3_calculate_and_display_det_curves

February 6, 2020

1 Generate DET Curves specific to the different subgroups of the Balance Faces in the Wild (BFW) dataset.

Uses the data in `data/bfw-datatable.pkl` to evaluate DET curves of different attributes.

[1]:
```
import pathlib
path_package=f'../'
import sys
if path_package not in sys.path:
    sys.path.append(path_package)
```

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick

import seaborn as sns

sns.set(font_scale=1.1)
# Load out custom tool for loading and processing the data
from facebias.iotools import load_bfw_datatable, makedir
from facebias.visualization import draw_det_curve
from facebias.metrics import calculate_det_curves

%matplotlib inline
```

1.1 Setings for notebook run

set values per preferences for current session

[4]:
```
fontsize=12

# plotting
sns.set_style("whitegrid")
sns.set_context("paper", rc={"font.size": fontsize, "axes.titlesize": fontsize,␣
 ↪"axes.labelsize": fontsize})

# parameters and filepaths
```

```
ticks_to_use_x=(1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1e-0)

# datatable (See Load Data)
dir_data = '../../data/bfw-data/bfw/'
dir_features = f'{dir_data}features/sphereface/'
f_datatable = f'{dir_data}meta/bfw-v0.1.5-datatable.pkl'
use_feature = 'sphereface'

dir_results = f"../../results/{use_feature}/"
makedir(dir_results)


save_plots = True # save plots to disc, in folder 'dir_out'
save_intermediate = True # save data to replot det curves without waiting for␣
 ↪calculations
overwrite_existing = False # only save if files do not exist

opts = [{'color': 'g', 'alpha': 0.7, 'norm_hist': True}, {'color': 'r', 'alpha':
 ↪ 0.7, 'norm_hist': True}] # currently not used
```

```
/Users/jrobby/WORK/src/facebias/code/facebias/iotools.py:20: UserWarning:
Directory ../../results/sphereface/ exists
  warnings.warn(f"Directory {din} exists")
```

1.2 Load the data

Read in the data as a pandas.DataFrame and display the first few rows.

More information and process to build datatable is exemplified in 0_prepare_datatable.ipynb.

Scores for pairs are assumed to be calculated and added as column of datatable. The process for adding this column is demonstrated in 1_compare_features.ipynb

Note that the demo uses scores from setnet50, as it creates a column *score* with the respective values. Just set *score* column to any set of scores that are intended for analysis.

[8]:
```
data = load_bfw_datatable(f_datatable)
data['score'] = data[use_feature]
data.head()
```

[8]:

	fold	p1	…	sphereface	score
0	1	asian_females/n000009/0010_01.jpg	…	0.392526	0.392526
1	1	asian_females/n000009/0010_01.jpg	…	0.354262	0.354262
2	1	asian_females/n000009/0010_01.jpg	…	0.302028	0.302028
3	1	asian_females/n000009/0010_01.jpg	…	-0.009217	-0.009217
4	1	asian_females/n000009/0010_01.jpg	…	0.132534	0.132534

[5 rows x 19 columns]

[9]:
```
classes_abbreviated = np.unique(list(np.unique(data.a1)) + list(np.unique(data.
↪a2)))
classes_abbreviated.sort()

print(f"there are {len(classes_abbreviated)} types: {classes_abbreviated}")
```

```
there are 8 types: ['AF' 'AM' 'BF' 'BM' 'IF' 'IM' 'WF' 'WM']
```

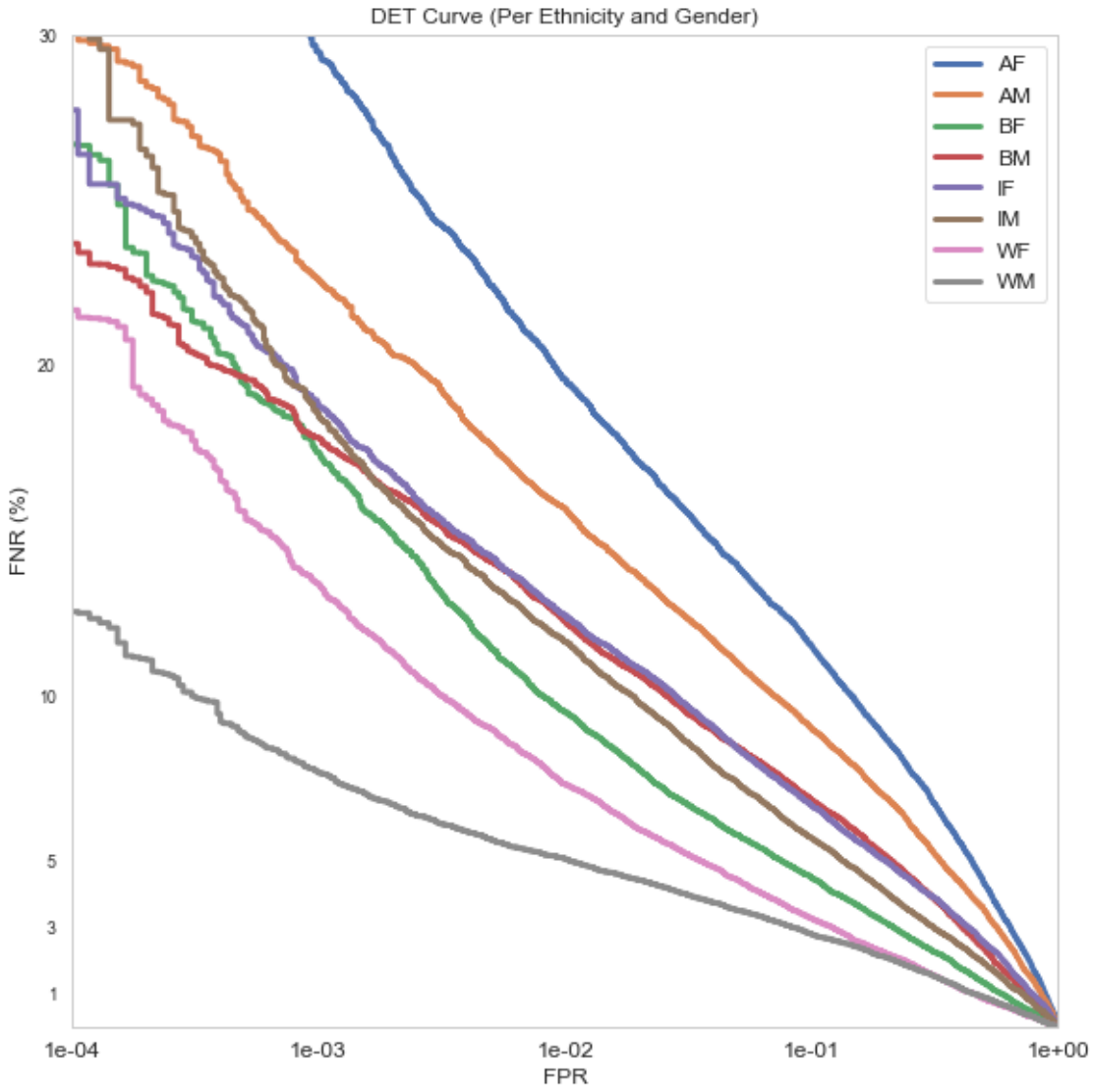
[10]:
```
subgroups = data.groupby('a1')
li_subgroups = subgroups.groups
```

[13]:
```
fig = plt.figure(figsize=(8, 8), constrained_layout=True)
gs = fig.add_gridspec(1, 3)

ax1 = fig.add_subplot(gs[0, :])
for i, subgroup in enumerate(li_subgroups):
    # for each subgroup
    fout = f"{dir_results}/det_data_{subgroup}.pkl"
    if pathlib.Path(fout).is_file() and not overwrite_existing:
        det_data = pd.read_pickle(fout)
        fpr, fnr, thresholds = det_data['fpr'], det_data['fnr'],␣
↪det_data['thresholds']
    else:
        df_subgroup = subgroups.get_group(subgroup)
        labels, scores = df_subgroup['label'].values.astype(int),␣
↪df_subgroup['score'].values
        fpr, fnr, thresholds = calculate_det_curves(labels, scores)
        if save_intermediate:
            pd.to_pickle({'fpr': fpr, 'fnr': fnr, 'thresholds': thresholds},␣
↪fout)
    ax1 = draw_det_curve(fpr, fnr, ax=ax1, label=subgroup,␣
↪fontsize=fontsize,title='DET Curve (Per Ethnicity and Gender)')

ax1.xaxis.set_major_formatter(mtick.FormatStrFormatter('%.e'))
plt.minorticks_off()
ax1.set_ylabel('FNR (%)',fontsize=fontsize)
ax1.set_xlabel('FPR', fontsize=fontsize)
plt.legend(fontsize=fontsize)
ax1.set_xlim([1e-4, 1])
ax1.set_ylim([0, 30])

for tick in ax1.xaxis.get_major_ticks():
    tick.label.set_fontsize(12)
plt.grid(False)
if save_plots and overwrite_existing:
    plt.savefig(f"{dir_results}curve_subgroups.pdf")
    plt.savefig(f"{dir_results}curve_subgroups.png")
```



[14]:
```
subgroups = data.groupby('g1')
li_subgroups = subgroups.groups
classes_abbreviated=list(li_subgroups.keys())
print(f"there are {len(classes_abbreviated)} types: {classes_abbreviated}")
```

```
there are 2 types: ['F', 'M']
```

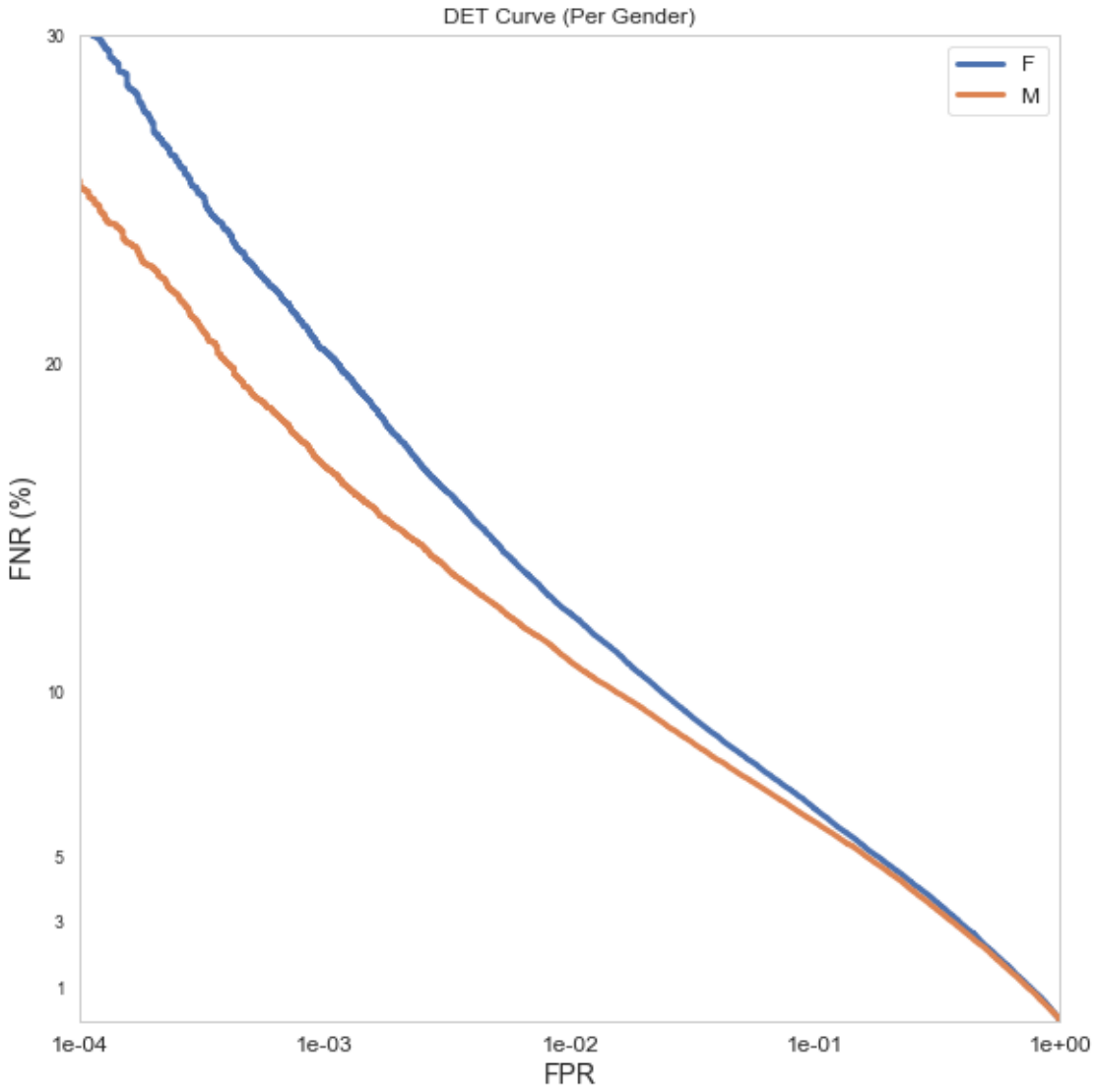
[16]:
```
fig = plt.figure(figsize=(8, 8), constrained_layout=True)
gs = fig.add_gridspec(1, 3)

ax1 = fig.add_subplot(gs[0, :])
for i, subgroup in enumerate(li_subgroups):
    # for each subgroup
```

```
    fout = f"{dir_results}/det_data_{subgroup}.pkl"
    if pathlib.Path(fout).is_file() and not overwrite_existing:
        det_data = pd.read_pickle(fout)
        fpr, fnr, thresholds = det_data['fpr'], det_data['fnr'],␣
↪det_data['thresholds']
    else:
        df_subgroup = subgroups.get_group(subgroup)
        labels, scores = df_subgroup['label'].values.astype(int),␣
↪df_subgroup['score'].values
        fpr, fnr, thresholds = calculate_det_curves(labels, scores)
        if save_intermediate:
            pd.to_pickle({'fpr': fpr, 'fnr': fnr, 'thresholds': thresholds},␣
↪fout)
    ax1 = draw_det_curve(fpr, fnr, ax=ax1, label=subgroup, fontsize=fontsize,␣
↪title='DET Curve (Per Gender)')

ax1.xaxis.set_major_formatter(mtick.FormatStrFormatter('%.e'))
plt.minorticks_off()
ax1.set_ylabel('FNR (%)',fontsize=fontsize+2)
ax1.set_xlabel('FPR', fontsize=fontsize+2)
plt.legend(fontsize=fontsize)
ax1.set_xlim([1e-4, 1])
ax1.set_ylim([0, 30])
for tick in ax1.xaxis.get_major_ticks():
    tick.label.set_fontsize(12)
plt.grid(False)
if save_plots and overwrite_existing:

    plt.savefig(f"{dir_results}/curve_genders.pdf")
    plt.savefig(f"{dir_results}/curve_genders.png")
```



[17]:
```
subgroups = data.groupby('e1')
li_subgroups = subgroups.groups
classes_abbreviated=list(li_subgroups.keys())
print(f"there are {len(classes_abbreviated)} types: {classes_abbreviated}")
```

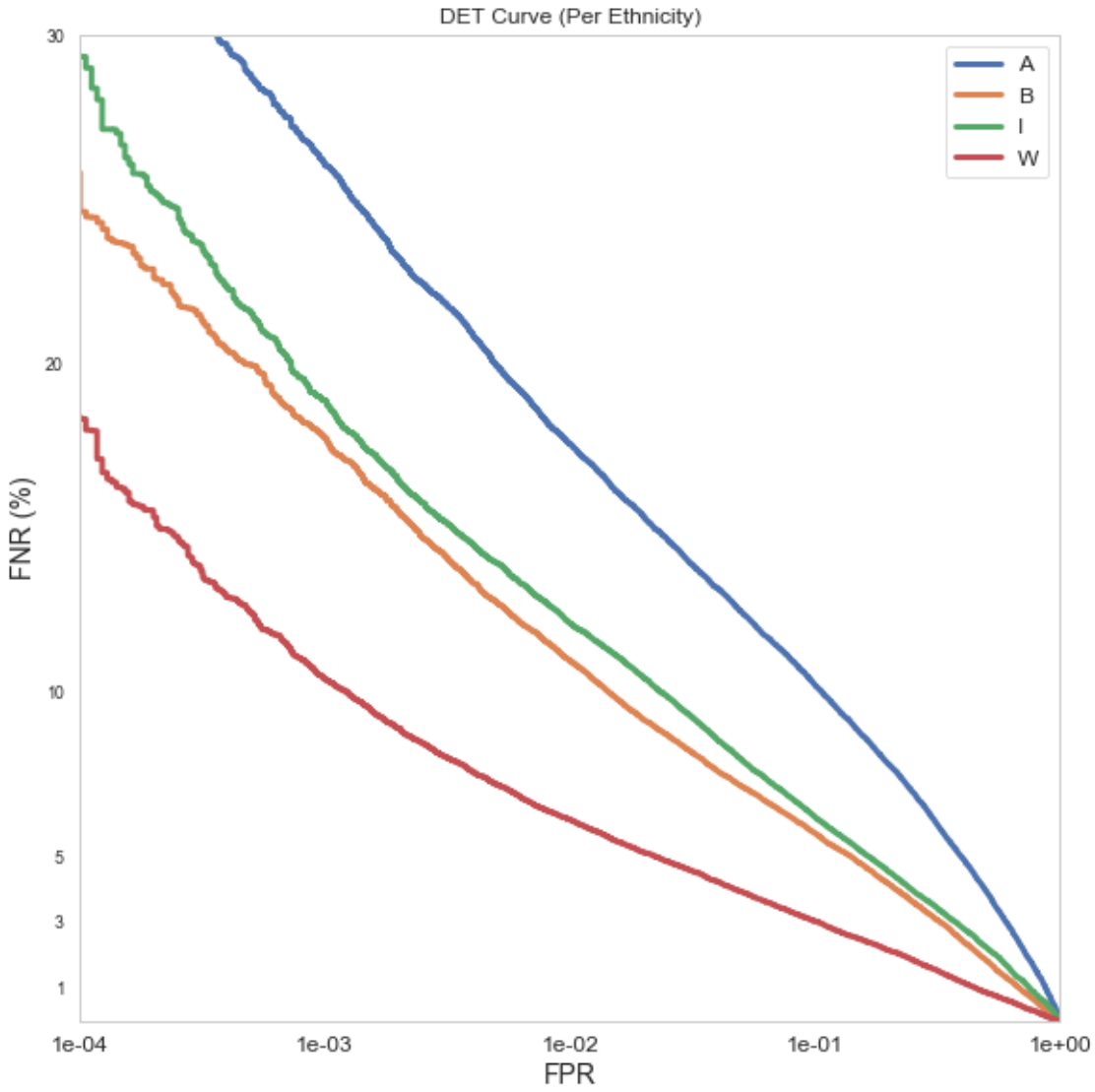
```
there are 4 types: ['A', 'B', 'I', 'W']
```

[18]:
```
fig = plt.figure(figsize=(8, 8), constrained_layout=True)
gs = fig.add_gridspec(1, 3)

ax1 = fig.add_subplot(gs[0, :])
for i, subgroup in enumerate(li_subgroups):
    # for each subgroup
```

```
    fout = f"{dir_results}/det_data_{subgroup}.pkl"
    if pathlib.Path(fout).is_file() and not overwrite_existing:
        det_data = pd.read_pickle(fout)
        fpr, fnr, thresholds = det_data['fpr'], det_data['fnr'],␣
↪det_data['thresholds']
    else:
        df_subgroup = subgroups.get_group(subgroup)
        labels, scores = df_subgroup['label'].values.astype(int),␣
↪df_subgroup['score'].values
        fpr, fnr, thresholds = calculate_det_curves(labels, scores)
        if save_intermediate:
            pd.to_pickle({'fpr': fpr, 'fnr': fnr, 'thresholds': thresholds},␣
↪fout)
    ax1 = draw_det_curve(fpr, fnr, ax=ax1, label=subgroup, fontsize=fontsize,␣
↪title='DET Curve (Per Ethnicity)')

ax1.xaxis.set_major_formatter(mtick.FormatStrFormatter('%.e'))
plt.minorticks_off()
ax1.set_ylabel('FNR (%)',fontsize=fontsize+2)
ax1.set_xlabel('FPR', fontsize=fontsize+2)
plt.legend(fontsize=fontsize)
ax1.set_xlim([1e-4, 1])
ax1.set_ylim([0, 30])
for tick in ax1.xaxis.get_major_ticks():
    tick.label.set_fontsize(12)
plt.grid(False)
if save_plots and overwrite_existing:
    plt.savefig(f"{dir_results}/curve_ethnicity.pdf")
    plt.savefig(f"{dir_results}/curve_ethnicity.png")
```


DET Curve (Per Ethnicity)
A
B
I
W
30
20
10
5
3
1
FNR (%)
1e-04
1e-03
1e-02
1e-01
1e+00
FPR