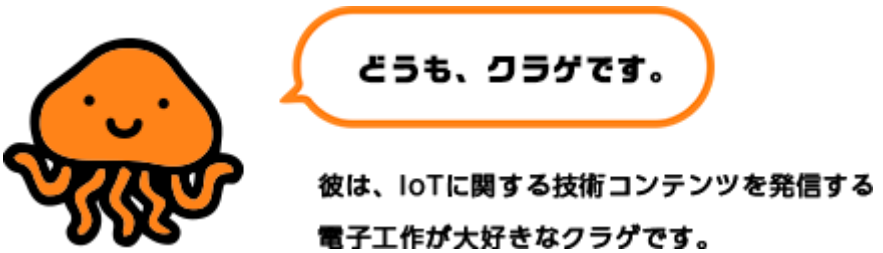
JELLYWARE

MENU

2019/11/27 update

Inference Engineを学んで感情分類



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７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７回目はOpenVINO™のInference Engineを使ってディープラーニング推論を学びます。具体的には顔画像から感情分類を行います。

【 目次 】

Inference Engine

Inference Engineは推論エンジンのことで、プログラミングで呼び出して使います。
４つのステップで簡単に使うことができます

- モジュール読み込み
- ターゲットデバイスの指定
- モデルの読み込み
- 推論実行

C++とPythonの両方が用意されていますが、このシリーズではPythonで使います。
Inference Engine Python APIの詳細はこちらにありますが、クラゲ視点で一つずつ解説してゆきます！

モジュール読み込み

Inference Engineの中の`IENetwork`と`IEPlugin`というクラスを使います。
Python基礎で習った`import`の３つ目の形式を使います。

```
# 現状だとエラー
from openvino.inference_engine import IENetwork, IEPlugin
```

しかし、RaspberryPiの場合、現状では`ImportError: No module named 'inference_engine'`などとエラーが出てしまいます。
これはパスが通っていないことが原因なので、`path.append`を使って強

制的にパスを追加することにします。なお、この関数を使うには`sys`モジュールをインポートする必要があります。

```
# 対策盛り込み
import sys
sys.path.append('/opt/intel/openvino/python/python3.5/armv7l')
from openvino.inference_engine import IENetwork, IEPlugin
```

これで`IENetwork`と`IEPlugin`が使えるようになりました

ターゲットデバイスの指定

ターゲットデバイス（プロセッサ）には以下のものがあります

- CPU
- GPU
- FPGA
- MYRIAD
- HETERO

今回はNCS(Neural Compute Stick)を使いますので`MYRIAD`を引数に渡します。以下のコード例ではキーワード引数を用いて指定しています。
※NCSを使わないPCのみの環境の方は`CPU`などに変更する必要があります

```
plugin = IEPlugin(device="MYRIAD")
```

IEPluginクラスのインスタンス生成し`plugin`という名前にしています

モデルの読み込み

概要で説明したmodelとweightsの読み込みです
modelファイル名が`xxx.xml`でweightsファイル名が`xxx.bin`だった場合、以下のコードで、読み込むことができます。

```
net = IENetwork(model='xxx.xml', weights='xxx.bin')
exec_net = plugin.load(network=net)
```

1行目はIENetworkクラスのインスタンス生成し`net`という名前にしてます

2行目はIEPluginのメソッド`load`を呼び出しています。引数に'net'を渡しています。

結果として`exec_net`という推論実行可能なネットワーク（ExecutableNetworkクラス）を作成しています。

推論実行

推論実行は、ExecutableNetworkのメソッド`infer`を呼び出すだけで行われ、戻り値に結果が入ります。

引数には具体的な入力データを渡します。具体例は後ほど。

```
out = exec_net.infer(inputs)
```

以上が、ディープラーニング推論で使うAPIです。数行程度で簡単に書けることが理解できたかと思います。

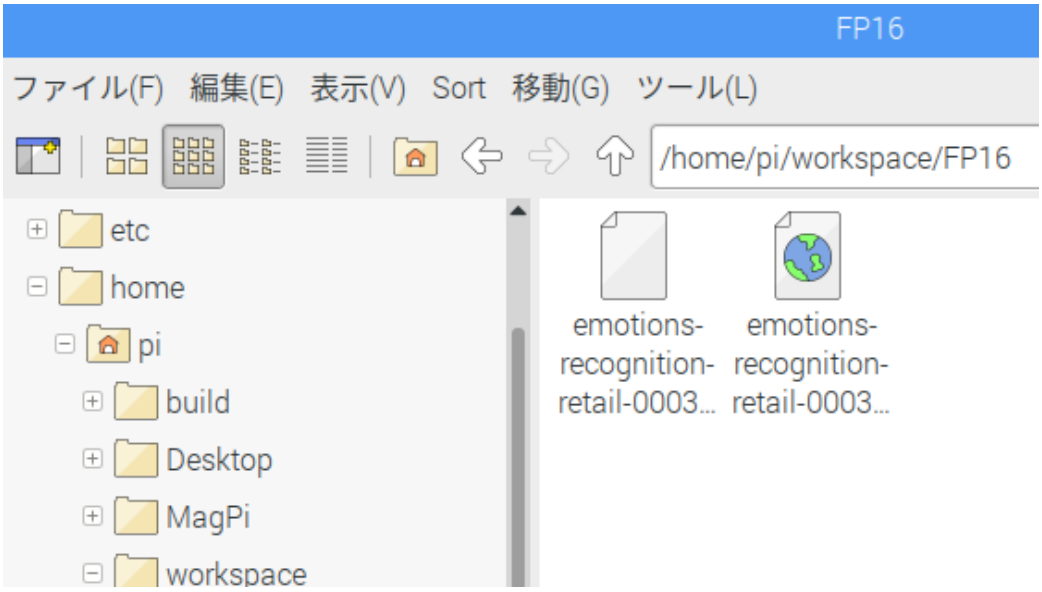
ディープラーニングで感情推論

推論エンジンの使い方が分かったところで、具体的な学習済みモデルと入力データを使って推論させ、出力結果を出したいと思います。今回は「感情推論」を行います。

学習済みモデルの取得

インテルの学習済みモデルを使用します。

modelとweightをこちらのサイトからダウンロードしてください。`workspace`の中に`FP16`フォルダを作成して、そこに移動してください。



先程のリンクの`Parent Directory`を見ると以下の３つのフォルダに分かれていると思います

- FP16
- FP32
- INT8

FPとはFloating Pointの略、INTはIntegerの略で、数値はバイト数を表しています。
これらは好きに選べる訳ではなく、デバイス（プロセッサ）により決まっています。
NCSを使った環境の場合はFP16を使います。逆に**PCのみ環境の場合は場合はFP32**を使ってください。
PCのみ環境の場合はINT8を使うことでパフォーマンスを向上させることも可能ですが、推奨はFP32になっています。
※サポート対応表はこちら

入力データ

人の感情を推論するので、入力データは顔画像です。
しかし、任意の画像をそのまま推論エンジンに入力するのではなく、学習済みモデルが要求する入力フォーマットに合わせる必要があります。
ただし、これらは手動ではなくプログラミングを用いて簡単に合わせることができます。
このモデルについてこちらのページに詳細が書いてあります。

ページの下の方を見ると、入力フォーマットが書いてあります。

> Inputs
> name: "input" , shape: [1x3x64x64] - An input image in [1xCxHxW] format. Expected color order is BGR.

入力データの名前に `'input'` とありますが、Python APIで使う場合は `'data'` が正しいようです。
型は `[1x3x64x64]` と書いてあります。 フォーマットは `[1xCxHxW]` で、カラーの順番は `BGR` という記載があります。
OpenCVのimreadを使って画像を読み込んだ場合は、カラーの順番はBGR(青, 緑, 赤)になるため、color order についてはそのままでOKです。
一方で画像のサイズについては64x64である必要があります。また、Numpyで習った通り、画像の次元は３次元でHCWフォーマットです。
つまり、「画像サイズ」と「HCWの順番」及び「次元数」についてはフォーマット変更が必要であるということが分かります。
具体的なコードは以下の通りです

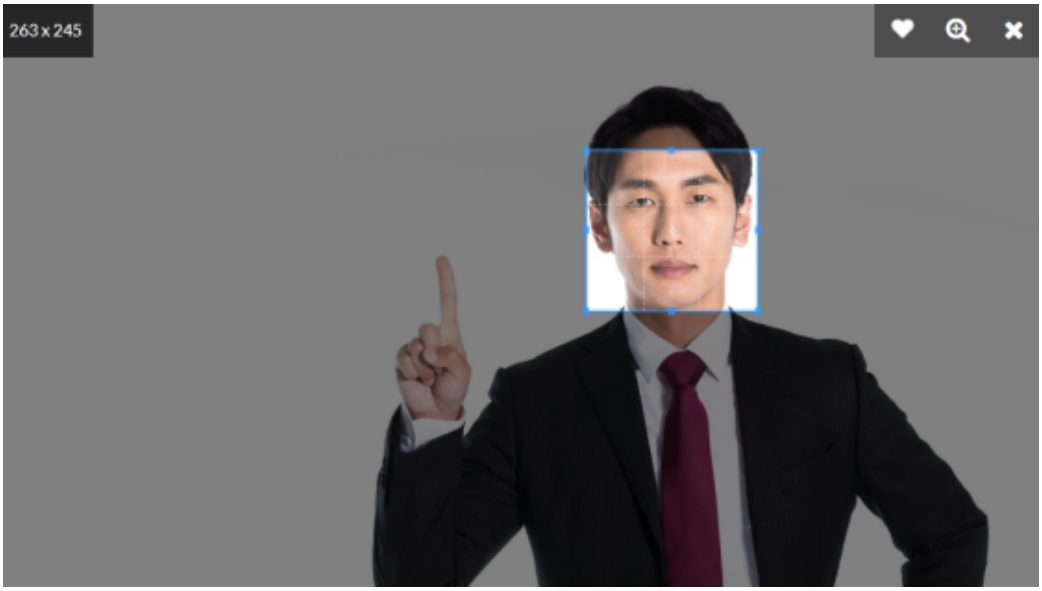
```
# 画像サイズを64x64にする
img = cv2.resize(img, (64, 64))

# HWCからCHWに変更
img = img.transpose((2, 0, 1))

# 大きさ1の次元を追加し4次元にする。省略OK
img = np.expand_dims(img, axis=0)
```

サイズ変更はOpenCVで習った項目、HWC>CHW変更と次元追加はNumPyで習った項目です。ちなみに、最後の次元追加は省略したとしても、推論エンジン側で自動的に解釈するため問題なさそうです。

では画像データを準備しましょう。最低64x64ピクセル以上のサイズがあった方が良いです。
人の顔画像であれば何でもOKです。クラゲは以下のサイトからトリミングして保存しました。出来る限り顔のみが写るようにトリミングして

ください

https://www.pakutaso.com/20190610177post-21595.html



ファイル名は`face.jpg`に変更し、`workspace`フォルダに移動しました

出力データ

先程のモデル説明ページをみると出力フォーマットについても記載があります。

> Outputs
> name: "prob", shape: [1, 5, 1, 1] - Softmax output across five emotions ('neutral', 'happy', 'sad', 'surprise', 'anger').

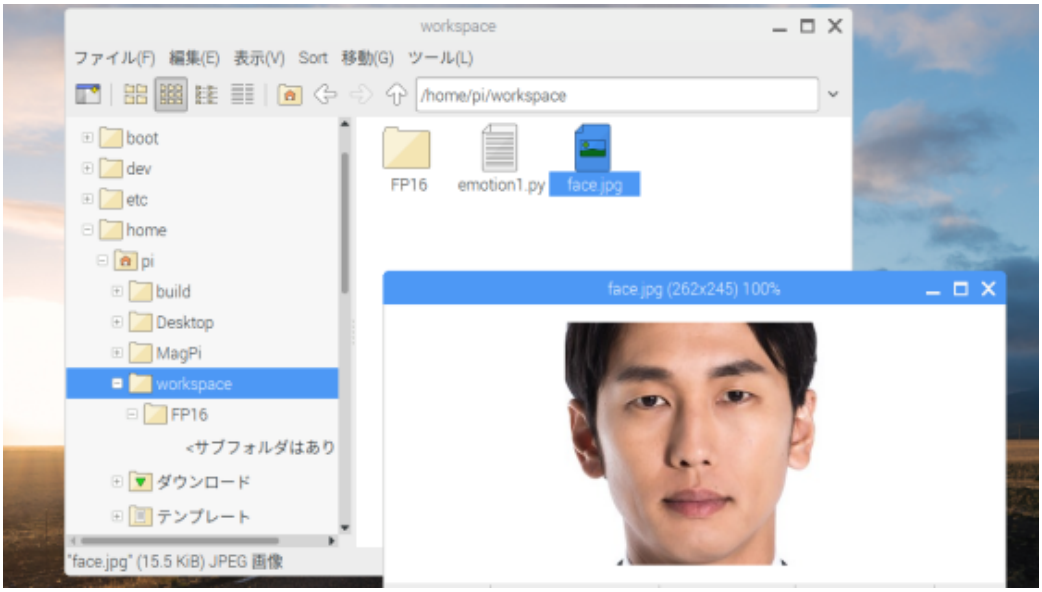
出力データの名前が`'prob'`となっていますが、実際は違うようです。
実際の名前の取得方法を後ほど説明します。
型は`[1, 5, 1, 1]`で、`Softmax`という形式で５つの感情を出力するということも読み取れます。`Softmax`の形式は割合です。つまり各感情に対応した5つの小数値が配列として出力されるということです。
大きさが1の次元は取り除いた方が見やすいので、NumPyのときに習った`squeeze`を使って次元削減します。

```
# 次元の削減
out = np.squeeze(out)
```

全体プログラム

これで関連ファイルの準備、入力・出力のフォーマットも分かりましたので、全体プログラムを作ってゆきたいと思います。

emotion1.pyというファイルを新規作成しましたが、任意のファイル名でOKです。同じディレクトリにFP16フォルダとface.jpgがあることを確認してください。以下の画像は画像ファイル`face.jpg`をダブルクリックしただけです。



これまでの組合せで一連のプログラムが出来ます

```
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
net = IENetwork(model='FP16/emotions-recognition-retail-0003.xml', weig
exec_net = plugin.load(network=net)

# 入力画像読み込み
img = cv2.imread('face.jpg')

# 入力データフォーマットへ変換
img = cv2.resize(img, (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.infer(inputs={'data': img})
```

```
# 出力
print(out)
```

推論実行の`infer`メソッドの引数は`inputs={'data': img}`となっていますが、Python基礎で習ったキーワード引数と辞書の組合せです。入力データのキーワードは`'data'`で、値として`img`を入れています
実行結果は以下のように出力されたと思います。ただし各数値は入力画像により結果は異なります。

```
{'prob_emotion': array([[[[ 0.85107422]],

        [[ 0.09069824]],

        [[ 0.03393555]],

        [[ 0.01065063]],

        [[ 0.01377869]]]], dtype=float32)}
```

これを見ると出力データの名前が`'prob_emotion'`であることが分かると思います。
必要なデータのみを取得するには、以下のコードを最後の出力前に追記します

```
# 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
```

これで実行すると以下が出力されます
※もし、エラーになった場合は次の項を参照してください

```
[ 0.85107422  0.09069824  0.03393555  0.01065063  0.01377869]
```

それぞれの値は、'neutral', 'happy', 'sad', 'surprise', 'anger'の割合を示しています。今回入力した画像だと 'neutral' が群を抜いて最も高いことが分かります。
ちなみに、割合なので当然ですが、これらの数値を合計するとほぼ1.0になります

実行時のエラー対応

Thonnyにて以下のようなエラーが表示された場合の対応方法です

> RuntimeError: Can not init Myriad device: NC_ERROR

このエラーはThonnyでNCSを使うときの相性の問題です。どうやらNCSを使ったプログラムを実行すると、プログラムが終了したとしても、しばらくの間はNCSが他で使えなくなってしまうようです。
対策として２つあります

1. NCSを一旦抜いてまた挿す
2. LXTerminalから実行する

対策1：NCSを一旦抜いてまた挿す

書いてある通りそのままです。NCSを使ったプログラムを実行して終了したら、一旦NCSを抜いて挿します。これですぐに使えるようになりますが、何度も挿抜を繰り返すのは面倒ですので一時的な対策となります。恒久対策は次の対策2をお勧めします。

対策2：LXTerminalから実行する

`LXTerminal`を開いて、`workspace`　に移動します

```
cd workspace
```

もし、実行したいファイル名が `test.py` の場合は、以下のように打ち込みます

```
python3 test.py
```

これで毎回挿抜しなくてもエラーにならずに済みます。
一方で、毎回 `python3 ...` と打ち込むのは面倒です。実はキーボードで↑キーを押すとコマンドの履歴を呼び出すことができます。これを利用することで簡単に同じプログラムを再実行したり、ファイル名の部分だけ書き換えれば別のプログラムを実行することができます。

コードの編集と保存については、これまで通りThonnyを使うことをお勧めします。もし、間違ってThonny上で実行した場合は、対策1を行えば大丈夫です。

感情分類アプリ

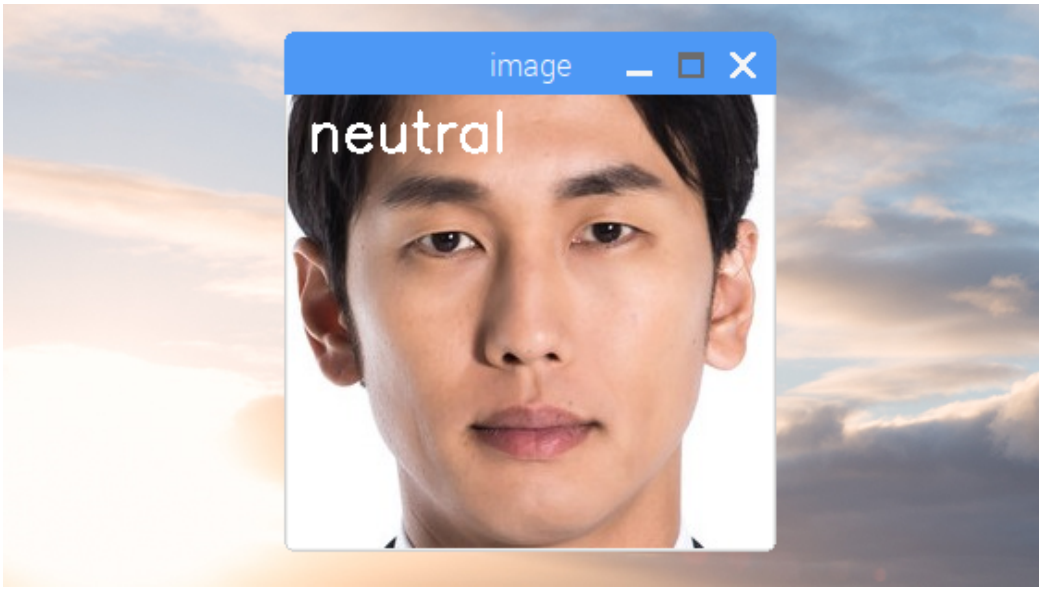
最後にアプリっぽくするために、元画像を表示してそこに分類結果を文字列描画したいと思います

```
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
net = IENetwork(model='FP16/emotions-recognition-retail-0003.xml', weig
exec_net = plugin.load(network=net)

# 入力画像読み込み
img_face = cv2.imread('face.jpg')

# 入力データフォーマットへ変換
img = cv2.resize(img_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.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
print(out)

# 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(img_face, list_emotion[index_max], (10, 30), cv2.FONT_HERS

# 画像表示
cv2.imshow('image', img_face)

# キーが押されたら終了
cv2.waitKey(0)
cv2.destroyAllWindows()
```

先程のプログラムだと元画像もフォーマット変換後の画像も同じ`img`という変数になっているので、元画像の部分だけを`img_face`に変更しています。（17行目と20行目)
それ以外は、先ほどのプログラムへの追記です。NumPyで習った`argmax`を使い、最も出力値が高いインデックスを得ます。各感情の文字列をリスト化し、先ほどのインデックスを用いて`putText`で文字列描画を行っています。画像表示に関しては「OpenCVを学ぶ」で習った項目を使っています。

実行結果



別の画像も色々と入力して試してみてください！

現状だと、手動で顔の領域をトリミングしなければなりません。これだとリアルタイムカメラ入力に対応できないため、次回はディープラーニングで自動的に顔領域の検出を行いたいと思います。

以上、「Inference Engineを学んで感情分類」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

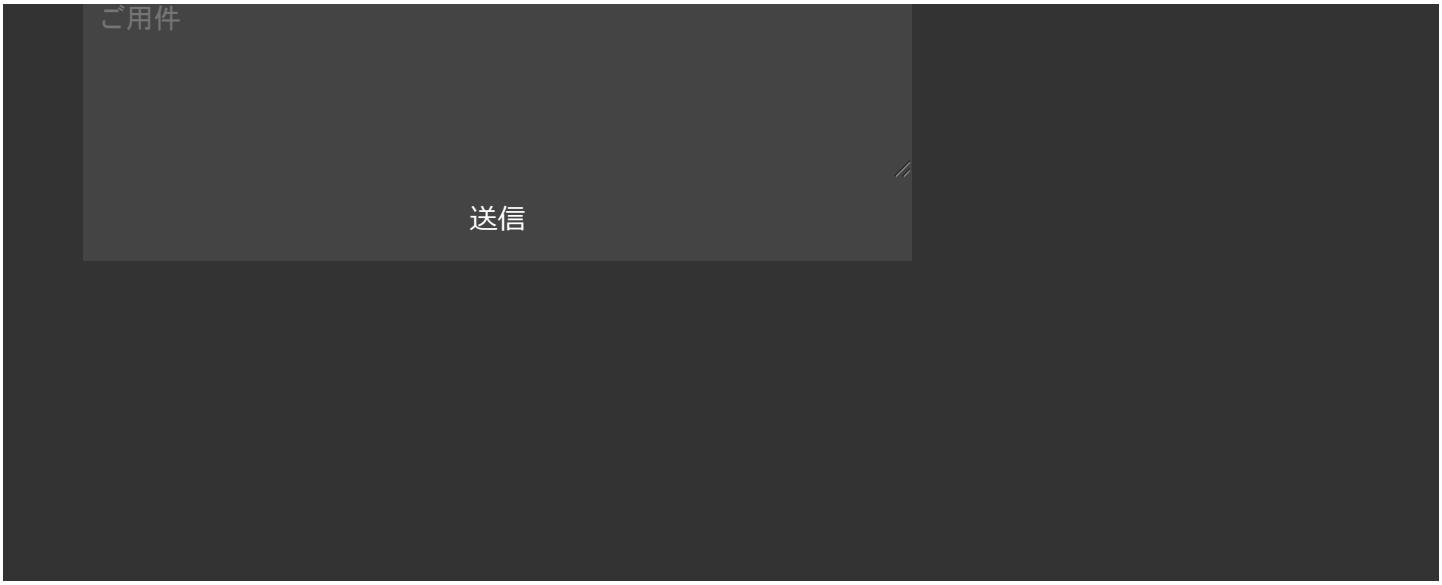
JELLYWARE

〒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