dnn-locate

Ben Dai

Jun 16, 2023

CONTENTS

dnn-locate is a Python module for discriminative features localization given a fitted discriminator model, including **neural networks**. **dnn-locate** has the following key features:

1. **Adaptiveness**. For different instances, **dnn-locate** is able to provide **adaptive** discriminative features.
2. **Sparseness**. The discriminative features provided by **dnn-locate** is sparse.
3. *Statistically guaranteed interpretation* in R-square (R2). **dnn-locate** is able to *effectively* localize the discriminative features with a target R2 of prediction.

You can find more information for **dnn-locate**:

- GitHub repo: https://github.com/statmlben/dnn-inference

- Documentation: https://dnn-locate.readthedocs.io
- PyPi: https://pypi.org/project/dnn-locate
- Open Source: MIT license

CHAPTER

ONE

TRUST (TANH RELU SOFTMAX) ACTIVATION FUNCTION

We achieve the (1)-(3) by using the **Magic** activation: *tanh* + *relu* + *softmax*

```
from tensorflow.keras import backend as K

def trust(x, tau, axis_=(1,2)):
   z = tau*K.softmax(x, axis=axis_)
   z = backend.tanh(backend.relu(z))
   return z
```

trust(x) satisfies that: (i) *trust(x)* <= *1*; (ii) *trust(x)* <= *tau*, that is, each element if controlled by 1, and the sum of all elements is controlled by *tau*.

CHAPTER

TWO

REFERENCE

If you use this code please star the repository and cite the following paper:

```
@article{dai2022locate,
   title={Data-adaptive discriminative feature localization with statistically␣
↪guaranteed interpretation},
   author={Dai, Ben and Shen, Xiaotong and Chen, Lin Yee and Li, Chunlin and Pan, Wei},
   journal={Annals of Applied Statistics},
   year={2022},
   publisher={Institute of Mathematical Statistics}
}
```

CHAPTER

THREE

CONTENTS

3.1 Installation

3.1.1 Dependencies

`dnn-locate` requires: **Python>=3.8** + [pip libs](./requirements.txt)

```
pip install -r requirements.txt
```

3.1.2 User installation

Install `dnn-locate` using `pip`

```
pip install dnn_locate
pip install git+https://github.com/statmlben/dnn-locate.git
```

3.1.3 Source code

You can check the latest sources with the command.

```
git clone https://github.com/statmlben/dnn-locate.git
```

3.2 Examples

3.2.1 ECG MIT-BIH dataset

Train ECG localization network

[1]:
```
# Author: Ben Dai
# Licensed under the Apache License, Version 2.0 (the "License");
# Train ECG localization network
```

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.layers import Dense, Activation, Flatten, Convolution1D, Dropout,
↪MaxPooling1D,GlobalAveragePooling1D
from tensorflow.keras import Model, layers,Sequential,regularizers
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping,ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import LearningRateScheduler
```

```
2022-10-14 21:52:36.322750: I tensorflow/core/util/util.cc:169] oneDNN custom operations␣
↪are on. You may see slightly different numerical results due to floating-point round-
↪off errors from different computation orders. To turn them off, set the environment␣
↪variable `TF_ENABLE_ONEDNN_OPTS=0`.
```

[2]:
```
## Load data and pretrained model

discriminator=tf.keras.models.load_model('./tests/ECG_model/pretrained_model.h5')
# discriminator.summary()

mit_train_path="./dataset/mitbih_train.csv"
mit_test_path="./dataset/mitbih_test.csv"

def create_pd(train_path,test_path):
    train=pd.read_csv(train_path)
    test=pd.read_csv(test_path)
    train.columns=[x for x in range(188)]
    test.columns=[x for x in range(188)]
    return pd.concat([train,test], axis=0, join='inner').sort_index()

mit= create_pd(mit_train_path,mit_test_path)

X = np.asarray(mit.iloc[:,:187].values)
y = mit.iloc[:,187].values
y = to_categorical(y)

X = X.reshape(-1, 187, 1)
input_shape = X.shape[1:]

from sklearn.model_selection import train_test_split
X, X_test, y, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

```
2022-10-14 21:52:39.025442: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-10-14 21:52:39.030579: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-10-14 21:52:39.030931: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
```

(continues on next page)

(continued from previous page)

```
2022-10-14 21:52:39.031751: I tensorflow/core/platform/cpu_feature_guard.cc:193] This␣
↪TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use␣
↪the following CPU instructions in performance-critical operations:  AVX2 AVX512F␣
↪AVX512_VNNI FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler␣
↪flags.
2022-10-14 21:52:39.032337: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-10-14 21:52:39.032622: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-10-14 21:52:39.032886: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-10-14 21:52:39.390903: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-10-14 21:52:39.391197: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-10-14 21:52:39.391438: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2022-10-14 21:52:39.391670: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1532]␣
↪Created device /job:localhost/replica:0/task:0/device:GPU:0 with 3647 MB memory:  ->␣
↪device: 0, name: NVIDIA GeForce RTX 2060, pci bus id: 0000:01:00.0, compute capability:
↪ 7.5
```

[3]:
```
## Model
from dnn_locate import loc_model
## define the backend localizer before TRELU activation

localizer_backend = tf.keras.Sequential(
    [
        layers.Input(shape=(input_shape[0], input_shape[1])),
        layers.Conv1D(
            filters=32, kernel_size=5, padding="same", strides=1, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1D(
            filters=16, kernel_size=5, padding="same", strides=1, activation="relu"
        ),
        layers.Conv1DTranspose(
            filters=16, kernel_size=5, padding="same", strides=1, activation="relu"
        ),
        layers.Dropout(rate=0.2),
        layers.Conv1DTranspose(
            filters=32, kernel_size=5, padding="same", strides=1, activation="relu"
        ),
        layers.Conv1DTranspose(filters=1, kernel_size=5, padding="same"),
    ]
```

(continues on next page)

(continued from previous page)

```
)

es_detect1 = ReduceLROnPlateau(monitor="loss", factor=0.382, min_lr=1e-6,
                    verbose=1, patience=3, mode="min")
es_detect2 = EarlyStopping(monitor='loss', mode='min', min_delta=.0001,
                        verbose=1, patience=10, restore_best_weights=True)


fit_params={'callbacks': [es_detect1, es_detect2],
            'epochs': 100, 'batch_size': 64}

tau_range = [10., 20, 30]

## define framework
cue = loc_model(input_shape=input_shape,
                localizer_backend=localizer_backend,
                discriminator=discriminator,
                target_r_square='auto',
                r_metric='acc',
                # r_metric='loss',
                tau_range=tau_range)
```

```
/home/ben/tf/lib/python3.10/site-packages/keras/optimizers/optimizer_v2/gradient_descent.
↪py:108: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(SGD, self).__init__(name, **kwargs)
```

[4]:
```
cue.fit(X_train=X, y_train=y,
            fit_params=fit_params,
            optimizer=Adam(learning_rate=.01)
            # optimizer=SGDW(learning_rate=.1, weight_decay=.0001, momentum=.9)
            )
```

```
2022-10-14 21:52:44.663196: I tensorflow/stream_executor/cuda/cuda_dnn.cc:384] Loaded␣
↪cuDNN version 8101
```

```
2292/2292 [==============================] - 4s 1ms/step - loss: 0.0483 - accuracy: 0.
↪9861 - auc: 0.9990
2292/2292 [==============================] - 3s 1ms/step - loss: 5.1039 - accuracy: 0.
↪7042 - auc: 0.8160
Epoch 1/100
1146/1146 [==============================] - 7s 5ms/step - loss: -5.3217 - lr: 0.0100
Epoch 2/100
1146/1146 [==============================] - 4s 4ms/step - loss: -7.3224 - lr: 0.0100
Epoch 3/100
1146/1146 [==============================] - 4s 4ms/step - loss: -7.9352 - lr: 0.0100
Epoch 4/100
1146/1146 [==============================] - 4s 4ms/step - loss: -8.1900 - lr: 0.0100
Epoch 5/100
1146/1146 [==============================] - 4s 4ms/step - loss: -8.1250 - lr: 0.0100
Epoch 6/100
1146/1146 [==============================] - 4s 4ms/step - loss: -8.2364 - lr: 0.0100
Epoch 7/100
1146/1146 [==============================] - 4s 4ms/step - loss: -8.4304 - lr: 0.0100
```

(continues on next page)

(continued from previous page)

```
Epoch 8/100
1146/1146 [==============================] - 4s 4ms/step - loss: -8.7386 - lr: 0.0100
Epoch 9/100
1146/1146 [==============================] - 5s 4ms/step - loss: -8.6743 - lr: 0.0100
Epoch 10/100
1146/1146 [==============================] - 4s 4ms/step - loss: -8.7296 - lr: 0.0100
Epoch 11/100
1146/1146 [==============================] - 4s 4ms/step - loss: -8.8596 - lr: 0.0100
Epoch 12/100
1146/1146 [==============================] - 4s 4ms/step - loss: -8.9157 - lr: 0.0100
Epoch 13/100
1146/1146 [==============================] - 4s 4ms/step - loss: -8.7552 - lr: 0.0100
Epoch 14/100
1146/1146 [==============================] - 4s 4ms/step - loss: -9.0108 - lr: 0.0100
Epoch 15/100
1146/1146 [==============================] - 4s 4ms/step - loss: -8.7194 - lr: 0.0100
Epoch 16/100
1146/1146 [==============================] - 4s 4ms/step - loss: -9.1780 - lr: 0.0100
Epoch 17/100
1146/1146 [==============================] - 4s 4ms/step - loss: -9.0624 - lr: 0.0100
Epoch 18/100
1146/1146 [==============================] - 4s 4ms/step - loss: -9.2963 - lr: 0.0100
Epoch 19/100
1146/1146 [==============================] - 4s 4ms/step - loss: -9.2474 - lr: 0.0100
Epoch 20/100
1146/1146 [==============================] - 4s 4ms/step - loss: -9.2477 - lr: 0.0100
Epoch 21/100
1146/1146 [==============================] - 4s 4ms/step - loss: -9.3848 - lr: 0.0100
Epoch 22/100
1146/1146 [==============================] - 4s 4ms/step - loss: -9.4511 - lr: 0.0100
Epoch 23/100
1146/1146 [==============================] - 4s 4ms/step - loss: -9.5072 - lr: 0.0100
Epoch 24/100
1146/1146 [==============================] - 4s 4ms/step - loss: -9.4762 - lr: 0.0100
Epoch 25/100
1146/1146 [==============================] - 4s 4ms/step - loss: -9.5786 - lr: 0.0100
Epoch 26/100
1146/1146 [==============================] - 5s 4ms/step - loss: -9.6250 - lr: 0.0100
Epoch 27/100
1146/1146 [==============================] - 5s 4ms/step - loss: -9.3385 - lr: 0.0100
Epoch 28/100
1146/1146 [==============================] - 4s 4ms/step - loss: -9.3302 - lr: 0.0100
Epoch 29/100
1140/1146 [============================>.] - ETA: 0s - loss: -9.5342
Epoch 29: ReduceLROnPlateau reducing learning rate to 0.0038199999146163463.
1146/1146 [==============================] - 4s 4ms/step - loss: -9.5332 - lr: 0.0100
Epoch 30/100
1146/1146 [==============================] - 4s 4ms/step - loss: -9.8781 - lr: 0.0038
Epoch 31/100
1146/1146 [==============================] - 4s 4ms/step - loss: -9.9837 - lr: 0.0038
Epoch 32/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.1294 - lr: 0.0038
```

(continues on next page)

(continued from previous page)

```
Epoch 33/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.1395 - lr: 0.0038
Epoch 34/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.1378 - lr: 0.0038
Epoch 35/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.0610 - lr: 0.0038
Epoch 36/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.1958 - lr: 0.0038
Epoch 37/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.2177 - lr: 0.0038
Epoch 38/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.1821 - lr: 0.0038
Epoch 39/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.2419 - lr: 0.0038
Epoch 40/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.1798 - lr: 0.0038
Epoch 41/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.2267 - lr: 0.0038
Epoch 42/100
1139/1146 [============================>.] - ETA: 0s - loss: -10.2399
Epoch 42: ReduceLROnPlateau reducing learning rate to 0.0014592399937100708.
1146/1146 [==============================] - 4s 4ms/step - loss: -10.2379 - lr: 0.0038
Epoch 43/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.2871 - lr: 0.0015
Epoch 44/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.4394 - lr: 0.0015
Epoch 45/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.4516 - lr: 0.0015
Epoch 46/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.4511 - lr: 0.0015
Epoch 47/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.4310 - lr: 0.0015
Epoch 48/100
1137/1146 [============================>.] - ETA: 0s - loss: -10.4450
Epoch 48: ReduceLROnPlateau reducing learning rate to 0.0005574296731501818.
1146/1146 [==============================] - 4s 4ms/step - loss: -10.4506 - lr: 0.0015
Epoch 49/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5040 - lr: 5.5743e-
↪04
Epoch 50/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5365 - lr: 5.5743e-
↪04
Epoch 51/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.4818 - lr: 5.5743e-
↪04
Epoch 52/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.4441 - lr: 5.5743e-
↪04
Epoch 53/100
1146/1146 [==============================] - ETA: 0s - loss: -10.4994
Epoch 53: ReduceLROnPlateau reducing learning rate to 0.00021293813816737384.
1146/1146 [==============================] - 4s 4ms/step - loss: -10.4994 - lr: 5.5743e-
```

(continues on next page)

(continued from previous page)

```
↪04
Epoch 54/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5423 - lr: 2.1294e-
↪04
Epoch 55/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5513 - lr: 2.1294e-
↪04
Epoch 56/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5156 - lr: 2.1294e-
↪04
Epoch 57/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5144 - lr: 2.1294e-
↪04
Epoch 58/100
1136/1146 [============================>.] - ETA: 0s - loss: -10.5268
Epoch 58: ReduceLROnPlateau reducing learning rate to 8.134236637852155e-05.
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5221 - lr: 2.1294e-
↪04
Epoch 59/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5352 - lr: 8.1342e-
↪05
Epoch 60/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5340 - lr: 8.1342e-
↪05
Epoch 61/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5820 - lr: 8.1342e-
↪05
Epoch 62/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5997 - lr: 8.1342e-
↪05
Epoch 63/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5458 - lr: 8.1342e-
↪05
Epoch 64/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5407 - lr: 8.1342e-
↪05
Epoch 65/100
1145/1146 [============================>.] - ETA: 0s - loss: -10.5950
Epoch 65: ReduceLROnPlateau reducing learning rate to 3.107278476818465e-05.
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5960 - lr: 8.1342e-
↪05
Epoch 66/100
1146/1146 [==============================] - 5s 4ms/step - loss: -10.5197 - lr: 3.1073e-
↪05
Epoch 67/100
1146/1146 [==============================] - 5s 4ms/step - loss: -10.5252 - lr: 3.1073e-
↪05
Epoch 68/100
1146/1146 [==============================] - 5s 4ms/step - loss: -10.6115 - lr: 3.1073e-
↪05
Epoch 69/100
1146/1146 [==============================] - 5s 4ms/step - loss: -10.5617 - lr: 3.1073e-
```

(continues on next page)

(continued from previous page)

```
↪05
Epoch 70/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.6030 - lr: 3.1073e-
↪05
Epoch 71/100
1146/1146 [==============================] - ETA: 0s - loss: -10.5464
Epoch 71: ReduceLROnPlateau reducing learning rate to 1.1869803725858219e-05.
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5464 - lr: 3.1073e-
↪05
Epoch 72/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.6003 - lr: 1.1870e-
↪05
Epoch 73/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5594 - lr: 1.1870e-
↪05
Epoch 74/100
1133/1146 [============================>.] - ETA: 0s - loss: -10.5983
Epoch 74: ReduceLROnPlateau reducing learning rate to 4.534264948233613e-06.
1146/1146 [==============================] - 4s 4ms/step - loss: -10.6071 - lr: 1.1870e-
↪05
Epoch 75/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5452 - lr: 4.5343e-
↪06
Epoch 76/100
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5684 - lr: 4.5343e-
↪06
Epoch 77/100
1143/1146 [============================>.] - ETA: 0s - loss: -10.5494
Epoch 77: ReduceLROnPlateau reducing learning rate to 1.7320891984127228e-06.
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5420 - lr: 4.5343e-
↪06
Epoch 78/100
1137/1146 [============================>.] - ETA: 0s - loss: -10.5708Restoring model␣
↪weights from the end of the best epoch: 68.
1146/1146 [==============================] - 4s 4ms/step - loss: -10.5738 - lr: 1.7321e-
↪06
Epoch 78: early stopping
##################################################
compute the R2 for the fitted localizer.
##################################################
2292/2292 [==============================] - 3s 1ms/step - loss: 0.0483 - accuracy: 0.
↪9861 - auc: 0.9990
2292/2292 [==============================] - 2s 945us/step
2292/2292 [==============================] - 3s 1ms/step - loss: 11.0031 - accuracy: 0.
↪1729 - auc: 0.4940
1/1 [==============================] - 0s 92ms/step
early stop in tau = 10.000, R2: 0.983; target R2: 0.953 is reached
```

[5]:
```
## print R_square for Test set
cue.R_square(X_test, y_test)
```

```
##################################################
```

(continues on next page)

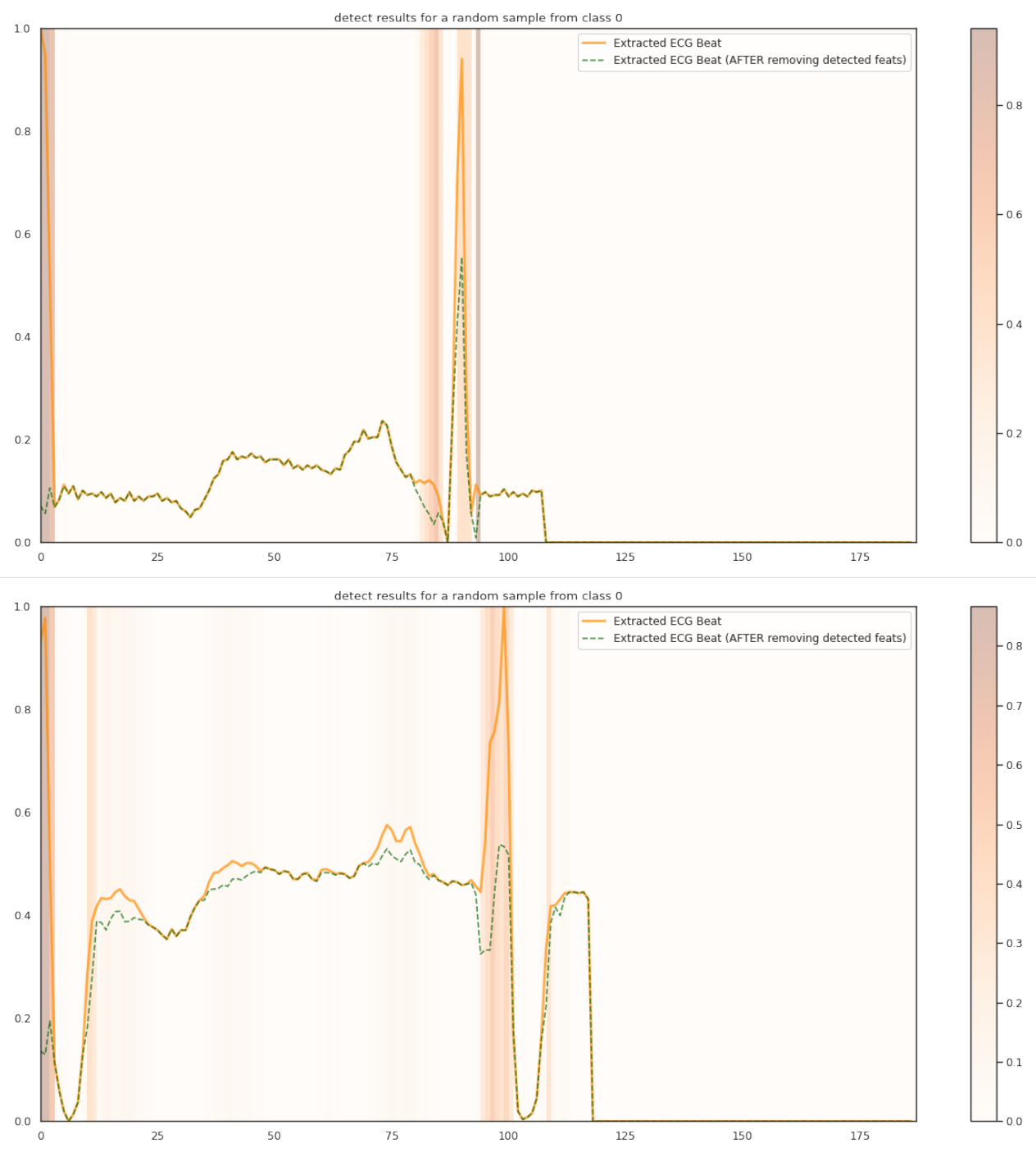
(continued from previous page)

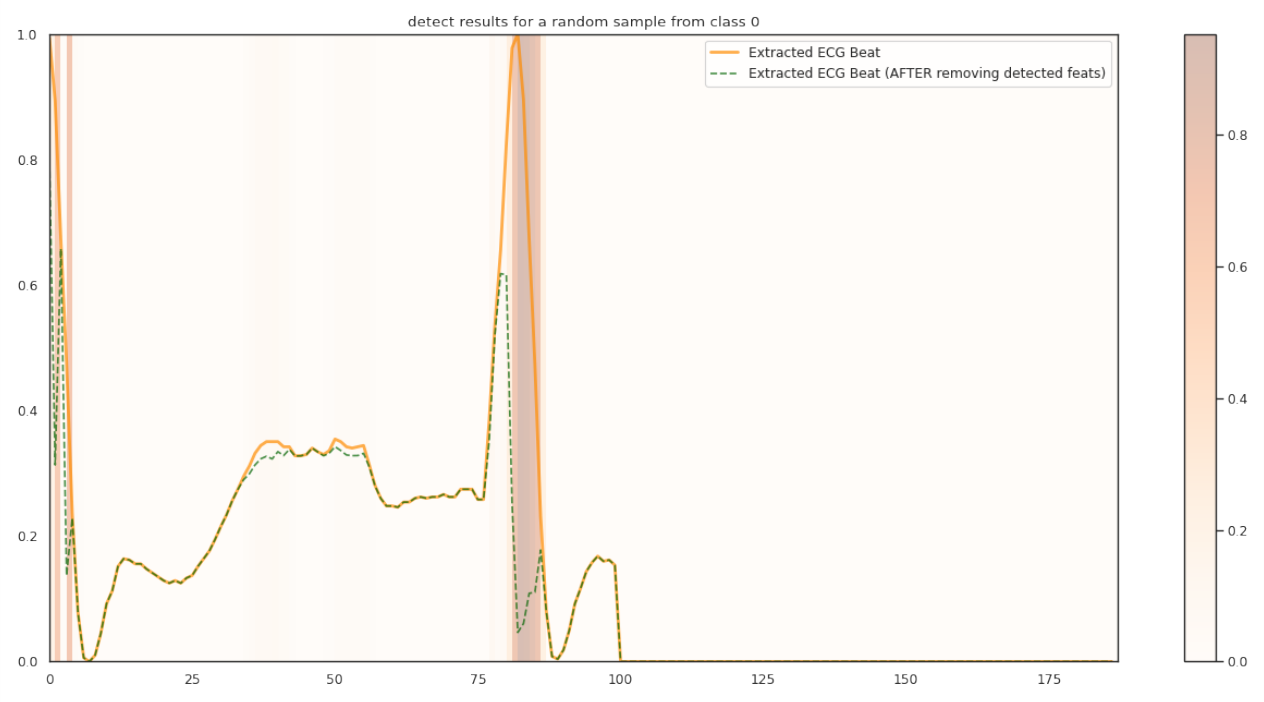
```
compute the R2 for the fitted localizer.
##################################################
1129/1129 [==============================] - 2s 2ms/step - loss: 0.0455 - accuracy: 0.
↪9862 - auc: 0.9992
1129/1129 [==============================] - 1s 939us/step
1129/1129 [==============================] - 2s 1ms/step - loss: 11.0295 - accuracy: 0.
↪1713 - auc: 0.4927
```

[5]:
```
0.9833611752340135
```

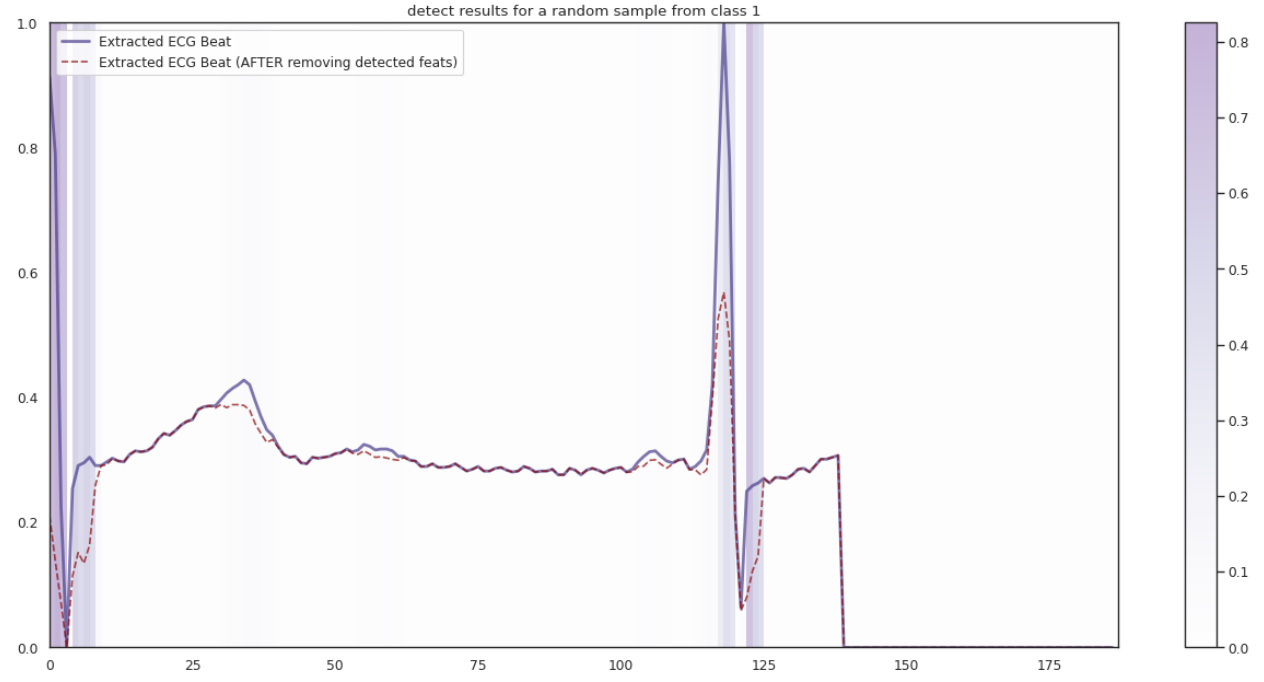
[6]:
```
## Plot the localization results by the fitted network for novel instances
import seaborn as sns
import matplotlib

import matplotlib.pyplot as plt

n_label = y.shape[1]
cmap = ["Oranges", "Purples", "Reds", "Blues", "Greens"]
c1 = ['darkorange', 'darkslateblue', 'darkred', 'darkblue', 'darkgreen']
c2 = ['darkgreen', 'darkred', 'darkslateblue', 'darkred', 'darkorange']

n_demo = 3
timepoint = list(range(input_shape[0]))

for k in range(n_label):
    demo_ind = np.array([np.random.choice(np.where(y_test[:,k] == 1)[0]) for i in␣
↪range(n_demo)])
    X_demo = X_test[demo_ind]
    X_demo_detect = cue.localizer.predict(X_demo)
    X_demo_hl = cue.locate(X_demo)

    sns.set_theme(style= 'white', palette=None)
    for i in range(len(X_demo)):
        X_tmp, X_detect_tmp, X_hl_tmp = X_demo[i], X_demo_detect[i], X_demo_hl[i]
        plt.figure(figsize=(16, 8), dpi=80)
        plt.title('detect results for a random sample from class %s' %k)
        plt.imshow(X_hl_tmp[np.newaxis,:], cmap=cmap[k], aspect='auto', alpha=0.3,
                                           extent = (0, 187, 0, 1))
        plt.colorbar()
        plt.plot(timepoint, X_tmp, linewidth=2.5, alpha=.7, color=c1[k],
            label='Extracted ECG Beat')
        plt.plot(timepoint, X_detect_tmp, linewidth=1.5, alpha=.7, color=c2[k],␣
↪linestyle='--',
            label='Extracted ECG Beat (AFTER removing detected feats)')
        plt.legend(loc='best')
        plt.tight_layout()
        plt.show()
```

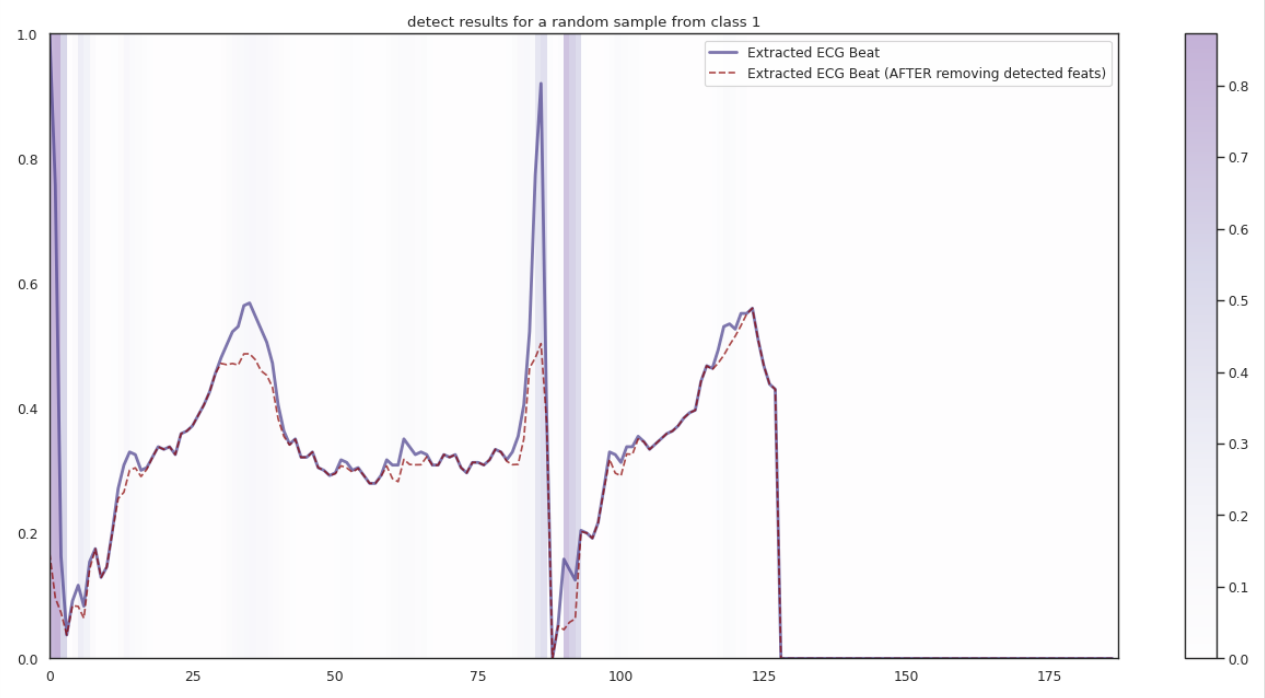
```
1/1 [==============================] - 0s 101ms/step
1/1 [==============================] - 0s 12ms/step
```

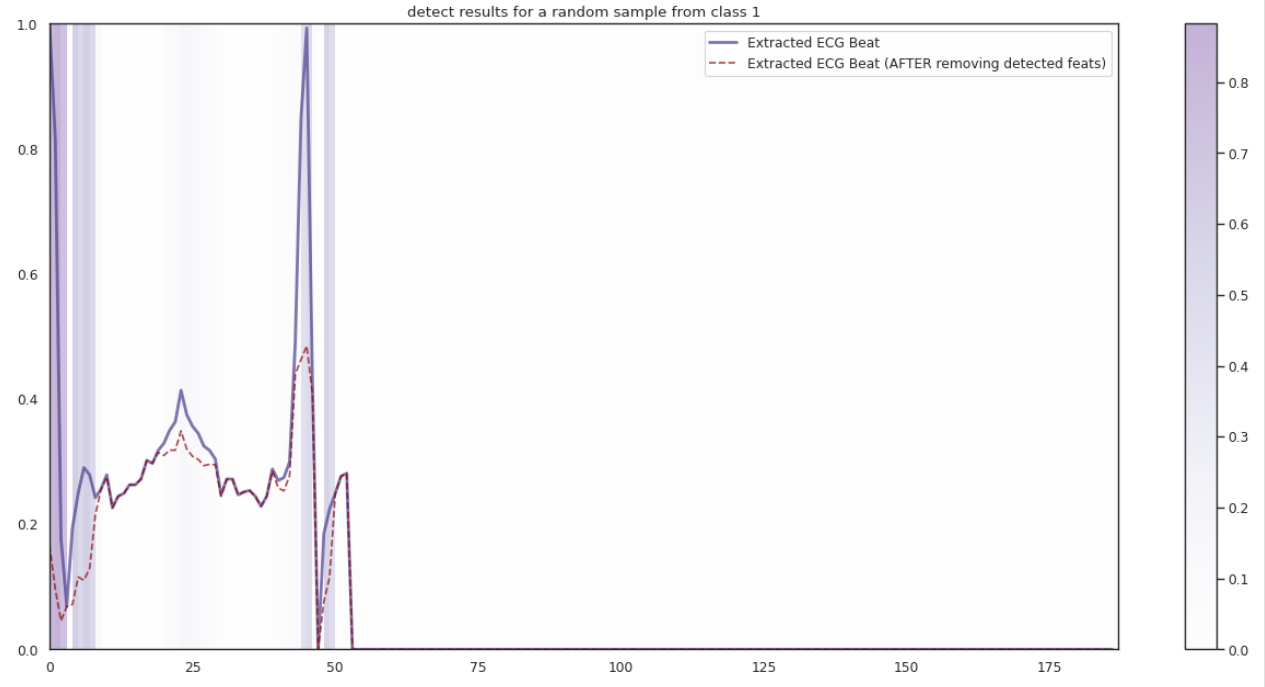

detect results for a random sample from class 0
Extracted ECG Beat
Extracted ECG Beat (AFTER removing detected feats)
detect results for a random sample from class 0
Extracted ECG Beat
Extracted ECG Beat (AFTER removing detected feats)



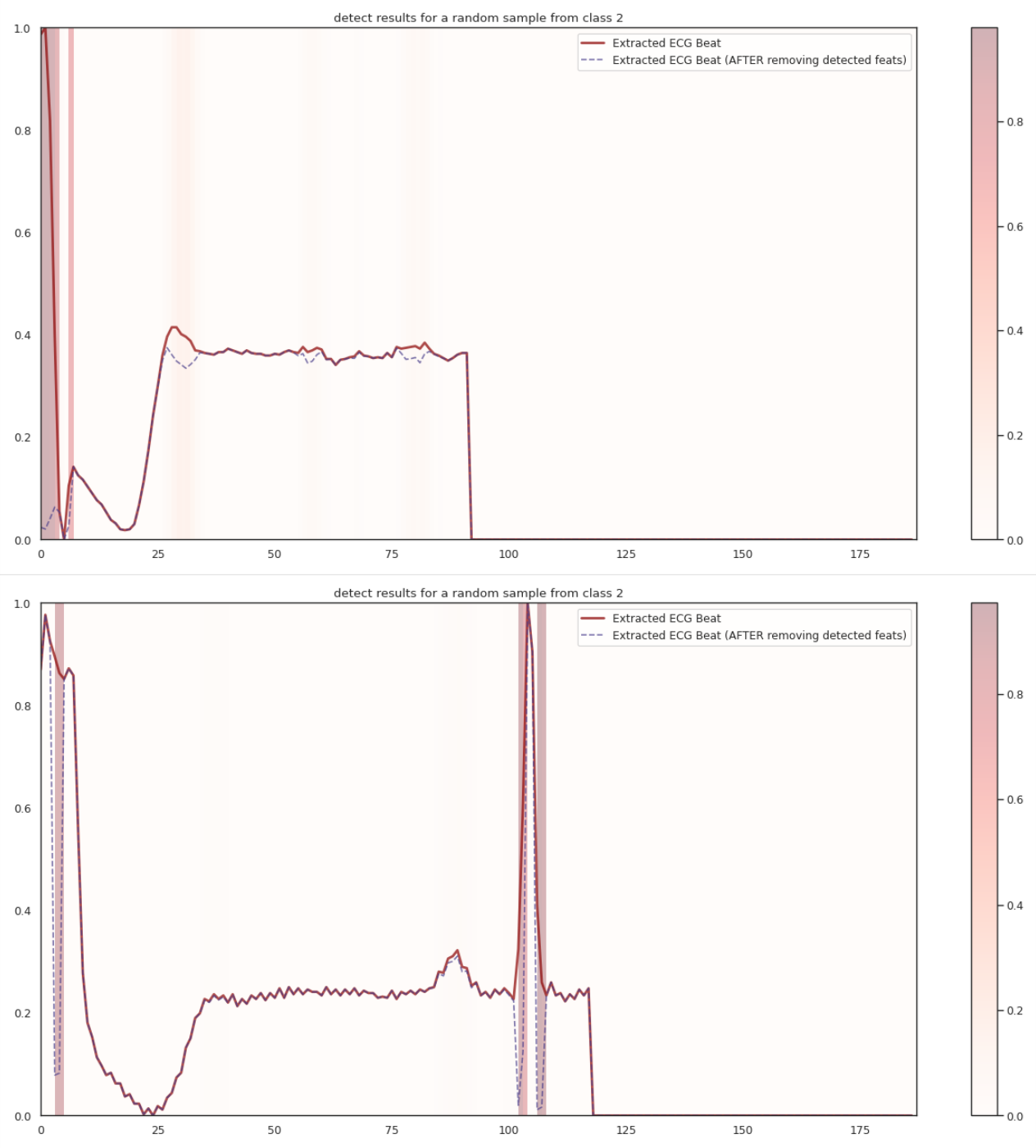

```
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 12ms/step
```

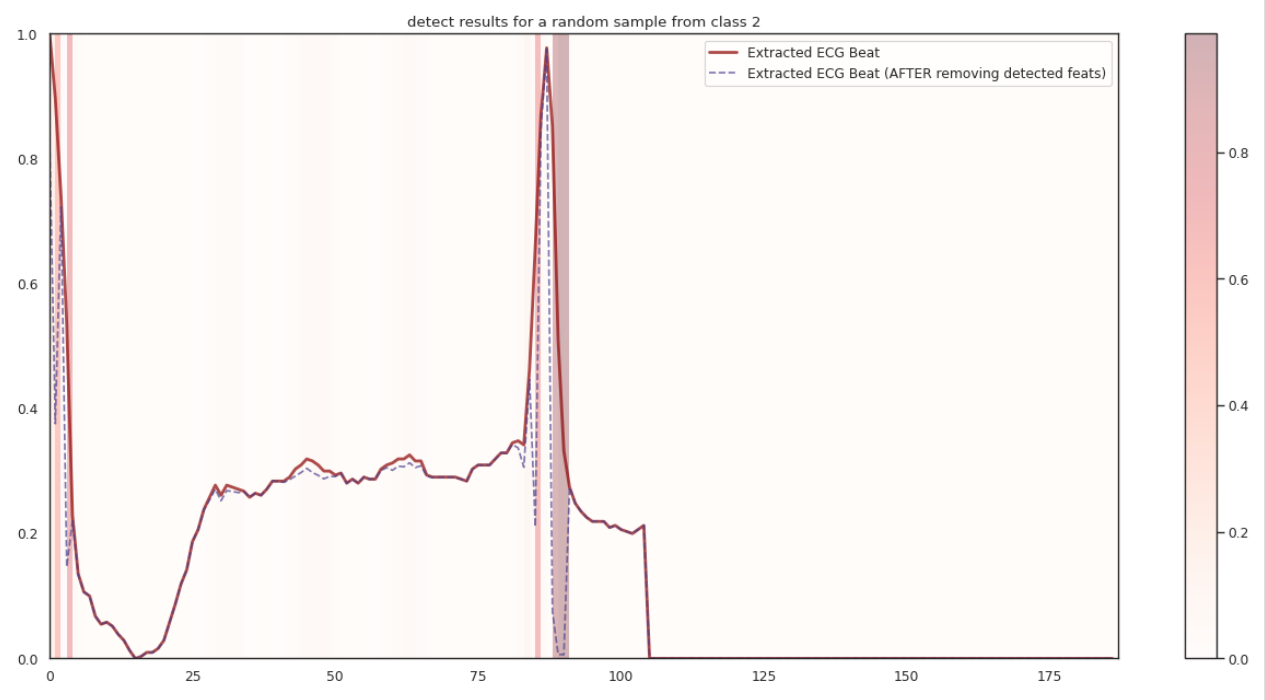




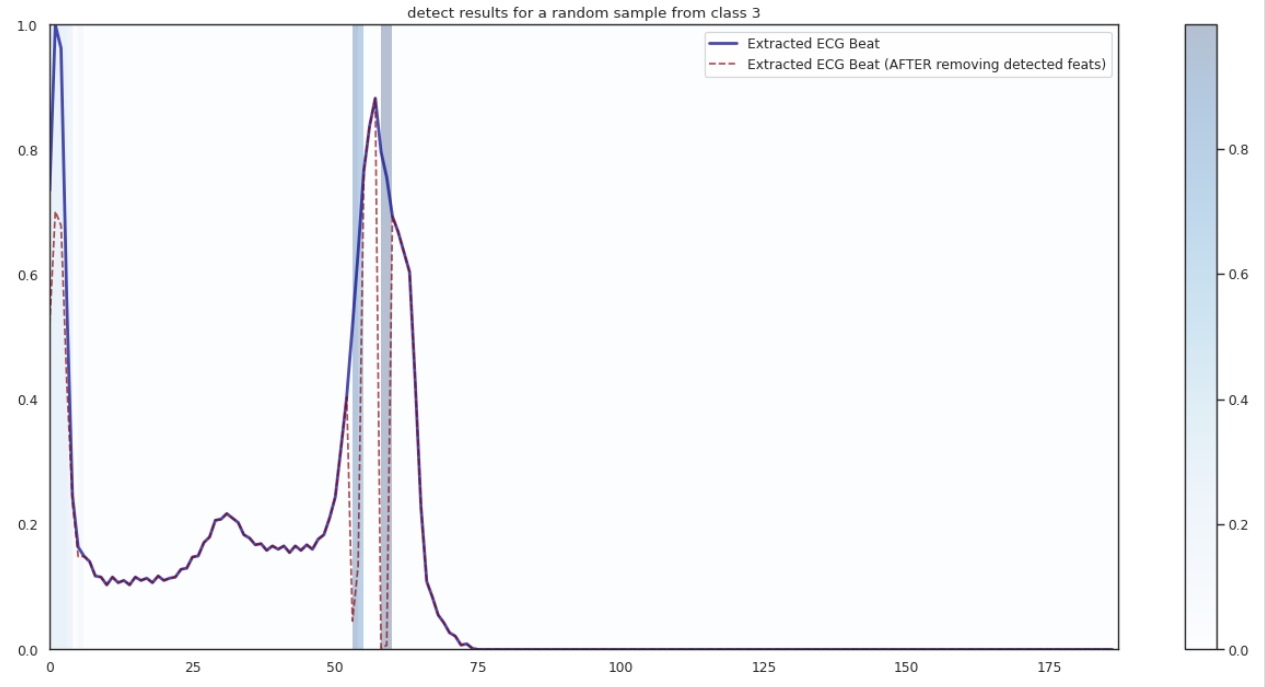


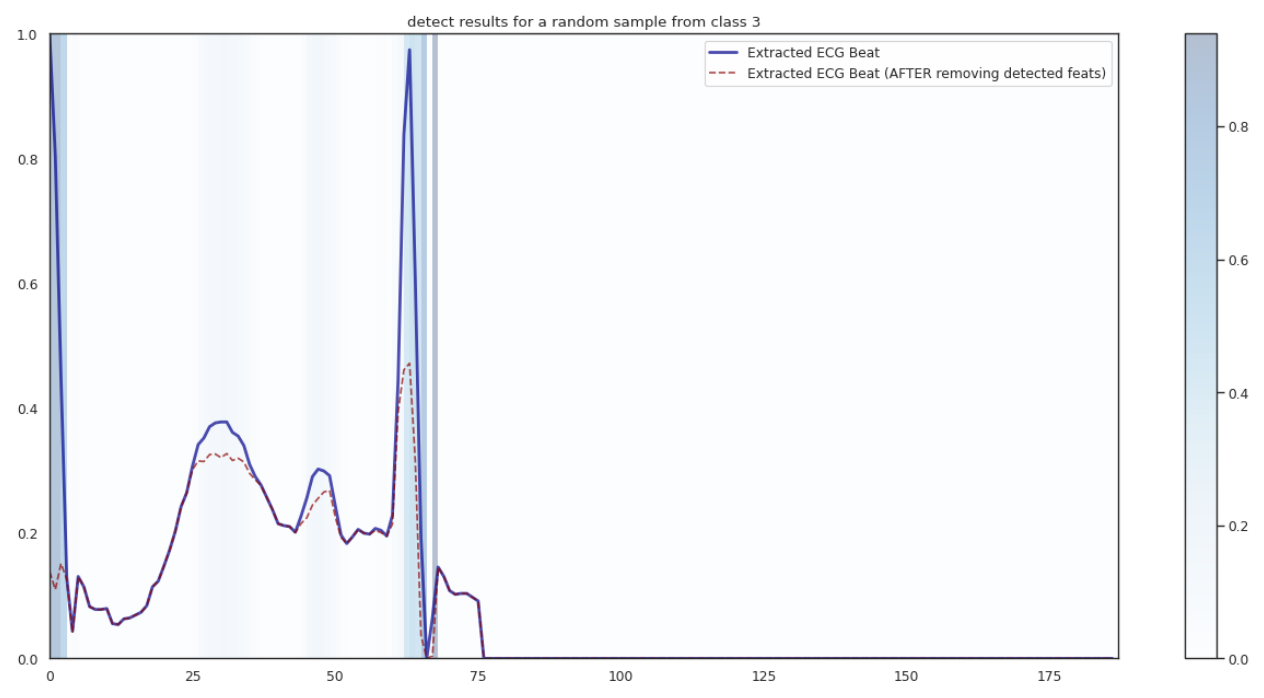

```
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 12ms/step
```

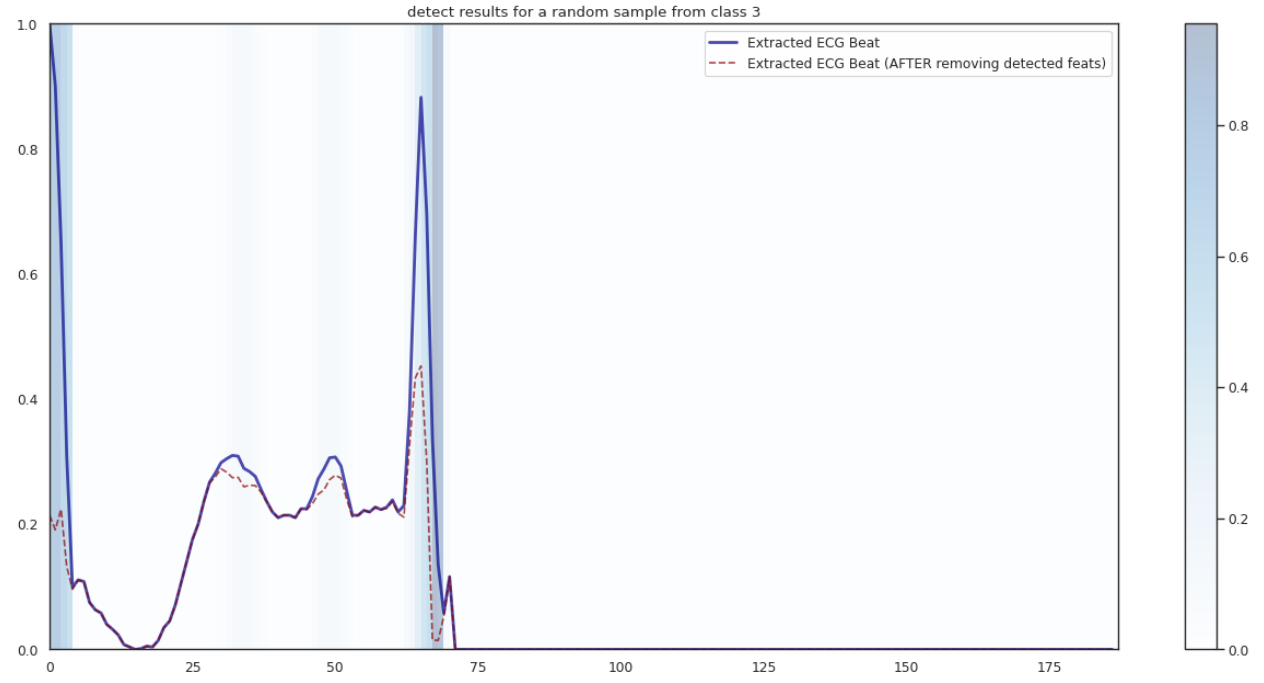

detect results for a random sample from class 2
Extracted ECG Beat
Extracted ECG Beat (AFTER removing detected feats)
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
detect results for a random sample from class 2
Extracted ECG Beat
Extracted ECG Beat (AFTER removing detected feats)



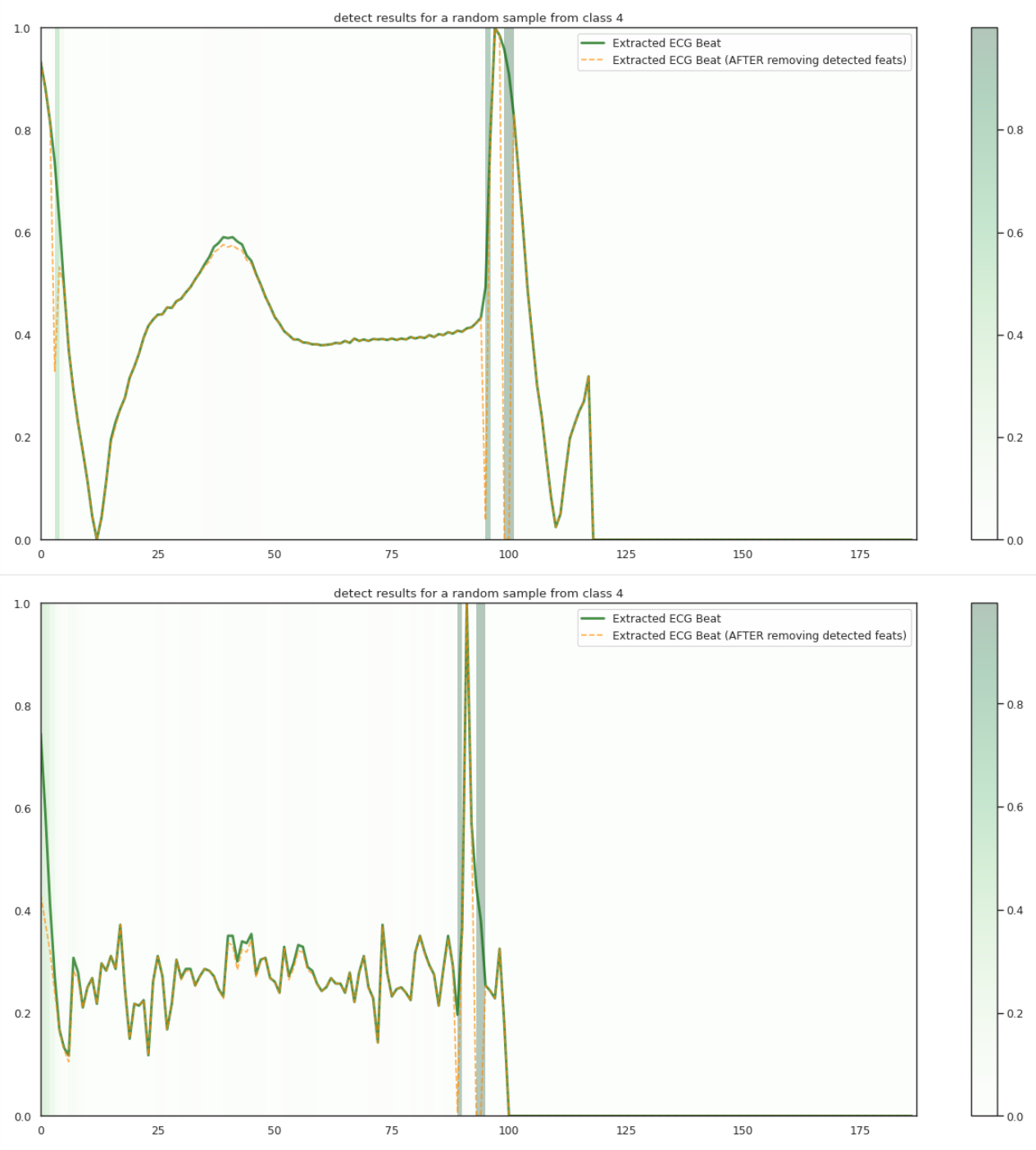

```
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 12ms/step
```

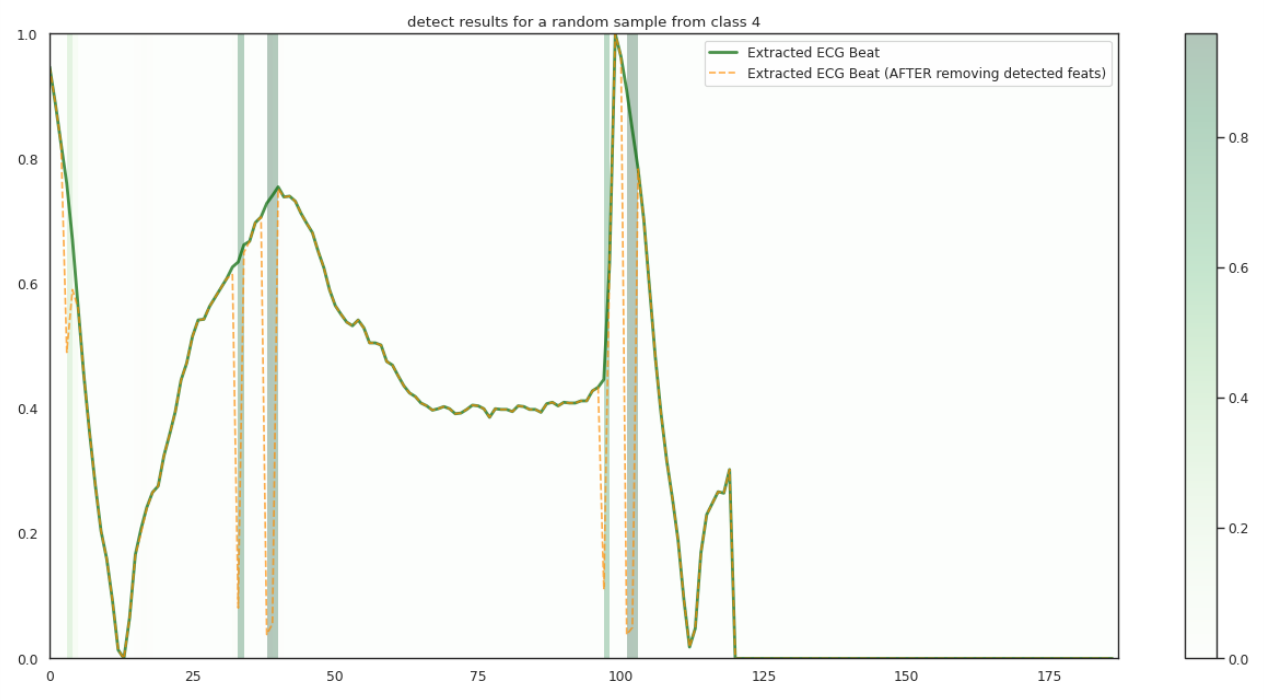







```
1/1 [==============================] - 0s 12ms/step
1/1 [==============================] - 0s 12ms/step
```


detect results for a random sample from class 4
Extracted ECG Beat
Extracted ECG Beat (AFTER removing detected feats)
detect results for a random sample from class 4
Extracted ECG Beat
Extracted ECG Beat (AFTER removing detected feats)



Results

- The localized regions of ECG complexes in sinus rhythm are most informative in distinguishing presence of ventricular ectopic beats from supraventricular ectopic beats in a particular individual. The localized regions lie in the **QRS complex**, which correlates with ventricular depolarization or electrical propagation in the ventricles. Ion channel aberrations and structural abnormalities in the ventricles can affect electrical conduction in the ventricles, manifesting with subtle anomalies in the QRS complex in sinus rhythm that **may not be discernible by the naked eye but is detectable by the convolutional auto-encoder**. Of note, as the R^2 increases from 10% to 88%, the highlighted color bar is progressively broader, covering a higher proportion of the QRS complex. The foregoing observations are sensible: the regions of interest resided in the QRS complex are biologically plausible and **consistent with cardiac electrophysiological principles**.
- As the R2 increases from 80% to 84% and finally 88%, the blue bar progressively highlights the P wave of ECG complexes in sinus rhythm. This observation is **consistent with our understanding of the mechanistic underpinnings of atrial depolarization**, which correlates with the P wave. Ion channel alterations and structural changes in the atria can affect electrical conduction in the atria manifesting with subtle anomalies in the P wave in sinus rhythm that may not be discernible by the naked eye but are detectable by the convolutional auto-encoder.
- Collectively, the examples underscore the fact that the discriminative regions of interest identified by our proposed method are biologically plausible and consistent with cardiac electrophysiological principles while **locating subtle anomalies in the P wave and QRS complex that may not be discernible by the naked eye**. By inspecting our results with an ECG clinician Dr. Lin Yee Chen, the localized discriminative features of the ECG are consistent with medical interpretation in ECG diagnosis.

3.2.2 MNIST dataset

Train MNIST localization network

[2]:
```
from keras.datasets import mnist
from keras.optimizers import Adam, SGD
from keras.callbacks import EarlyStopping, ReduceLROnPlateau

from dnn_locate import Dnn_Locate
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
```

[3]:
```
## load data
np.random.seed(3)
tf.random.set_seed(3)

input_shape, labels = (28, 28, 1), 10

## load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train / 255.
X_test = X_test / 255.

ind_set = np.array([i for i in range(len(y_train)) if y_train[i] in [7, 9]])
ind_set_test = np.array([i for i in range(len(y_test)) if y_test[i] in [7, 9]])

X_train, y_train = X_train[ind_set], y_train[ind_set]
X_test, y_test = X_test[ind_set_test], y_test[ind_set_test]

X_train = np.expand_dims(X_train, axis=3)
X_test = np.expand_dims(X_test, axis=3)
```

[ ]:
```
## define models
from keras import initializers
from keras.models import Sequential, Model
from keras.layers import UpSampling2D, Conv2D
from keras.layers import Input, Dense, Reshape, Flatten, Dropout, Add, Multiply,␣
↪Conv2DTranspose
from keras.layers import BatchNormalization, Activation, ZeroPadding2D, MaxPooling2D,␣
↪GlobalAveragePooling2D

initializer = initializers.glorot_uniform(seed=0)

## define the detector before TRELU activation
detector = Sequential()
detector.add(Conv2D(32, (2,2),
    padding="same",
    input_shape=input_shape,
    kernel_initializer=initializer,
    bias_initializer=initializer))
detector.add(Flatten())
```

(continues on next page)

(continued from previous page)

```
detector.add(Dense(128, activation='relu',
    kernel_initializer=initializer,
    bias_initializer=initializer))
detector.add(Dense(128, activation='relu',
    kernel_initializer=initializer,
    bias_initializer=initializer))
detector.add(Dense(np.prod(input_shape),
    activation ='softmax',
    kernel_initializer=initializer,
    bias_initializer=initializer))
detector.add(Reshape(input_shape))

## define discriminator
discriminator = Sequential()
discriminator.add(Conv2D(32, (3, 3),
            activation='relu', name='last_conv',
            kernel_initializer=initializer,
            bias_initializer=initializer,
            kernel_regularizer=tf.keras.regularizers.l1(0.001),
            bias_regularizer=tf.keras.regularizers.l1(0.001),
            input_shape=input_shape))
discriminator.add(MaxPooling2D((2, 2)))
discriminator.add(Flatten())
discriminator.add(Dense(100, activation='relu',
    kernel_regularizer=tf.keras.regularizers.l1(0.001),
    bias_regularizer=tf.keras.regularizers.l1(0.001),
    kernel_initializer=initializer))
discriminator.add(Dense(labels, activation='softmax',
    kernel_initializer=initializer,
    kernel_regularizer=tf.keras.regularizers.l1(0.001),
    bias_regularizer=tf.keras.regularizers.l1(0.001),
    bias_initializer=initializer))
discriminator.compile(loss='sparse_categorical_crossentropy',
                                        optimizer=Adam(lr=0.001),
                                        metrics=['accuracy'])
```

[ ]:
```
## define framework
tau_range = [4., 6., 8., 10., 12., 14., 16., ]
shiing = Dnn_Locate(input_shape=input_shape,
                            discriminator=discriminator,
                            tau_range=tau_range,
                            task='classification')

es_detect1 = ReduceLROnPlateau(monitor="loss", factor=0.382, min_lr=.0001,
                                    verbose=1, patience=5, mode="min")
es_detect2 = EarlyStopping(monitor='loss', mode='min', min_delta=.0001,
                                        verbose=1, patience=15, restore_best_
↪weights=True)
es_learn = EarlyStopping(monitor='val_accuracy', mode='max',
                                        verbose=1, patience=10, restore_best_
↪weights=True)
```

[ ]:
```
print('###'*20)
print('###'*5+' '*6+'Load learner'+' '*5+'###'*5)
print('###'*20)

# learn_tmp = shiing.discriminator.fit(x=X_train, y=y_train, callbacks=[es_learn],␣
↪epochs=50, batch_size=128, validation_split=.2)
# shiing.discriminator.save_weights("./saved_model/model1107.h5")
# shiing.discriminator.load_weights("./saved_model/model1107.h5")
shiing.discriminator.load_weights("../tests/saved_model/model1119.h5")
# shiing.discriminator.load_weights("./saved_model/model1126.h5")

print('###'*20)
print('#'*16+' '*5+'Train detector'+' '*5+'#'*16)
print('###'*20)

## fit detector for a range of tau
fit_params={'callbacks': [es_detect1, es_detect2],
                    'epochs': 1000, 'batch_size': 64}

shiing.fit(X_train=X_train, y_train=y_train, detector=detector,
                    optimizer=SGD(lr=1.), fit_params=fit_params)
```

[ ]:
```
## Visualize the results
shiing.R_sqaure_path()
shiing.path_plot()
shiing.DA_plot(X=X_test, y=y_test)
```

3.3 API Reference

3.3.1 dnn_locate.loc_model

loc_model

class dnn_locate.loc_model.**loc_model**(*input_shape*, *discriminator*, *localizer_backend=None*, *tau_range=array([1, 2, 3, 4, 5, 6, 7, 8, 9])*, *target_r_square='auto'*, *r_metric='acc'*, *activation='tanh+relu'*, *save_path=False*)

Bases: `object`

class for discriminative feature detection for deep learning models.

Parameters

input_shape
[{tuple-like} (shape of the feature/image)] For example, in MNIST dataset `input_shape = (28, 28, 1)`.

localizer_backend: {keras-defined neural network}
A backend neural network before reshape and Truncated ReLU activation.

discriminator: {keras-defined neural network}
A pretrained neural network needs to be explained.

tau_range
[{list-like}] List of tau to define the localizers.

target_r_square: {float, [0,1]}
A target R_square to be explained.

R_square: {list-like}
Records for R_square values based on a dataset.

`DA_plot`(*X*, *y*, *demo_ind=None*, *threshold=None*, *plt1_params={'cmap': 'binary'}*, *plt2_params={'cmap': 'OrRd'}*)

Plots data-adaptive detected region for the fitted localizer.

Parameters

X
[{array-like} of shape (n_samples, dim_features)] Instances matrix/tensor, where n_samples in the number of samples and dim_features is the dimension of the features.

y
[{array-like} of shape (n_samples,)] Output vector/matrix relative to X.

demo_ind
[{array-like} of shape (num_instance, num_labels) default=None]

threshold
[{array-like} or float, default=None] threshold to truncate the small detected pixels

`R_sqaure_path`()

Plot solution path for the proposed method wrt tau_range

`R_square`(*X*, *y*)

Report R_square for the fitted localizer based on a given dataset

Parameters

X
[{array-like} of shape (n_samples, dim_features)] Instances matrix/tensor, where n_samples in the number of samples and dim_features is the dimension of the features.

y
[{array-like} of shape (n_samples,)] Output vector/matrix relative to X.

`build_combined`(*optimizer=<keras.optimizers.optimizer_v2.adam.Adam object>*)

Building a localizer and a combined model for the proposed framework

Parameters

localizer: {keras-defined neural network}
A neural network for localizer.

optimizer: {keras-defined optimizer: ``tensorflow.keras.optimizers``}, default = 'SGD(lr=.0005)'
A optimizer used to train the localizer.

`build_localizer`(*tau=10.0*, *max_value=1.0*)

Building a localizer for the proposed framework

Parameters

tau: {float}
The magnitude of the localizer.

max_value: {float, (0, 1]}
The maximum proportion of the localizer.

`fit`(*X_train*, *y_train*, *fit_params*, *datagen=None*, *demo_ind=None*, *X_test=None*, *y_test=None*, *optimizer=<keras.optimizers.optimizer_v2.gradient_descent.SGD object>*)

Fitting the localizer based on a given dataset.

Parameters

X_train
[{array-like} of shape (n_samples, dim_features)] Instances matrix/tensor, where n_samples in the number of samples and dim_features is the dimension of the features.

y_train
[{array-like} of shape (n_samples,)] Output vector/matrix relative to X.

X_test
[{array-like} of shape (n_samples, dim_features), default = None] Instances features to compute the r_square. If `None`, X_test = X_train

y_test
[{array-like} of shape (n_samples,), default = None] Output to compute the r_square. If `None`, y_test = y_train

localizer
[{keras-defined neural network}] A neural network for localizer before Truncated RELU activation.

fit_params: {dict of fitting parameters}

See keras `fit`: (https://keras.rstudio.com/reference/fit.html), including `batch_size`, `epoch`, `callbacks`,
`validation_split`, `validation_data`, and so on.

demo_ind
[{array-like} default=None] The index set for demonstrated instances.

optimizer: {keras-defined optimizer: ``tensorflow.keras.optimizers``}, default = 'SGD(lr=.0005)'
A optimizer used to train the localizer.

`locate`(*X*)

Return the localized discriminative features by the fitted localizer.

Parameters

X
[{array-like} of shape (n_samples, dim_features)] Instances matrix/tensor, where n_samples in the number of samples and dim_features is the dimension of the features.

`path_plot`(*threshold=None*, *plt1_params={'cmap': 'binary'}*, *plt2_params={'cmap': 'OrRd'}*)

Plots generalized partial R values and its corresponding images wrt tau_range.

Parameters

threshold
[{array-like} or float] threshold to truncate the small detected pixels

plt1_params
[{dict-like}] dict for imshow for X_demo

plt2_params
[{dict-like}] dict for imshow for X_diff_demo

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- search

INDEX