P4_RajasekahranPillai_MualiKrishnan_AML

November 19, 2019

Student Name: Murali Krishnan Rajasekharan Pillai

ECE 595 Machine Learning II

Project 4: Adversarial Machine Learning

[1]:
```
#Install Cleverhans (version Cleverhans 2.1.0 is most compatable with Python 2.x)
!pip install cleverhans==2.1.0
```

```
Collecting cleverhans==2.1.0
  Downloading https://files.pythonhosted.org/packages/cc/91/1f6f2d1f1bf226
8dcebd097347b4d8fa952743ed46b6fb160a314ca67287/cleverhans-2.1.0-py3-none-any.whl
(74kB)
     || 81kB 2.7MB/s
Collecting pycodestyle
  Downloading https://files.pythonhosted.org/packages/0e/0c/04a353e104d2f3
24f8ee5f4b32012618c1c86dd79e52a433b64fceed511b/pycodestyle-2.5.0-py2.py3-none-
any.whl (51kB)
     || 51kB 8.0MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-
packages (from cleverhans==2.1.0) (1.17.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages
(from cleverhans==2.1.0) (1.3.2)
Collecting mnist~=0.2
  Downloading https://files.pythonhosted.org/packages/c6/c4/5db3bfe009f8d71f1d53
2bbadbd0ec203764bba3a469e4703a889db8e5e0/mnist-0.2.2-py2.py3-none-any.whl
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-
packages (from cleverhans==2.1.0) (3.1.1)
Collecting nose
  Downloading https://files.pythonhosted.org/packages/15/d8/dd071918c040f5
0fa1cf80da16423af51ff8ce4a0f2399b7bf8de45ac3d9/nose-1.3.7-py3-none-any.whl
(154kB)
     || 163kB 8.7MB/s
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.6/dist-packages (from matplotlib->cleverhans==2.1.0)
(0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in
```

```
/usr/local/lib/python3.6/dist-packages (from matplotlib->cleverhans==2.1.0)
(2.6.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.6/dist-packages (from matplotlib->cleverhans==2.1.0)
(2.4.5)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.6/dist-packages (from matplotlib->cleverhans==2.1.0)
(1.1.0)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages
(from cycler>=0.10->matplotlib->cleverhans==2.1.0) (1.12.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-
packages (from kiwisolver>=1.0.1->matplotlib->cleverhans==2.1.0) (41.6.0)
Installing collected packages: pycodestyle, mnist, nose, cleverhans
Successfully installed cleverhans-2.1.0 mnist-0.2.2 nose-1.3.7 pycodestyle-2.5.0
```

[2]:
```
#Import necessary packages
import pickle
import tensorflow as tf
import numpy as np
import keras
from keras import Sequential, backend
from keras.datasets import mnist
from keras.models import load_model, model_from_json
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPool2D,␣
 ↪BatchNormalization
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from cleverhans.utils_keras import KerasModelWrapper
from cleverhans.attacks import FastGradientMethod, MadryEtAl, DeepFool,␣
 ↪CarliniWagnerL2
```

```
<IPython.core.display.HTML object>


Using TensorFlow backend.
```

[3]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20http
s%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.reado
nly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
```

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

[0]:
```
MODEL_LOCATION = './gdrive/My Drive/ece595_ml2/models/'
```

Part 1: Training a target classifier

[5]:
```
# Load data MNIST data and normalize to [0, 1]
(data_train, labels_train), (data_test, labels_test) = mnist.load_data()
data_train = data_train / 255.
data_test = data_test / 255.
```

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
11493376/11490434 [==============================] - 2s 0us/step
```

[0]:
```
#Reshape training and testing data into 784-dimensional vectors
data_train = data_train.reshape(-1, 784)
data_test = data_test.reshape(-1, 784)
```

[0]:
```
#Convert integer labels for training and testing data into one-hot vectors
labels_train = keras.utils.np_utils.to_categorical(labels_train, num_classes=10)
labels_test = keras.utils.np_utils.to_categorical(labels_test, num_classes=10)
```

[0]:
```
#Create classifier architecture, compile it, and train it
class model_methods(object):
  def __init__(self, loss_fn, optim, ndim):
    """
    Try to develop a class which contains common functionality of
    NN models. Like saving a model and it's weights
    """
    self.loss_fn = loss_fn
    self.optim = optim
    self.ndim = ndim

  def save_model_weights(self, h5_file_name):
    """
    Save weights of the model in .h5 format
    Parameters:
      :h5_file_name:    Identifier of the model weights h5 file
    """
    self.model.save_weights(h5_file_name)

  def save_model(self, json_file_name):
    """
    Save the file in a .json file
    Parameters:
      :json_file_name:   Identifier of the model in json file
```

```
    """
    model_json = self.model.to_json()
    with open(json_file_name, 'w') as json_file:
      json_file.write(model_json)

  def load_model(self, json_file_name, h5_file_name):
    json_file = open(MODEL_LOCATION + json_file_name, 'r')
    loaded_from_json = json_file.read()
    json_file.close()
    model = model_from_json(loaded_from_json)
    model.load_weights(MODEL_LOCATION + h5_file_name)
    return model

  def save_model_history(self, model_history, file_name):
    """
    Save model history as a pickle file
    """
    with open(file_name, 'wb') as f:
      pickle.dump(model_history, f)

  def load_model_history(self, pkl_file_name):
    """
    Load model history pickle file
    """
    with open(MODEL_LOCATION + pkl_file_name, 'rb') as f:
      model_history = pickle.load(f)
    return model_history
```

[0]:
```
class CNN(model_methods):
  def __init__(self, loss_fn, optim, ndim, num_classes):
    super().__init__(loss_fn, optim, ndim)
    self.num_classes = num_classes
    self.model = self._build_model()

  def _build_model(self):
    """
    Defines and compiles the architecture
    Parameters:
      :loss_fn:      The loss function used in the model
      :optim:        The optimizer used for the model
    Returns:
      :model:        The compiled model
    """
    model = Sequential()
    model.add(Dense(self.ndim*self.ndim,
                    activation='relu',
                    kernel_initializer='normal'))
```

```
    model.add(BatchNormalization())
    model.add(Dense(100,
                    activation='relu',
                    kernel_initializer='normal'))
    model.add(BatchNormalization())
    model.add(Dense(self.num_classes,
                    activation='softmax'))
    model.compile(loss=self.loss_fn,
                  optimizer=self.optim,
                  metrics=['accuracy'])
    return model

  def fit(self, d_train, d_test,
          n_epochs=100, batch_size=50, display=25):
    """
    Fit the model
    Parameters:
      :d_train:    Tuple of (training data, training labels)
      :d_test:     Tuple of (testing data, testing labels)
      :n_epochs:   Number of epochs for fit
      :batch_size: Number of samples per gradient update
    Returns:
      :model_hist: History object containing all model history info
    """
    data_train, labels_train = d_train
    model_hist = self.model.fit(data_train, labels_train,
                                validation_data=d_test,
                                epochs=n_epochs,
                                batch_size=batch_size,
                                shuffle=True)
    return model_hist
```

[0]:
```
cnn = CNN('categorical_crossentropy', 'adam', 28, 10)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated.
Please use tf.compat.v1.train.Optimizer instead.
```

[0]:
```
cnn_hist = cnn.fit((data_train, labels_train),
                   (data_test, labels_test),
                   n_epochs=50,
                   batch_size=256)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4409: The name tf.random_normal is
deprecated. Please use tf.random.normal instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:148: The name
tf.placeholder_with_default is deprecated. Please use
tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3576: The name tf.log is
deprecated. Please use tf.math.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/ops/math_grad.py:1424: where (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3005: The name tf.Session is
deprecated. Please use tf.compat.v1.Session instead.

Train on 60000 samples, validate on 10000 samples
Epoch 1/50
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:190: The name
tf.get_default_session is deprecated. Please use
tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is
```

```
deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:216: The name
tf.is_variable_initialized is deprecated. Please use
tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:223: The name
tf.variables_initializer is deprecated. Please use
tf.compat.v1.variables_initializer instead.

60000/60000 [==============================] - 7s 119us/step - loss: 0.1955 -
acc: 0.9426 - val_loss: 0.1049 - val_acc: 0.9679
Epoch 2/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0628 -
acc: 0.9819 - val_loss: 0.0729 - val_acc: 0.9761
Epoch 3/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0347 -
acc: 0.9903 - val_loss: 0.0790 - val_acc: 0.9761
Epoch 4/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0247 -
acc: 0.9929 - val_loss: 0.0863 - val_acc: 0.9739
Epoch 5/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0166 -
acc: 0.9952 - val_loss: 0.0629 - val_acc: 0.9808
Epoch 6/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0107 -
acc: 0.9975 - val_loss: 0.0700 - val_acc: 0.9800
Epoch 7/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0106 -
acc: 0.9969 - val_loss: 0.0792 - val_acc: 0.9795
Epoch 8/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0119 -
acc: 0.9964 - val_loss: 0.0881 - val_acc: 0.9735
Epoch 9/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0126 -
acc: 0.9961 - val_loss: 0.0952 - val_acc: 0.9742
Epoch 10/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0114 -
acc: 0.9967 - val_loss: 0.0691 - val_acc: 0.9795
Epoch 11/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0071 -
acc: 0.9981 - val_loss: 0.0791 - val_acc: 0.9789
```

```
Epoch 12/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0046 -
acc: 0.9988 - val_loss: 0.0689 - val_acc: 0.9815
Epoch 13/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0058 -
acc: 0.9983 - val_loss: 0.0880 - val_acc: 0.9788
Epoch 14/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0092 -
acc: 0.9971 - val_loss: 0.0812 - val_acc: 0.9789
Epoch 15/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0103 -
acc: 0.9968 - val_loss: 0.0835 - val_acc: 0.9789
Epoch 16/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0070 -
acc: 0.9978 - val_loss: 0.0753 - val_acc: 0.9795
Epoch 17/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0057 -
acc: 0.9984 - val_loss: 0.0794 - val_acc: 0.9803
Epoch 18/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0055 -
acc: 0.9984 - val_loss: 0.0972 - val_acc: 0.9781
Epoch 19/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0074 -
acc: 0.9976 - val_loss: 0.0844 - val_acc: 0.9795
Epoch 20/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0057 -
acc: 0.9983 - val_loss: 0.0721 - val_acc: 0.9826
Epoch 21/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0029 -
acc: 0.9991 - val_loss: 0.0840 - val_acc: 0.9807
Epoch 22/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0020 -
acc: 0.9996 - val_loss: 0.0831 - val_acc: 0.9801
Epoch 23/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0041 -
acc: 0.9987 - val_loss: 0.0778 - val_acc: 0.9815
Epoch 24/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0077 -
acc: 0.9975 - val_loss: 0.0890 - val_acc: 0.9810
Epoch 25/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0087 -
acc: 0.9972 - val_loss: 0.0837 - val_acc: 0.9807
Epoch 26/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0052 -
acc: 0.9983 - val_loss: 0.0874 - val_acc: 0.9811
Epoch 27/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0044 -
acc: 0.9984 - val_loss: 0.0847 - val_acc: 0.9808
```

```
Epoch 28/50
60000/60000 [==============================] - 2s 34us/step - loss: 0.0033 -
acc: 0.9989 - val_loss: 0.0776 - val_acc: 0.9832
Epoch 29/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0024 -
acc: 0.9993 - val_loss: 0.0844 - val_acc: 0.9818
Epoch 30/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0031 -
acc: 0.9992 - val_loss: 0.0749 - val_acc: 0.9837
Epoch 31/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0015 -
acc: 0.9996 - val_loss: 0.0798 - val_acc: 0.9840
Epoch 32/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0021 -
acc: 0.9994 - val_loss: 0.0849 - val_acc: 0.9833
Epoch 33/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0029 -
acc: 0.9992 - val_loss: 0.0932 - val_acc: 0.9801
Epoch 34/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0075 -
acc: 0.9974 - val_loss: 0.1185 - val_acc: 0.9765
Epoch 35/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0076 -
acc: 0.9976 - val_loss: 0.0923 - val_acc: 0.9815
Epoch 36/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0044 -
acc: 0.9987 - val_loss: 0.0736 - val_acc: 0.9846
Epoch 37/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0026 -
acc: 0.9993 - val_loss: 0.0739 - val_acc: 0.9849
Epoch 38/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0016 -
acc: 0.9996 - val_loss: 0.0749 - val_acc: 0.9855
Epoch 39/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0010 -
acc: 0.9997 - val_loss: 0.0832 - val_acc: 0.9828
Epoch 40/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0019 -
acc: 0.9996 - val_loss: 0.0719 - val_acc: 0.9849
Epoch 41/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0018 -
acc: 0.9996 - val_loss: 0.0827 - val_acc: 0.9828
Epoch 42/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0027 -
acc: 0.9991 - val_loss: 0.0890 - val_acc: 0.9814
Epoch 43/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0041 -
acc: 0.9987 - val_loss: 0.0884 - val_acc: 0.9821
```

```
Epoch 44/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0050 -
acc: 0.9983 - val_loss: 0.0821 - val_acc: 0.9814
Epoch 45/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0031 -
acc: 0.9992 - val_loss: 0.0759 - val_acc: 0.9842
Epoch 46/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0024 -
acc: 0.9992 - val_loss: 0.0750 - val_acc: 0.9846
Epoch 47/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0018 -
acc: 0.9994 - val_loss: 0.0711 - val_acc: 0.9849
Epoch 48/50
60000/60000 [==============================] - 2s 34us/step - loss: 7.9007e-04 -
acc: 0.9998 - val_loss: 0.0778 - val_acc: 0.9842
Epoch 49/50
60000/60000 [==============================] - 2s 34us/step - loss: 6.2259e-04 -
acc: 0.9999 - val_loss: 0.0703 - val_acc: 0.9855
Epoch 50/50
60000/60000 [==============================] - 2s 35us/step - loss: 0.0012 -
acc: 0.9996 - val_loss: 0.0865 - val_acc: 0.9841
```

[0]:
```
cnn.save_model_weights("p4_model_weights.h5")
cnn.save_model("p4_model.json")
cnn.save_model_history(cnn_hist, "p4_model_history.pkl")
cnn.model.save("p4_full_model.h5")
```

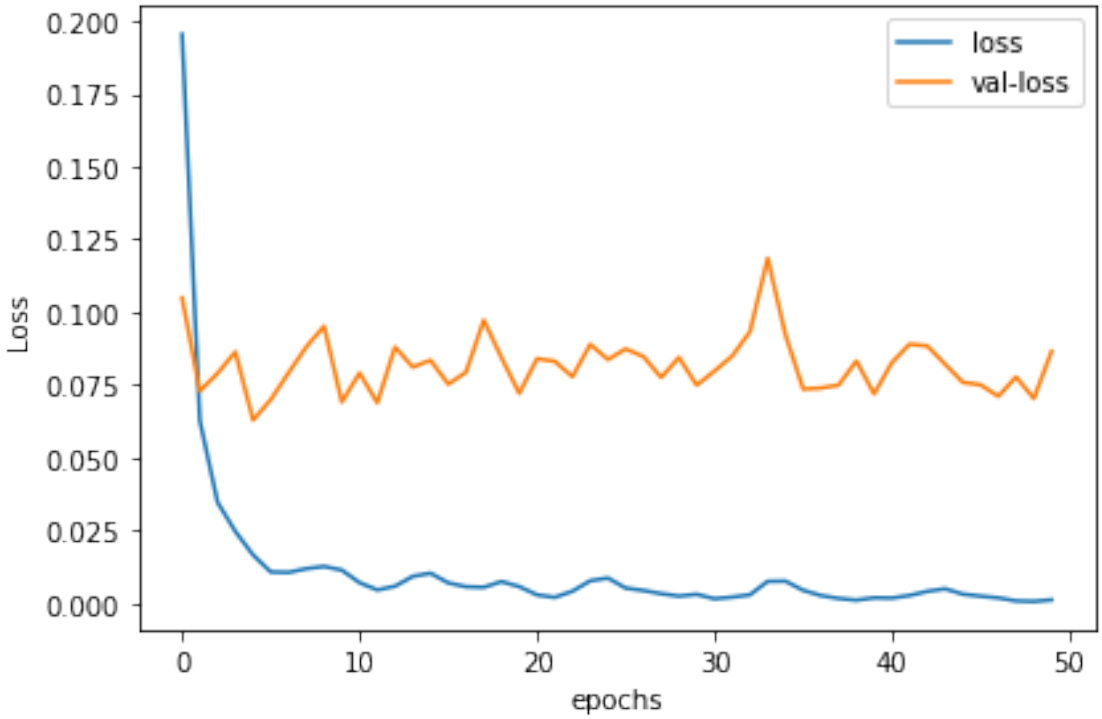
[0]:
```
! cp -r p4_full_model.h5 p4_model.json p4_model_weights.h5 p4_model_history.pkl .
 ↪/gdrive/My\ Drive/ece595_ml2/models/
```

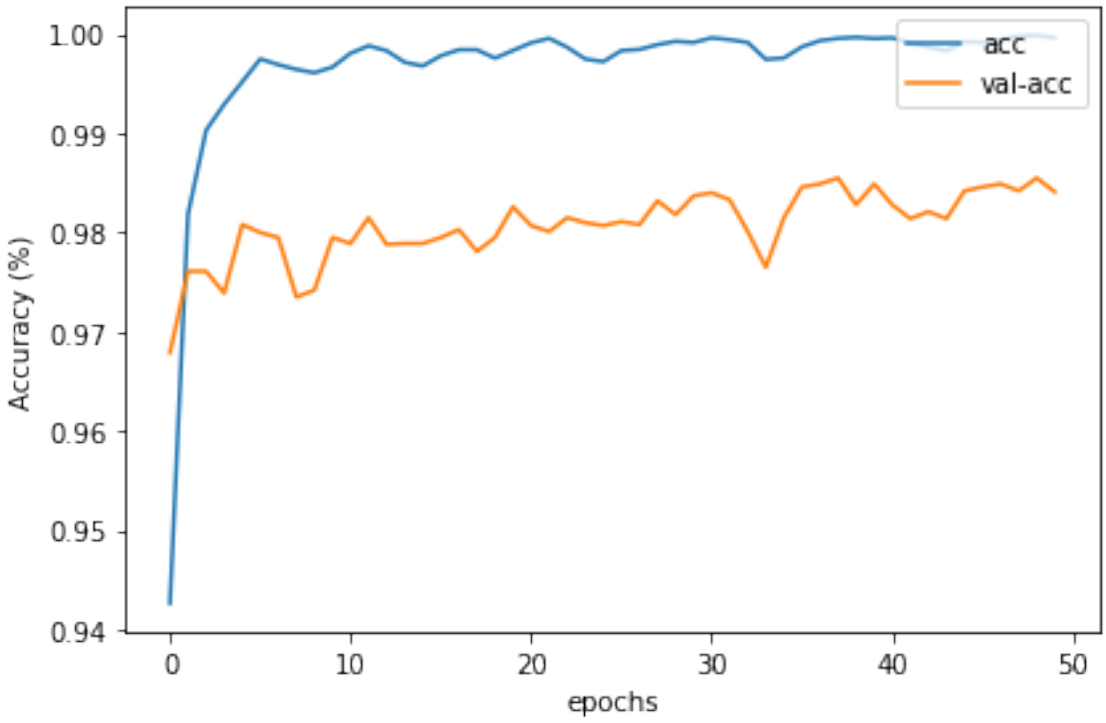
[0]:
```
cnn_model = cnn.load_model("p4_model.json", "p4_model_weights.h5")
```

[0]:
```
cnn_reconstructions = cnn_model.predict(data_test)
```

[0]:
```
def plot(hist, param):
  plt.plot(hist.history[param])
  plt.plot(hist.history['val_' + param])
  plt.legend([param, 'val-' + param], loc='upper right')
  plt.xlabel("epochs")
  if param == 'acc':
    plt.ylabel(r"Accuracy (%)")
  elif param == 'loss':
    plt.ylabel(r"Loss")
  plt.tight_layout()
  plt.show()
```

[0]:
```
plot(cnn_hist, 'loss')
plot(cnn_hist, 'acc')
```





[0]:
```
scores = cnn.model.evaluate(data_test, labels_test)
print("Accuracy: %.2f%%"%(scores[1]*100))
```

```
10000/10000 [==============================] - 1s 63us/step
Accuracy: 98.41%
```

[8]:
```
# Import pre-trained classifier
#mnist_classifier = cnn.load_model("p4_model.json", "p4_model_weights.h5")
mnist_classifier = load_model(MODEL_LOCATION + "p4_full_model.h5")
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4409: The name tf.random_normal is
deprecated. Please use tf.random.normal instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:148: The name
tf.placeholder_with_default is deprecated. Please use
tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:190: The name
tf.get_default_session is deprecated. Please use
tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is
deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:203: The name tf.Session is
deprecated. Please use tf.compat.v1.Session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
```

```
packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:216: The name
tf.is_variable_initialized is deprecated. Please use
tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:223: The name
tf.variables_initializer is deprecated. Please use
tf.compat.v1.variables_initializer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated.
Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:3576: The name tf.log is
deprecated. Please use tf.math.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/ops/math_grad.py:1424: where (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.
```

[0]:
```
#Get TensorFlow Session to pass into Cleverhans modules
sess = backend.get_session()
```

[0]:
```
#Create wrapper for classifier model so that it can be passed into Cleverhans␣
 ↪modules
wrap = KerasModelWrapper(mnist_classifier)
```

Part 2: The Fast Gradient Method (FGM)

[0]:
```
#Implementing the FGSM attack

#FGM Instance on trained classifier from Part 1
fgm = FastGradientMethod(wrap, sess=sess)
```

[0]:
```
#Attack parameters
fgm_params = {'eps': 0.25,
              'clip_min': 0.,
              'clip_max': 1.}
```

[36]:
```
#Generate adversarial data
fgm_attack_data = fgm.generate_np(data_test, **fgm_params)
```

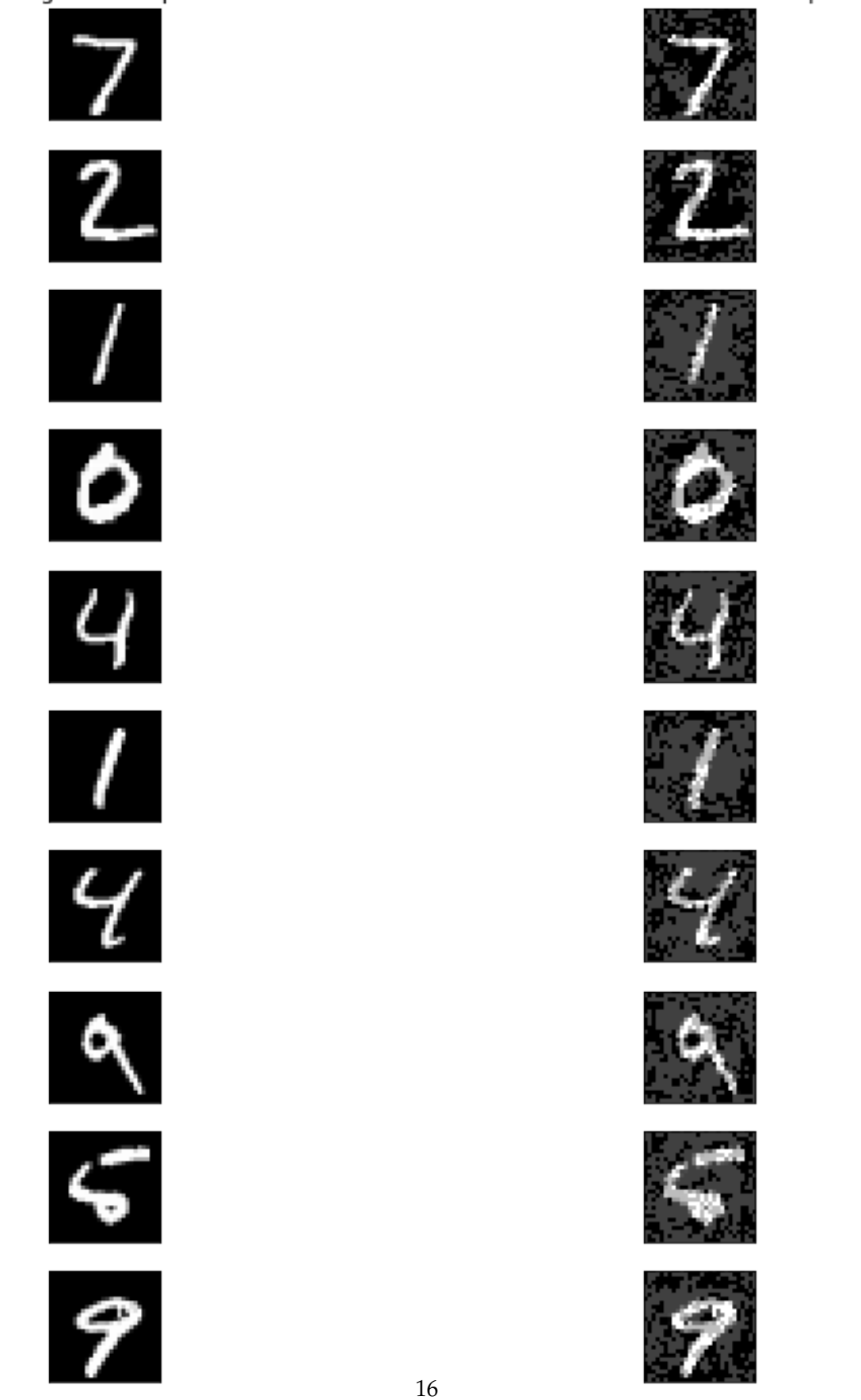
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/cleverhans/attacks_tf.py:62: calling reduce_sum_v1 (from
tensorflow.python.ops.math_ops) with keep_dims is deprecated and will be removed
in a future version.
Instructions for updating:
keep_dims is deprecated, use keepdims instead
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/cleverhans/utils_tf.py:37: softmax_cross_entropy_with_logits (from
tensorflow.python.ops.nn_ops) is deprecated and will be removed in a future
version.
Instructions for updating:

Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.

See `tf.nn.softmax_cross_entropy_with_logits_v2`.
```

[38]:
```
#Evaluate accuracy on target classifier
fgm_adv_scores = mnist_classifier.evaluate(fgm_attack_data, labels_test)
print("Accuracy: %.2f%%"%(fgm_adv_scores[1]*100))
```

```
10000/10000 [==============================] - 1s 82us/step
Accuracy: 11.81%
```

[39]:
```
#Show ten original samples and their corresponding adversarial samples
data_test_show = data_test.reshape(-1, 28, 28)
fgm_attack_data_show = fgm_attack_data.reshape(-1, 28, 28)
cols = [' Original Samples',
        ' FGM Adversarial Samples']
fig, axes = plt.subplots(nrows=10, ncols=2, figsize=(10, 10))
for i in range(10):
```

```
  axes[i, 0].imshow(data_test_show[i,:,:],
                    cmap=plt.cm.gray)
  axes[i, 1].imshow(fgm_attack_data_show[i,:,:],
                     cmap=plt.cm.gray)
  for ax, col in zip(axes[0], cols):
    ax.set_title(col)
  for j in range(2):
    axes[i, j].get_xaxis().set_visible(False)
    axes[i, j].get_yaxis().set_visible(False)
fig.tight_layout()
plt.show()
```


Original Samples
FGM Adversarial Samples

[0]:
```
# Load data MNIST data and normalize to [0, 1]
(data_train, labels_train), (data_test, labels_test) = mnist.load_data()
data_train = data_train / 255.
data_test = data_test / 255.
#Reshape training and testing data into 784-dimensional vectors
data_train = data_train.reshape(-1, 784)
data_test = data_test.reshape(-1, 784)
#Convert integer labels for training and testing data into one-hot vectors
labels_train = keras.utils.np_utils.to_categorical(labels_train, num_classes=10)
labels_test = keras.utils.np_utils.to_categorical(labels_test, num_classes=10)
```

[0]:
```
def autoencoder():
    ae = Sequential()
    ae.add(Dense(400, activation='relu', kernel_initializer="normal",␣
 ↪input_dim=784))
    ae.add(Dense(200, activation='relu', kernel_initializer='normal'))
    ae.add(Dense(100, activation='relu', kernel_initializer='normal'))
    ae.add(Dense(200, activation='relu', kernel_initializer='normal'))
    ae.add(Dense(400, activation='relu', kernel_initializer='normal'))
    ae.add(Dense(784, activation='sigmoid', kernel_initializer='normal'))
    return ae
```

[0]:
```
#Using the autoencoder for detection and to determine a threshold
ae_model = autoencoder()
ae_model.compile(loss='mse',
                         optimizer='adam')
```

[43]:
```
#Create and train the autoencoder using the mean squared error loss and adam␣
 ↪optimizer
ae_model_history = ae_model.fit(data_train, data_train,
                                validation_data=(data_test, data_test),
                                epochs=50,
                                batch_size=256,
                                shuffle=True)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/50
60000/60000 [==============================] - 2s 41us/step - loss: 0.0542 -
val_loss: 0.0260
Epoch 2/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0213 -
val_loss: 0.0173
Epoch 3/50
60000/60000 [==============================] - 1s 25us/step - loss: 0.0158 -
val_loss: 0.0135
```

```
Epoch 4/50
60000/60000 [==============================] - 2s 25us/step - loss: 0.0129 -
val_loss: 0.0117
Epoch 5/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0112 -
val_loss: 0.0107
Epoch 6/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0100 -
val_loss: 0.0095
Epoch 7/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0092 -
val_loss: 0.0088
Epoch 8/50
60000/60000 [==============================] - 2s 25us/step - loss: 0.0086 -
val_loss: 0.0083
Epoch 9/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0081 -
val_loss: 0.0078
Epoch 10/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0076 -
val_loss: 0.0074
Epoch 11/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0072 -
val_loss: 0.0068
Epoch 12/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0069 -
val_loss: 0.0066
Epoch 13/50
60000/60000 [==============================] - 2s 25us/step - loss: 0.0066 -
val_loss: 0.0064
Epoch 14/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0063 -
val_loss: 0.0061
Epoch 15/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0061 -
val_loss: 0.0060
Epoch 16/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0058 -
val_loss: 0.0056
Epoch 17/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0056 -
val_loss: 0.0055
Epoch 18/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0054 -
val_loss: 0.0053
Epoch 19/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0053 -
val_loss: 0.0053
```

```
Epoch 20/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0052 -
val_loss: 0.0052
Epoch 21/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0050 -
val_loss: 0.0049
Epoch 22/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0049 -
val_loss: 0.0048
Epoch 23/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0048 -
val_loss: 0.0047
Epoch 24/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0047 -
val_loss: 0.0047
Epoch 25/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0046 -
val_loss: 0.0048
Epoch 26/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0045 -
val_loss: 0.0045
Epoch 27/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0044 -
val_loss: 0.0044
Epoch 28/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0044 -
val_loss: 0.0044
Epoch 29/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0043 -
val_loss: 0.0045
Epoch 30/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0042 -
val_loss: 0.0042
Epoch 31/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0042 -
val_loss: 0.0042
Epoch 32/50
60000/60000 [==============================] - 1s 25us/step - loss: 0.0041 -
val_loss: 0.0041
Epoch 33/50
60000/60000 [==============================] - 1s 25us/step - loss: 0.0040 -
val_loss: 0.0040
Epoch 34/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0040 -
val_loss: 0.0041
Epoch 35/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0039 -
val_loss: 0.0040
```

```
Epoch 36/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0039 -
val_loss: 0.0041
Epoch 37/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0038 -
val_loss: 0.0040
Epoch 38/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0038 -
val_loss: 0.0039
Epoch 39/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0038 -
val_loss: 0.0038
Epoch 40/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0037 -
val_loss: 0.0038
Epoch 41/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0037 -
val_loss: 0.0037
Epoch 42/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0036 -
val_loss: 0.0038
Epoch 43/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0036 -
val_loss: 0.0037
Epoch 44/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0035 -
val_loss: 0.0036
Epoch 45/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0035 -
val_loss: 0.0036
Epoch 46/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0034 -
val_loss: 0.0034
Epoch 47/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0034 -
val_loss: 0.0036
Epoch 48/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0034 -
val_loss: 0.0036
Epoch 49/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0033 -
val_loss: 0.0035
Epoch 50/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0033 -
val_loss: 0.0035
```

[0]:
```
# Create adversarial examples using FGSM on training data
fgm_attack_data = fgm.generate_np(data_train, **fgm_params)
```

[0]:
```
# Obtain reconstruction errors on training set and determine a threshold
reconstructions = ae_model.predict(fgm_attack_data)
```

[0]:
```
# Convert error tensor into NumPy array
error = keras.losses.mean_squared_error(fgm_attack_data, reconstructions)
error = error.eval(session=sess)
```

[47]:
```
# Determine threshold (based on min in this case) and print it
threshold = min(error)
print("The threshold error is: {}".format(threshold))
```

```
The threshold error is: 0.019591733813285828
```

[0]:
```
# Calculate error of adversarial testing set
fgm_test_attack_data = fgm.generate_np(data_test, **fgm_params)
reconstructions_test = ae_model.predict(fgm_test_attack_data)
# Convert error tensor into NumPy array
test_error = keras.losses.mean_squared_error(fgm_test_attack_data,␣
 ↪reconstructions_test)
test_error = test_error.eval(session=sess)
```

[0]:
```
true_positives = sum(np.where(test_error > threshold, 1, 0))
```

[50]:
```
print(true_positives)
```

```
10000
```

[0]:
```
reconstructions_tests = ae_model.predict(data_test)
```

[0]:
```
test_errors = keras.losses.mean_squared_error(data_test, reconstructions_tests)
test_errors = test_errors.eval(session=sess)
```

[0]:
```
false_positives = sum(np.where(test_errors > threshold, 1, 0))
```

[56]:
```
print(false_positives)
```

```
1
```

Part 3: Projected Gradient Descent

[0]:
```
#Implementing the PGD attack

#PGD Instance on trained classifier from Part 1
```

```
pgd = MadryEtAl(wrap, sess=sess)
```

[0]:
```
print(data_test.shape)
print(labels_test.shape)
```

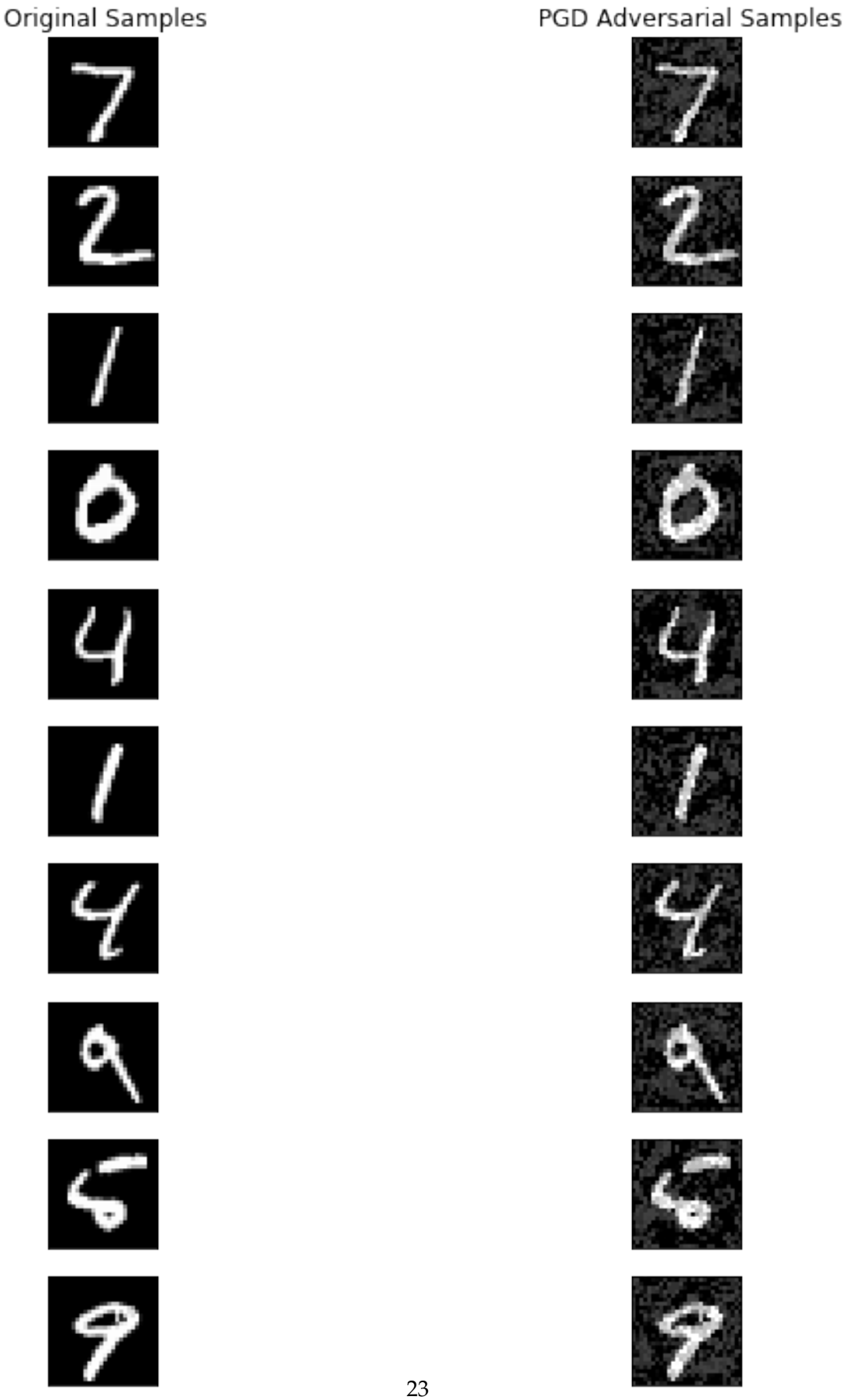
```
(10000, 784)
(10000, 10)
```

[0]:
```
#Attack parameters
pgd_params = {'eps': 0.25,
              'eps_iter': 0.01,
              'nb_iter': 20,
              'clip_min': 0.,
              'clip_max': 1.}

# Generating adversarial perturbations
pgd_perturbations = pgd.generate_np(data_test, **pgd_params)
```

[0]:
```
#Evaluate accuracy on target classifier
pgd_adv_scores = mnist_classifier.evaluate(pgd_perturbations, labels_test)
print("Accuracy: %.2f%%"%(pgd_adv_scores[1]*100))
```

```
10000/10000 [==============================] - 3s 270us/step
Accuracy: 1.00%
```

[0]:
```
#Show ten original samples and their corresponding adversarial samples
#Show ten original samples and their corresponding adversarial samples
data_test_show = data_test.reshape(-1, 28, 28)
pgd_perturbations_show = pgd_perturbations.reshape(-1, 28, 28)
cols = [' Original Samples',
        ' PGD Adversarial Samples']
fig, axes = plt.subplots(nrows=10, ncols=2, figsize=(10, 10))
for i in range(10):
  axes[i, 0].imshow(data_test_show[i,:,:],
                    cmap=plt.cm.gray)
  axes[i, 1].imshow(pgd_perturbations_show[i,:,:],
                     cmap=plt.cm.gray)
  for ax, col in zip(axes[0], cols):
    ax.set_title(col)
  for j in range(2):
    axes[i, j].get_xaxis().set_visible(False)
    axes[i, j].get_yaxis().set_visible(False)
fig.tight_layout()
plt.show()
```


Original Samples
PGD Adversarial Samples

[0]:
```
adv_train_clf = keras.models.clone_model(mnist_classifier)
#modify model here if needed
adv_train_clf.build()
adv_train_clf.set_weights(mnist_classifier.get_weights())
```

[0]:
```
pgd_data = pgd.generate_np(data_train, **pgd_params)
data_train1 = np.concatenate((data_train, pgd_data))
```

[0]:
```
labels_train1 = np.concatenate((labels_train, labels_train))
```

[0]:
```
print(data_train1.shape)
print(labels_train1.shape)
```

```
(120000, 784)
(120000, 10)
```

[0]:
```
print(data_test.shape)
```

```
(10000, 784)
```

[0]:
```
#Implementing the adversarial training defense
adv_train_clf.compile(loss="binary_crossentropy",
                      optimizer='adam',
                      metrics=['accuracy'])
adv_train_clf_hist = adv_train_clf.fit(data_train1, labels_train1,
                  validation_data=(data_test.reshape(-1, 784), labels_test),
                  epochs=50,
                  batch_size=256,
                  shuffle=True)
```

```
Train on 120000 samples, validate on 10000 samples
Epoch 1/50
120000/120000 [==============================] - 8s 69us/step - loss: 0.0206 -
acc: 0.9933 - val_loss: 0.0172 - val_acc: 0.9940
Epoch 2/50
120000/120000 [==============================] - 6s 46us/step - loss: 0.0071 -
acc: 0.9977 - val_loss: 0.0161 - val_acc: 0.9948
Epoch 3/50
120000/120000 [==============================] - 6s 47us/step - loss: 0.0048 -
acc: 0.9985 - val_loss: 0.0138 - val_acc: 0.9954
Epoch 4/50
120000/120000 [==============================] - 6s 49us/step - loss: 0.0035 -
acc: 0.9989 - val_loss: 0.0147 - val_acc: 0.9954
Epoch 5/50
120000/120000 [==============================] - 6s 51us/step - loss: 0.0026 -
```

```
acc: 0.9992 - val_loss: 0.0138 - val_acc: 0.9955
Epoch 6/50
120000/120000 [==============================] - 6s 52us/step - loss: 0.0023 -
acc: 0.9993 - val_loss: 0.0133 - val_acc: 0.9958
Epoch 7/50
120000/120000 [==============================] - 6s 52us/step - loss: 0.0020 -
acc: 0.9993 - val_loss: 0.0138 - val_acc: 0.9958
Epoch 8/50
120000/120000 [==============================] - 6s 52us/step - loss: 0.0019 -
acc: 0.9994 - val_loss: 0.0126 - val_acc: 0.9963
Epoch 9/50
120000/120000 [==============================] - 6s 51us/step - loss: 0.0019 -
acc: 0.9994 - val_loss: 0.0147 - val_acc: 0.9958
Epoch 10/50
120000/120000 [==============================] - 6s 52us/step - loss: 0.0010 -
acc: 0.9997 - val_loss: 0.0142 - val_acc: 0.9963
Epoch 11/50
120000/120000 [==============================] - 6s 52us/step - loss: 0.0010 -
acc: 0.9997 - val_loss: 0.0131 - val_acc: 0.9961
Epoch 12/50
120000/120000 [==============================] - 6s 50us/step - loss: 0.0015 -
acc: 0.9995 - val_loss: 0.0149 - val_acc: 0.9958
Epoch 13/50
120000/120000 [==============================] - 6s 50us/step - loss: 0.0013 -
acc: 0.9996 - val_loss: 0.0158 - val_acc: 0.9958
Epoch 14/50
120000/120000 [==============================] - 6s 50us/step - loss: 9.7624e-04
- acc: 0.9997 - val_loss: 0.0130 - val_acc: 0.9963
Epoch 15/50
120000/120000 [==============================] - 6s 53us/step - loss: 9.8632e-04
- acc: 0.9997 - val_loss: 0.0139 - val_acc: 0.9962
Epoch 16/50
120000/120000 [==============================] - 6s 52us/step - loss: 9.4157e-04
- acc: 0.9997 - val_loss: 0.0148 - val_acc: 0.9961
Epoch 17/50
120000/120000 [==============================] - 6s 50us/step - loss: 9.4647e-04
- acc: 0.9997 - val_loss: 0.0142 - val_acc: 0.9962
Epoch 18/50
120000/120000 [==============================] - 6s 49us/step - loss: 8.7492e-04
- acc: 0.9997 - val_loss: 0.0135 - val_acc: 0.9962
Epoch 19/50
120000/120000 [==============================] - 6s 49us/step - loss: 6.6374e-04
- acc: 0.9998 - val_loss: 0.0144 - val_acc: 0.9959
Epoch 20/50
120000/120000 [==============================] - 6s 49us/step - loss: 8.0419e-04
- acc: 0.9997 - val_loss: 0.0161 - val_acc: 0.9959
Epoch 21/50
120000/120000 [==============================] - 6s 51us/step - loss: 6.7951e-04
```

```
- acc: 0.9998 - val_loss: 0.0123 - val_acc: 0.9970
Epoch 22/50
120000/120000 [==============================] - 6s 49us/step - loss: 7.3342e-04
- acc: 0.9998 - val_loss: 0.0136 - val_acc: 0.9966
Epoch 23/50
120000/120000 [==============================] - 6s 51us/step - loss: 9.0528e-04
- acc: 0.9997 - val_loss: 0.0131 - val_acc: 0.9968
Epoch 24/50
120000/120000 [==============================] - 6s 50us/step - loss: 3.7030e-04
- acc: 0.9999 - val_loss: 0.0139 - val_acc: 0.9967
Epoch 25/50
120000/120000 [==============================] - 6s 50us/step - loss: 4.6547e-04
- acc: 0.9998 - val_loss: 0.0150 - val_acc: 0.9963
Epoch 26/50
120000/120000 [==============================] - 6s 50us/step - loss: 3.6252e-04
- acc: 0.9999 - val_loss: 0.0143 - val_acc: 0.9967
Epoch 27/50
120000/120000 [==============================] - 6s 51us/step - loss: 5.9939e-04
- acc: 0.9998 - val_loss: 0.0166 - val_acc: 0.9961
Epoch 28/50
120000/120000 [==============================] - 6s 50us/step - loss: 7.6251e-04
- acc: 0.9997 - val_loss: 0.0167 - val_acc: 0.9962
Epoch 29/50
120000/120000 [==============================] - 6s 49us/step - loss: 5.6537e-04
- acc: 0.9998 - val_loss: 0.0127 - val_acc: 0.9968
Epoch 30/50
120000/120000 [==============================] - 6s 50us/step - loss: 5.2206e-04
- acc: 0.9998 - val_loss: 0.0146 - val_acc: 0.9966
Epoch 31/50
120000/120000 [==============================] - 6s 50us/step - loss: 4.1741e-04
- acc: 0.9999 - val_loss: 0.0154 - val_acc: 0.9964
Epoch 32/50
120000/120000 [==============================] - 6s 50us/step - loss: 4.3015e-04
- acc: 0.9999 - val_loss: 0.0154 - val_acc: 0.9965
Epoch 33/50
120000/120000 [==============================] - 6s 49us/step - loss: 3.1711e-04
- acc: 0.9999 - val_loss: 0.0148 - val_acc: 0.9966
Epoch 34/50
120000/120000 [==============================] - 6s 50us/step - loss: 6.0660e-04
- acc: 0.9998 - val_loss: 0.0147 - val_acc: 0.9965
Epoch 35/50
120000/120000 [==============================] - 6s 50us/step - loss: 5.3772e-04
- acc: 0.9998 - val_loss: 0.0139 - val_acc: 0.9968
Epoch 36/50
120000/120000 [==============================] - 6s 49us/step - loss: 4.3571e-04
- acc: 0.9998 - val_loss: 0.0154 - val_acc: 0.9963
Epoch 37/50
120000/120000 [==============================] - 6s 50us/step - loss: 3.5285e-04
```

```
- acc: 0.9999 - val_loss: 0.0153 - val_acc: 0.9966
Epoch 38/50
120000/120000 [==============================] - 6s 51us/step - loss: 3.2939e-04
- acc: 0.9999 - val_loss: 0.0159 - val_acc: 0.9964
Epoch 39/50
120000/120000 [==============================] - 6s 51us/step - loss: 3.0955e-04
- acc: 0.9999 - val_loss: 0.0145 - val_acc: 0.9969
Epoch 40/50
120000/120000 [==============================] - 6s 50us/step - loss: 2.6533e-04
- acc: 0.9999 - val_loss: 0.0156 - val_acc: 0.9968
Epoch 41/50
120000/120000 [==============================] - 6s 51us/step - loss: 5.5736e-04
- acc: 0.9998 - val_loss: 0.0153 - val_acc: 0.9964
Epoch 42/50
120000/120000 [==============================] - 6s 50us/step - loss: 4.9232e-04
- acc: 0.9998 - val_loss: 0.0149 - val_acc: 0.9967
Epoch 43/50
120000/120000 [==============================] - 6s 51us/step - loss: 2.7092e-04
- acc: 0.9999 - val_loss: 0.0145 - val_acc: 0.9968
Epoch 44/50
120000/120000 [==============================] - 6s 51us/step - loss: 1.3696e-04
- acc: 1.0000 - val_loss: 0.0137 - val_acc: 0.9971
Epoch 45/50
120000/120000 [==============================] - 6s 50us/step - loss: 1.2502e-04
- acc: 1.0000 - val_loss: 0.0137 - val_acc: 0.9971
Epoch 46/50
120000/120000 [==============================] - 6s 51us/step - loss: 5.5670e-05
- acc: 1.0000 - val_loss: 0.0146 - val_acc: 0.9970
Epoch 47/50
120000/120000 [==============================] - 6s 51us/step - loss: 5.3551e-04
- acc: 0.9998 - val_loss: 0.0171 - val_acc: 0.9963
Epoch 48/50
120000/120000 [==============================] - 6s 51us/step - loss: 6.9112e-04
- acc: 0.9998 - val_loss: 0.0146 - val_acc: 0.9969
Epoch 49/50
120000/120000 [==============================] - 6s 51us/step - loss: 3.8870e-04
- acc: 0.9999 - val_loss: 0.0147 - val_acc: 0.9968
Epoch 50/50
120000/120000 [==============================] - 6s 50us/step - loss: 1.9802e-04
- acc: 0.9999 - val_loss: 0.0144 - val_acc: 0.9970
```

[0]:
```
adv_train_clf.save('adversarial_trained_classifier.h5')
```

[0]:
```
!cp adversarial_trained_classifier.h5 ./gdrive/My\ Drive/ece595_ml2/models/
```

[0]:
```
 adv_mnist_classifier = load_model("adversarial_trained_classifier.h5")
```

[0]:
```
print(pgd_perturbations.shape)
print(labels_test.shape)
```

```
(10000, 784)
(10000, 10)
```

[0]:
```
#Evaluate accuracy on target classifier
pgd_adv_scores = adv_mnist_classifier.evaluate(pgd_perturbations, labels_test)
print("Accuracy: %.2f%%"%(pgd_adv_scores[1]*100))
```

```
10000/10000 [==============================] - 1s 76us/step
Accuracy: 99.67%
```

[0]:
```
#Using the defense to evaluate the accuracy of the perturbed data
#FILL THIS IN
```

Part 4: Carlini and Wagner Attack

[15]:
```
#Implementing the CW attack

#CW Instance on trained classifier from Part 1
cwa = CarliniWagnerL2(wrap, sess=sess)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/cleverhans/utils_tf.py:368: The name tf.GraphKeys is deprecated. Please
use tf.compat.v1.GraphKeys instead.
```

[0]:
```
#Attack parameters
cwa_params = {'binary_search_steps': 1,
              'y': None,
              'learning_rate': 1.25,
              'batch_size': 16,
              'initial_const': 10,
              'clip_min': 0.0,
              'clip_max': 1.0}
```

[17]:
```
# Generating adversarial perturbations
cwa_perturbations = cwa.generate_np(data_test, **cwa_params)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/cleverhans/attacks.py:216: calling reduce_max_v1 (from
tensorflow.python.ops.math_ops) with keep_dims is deprecated and will be removed
in a future version.
Instructions for updating:
keep_dims is deprecated, use keepdims instead
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
```

```
packages/cleverhans/attacks.py:218: to_float (from
tensorflow.python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use `tf.cast` instead.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/cleverhans/attacks_tf.py:725: The name tf.train.AdamOptimizer is
deprecated. Please use tf.compat.v1.train.AdamOptimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/cleverhans/attacks.py:909: py_func (from
tensorflow.python.ops.script_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
tf.py_func is deprecated in TF V2. Instead, there are two
    options available in V2.
    - tf.py_function takes a python function which manipulates tf eager
    tensors instead of numpy arrays. It's easy to convert a tf eager tensor to
    an ndarray (just call tensor.numpy()) but having access to eager tensors
    means `tf.py_function`s can use accelerators such as GPUs as well as
    being differentiable using a gradient tape.
    - tf.numpy_function maintains the semantics of the deprecated tf.py_func
    (it is not differentiable, and manipulates numpy arrays). It drops the
    stateful argument making all functions stateful.
```

[0]:
```
with open('cwa_perturbations.pkl', 'wb') as f:
      pickle.dump(cwa_perturbations, f)
```

[0]:
```
!cp cwa_perturbations.pkl ./gdrive/My\ Drive/ece595_ml2/models/
```

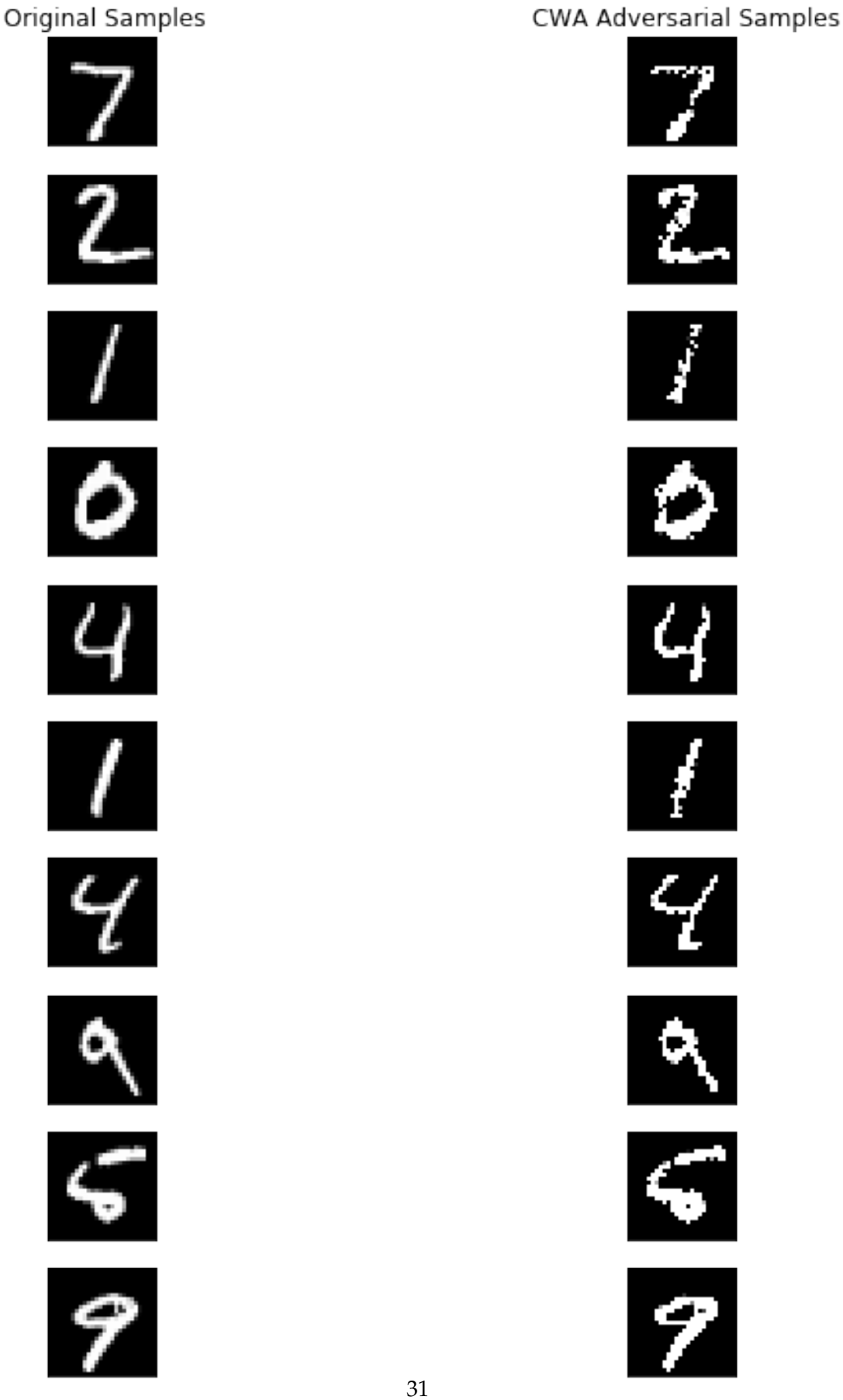
[0]:
```
with open(MODEL_LOCATION + 'cwa_perturbations.pkl', 'rb') as f:
      cwa_perturbations = pickle.load(f)
```

[22]:
```
#Evaluate accuracy on target classifier
cwa_adv_scores = mnist_classifier.evaluate(cwa_perturbations, labels_test)
print("Accuracy: %.2f%%"%(cwa_adv_scores[1]*100))
```

```
10000/10000 [==============================] - 0s 49us/step
Accuracy: 1.11%
```

[23]:
```
#Show ten original samples and their corresponding adversarial samples
data_test_show = data_test.reshape(-1, 28, 28)
cwa_perturbations_show = cwa_perturbations.reshape(-1, 28, 28)
cols = [' Original Samples',
        ' CWA Adversarial Samples']
fig, axes = plt.subplots(nrows=10, ncols=2, figsize=(10, 10))
```

```
for i in range(10):
  axes[i, 0].imshow(data_test_show[i,:,:],
                    cmap=plt.cm.gray)
  axes[i, 1].imshow(cwa_perturbations_show[i,:,:],
                     cmap=plt.cm.gray)
  for ax, col in zip(axes[0], cols):
    ax.set_title(col)
  for j in range(2):
    axes[i, j].get_xaxis().set_visible(False)
    axes[i, j].get_yaxis().set_visible(False)
fig.tight_layout()
plt.show()
```


Original Samples
CWA Adversarial Samples

[0]:
```
#Implementing the dimensionality reduction (PCA) defense

#Calculate PCA projection
pca = PCA(100)
pca.fit(data_train)
pca_train = pca.transform(data_train)
pca_test = pca.transform(data_test)
```

[25]:
```
print(pca_train.shape, pca_test.shape)
```

```
(60000, 100) (10000, 100)
```

[0]:
```
#Transform perturbed CW data using the subspace from the original training data
cwa_pert_pca = pca.transform(cwa_perturbations)
```

[0]:
```
# create model for PCA
def pca_model():
    model = Sequential()
    model.add(Dense(100,
                    activation='relu'))
    model.add(BatchNormalization())
    model.add(Dense(100,
                    activation='relu'))
    model.add(BatchNormalization())
    model.add(Dense(10,
                    activation='softmax'))

    return model
```

[0]:
```
#Create model graph, compile it, and train it using pca_train lables_train
model = pca_model()
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

[29]:
```
model_hist = model.fit(pca_train, labels_train,
                       validation_data=(pca_test, labels_test),
                       shuffle=True,
                       batch_size=256,
                       epochs=50)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/50
60000/60000 [==============================] - 2s 38us/step - loss: 0.4592 -
acc: 0.8616 - val_loss: 0.1805 - val_acc: 0.9479
```

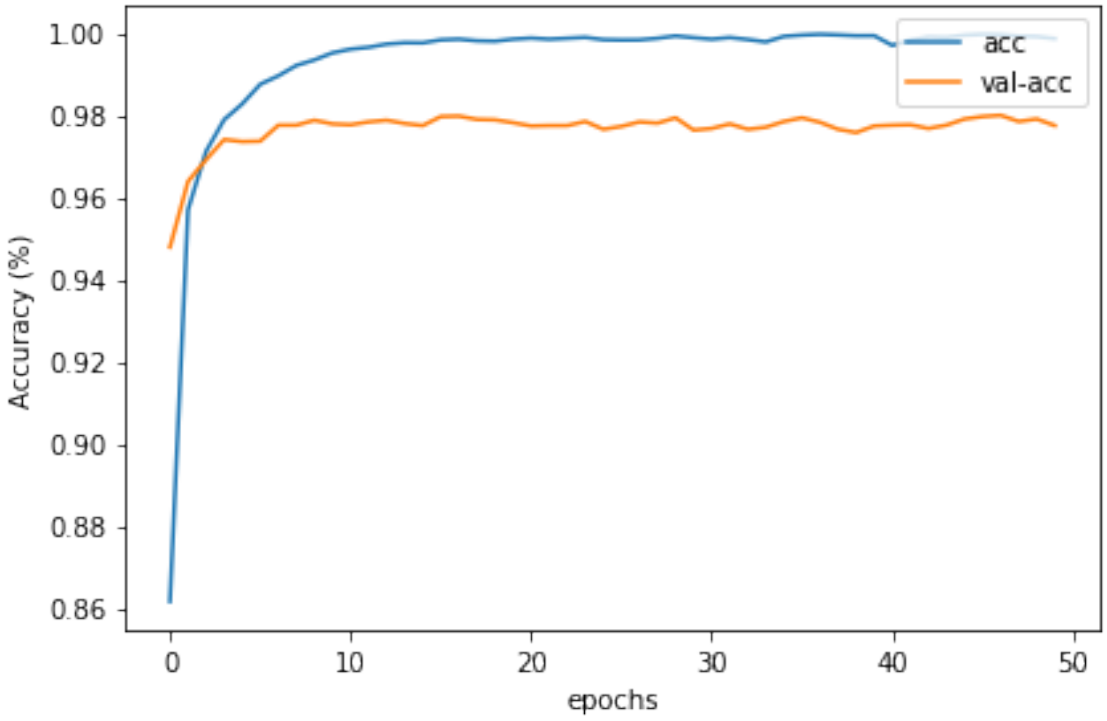
```
Epoch 2/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.1505 -
acc: 0.9570 - val_loss: 0.1226 - val_acc: 0.9639
Epoch 3/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.1002 -
acc: 0.9712 - val_loss: 0.1012 - val_acc: 0.9692
Epoch 4/50
60000/60000 [==============================] - 1s 25us/step - loss: 0.0746 -
acc: 0.9789 - val_loss: 0.0889 - val_acc: 0.9741
Epoch 5/50
60000/60000 [==============================] - 1s 25us/step - loss: 0.0591 -
acc: 0.9828 - val_loss: 0.0846 - val_acc: 0.9736
Epoch 6/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0457 -
acc: 0.9876 - val_loss: 0.0814 - val_acc: 0.9737
Epoch 7/50
60000/60000 [==============================] - 1s 25us/step - loss: 0.0365 -
acc: 0.9896 - val_loss: 0.0745 - val_acc: 0.9776
Epoch 8/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0297 -
acc: 0.9921 - val_loss: 0.0778 - val_acc: 0.9776
Epoch 9/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0244 -
acc: 0.9935 - val_loss: 0.0749 - val_acc: 0.9788
Epoch 10/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0195 -
acc: 0.9952 - val_loss: 0.0798 - val_acc: 0.9779
Epoch 11/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0166 -
acc: 0.9961 - val_loss: 0.0768 - val_acc: 0.9777
Epoch 12/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0138 -
acc: 0.9966 - val_loss: 0.0789 - val_acc: 0.9784
Epoch 13/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0120 -
acc: 0.9973 - val_loss: 0.0812 - val_acc: 0.9788
Epoch 14/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0103 -
acc: 0.9977 - val_loss: 0.0828 - val_acc: 0.9780
Epoch 15/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0093 -
acc: 0.9977 - val_loss: 0.0851 - val_acc: 0.9775
Epoch 16/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0076 -
acc: 0.9984 - val_loss: 0.0822 - val_acc: 0.9797
Epoch 17/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0069 -
acc: 0.9985 - val_loss: 0.0814 - val_acc: 0.9798
```

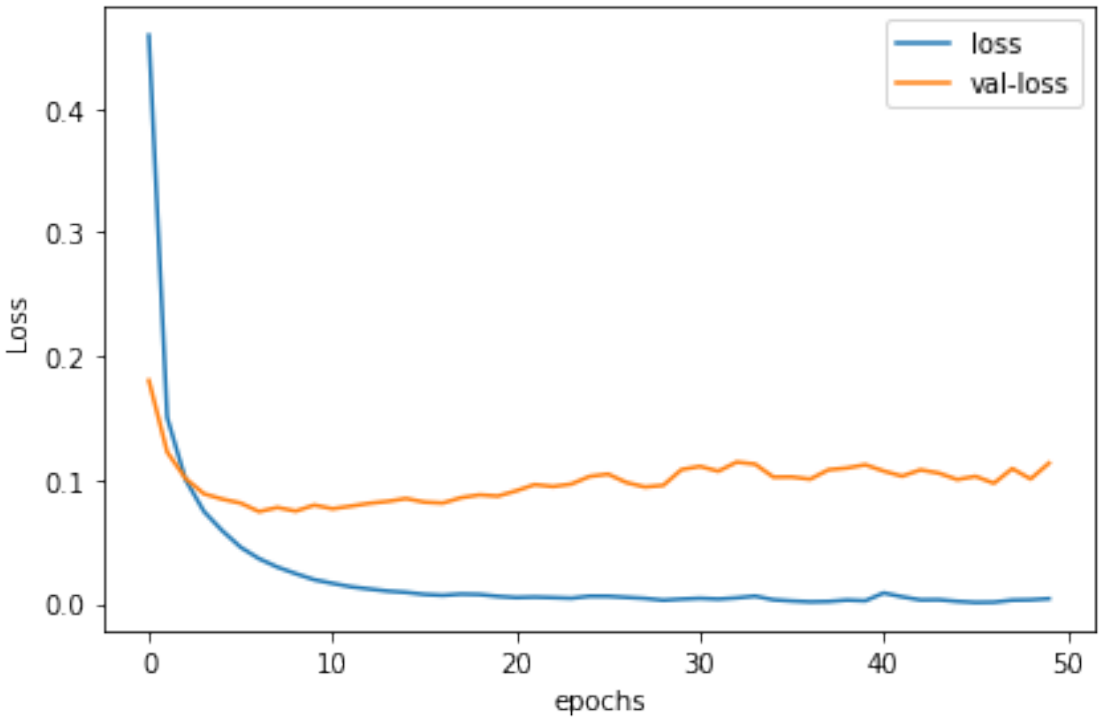
```
Epoch 18/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0079 -
acc: 0.9981 - val_loss: 0.0858 - val_acc: 0.9790
Epoch 19/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0076 -
acc: 0.9980 - val_loss: 0.0880 - val_acc: 0.9789
Epoch 20/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0061 -
acc: 0.9986 - val_loss: 0.0871 - val_acc: 0.9782
Epoch 21/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0051 -
acc: 0.9988 - val_loss: 0.0913 - val_acc: 0.9774
Epoch 22/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0055 -
acc: 0.9986 - val_loss: 0.0963 - val_acc: 0.9775
Epoch 23/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0051 -
acc: 0.9988 - val_loss: 0.0948 - val_acc: 0.9775
Epoch 24/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0046 -
acc: 0.9990 - val_loss: 0.0967 - val_acc: 0.9785
Epoch 25/50
60000/60000 [==============================] - 1s 22us/step - loss: 0.0062 -
acc: 0.9984 - val_loss: 0.1030 - val_acc: 0.9766
Epoch 26/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0061 -
acc: 0.9983 - val_loss: 0.1048 - val_acc: 0.9773
Epoch 27/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0052 -
acc: 0.9984 - val_loss: 0.0980 - val_acc: 0.9784
Epoch 28/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0044 -
acc: 0.9988 - val_loss: 0.0944 - val_acc: 0.9781
Epoch 29/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0030 -
acc: 0.9993 - val_loss: 0.0957 - val_acc: 0.9794
Epoch 30/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0039 -
acc: 0.9989 - val_loss: 0.1085 - val_acc: 0.9764
Epoch 31/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0045 -
acc: 0.9986 - val_loss: 0.1110 - val_acc: 0.9768
Epoch 32/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0039 -
acc: 0.9990 - val_loss: 0.1071 - val_acc: 0.9779
Epoch 33/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0048 -
acc: 0.9985 - val_loss: 0.1145 - val_acc: 0.9766
```

```
Epoch 34/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0063 -
acc: 0.9979 - val_loss: 0.1129 - val_acc: 0.9771
Epoch 35/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0033 -
acc: 0.9992 - val_loss: 0.1024 - val_acc: 0.9785
Epoch 36/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0023 -
acc: 0.9996 - val_loss: 0.1025 - val_acc: 0.9794
Epoch 37/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0016 -
acc: 0.9997 - val_loss: 0.1007 - val_acc: 0.9783
Epoch 38/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0019 -
acc: 0.9996 - val_loss: 0.1083 - val_acc: 0.9766
Epoch 39/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0031 -
acc: 0.9994 - val_loss: 0.1097 - val_acc: 0.9758
Epoch 40/50
60000/60000 [==============================] - 1s 22us/step - loss: 0.0025 -
acc: 0.9994 - val_loss: 0.1123 - val_acc: 0.9774
Epoch 41/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0088 -
acc: 0.9970 - val_loss: 0.1073 - val_acc: 0.9776
Epoch 42/50
60000/60000 [==============================] - 1s 25us/step - loss: 0.0057 -
acc: 0.9983 - val_loss: 0.1031 - val_acc: 0.9777
Epoch 43/50
60000/60000 [==============================] - 1s 23us/step - loss: 0.0032 -
acc: 0.9991 - val_loss: 0.1082 - val_acc: 0.9768
Epoch 44/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0033 -
acc: 0.9990 - val_loss: 0.1056 - val_acc: 0.9776
Epoch 45/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0020 -
acc: 0.9995 - val_loss: 0.1004 - val_acc: 0.9791
Epoch 46/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0012 -
acc: 0.9997 - val_loss: 0.1030 - val_acc: 0.9797
Epoch 47/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0014 -
acc: 0.9996 - val_loss: 0.0974 - val_acc: 0.9800
Epoch 48/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0030 -
acc: 0.9993 - val_loss: 0.1092 - val_acc: 0.9785
Epoch 49/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0033 -
acc: 0.9992 - val_loss: 0.1009 - val_acc: 0.9791
```

```
Epoch 50/50
60000/60000 [==============================] - 1s 24us/step - loss: 0.0042 -
acc: 0.9987 - val_loss: 0.1137 - val_acc: 0.9775
```

[32]:
```
plot(model_hist, 'acc')
plot(model_hist, 'loss')
```





[33]:
```
#Evaluate accuracy on target classifier
pca_adv_scores = model.evaluate(cwa_pert_pca, labels_test)
print("Accuracy: %.2f%%"%(pca_adv_scores[1]*100))
```

```
10000/10000 [==============================] - 0s 41us/step
Accuracy: 80.82%
```

[0]:
```
#Using the defense (and comparing to baseline accuracy)
#FILL THIS IN
# Print the accuracy of the classifier, afterre-training it, on the perturbed␣
 ↪data from #3 ????
```

Part 5: DeepFool

[0]:
```
#Implementing the DeepFool attack

#DeepFool Instance on trained classifier from Part 1
dfa = DeepFool(wrap, sess=sess)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/cleverhans/utils_tf.py:368: The name tf.GraphKeys is deprecated. Please
use tf.compat.v1.GraphKeys instead.
```

[0]:
```
#Attack parameters
dfa_params = {'nb_candidate': 10,
              'max_iter': 50,
              'clip_min': 0.0,
              'clip_max': 1.0}
```

[0]:
```
# Generating adversarial perturbations
dfa_perturbations = dfa.generate_np(data_test, **dfa_params)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/cleverhans/attacks.py:1119: py_func (from
tensorflow.python.ops.script_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
tf.py_func is deprecated in TF V2. Instead, there are two
    options available in V2.
    - tf.py_function takes a python function which manipulates tf eager
    tensors instead of numpy arrays. It's easy to convert a tf eager tensor to
    an ndarray (just call tensor.numpy()) but having access to eager tensors
    means `tf.py_function`s can use accelerators such as GPUs as well as
    being differentiable using a gradient tape.
    - tf.numpy_function maintains the semantics of the deprecated tf.py_func
    (it is not differentiable, and manipulates numpy arrays). It drops the
    stateful argument making all functions stateful.
```

[0]:
```
#Evaluate accuracy of perturbed data on target classifier
dfa_adv_scores = mnist_classifier.evaluate(dfa_perturbations, labels_test)
print("Accuracy: %.2f%%"%(dfa_adv_scores[1]*100))
```

```
10000/10000 [==============================] - 1s 116us/step
Accuracy: 1.14%
```

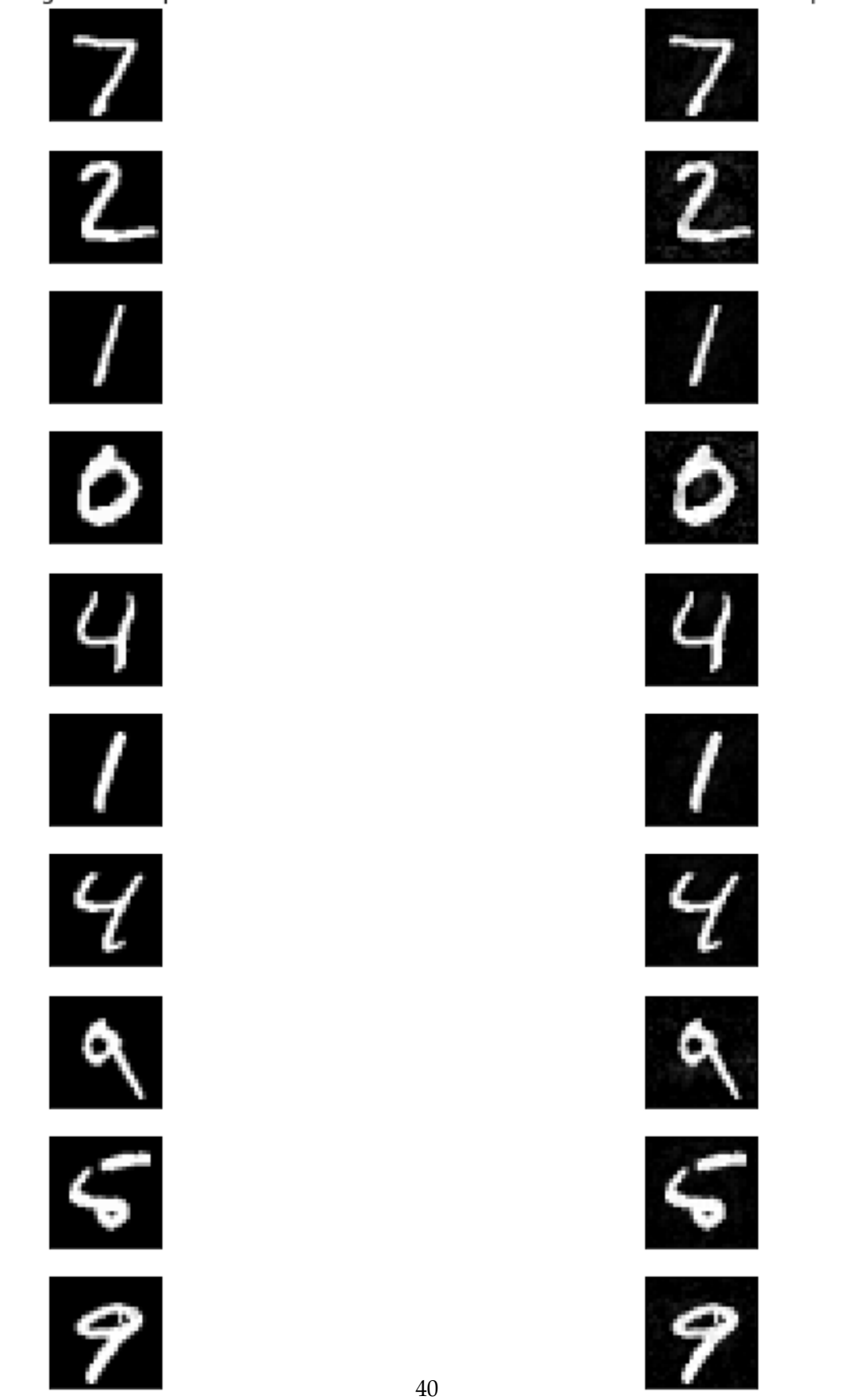
[0]:
```
data_test_show = data_test.reshape(-1, 28, 28)
dfa_perturbations_show = dfa_perturbations.reshape(-1, 28, 28)
```

[0]:
```
print(data_test_show.shape)
print(dfa_perturbations_show.shape)
```

```
(10000, 28, 28)
(10000, 28, 28)
```

[0]:
```
#Show ten original samples and their corresponding adversarial samples
cols = [' Original Samples',
        ' DFA Adversarial Samples']
fig, axes = plt.subplots(nrows=10, ncols=2, figsize=(10, 10))
for i in range(10):
```

```
  axes[i, 0].imshow(data_test_show[i,:,:],
                    cmap=plt.cm.gray)
  axes[i, 1].imshow(dfa_perturbations_show[i,:,:],
                     cmap=plt.cm.gray)
  for ax, col in zip(axes[0], cols):
    ax.set_title(col)
  for j in range(2):
    axes[i, j].get_xaxis().set_visible(False)
    axes[i, j].get_yaxis().set_visible(False)
fig.tight_layout()
plt.show()
```


Original Samples
DFA Adversarial Samples

[0]:
```
#Implementing the Denoising Autoencoder Defense

def autoencoder():
    ae = Sequential()
    ae.add(Dense(400, activation='relu', kernel_initializer="normal",␣
 ↪input_dim=784))
    ae.add(Dense(200, activation='relu', kernel_initializer='normal'))
    ae.add(Dense(100, activation='relu', kernel_initializer='normal'))
    ae.add(Dense(200, activation='relu', kernel_initializer='normal'))
    ae.add(Dense(400, activation='relu', kernel_initializer='normal'))
    ae.add(Dense(784, activation='sigmoid', kernel_initializer='normal'))
    return ae

#Create training data for DAE
df_attack_data_train = dfa.generate_np(data_train, **dfa_params)
data_total_train = np.concatenate([df_attack_data_train, data_train])
labels_total_train = np.concatenate([labels_train, labels_train])
```

[0]:
```
#Create and train DAE graph
ae_model = autoencoder()
ae_model.compile(loss="categorical_crossentropy",
                 optimizer='adam',
                 metrics=['accuracy'])
```

[0]:
```
ae_hist = ae_model.fit(data_total_train, data_total_train,
                       validation_data=(data_test, data_test),
                       shuffle=True,
                       epochs=50,
                       batch_size=256)
```

```
Train on 120000 samples, validate on 10000 samples
Epoch 1/50
120000/120000 [==============================] - 4s 37us/step - loss: 606.4622 -
acc: 0.0057 - val_loss: 534.9524 - val_acc: 0.0093
Epoch 2/50
120000/120000 [==============================] - 3s 28us/step - loss: 576.1069 -
acc: 0.0081 - val_loss: 526.1260 - val_acc: 0.0105
Epoch 3/50
120000/120000 [==============================] - 3s 29us/step - loss: 570.7725 -
acc: 0.0099 - val_loss: 522.5440 - val_acc: 0.0128
Epoch 4/50
120000/120000 [==============================] - 3s 28us/step - loss: 568.2008 -
acc: 0.0112 - val_loss: 520.8387 - val_acc: 0.0126
Epoch 5/50
120000/120000 [==============================] - 3s 28us/step - loss: 566.6137 -
```

```
acc: 0.0116 - val_loss: 519.8814 - val_acc: 0.0129
Epoch 6/50
120000/120000 [==============================] - 3s 28us/step - loss: 565.4910 -
acc: 0.0117 - val_loss: 518.8726 - val_acc: 0.0099
Epoch 7/50
120000/120000 [==============================] - 3s 28us/step - loss: 564.6161 -
acc: 0.0119 - val_loss: 518.3271 - val_acc: 0.0111
Epoch 8/50
120000/120000 [==============================] - 3s 28us/step - loss: 563.9767 -
acc: 0.0122 - val_loss: 517.7158 - val_acc: 0.0121
Epoch 9/50
120000/120000 [==============================] - 3s 28us/step - loss: 563.4158 -
acc: 0.0124 - val_loss: 517.3348 - val_acc: 0.0122
Epoch 10/50
120000/120000 [==============================] - 3s 28us/step - loss: 562.9612 -
acc: 0.0131 - val_loss: 517.1766 - val_acc: 0.0139
Epoch 11/50
120000/120000 [==============================] - 3s 28us/step - loss: 562.5661 -
acc: 0.0127 - val_loss: 516.8055 - val_acc: 0.0120
Epoch 12/50
120000/120000 [==============================] - 3s 28us/step - loss: 562.2514 -
acc: 0.0127 - val_loss: 516.5296 - val_acc: 0.0128
Epoch 13/50
120000/120000 [==============================] - 3s 28us/step - loss: 561.9617 -
acc: 0.0134 - val_loss: 516.4581 - val_acc: 0.0119
Epoch 14/50
120000/120000 [==============================] - 3s 28us/step - loss: 561.6841 -
acc: 0.0132 - val_loss: 516.1995 - val_acc: 0.0113
Epoch 15/50
120000/120000 [==============================] - 3s 28us/step - loss: 561.4894 -
acc: 0.0136 - val_loss: 516.1145 - val_acc: 0.0125
Epoch 16/50
120000/120000 [==============================] - 3s 29us/step - loss: 561.2463 -
acc: 0.0137 - val_loss: 515.8139 - val_acc: 0.0121
Epoch 17/50
120000/120000 [==============================] - 3s 28us/step - loss: 561.0834 -
acc: 0.0143 - val_loss: 515.9232 - val_acc: 0.0121
Epoch 18/50
120000/120000 [==============================] - 3s 28us/step - loss: 560.8825 -
acc: 0.0142 - val_loss: 515.6699 - val_acc: 0.0129
Epoch 19/50
120000/120000 [==============================] - 3s 28us/step - loss: 560.7406 -
acc: 0.0146 - val_loss: 515.3757 - val_acc: 0.0126
Epoch 20/50
120000/120000 [==============================] - 3s 27us/step - loss: 560.5951 -
acc: 0.0144 - val_loss: 515.2499 - val_acc: 0.0141
Epoch 21/50
120000/120000 [==============================] - 3s 28us/step - loss: 560.4891 -
```

```
acc: 0.0143 - val_loss: 515.5526 - val_acc: 0.0145
Epoch 22/50
120000/120000 [==============================] - 3s 28us/step - loss: 560.3356 -
acc: 0.0145 - val_loss: 515.0359 - val_acc: 0.0163
Epoch 23/50
120000/120000 [==============================] - 3s 28us/step - loss: 560.2134 -
acc: 0.0148 - val_loss: 515.0584 - val_acc: 0.0127
Epoch 24/50
120000/120000 [==============================] - 3s 28us/step - loss: 560.1382 -
acc: 0.0147 - val_loss: 515.0900 - val_acc: 0.0124
Epoch 25/50
120000/120000 [==============================] - 3s 28us/step - loss: 560.0317 -
acc: 0.0146 - val_loss: 514.8468 - val_acc: 0.0130
Epoch 26/50
120000/120000 [==============================] - 3s 28us/step - loss: 559.9374 -
acc: 0.0153 - val_loss: 515.0401 - val_acc: 0.0134
Epoch 27/50
120000/120000 [==============================] - 3s 28us/step - loss: 559.8609 -
acc: 0.0151 - val_loss: 514.7410 - val_acc: 0.0145
Epoch 28/50
120000/120000 [==============================] - 3s 28us/step - loss: 559.7781 -
acc: 0.0148 - val_loss: 514.7826 - val_acc: 0.0112
Epoch 29/50
120000/120000 [==============================] - 4s 29us/step - loss: 559.6891 -
acc: 0.0151 - val_loss: 514.7620 - val_acc: 0.0125
Epoch 30/50
120000/120000 [==============================] - 3s 28us/step - loss: 559.6127 -
acc: 0.0153 - val_loss: 514.6291 - val_acc: 0.0158
Epoch 31/50
120000/120000 [==============================] - 3s 28us/step - loss: 559.5683 -
acc: 0.0152 - val_loss: 514.4706 - val_acc: 0.0144
Epoch 32/50
120000/120000 [==============================] - 3s 29us/step - loss: 559.4929 -
acc: 0.0155 - val_loss: 514.5157 - val_acc: 0.0146
Epoch 33/50
120000/120000 [==============================] - 3s 29us/step - loss: 559.4222 -
acc: 0.0157 - val_loss: 514.5098 - val_acc: 0.0137
Epoch 34/50
120000/120000 [==============================] - 3s 28us/step - loss: 559.3625 -
acc: 0.0157 - val_loss: 514.3647 - val_acc: 0.0143
Epoch 35/50
120000/120000 [==============================] - 3s 28us/step - loss: 559.3242 -
acc: 0.0154 - val_loss: 514.4526 - val_acc: 0.0129
Epoch 36/50
120000/120000 [==============================] - 3s 28us/step - loss: 559.2682 -
acc: 0.0153 - val_loss: 514.2844 - val_acc: 0.0155
Epoch 37/50
120000/120000 [==============================] - 3s 28us/step - loss: 559.2117 -
```

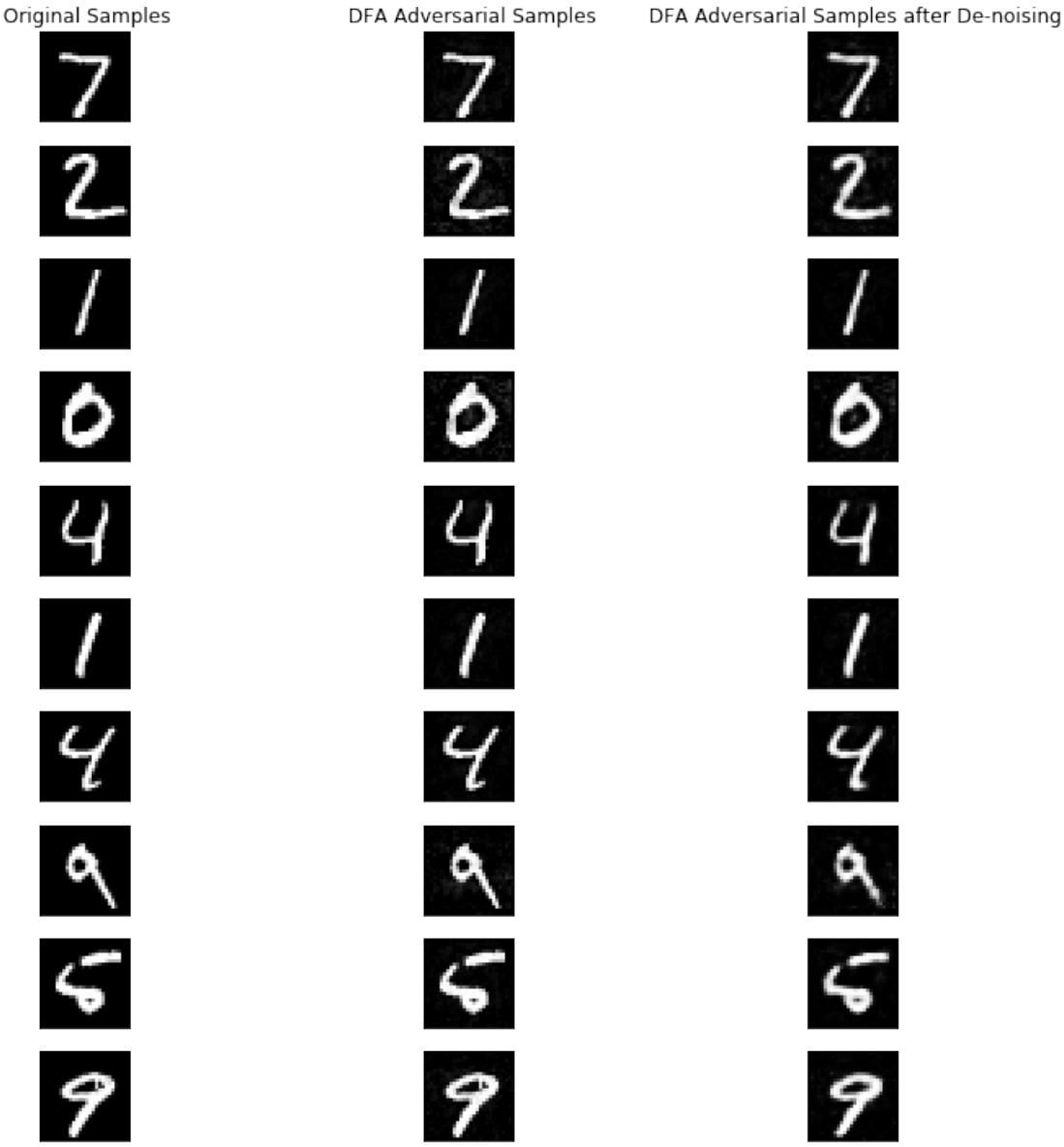
```
acc: 0.0159 - val_loss: 514.3456 - val_acc: 0.0139
Epoch 38/50
120000/120000 [==============================] - 3s 28us/step - loss: 559.1686 -
acc: 0.0155 - val_loss: 514.2649 - val_acc: 0.0159
Epoch 39/50
120000/120000 [==============================] - 3s 28us/step - loss: 559.1017 -
acc: 0.0156 - val_loss: 514.2196 - val_acc: 0.0165
Epoch 40/50
120000/120000 [==============================] - 3s 28us/step - loss: 559.0844 -
acc: 0.0156 - val_loss: 514.1515 - val_acc: 0.0151
Epoch 41/50
120000/120000 [==============================] - 3s 27us/step - loss: 559.0168 -
acc: 0.0160 - val_loss: 514.1221 - val_acc: 0.0174
Epoch 42/50
120000/120000 [==============================] - 4s 29us/step - loss: 558.9881 -
acc: 0.0160 - val_loss: 514.1156 - val_acc: 0.0164
Epoch 43/50
120000/120000 [==============================] - 3s 28us/step - loss: 558.9494 -
acc: 0.0161 - val_loss: 514.1805 - val_acc: 0.0147
Epoch 44/50
120000/120000 [==============================] - 3s 28us/step - loss: 558.8885 -
acc: 0.0164 - val_loss: 514.0478 - val_acc: 0.0168
Epoch 45/50
120000/120000 [==============================] - 3s 28us/step - loss: 558.8644 -
acc: 0.0162 - val_loss: 514.0718 - val_acc: 0.0134
Epoch 46/50
120000/120000 [==============================] - 3s 28us/step - loss: 558.8346 -
acc: 0.0165 - val_loss: 513.9967 - val_acc: 0.0139
Epoch 47/50
120000/120000 [==============================] - 3s 28us/step - loss: 558.7825 -
acc: 0.0163 - val_loss: 514.0060 - val_acc: 0.0134
Epoch 48/50
120000/120000 [==============================] - 3s 28us/step - loss: 558.7659 -
acc: 0.0162 - val_loss: 513.9645 - val_acc: 0.0138
Epoch 49/50
120000/120000 [==============================] - 3s 28us/step - loss: 558.7247 -
acc: 0.0165 - val_loss: 513.9103 - val_acc: 0.0148
Epoch 50/50
120000/120000 [==============================] - 3s 28us/step - loss: 558.7012 -
acc: 0.0166 - val_loss: 513.9919 - val_acc: 0.0175
```

[0]:
```
#Using the defense

#Use DAE to to remove adversarial perturbation
restorations = ae_model.predict(dfa_perturbations)
```

[0]:
```
#Evaluate accuracy on target classifier
dfa_adv_scores = mnist_classifier.evaluate(restorations, labels_test)
print("Accuracy: %.2f%%"%(dfa_adv_scores[1]*100))
```

```
10000/10000 [==============================] - 1s 58us/step
Accuracy: 90.81%
```

[0]:
```
#Show ten samples of adversarial samples after denoising
data_test_show = data_test.reshape(-1, 28, 28)
dfa_perturbations_show = dfa_perturbations.reshape(-1, 28, 28)
restorations_show = restorations.reshape(-1, 28, 28)

cols = [' Original Samples',
        ' DFA Adversarial Samples',
        ' DFA Adversarial Samples after De-noising']
fig, axes = plt.subplots(nrows=10, ncols=3, figsize=(10, 10))
for i in range(10):
  axes[i, 0].imshow(data_test_show[i,:,:],
                    cmap=plt.cm.gray)
  axes[i, 1].imshow(dfa_perturbations_show[i,:,:],
                     cmap=plt.cm.gray)
  axes[i, 2].imshow(restorations_show[i,:,:],
                     cmap=plt.cm.gray)
  for ax, col in zip(axes[0], cols):
    ax.set_title(col)
  for j in range(3):
    axes[i, j].get_xaxis().set_visible(False)
    axes[i, j].get_yaxis().set_visible(False)
fig.tight_layout()
plt.show()
```


Original Samples
DFA Adversarial Samples
DFA Adversarial Samples after De-noising