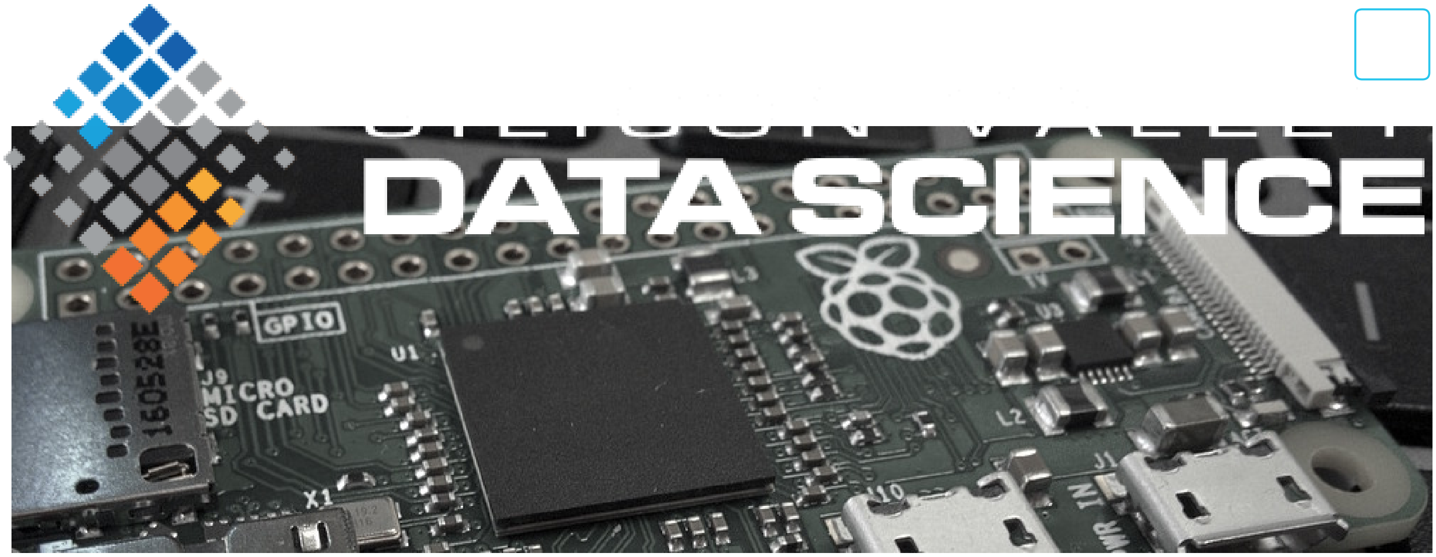


TensorFlow Image Recognition on a Raspberry Pi

FEBRUARY 8TH, 2017

Editor's note: This post is part of our Trainspotting series, a deep dive into the visual and audio detection components of our Caltrain project. You can find the introduction to the series here.

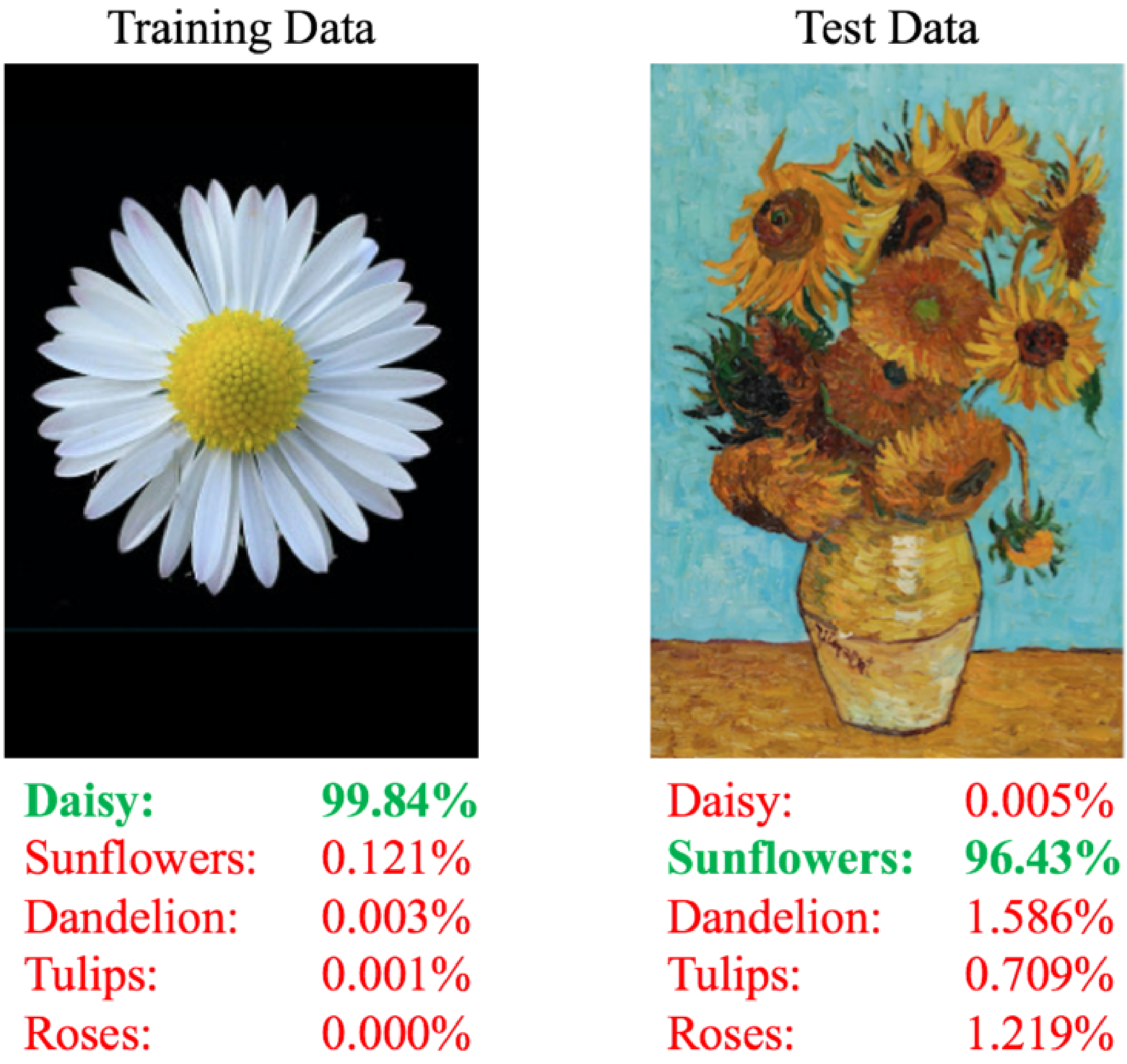
SVDS has previously used real-time, publicly available data to improve Caltrain arrival predictions. However, the station-arrival time data from Caltrain was not reliable enough to make accurate predictions. Using a Raspberry PiCamera and USB microphone, we were able to detect trains, their speed, and their direction. When we set up a new Raspberry Pi in our Mountain View office, we ran into a big problem: the Pi was not only detecting Caltrains (true positive), but also detecting Union Pacific freight trains and the VTA light rail (false positive). In order to reliably detect Caltrain delays, we would have to reliably classify the different trains.

Traditional contextual image classification techniques would not suffice, as the Raspberry Pis were placed throughout the Caltrain system at different distances, heights, and orientations from the train tracks. We were also working on a short deadline, and did not have enough time to manually select patterns and features for every Raspberry Pi in our system.

TensorFlow to the rescue

2016 was a good year to encounter this image classification problem, as several deep learning image recognition technologies had just been open sourced to the public. We chose to use Google's TensorFlow convolutional neural networks because of its handy Python libraries and ample online documentation. I had read TensorFlow for Poets by Pete Warden, which walked through how to create a custom image classifier on top of the high performing Inception V3 model. Moreover, I could use my laptop to train an augmented version of this new model overnight. It was useful to not need expensive GPU hardware, and to know that I could fine tune the model in the future.

I started with the Flowers tutorial on the TensorFlow tutorials page. I used the command line interface to classify images in the dataset, as well as custom images like Van Gough's Vase With Twelve Sunflowers.



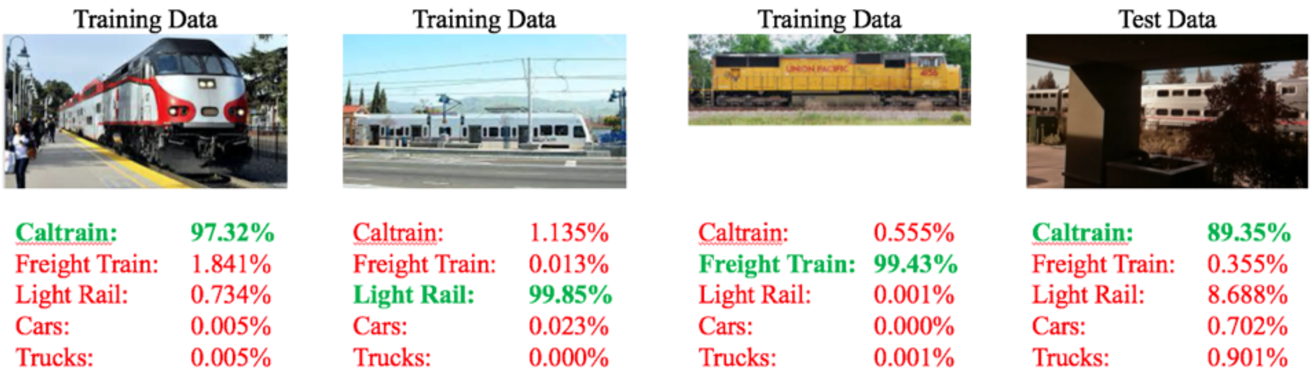
Training Data		Test Data	
Daisy:	99.84%	Daisy:	0.005%
Sunflowers:	0.121%	Sunflowers:	96.43%
Dandelion:	0.003%	Dandelion:	1.586%
Tulips:	0.001%	Tulips:	0.709%
Roses:	0.000%	Roses:	1.219%

Now that I had experience creating an image classifier using TensorFlow, I wanted to create a robust unbiased image recognition model for trains. While I could have used previous images captured by our Raspberry Pis, I decided to train on a larger more varied dataset. I also included cars and trucks, as these could also pass by the Raspberry Pi detectors at some locations. To get a training data set, I utilized Google Images to find 1000 images for the Vehicle classifier:

- Caltrains
- Freight Trains
- Light Rail
- Trucks
- Cars

Testing and deploying the model

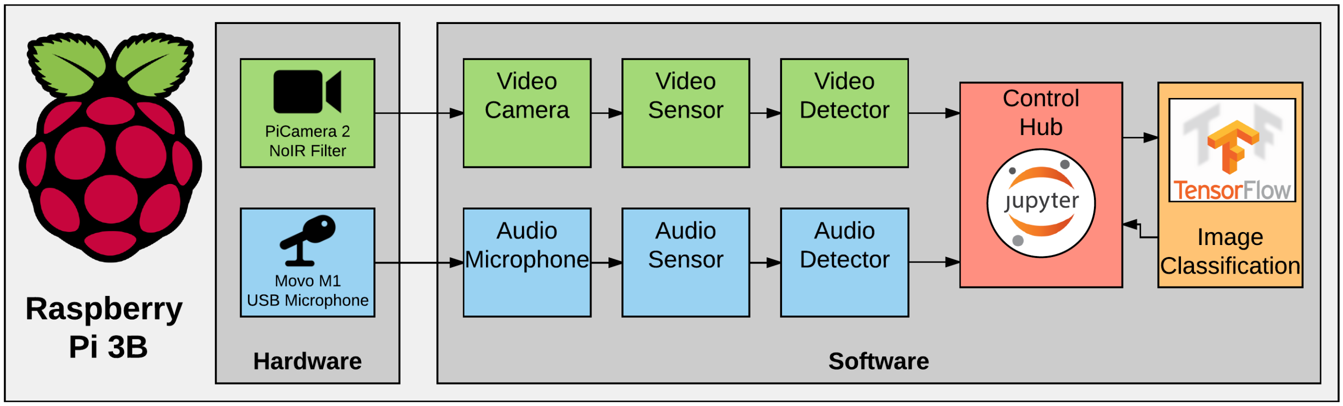
After letting the model train overnight, I returned to my desk the next morning to see how the model performed. I first tested against images not included in the training set, and was surprised to see that the classifier always seemed to pick the correct category. This included images withheld from the training set that were obtained from google images, and also included images taken from the Raspberry Pi.



Training Data		Training Data		Training Data		Test Data	
Caltrain:	97.32%	Caltrain:	1.135%	Caltrain:	0.555%	Caltrain:	89.35%
Freight Train:	1.841%	Freight Train:	0.013%	Freight Train:	99.43%	Freight Train:	0.355%
Light Rail:	0.734%	Light Rail:	99.85%	Light Rail:	0.001%	Light Rail:	8.688%
Cars:	0.005%	Cars:	0.023%	Cars:	0.000%	Cars:	0.702%
Trucks:	0.005%	Trucks:	0.000%	Trucks:	0.001%	Trucks:	0.901%

I performed the image classification on the Raspberry Pi to keep the devices affordable. Additionally, as I couldn't guarantee a speedy internet connection, I needed to perform the classification on the device to avoid delays in sending images to a central server.

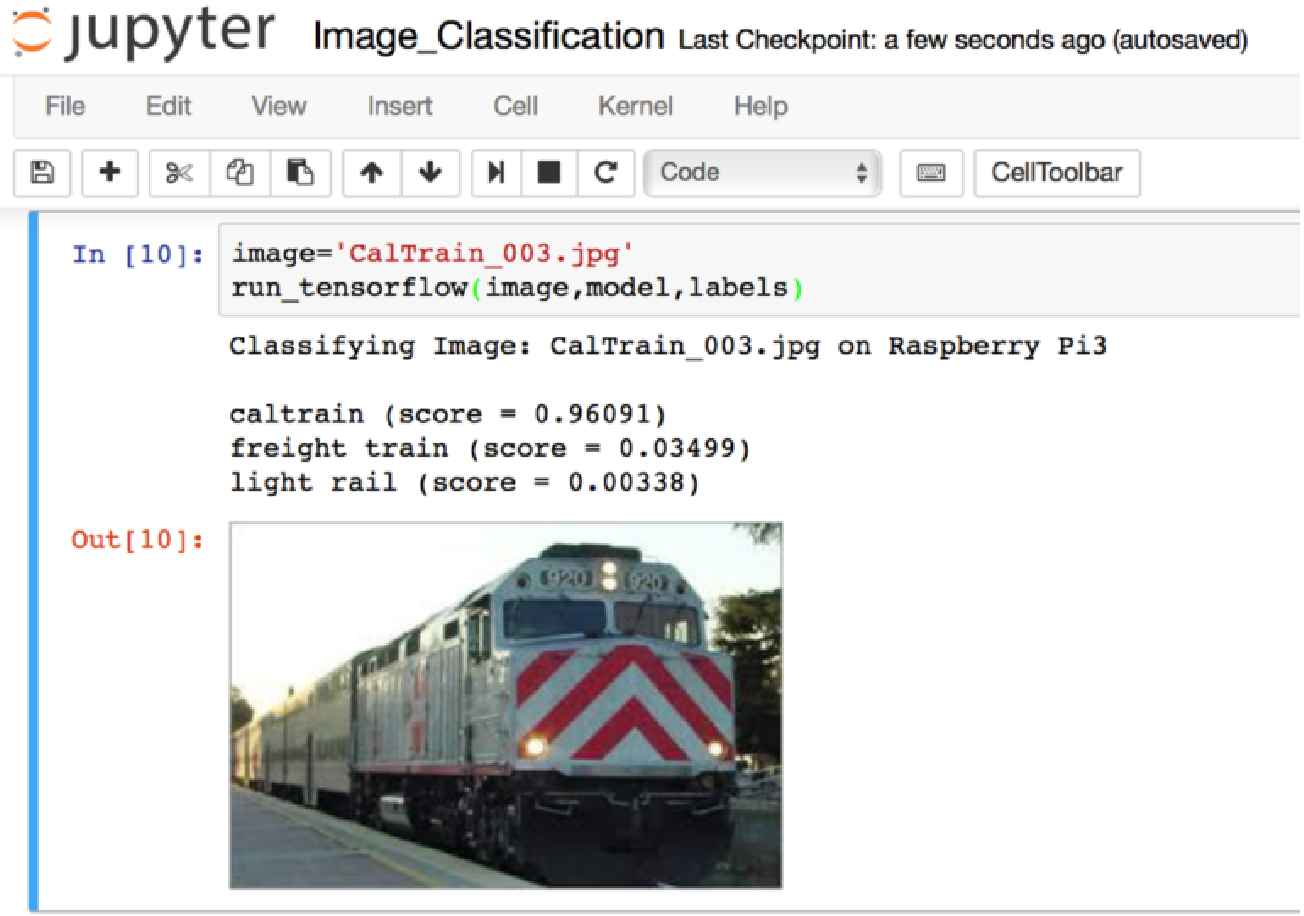
The Raspberry Pi3 has enough horsepower to do on-device stream processing so that we could send smaller, processed data streams over internet connections, and the parts are cheap. The total cost of the hardware for this sensor is $130, and the code relies only on open source libraries. I used JupyterHub for the testing, allowing me to control Raspberry Pis in multiple locations. With a working Vehicle classifier set, I next loaded the model onto a Raspberry Pi and implemented it in the audiovisual streaming architecture.



In order to compile TensorFlow on the Raspberry Pi 32 Bit ARM chip, I followed directions from Sam Abraham's small community of Pi-TensorFlow enthusiasts, and chatted with Pete Warden and the TensorFlow team at Google.

Troubleshooting TensorFlow on the Raspberry Pi

While it is well documented how to install TensorFlow on an Android or other small computer devices, most existing examples are for single images or batch processes, not for streaming image recognition use cases. Single images could be easily and robustly scored on the Pi, as a successful classification shows below.



```
image='CalTrain_003.jpg'
run_tensorflow(image,model,labels)
```

```
Classifying Image: CalTrain_003.jpg on Raspberry Pi3

caltrain (score = 0.96091)
freight train (score = 0.03499)
light rail (score = 0.00338)
```

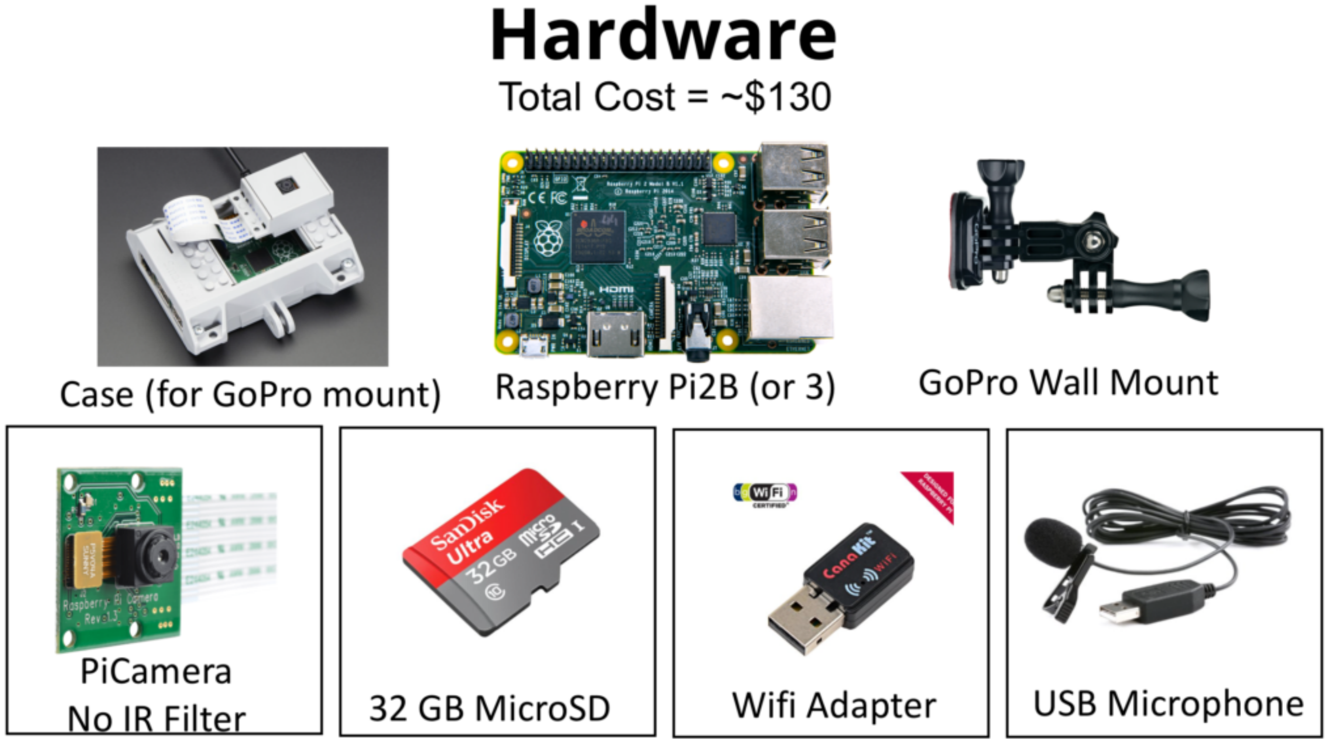
However, it was taking too long to load the 85 MB model into memory, therefore I needed to load the classifier graph to memory. With the graph now in memory, and the Raspberry Pi having a total of 1 GB of memory, plenty of computational resources exist to continuously run a camera and microphone on our custom train detection Python application.

```python
def create_and_persist_graph():
    with tf.Session() as persisted_sess:
        # Load Graph
        with tf.gfile.FastGFile(modelFullPath,'rb') as f:
            graph_def = tf.GraphDef()
            graph_def.ParseFromString(f.read())
            persisted_sess.graph.as_default()
            tf.import_graph_def(graph_def, name='')
        return persisted_sess.graph
```

That said, it was not feasible to analyze every image captured image from the PiCamera using TensorFlow, due to overheating of the Raspberry Pi when 100% of the CPU was being utilized In the end, only images of moving objects were fed to the image classification pipeline on the Pi, and TensorFlow was used to reliably discern between different types of vehicles.

Conclusion

If you are interested in classifying images in realtime using an IoT device, this is what you will need to start:



In the next post in this series, we'll look at connecting an IoT device to the cloud. Keep an eye out, and let us know in the comments if you have any questions or if there's something in particular you'd like to learn more about. To stay in touch, sign up for our newsletter.

9 Comments SVDS 1 Login

Recommend Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

rogerhyam • 8 months ago

Hi, Looks like a fun project. Could you say a little about how you did speed prediction. If the cameras were all at different heights, angles etc - or did you just do it as travel time between cameras with sufficient distance between them.

We just had a new residential speed limit set up and I'd love to do a Pi based monitoring system for how fast cars are now travelling but have yet to come up with a simple strategy.

1 ⌃ | ⌄ • Reply • Share ›

D Chandler ➦ rogerhyam • 7 months ago

Here's a project I used to set up one for my street.

https://github.com/pageauc/...

⌃ | ⌄ • Reply • Share ›

Matthew Rubashkin ➦ rogerhyam • 8 months ago

Hi **@rogerhyam** , thanks for the great question! We used a single PiCamera to detect the train speed. We did this by separating each image into three different areas of interest, and calculating the time that the train passed through the center of these areas. With knowledge of the physical separation distance of these areas within a single image, we could calculate the train speed. I describe this process in a previous blog post here: http://www.svds.com/streami...

⌃ | ⌄ • Reply • Share ›

David Tian • 2 months ago

Very interesting project! Thanks for the post.
I wonder if you could share your training and test data? I have a student project, would love to use your data. Also if your notebook available from github?

⌃ | ⌄ • Reply • Share ›

David • 4 months ago

I like to use simple user interface at https://vize.ai
Prepare dataset: https://blog.vize.ai/how-to...

⌃ | ⌄ • Reply • Share ›

SHIJITH N • 6 months ago

Hi, could you please give the procedures for installing and setting up the tensorflow and inception in raspberry pi. I have already installed tensorflow successfully but when the image recognition code in python is run, it shows error but the same code could be run in linux without any error.

⌃ | ⌄ • Reply • Share ›

Matthew Rubashkin ➦ SHIJITH N • 6 months ago

Hi **@SHIJITH N**, please see https://github.com/samjabra... for detailed instructions for setting up tensorflow on a raspberry pi

⌃ | ⌄ • Reply • Share ›

Shijith Nair ➦ Matthew Rubashkin • 6 months ago

Thank you.

⌃ | ⌄ • Reply • Share ›

Omkar Prabhu • 6 months ago

Hi, this indeed looks like a fun and helpful project. Can you please share the processing

Hi, this indeed looks like a fun and helpful project. Can you please share the processing time you ended up with for tensorflow image classification per image on the raspberry pi?

Reply • Share ›

ALSO ON **SVDS**

Pivoting Data in SparkSQL – Silicon Valley Data Science

5 comments • a year ago

David Howell — Pivoting creates Pivot-table (sometimes called a cross-tab) which has 3 (or more) dimensions. Normally a SQL table

IoT and Resilient Systems – Silicon Valley Data Science

1 comment • a year ago

Intransure Technologies — IoT is now most important thing in life. It also decreases the efforts and in every task it also reduces the

Open Source Toolkits for Speech Recognition - Silicon Valley Data Science

1 comment • 8 months ago

Harold Squid — What do you think of :CNTKhttps://github.com/microsof...English Conversational Telephone Speech …

Page not found - Silicon Valley Data Science

5 comments • 7 months ago

anson — Great article! Did you guys provide the weights to your trained model somewhere? I'd prefer to improve on it

✉ Subscribe   Add Disqus to your siteAdd DisqusAdd   🔒 Privacy   DISQUS

MATTHEW RUBASHKIN

With a background in optical physics and biomedical research, Matt has a broad range of experiences in software development, database engineering, and data analytics.

SHARE

59 507 70