人脸属性识别（表情检测）

——模型的应用

4 属性识别

⑤模型应用

emotion_detector.py

```
# import the necessary packages
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import numpy as np
import argparse
import imutils
import cv2

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-c", "--cascade", required=True,
     help="path to where the face cascade resides")
ap.add_argument("-m", "--model", required=True,
     help="path to pre-trained emotion detector CNN")
ap.add_argument("-v", "--video",
     help="path to the (optional) video file")
args = vars(ap.parse_args())
```

4 属性识别

⑤模型应用

从那里，我们加载人脸检测级联，情感检测CNN并初始化

我们的CNN可以预测的情感标签列表如23行所示：

```
19  # load the face detector cascade, emotion detection CNN, then define
20  # the list of emotion labels
21  detector = cv2.CascadeClassifier(args["cascade"])
22  model = load_model(args["model"])
23  EMOTIONS = ["angry", "scared", "happy", "sad", "surprised",
24      "neutral"]
```

4 属性识别

⑤模型应用

cv2.VideoCapture对象实例化：

（1）访问网络摄像头或（2）从视频文件读取：

```
26  # if a video path was not supplied, grab the reference to the webcam
27  if not args.get("video", False):
28        camera = cv2.VideoCapture(0)
29
30  # otherwise, load the video
31  else:
32        camera = cv2.VideoCapture(args["video"])
```

4 属性识别

⑤模型应用

我们现在准备从视频指针开始在帧上循环：

第37行从视频流中读取下一帧，如果未抓取框架（即设置为False），如果我们正在从视频流中读取帧，已经到达文件末尾时，因此我们应该从循环中中断（第41和42行）。

```
34  # keep looping
35  while True:
36      # grab the current frame
37      (grabbed, frame) = camera.read()
38
39      # if we are viewing a video and we did not grab a
40      # frame, then we have reached the end of the video
41      if args.get("video") and not grabbed:
42          break
```

4 属性识别

⑤模型应用

我们初始化一个空的NumPy画布（第50行），其宽度为300px，高度为200px。我们将使用画布绘制CNN预测的概率分布，使我们能够可视化情感的概率和混合性。然后，第55-57行使用OpenCV的预先训练的Haar级联来检测帧中的人脸。

```
44        # resize the frame and convert it to grayscale
45        frame = imutils.resize(frame, width=300)
46        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
47
48        # initialize the canvas for the visualization, then clone
49        # the frame so we can draw on it
50        canvas = np.zeros((220, 300, 3), dtype="uint8")
51        frameClone = frame.copy()
52
53        # detect faces in the input frame, then clone the frame so that
54        # we can draw on it
55        rects = detector.detectMultiScale(gray, scaleFactor=1.1,
56             minNeighbors=5, minSize=(30, 30),
57             flags=cv2.CASCADE_SCALE_IMAGE)
```

4 属性识别

⑤模型应用

第60行确保在帧中至少检测到一张脸。假设至少有检测到一张脸后，我们根据边界框的大小对边界框列表rect进行排序，在列表的最前面放大脸（第62和64行）。

我们通过将ROI调整为固定的48×48像素(EmotionVGGNet所需的输入大小)来预处理ROI。

从那里，我们将ROI转换为浮点数据 类型，将其缩放到范围[0，1]，并将其转换为与Keras兼容的数组（第69-72行）。

```
59        # ensure at least one face was found before continuing
60        if len(rects) > 0:
61            # determine the largest face area
62            rect = sorted(rects, reverse=True,
63                key=lambda x: (x[2] - x[0]) * (x[3] - x[1]))[0]
64            (fX, fY, fW, fH) = rect
65
66            # extract the face ROI from the image, then pre-process
67            # it for the network
68            roi = gray[fY:fY + fH, fX:fX + fW]
69            roi = cv2.resize(roi, (48, 48))
70            roi = roi.astype("float") / 255.0
71            roi = img_to_array(roi)
72            roi = np.expand_dims(roi, axis=0)
```

4 属性识别

⑤模型应用

第76行调用模型的预测方法，该方法返回预测的类标签概率。因此，标签是关联概率最大的标签（第77行）。但是，由于人的面部表情通常是多种情绪的结合，因此需要查看标签的概率分布。

在80行上遍历标签和相关概率。86行和87行绘制条形图其中每个条形宽度与预测的类别标签概率成比例。然后画88-90行画布上标签的名称。

```
74          # make a prediction on the ROI, then lookup the class
75          # label
76          preds = model.predict(roi)[0]
77          label = EMOTIONS[preds.argmax()]
78
79          # loop over the labels + probabilities and draw them
80          for (i, (emotion, prob)) in enumerate(zip(EMOTIONS, preds)):
81               # construct the label text
82               text = "{}: {:.2f}%".format(emotion, prob * 100)
83
84               # draw the label + probability bar on the canvas
85               w = int(prob * 300)
86               cv2.rectangle(canvas, (5, (i * 35) + 5),
87                    (w, (i * 35) + 35), (0, 0, 255), -1)
88               cv2.putText(canvas, text, (10, (i * 35) + 23),
89                    cv2.FONT_HERSHEY_SIMPLEX, 0.45,
90                    (255, 255, 255), 2)
```

4 属性识别

⑤模型应用

```
92              # draw the label on the frame
93              cv2.putText(frameClone, label, (fX, fY - 10),
94                    cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)
95              cv2.rectangle(frameClone, (fX, fY), (fX + fW, fY + fH),
96                    (0, 0, 255), 2)
98          # show our classifications + probabilities
99          cv2.imshow("Face", frameClone)
100         cv2.imshow("Probabilities", canvas)
101
102         # if the 'q' key is pressed, stop the loop
103         if cv2.waitKey(1) & 0xFF == ord("q"):
104               break
105
106 # cleanup the camera and close any open windows
107 camera.release()
108 cv2.destroyAllWindows()
```

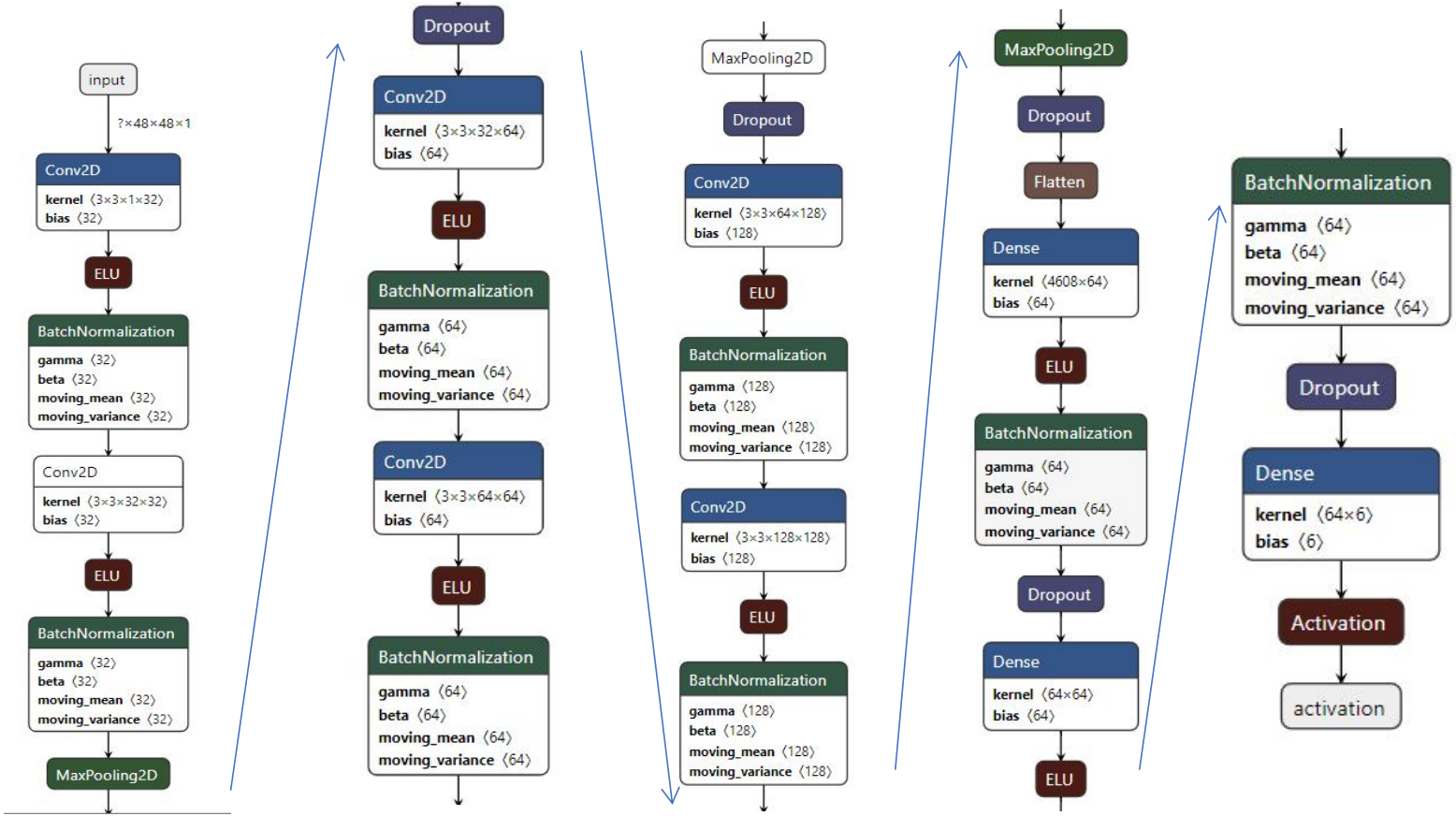
4 属性识别

⑤模型应用

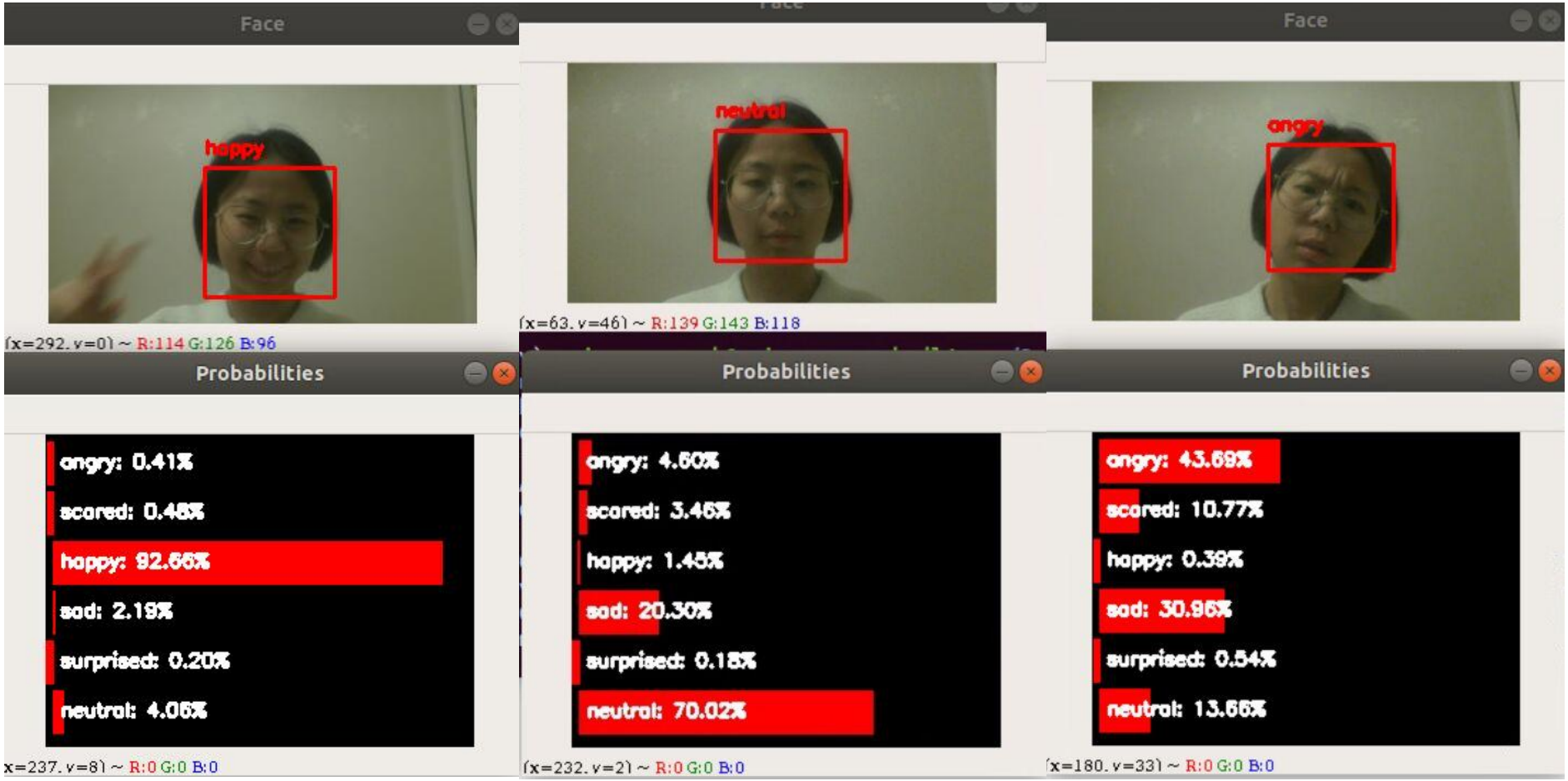
```
$ python emotion_detector.py --cascade haarcascade_frontalface_default.xml --model checkpoints/epoch_75.hdf5
```

```
$ python emotion_detector.py --cascade haarcascade_frontalface_default.xml \
--model checkpoints/epoch_75.hdf5 --video path/to/your/video.mp4
```

4 属性识别


input
?×48×48×1
Conv2D
kernel 〈3×3×1×32〉
bias 〈32〉
ELU
BatchNormalization
gamma 〈32〉
beta 〈32〉
moving_mean 〈32〉
moving_variance 〈32〉
Conv2D
kernel 〈3×3×32×32〉
bias 〈32〉
ELU
BatchNormalization
gamma 〈32〉
beta 〈32〉
moving_mean 〈32〉
moving_variance 〈32〉
MaxPooling2D
Dropout
Conv2D
kernel 〈3×3×32×64〉
bias 〈64〉
ELU
BatchNormalization
gamma 〈64〉
beta 〈64〉
moving_mean 〈64〉
moving_variance 〈64〉
Conv2D
kernel 〈3×3×64×64〉
bias 〈64〉
ELU
BatchNormalization
gamma 〈64〉
beta 〈64〉
moving_mean 〈64〉
moving_variance 〈64〉
MaxPooling2D
Dropout
Conv2D
kernel 〈3×3×64×128〉
bias 〈128〉
ELU
BatchNormalization
gamma 〈128〉
beta 〈128〉
moving_mean 〈128〉
moving_variance 〈128〉
Conv2D
kernel 〈3×3×128×128〉
bias 〈128〉
ELU
BatchNormalization
gamma 〈128〉
beta 〈128〉
moving_mean 〈128〉
moving_variance 〈128〉
MaxPooling2D
Dropout
Flatten
Dense
kernel 〈4608×64〉
bias 〈64〉
ELU
BatchNormalization
gamma 〈64〉
beta 〈64〉
moving_mean 〈64〉
moving_variance 〈64〉
Dropout
Dense
kernel 〈64×64〉
bias 〈64〉
ELU
BatchNormalization
gamma 〈64〉
beta 〈64〉
moving_mean 〈64〉
moving_variance 〈64〉
Dropout
Dense
kernel 〈64×6〉
bias 〈6〉
Activation
activation

4 属性识别


Face
happy
(x=292, y=0) ~ R:114 G:126 B:96
Probabilities
angry: 0.41%
scared: 0.48%
happy: 92.66%
sad: 2.19%
surprised: 0.20%
neutral: 4.06%
(x=237, y=8) ~ R:0 G:0 B:0
neutral
(x=63, y=46) ~ R:139 G:143 B:118
Probabilities
angry: 4.60%
scared: 3.46%
happy: 1.45%
sad: 20.30%
surprised: 0.18%
neutral: 70.02%
(x=232, y=2) ~ R:0 G:0 B:0
Face
angry
Probabilities
angry: 43.69%
scared: 10.77%
happy: 0.39%
sad: 30.96%
surprised: 0.54%
neutral: 13.66%
(x=180, y=33) ~ R:0 G:0 B:0