Setup

[ ] ↳ *6 cells hidden*

EDA

Visual Similarity using NN

Here we are trying to implement the research work done on the following paper: https://arxiv.org/pdf/1901.03546v1.pdf

We'll try to implement the same using TensorFlow 2. In the above work, multi-scaled triplet based architecture was proposed, which uses Siamese network and contrastive loss to outperform the current state-of-the-art methods.

Concepts

Visual Recommendation is a crucial feature for any ecommerce platform. It gives the platform power of instantly suggesting similar looking products to what a user is browsing, thus capturing his/her immediate intent which could result in higher customer engagment (CTR) and hence the conversion.

The task of identifying similar products is not trivial as the details concerned here (pattern, structure etc.) are complexely grained in the product image pixels and these product comes in various variety even within the same class. CNNs have showed great understanding and results in this task.

The base of such a system is a CNN extracting key features from product images and returning a vector respresenting those features. When these embeddings for all the products are mapped on an n-dimensional space, it places similar products closer to non-similar items. The nearest neighbours are then the top most visual similar items.

Data Used

Triplet data consists of a query image, positive image (similar to query image) and a negative image (relatively dissimilar to query image as positive image). The query image can either be

Wild Image: where people wearing the cloth in everyday uncontrolled settings. Catalog Image: model wearing cloth in controlled settings as shown in an ecommerce app. While the positive and negative images can also be

In-class: same product category as query image Out-of-class: other product category than query image

The data used in this case study is provided under the following link by the authors of the paper. They have programitically generated these triplets from 4 different data set. The link to the data is:

https://console.cloud.google.com/storage/browser/fynd-open-source/research/MILDNet/

From the above link we were provided with 3 files:

1. train csv
2. val csv
3. test csv

All these files were downloaded, these downloaded csv contains 3 images per row, first and second image being visually similar, while first and third being dissimilar.

Displaying top 10 triplets from each of these dataset

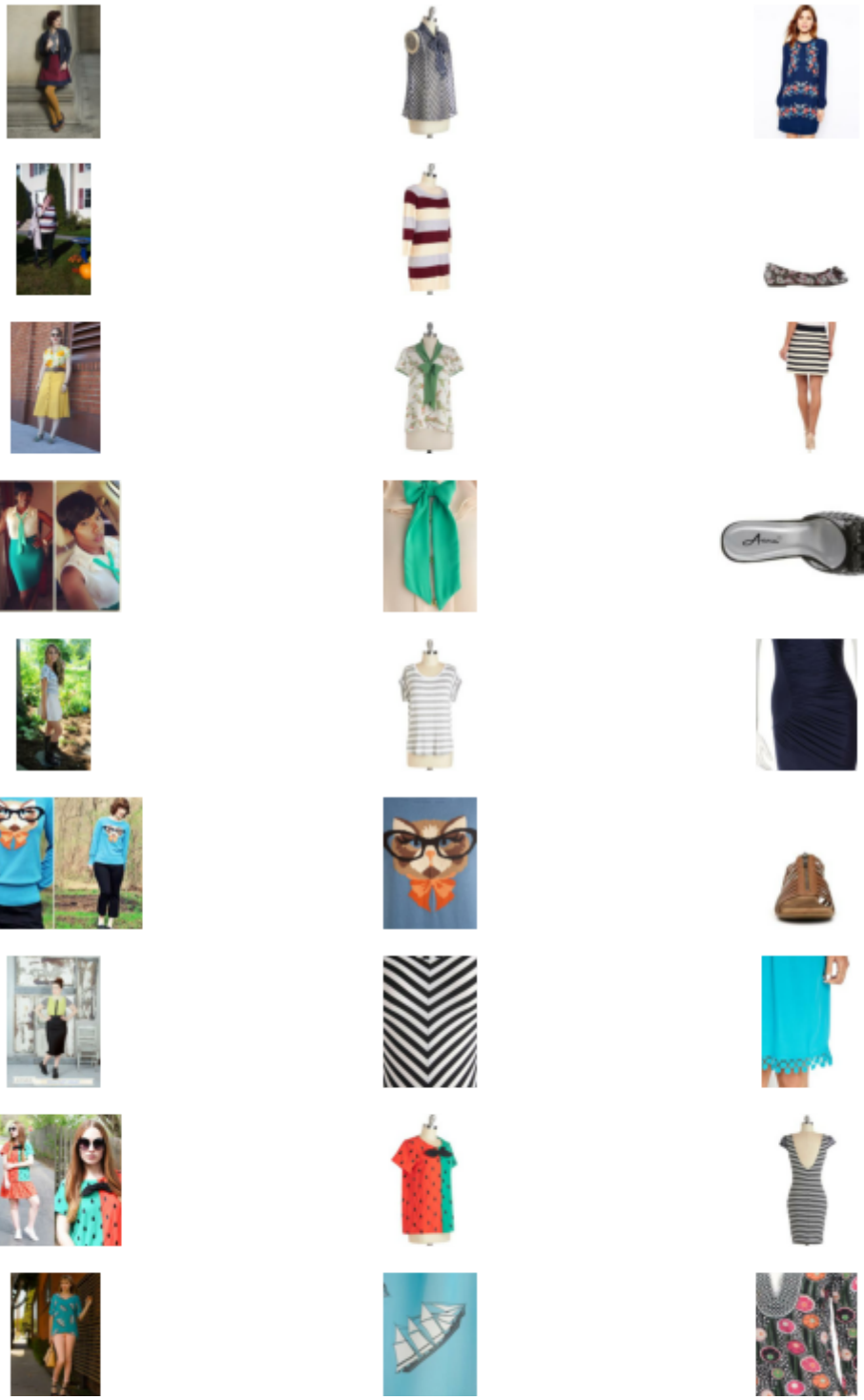
For Train

```
with open('/content/drive/My Drive/data/tops_train.csv', 'r') as file:
  train_images = file.read().split('\n')

fig=plt.figure(figsize=(10, 15))
shuffle(train_images)
subplot_index = 1

for triplet in train_images[:10]:
  triplet = triplet.split(",")
  for index in range(len(triplet)):
    img = tf.image.decode_jpeg(tf.io.read_file("/content/tops/{}".format(triplet[index])),
    fig.add_subplot(10, 3, subplot_index)
    plt.axis('off')
    plt.imshow(img)
    subplot_index += 1

plt.axis('off')
plt.show()
```



For Val

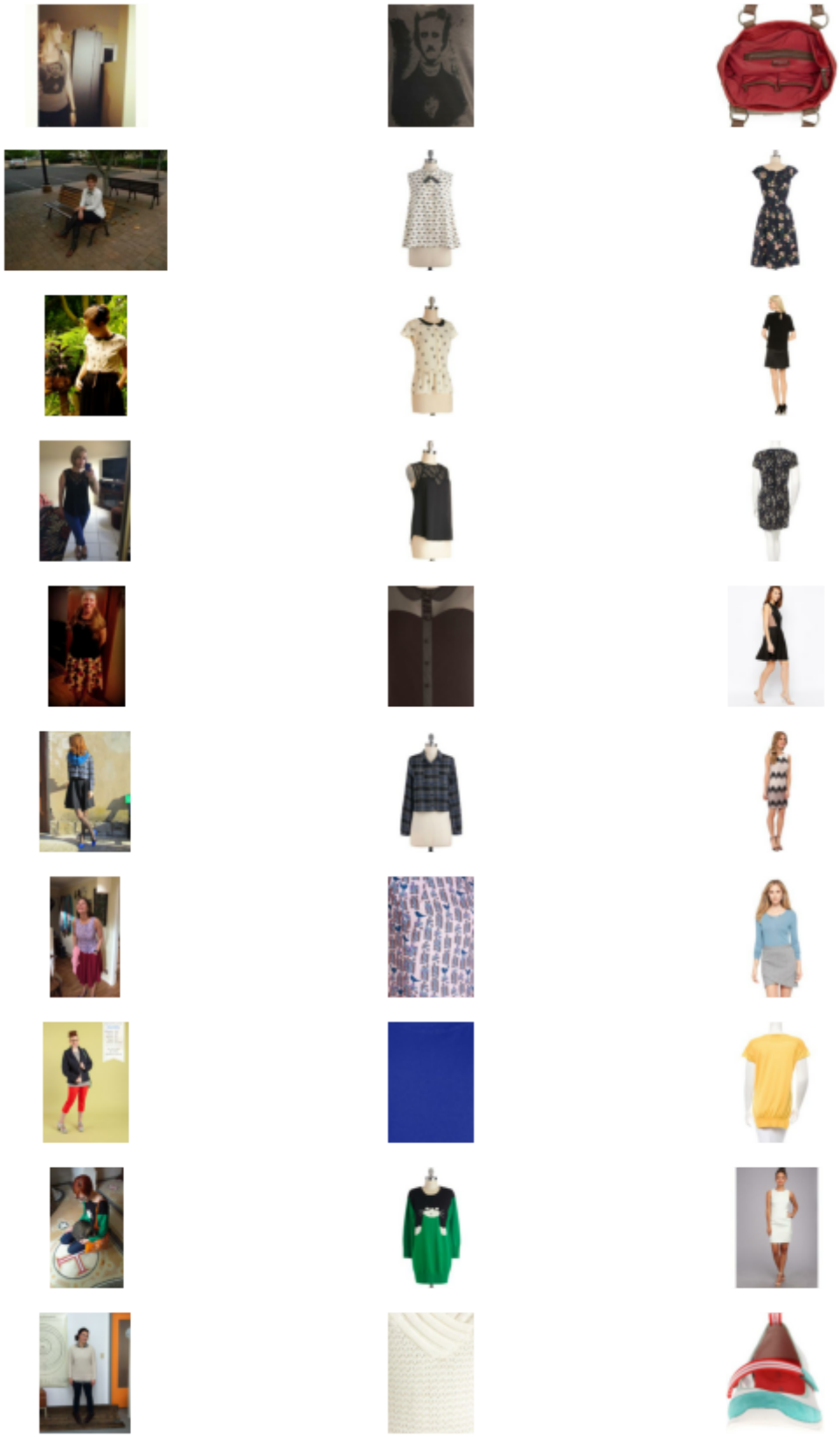
```
with open('/content/drive/My Drive/data/tops_val.csv', 'r') as file:
  train_images = file.read().split('\n')

fig=plt.figure(figsize=(10, 15))
shuffle(train_images)
subplot_index = 1

for triplet in train_images[:10]:
  triplet = triplet.split(",")
  for index in range(len(triplet)):
    img = tf.image.decode_jpeg(tf.io.read_file("/content/tops/{}".format(triplet[index])),
    fig.add_subplot(10, 3, subplot_index)
```

```
        fig.add_subplot(10, 3, subplot_index)
        plt.axis('off')
        plt.imshow(img)
        subplot_index += 1

    plt.axis('off')
    plt.show()
```



For Test

```
with open('/content/drive/My Drive/tops_test.csv', 'r') as file:
  train_images = file.read().split('\n')

fig=plt.figure(figsize=(10, 15))
shuffle(train_images)
subplot_index = 1

for triplet in train_images[:10]:
  triplet = triplet.split(",")
  for index in range(len(triplet)):
    img = tf.image.decode_jpeg(tf.io.read_file("/content/tops/{}".format(triplet[index])),
    fig.add_subplot(10, 3, subplot_index)
    plt.axis('off')
    plt.imshow(img)
    subplot_index += 1

plt.axis('off')
plt.show()
```