CS 307 Final Project

Jonathan Tang - jtang207

Aditi Sri Raj Jetthi - 1366 & Kusumsuder

Topic

1. Should detecting auto-encoder for DBA?

Research Question

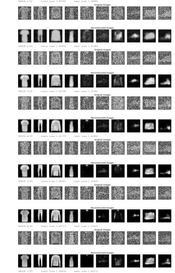
1. Should only detected do more harm than good in terms of improving or yielding better performance of auto-encoder? If so, which every kind of anomaly should be detected by auto-encoder? Why?
2. What extended version of one of the Deep-SVDD models yields the best performance?

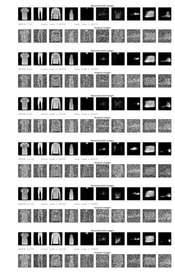
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

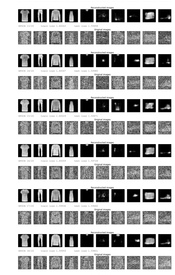
```
# Kernel Functionalities Imported
!pip install -q kaggle
# Utilities
import os, shutil, glob, random, numpy as np, pandas as pd, time, warnings, gc, pickle, zipfile, tarfile, pathlib, urllib.request, hashlib, sys, math, collections, itertools, textwrap, IPython
from pathlib import Path, PurePosix
from collections import defaultdict, Counter, OrderedDict
warnings.filterwarnings('ignore')
np.random.seed(42)
random.seed(42)

# Kernel Functionalities Imported
!pip install -q kaggle
# Scikit-learn
from sklearn.model_selection import train_test_split, StratifiedKfold
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
from sklearn.metrics import roc_auc_score, roc_curve, precision_recall_curve, average_precision_score, confusion_matrix, classification_report, f1_score, precision_score, recall_score, accuracy_score
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans
from sklearn.neighbors import NearestNeighbors
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import VarianceThreshold
from sklearn.utils import shuffle, resample
import scipy.stats as stats
from scipy.spectral import eigvalsh
from scipy.linalg import sqrtm
import scipy.sparse as sp
import joblib, pickle, dill
import matplotlib, matplotlib.pyplot as plt, matplotlib.gridspec as gridspec, matplotlib.patches as mpatches, matplotlib.colors as mcolors, matplotlib.ticker as mticker, matplotlib.mpl_toolkits.mplot3d.axes3d as mpl_3d
import seaborn as sns
import tqdm, tqdm.notebook, tqdm.auto
import nltk, nltk.stopwords, nltk.tokenize, nltk.stem, nltk.corpus
import wordcloud, PIL, PIL.Image, PIL.ImageDraw, PIL.ImageFont
import IPython, IPython.display, IPython.core.display, IPython.core.magics.display
import tabulate, prettytable, rich, rich.console, rich.table, rich.panel, rich.progress
import kagglehub, kaggle, kaggle.cli
import os, sys, gc, psutil, platform, datetime, time, pathlib, shutil, glob, random, numpy, pandas, sklearn, scipy, matplotlib, seaborn, torch, tqdm, PIL, cv2, joblib, dill, rich, kagglehub
print("Kernel Functionalities Imported")
```

```
Kernel Functionalities Imported
Kernel Functionalities Imported
```

```
# Kernel Functionalities Imported
!pip install -q kaggle
# Deep Learning
import torch, torch.nn, torch.nn.functional, torch.optim, torch.utils.data, torch.utils.data.dataset, torch.utils.data.dataloader, torch.utils.data.sampler, torch.utils.data.distributed, torch.utils.data.utils, torch.utils.data.utils.collate, torch.utils.data.utils.pin_memory, torch.utils.data.utils.worker, torch.utils.data.utils.fetch, torch.utils.data.utils.collate_fn, torch.utils.data.utils.pin_memory_fn, torch.utils.data.utils.worker_fn, torch.utils.data.utils.fetch_fn
```

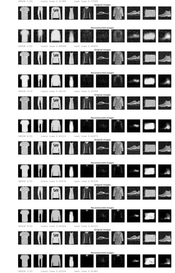






```
# 1 / 1 [==============================] - 0s 0s/step
tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
                       input_shape=(28, 28, 1))
tf.keras.layers.MaxPooling2D(pool_size=(2, 2))
tf.keras.layers.Flatten()

model_layer_outputs = [layer.output for layer in model.layers]
activation_model = tf.keras.models.Model(inputs=model.input,
                                         outputs=model_layer_outputs)
activation_model.summary()

# 1 / 1 [==============================] - 0s 0s/step
tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
                       input_shape=(28, 28, 1))
tf.keras.layers.MaxPooling2D(pool_size=(2, 2))
tf.keras.layers.Flatten()

model_layer_outputs = [layer.output for layer in model.layers]
activation_model = tf.keras.models.Model(inputs=model.input,
                                         outputs=model_layer_outputs)
activation_model.summary()

# 1 / 1 [==============================] - 0s 0s/step
tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
                       input_shape=(28, 28, 1))
tf.keras.layers.MaxPooling2D(pool_size=(2, 2))
tf.keras.layers.Flatten()

model_layer_outputs = [layer.output for layer in model.layers]
activation_model = tf.keras.models.Model(inputs=model.input,
                                         outputs=model_layer_outputs)
activation_model.summary()
```

```
# Preprocess image features from keras.preprocessing.image import image
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
image = load_img('/content/', target_size=(150, 150))
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
# Rescale by 1/255
image = image / 255.0
# Generate batch of augmented images
aug = ImageDataGenerator(rotation_range=40,
                         width_shift_range=0.2,
                         height_shift_range=0.2,
                         shear_range=0.2,
                         zoom_range=0.2,
                         horizontal_flip=True,
                         fill_mode='nearest')
aug_iter = aug.flow(image, batch_size=1)
# Plot the augmented images
for i in range(9):
    plt.subplot(3, 3, i+1)
    batch = next(aug_iter)
    plot_image = batch[0]
    plt.imshow(plot_image)
plt.show()
```



```
epoch 100/100
[1/100] [D: 0.0000] [G: 0.0000]
```