinstitute.org/author/fjodorvanveen/

More on the Zoo from Towards Data Science write Andrew Tch:
https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

Basic NNs & Code - Starting Points!

Deep Learning for Developers: Tools You Can Use to Code Neural Networks on Day 1:
https://www.freecodecamp.org/news/deep-learning-for-developers-tools-you-can-use-to-code-neural-networks-on-day-1-34c4435ae6b

Learning code examples (including exercise #2) from "*AI and Machine Learning for Coders*" by Laurence Moroney: https://github.com/lmoroney/tfbook

The Shape of Tensor: https://medium.com/@schartz/the-shape-of-tensor-bab75001d7bc

Google Intro to ML: https://developers.google.com/machine-learning/intro-to-ml
Google ML Crash Course: https://developers.google.com/machine-learning/crash-course

Optimization, Error/Cost

Training Deep Neural Networks:
https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964

3Blue1Brown (Neural Networks - Visually Explained):
https://www.3blue1brown.com/topics/neural-networks

Loss Functions in Neural Networks:
https://www.theaidream.com/post/loss-functions-in-neural-networks

Optimizers - EXPLAINED!: https://youtu.be/mdKjMPmcWjY?si=XyGVOsuQ5iWryIQ5

Convolutional Neural Networks

Introduction to Pooling Layers: https://towardsai.net/p/l/introduction-to-pooling-layers-in-cnn

Convolutional Neural Networks (CNNs) explained:
https://youtu.be/YRhxdVk_sIs?si=7Uw_2ryHtPQwGiVO

Retrieval Augmented Generation

Running a RAG Locally:
https://mobiarch.wordpress.com/2024/02/19/run-rag-locally-using-mistral-ollama-and-langchain/