Robustness_untargeted

May 14, 2021

[ ]:
```
from google.colab import drive
import os
import sys
import numpy as np
import matplotlib.pyplot as plt
%load_ext autoreload
%autoreload 2
```

[ ]:
```
drive.mount('/content/gdrive', force_remount=True)
```

```
Mounted at /content/gdrive
```

[ ]:
```
sys.path.append('/content/gdrive/My Drive/')
import fp_utilities as fp_util
import plot_utilities as plot_util
```

1 Noise Injection

[ ]:
```
# attack strength = norm of noise being inserted = relative norm * average norm␣
 ↪of test images
rel_norms = np.arange(0.0001, 0.001, 0.0004).tolist() + np.arange(0.001, 0.01,␣
 ↪0.004).tolist() + np.arange(0.01, 0.1, 0.04).tolist() + np.arange(0.1, 0.4,␣
 ↪0.1).tolist()
atk_strengths = [rel_norm*fp_util.avg_l2_norm_imgs for rel_norm in rel_norms]
```

[ ]:
```
from sklearn.metrics import accuracy_score
marra_accuracies = []
yu_accuracies = []
```

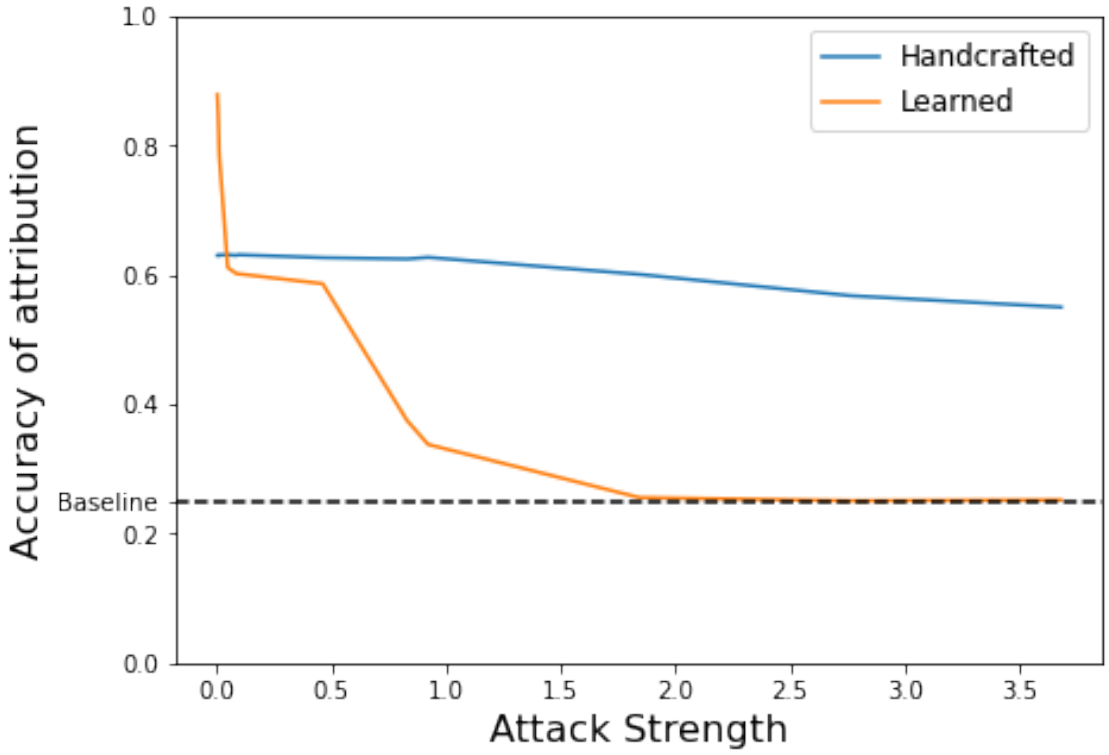
[ ]:
```
# calculate accuracy of attribution at different attack strengths
for atk_strength in atk_strengths:
  marra_preds = fp_util.get_predictions('Marra', attack_mode="gaussian",␣
 ↪attack_strength=atk_strength)
  marra_accuracies.append(accuracy_score(fp_util.ground_truth, marra_preds))
  yu_preds = fp_util.get_predictions('Yu', attack_mode="gaussian",␣
 ↪attack_strength=atk_strength)
```

```
  yu_accuracies.append(accuracy_score(fp_util.ground_truth, yu_preds))
```

```
/usr/local/lib/python3.7/dist-packages/torch/nn/functional.py:1709: UserWarning:
nn.functional.sigmoid is deprecated. Use torch.sigmoid instead.
  warnings.warn("nn.functional.sigmoid is deprecated. Use torch.sigmoid
instead.")
```

[ ]:
```
np.save('marra_accuracies_gaussian.npy', marra_accuracies)
np.save('yu_accuracies_gaussian.npy', yu_accuracies)
```

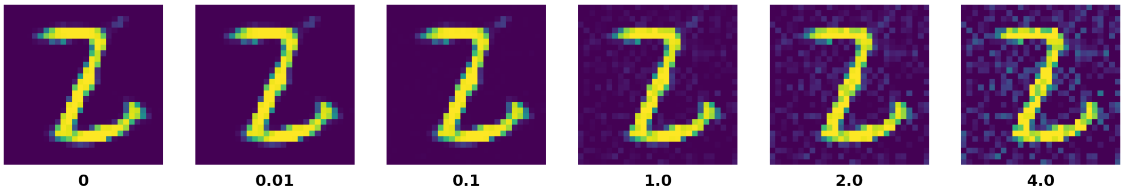
[ ]:
```
import matplotlib.pyplot as plt

plot_util.plot_atk_accuracy_lines(atk_strengths, marra_accuracies,␣
 ↪yu_accuracies)
plt.savefig('gaussian-eval.pdf', bbox_inches = 'tight')
```



[ ]:
```
#load in a test image to visualise the effect of the attack on
test_img = fp_util.load_test_images(1)[2]
atk_strengths_plot = [0, 0.01, 0.1, 1.0, 2.0, 4.0]
atk_images = []

gauss_noise = np.random.normal(0,0.1, test_img.shape)
for atk_strength in atk_strengths_plot:
```

```
  #scale perturbation to achieved desired attack strength
  noise_scaled = gauss_noise * atk_strength/np.linalg.norm(gauss_noise)
  atk_images.append(np.float32(np.clip(test_img + noise_scaled,0,1)))
```

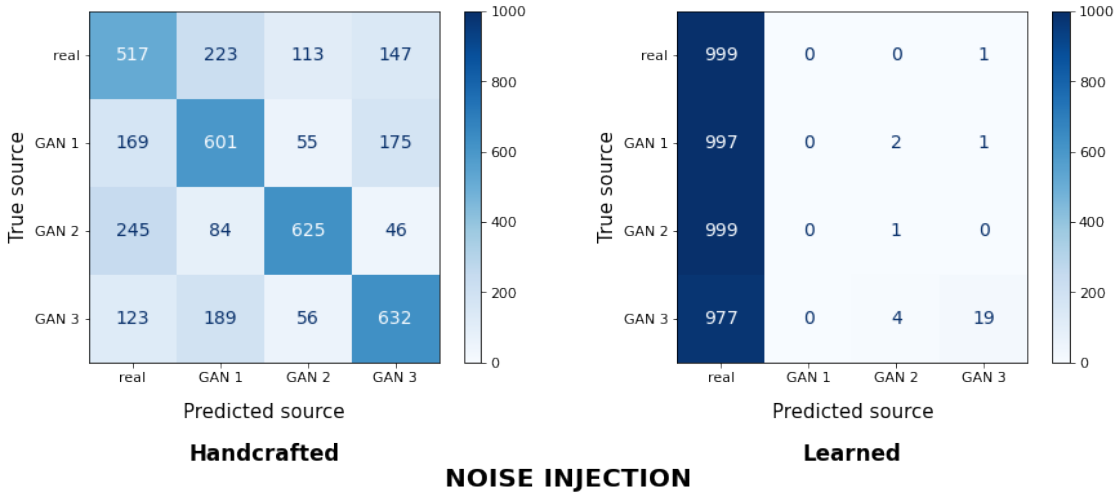
[ ]:
```
plot_util.plot_atk_images(atk_strengths_plot, atk_images)
plt.savefig('gaussian-noise-imgs.pdf', bbox_inches = 'tight')
```



[ ]:
```
#get predictions to visualise as confusion matrices for further analysis
marra_preds = fp_util.get_predictions('Marra', attack_mode="gaussian",␣
 ↪attack_strength=2)
yu_preds = fp_util.get_predictions('Yu', attack_mode="gaussian",␣
 ↪attack_strength=2)
```

```
/usr/local/lib/python3.7/dist-packages/torch/nn/functional.py:1709: UserWarning:
nn.functional.sigmoid is deprecated. Use torch.sigmoid instead.
  warnings.warn("nn.functional.sigmoid is deprecated. Use torch.sigmoid
instead.")
```

[ ]:
```
plt.rcParams['font.size']=14
plt.rcParams['xtick.labelsize'] = 11
plt.rcParams['ytick.labelsize'] = 11

fig, ax = plt.subplots(1,2, figsize = (15,5))

plot_util.plot_confusion_matrix('Handcrafted' , marra_preds, ax[0])
plot_util.plot_confusion_matrix('Learned', yu_preds, ax[1])

plt.suptitle('NOISE INJECTION', y = -0.1, fontsize = 20, fontweight='bold')
plt.savefig('gaussian-conf-matrix.pdf', bbox_inches = 'tight')
```



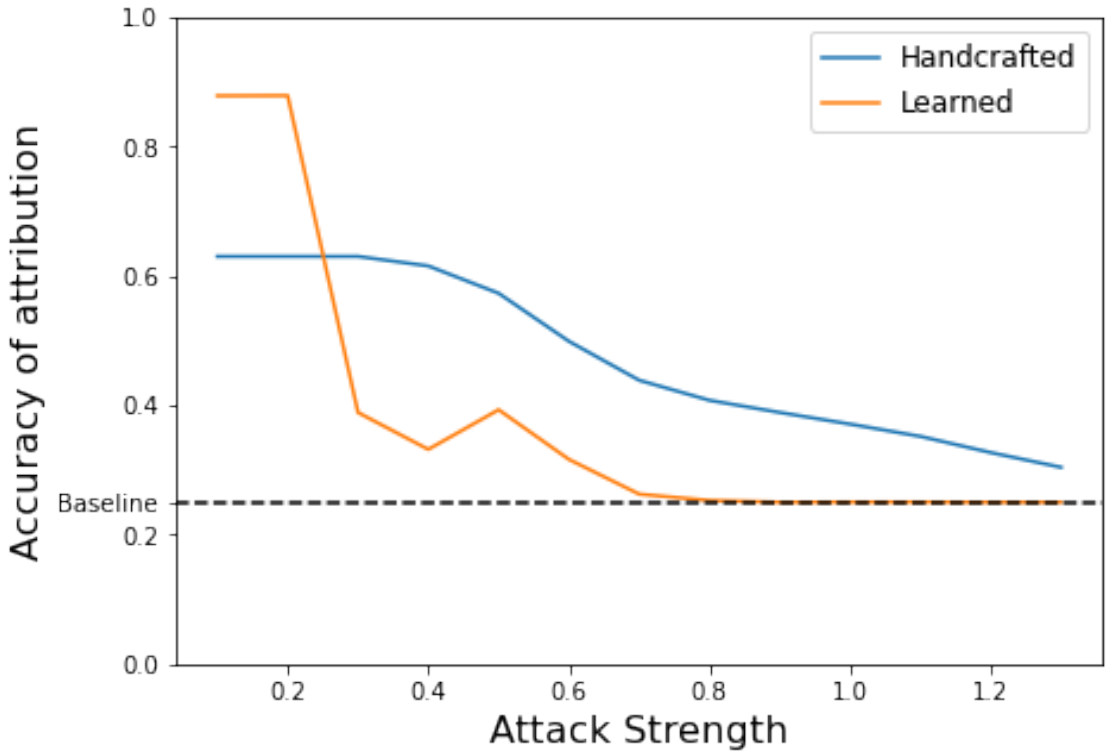
2 Blurring

[ ]:
```
from sklearn.metrics import accuracy_score
marra_accuracies = []
yu_accuracies = []
```

[ ]:
```
#attack strength = std dev of gaussian blur being applied
atk_strengths = np.arange(0.1,1.4, 0.1)

# calculate accuracy of attribution at different attack strengths
for atk_strength in atk_strengths:
  marra_preds = fp_util.get_predictions('Marra', attack_mode="blur",␣
 ↪attack_strength=atk_strength)
  marra_accuracies.append(accuracy_score(fp_util.ground_truth, marra_preds))
  yu_preds = fp_util.get_predictions('Yu', attack_mode="blur",␣
 ↪attack_strength=atk_strength)
  yu_accuracies.append(accuracy_score(fp_util.ground_truth, yu_preds))
```

```
/usr/local/lib/python3.7/dist-packages/torch/nn/functional.py:1709: UserWarning:
nn.functional.sigmoid is deprecated. Use torch.sigmoid instead.
  warnings.warn("nn.functional.sigmoid is deprecated. Use torch.sigmoid
instead.")
```

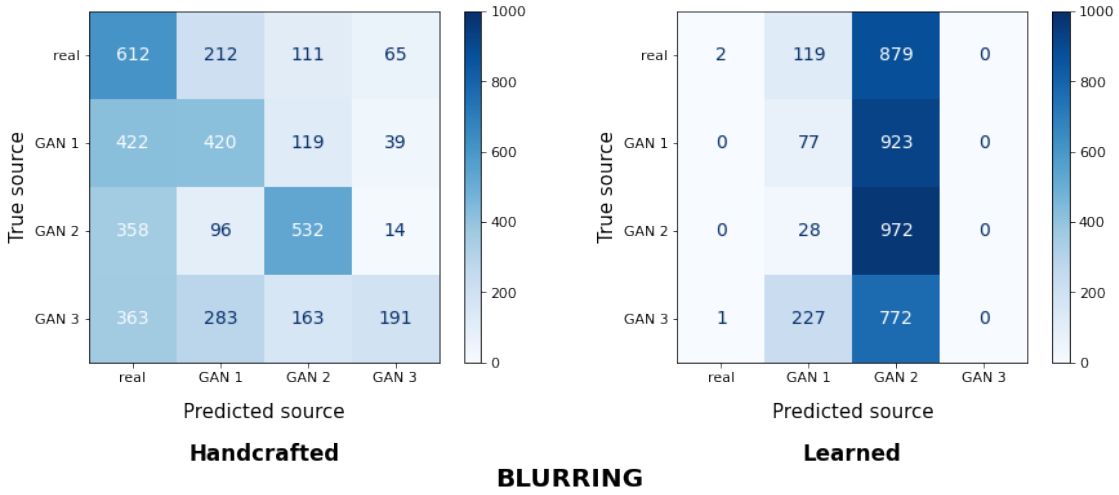
[ ]:
```
import matplotlib.pyplot as plt

plot_util.plot_atk_accuracy_lines(atk_strengths, marra_accuracies,␣
 ↪yu_accuracies)
plt.savefig('blur-eval.pdf', bbox_inches = 'tight')
```



[ ]:
```
#get predictions to visualise as confusion matrices for further analysis
marra_preds = fp_util.get_predictions('Marra', attack_mode='blur',␣
 ↪attack_strength = 0.7)
yu_preds = fp_util.get_predictions('Yu', attack_mode='blur', attack_strength =␣
 ↪0.7)
```

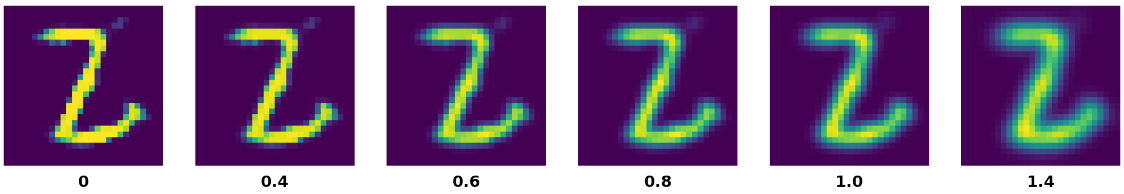
```
/usr/local/lib/python3.7/dist-packages/torch/nn/functional.py:1709: UserWarning:
nn.functional.sigmoid is deprecated. Use torch.sigmoid instead.
  warnings.warn("nn.functional.sigmoid is deprecated. Use torch.sigmoid
instead.")
```

[ ]:
```
plt.rcParams['font.size']=14
plt.rcParams['xtick.labelsize'] = 11
plt.rcParams['ytick.labelsize'] = 11

fig, ax = plt.subplots(1,2, figsize = (15,5))

plot_util.plot_confusion_matrix('Handcrafted' , marra_preds, ax[0])
plot_util.plot_confusion_matrix('Learned', yu_preds, ax[1])

plt.suptitle('BLURRING', y = -0.1, fontsize = 20, fontweight='bold')
plt.savefig('blur-conf-matrix.pdf', bbox_inches = 'tight')
```



[ ]:
```
#load in a test image to visualise the effect of the attack on
test_img = fp_util.load_test_images(1)[2]
atk_strengths_plot = [0, 0.4, 0.6, 0.8, 1.0,1.4]
atk_images = []

for atk_strength in atk_strengths_plot:
  atk_images.append(fp_util.preprocess_blurring(test_img.reshape(1,28,28),␣
 ↪atk_strength)[0].reshape(28,28))
```

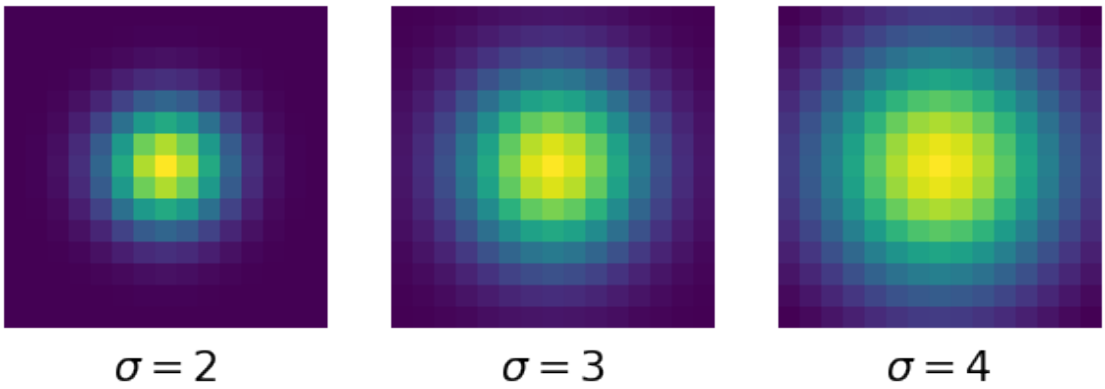
[ ]:
```
plot_util.plot_atk_images(atk_strengths_plot, atk_images)
plt.savefig('blur-images.pdf', bbox_inches='tight')
```



[ ]:
```
from scipy import signal

def get_gaussian_filter(size, std):
    filter_1d = signal.gaussian(size, std=std).reshape(size, 1)
    return np.outer(filter_1d, filter_1d)
```

[ ]:
```
#visualise the filter used to implement blurring at different attack strengths

std_list = [2,3, 4]
fig, ax = plt.subplots(1,3, figsize=(10, 5))
```

```
for i in range(3):
  ax[i].imshow(get_gaussian_filter(15, std_list[i]), interpolation='none')
  ax[i].axis('off')

  ttl = ax[i].title
  ttl.set_text("$\sigma = {}$".format(std_list[i]))
  ttl.set_fontsize(22)
  ttl.set_position([0.5, -0.2])

plt.savefig('gaussian-std_vis.pdf', bbox_inches='tight')
```



[ ]: