

Week 4

Multiclass Classifications

LEE SEUNGWOO

주

1



완료하는 데 4시간 필요

Exploring a Larger Dataset

In the first course in this specialization, you had an introduction to TensorFlow, and how, with its high level APIs you could do basic image classification, and you learned a little bit about Convolutional Neural Networks (ConvNets). In this course you'll go deeper into using ConvNets with real-world data, and learn about techniques...

[모두 표시](#)



8 videos (Total 18 min), 5 readings, 3 quizzes [모두 보기](#)

주

2



완료하는 데 4시간 필요

Augmentation: A technique to avoid overfitting

You've heard the term overfitting a number of times to this point. Overfitting is simply the concept of being over specialized in training -- namely that your model is very good at classifying what it is trained for, but not so good at classifying things that it hasn't seen. In order to generalize your model more effectively, you will ...

[모두 표시](#)



7 videos (Total 14 min), 6 readings, 3 quizzes [모두 보기](#)

주

3



완료하는 데 4시간 필요

Transfer Learning

Building models for yourself is great, and can be very powerful. But, as you've seen, you can be limited by the data you have on hand. Not everybody has access to massive datasets or the compute power that's needed to train them effectively. Transfer learning can help solve this -- where people with models trained on...

[모두 표시](#)



7 videos (Total 14 min), 5 readings, 3 quizzes [모두 보기](#)

주

4



완료하는 데 4시간 필요

Multiclass Classifications

You've come a long way, Congratulations! One more thing to do before we move off of ConvNets to the next module, and that's to go beyond binary classification. Each of the examples you've done so far involved classifying one thing or another -- horse or human, cat or dog. When moving beyond binary into Categorical classification...

[모두 표시](#)



6 videos (Total 12 min), 5 readings, 3 quizzes [모두 보기](#)

주

1



완료하는 데 4시간 필요

Exploring a Larger Dataset

In the first course in this specialization, you had an introduction to TensorFlow, and how, with its high level APIs you could do basic image classification, and you learned a little bit about Convolutional Neural Networks (ConvNets). In this course you'll go deeper into using ConvNets with real-world data, and learn about techniques... [모두 표시](#)



8 videos (Total 18 min), 5 readings, 3 quizzes [모두 보기](#)

주

2



완료하는 데 4시간 필요

Augmentation: A technique to avoid overfitting

You've heard the term overfitting a number of times to this point. Overfitting is simply the concept of being over specialized in training -- namely that your model is very good at classifying what it is trained for, but not so good at classifying things that it hasn't seen. In order to generalize your model more effectively, you will ... [모두 표시](#)



7 videos (Total 14 min), 6 readings, 3 quizzes [모두 보기](#)

주

3



완료하는 데 4시간 필요

Transfer Learning

Building models for yourself is great, and can be very powerful. But, as you've seen, you can be limited by the data you have on hand. Not everybody has access to massive datasets or the compute power that's needed to train them effectively. Transfer learning can help solve this -- where people with models trained on... [모두 표시](#)



7 videos (Total 14 min), 5 readings, 3 quizzes [모두 보기](#)

주

4



완료하는 데 4시간 필요

Multiclass Classifications

You've come a long way, Congratulations! One more thing to do before we move off of ConvNets to the next module, and that's to go beyond binary classification. Each of the examples you've done so far involved classifying one thing or another -- horse or human, cat or dog. When moving beyond binary into Categorical cla... [모두 표시](#)

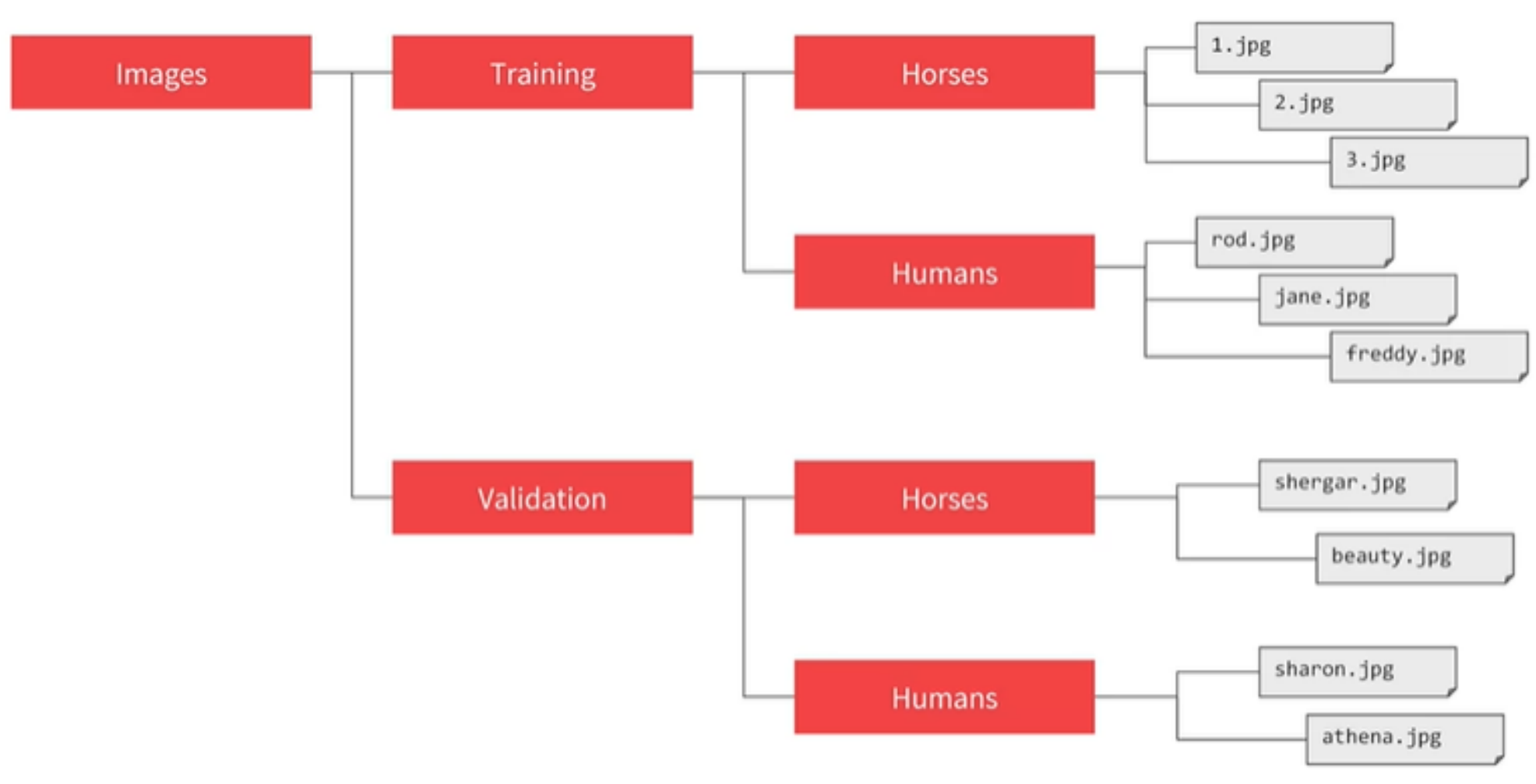


6 videos (Total 12 min), 5 readings, 3 quizzes [모두 보기](#)

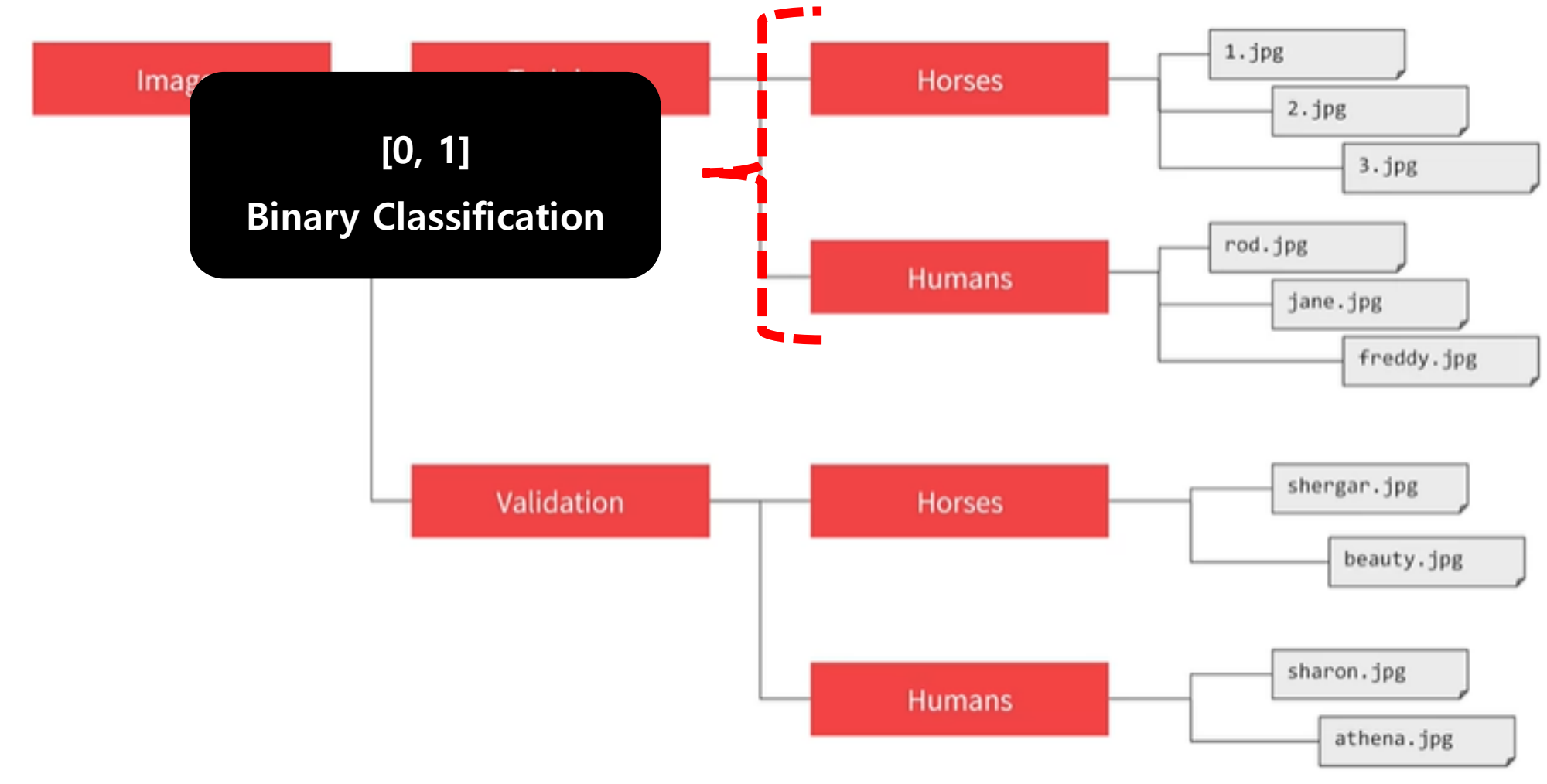
Table Of Contents

1. Moving from Binary to multi-class classification
2. Explore multi-class with Rock Paper Scissors dataset
3. Train a classifier with Rock Paper Scissors
4. Test the Rock Paper Scissors classifier

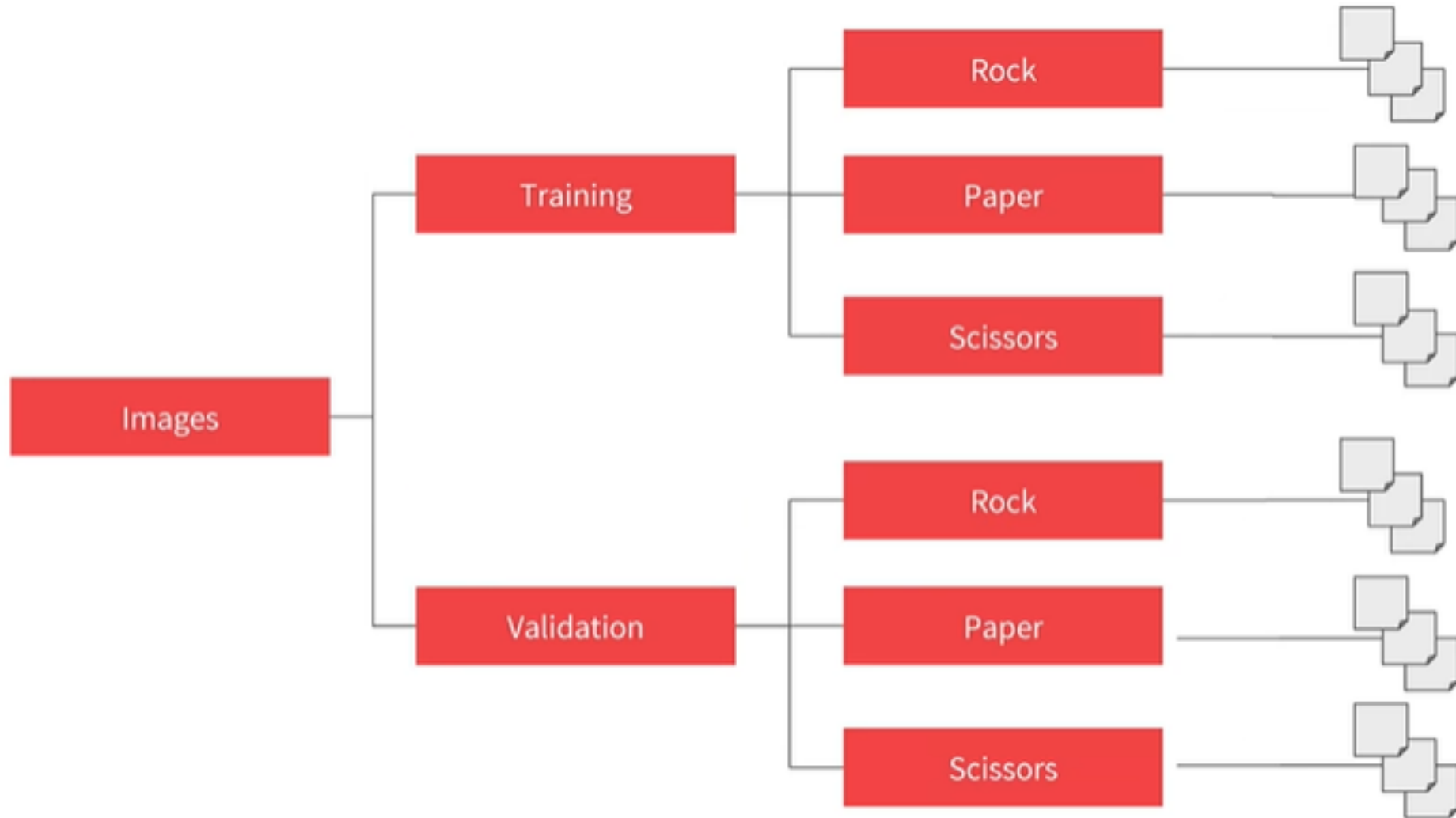
Moving from Binary to multi-class classification



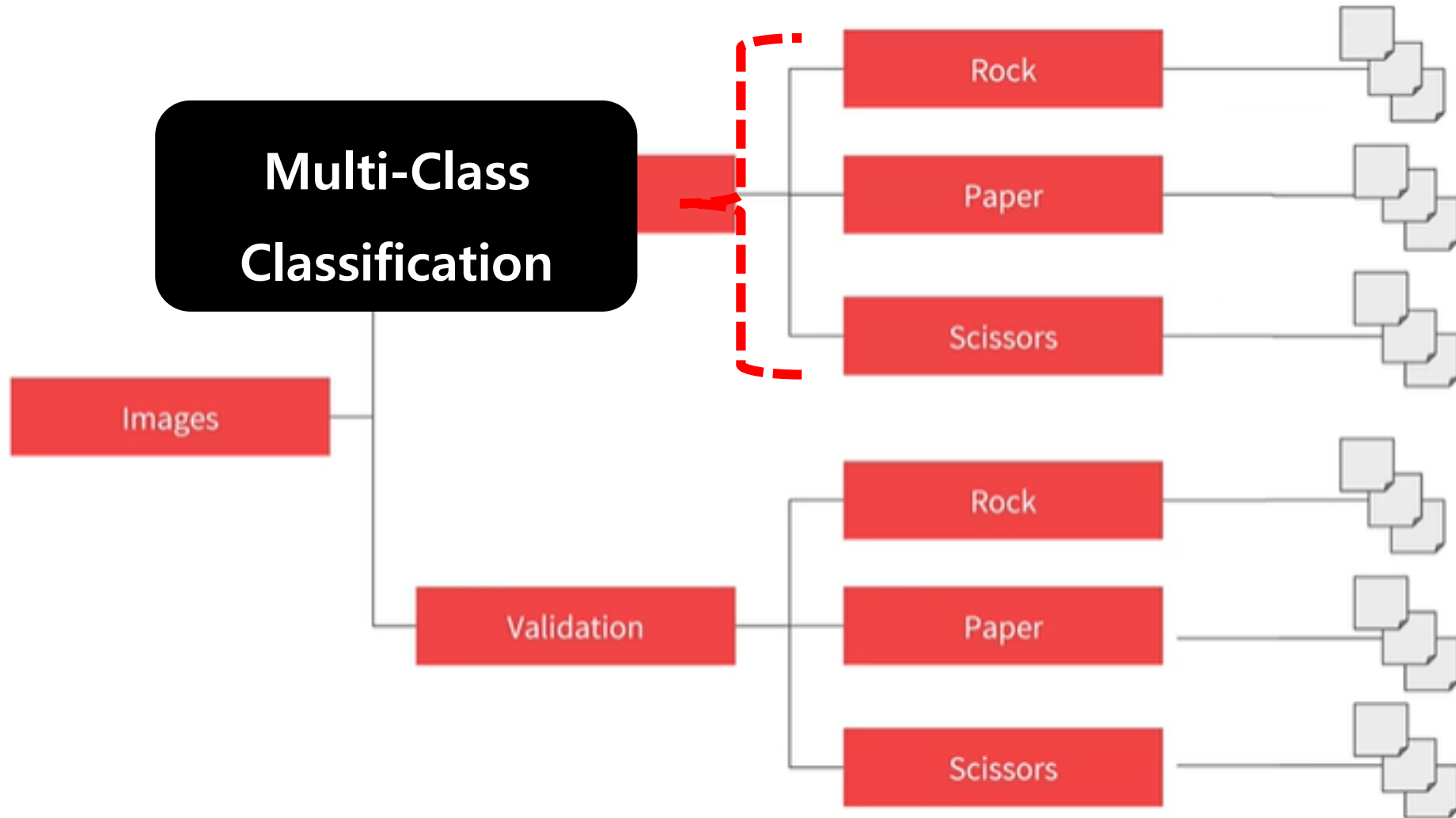
Moving from Binary to multi-class classification



Moving from Binary to multi-class classification



Moving from Binary to multi-class classification




Explore multi-class with Rock Paper Scissors dataset



Explore multi-class with Rock Paper Scissors dataset

Imoroney@
Developer Advocate // Googler // Author and more!

ABOUT ME COFFEE WITH A GOOGLER GOOGLE FIREBASE SCIFI COMICS ART



Rock Paper Scissors Dataset

Home / [Rock Paper Scissors Dataset](#)

Rock Paper Scissors Dataset

Introducing Rock Paper Scissors – A multi class learning dataset

Abstract

Rock Paper Scissors is a dataset containing 2,892 images of diverse hands in Rock/Paper/Scissors poses. It is licensed [CC By 2.0](#) and available for all purposes, but its intent is primarily for learning and research.


Overview


Rock Paper Scissors contains images from a variety of different hands, from different races, ages and genders, posed into Rock / Paper or Scissors and labelled as such. You can download the [training set here](#), and the [test set here](#). These images have all been generated using CGI techniques as an experiment in determining if a CGI-based dataset can be used for classification against real images. I also generated a few images that you can use for predictions. You can find them [here](#).

Note that all of this data is posed against a white background.



Each image is 300x300 pixels in 24-bit color.


FOLLOW ME ON TWITTER

 Laurence Moroney Retweeted


 **healthproai**
@healthproai

@deeplearningai_ & @Google recently launched a #TensorFlow specialization. Thanks to their brilliant efforts, the #DeepLearning community will grow bigger and smarter. Special thanks to @AndrewYNg and @Imoroney https://twitter.com/deeplearningai_/status/1157023755744362496

  Aug 4, 2019

 **Laurence Moroney**
@Imoroney

At #StarTrek night with the #SeattleMariners .. Live long and prosper!



<http://www.laurencemoroney.com/rock-paper-scissors-dataset/>

Explore multi-class with Rock Paper Scissors dataset

```
1
2  from tensorflow.keras.preprocessing.image import ImageDataGenerator
3
4  train_datagen = ImageDataGenerator(rescale=1./255)
5
6  train_generator = train_datagen.flow_from_directory(
7      train_dir,
8      target_size=(300, 300), # 300 * 300 사이즈
9      batch_size=128,         # 배치 사이즈 128개
10     class_mode='binary'     # Binary
11 )
```

Explore multi-class with Rock Paper Scissors dataset

```
1
2  from tensorflow.keras.preprocessing.image import ImageDataGenerator
3
4  train_datagen = ImageDataGenerator(rescale=1./255)
5
6  train_generator = train_datagen.flow_from_directory(
7      train_dir,
8      target_size=(300, 300),    # 300 * 300 사이즈
9      batch_size=128,            # 배치 사이즈 128개
10     class_mode='categorical'    # Categorical
11 )
```

Explore multi-class with Rock Paper Scissors dataset

```
1
2 from tensorflow.keras.preprocessing.image import ImageDataGenerator
3
4 train_datagen = ImageDataGenerator(rescale=1./255)
5
6 train_generator = train_datagen.flow_from_directory(
7     train_dir,
8     target_size=(300, 300),    # 300 * 300 사이즈
9     batch_size=128,           # 배치 사이즈 128개
10    class_mode='categorical'   # Categorical
11 )
```

Explore multi-class with Rock Paper Scissors dataset

```
13 import tensorflow as tf
14
15 model = tf.keras.models.Sequential([
16     tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(300, 300, 3))
17     tf.keras.layers.MaxPooling2D(2, 2),
18     tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
19     tf.keras.layers.MaxPooling2D(2, 2),
20     tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
21     tf.keras.layers.MaxPooling2D(2, 2),
22     tf.keras.layers.Flatten(),
23     tf.keras.layers.Dense(512, activation='relu'),
24     tf.keras.layers.Dense(1, activation='sigmoid')
25 ])
```

Explore multi-class with Rock Paper Scissors dataset

```
13 import tensorflow as tf
14
15 model = tf.keras.models.Sequential([
16     tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(300, 300, 3))
17     tf.keras.layers.MaxPooling2D(2, 2),
18     tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
19     tf.keras.layers.MaxPooling2D(2, 2),
20     tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
21     tf.keras.layers.MaxPooling2D(2, 2),
22     tf.keras.layers.Flatten(),
23     tf.keras.layers.Dense(512, activation='relu'),
24     tf.keras.layers.Dense(3, activation='softmax')
25 ])
```

Explore multi-class with Rock Paper Scissors dataset

```
13 import tensorflow as tf
14
15 model = tf.keras.models.Sequential([
16     tf.keras.layers.Conv2D(16, (3, 3), activation='relu', input_shape=(300, 300, 3))
17     tf.keras.layers.MaxPooling2D(2, 2),
18     tf.keras.layers.Conv2D(32, (3, 3), activation='relu'),
19     tf.keras.layers.MaxPooling2D(2, 2),
20     tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
21     tf.keras.layers.MaxPooling2D(2, 2),
22     tf.keras.layers.Flatten(),
23     tf.keras.layers.Dense(512, activation='relu'),
24     tf.keras.layers.Dense(3, activation='softmax')
25 ])
```


Q: Why Activation Function is “Softmax”?

How does Sigmoid activation work in multi-class classification problems

Asked 10 months ago Active 6 months ago Viewed 3k times

▲
3
▼ I know that for a problem with multiple classes we usually use softmax, but can we also use sigmoid? I have tried to implement digit classification with sigmoid at the output layer, it works. What I don't understand is how does it work?

machine-learning

neural-network

deep-learning

multiclass-classification

activation-function



share improve this question

2

add a comment

edited Oct 6 '18 at 19:56



Vaalizaadeh

8,548 ● 6 ● 25 ● 68

asked Oct 6 '18 at 8:41



bharath chandra

16 ● 1 ● 3

Blog

What Every D
Early On

DEF CON and
Traffic Says A

Featured on Meta

<https://datascience.stackexchange.com/questions/39264/how-does-sigmoid-activation-work-in-multi-class-classification-problems>

Q: Why Activation Function is “Softmax”?

Hov 2

Asked

softmax() will give you the probability distribution which means all output will sum to 1. While, sigmoid() will make sure the output value of neuron is between 0 to 1.

In case of digit classification and sigmoid(), you will have output of 10 output neurons between 0 to 1. Then, you can take biggest one of them and classify as that digit.

share improve this answer

answered Oct 6 '18 at 19:01



Preet

458 ● 5

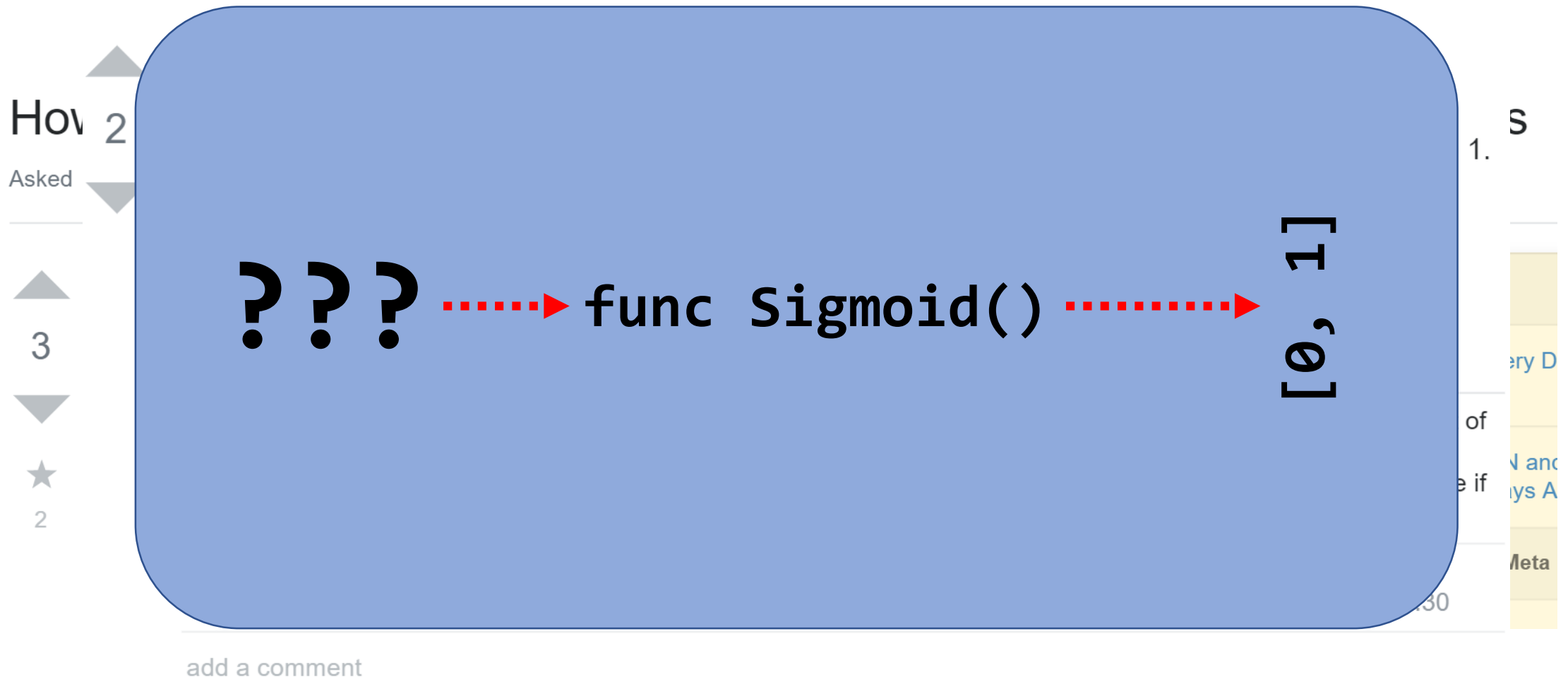
So what you are saying is both works same? So softmax calculates the probability of one neuron with respect of all others and then returns neuron that has maximum probability whereas when using sigmoid it generates output for each neuron independently and the neuron that has maximum output is returned. Please correct me if I am wrong.. – [bharath chandra](#) Oct 7 '18 at 3:00

Yes, both work the same way. Softmax is an extension of sigmoid for multi-class classifications problem. Softmax in multiclass logistic regression with K=2 takes the form of sigmoid function. – [Preet](#) Feb 10 at 11:30

add a comment

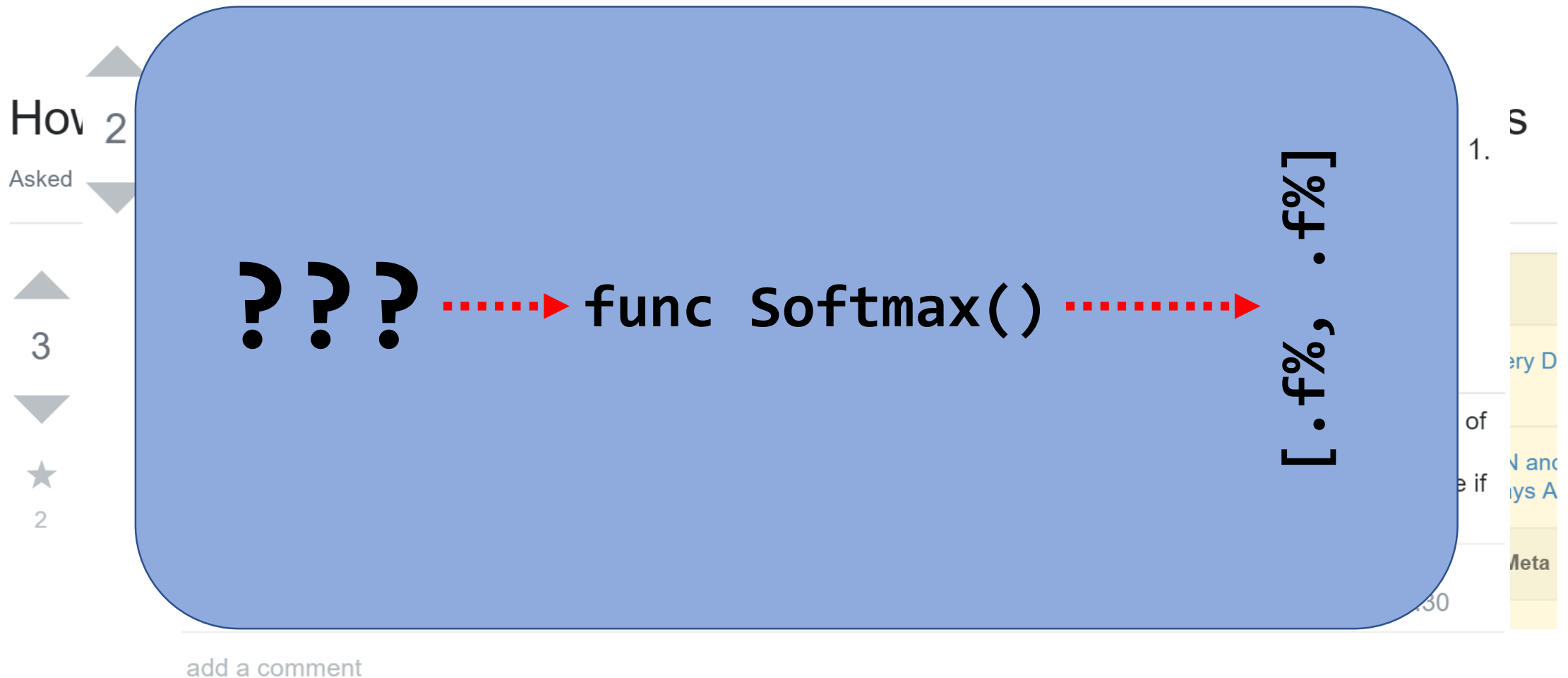
<https://datascience.stackexchange.com/questions/39264/how-does-sigmoid-activation-work-in-multi-class-classification-problems>

Q: Why Activation Function is “Softmax”?



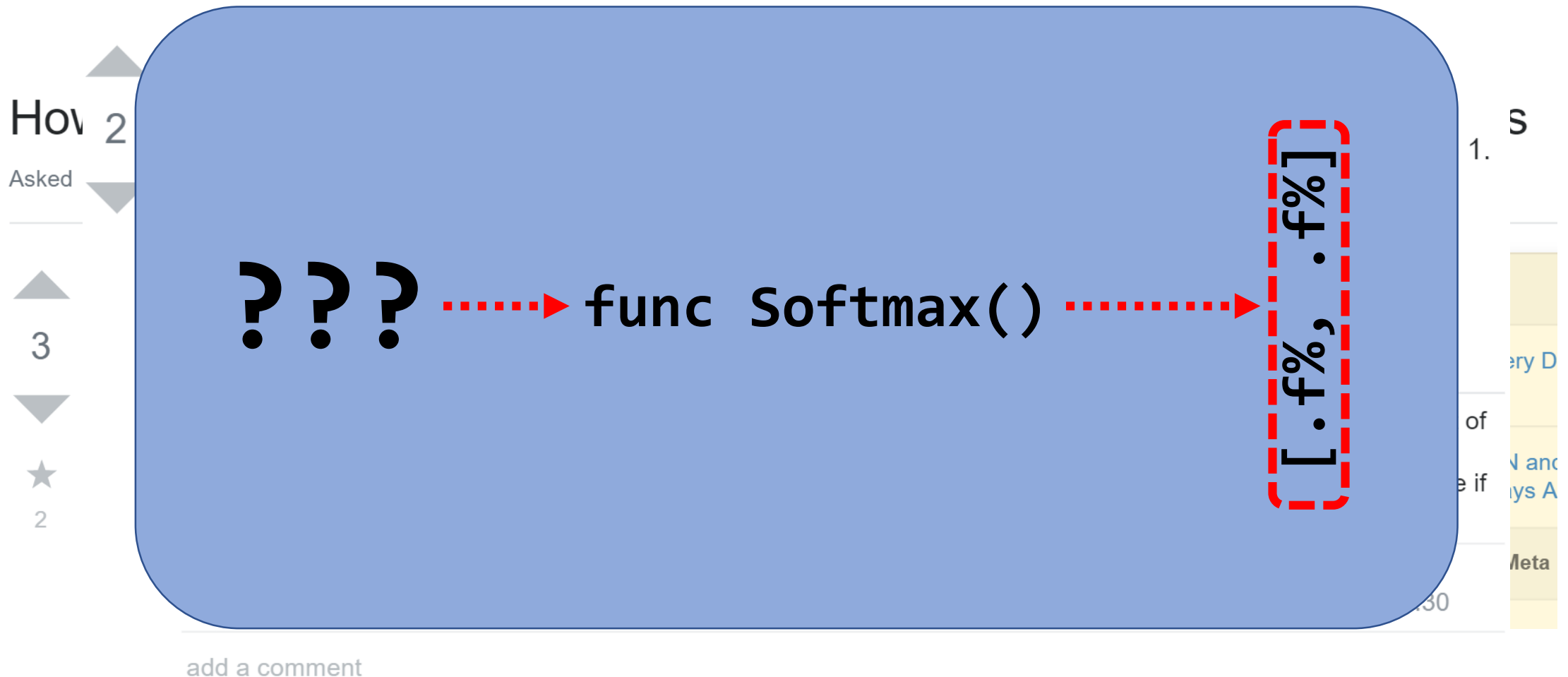
<https://datascience.stackexchange.com/questions/39264/how-does-sigmoid-activation-work-in-multi-class-classification-problems>

Q: Why Activation Function is “Softmax”?



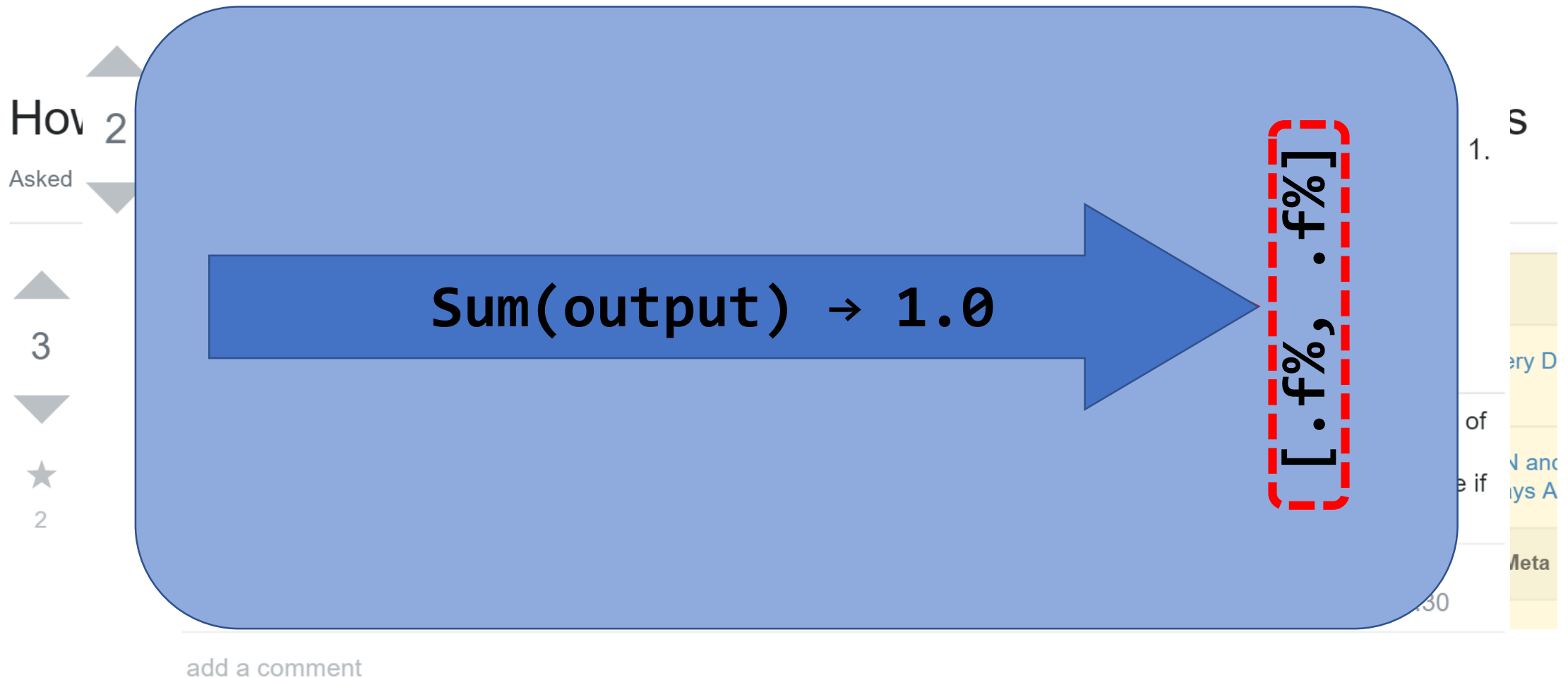
<https://datascience.stackexchange.com/questions/39264/how-does-sigmoid-activation-work-in-multi-class-classification-problems>

Q: Why Activation Function is “Softmax”?



<https://datascience.stackexchange.com/questions/39264/how-does-sigmoid-activation-work-in-multi-class-classification-problems>

Q: Why Activation Function is “Softmax”?



<https://datascience.stackexchange.com/questions/39264/how-does-sigmoid-activation-work-in-multi-class-classification-problems>

Explore multi-class with Rock Paper Scissors dataset



Rock: 0.001 Paper: 0.647 Scissors: 0.352

Explore multi-class with Rock Paper Scissors dataset



Rock: 0.001

Paper: 0.647

Scissors: 0.352

Explore multi-class with Rock Paper Scissors dataset

```
27  from tensorflow.keras.optimizers import RMSprop
28
29  model.compile(
30      loss = 'binary_crossentropy',
31      optimizer = RMSprop(lr=0.001),
32      metrics=['acc']
33  )
```

Explore multi-class with Rock Paper Scissors dataset

```
27 from tensorflow.keras.optimizers import RMSprop
28
29 model.compile(
30     loss = 'binary_crossentropy',
31     optimizer = RMSprop(lr=0.001),
32     metrics=['acc']
33 )
```

Explor

Methods

compile

[View source](#)

taset

prop

```
27 f compile(  
28     optimizer,  
29     loss=None,  
30     metrics=None,  
31     loss_weights=None,  
32     sample_weight_mode=None,  
33     weighted_metrics=None,  
    target_tensors=None,  
    distribute=None,  
    **kwargs  
)
```

Configures the model for training.

Arguments:

- **optimizer**: String (name of optimizer) or optimizer instance. See [tf.keras.optimizers](#).
- **loss**: String (name of objective function), objective function or `tf.losses.Loss` instance. See [tf.losses](#). If the model has multiple outputs, you can use a different loss on each output by passing a dictionary or a list of losses. The loss value that will be minimized by the model will then be the sum of all individual losses.

https://www.tensorflow.org/api_docs/python/tf/keras/Model

Explor

Methods



compile

taset

TensorFlow 2.0 Beta is available [Learn more](#)

TensorFlow > API > TensorFlow Core r1.14 > Python ☆☆☆☆☆

Module: tf.keras.losses

 TensorFlow 2.0 version  View source on GitHub

Aliases:

- Module `tf.compat.v1.keras.losses`

https://www.tensorflow.org/api_docs/python/tf/keras/losses

- `loss`: String (name of objective function), objective function or `tf.losses.Loss` instance. See `tf.losses`. If the model has multiple outputs, you can use a different loss on each output by passing a dictionary or a list of losses. The loss value that will be minimized by the model will then be the sum of all individual losses.

https://www.tensorflow.org/api_docs/python/tf/keras/Model

Explor

Method
Classes

comp



TensorFlow

Model



Aliases:

- Model

`class BinaryCrossentropy`: Computes the cross-entropy loss between true labels and predicted labels.

`class CategoricalCrossentropy`: Computes the crossentropy loss between the labels and predictions.

`class CategoricalHinge`: Computes the categorical hinge loss between `y_true` and `y_pred`.

`class CosineSimilarity`: Computes the cosine similarity between `y_true` and `y_pred`.

`class Hinge`: Computes the hinge loss between `y_true` and `y_pred`.

`class Huber`: Computes the Huber loss between `y_true` and `y_pred`.

`class KLDivergence`: Computes Kullback Leibler divergence loss between `y_true` and `y_pred`.

`class LogCosh`: Computes the logarithm of the hyperbolic cosine of the prediction error.

`class Loss`: Loss base class.

`class MeanAbsoluteError`: Computes the mean of absolute difference between labels and predictions.

`class MeanAbsolutePercentageError`: Computes the mean absolute percentage error between `y_true` and `y_pred`.

`class MeanSquaredError`: Computes the mean of squares of errors between labels and predictions.

`class MeanSquaredLogarithmicError`: Computes the mean squared logarithmic error between `y_true` and `y_pred`.

`class Poisson`: Computes the Poisson loss between `y_true` and `y_pred`.

`class SparseCategoricalCrossentropy`: Computes the crossentropy loss between the labels and predictions.

`class SquaredHinge`: Computes the squared hinge loss between `y_true` and `y_pred`.

taset



P

[as/losses](#)

. If
of

[python/tf/keras/Model](#)

Explor

Method
Classes

comp

`class BinaryCrossentropy` : Computes the cross-entropy loss between true labels and predicted labels.

`class CategoricalCrossentropy` : Computes the crossentropy loss between the labels and predictions.

`class CategoricalHinge` : Computes the categorical hinge loss between `y_true` and `y_pred`.

`class CosineSimilarity` : Computes the cosine similarity between `y_true` and `y_pred`.

`class Hinge` : Computes the hinge loss between `y_true` and `y_pred`.

`class Huber` : Computes the Huber loss between `y_true` and `y_pred`.

`class KLDivergence` : Computes Kullback Leibler divergence loss between `y_true` and `y_pred`.

`class LogCosh` : Computes the logarithm of the hyperbolic cosine of the prediction error.

`class Loss` : Loss base class.

`class MeanAbsoluteError` : Computes the mean of absolute difference between labels and predictions.

`class MeanAbsolutePercentageError` : Computes the mean absolute percentage error between `y_true` and `y_pred`.

• `class MeanSquaredError` : Computes the mean of squares of errors between labels and predictions.

`class MeanSquaredLogarithmicError` : Computes the mean squared logarithmic error between `y_true` and `y_pred`.

• `class Poisson` : Computes the Poisson loss between `y_true` and `y_pred`.

`class SparseCategoricalCrossentropy` : Computes the crossentropy loss between the labels and predictions.

`class SquaredHinge` : Computes the squared hinge loss between `y_true` and `y_pred`.

TensorFlow

Model



Aliases:

• Model

taset



P

[as/losses](#)

. If
of

[python/tf/keras/Model](#)

Explore multi-class with Rock Paper Scissors dataset

```
27  from tensorflow.keras.optimizers import RMSprop
28
29  model.compile(
30      loss = 'categorical_crossentropy',
31      optimizer = RMSprop(lr=0.001),
32      metrics=['acc']
33  )
```

Explore multi-class with Rock Paper Scissors dataset

```
27  from tensorflow.keras.optimizers import RMSprop
28
29  model.compile(
30      |   loss = 'categorical_crossentropy',
31      |   optimizer = RMSprop(lr=0.001),
32      |   metrics=['acc']
33  )
```


Explore multi-class with Rock Paper Scissors dataset



Train a classifier with Rock Paper Scissors

```

lwget --no-check-certificate \
https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps.zip \
-O /tmp/rps.zip

lwget --no-check-certificate \
https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps-test-set.zip \
-O /tmp/rps-test-set.zip

[2019-02-13 22:46:40] -- https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps.zip
Resolving storage.googleapis.com... 2607:f8b0:4003:c0a::80, 64.233.171.128
Connecting to storage.googleapis.com|2607:f8b0:4003:c0a::80|:443... connected.
WARNING: cannot verify storage.googleapis.com's certificate, issued by 'CN=Google Internet Authority G3,O=Google Trust Services,C=US':
Unable to locally verify the issuer's authority.
HTTP request sent, awaiting response... 200 OK
Length: 200682221 (191M) [application/zip]
Saving to: '/tmp/rps.zip'

/tmp/rps.zip 100%[=====] 191.38M 168MB/s in 1.1s

2019-02-13 22:46:41 (168 MB/s) - '/tmp/rps.zip' saved [200682221/200682221]

[2019-02-13 22:46:42] -- https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps-test-set.zip
Resolving storage.googleapis.com... 2607:f8b0:4003:c11::80, 74.125.127.128
Connecting to storage.googleapis.com|2607:f8b0:4003:c11::80|:443... connected.
WARNING: cannot verify storage.googleapis.com's certificate, issued by 'CN=Google Internet Authority G3,O=Google Trust Services,C=US':
Unable to locally verify the issuer's authority.
HTTP request sent, awaiting response... 200 OK
Length: 29516758 (28M) [application/zip]
Saving to: '/tmp/rps-test-set.zip'

/tmp/rps-test-set.z 100%[=====] 28.15M 128MB/s in 0.2s

2019-02-13 22:46:43 (128 MB/s) - '/tmp/rps-test-set.zip' saved [29516758/29516758]

```

Train a classifier with Rock Paper Scissors

!wget 명령을 사용하여
파일 다운로드

```
!wget --no-check-certificate \
https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps.zip \
-O /tmp/rps.zip

!wget --no-check-certificate \
https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps-test-set.zip \
-O /tmp/rps-test-set.zip

--2019-02-13 22:46:40-- https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps.zip
Resolving storage.googleapis.com... 2607:f8b0:4003:c0a::80, 64.233.171.128
Connecting to storage.googleapis.com|2607:f8b0:4003:c0a::80|:443... connected.
WARNING: cannot verify storage.googleapis.com's certificate, issued by 'CN=Google Internet Authority G3,O=Google Trust Services,C=US':
Unable to locally verify the issuer's authority.
HTTP request sent, awaiting response... 200 OK
Length: 200682221 (191M) [application/zip]
Saving to: '/tmp/rps.zip'

/tmp/rps.zip      100%[=====>] 191.38M  168MB/s  in 1.1s

2019-02-13 22:46:41 (168 MB/s) - '/tmp/rps.zip' saved [200682221/200682221]

--2019-02-13 22:46:42-- https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps-test-set.zip
Resolving storage.googleapis.com... 2607:f8b0:4003:c11::80, 74.125.127.128
Connecting to storage.googleapis.com|2607:f8b0:4003:c11::80|:443... connected.
WARNING: cannot verify storage.googleapis.com's certificate, issued by 'CN=Google Internet Authority G3,O=Google Trust Services,C=US':
Unable to locally verify the issuer's authority.
HTTP request sent, awaiting response... 200 OK
Length: 29516758 (28M) [application/zip]
Saving to: '/tmp/rps-test-set.zip'

/tmp/rps-test-set.z 100%[=====>] 28.15M  128MB/s  in 0.2s

2019-02-13 22:46:43 (128 MB/s) - '/tmp/rps-test-set.zip' saved [29516758/29516758]
```

Train a classifier with Rock Paper Scissors

```
import os
import zipfile

local_zip = '/tmp/rps.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/tmp/')
zip_ref.close()

local_zip = '/tmp/rps-test-set.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/tmp/')
zip_ref.close()
```

Train a classifier with Rock Paper Scissors

```
rock_dir = os.path.join('/tmp/rps/rock')
paper_dir = os.path.join('/tmp/rps/paper')
scissors_dir = os.path.join('/tmp/rps/scissors')

print('total training rock images:', len(os.listdir(rock_dir)))
print('total training paper images:', len(os.listdir(paper_dir)))
print('total training scissors images:', len(os.listdir(scissors_dir)))

rock_files = os.listdir(rock_dir)
print(rock_files[:10])

paper_files = os.listdir(paper_dir)
print(paper_files[:10])

scissors_files = os.listdir(scissors_dir)
print(scissors_files[:10])
```

```
[> ('total training rock images:', 840)
('total training paper images:', 840)
('total training scissors images:', 840)
['rock06ck02-084.png', 'rock01-024.png', 'rock06ck02-069.png', 'rock03-086.png', 'rock06ck02-033.png', 'rock01-058.png', 'rock03-036.png', 'rock01-086.png', 'rock07-017.png', 'rock06ck02-084.png']
['paper01-079.png', 'paper03-059.png', 'paper04-108.png', 'paper02-048.png', 'paper02-007.png', 'paper04-022.png', 'paper01-103.png', 'paper07-043.png', 'paper03-017.png', 'paper01-079.png']
['scissors04-003.png', 'testscissors03-082.png', 'scissors03-102.png', 'scissors02-004.png', 'testscissors02-080.png', 'scissors01-081.png', 'scissors02-053.png', 'scissors04-003.png']
```

Train a classifier with Rock Paper Scissors

```
%matplotlib inline

import matplotlib.pyplot as plt
import matplotlib.image as mpimg

pic_index = 2

next_rock = [os.path.join(rock_dir, fname)
             for fname in rock_files[pic_index-2:pic_index]]
next_paper = [os.path.join(paper_dir, fname)
              for fname in paper_files[pic_index-2:pic_index]]
next_scissors = [os.path.join(scissors_dir, fname)
                 for fname in scissors_files[pic_index-2:pic_index]]

for i, img_path in enumerate(next_rock+next_paper+next_scissors):
    #print(img_path)
    img = mpimg.imread(img_path)
    plt.imshow(img)
    plt.axis('Off')
    plt.show()
```



Train a classifier with Rock Paper Scissors

```
[16] import tensorflow as tf
import keras_preprocessing
from keras_preprocessing import image
from keras_preprocessing.image import ImageDataGenerator

TRAINING_DIR = "/tmp/rps/"
training_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

VALIDATION_DIR = "/tmp/rps-test-set/"
validation_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = training_datagen.flow_from_directory(
    TRAINING_DIR,
    target_size=(150,150),
    class_mode='categorical'
)

validation_generator = validation_datagen.flow_from_directory(
    VALIDATION_DIR,
    target_size=(150,150),
    class_mode='categorical'
)

model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 150x150 with 3 bytes color
    # This is the first convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu')
])
```


Train a classifier with Rock Paper Scissors

```
[16] import tensorflow as tf
import keras_preprocessing
from keras_preprocessing import image
from keras_preprocessing.image import ImageDataGenerator

TRAINING_DIR = "/tmp/rps/"
training_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

VALIDATION_DIR = "/tmp/rps-test-set/"
validation_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = training_datagen.flow_from_directory(
    TRAINING_DIR,
    target_size=(150,150),
    class_mode='categorical'
)

validation_generator = validation_datagen.flow_from_directory(
    VALIDATION_DIR,
    target_size=(150,150),
    class_mode='categorical'
)

model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 150x150 with 3 bytes color
    # This is the first convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu')
])
```

} 학습 데이터 정의

I

Train a classifier with Rock Paper Scissors

```
[16] import tensorflow as tf
import keras_preprocessing
from keras_preprocessing import image
from keras_preprocessing.image import ImageDataGenerator

TRAINING_DIR = "/tmp/rps/"
training_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

VALIDATION_DIR = "/tmp/rps-test-set/"
validation_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = training_datagen.flow_from_directory(
    TRAINING_DIR,
    target_size=(150,150),
    class_mode='categorical'
)

validation_generator = validation_datagen.flow_from_directory(
    VALIDATION_DIR,
    target_size=(150,150),
    class_mode='categorical'
)

model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 150
    # This is the first convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])
```

```
35 from keras_preprocessing.image import ImageDataGenerator
36
37 TRAINING_DIR = "/tmp/rps/"
38
39 training_datagen = ImageDataGenerator(
40     rescale=1./255,
41     rotation_range=40,
42     width_shift_range=0.2,
43     height_shift_range=0.2,
44     shear_range=0.2,
45     zoom_range=0.2,
46     horizontal_flip=True,
47     fill_mode='nearest'
48 )
```

Train a classifier with Rock Paper Scissors

```
[16] import tensorflow as tf
import keras_preprocessing
from keras_preprocessing import image
from keras_preprocessing.image import ImageDataGenerator

TRAINING_DIR = "/tmp/rps/"
training_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,

model.summary()

model.compile(loss = 'categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

history = model.fit_generator(train_generator, epochs=25, validation_data = validation_generator, verbose = 1)

model.save("rps.h5")

VALIDATION_DIR,
target_size=(150,150),
class_mode='categorical'
)

model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 150
    # This is the first convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(150,150,3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])

validation_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Train a classifier with Rock Paper Scissors

```
[16] import tensorflow as tf
import keras_preprocessing
from keras_preprocessing import image
from keras_preprocessing.image import ImageDataGenerator
```

```
TRAINING_DIR = "/tmp/rps/"
training_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
```

```
35 from keras_preprocessing.image import ImageDataGenerator
36
37 TRAINING_DIR = "/tmp/rps/"
```

```
model.summary()
```

```
model.compile(loss = 'categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
```

```
history = model.fit_generator(train_generator, epochs=25, validation_data = validation_generator, verbose = 1)
```

```
model.save("rps.h5")
```

모델을 h5 파일로 저장

```
VALIDATION_DIR,
target_size=(150,150),
class_mode='categorical'
)
```

```
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 150
    # This is the first convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(150,150,3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])
```

```
44 snear_range=0.2,
45 zoom_range=0.2,
46 horizontal_flip=True,
47 fill_mode='nearest'
48 )
```

Train a classifier with Rock Paper Scissors

```
79/79=====] - 19s 240ms/step - loss: 0.1005 - acc: 0.9667 - val_loss: 0.0431 - val_acc: 0.9812
Epoch 19/25
79/79=====] - 19s 244ms/step - loss: 0.0983 - acc: 0.9698 - val_loss: 0.0627 - val_acc: 0.9785
Epoch 20/25
79/79=====] - 20s 249ms/step - loss: 0.0830 - acc: 0.9738 - val_loss: 0.3355 - val_acc: 0.8575
Epoch 21/25
79/79=====] - 19s 245ms/step - loss: 0.1141 - acc: 0.9647 - val_loss: 0.0584 - val_acc: 0.9731
Epoch 22/25
79/79=====] - 19s 243ms/step - loss: 0.0803 - acc: 0.9750 - val_loss: 0.0380 - val_acc: 0.9812
Epoch 23/25
79/79=====] - 20s 253ms/step - loss: 0.0762 - acc: 0.9754 - val_loss: 0.3842 - val_acc: 0.9167
Epoch 24/25
79/79=====] - 19s 246ms/step - loss: 0.0781 - acc: 0.9758 - val_loss: 0.0176 - val_acc: 0.9892
Epoch 25/25
79/79=====] - 19s 237ms/step - loss: 0.0708 - acc: 0.9810 - val_loss: 0.1145 - val_acc: 0.9543
```

Train a classifier with Rock Paper Scissors

```
79/79=====] - 19s 240ms/step - loss: 0.1005 - acc: 0.9667 - val_loss: 0.0431 - val_acc: 0.9812
Epoch 19/25
79/79=====] - 19s 244ms/step - loss: 0.0983 - acc: 0.9698 - val_loss: 0.0627 - val_acc: 0.9785
Epoch 20/25
79/79=====] - 20s 249ms/step - loss: 0.0830 - acc: 0.9738 - val_loss: 0.3355 - val_acc: 0.8575
Epoch 21/25
79/79=====] - 19s 245ms/step - loss: 0.1141 - acc: 0.9647 - val_loss: 0.0584 - val_acc: 0.9731
Epoch 22/25
79/79=====] - 19s 243ms/step - loss: 0.0803 - acc: 0.9750 - val_loss: 0.0380 - val_acc: 0.9812
Epoch 23/25
79/79=====] - 20s 253ms/step - loss: 0.0762 - acc: 0.9754 - val_loss: 0.3842 - val_acc: 0.9167
Epoch 24/25
79/79=====] - 19s 246ms/step - loss: 0.0781 - acc: 0.9758 - val_loss: 0.0176 - val_acc: 0.9892
Epoch 25/25
79/79=====] - 19s 237ms/step - loss: 0.0708 - acc: 0.9810 - val_loss: 0.1145 - val_acc: 0.9543
```

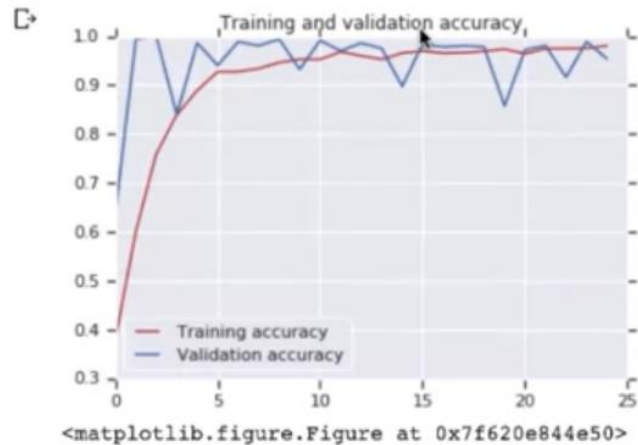
Train a classifier with Rock Paper Scissors

```
import matplotlib.pyplot as plt
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

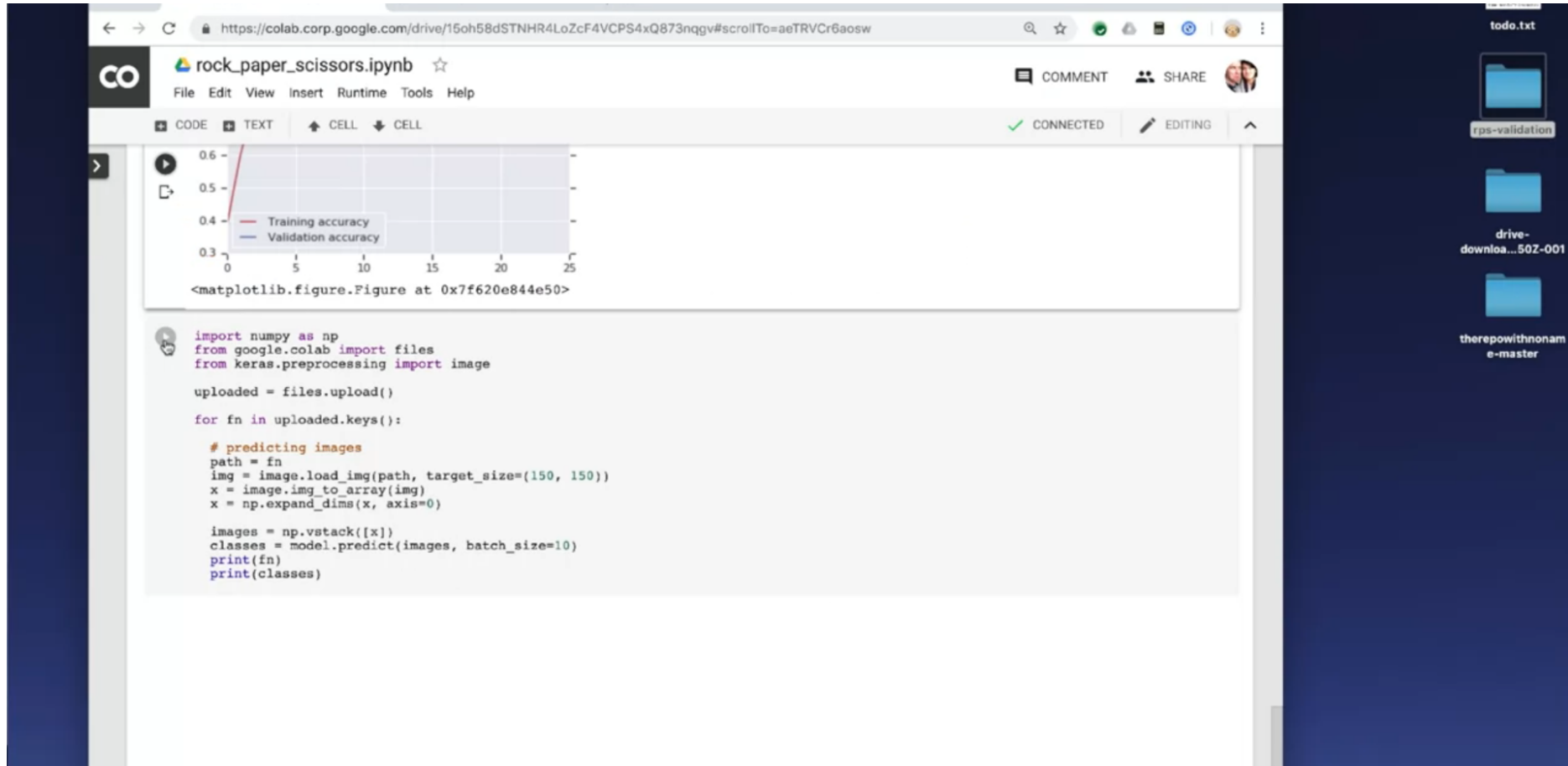
epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc=0)
plt.figure()

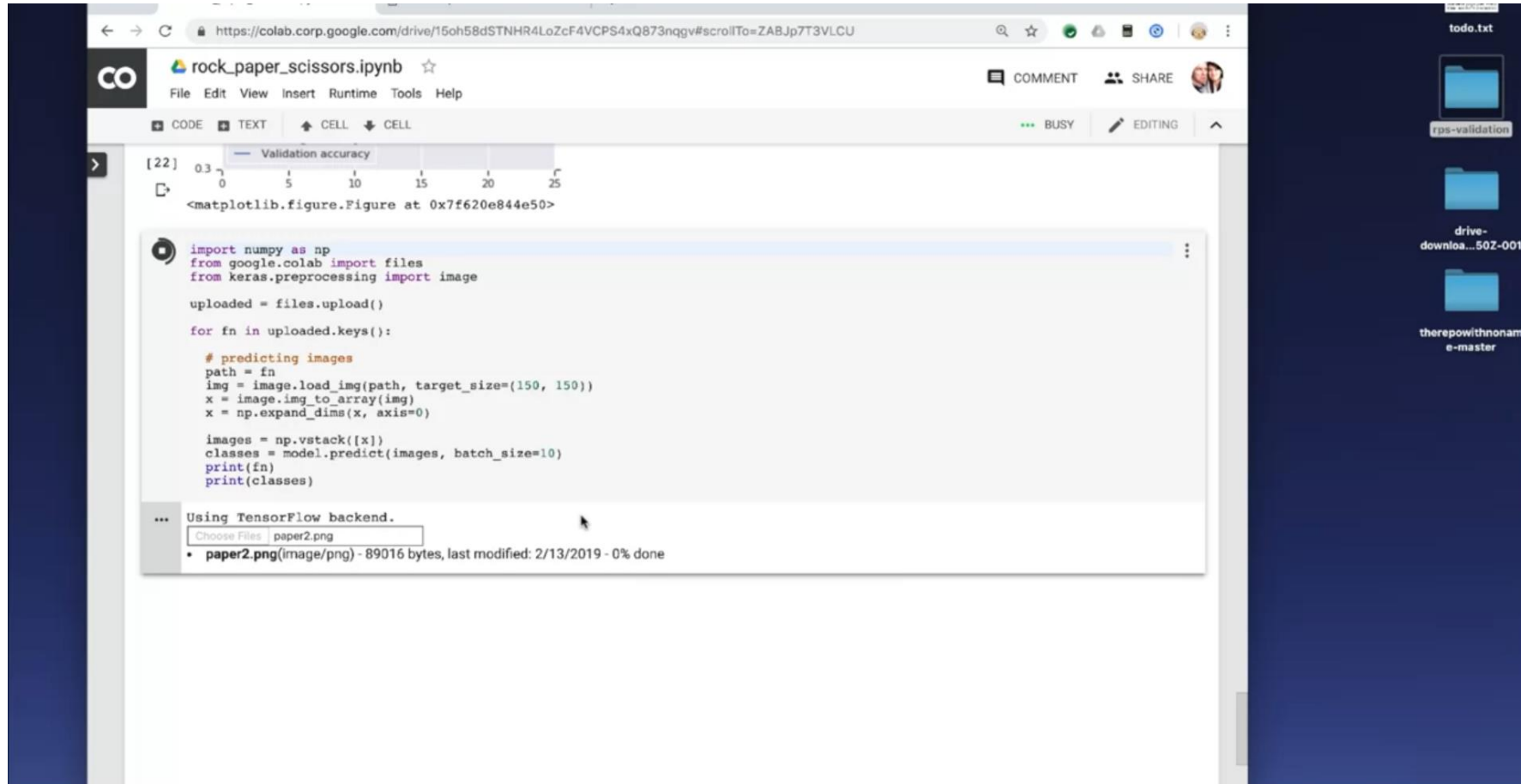
plt.show()
```



Test the Rock Paper Scissors classifier



Test the Rock Paper Scissors classifier



The screenshot displays a Google Colab notebook interface. The browser address bar shows the URL: <https://colab.corp.google.com/drive/15oh58dSTNHR4LoZcF4VCPS4xQ873nqgv#scrollTo=ZABJp7T3VLCU>. The notebook title is "rock_paper_scissors.ipynb". The interface includes tabs for "CODE", "TEXT", "CELL", and "CELL", along with "COMMENT" and "SHARE" buttons. A plot titled "Validation accuracy" is shown, with the x-axis ranging from 0 to 25 and the y-axis from 0.3 to 1.0. Below the plot, a code cell is visible, containing the following Python code:

```
[22] import numpy as np
from google.colab import files
from keras.preprocessing import image

uploaded = files.upload()

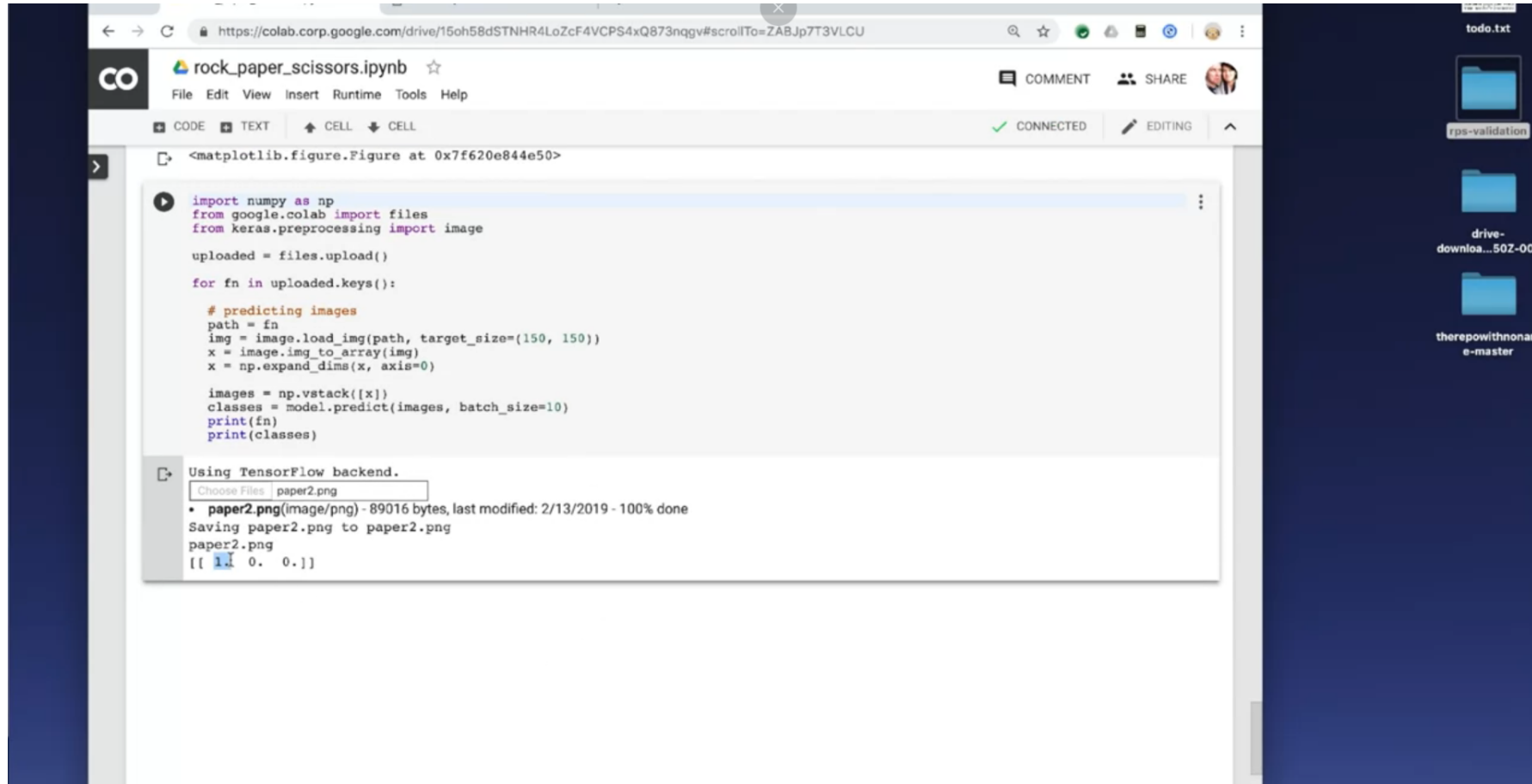
for fn in uploaded.keys():

    # predicting images
    path = fn
    img = image.load_img(path, target_size=(150, 150))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

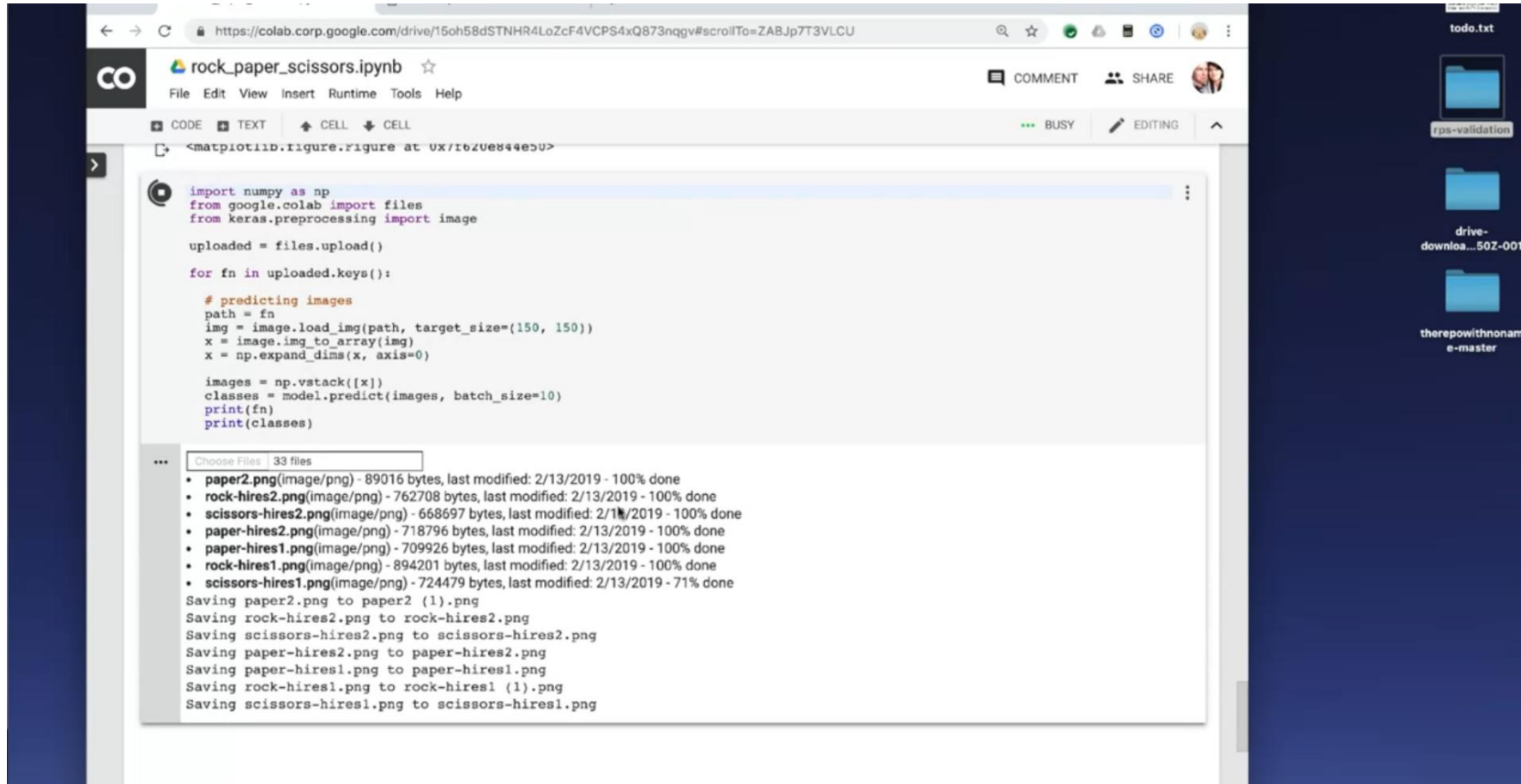
    images = np.vstack([x])
    classes = model.predict(images, batch_size=10)
    print(fn)
    print(classes)
```

Below the code cell, a message indicates "Using TensorFlow backend." and a file upload section shows "paper2.png" (89016 bytes, last modified: 2/13/2019 - 0% done).

Test the Rock Paper Scissors classifier



Test the Rock Paper Scissors classifier



The screenshot shows a Google Colab notebook titled "rock_paper_scissors.ipynb". The code in the notebook is as follows:

```
import numpy as np
from google.colab import files
from keras.preprocessing import image

uploaded = files.upload()

for fn in uploaded.keys():

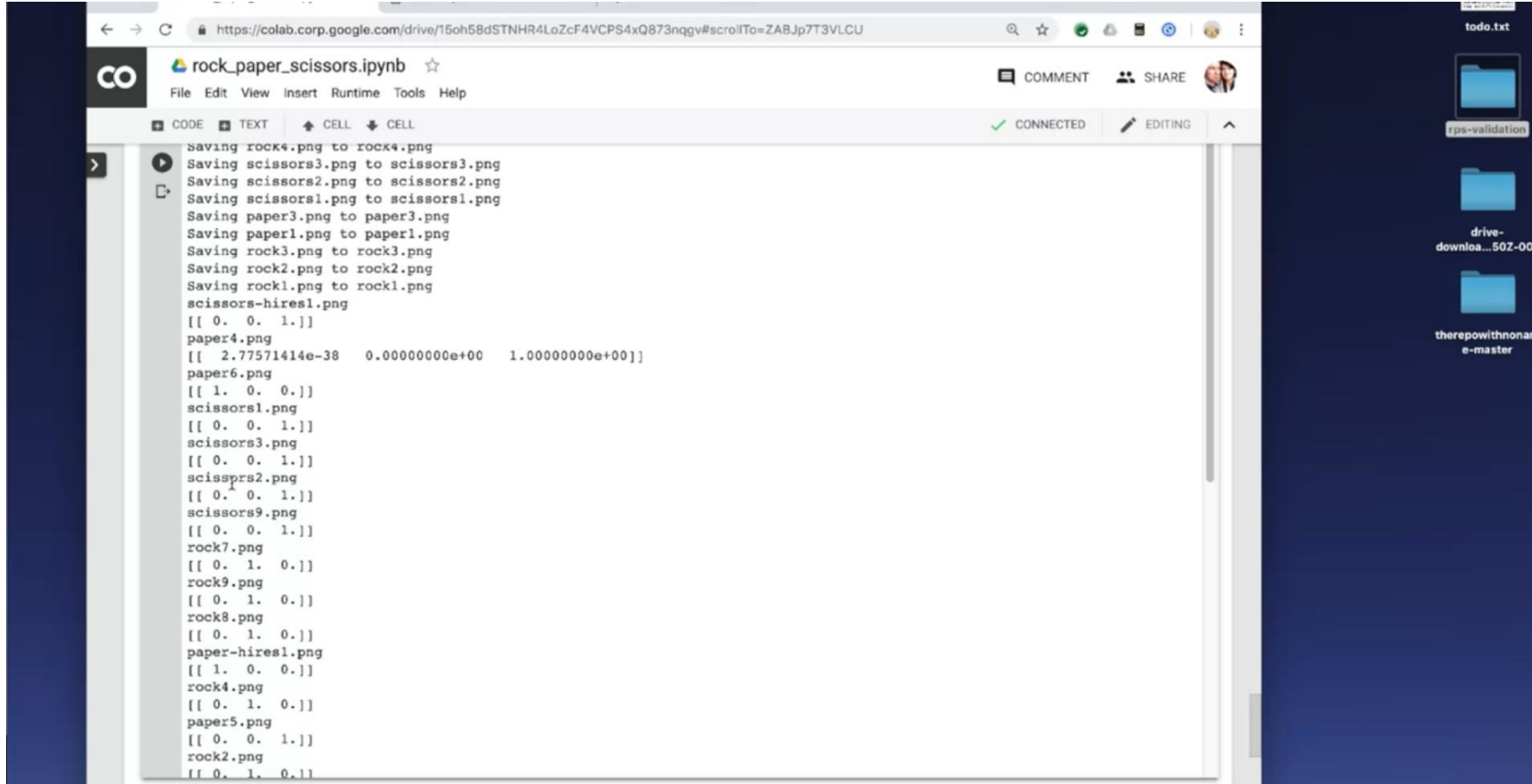
    # predicting images
    path = fn
    img = image.load_img(path, target_size=(150, 150))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    images = np.vstack([x])
    classes = model.predict(images, batch_size=10)
    print(fn)
    print(classes)
```

The output of the code shows a list of files and their predicted classes:

```
... Choose Files 33 files
• paper2.png(image/png) - 89016 bytes, last modified: 2/13/2019 - 100% done
• rock-hires2.png(image/png) - 762708 bytes, last modified: 2/13/2019 - 100% done
• scissors-hires2.png(image/png) - 668697 bytes, last modified: 2/13/2019 - 100% done
• paper-hires2.png(image/png) - 718796 bytes, last modified: 2/13/2019 - 100% done
• paper-hires1.png(image/png) - 709926 bytes, last modified: 2/13/2019 - 100% done
• rock-hires1.png(image/png) - 894201 bytes, last modified: 2/13/2019 - 100% done
• scissors-hires1.png(image/png) - 724479 bytes, last modified: 2/13/2019 - 71% done
Saving paper2.png to paper2 (1).png
Saving rock-hires2.png to rock-hires2.png
Saving scissors-hires2.png to scissors-hires2.png
Saving paper-hires2.png to paper-hires2.png
Saving paper-hires1.png to paper-hires1.png
Saving rock-hires1.png to rock-hires1 (1).png
Saving scissors-hires1.png to scissors-hires1.png
```

Test the Rock Paper Scissors classifier



The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL: <https://colab.corp.google.com/drive/15oh58dSTNHR4LoZcF4VCPS4xQ873nqgv#scrollTo=ZABJp7T3VLCU>. The notebook title is "rock_paper_scissors.ipynb". The interface includes tabs for "CODE", "TEXT", "CELL", and "CELL", along with "COMMENT" and "SHARE" buttons. The "CODE" tab is active, showing a list of file saving operations and classification results for various images. The operations include saving rock, scissors, and paper images to specific files. The classification results are shown as lists of values, such as `[[0. 0. 1.]]` for "paper4.png" and `[[2.77571414e-38 0.00000000e+00 1.00000000e+00]]` for "paper6.png".

```
saving rock4.png to rock4.png
Saving scissors3.png to scissors3.png
Saving scissors2.png to scissors2.png
Saving scissors1.png to scissors1.png
Saving paper3.png to paper3.png
Saving paper1.png to paper1.png
Saving rock3.png to rock3.png
Saving rock2.png to rock2.png
Saving rock1.png to rock1.png
scissors-hires1.png
[[ 0. 0. 1.]]
paper4.png
[[ 2.77571414e-38 0.00000000e+00 1.00000000e+00]]
paper6.png
[[ 1. 0. 0.]]
scissors1.png
[[ 0. 0. 1.]]
scissors3.png
[[ 0. 0. 1.]]
scissors2.png
[[ 0. 0. 1.]]
scissors9.png
[[ 0. 0. 1.]]
rock7.png
[[ 0. 1. 0.]]
rock9.png
[[ 0. 1. 0.]]
rock8.png
[[ 0. 1. 0.]]
paper-hires1.png
[[ 1. 0. 0.]]
rock4.png
[[ 0. 1. 0.]]
paper5.png
[[ 0. 0. 1.]]
rock2.png
[[ 0. 1. 0.]]
```