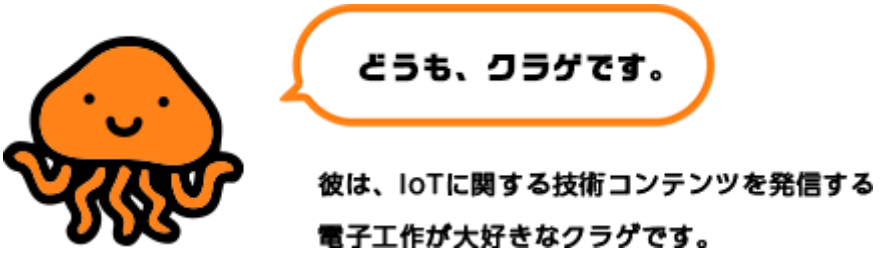
2019/08/09

リアルタイム感情分析アプリ



どうも、ディープなクラゲです。

「OpenVINO™ でゼロから学ぶディープラーニング推論」シリーズの９回目記事です。

このシリーズは、ディープラーニング概要、OpenVINO™ツールキット、Neural Compute Stick、RaspberryPiの使い方、Pythonプログラミングをゼロから徹底的に学び、成果としてディープラーニング推論アプリケーションが理解して作れるようになることを目指します。

第９回目は「顔検出」と「感情分類」を組合せ、リアルタイムにグラフ表示や画像表示を行うアプリケーションを完成させます。

【 目次 】

- 顔検出 + 感情分類
- 棒グラフの表示
- 画像オーバーレイ

顔検出 + 感情分類

前回のリアルタイム顔検出に、感情分類のコードを加えることで出来ます。

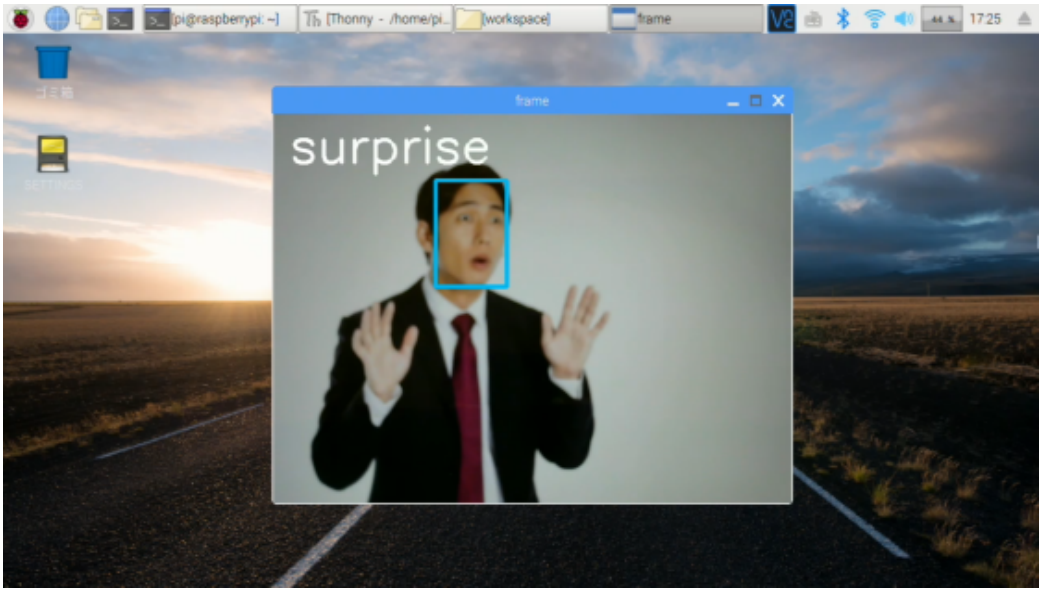
先に実行できる全体コードを示します。

```
import cv2
import numpy as np

# モジュール読み込み
import sys
sys.path.append('/opt/intel/openvino/python/python3.5/armv7l')
from openvino.inference_engine import IENetwork, IEPlugin

# ターゲットデバイスの指定
plugin = IEPlugin(device="MYRIAD")

# モデルの読み込み（顔検出）
net = IENetwork(model='FP16/face-detection-retail-0004.xml', weights='FP
exec_net = plugin.load(network=net)

# モデルの読み込み（感情分類）
net_emotion = IENetwork(model='FP16/emotions-recognition-retail-0003.x
exec_net_emotion = plugin.load(network=net_emotion)

# カメラ準備
cap = cv2.VideoCapture(0)

# メインループ
while True:
    ret, frame = cap.read()

    # Reload on error
    if ret == False:

        continue

    # 入力データフォーマットへ変換
    img = cv2.resize(frame, (300, 300))  # サイズ変更
    img = img.transpose((2, 0, 1))    # HWC > CHW
    img = np.expand_dims(img, axis=0) # 次元合せ

    # 推論実行
    out = exec_net.infer(inputs={'data': img})
```

```
    # 出力から必要なデータのみ取り出し
    out = out['detection_out']
    out = np.squeeze(out) #サイズ1の次元を全て削除

    # 検出されたすべての顔領域に対して１つずつ処理
    for detection in out:
        # conf値の取得
        confidence = float(detection[2])

        # バウンディングボックス座標を入力画像のスケールに変換
        xmin = int(detection[3] * frame.shape[1])
        ymin = int(detection[4] * frame.shape[0])
        xmax = int(detection[5] * frame.shape[1])
        ymax = int(detection[6] * frame.shape[0])

        # conf値が0.5より大きい場合のみ感情推論とバウンディングボックス表示
        if confidence > 0.5:
            # 顔検出領域はカメラ範囲内に補正する。特にminは補正しないとエラー
            if xmin < 0:
                xmin = 0
            if ymin < 0:
                ymin = 0
            if xmax > frame.shape[1]:
                xmax = frame.shape[1]
            if ymax > frame.shape[0]:
                ymax = frame.shape[0]

            # 顔領域のみ切り出し
            frame_face = frame[ ymin:ymax, xmin:xmax ]

            # 入力データフォーマットへ変換
            img = cv2.resize(frame_face, (64, 64))  # サイズ変更
            img = img.transpose((2, 0, 1))    # HWC > CHW
            img = np.expand_dims(img, axis=0) # 次元合せ

            # 推論実行
            out = exec_net_emotion.infer(inputs={'data': img})

            # 出力から必要なデータのみ取り出し
            out = out['prob_emotion']
            out = np.squeeze(out) #不要な次元の削減

            # 出力値が最大のインデックスを得る
            index_max = np.argmax(out)

            # 各感情の文字列をリスト化
            list_emotion = ['neutral', 'happy', 'sad', 'surprise', 'anger']

            # 文字列描画
            cv2.putText(frame, list_emotion[index_max], (20, 60), cv2.FONT_H

            # バウンディングボックス表示
            cv2.rectangle(frame, (xmin, ymin), (xmax, ymax), color=(240, 180

            # 1つの顔で終了
```

```
            # １つの顔で終了
            break

    # 画像表示
    cv2.imshow('frame', frame)

    # 何らかのキーが押されたら終了
    key = cv2.waitKey(1)
    if key != -1:
        break

# 終了処理
cap.release()
cv2.destroyAllWindows()
```

実行の静止画です。※実際はリアルタイムカメラ映像です



いくつかポイントがあります。

モデル読み込みの追加

顔検出では`net`, `exec_net`という変数を用いていますので、感情分類では`net_emotion`, `exec_net_emotion`という名前にしています。

```
# モデルの読み込み（感情分類）
net_emotion = IENetwork(model='FP16/emotions-recognition-retail-0003.x
exec_net_emotion = plugin.load(network=net_emotion)
```

顔領域の切り出し

これは「NumPyを学ぶ」で習った「スライス」を活用することで簡単にできます

```
# 顔領域のみ切り出し
frame_face = frame[ ymin:ymax, xmin:xmax ]
```

ただし、顔検出はカメラの枠から顔がはみ出た場合に、負の値になることがあり、`cv2.resize`時にエラーになってしまいます。そこで、顔領域切り出し前に、顔検出領域をカメラ範囲内に補正しています

```
# 顔検出領域はカメラ範囲内に補正する。特にminは補正しないとエラーになる
if xmin < 0:
  xmin = 0
if ymin < 0:
  ymin = 0
if xmax > frame.shape[1]:
  xmax = frame.shape[1]
if ymax > frame.shape[0]:
  ymax = frame.shape[0]
```

感情推論結果の表示

感情の文字列は、顔検出領域`frame_face`ではなく、カメラフレーム`framae`に対して描きます
文字の大きさや位置などは少し変えています。

```
# 文字列描画
cv2.putText(frame, list_emotion[index_max], (20, 60), cv2.FONT_HERSHEY
```

また、処理を軽くするために、感情推論は1フレーム辺り１つの顔のみとします

```
#  1 つの顔で終了
break
```

棒グラフの表示

これまではインデックスが最大の感情のみを表示していましたが、その他の感情の値も棒グラフを使って見える化したいと思います。
先に実行できる全体コードを示します。

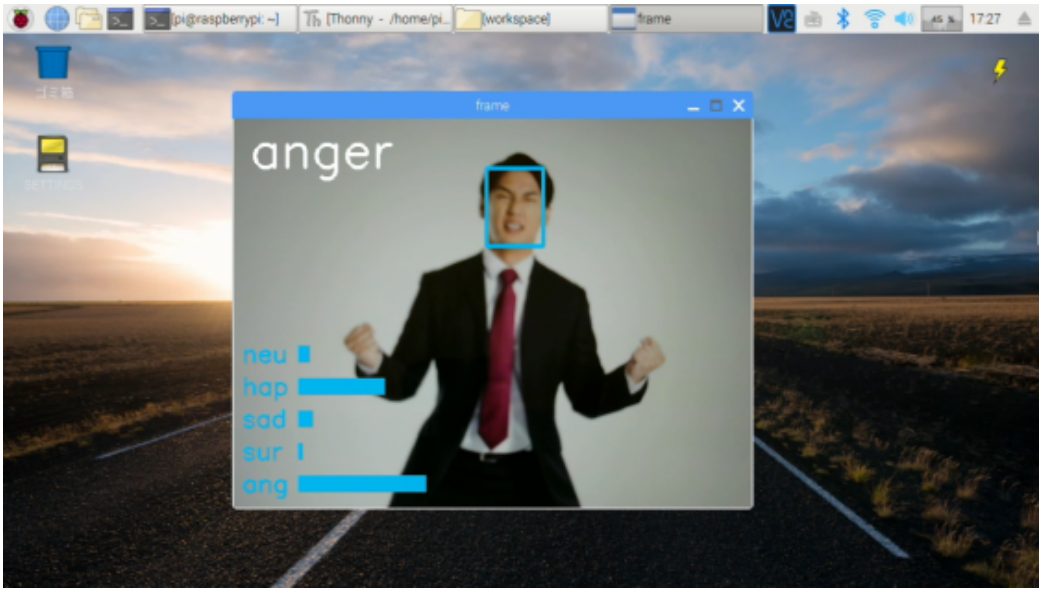
```
import cv2
import numpy as np
```

```
# モジュール読み込み
import sys
sys.path.append('/opt/intel/openvino/python/python3.5/armv7l')
from openvino.inference_engine import IENetwork, IEPlugin

# ターゲットデバイスの指定
plugin = IEPlugin(device="MYRIAD")


# モデルの読み込み（顔検出）
net = IENetwork(model='FP16/face-detection-retail-0004.xml', weights='FP
exec_net = plugin.load(network=net)

# モデルの読み込み（感情分類）
net_emotion = IENetwork(model='FP16/emotions-recognition-retail-0003.x
exec_net_emotion = plugin.load(network=net_emotion)

# カメラ準備
cap = cv2.VideoCapture(0)

# メインループ
while True:
    ret, frame = cap.read()

    # Reload on error
    if ret == False:
        continue

    # 入力データフォーマットへ変換
    img = cv2.resize(frame, (300, 300))   # サイズ変更
    img = img.transpose((2, 0, 1))    # HWC > CHW
    img = np.expand_dims(img, axis=0) # 次元合せ

    # 推論実行
    out = exec_net.infer(inputs={'data': img})

    # 出力から必要なデータのみ取り出し
    out = out['detection_out']
    out = np.squeeze(out) #サイズ1の次元を全て削除

    # 検出されたすべての顔領域に対して１つずつ処理
    for detection in out:
        # conf値の取得
        confidence = float(detection[2])

        # バウンディングボックス座標を入力画像のスケールに変換
        xmin = int(detection[3] * frame.shape[1])
        ymin = int(detection[4] * frame.shape[0])
        xmax = int(detection[5] * frame.shape[1])
        ymax = int(detection[6] * frame.shape[0])

        # conf値が0.5より大きい場合のみ感情推論とバウンディングボックス表示
        if confidence > 0.5:
            # 顔検出領域はカメラ範囲内に補正する。特にminは補正しないとエラー
            if xmin < 0:
```

```
            xmin = 0
        if ymin < 0:
            ymin = 0
        if xmax > frame.shape[1]:
            xmax = frame.shape[1]
        if ymax > frame.shape[0]:
            ymax = frame.shape[0]

        # 顔領域のみ切り出し
        frame_face = frame[ ymin:ymax, xmin:xmax ]

        # 入力データフォーマットへ変換
        img = cv2.resize(frame_face, (64, 64))  # サイズ変更
        img = img.transpose((2, 0, 1))  # HWC > CHW
        img = np.expand_dims(img, axis=0) # 次元合せ

        # 推論実行
        out = exec_net_emotion.infer(inputs={'data': img})

        # 出力から必要なデータのみ取り出し
        out = out['prob_emotion']
        out = np.squeeze(out) #不要な次元の削減

        # 出力値が最大のインデックスを得る
        index_max = np.argmax(out)

        # 各感情の文字列をリスト化
        list_emotion = ['neutral', 'happy', 'sad', 'surprise', 'anger']

        # 文字列描画
        cv2.putText(frame, list_emotion[index_max], (20, 60), cv2.FONT_H

        # バウンディングボックス表示
        cv2.rectangle(frame, (xmin, ymin), (xmax, ymax), color=(240, 180

        # 棒グラフ表示
        str_emotion = ['neu', 'hap', 'sad', 'sur', 'ang']
        text_x = 10
        text_y = frame.shape[0] - 180
        rect_x = 80
        rect_y = frame.shape[0] - 200
        for i in range(5):
            cv2.putText(frame, str_emotion[i], (text_x, text_y), cv2.FONT_H
            cv2.rectangle(frame, (rect_x, rect_y), (rect_x + int(300 * out[i]
            text_y = text_y + 40
            rect_y = rect_y + 40

        #  1 つの顔で終了
        break

    # 画像表示
    cv2.imshow('frame', frame)

    # 何らかのキーが押されたら終了
    key = cv2.waitKey(1)
```

```
    if key != -1:
        break

# 終了処理
cap.release()
cv2.destroyAllWindows()
```

実行の静止画です。※実際はリアルタイムカメラ映像です



解説ですが、先程のコードに以下の部分を追加しただけです。

ポイントは、テキスト位置や長方形の位置を `for ... in range( )` を活用して繰り返し表示している点です。最後に `40` を加えることにより、Y軸方向に40ピクセルの間隔でテキストと棒グラフが並ぶことになります。

```
# 棒グラフ表示
str_emotion = ['neu', 'hap', 'sad', 'sur', 'ang']
text_x = 10
text_y = frame.shape[0] - 180
rect_x = 80
rect_y = frame.shape[0] - 200
for i in range(5):
    cv2.putText(frame, str_emotion[i], (text_x, text_y), cv2.FONT_HERSHEY
    cv2.rectangle(frame, (rect_x, rect_y), (rect_x + int(300 * out[i]), rect_y
    text_y = text_y + 40
    rect_y = rect_y + 40
```

`out[i]` にはそれぞれの感情推論の値が `0` ～ `1.0` の数値で入っているので、適当に `300` を掛けて長方形の横幅として使うことで、リアルタイムな棒グラフが表現できています。

画像オーバーレイ

最後の仕上げとして、インデックスが最大である感情をテキスト表示からPNG顔アイコン画像に変えます。

画像の準備

５つの感情それぞれに対し、次のような顔アイコンを割り当てています。

クラゲは こちらのページから顔アイコン画像をダウンロードし、トリミングとサイズ変更（200×200ピクセルに縮小）を行いました。皆さんは他のお好きな画像を使ってもらってOKです。
`workspace`フォルダに`image`を作成し、その中に各画像を入れて下さい

クラスの準備

OpenCVには画像に画像を重ねるオーバーレイ関数がありません。そこで、クラスを活用することとしましょう。
「OpenCVで透過PNGファイルの重ね合わせ」を行うクラス
右上の`Download ZIP`ボタンを押します。`ダウンロード`フォルダに保存されているので、右クリックからファイルを展開します。`pngoverlay.py`を`workspace`フォルダへ移動します

全体コード

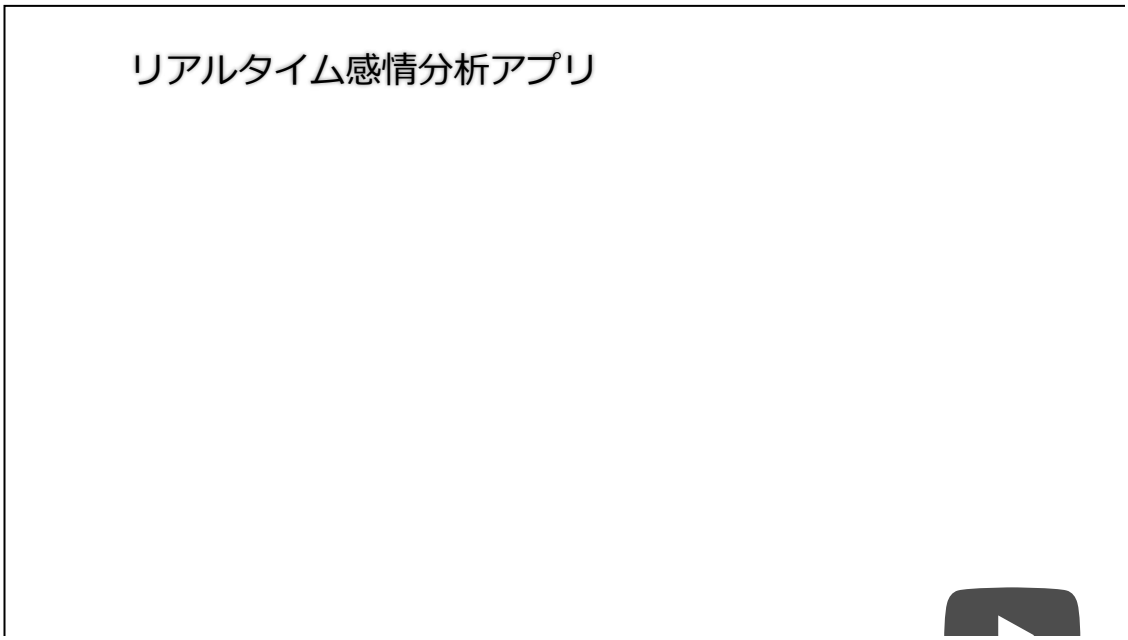
先に実行できる全体コードを示します。

```
import cv2
import numpy as np

# モジュール読み込み
import sys
sys.path.append('/opt/intel/openvino/python/python3.5/armv7l')
from openvino.inference_engine import IENetwork, IEPlugin

# pngoverlayクラス読み込み
from pngoverlay import PNGOverlay
```

```
# インスタンス生成
icon_neutral = PNGOverlay('image/icon_neutral.png')
icon_happy = PNGOverlay('image/icon_happy.png')
icon_sad = PNGOverlay('image/icon_sad.png')
icon_surprise = PNGOverlay('image/icon_surprise.png')
icon_anger = PNGOverlay('image/icon_anger.png')

# インスタンス変数をリストにまとめる
icon_emotion = [icon_neutral, icon_happy, icon_sad, icon_surprise, icon_ar

# ターゲットデバイスの指定
plugin = IEPlugin(device="MYRIAD")

# モデルの読み込み（顔検出）
net = IENetwork(model='FP16/face-detection-retail-0004.xml', weights='FP
exec_net = plugin.load(network=net)

# モデルの読み込み（感情分類）

net_emotion = IENetwork(model='FP16/emotions-recognition-retail-0003.x
exec_net_emotion = plugin.load(network=net_emotion)

# カメラ準備
cap = cv2.VideoCapture(0)

# メインループ
while True:
    ret, frame = cap.read()

    # Reload on error
    if ret == False:
        continue

    # 入力データフォーマットへ変換
    img = cv2.resize(frame, (300, 300))   # サイズ変更
    img = img.transpose((2, 0, 1))    # HWC > CHW
    img = np.expand_dims(img, axis=0) # 次元合せ

    # 推論実行
    out = exec_net.infer(inputs={'data': img})

    # 出力から必要なデータのみ取り出し
    out = out['detection_out']
    out = np.squeeze(out) #サイズ1の次元を全て削除

    # 検出されたすべての顔領域に対して１つずつ処理
    for detection in out:
        # conf値の取得
        confidence = float(detection[2])

        # バウンディングボックス座標を入力画像のスケールに変換
        xmin = int(detection[3] * frame.shape[1])
        ymin = int(detection[4] * frame.shape[0])
        xmax = int(detection[5] * frame.shape[1])
        ymax = int(detection[6] * frame.shape[0])
```

```
        # conf値が0.5より大きい場合のみ感情推論とバウンディングボックス表示
        if confidence > 0.5:
            # 顔検出領域はカメラ範囲内に補正する。特にminは補正しないとエラー(
            if xmin < 0:
                xmin = 0
            if ymin < 0:
                ymin = 0
            if xmax > frame.shape[1]:
                xmax = frame.shape[1]
            if ymax > frame.shape[0]:
                ymax = frame.shape[0]

            # 顔領域のみ切り出し
            frame_face = frame[ ymin:ymax, xmin:xmax ]

            # 入力データフォーマットへ変換
            img = cv2.resize(frame_face, (64, 64))  # サイズ変更
            img = img.transpose((2, 0, 1))  # HWC > CHW
            img = np.expand_dims(img, axis=0) # 次元合せ

            # 推論実行
            out = exec_net_emotion.infer(inputs={'data': img})

            # 出力から必要なデータのみ取り出し
            out = out['prob_emotion']
            out = np.squeeze(out) #不要な次元の削減

            # 出力値が最大のインデックスを得る
            index_max = np.argmax(out)

            # 各感情の文字列をリスト化
            list_emotion = ['neutral', 'happy', 'sad', 'surprise', 'anger']

            # 文字列描画
            #cv2.putText(frame, list_emotion[index_max], (20, 60), cv2.FONT_

            # バウンディングボックス表示
            cv2.rectangle(frame, (xmin, ymin), (xmax, ymax), color=(240, 180

            # 棒グラフ表示
            str_emotion = ['neu', 'hap', 'sad', 'sur', 'ang']
            text_x = 10
            text_y = frame.shape[0] - 180
            rect_x = 80
            rect_y = frame.shape[0] - 200
            for i in range(5):
                cv2.putText(frame, str_emotion[i], (text_x, text_y), cv2.FONT_H
                cv2.rectangle(frame, (rect_x, rect_y), (rect_x + int(300 * out[i]
                text_y = text_y + 40
                rect_y = rect_y + 40

            # 顔アイコン表示
            icon_emotion[index_max].show(frame, frame.shape[1] - 110, fram

            # 1つの顔で終了
```

```
            break

    # 画像表示
    cv2.imshow('frame', frame)

    # 何らかのキーが押されたら終了
    key = cv2.waitKey(1)
    if key != -1:
        break

# 終了処理
cap.release()
cv2.destroyAllWindows()
```



追加したのは以下の3箇所です

モジュール読み込みの後に `pngoverlay` クラスの読み込みを行い、５つの画像それぞれに対しインスタンスを生成しています。
また、プログラムで扱いやすいように５つのインスタンスをリストにまとめています

```
# pngoverlayクラス読み込み
from pngoverlay import PNGOverlay

# インスタンス生成
icon_neutral = PNGOverlay('image/icon_neutral.png')
icon_happy = PNGOverlay('image/icon_happy.png')
icon_sad = PNGOverlay('image/icon_sad.png')
icon_surprise = PNGOverlay('image/icon_surprise.png')
icon_anger = PNGOverlay('image/icon_anger.png')

# インスタンス変数をリストにまとめる
icon_emotion = [icon_neutral, icon_happy, icon_sad, icon_surprise, icon_ar
```

左上に表示していた文字列をコメントアウトして非表示にしました。

```
# 文字列描画
#cv2.putText(frame, list_emotion[index_max], (20, 60), cv2.FONT_HERSHI
```

forループの最後の`brake`の前に追加しています。
感情分類が最大のインデックスであるインスタンスに対し`show`メソッド呼び出しを行い、顔アイコン画像を表示しています。

```
# 顔アイコン表示
icon_emotion[index_max].show(frame, frame.shape[1] - 110, frame.shape
```

以上、「リアルタイム感情分析アプリ」でした。
これで学習済みモデルを利用してPython APIでディープラーニング推論を実施し、実践的な表示まで行うことが身に付いたと思います。今回は「感情分類」と「顔検出」を使いましたが、インテルの学習済みモデルは他にもたくさんあります。

例えば、「歩行者の検出」「車の検出」「目・鼻・口の位置推定」「頭の方向推定」「性別・年齢の推定」などは、今までの応用ですぐにできてやりやすいと思います。

色々と挑戦してみて下さい！ではまた、お会いしましょう。お疲れさまでした！！

Newsletter

ご登録いただくと、新商品やイベント情報をお知らせします。

メールアドレスを入力してください 登録

個人情報について

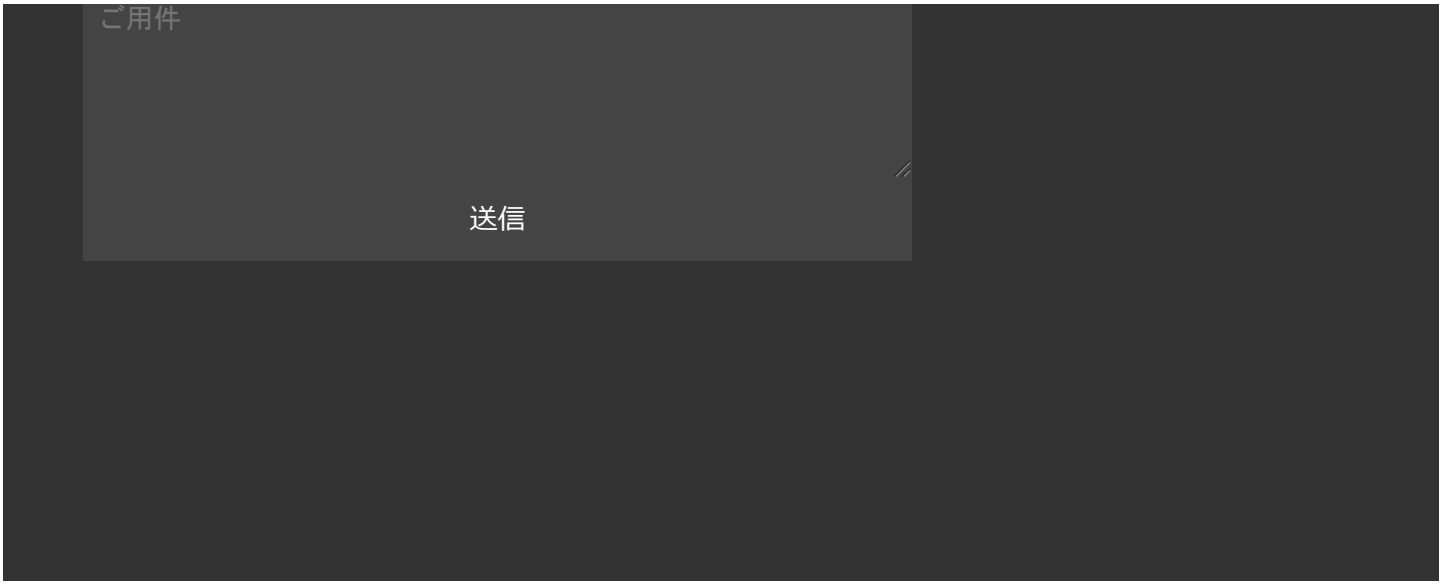
JELLYWARE

〒160-0004 東京都新宿区四谷2-3-6 パルム四谷702号室
03-6273-0758
info@jellyware.jp

個人情報について

CONTACT

お名前
メールアドレス
メールアドレス確認
お電話番号（任意）
ご件名


ご用件
送信