# Frame Similarity Detection and Frame Clustering Using Variational Autoencoders and K-Means on News Videos From Different Affinity Groups

Omer Fahri Onder
Department of Computer Science
Columbia University
oo2218@columbia.edu

Directed Research Report – Spring 2021
Advisor: Prof. John R. Kender

May 13, 2021

Abstract

Keyframe extraction and analysis is a helpful domain to analyze a video and its content. In this report, we experiment with a new method to find similarities between frames, extract keyframes based on context by clustering similar frames in news videos from two different affinity groups(English and Chines) and the same context(AlphaGo). Previously, manual and VGG-19 encoding-based approaches were used to analyze similarity for the "Tagging and Browsing Videos According to the Preferences of Differing Affinity Groups" project[1]. However, in light of the emerging popularity of Variational Autoencoders in frame extraction[3,4], we are developing a new way based on a simple, fully Convolutional Variational Autoencoder and K-Means to improve on the previously used methods. We also analyze the Variational Autoencoder's ability to capture affinity groups as contextual information in this report.

## 1 Introduction

### 1.1 Background and Previous Work

Initially, the project on "Tagging and Browsing Videos According to the Preferences of Differing Affinity Groups" used a manual approach for identifying essential frames and matching them across videos. As this approach has been quite demanding, there had been studies by other students[1]. Xu Han[1] has developed an automated unsupervised method to find these frames

[1] "Tagging and Browsing Videos According to the Preferences of Differing Affinity Groups" is an NFS Information and Intelligent Systems sponsored project to analyze the preferences of

using a  VGG-19[2][3] and hashing tool for encoding the frames and using L1, L2, and cosine distances to find the most similar frames. However, the VGG network used in this work is a pre-trained model for general purposes. Thus the model used for this approach cannot learn task-specific features, and in some cases, it does not provide desirable matches. The finetuning of this model would require manual labeling, which is very costly.
The proposed method has been inspired by the excellent work of Xu Han and started as an improvement for the encoding approach with a Variational Autoencoder(VAE), which is trained on task-specific videos.

## 1.2 Literature Study

The keyframe extraction using clustering has been a common area of research for video representation and video analysis for a while. In 2005, Yang and Lin[4] proposed an algorithm based on statistical modeling. Their paper uses an automated method by using a statistical model as thresholding for clustering and finding more critical content.
Although their method is quite impressive, over the years, there has been improvement newer studies on image representation with the emergence of Deep Learning methods such as Variational Autoencoders(VAEs). VAE autoencoders are Deep Neural Networks trained to extract abstract information by first encoding an input such as a video frame into a latent space, then decoding the latent variables back to their original form. This method can learn to create a representation space without supervision.

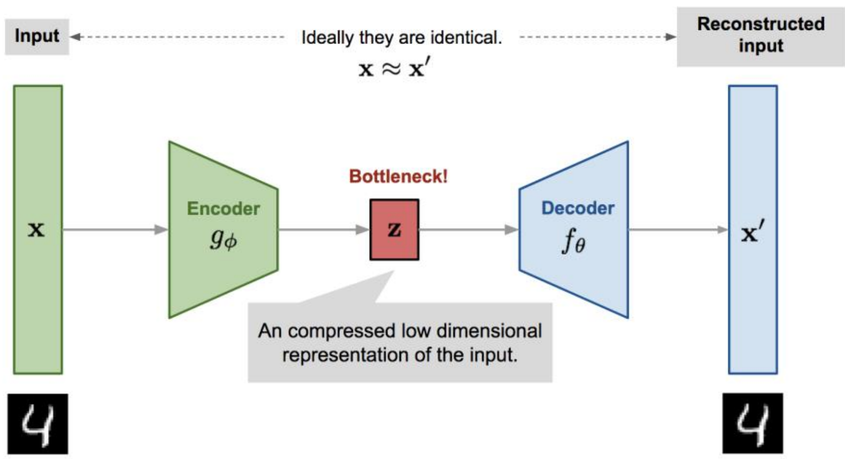



Figure 1: Representation of a generic Autoencoder.
https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

Both Pei et al.[6] and Yang et al.[7] have developed used this benefit of VAEs to develop task-specific frame extraction networks. Pei et al. use stacked VAEs to detect keyframes for action detection, while Yang et al.[7] developed a sophisticated algorithm with semi-supervision to extract video highlights.

In their approach, Yang et al. use Recurrent Autoencoders to encode the frames and semi-supervision based on reconstruction error to detect outliers between these frames.

As can be seen from the above examples, VAEs are potentially beneficial tools for extracting information from video frames without supervision and catch differences between these new embeddings. This study uses this benefit to skip the manual labeling that the VGG-19 model would require.

# 2 Methodolody

## 2.1 Data

We have collected videos from a single context to analyze the differences between different affinity groups: News reports on AlphaGo, and two groups: Chinese news videos and English news videos.

The English videos are all recently collected from YouTube, while Chinese videos are collected by previous research assistants, mainly from Bilibili and other Chinese video channels.

There are a total of 10 training and 2 test videos for both affinity groups. The training data are used to train the VAE, the clustering, and the Z-normalization distribution model.

The frames are sampled with a frequency of 1 frame per 2 seconds. The videos are rescaled to (240, 426, 3), then each sampled frame is cropped to get the input shape of (240, 400, 3) to match the input and output dimension of VAEs.

## 2.2 Variational Autoencoder(VAE) architecture

For the Variational Autoencoder(see Figure 2), we constructed a fully convolutional Autoencoder. The encoder is a stack of 2D Convolutional operations followed by Max Pooling operations. The Convolutional Layers in the decoder follow the same parameters as the encoder's, but it uses Transpose Convolutional instead of Upsampling to minimize loss of information. The bottleneck of the network has 128 filters which mean 128 different feature maps are detected.

The model is trained on the selected frames from data processing for both input and targets which have the same shape of (240,400,3).

The loss for the model is pixelwise mean squared error and the model is trained using Adam optimizer with a learning rate of 0.0001.

After training the network, we use Global Max pooling to get the highest value for each feature map to get our embeddings, reducing each feature map into a single vector of features. We keep the new model with Global Max Pooling as for embedding model.

We trained three separate models with the same parameters: 1 for English news videos only, 1 for Chinese news Videos only, and 1 for both English and Chinese news videos.

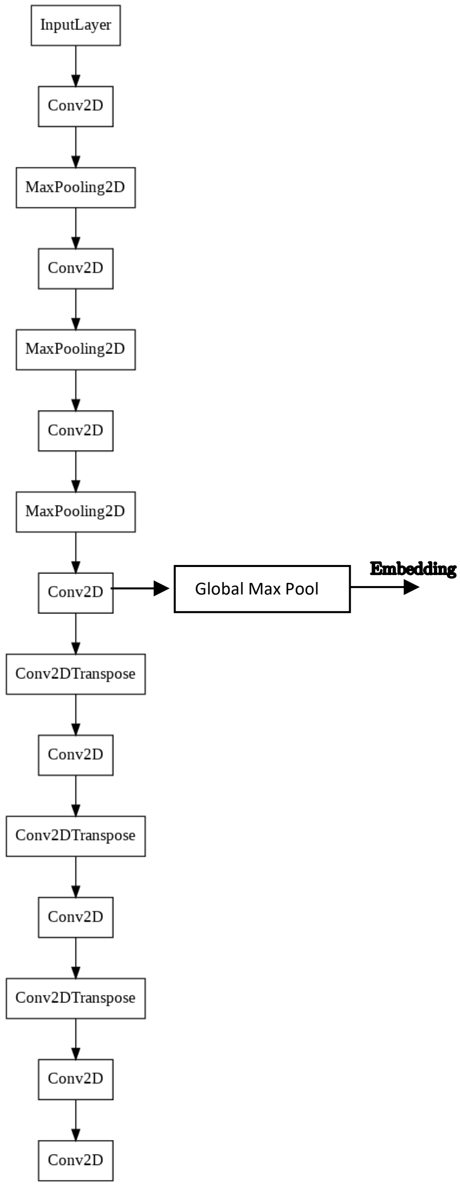



Figure 2: VAE architecture.

## 2.3 Frame Similarity Analysis

For this part, we use the approach previously used by Xu Han[1]. We check the L2 distance and Cosine similarity on the embedding vector with 128 variables to detect similar frames.

However, we mainly use similarity analysis on a video to see the similar frames in each video.

### 2.4 Frame Clustering

We trained K-Means[8] clustering models for each test video with 2 to 18 clusters for similar frame clustering.

For clustering, we have also tried different approaches to normalize the data.

1. Using the sigmoid activation function in the deepest Convolution layer of the VAE to automatically map the values between 0 and 1.
2. Using Z-normalization mapping based on distribution of the training data which is then used to normalize the test data.

We used the elbow method to choose the optimal number of clusters [9], which is a heuristic method of picking the cluster when the within-cluster sum square starts to converge(bend like an elbow).

Lastly, we trained a cluster with 2 and 3 clusters on the combination of all test videos to see if a simple clustering algorithm can separate videos based cultural affinity.

## 3 Results

### 3.1 VAE results

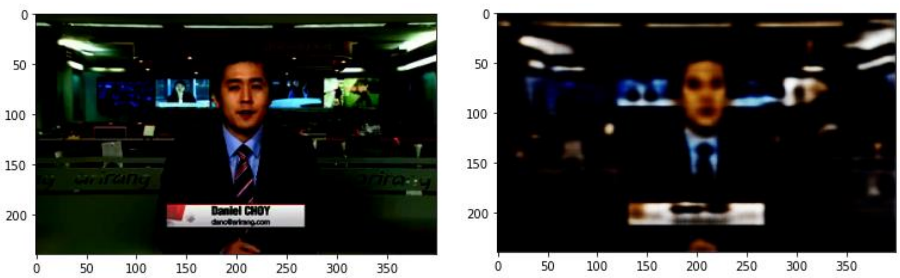
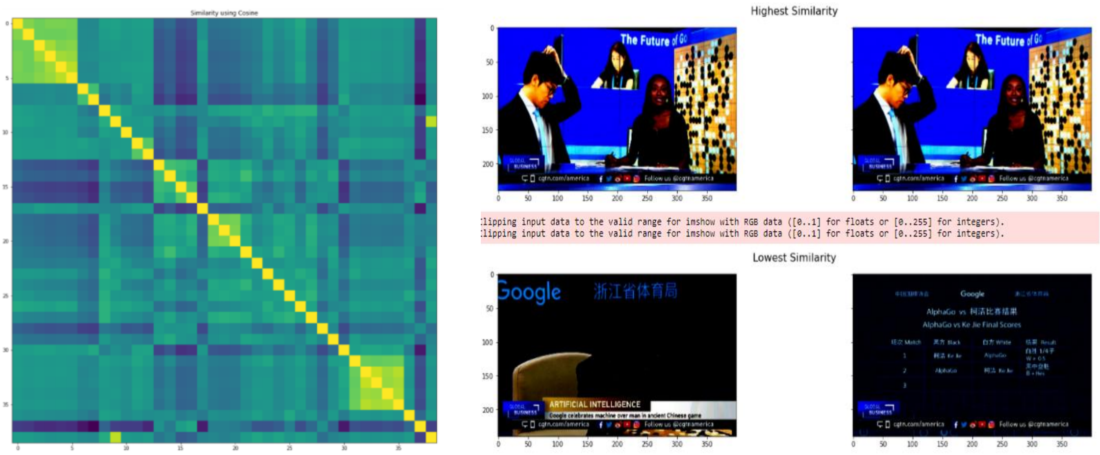



Figure 3: VAE qualitative evaluation: Input on the left and output on the right.

After the training, the model reached 61% pixel-wise accuracy on all three models, which does not tell enough information about the model to evaluate it. However, from Figure 3, we can see that the model could reconstruct many features to an abstract level, like the text box at the bottom of the human face on the screen at the back and the reporter's face.

## 3.2 Similarity Results




a. Cosine similarity b. Frames with highest similarity and lowest similarity.

Figure 4: Frame similarity analysis in a Chinese news video.

As shown in Figure 4 there is a high similarity between frames that are closer to each other in terms of their timestamp. This is likely because these frames are from the same shots or moving shots. There is a correlation between some frames that are not close to each other, likely the shots with reporters.

When we check the most similar frames, we can see that the most similar frames are almost identical; however, the most dissimilar from, although very similar to each other in terms of the color scheme, very different from each

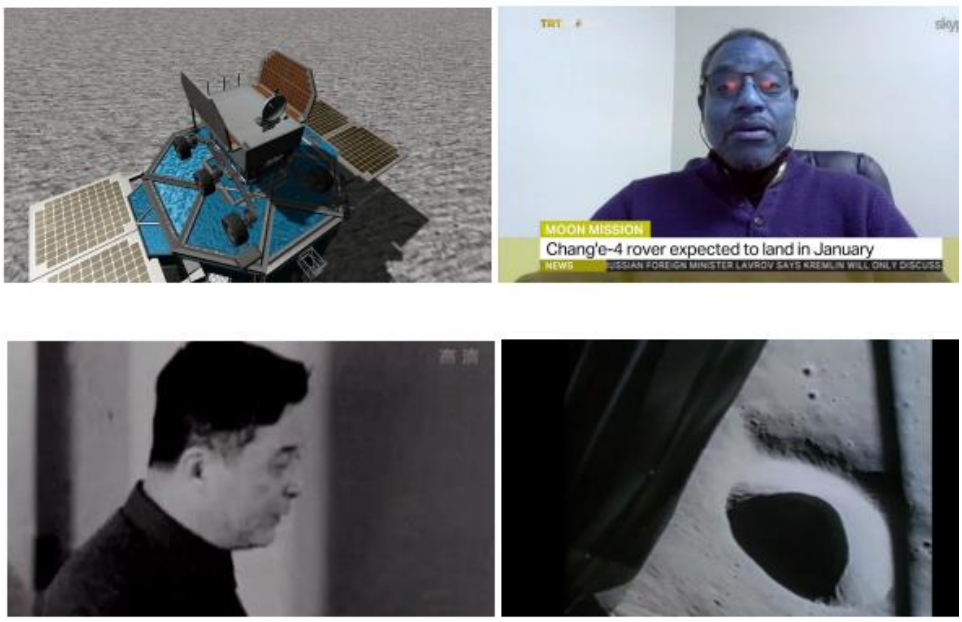



Figure 5: Examples picked from Xu Han[1] that were at the highest similarity with very different content but very similar color scheme.

other based on the content of the frames.

Comparing our result with the previous work[1], where the VGG19 + hashing would sometimes give a high similarity result based on the color scheme and some geometrical similarity, we can say that the VAE approach has shown promise in similarity detection.

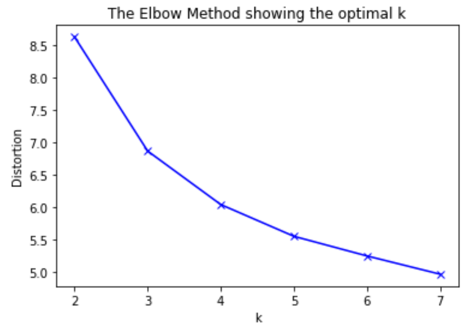
## 3.3 Clustering

Based on the elbow method, we see that the K-Means algorithm would start to converge around 2-4 clusters per minute of a news video.

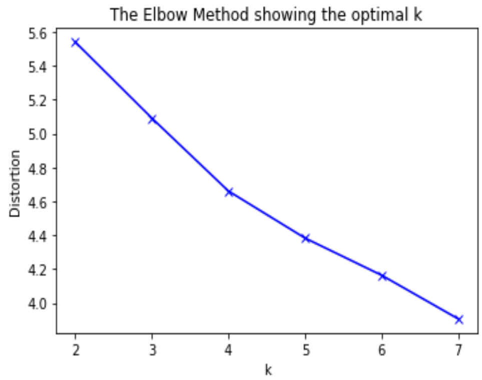
The model trained on news from both groups performed better on clustering frames from Chinese Videos.

On the other hand, when we combined all the test videos for a more comprehensive analysis on the VAE model trained on news from both affinity groups, we found that the clusters are well separated between their original videos or similar frames across videos(see Appendix 1).

We have also tested if a clustering algorithm trained on the combined test data can perform well enough to separate the frames into their affinity groups. Surprisingly the simple K-Means algorithm could divide the frames into two separate groups based on their affinity.




(a)




(b)

Figure 6: Cluster analysis on a one minute Chinese new video(a) and on an English news video(b).

When we experimented with the same method with 3 clusters instead of 2, the clusters were divided into Only English frames, only Chinese frames, and a mixture of both.

```
[2, 1, 1, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 1, 2, 2, 2, 2, 1,
 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 2, 2, 1, 1, 1, 2, 2, 2, 1, 1, 1, 2,
 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 2, 1, 1, 1, 2,
 2, 1, 2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
```

Figure 7: The cluster appointment result for each frame for K-Means clustering with K=3. The highlighted blue numbers represent the assignment of frames from English Videos. The non-highlighted frames belong to Chinese news videos. The values(0,1,2) are the cluster index.

As can be seen from Figure 7, cluster 0 exclusively belong to Chinese news videos while cluster 2 exclusively belongs to English news videos except for two frames. Cluster 1 are frames that share commonality between these two types of affinity groups.

# 4 Conclusion

A pretrained VGG model is lacking in capturing task-based context information unless trained with new data to tune it for the new context. However, training a VGG model would generally require supervision and extensive labeling of video frames; on the other hand, Autoencoders can be trained with just the training data alone. Using this benefit of Autoencoder, we tried to improve previous similar frame detection algorithms and develop a frame clustering pipeline.

The simple, fully Convolutional Autoencoder has performed very well in capturing contextual information and using it in the context to separate frames that have similar geometric or color schemes. Moreover, the frame clustering created meaningful clusters for each news video, especially for the Chinese news videos.

This framework was also able to group the frames into 3, separating some frames into clusters exclusive to their affinity group, exceeding our initial expectations.

We can conclude that the clustering pipeline based on VAE can improve on the previous similar frame searching method and open a new research direction for the "Tagging and Browsing Videos According to the Preferences of Differing Affinity Groups" project.

## 4.1 Possible Future Work

1. Other types of VAE: For a better embedding, a VAE with a bottleneck with fully connected(dense) layer could be used.
2. Gausian Mixture in VAE for clustering: With a gaussian constraint, the embedding space result from VAE can be also used for clustering.
3. Affinty group clustering research: The last finding from the experiment can start a new research direction.

# References


[1] H. Xu , "Identifying Near-duplicate Frames from Two Sets of Videos," *Tagging and Browsing Videos According to the Preferences of Differing Affinity Groups*.

[2] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *International Conference on Learning Representations*, 2015.

[3] K. Team, "Keras documentation: VGG16 and VGG19," *Keras*. [Online]. Available: https://keras.io/api/applications/vgg/. [Accessed: 14-May-2021].

[4] S. Yang and X. Lin, "Key frame extraction using unsupervised clustering based on a statistical model," *Tsinghua Science and Technology*, vol. 10, no. 2, pp. 169–173, 2005.

[5] D. P. Kingma and M. Welling, "An Introduction to Variational Autoencoders," *Foundations and Trends R in Machine Learning*, pp. 1–18, 2019.

[6] X. Pei, H. Fan, and Y. Tang, "An Unsupervised Feature learning and clustering method for key frame extraction on human action recognition," *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, Aug. 2017.

[7] H. Yang, B. Wang, S. Lin, D. Wipf, M. Guo, and B. Guo, "Unsupervised Extraction of Video Highlights via Robust Recurrent Auto-Encoders," *2015 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2015.

[8] "sklearn.cluster.KMeans¶," scikit. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html. [Accessed: 13-May-2021]

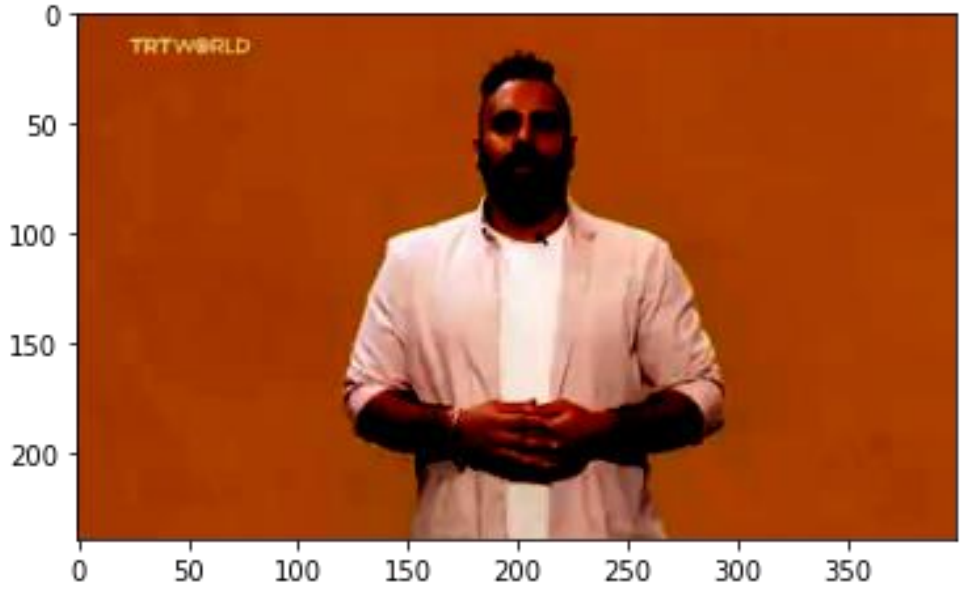
[9] "Determining The Optimal Number Of Clusters: 3 Must Know Methods," Datanovia, 21-Oct-2018. [Online]. Available: https://www.datanovia.com/en/lessons/determining-the-optimal-number-of-clusters-3-must-know-methods/. [Accessed: 13-May-2021].

# Appendices

## Appendix 1: 15 Clusters Result on Combined Test Videos

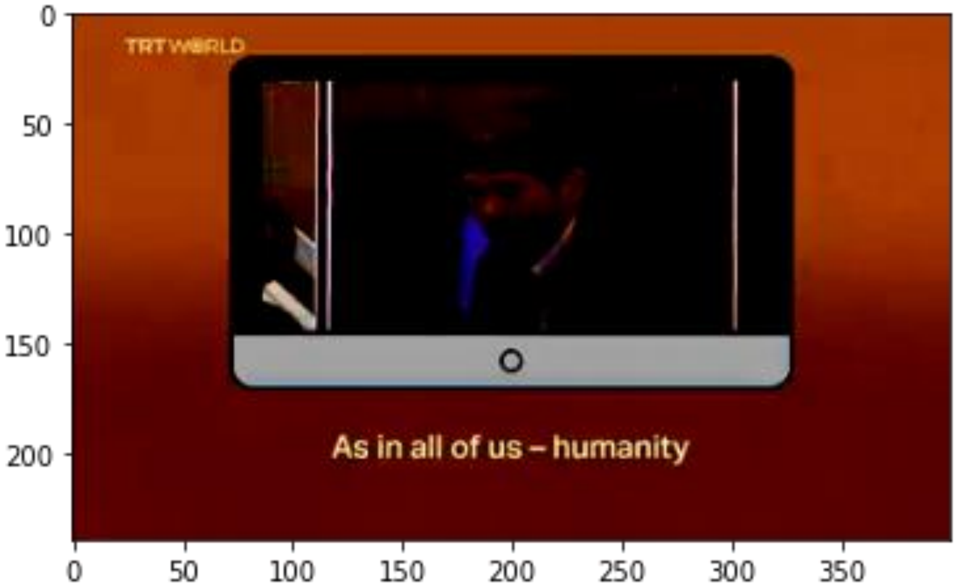
```
Some clusters may be missing.
```

#### 4.1.1 --------- GROUP: 6 ---------

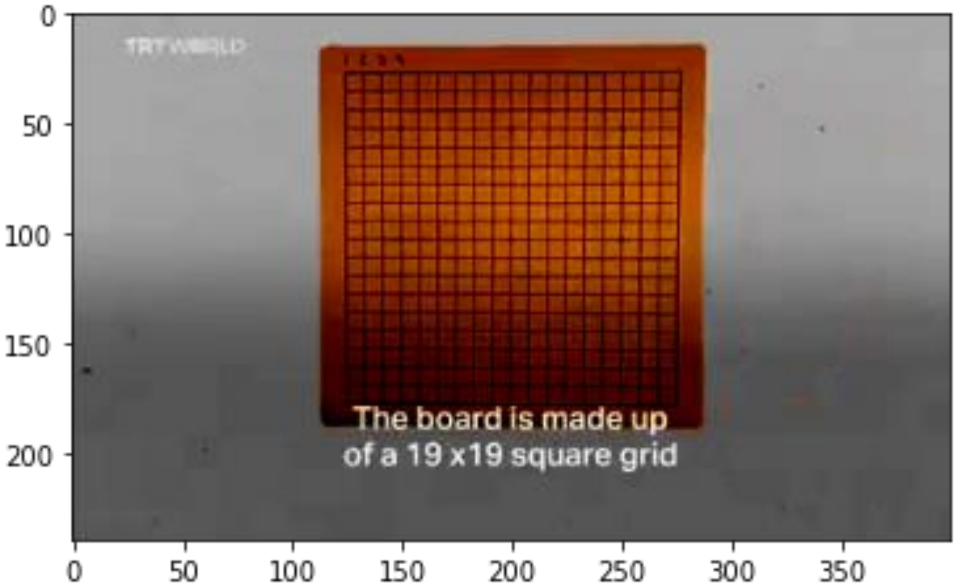



```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

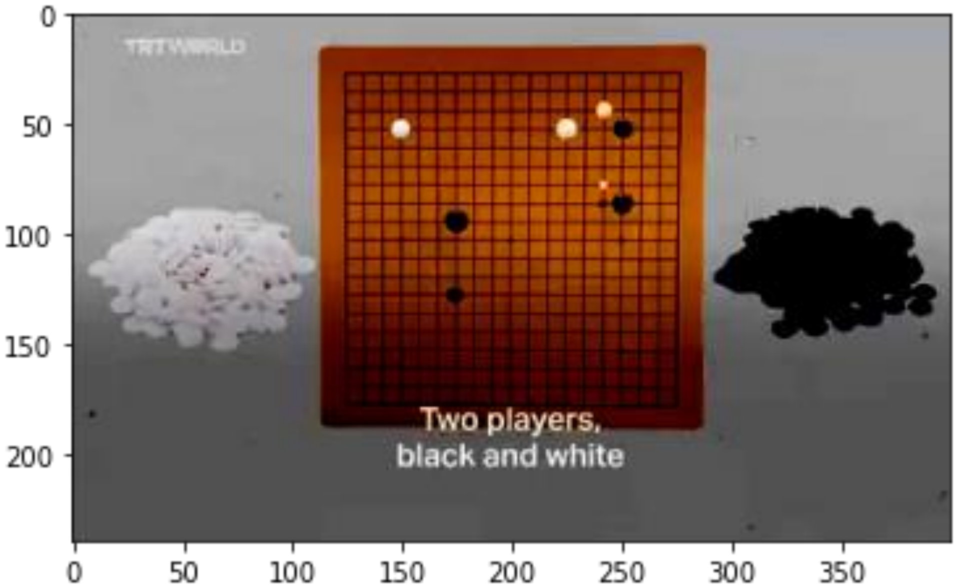
```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```



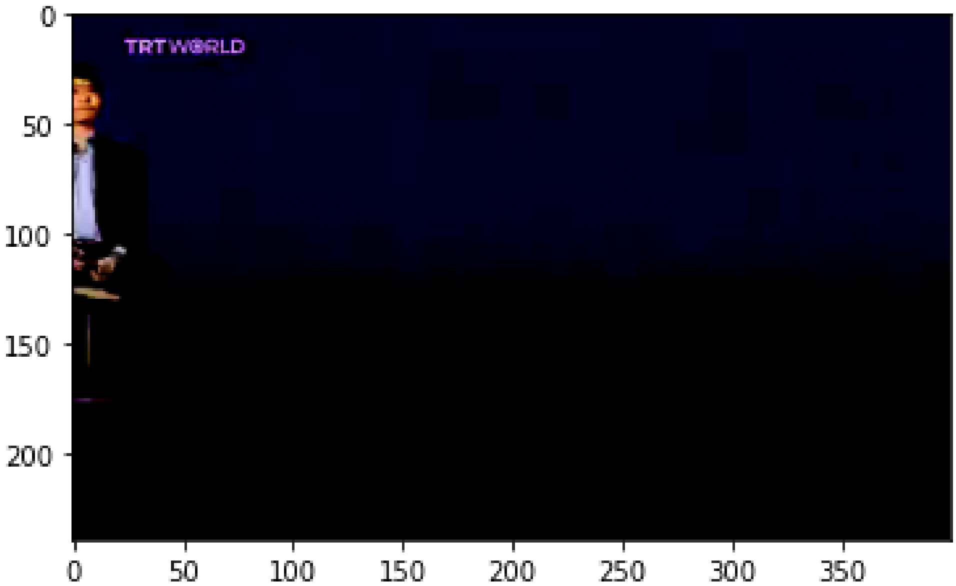

```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```



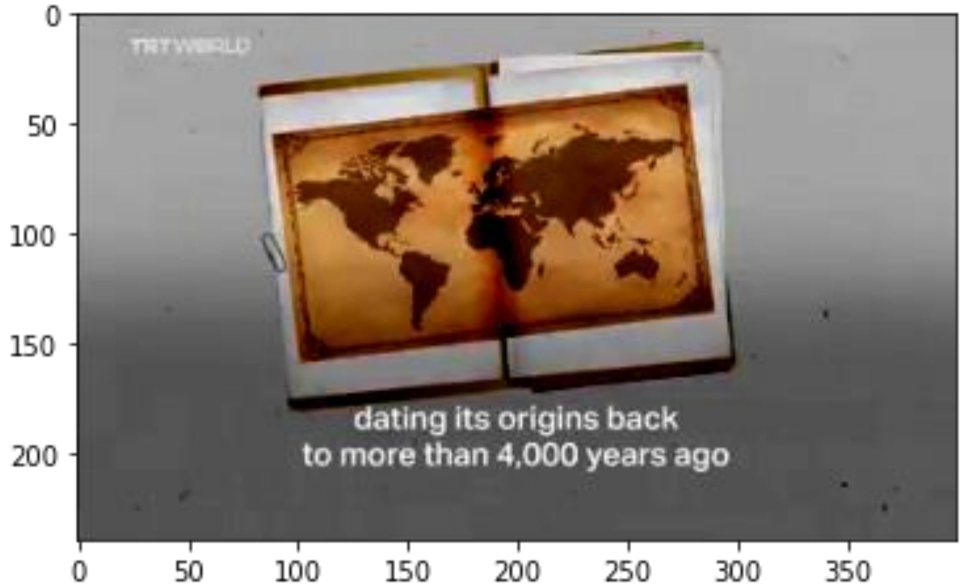

```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```




Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



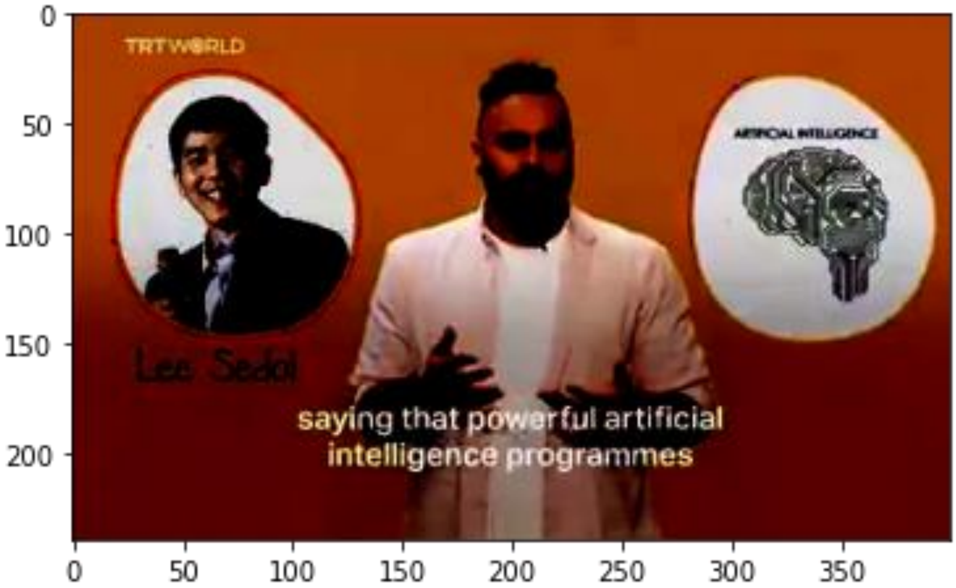

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).




```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

### 4.1.2 --------- GROUP: 8 ---------


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```




```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

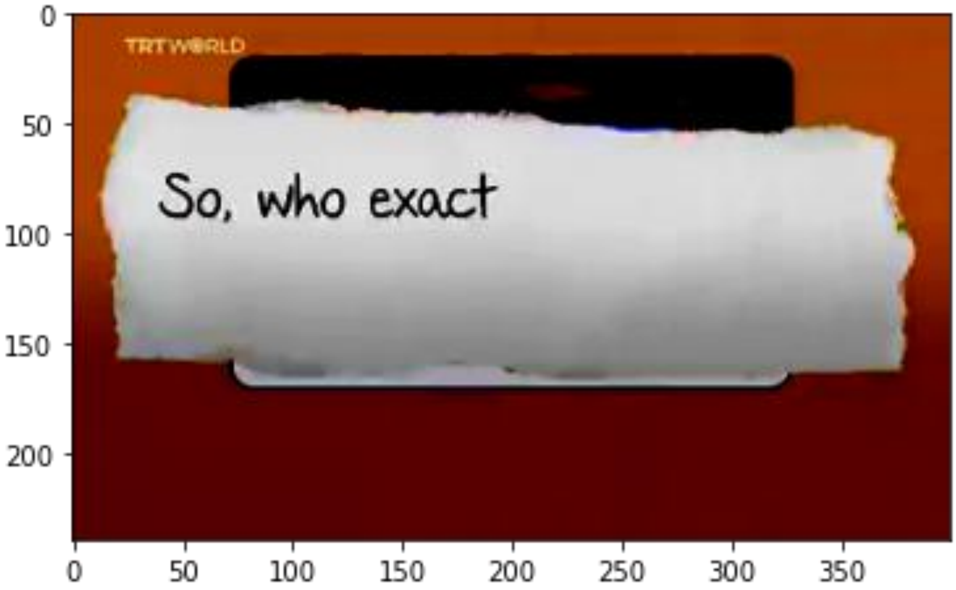

```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

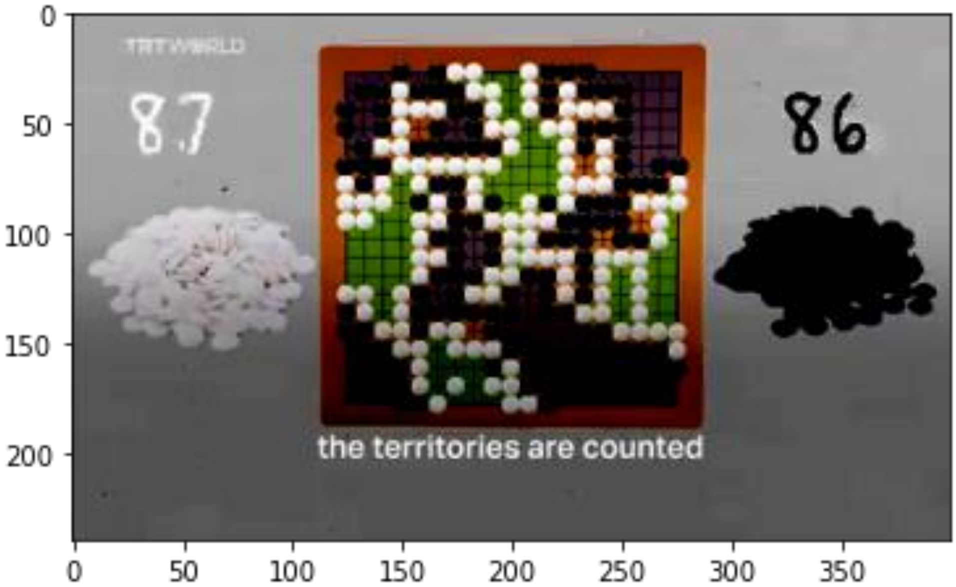

```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

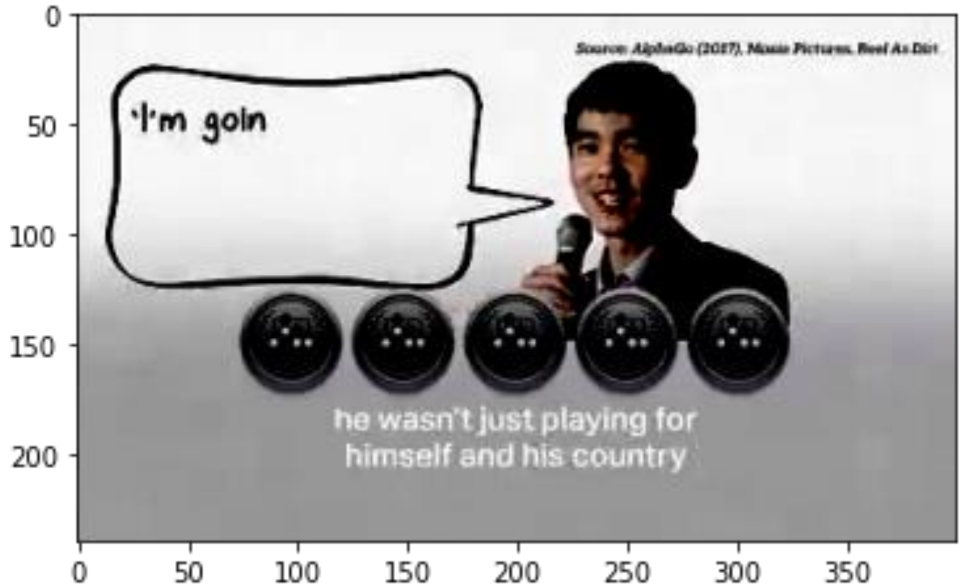
4.1.3 --------- GROUP: 10 ---------




```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

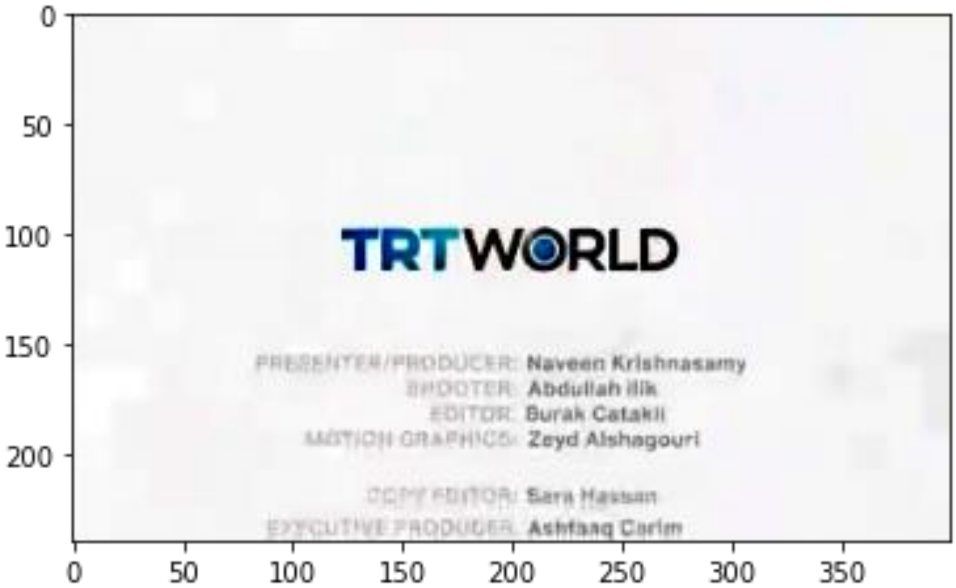



```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

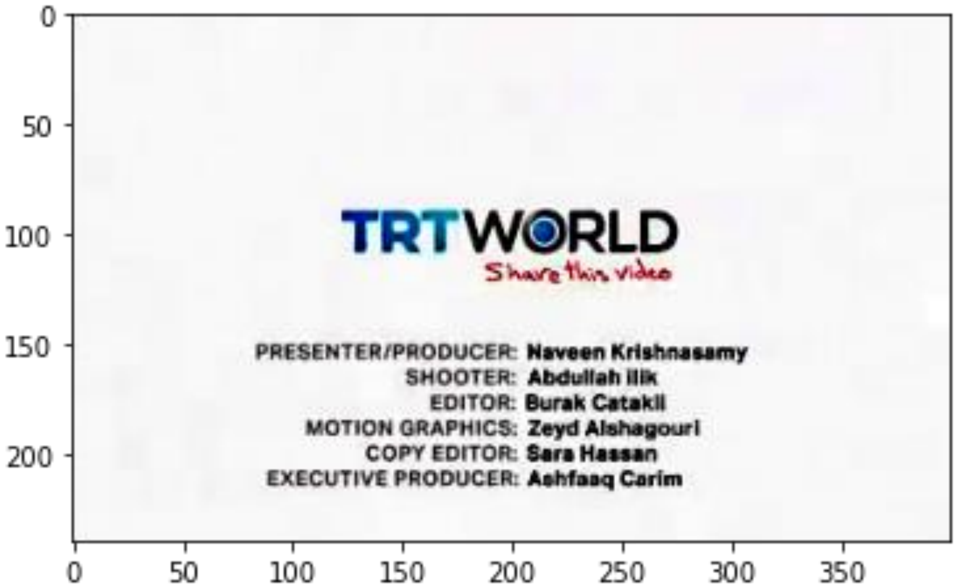



Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

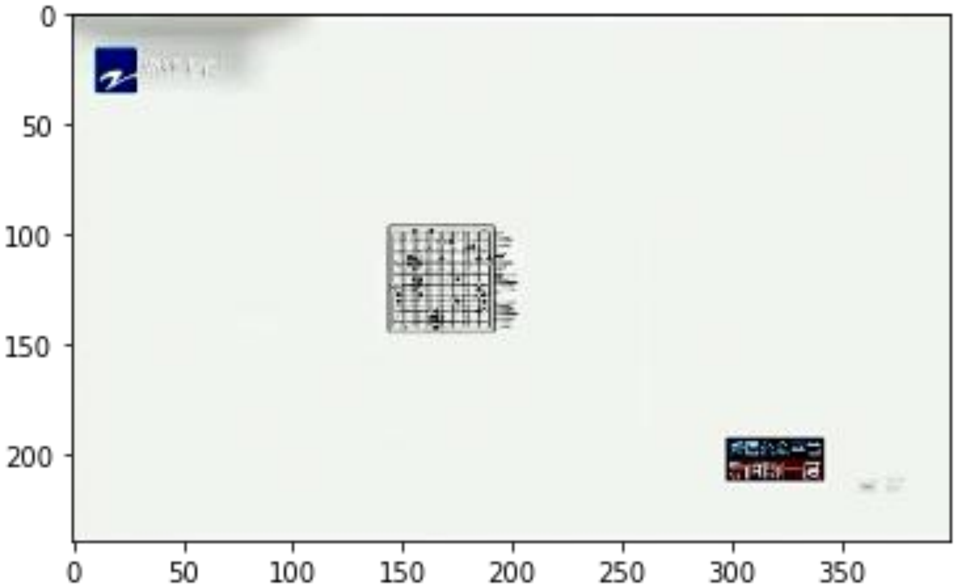

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).




Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).

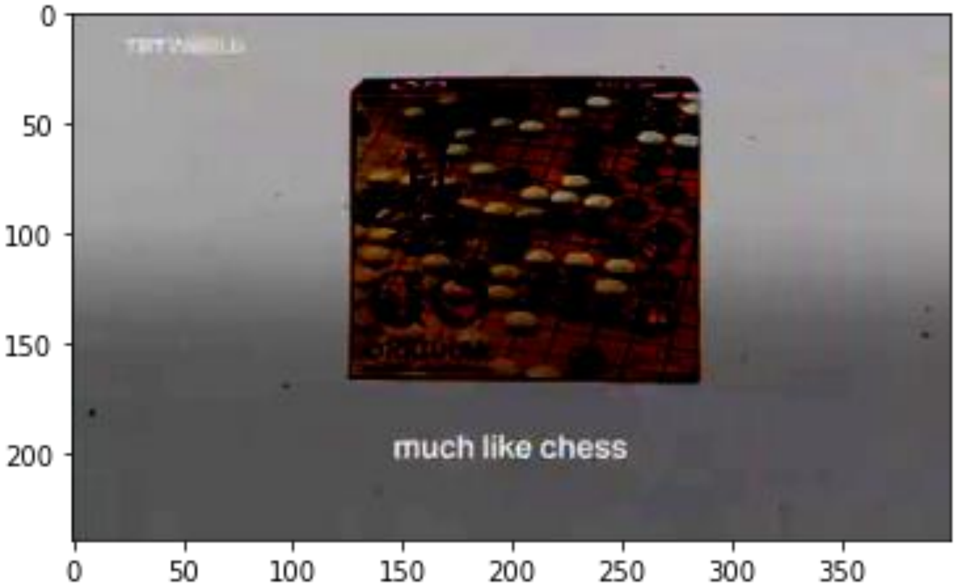



Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).

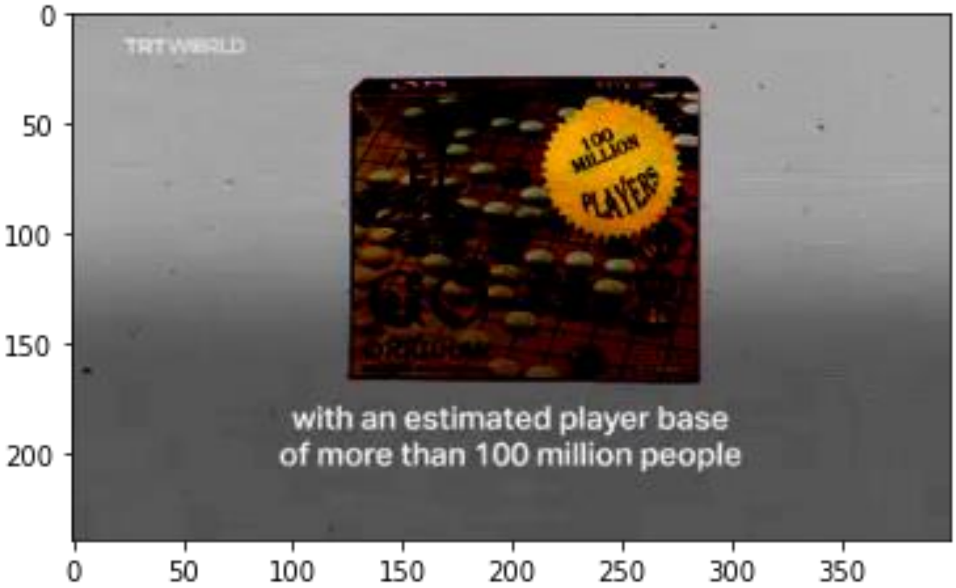



Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).

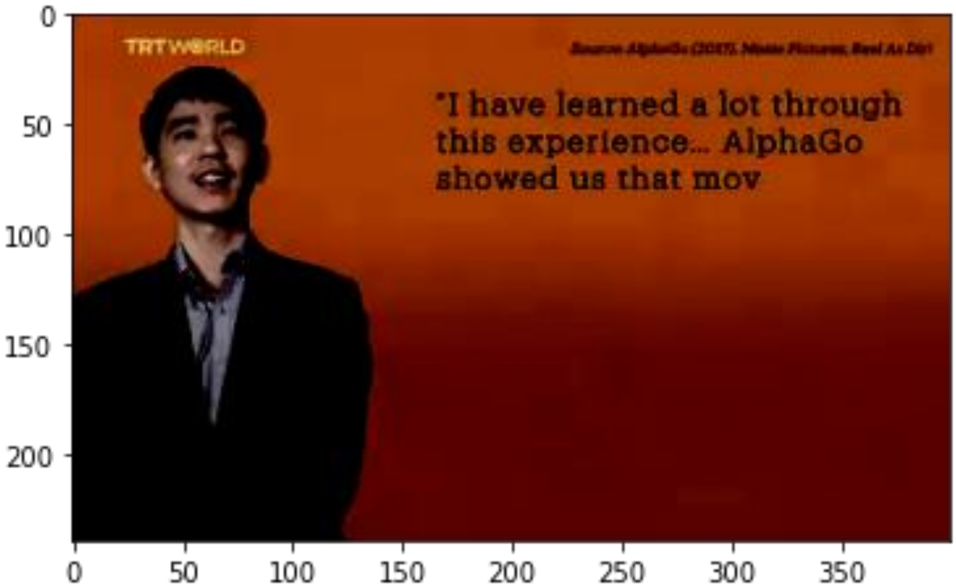
### 4.1.4 --------- GROUP: 12 ---------



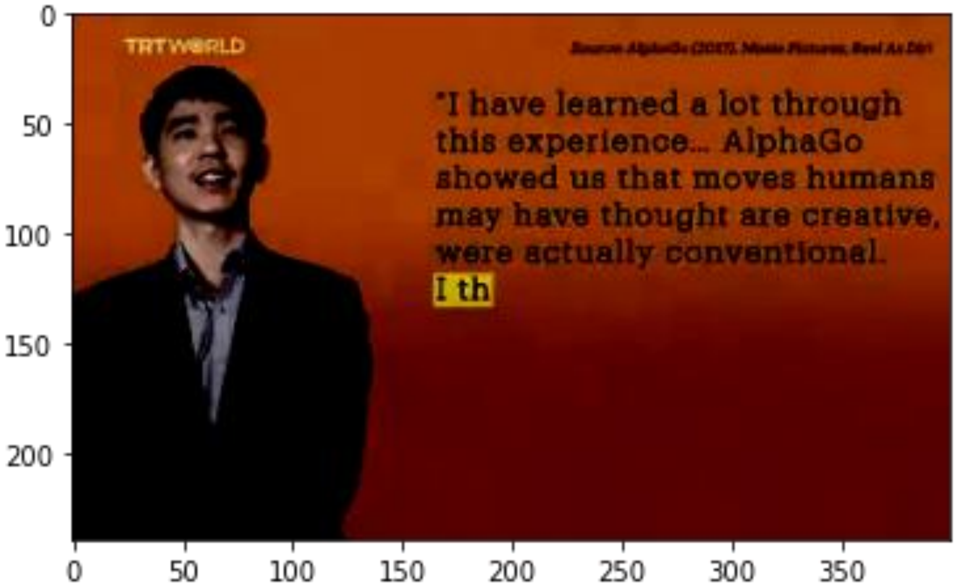

Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).



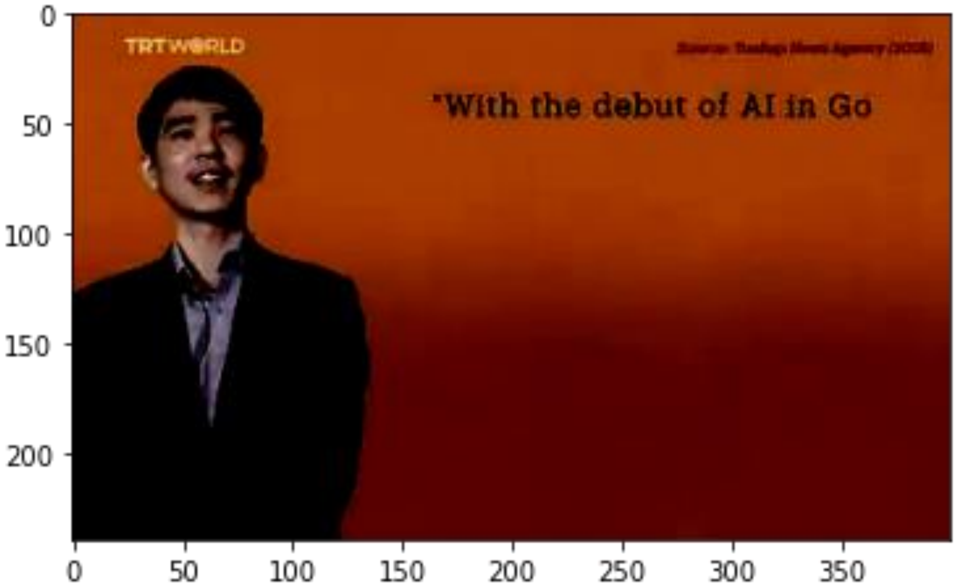

Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).



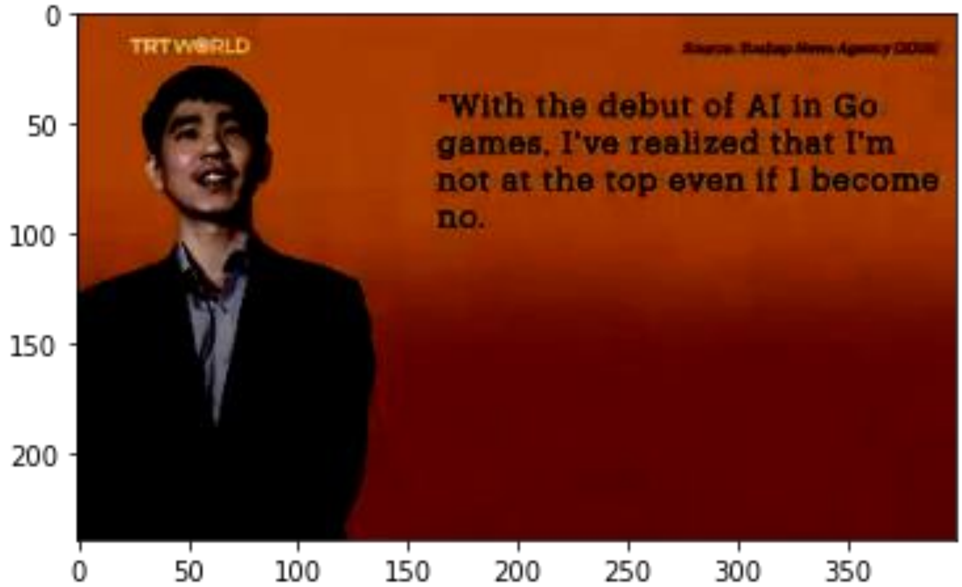

Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).




Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).



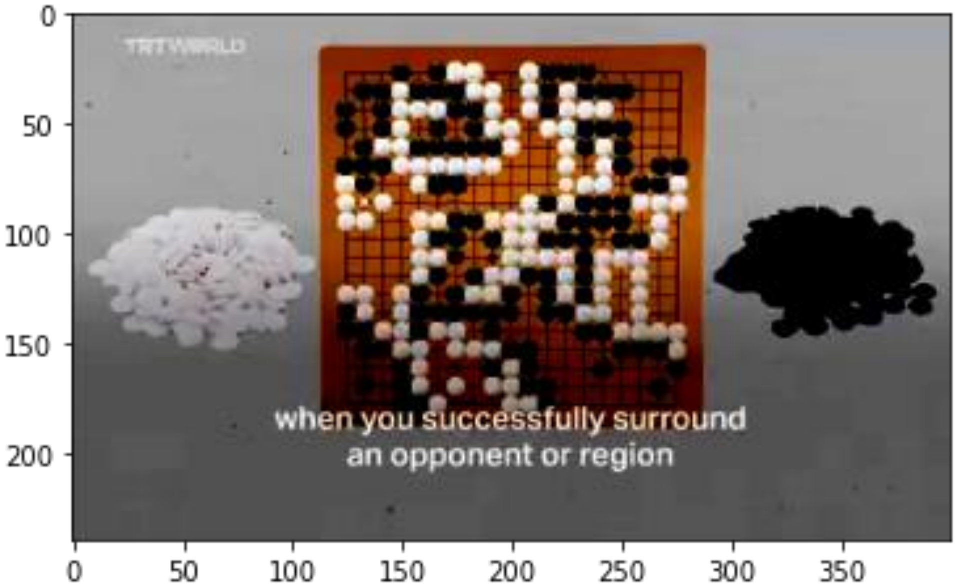

Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).



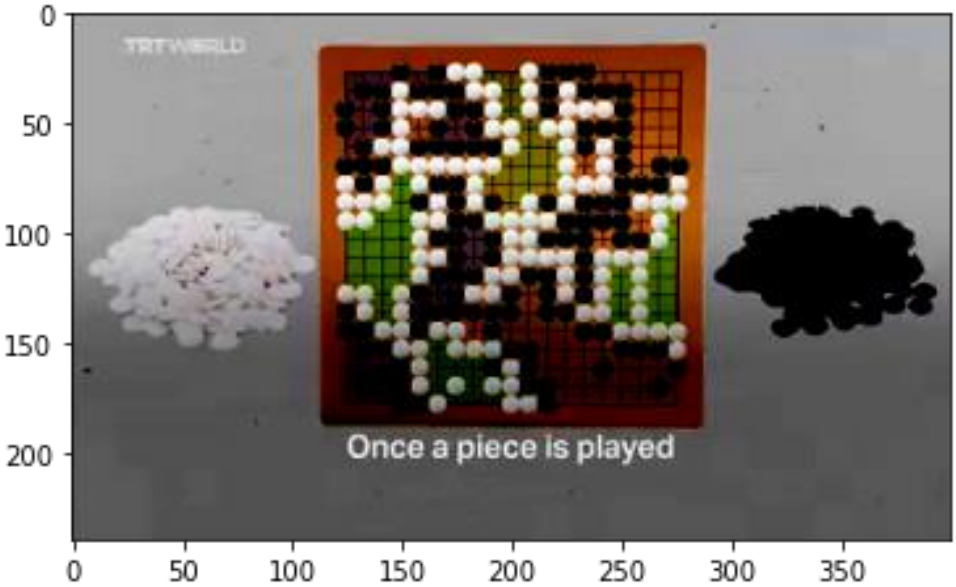

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).


Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

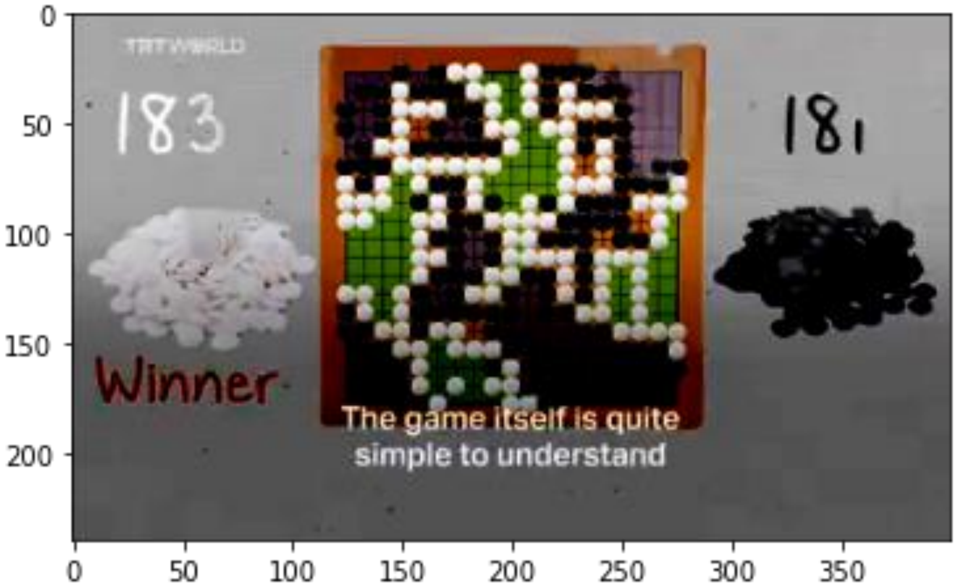
#### 4.1.5 --------- GROUP: 0 ---------




```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```




```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

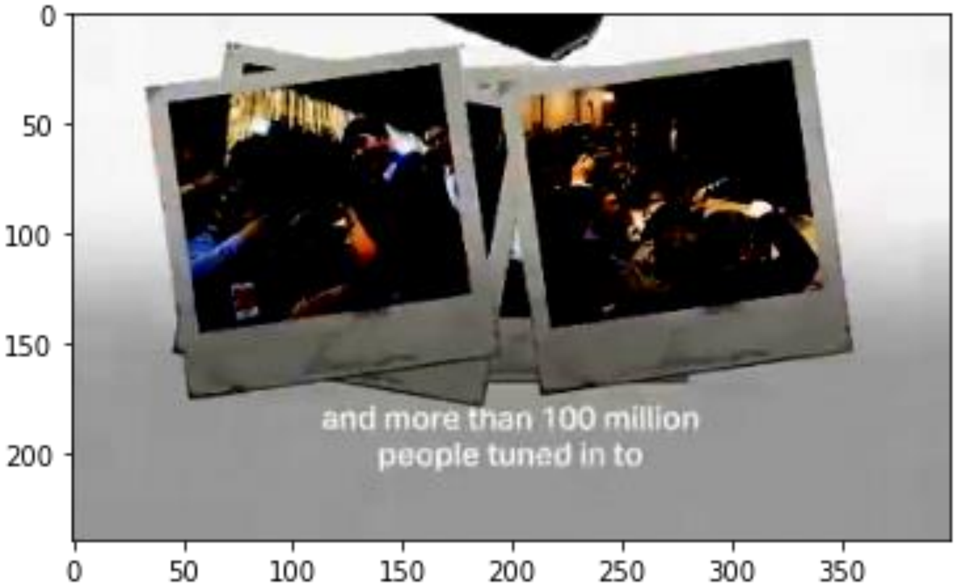



```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

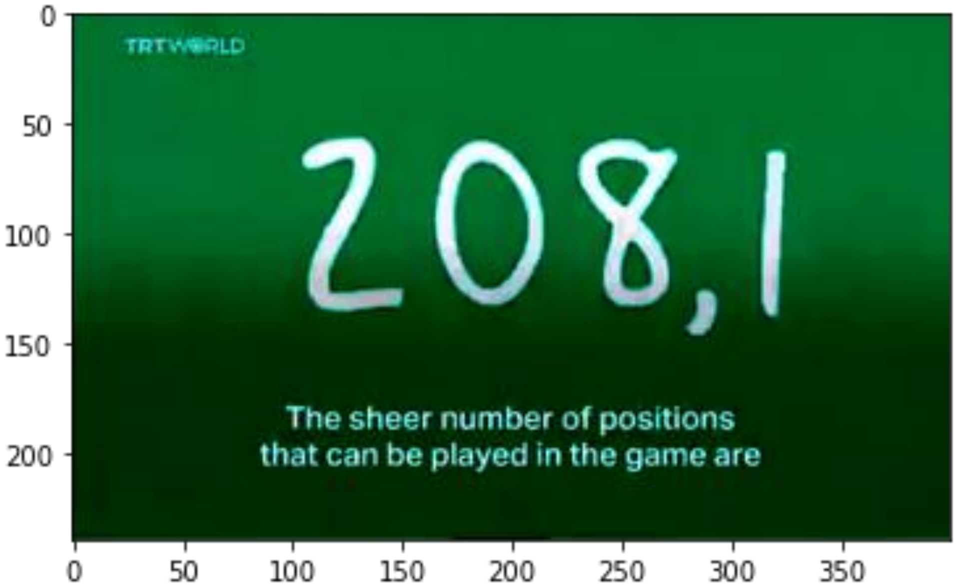

```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```



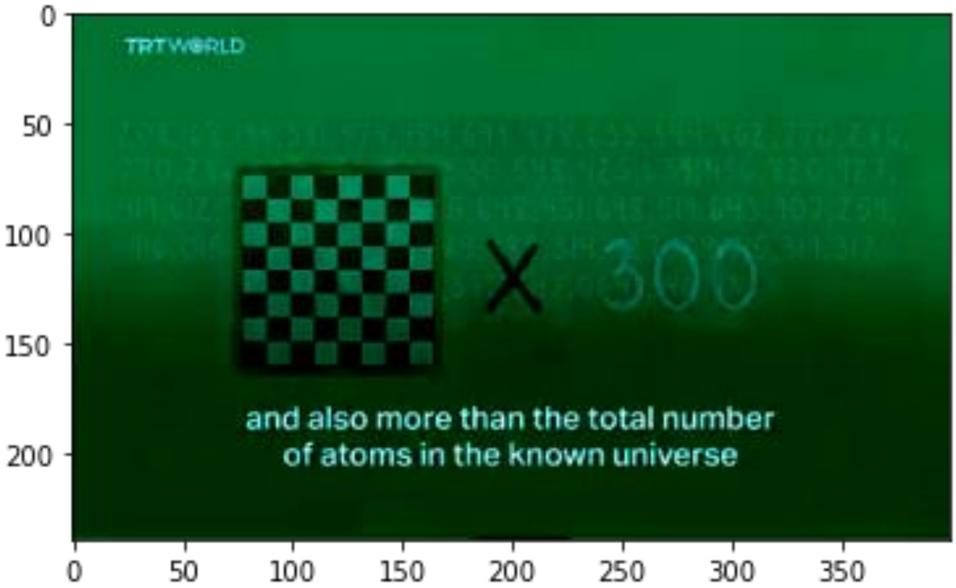

```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

### 4.1.6 --------- GROUP: 11 ---------

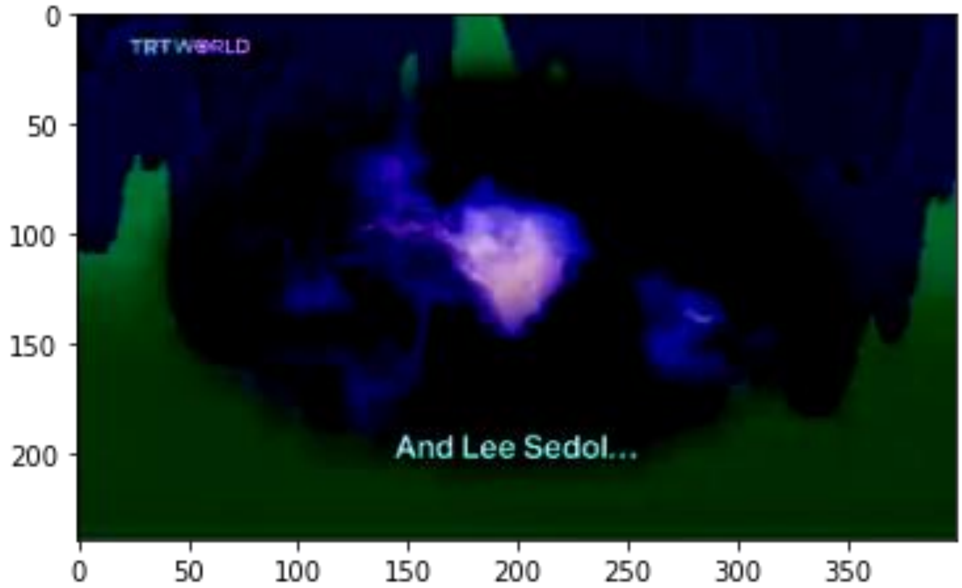



```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```


Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).




Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).




```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

### 4.1.7 --------- GROUP: 14 ---------

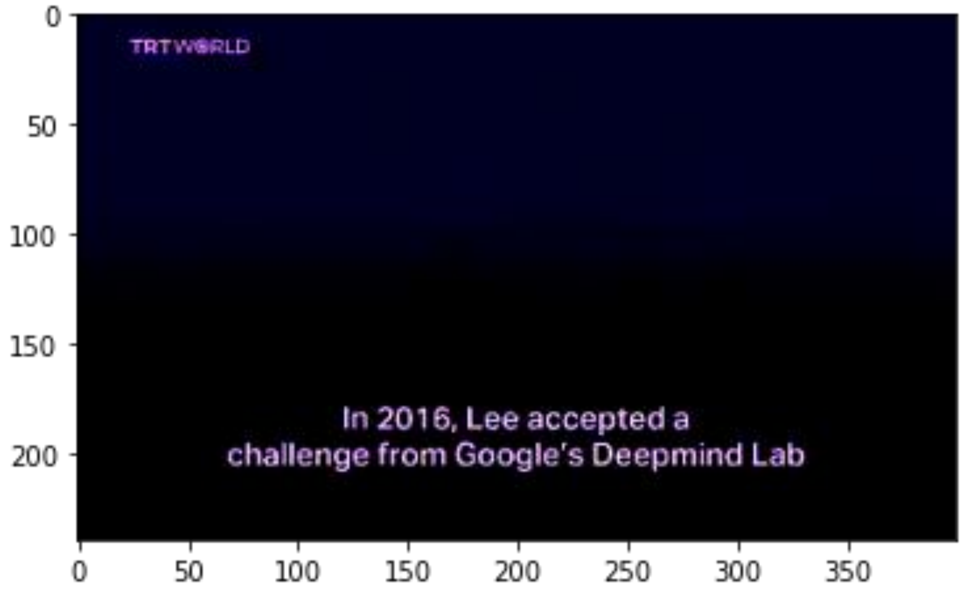

```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```


Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

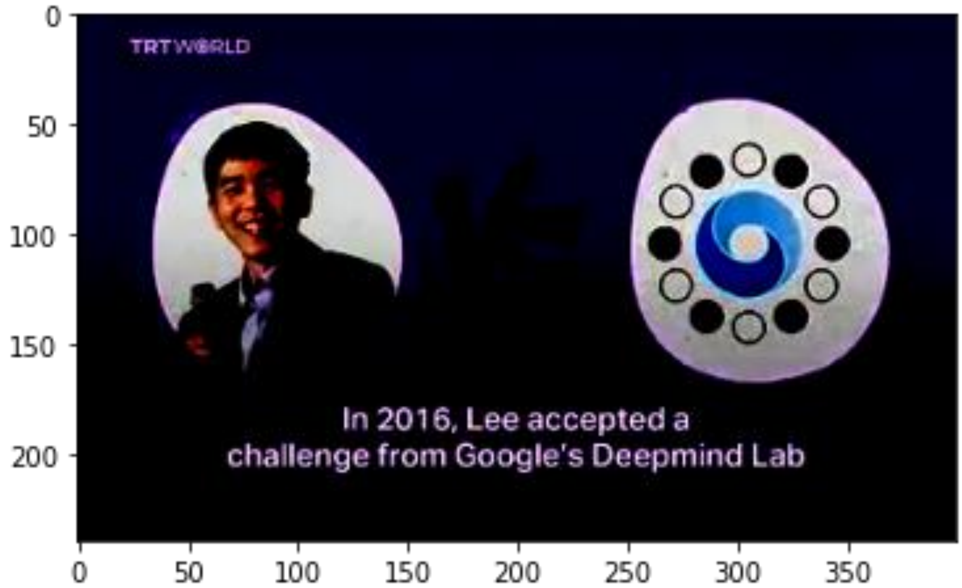

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).


Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).


Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).




```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```



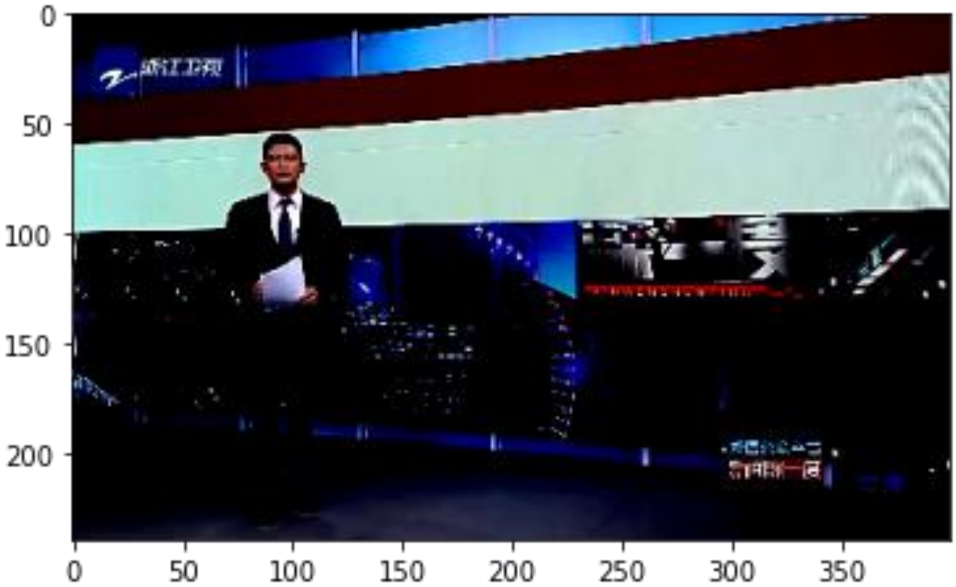

```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

4.1.8 --------- GROUP: 3 ---------


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```




```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```


Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

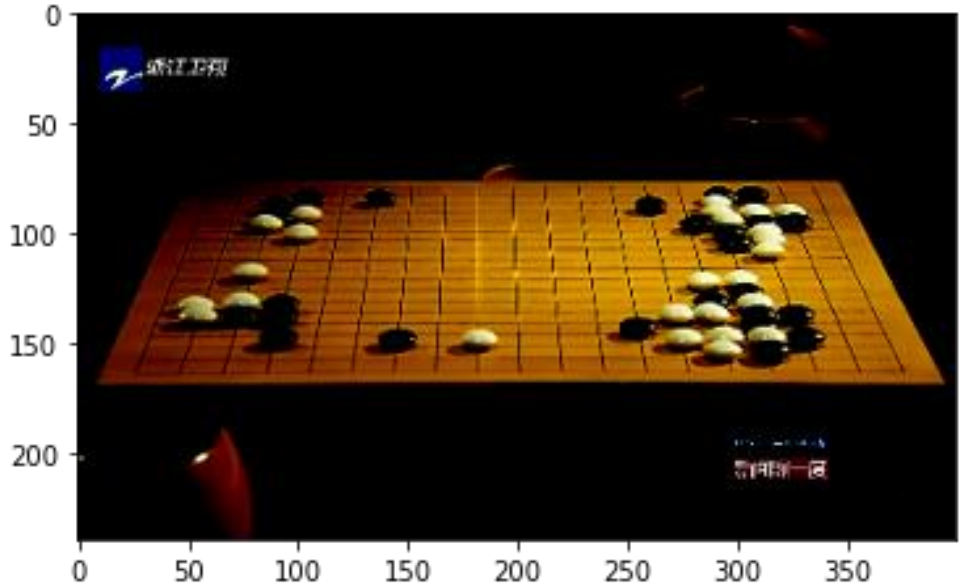

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

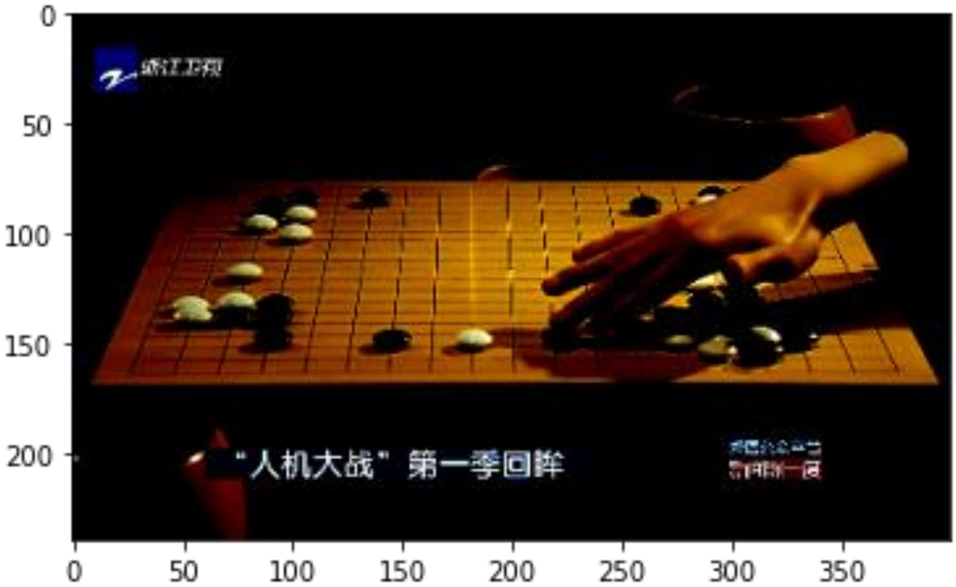
### 4.1.9 --------- GROUP: 7 ---------


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```


Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

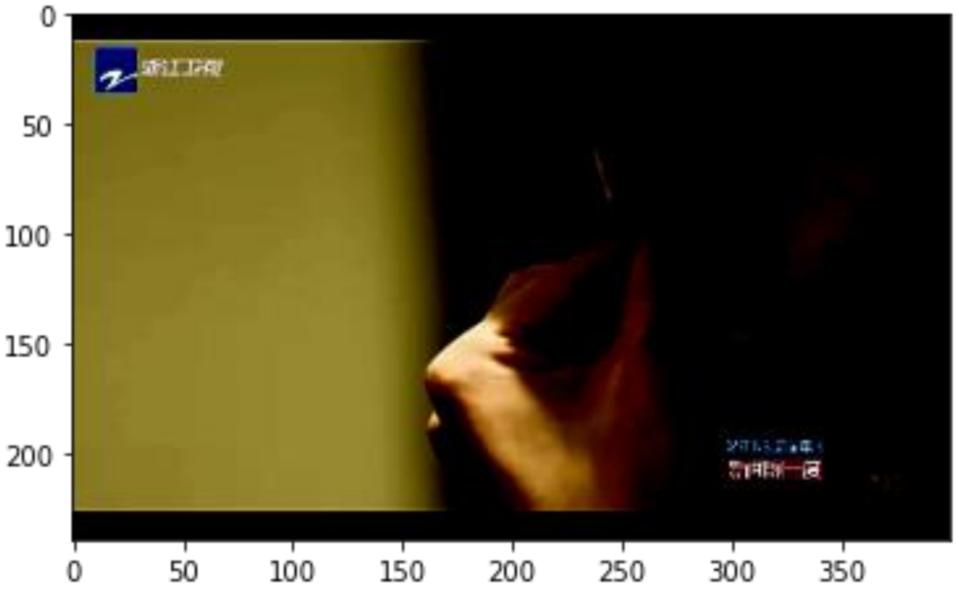



Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

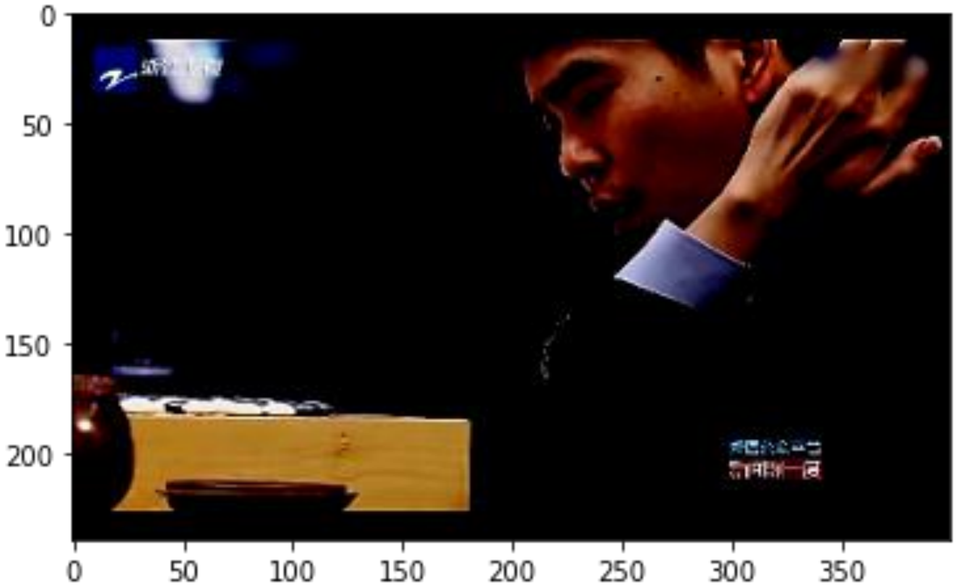



Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).




Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).




Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

#### 4.1.10 --------- GROUP: 5 ---------


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

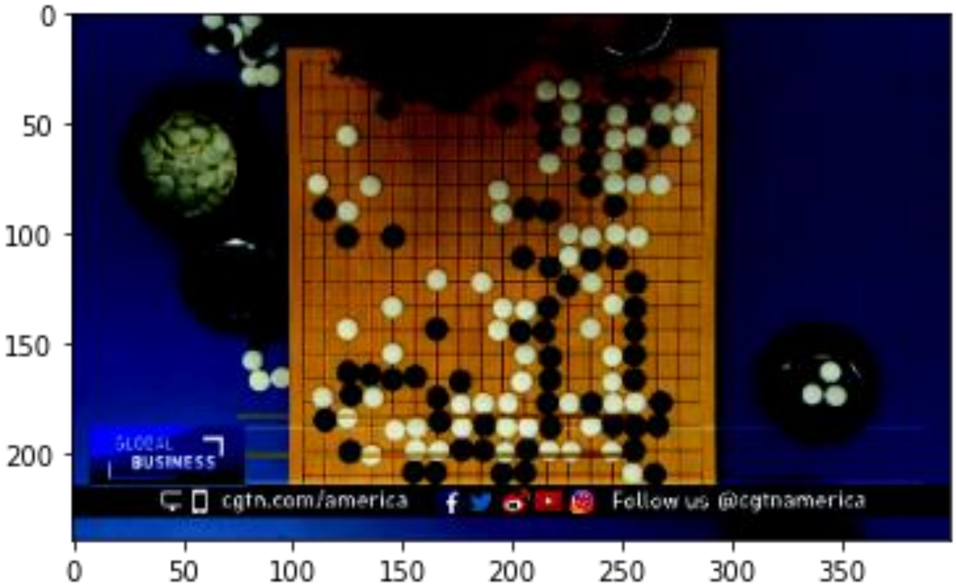

```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

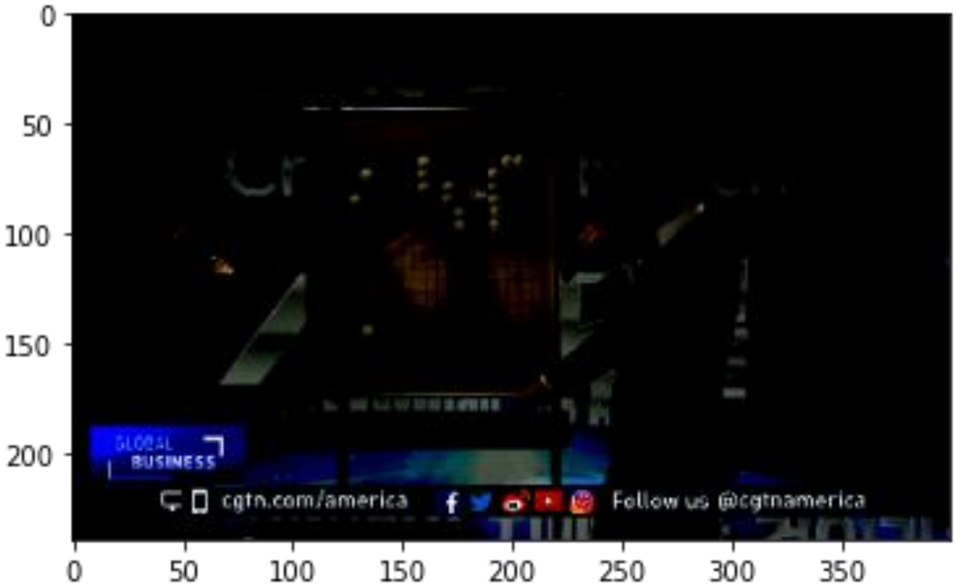

```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```



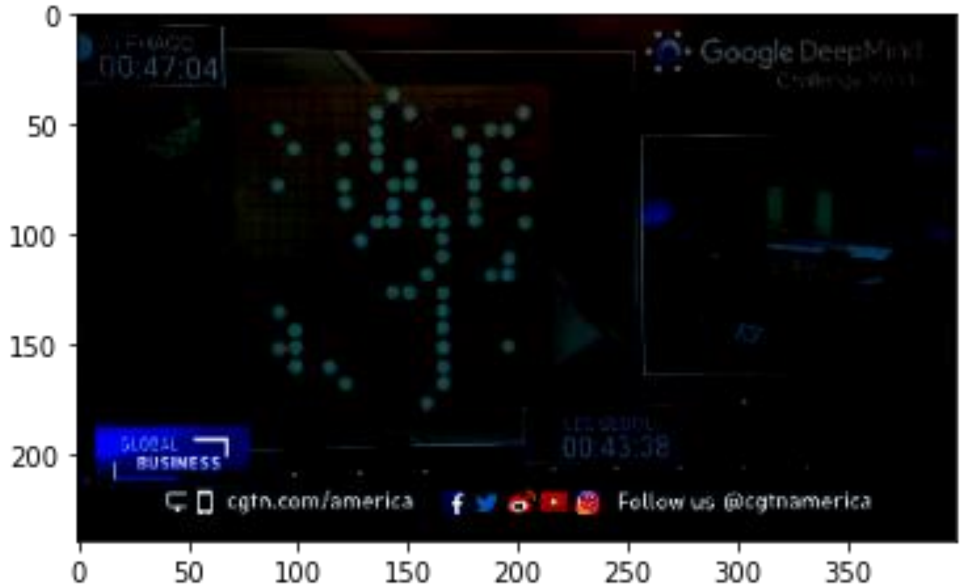

```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

#### 4.1.11 --------- GROUP: 13 ---------

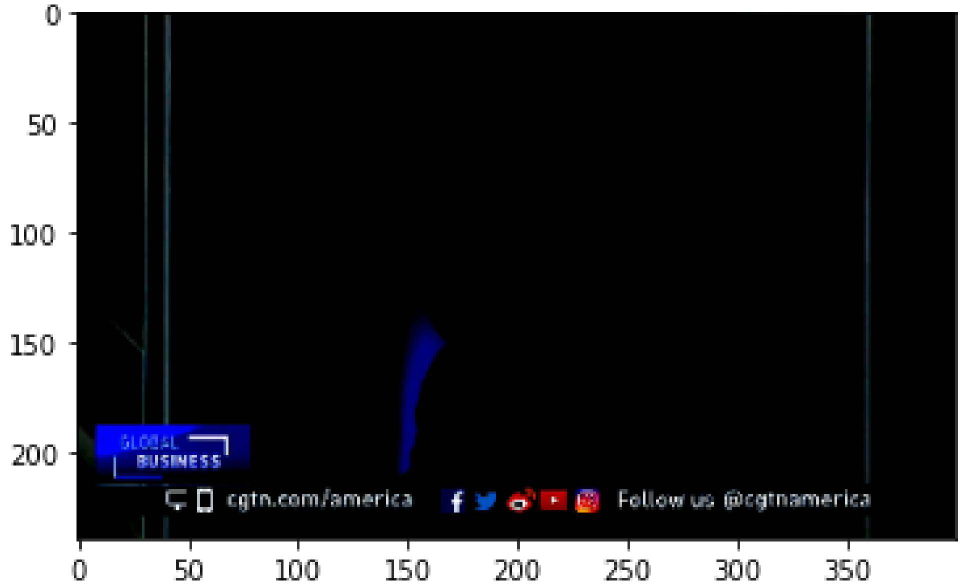



```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

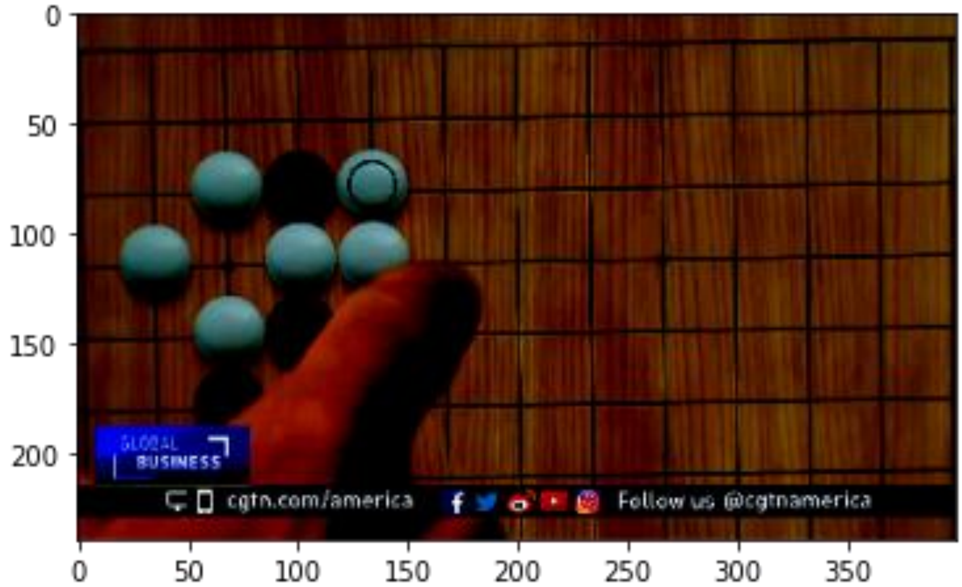

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



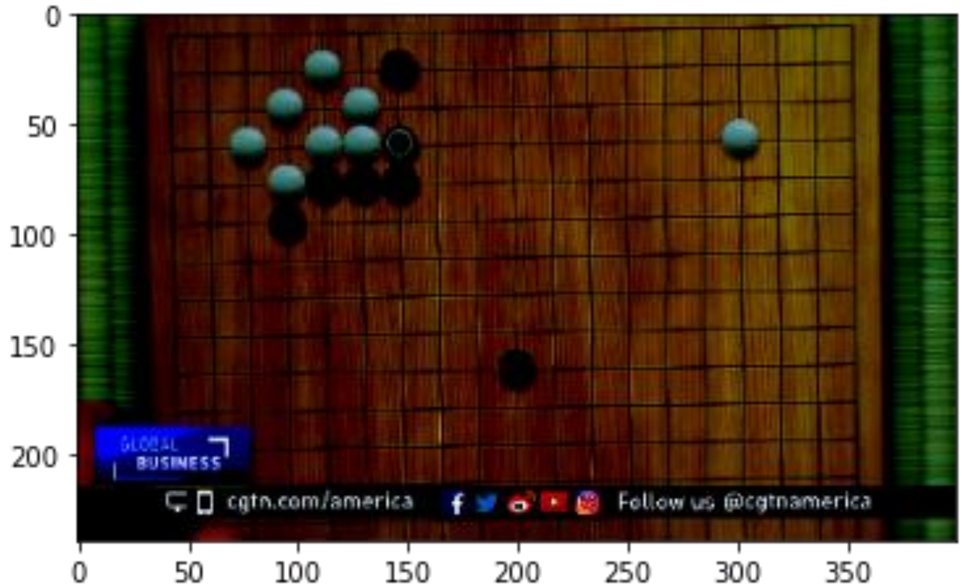

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



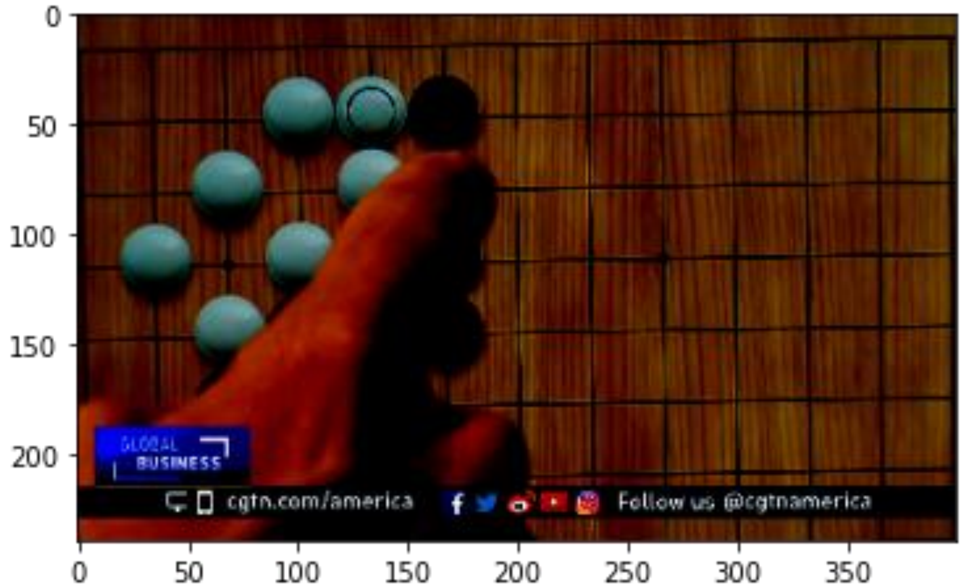

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).




Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).




```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

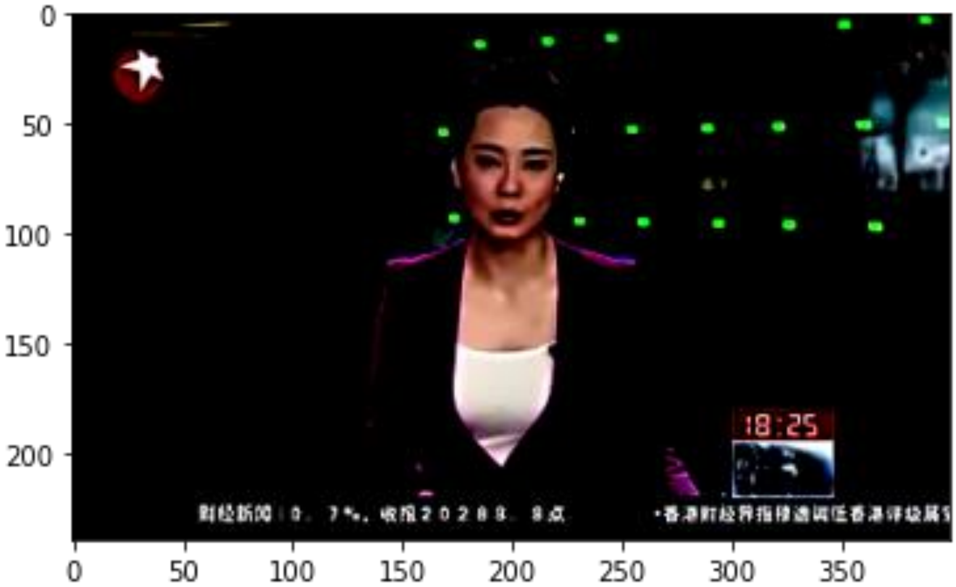



```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

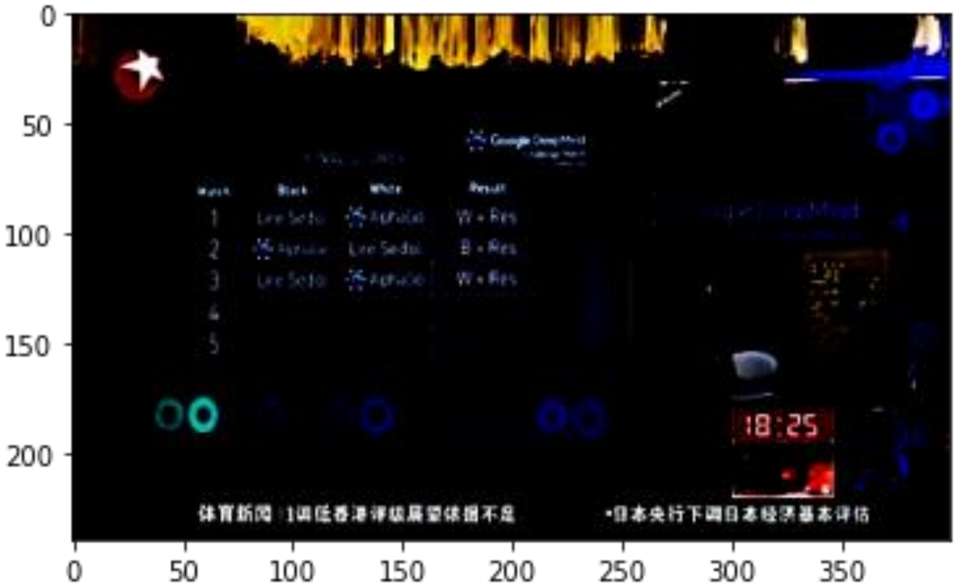
### 4.1.12 --------- GROUP: 9 ---------


```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

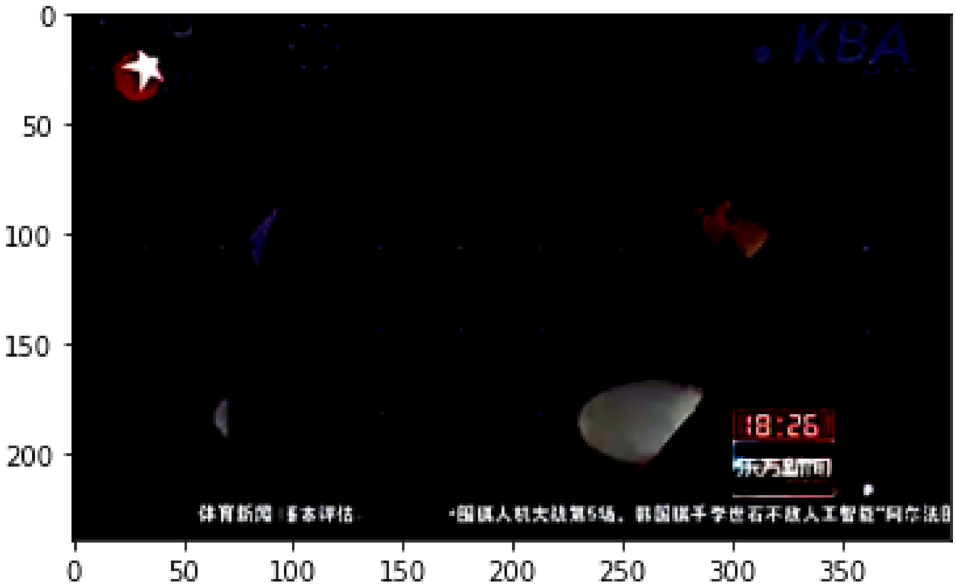

```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```



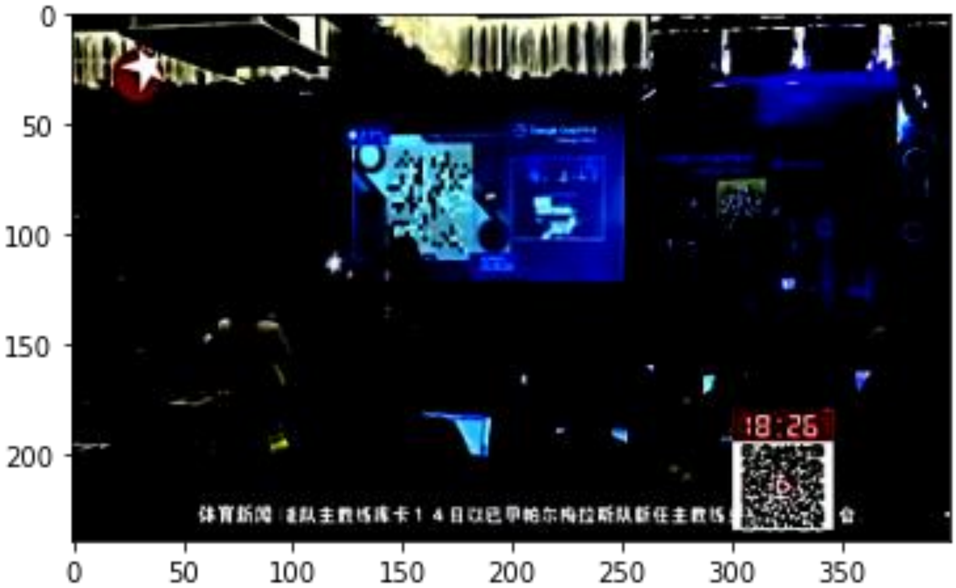

Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).



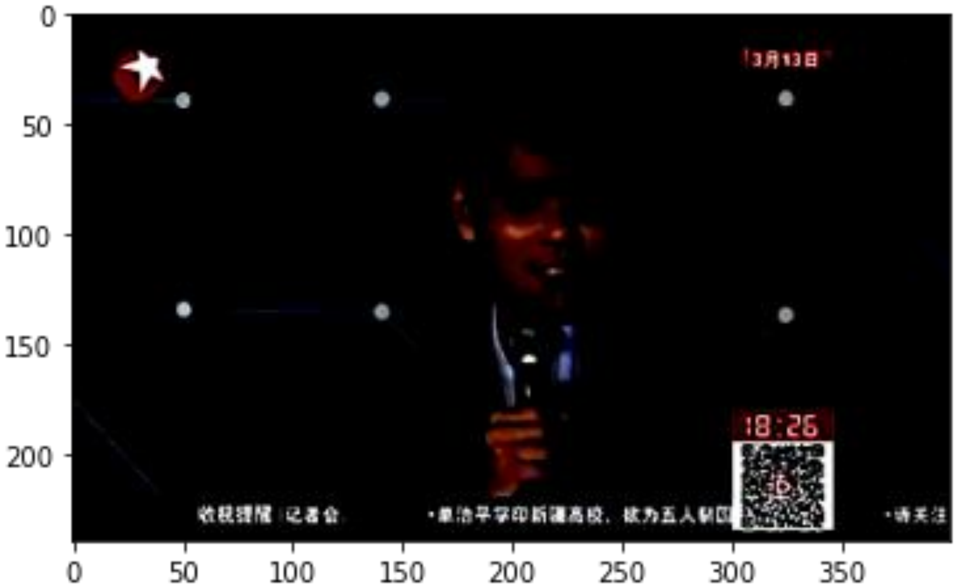

Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).




```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

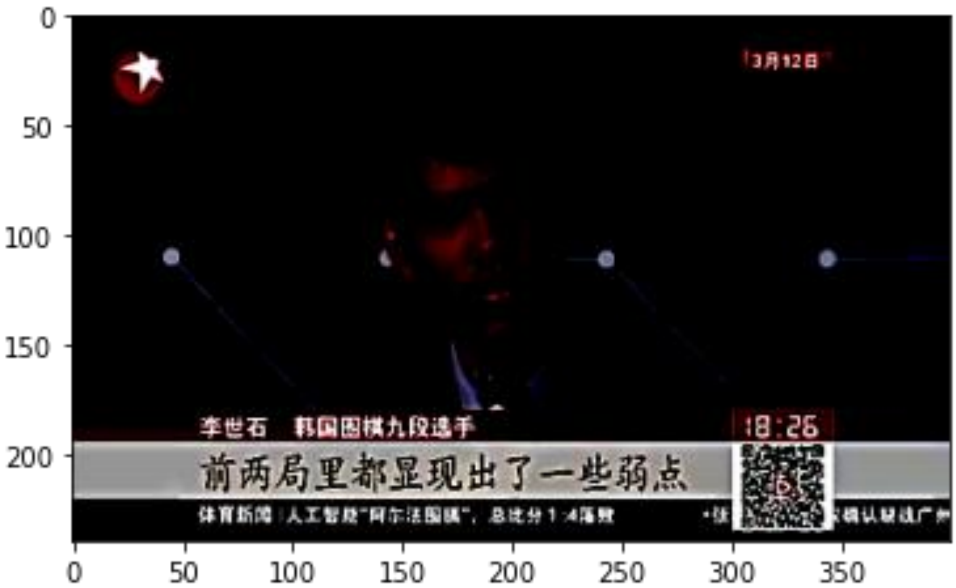



```
Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).
```

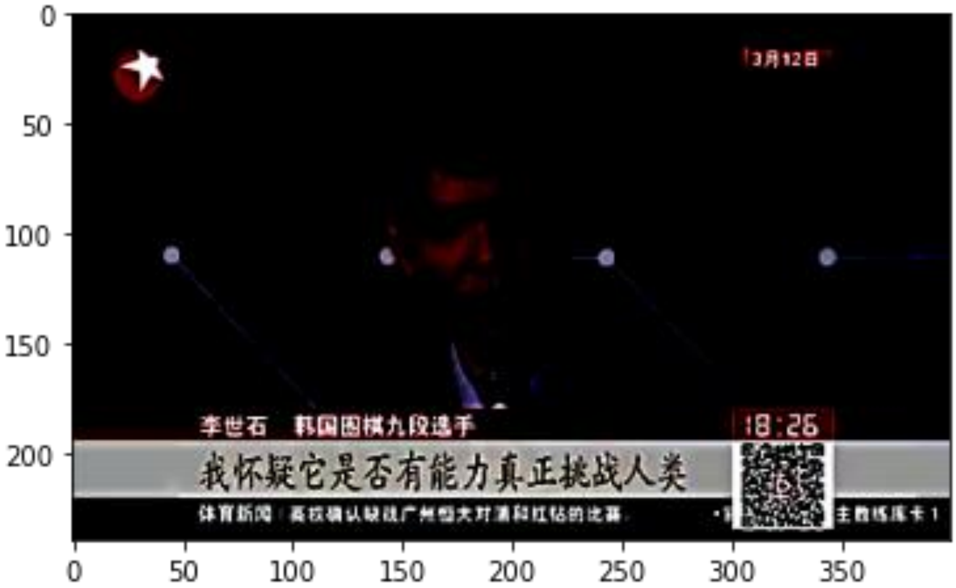



Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).

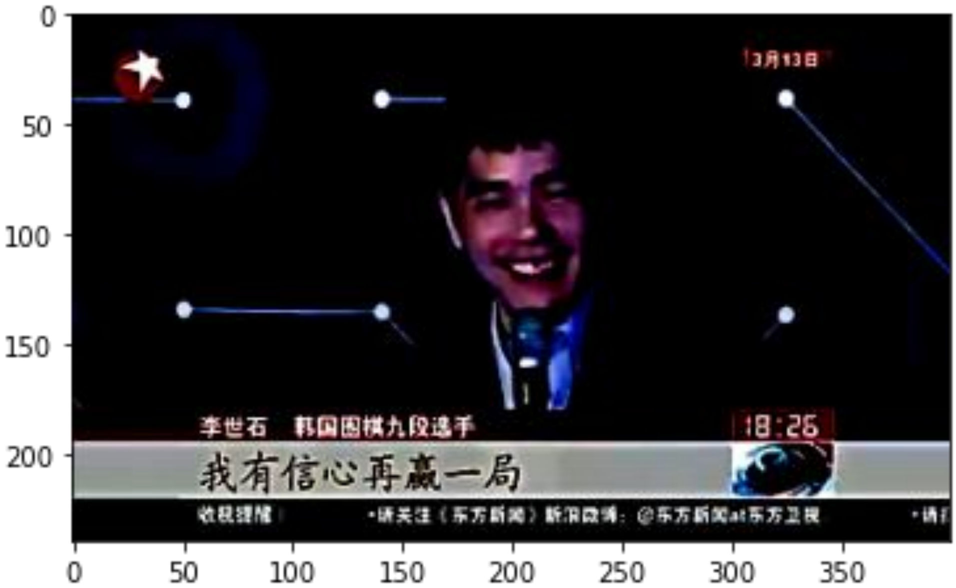
--------- GROUP: 1 ---------




Clipping input data to the valid range for imshow with RGB da
ta ([0..1] for floats or [0..255] for integers).

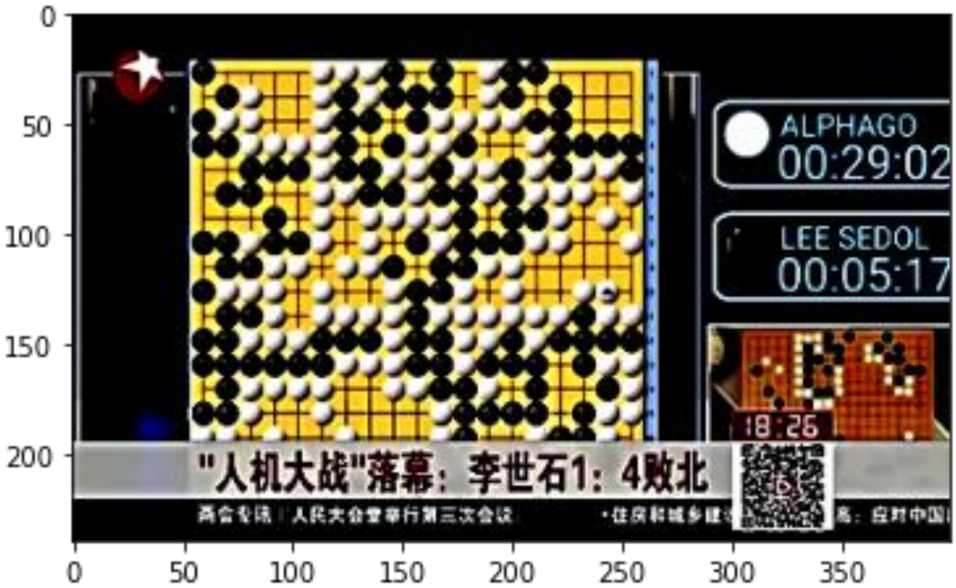



Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

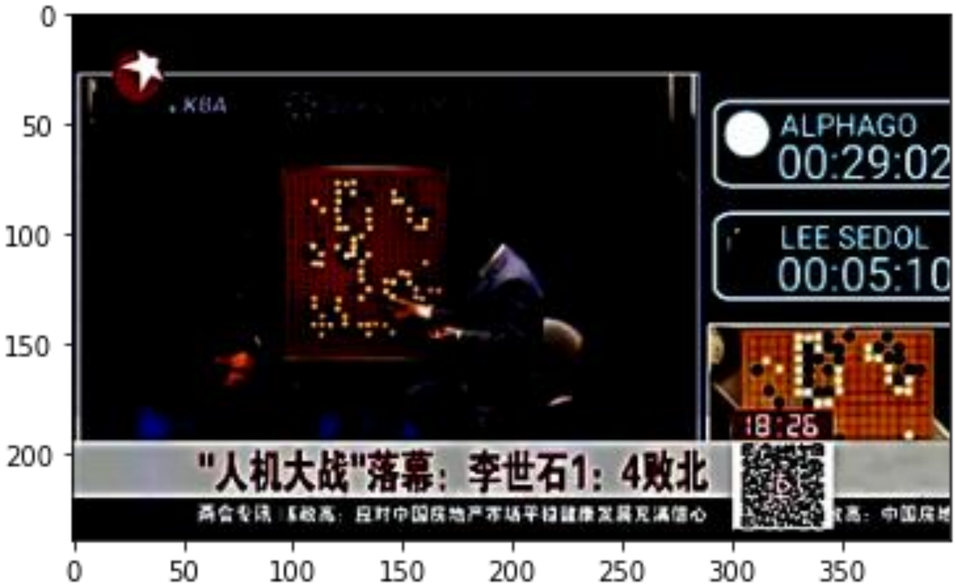


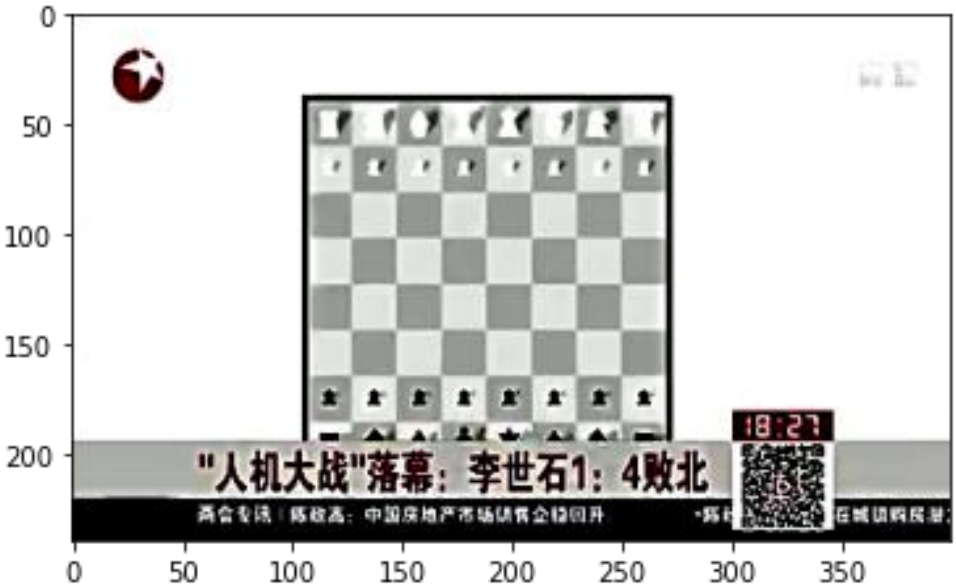

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).


Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

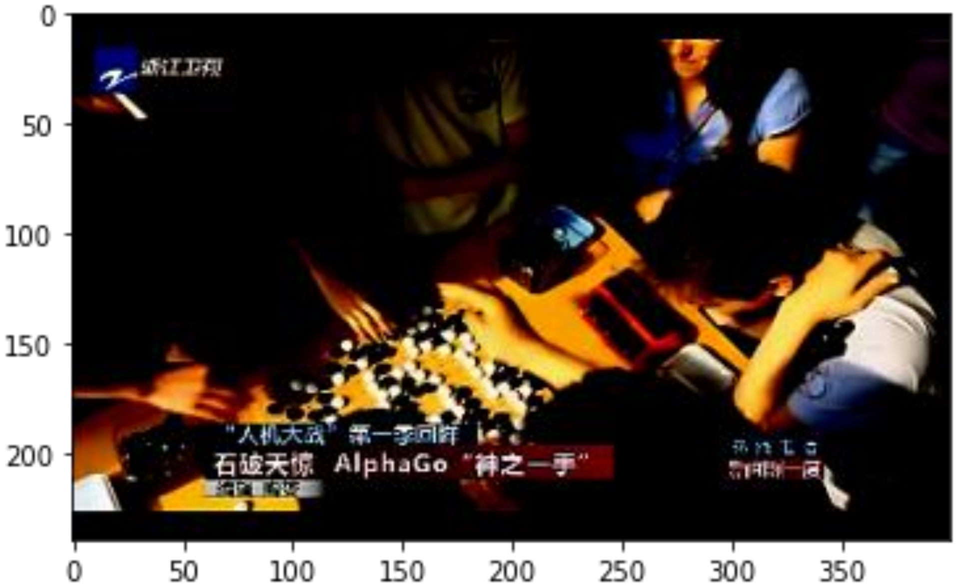


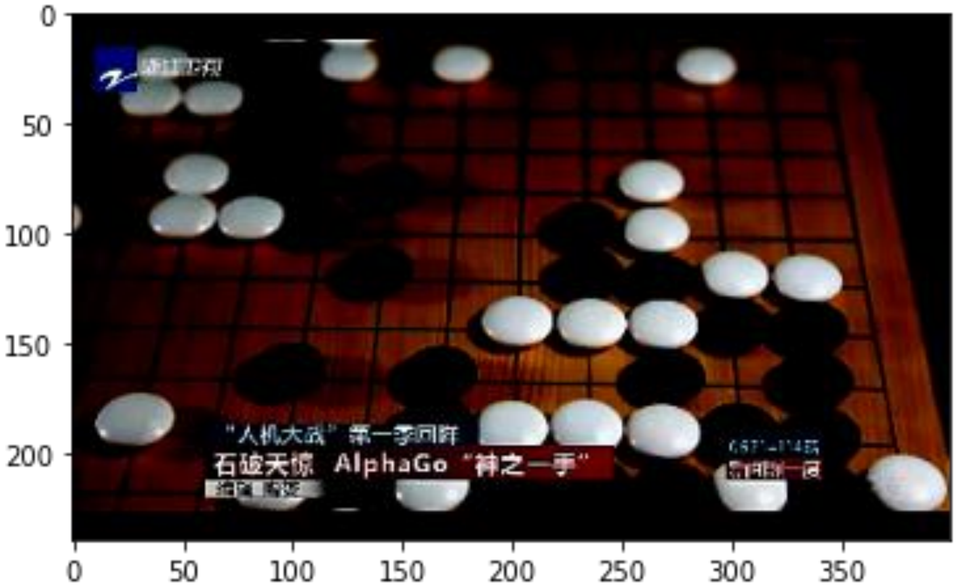

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

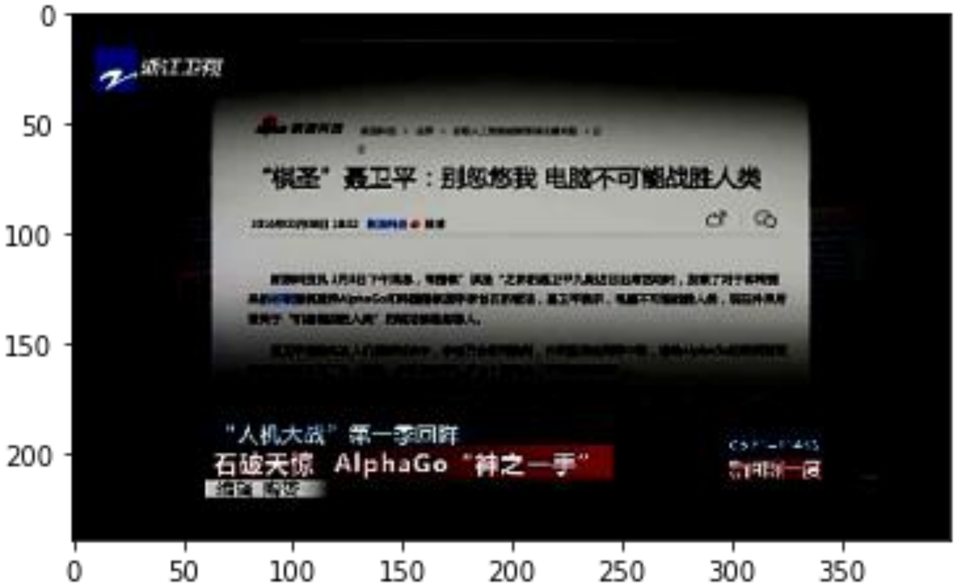

ALPHAGO
00:29:02
LEE SEDOL
00:05:10
18:26
"人机大战"落幕：李世石1：4败北
0
50
100
150
200
250
300
350

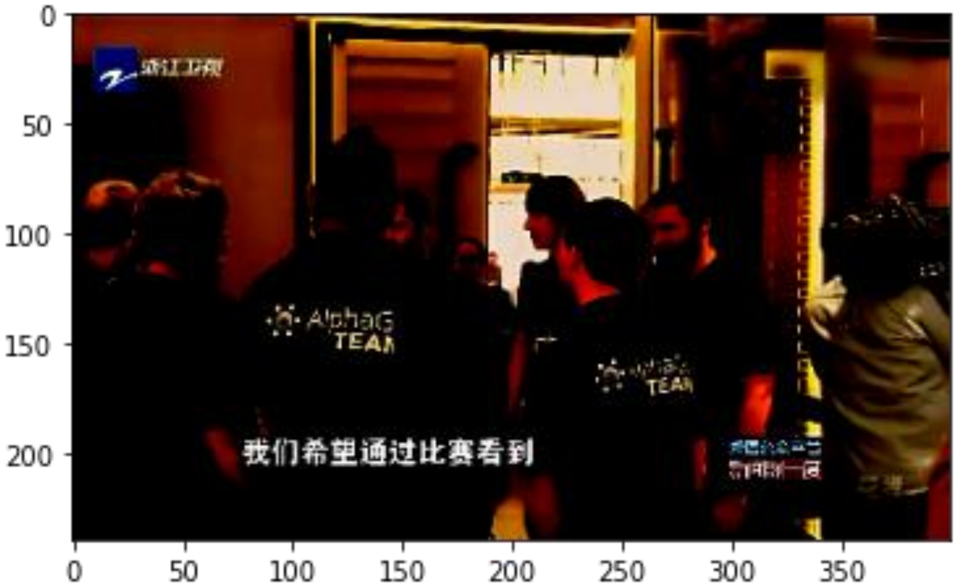

18:27
"人机大战"落幕：李世石1：4败北
0
50
100
150
200
250
300
350

4.1.13 --------- GROUP: 2 ---------


0
50
100
150
200
“人机大战”第一季回眸
石破天惊 AlphaGo“神之一手”
0
50
100
150
200
250
300
350

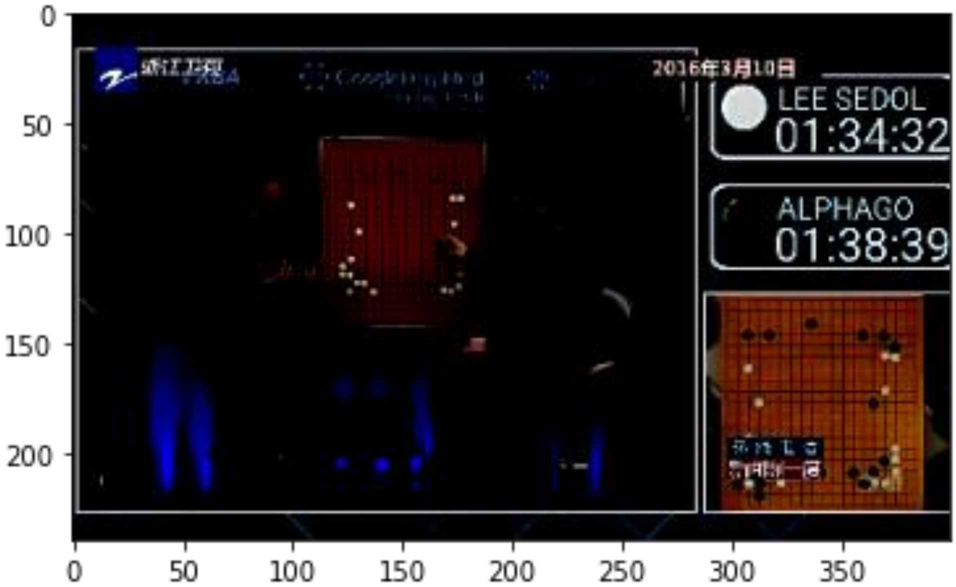

0
50
100
150
200
“棋圣”聂卫平：别忽悠我 电脑不可能战胜人类
“人机大战”第一季回眸
石破天惊 AlphaGo“神之一手”
0
50
100
150
200
250
300
350

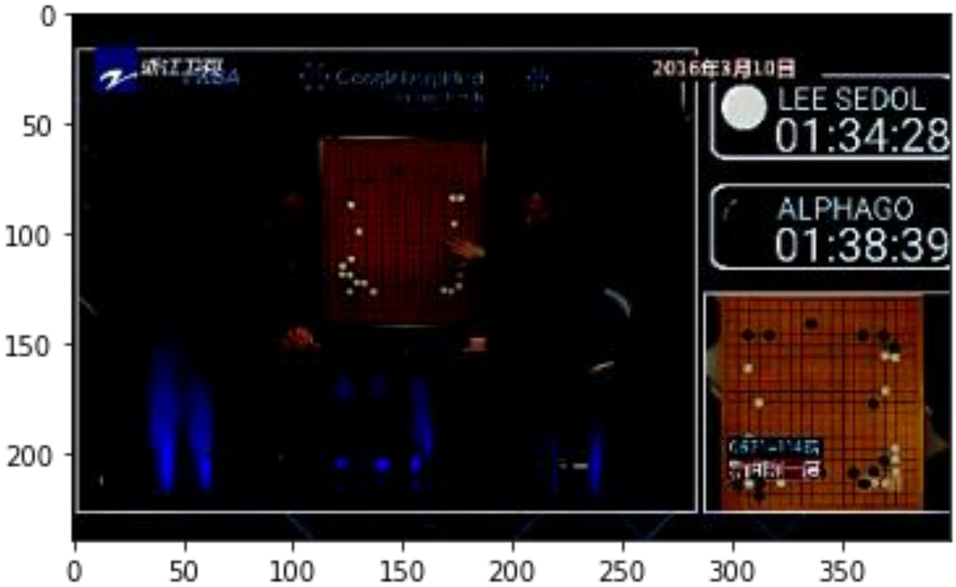

0
50
100
150
200
0
50
100
150
200
250
300
350
AlphaG
TEAM
我们希望通过比赛看到

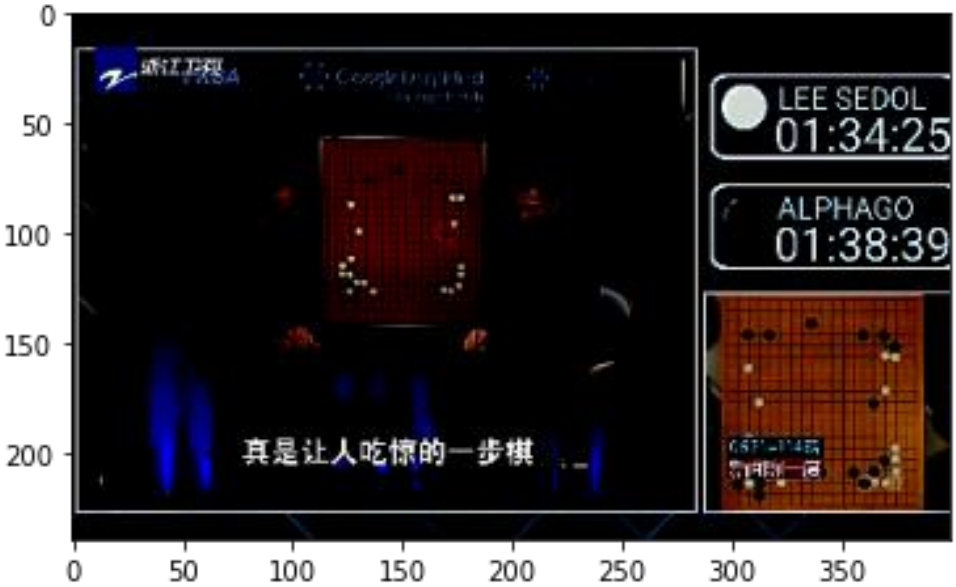

0
50
100
150
200
0
50
100
150
200
250
300
350
AlphaG
TEAM
它是否还有什么我们尚不知道的弱点

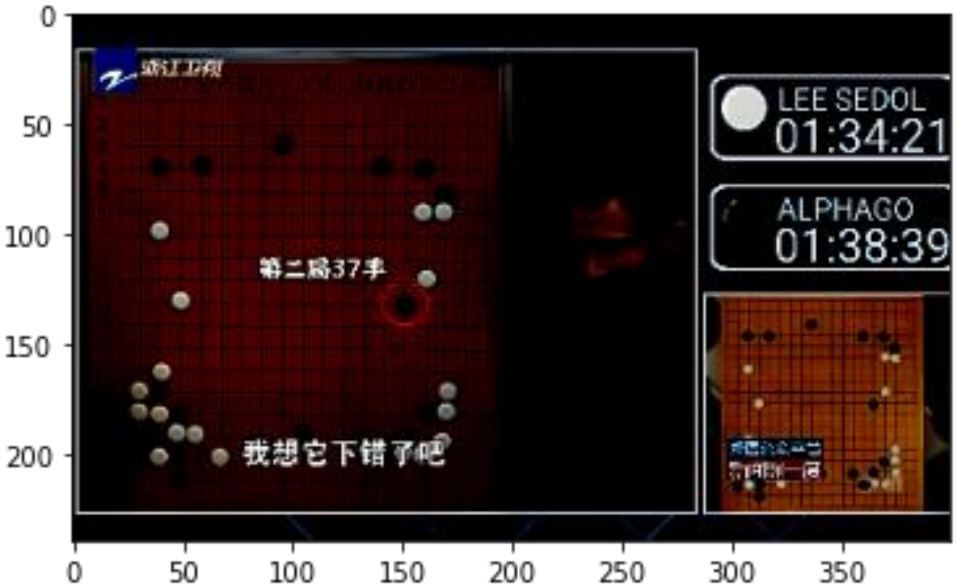

0
50
100
150
200
DeepMind CEO
有哪些事情我们的系统尚无法做到
0 50 100 150 200 250 300 350

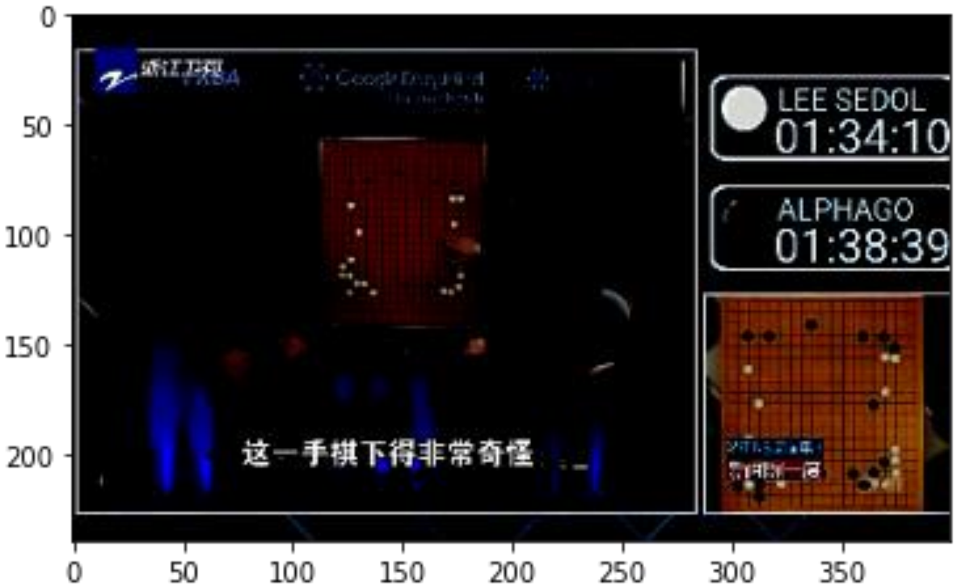
4.1.14 --------- GROUP: 4 ---------

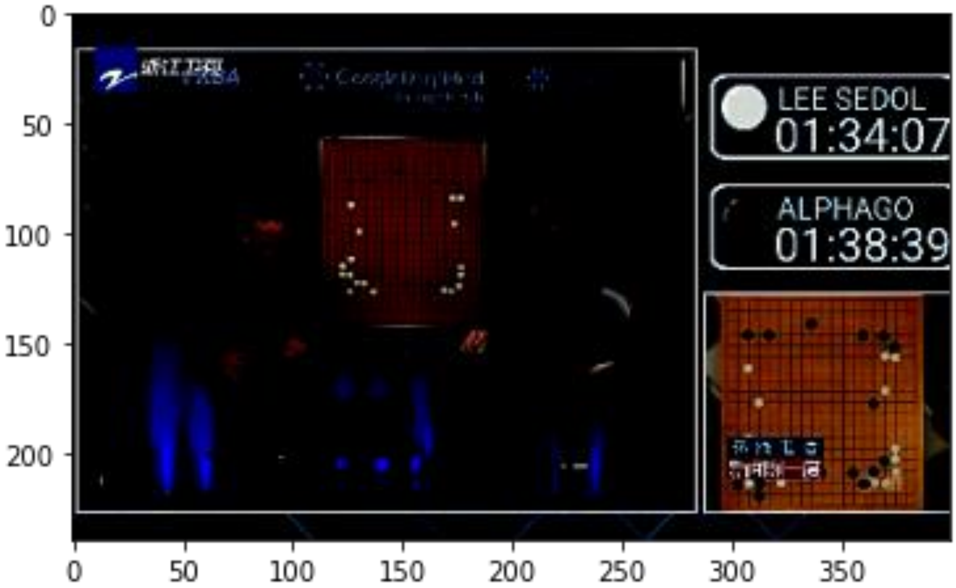

0
50
100
150
200
2016年3月10日
LEE SEDOL
01:34:32
ALPHAGO
01:38:39
0 50 100 150 200 250 300 350

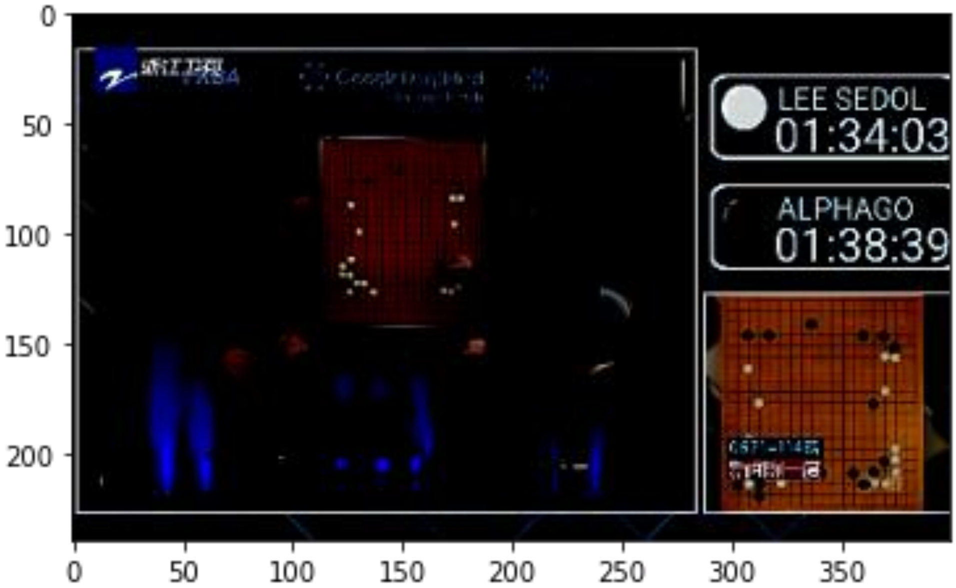

2016年3月10日
LEE SEDOL
01:34:28
ALPHAGO
01:38:39
0
50
100
150
200
250
300
350


LEE SEDOL
01:34:25
ALPHAGO
01:38:39
真是让人吃惊的一步棋
0
50
100
150
200
250
300
350


LEE SEDOL
01:34:21
ALPHAGO
01:38:39
第二局37手
我想它下错了吧
0
50
100
150
200
250
300
350


LEE SEDOL
01:34:10
ALPHAGO
01:38:39
这一手棋下得非常奇怪
0
50
100
150
200
250
300
350


LEE SEDOL
01:34:07
ALPHAGO
01:38:39
0
50
100
150
200
250
300
350


LEE SEDOL
01:34:03
ALPHAGO
01:38:39
0
50
100
150
200
250
300
350