Code wins arguments

Menu

Face Recognition with Facebook DeepFace in Keras

February 17, 2020 / Machine Learning

Support me on Patreon

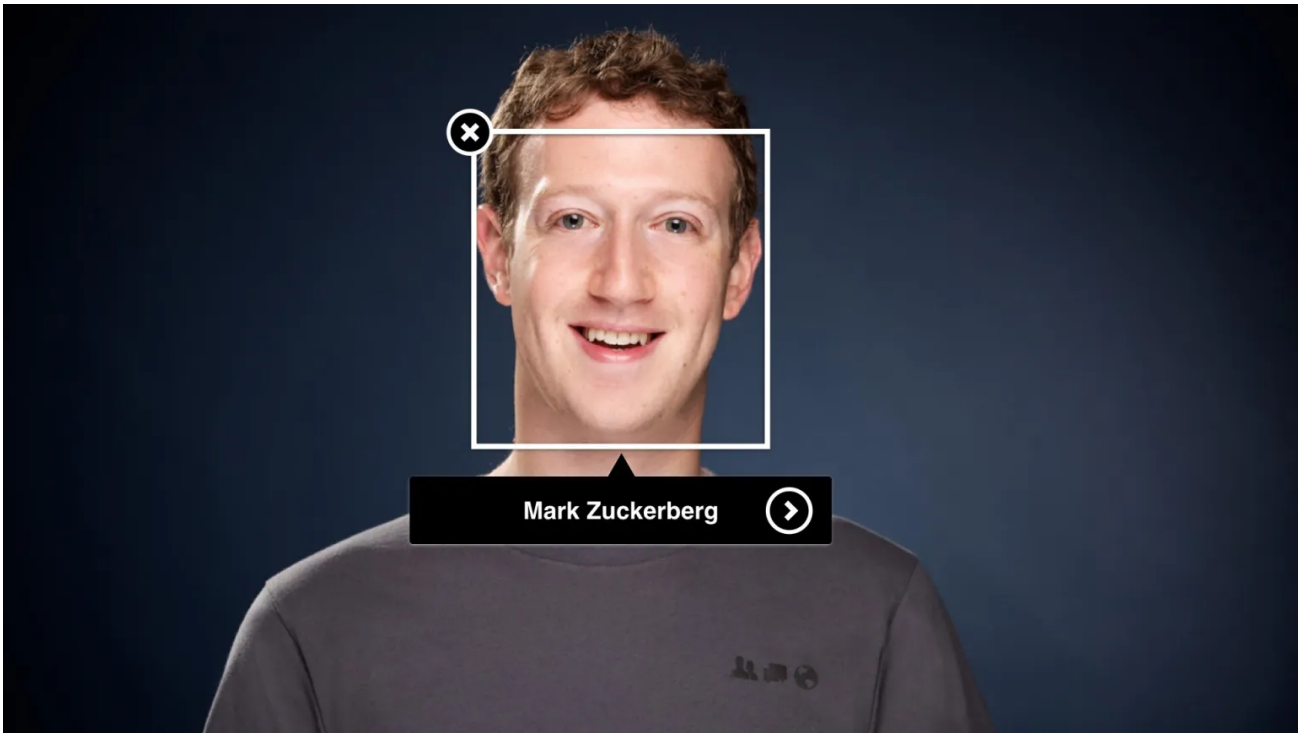
BECOME A PATRON

Haven't you subscribe my channel yet?

Follow me on Twitter

Follow @serengil

Facebook researchers announced its face recognition model DeepFace. It shows a very close performance to human level. Humans have 97.53% score whereas DeepFace model has 97.35% ± 0.25%. This means that the model can get higher score than human beings sometimes. We will mention DeepFace model within Keras for Python in this post.



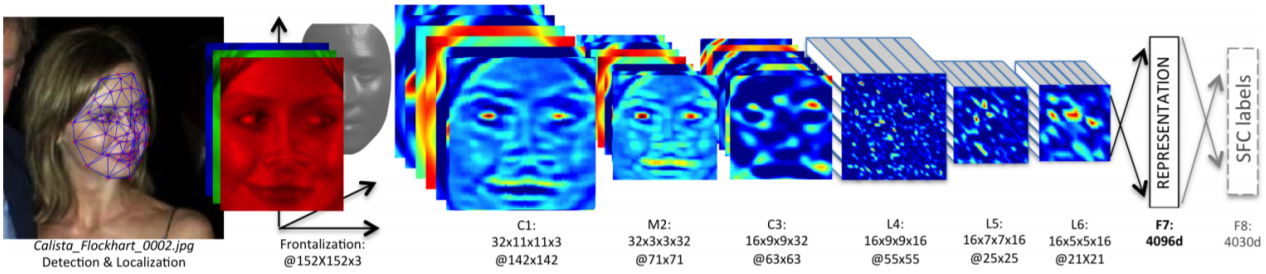
Mark Zuckerberg

Objective

Face recognition is a combination of CNN, Autoencoders and Transfer Learning studies. I strongly recommend you to read How Face Recognition Works post to understand what a face recognition pipeline exactly is.

DeepFace structure

DeepFace model is a 8 layered convolutional neural networks. Each layer is named with a letter and number as seen. Here, number refers to the index from 1 to 8 and letter states the type of layer. C refers to convolutional layer, M refers to max pooling, L refers to **locally connected layer** and F refers to fully connected layer.



DeepFace model

Even though it has a limited number of layers, its parameter size is huge. It is 6 times wider than Facenet and 36 times wider than OpenFace whereas complexity of VGG-Face and DeepFace is close. Imagine that how hard to train the network from scratch.

Model	**Num of Params**
VGG-Face	145,002,878
Facenet	22,808,144
OpenFace	3,743,280
DeepFace	137,774,071

Training the network from scratch

Swarup Ghosh trained the DeepFace model from scratch for Keras.

The original study trained the model for SFC Dataset. The data set has 4.4 millions of photos of 4030 people. That is exactly the number of nodes in the final layer – F8 in the illustration.

On the other hand, Swarup trained the same model for VGGFace2 dataset. This data set has 3.3 millions photos of 8631 people. That's why, the number of nodes in the final layer will be 8631. This is different than the original study.

Notice that the final layer F8 is drawed with dashed lines in the illustration. This is because the final layer F8 will be dropped in the face recognition task and the outputs of the early layer F7 will be used.

DeepFace model

DeepFace model can be built by the sequential API of Keras.

```
model = Sequential()
model.add(Convolution2D(32, (11, 11), activation='relu', name='C1', in
model.add(MaxPooling2D(pool_size=3, strides=2, padding='same', name='M
model.add(Convolution2D(16, (9, 9), activation='relu', name='C3'))
model.add(LocallyConnected2D(16, (9, 9), activation='relu', name='L4')
model.add(LocallyConnected2D(16, (7, 7), strides=2, activation='relu',
model.add(LocallyConnected2D(16, (5, 5), activation='relu', name='L6')
model.add(Flatten(name='F0'))
model.add(Dense(4096, activation='relu', name='F7'))
model.add(Dropout(rate=0.5, name='D0'))
model.add(Dense(8631, activation='softmax', name='F8'))
```

Loading pre-trained weights

Swarup shared the pre-trained weights of DeepFace model for Keras.

```
#Can be downloaded from https://github.com/swghosh/DeepFace/releases
model.load_weights("VGGFace2_DeepFace_weights_val-0.9034.h5")
```

Representation layer

As I mentioned before, F7 layer refers to the representation layer. So, we no longer need to include the final layer F8. We can drop it right now.

```
deepface_model = Model(inputs=model.layers[0].input, outputs=model.laye
```

So, DeepFace model expects 152×152 sized facial image as input and represent is as 4096 dimensional vector.

Comparing representations

The output of the deepface is 4096 dimensional vector (embedding). Euclidean distance in L2 form is used to find the similarity between two vectors in the original paper.

```
def l2_normalize(x):
return x / np.sqrt(np.sum(np.multiply(x, x)))

def findEuclideanDistance(source_representation, test_representation):
euclidean_distance = source_representation - test_representation
euclidean_distance = np.sum(np.multiply(euclidean_distance, euclidean_d
euclidean_distance = np.sqrt(euclidean_distance)
return euclidean_distance
```

We will represent two face images as vectors firstly. Representation means calling predict function for programmers actually. So, we will make predictions for two facial images and then find the euclidean distance between these two outputs.

```
img1_embedding = deepface_model.predict("img1.jpg")[0]
img2_embedding = deepface_model.predict("img2.jpg")[0]

euclidean_l2_distance = findEuclideanDistance(l2_normalize(img1_embeddi
, l2_normalize(img2_embedding))
```

The expectation is that the distance between two face images should be close for same person whereas it should be distant for different ones. My experiments show that the threshold value should be 0.55.

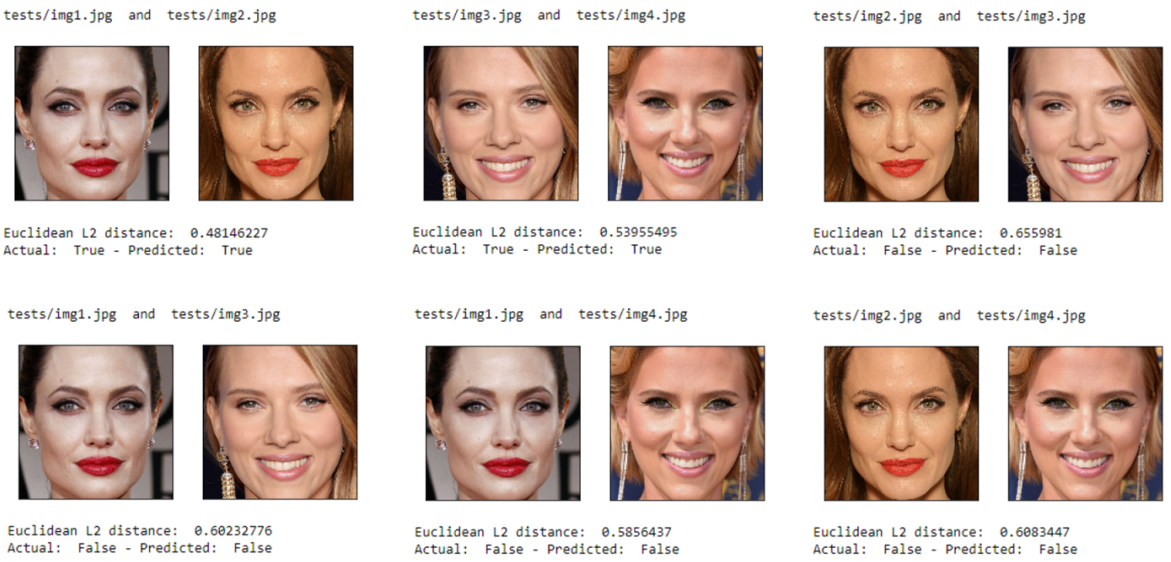
If you wonder how to determine the threshold value for this face recognition model, then this blog post explains it deeply.

```
if euclidean_l2_distance &amp;amp;amp;amp;amp;amp;amp;lt;= 0.55:
verified = True
```

```
    else:
    verified = False
```

Unit tests

I've tested the DeepFace model for a set of Angelina Jolie and Scarlett Johansson. The following 6 experiments got absolute accuracy. To be honest, it satisfies me a lot.



Testing DeepFace

Real time implementation

You can run Facebook DeepFace model in real time as well. You can find the source code for DeepFace in real time here. Results seems very fast and very satisfactory in the following video.

This is a sample study of transfer learning and autoencoders.

Early stages of face recognition pipeline

Modern face recognition pipelines consist of 4 stages: detect, align, represent and classify / verify. We've skipped the face detection and face alignment steps not to make this post so complex. However, it is really important for face recognition tasks.

Face detection can be done with many solutions such as OpenCV, Dlib or MTCNN. OpenCV offers haar cascade, single shot multibox detector (SSD). Dlib offers Histogram of Oriented Gradients (HOG) and Max-Margin Object Detection (MMOD). Finally, MTCNN is a popular solution in the open source community as well. Herein, SSD, MMOD and MTCNN are modern deep learning based approaches whereas haar cascade and HoG are legacy methods. Besides, SSD is the fastest one. You can monitor the detection performance of those methods in the following video.

Here, you can watch how to use different face detectors in Python.

For instance, Google declared that face alignment increases its face recognition model FaceNet from 98.87% to 99.63%. This is almost 1% accuracy improvement which means a lot for engineering studies. Here, you can find a detailed tutorial for face alignment in Python within OpenCV.

Face alignment

You can find out the math behind alignment more on the following video:

Besides, face detectors detect faces in a rectangle area. So, detected faces come with some noise such as background color. We can find 68 different landmarks of a face with dlib. In this way, we can get rid of any noise of a facial image.

In addition, MediaPipe can find 468 landmarks. Please see its real time implementation in the following video. Recommended tutorials: Deep Face Detection with MediaPipe, Zoom Style Virtual Background Setup with MediaPipe.

Here, retinaface is the cutting-edge face detection technology. It can even detect faces in the crowd and it finds facial landmarks including eye coordinates. That's why, its alignment score is very high.

Conclusion

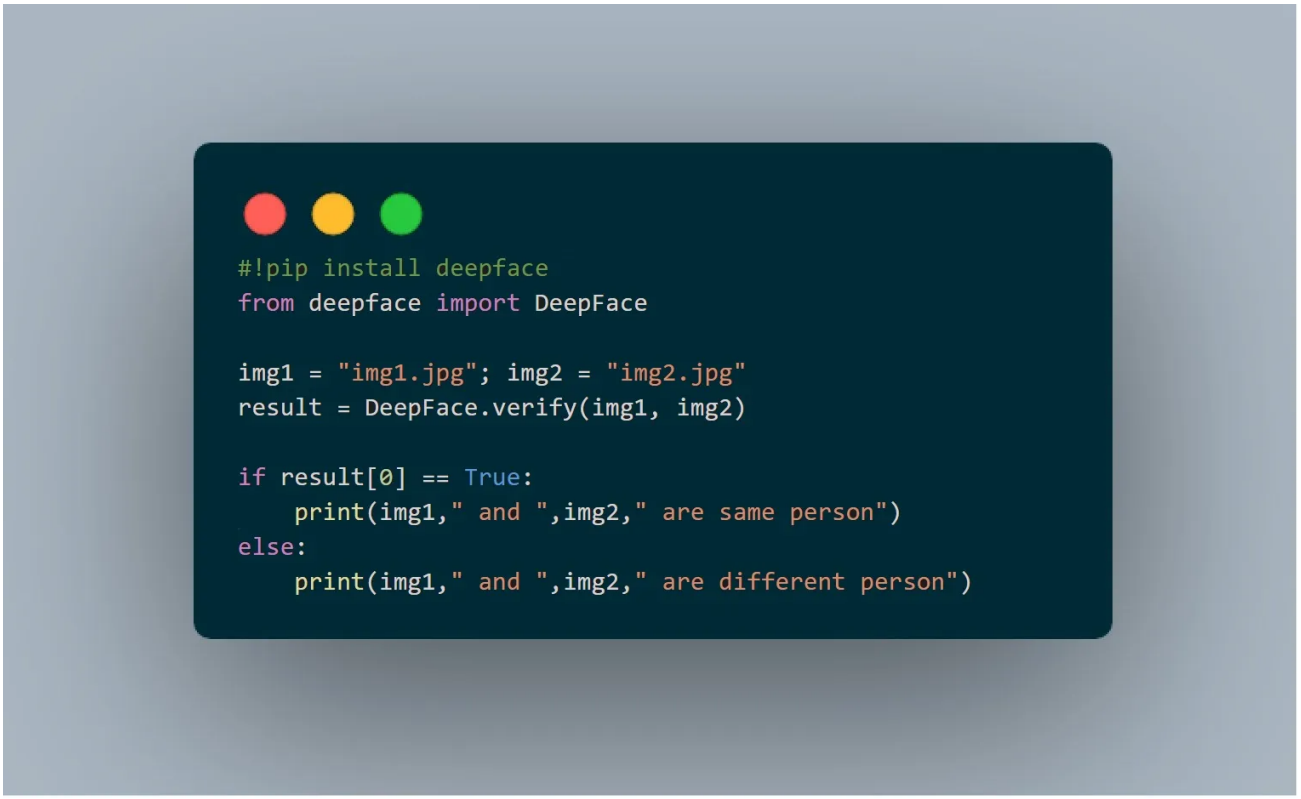
So, we've mentioned the Facebook's DeepFace face recognition model in this post. It comes with a human-level face recognition performance based on the declaration of Facebook researchers. It has a huge, wide and very complex architecture. I am very grateful to Swarup Ghosh because of his contributions to open source.

I've pushed the source code of this study as a notebook to GitHub. You can support this study by starring⭐ the repo.

Python library

You might not want to spend time to build FB DeepFace model from scratch, load pre-trained weights, find vector embeddings of faces and find distances between embeddings. Herein, deepface is a lightweight face recognition framework for Python. It currently supports the most common face recognition models including VGG-Face, Google Facenet, OpenFace, Facebook DeepFace and DeepID.

It handles all of those steps in the background. You can verify faces with a just few lines of code. It is available on PyPI as well. You should run the command "**pip install deepface**" to have it. Its code is also open-sourced in GitHub. Repo has a detailed documentation for developers. BTW, you can support this project by starring the repo.



deepface

Here, you can watch the video of deepface framework.

Besides, you can run deepface in real time with your webcam as well.

Large scale face recognition

Large scale face recognition requires to apply face verification several times. However, we can store the representations of ones in our database. In this way, we just need to find the representation of target image. Finding distances between representations can be handled very fast. So, we can find an identity in a large scale data set in just seconds. Deepface offers an out-of-the-box function to handle large scale face recognition as well.

Notice that face recognition has O(n) time complexity and this might be problematic for millions or billions level data. Herein, approximate nearest neighbor (a-nn) algorithm

reduces time complexity dramatically. Spotify Annoy, Facebook Faiss and NMSLIB are amazing a-nn libraries. Besides, Elasticsearch wraps an NMSLIB and it comes with highly scalability. You should run deepface within those a-nn libraries if you have really large scale data base.

On the other hand, a-nn algorithm does not guarantee to find the closest one always. We can still apply k-nn algorithm here. Map reduce technology of big data systems might satisfy the both speed and confidence here. mongoDB, Cassandra and Hadoop are the most popular solutions for no-sql databases. Besides, if you have a powerful database such as Oracle Exadata, then RDBMS and regular sql might satisfy your concerns as well.

Ensemble method

We've mentioned just a single face recognition model. On the other hand, there are several state-of-the-art models: VGG-Face, Google FaceNet, OpenFace, Facebook DeepFace and DeepID. Even though all of those models perform well, there is no absolute better model. Still, we can apply an ensemble method to build a grandmaster model. In this approach, we will feed the predictions of those models to a boosting model. Accuracy metrics including precision, recall and f1 score increase dramatically in ensemble method whereas running time lasts longer.

Tech Stack Recommendations

Face recognition is mainly based on representing facial images as vectors. Herein, storing the vector representations is a key factor for building robust facial recognition systems. I summarize the tech stack recommendations in the following video.

The Best Single Model

There are a few state-of-the art face recognition models: VGG-Face, FaceNet, OpenFace and DeepFace. Some are designed by tech giant companies such as Google and Facebook whereas some are designed by the top universities in the world such as University of

Oxford and Carnegie Mellon University. We will have a discussion about the best single face recognition model in this video.

Like this blog? Support me on Patreon

BECOME A PATRON

#face recognition, #face verification

« *Previous* / *Next* »

23 Comments

Amundeep Singh
February 26, 2020 at 6:30 pm

Dear Serengil,

Why you have not stored the images in folders? this would allow you to store for e.g. 10 images of 1 person under a folder name. This should increase the accuracy?

Reply

Sefik Serengil

February 28, 2020 at 8:21 am

You can but this is one shot learning. If one picture is enough, why would you need many pictures.

Reply

Yasser

March 4, 2020 at 9:13 pm

HI,

What is the resource that you get the facenet's parameters is equal to 22,808,144?

because some website said that facenet's parameters equal to 140M parameters

Reply

Sefik Serengil

March 5, 2020 at 5:43 am

I build the model scratch and call model.summary() in keras. You should follow the related link of facenet.

Reply

Yasser

March 16, 2020 at 11:12 am

Hi…
I read the deepface paper. It said that (the DeepFace runs at 0.33 seconds per image, accounting for image decoding, face detection and alignment, the feedforward network, and the final classification output)
– Using a single-core Intel 2.2GHz CPU.

-When I use the model here in the GPU of 11700 images (450 persons), it takes 24 seconds to predict.

Reply

Sefik Serengil

March 16, 2020 at 11:19 am

They might build the model on core c or c++ . This is a keras implementation. Besides, we do not know the hardware they have. We should not compare

performances.

Reply

Yasser

March 16, 2020 at 11:17 am

Hi…
I read the deepface paper. It said that (the DeepFace runs at 0.33 seconds per image, accounting for image decoding, face detection and alignment, the feedforward network, and the final classification output)
– Using a single-core Intel 2.2GHz CPU.

-When I use the model here in the GPU of 11700 images (450 persons), it takes 24 seconds to predict.

Reply

Yasser

March 16, 2020 at 11:19 am

Hi…
I read the deepface paper. It said that (the DeepFace runs at 0.33 seconds per image, accounting for image decoding, face detection and alignment, the feedforward network, and the final classification output)
– Using a single-core Intel 2.2GHz CPU.

-When I use the model here in the GPU of 11700 images (450 persons), it takes 24 seconds to predict.

Reply

Ammar

April 18, 2020 at 1:45 pm

Hi
How can I calculate the accuracy using accuracy_score function in your implementation?

Reply

Sefik Serengil

April 18, 2020 at 2:47 pm

You can get help from this program:
https://github.com/serengil/deepface/blob/master/tests/unit_tests.py

I basically call predict function and embeddings, find distance between embeddings, assume they are same if distance is less than the threshold. Finally, assumptions and real values could be stored in a data frame and you can pass this data frame to accuracy_score function.

Reply

Ammar
April 18, 2020 at 6:13 pm

thank you

I am using the KNN classification to train the model. the accuracy return is 17% !!! Why?

Reply

Sefik Serengil
April 18, 2020 at 6:16 pm

Aren't you use the pre-trained weights? Why you need training?

Reply

Ammar
April 18, 2020 at 6:28 pm

I am using deepface pre-trained weight that you put it here, but Instead of using your way to predict I am using KNN classification

yhat_class = knn_classifier.predict(test_faces_embedding)
score = accuracy_score(test_labels ,yhat_class)

Ammar
April 18, 2020 at 7:32 pm

Any help?

Tom
April 26, 2020 at 10:50 am

Hi...

I tested deepface model with 100 faces and it predicted only 18 persons correct.
– the dataset contains 46000 images (1000 persons)

Is that the accuracy of the model?

Reply

Sefik Serengil
April 26, 2020 at 11:33 am

You should apply face alignment first. Besides, using multiple images of a person increases the accuracy. As mentioned in the real time video, it finds correct faces in my experiments.

Reply

Tom
April 26, 2020 at 12:00 pm

I am using MTCNN to detect faces. It is by default using face alignment

Reply

Dinusha
July 14, 2020 at 4:50 am

Dear Safrik,

What are the versions you have used?
OpenCV –
Keras –
Tensorflow –
Thank You

Reply

Sefik Serengil
July 14, 2020 at 5:24 am

pip install opencv-python==3.4.4
pip install tensorflow==1.9.0
pip install keras==2.2.0

Reply

Pingback: DeepFace - Most Popular Deep Face Recognition in 2021 (Guide) | viso.ai

Nirali
August 27, 2021 at 9:48 am

Could you suggest a system configuration which would be optimal for deepface model using real time feed.
Thank you 🙂

Reply

Bob
March 9, 2022 at 2:27 pm

Dear Sefik,

I have a question, can I use the pre-trained weight to train a new different model. I'm planning to create a face recognition model for people wearing a face mask. Is it possible or do I have to train it from the scratch?.

Reply

Sefik Serengil
March 11, 2022 at 9:41 pm

Yes, you can. Please read the transfer learning posts in my blog (e.g. apparent age prediction)

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

ex: jane doe

Email *

ex: janedoe@gmail.com

Website

ex: http://janedoe.wordpress.com

Notify me of follow-up comments by email.

Notify me of new posts by email.

Submit

This site uses Akismet to reduce spam. Learn how your comment data is processed.

Please cite this post if it helps your research. Here is an example of BibTex entry:

```
@misc{sefiks14026,
  author ={Serengil, Sefik Ilkin},
  title = { Face Recognition with Facebook
DeepFace in Keras },
  howpublished = {
https://sefiks.com/2020/02/17/face-recognition-
with-facebook-deepface-in-keras/ },
  year = { 2020 },
  note = "[Online; accessed 2023-01-13]"
}
```