International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 04 | Apr 2021 www.irjet.net p-ISSN: 2395-0072

Deep Learning Based Product Recommendation System and its Applications

Akshit Tayade[1], Vidhi Sejpal[2], Ankit Khivasara[3]

[1,2]Department of Electronics and Telecommunication Engineering
[3]Assistant Professor, Department of Electronics and Telecommunication Engineering
K J Somaiya College of Engineering, Mumbai - 400077, India

Abstract - *On the Internet, where the number of choices is overwhelming, there is a need to filter, prioritize and efficiently deliver relevant information in order to alleviate the problem of information overload, which has created a potential problem for many Internet users. Recommender systems solve this problem by searching through large volumes of dynamically generated information to provide users with personalized content and services. In view of more personalized clothing requirements, an intelligent clothing recommendation system was designed and developed in this paper. By the use of Transfer Learning to elicit the rich information from the product images, and the use of cosine similarity approach, the user is provided with eclectic recommended products depending on their choices. This was implemented on a web e-commerce application, where users have a wide range to choose their clothing apparel from our database. Only those images are recommended to users which have 80% or above similarity with the product chosen by the user, which helps in solving personalized recommendations.*

Key Words - CNN, Fashion dataset, Transfer Learning, Visual Similarity.

1. INTRODUCTION

With the advent of emerging technologies and the rapid growth of the Internet, the world is moving towards an E-world where most things are digitized and available with a mouse click. E-Commerce is steadily becoming more important in changing the way people buy products and services. Customers are buying more and more products on the website. Whenever a customer wants to buy a product on the web, they visit an online store and look for items of their interest. There are many E-commerce applications on the internet today. A common shortcoming of every application is the lack of customer service. To overcome this problem, we have constructed a product recommendation system.

Recommendation Systems are becoming an emerging field in the world of E-commerce applications and web services. In online shopping, it takes a lot of time to look out for various products. A recommendation system can speed up finding a wide variety of various products that customers are interested in. The use of this efficient recommendation system is increasing day by day as it is easy and reliable for a customer to shop online and find the perfect options for them without any difficulties.

Recommender systems are beneficial to both service providers and users [1]. Prior studies demonstrate that recommendation engines help consumers to make better decisions, reduce search efforts and find the most suitable prices [2].

A product recommendation system is a software tool designed to generate and provide suggestions for items or content a specific user would like to purchase or engage with. Product recommendation will analyze the existing things more specifically, we focus on fashion products and develop a method that only requires a single input image to return a ranked list of similar-style recommendations.

This paper deals with the implementation of an e-commerce application for the sale of different types of products categorized under similar features. Deep Learning technique is implemented using Transfer Learning and then cosine similarity for providing most similar items related to the user's product choices. The features of the system include product details and similar item search methods. Additional implementation includes sending an email to the user about their purchased items and giving an estimated date of delivery for individual products.

2. LITERATURE REVIEW

In a very general way, recommender systems are algorithms aimed at suggesting relevant items to users (like items being movies to watch, text to read, products to buy, or anything else depending on industries). From e-commerce (suggest to buyers articles that could interest them) to online advertisement (suggest to users the right contents, matching their preferences), recommender systems are today unavoidable in our daily online journeys. The purpose of a recommender system is to suggest relevant items to users. To achieve this task, there exist two major categories of methods: collaborative filtering methods and content-based methods [3].

2.1. COLLABORATIVE FILTERING

Collaborative methods for recommender systems are methods that are based solely on the past interactions recorded between users and items to produce new recommendations. These interactions are stored in the so-called "user-item interactions matrix". Then, the main idea that rules collaborative methods is that these past user-item interactions are sufficient to detect similar users and/or similar items and make predictions based on these estimated proximities. The class of collaborative filtering algorithms is divided into two sub-categories that are generally called memory-based and model-based approaches [3]. However, as it only considers past interactions to make recommendations, collaborative filtering suffers from the "cold start problem": it is impossible to recommend anything to new users or to recommend a new item to any users and many users or items have too few interactions to be efficiently handled.

2.2. CONTENT-BASED FILTERING

Unlike collaborative methods that only rely on user-item interactions, content-based approaches use additional information about users and/or items. Then, the idea of content-based methods is to try to build a model, based on the available "features", that explain the observed user-item interactions. Content-based methods suffer far less from the cold start problem than collaborative approaches: new users or items can be described by their characteristics (content) and so relevant suggestions can be done for these new entities.

2.3. VISUAL SIMILARITY

Considering the drawbacks of collaborative and content-based filtering of cold start problems, where new users might face issues. We propose a visual similarity approach to eliminate the cold start problem.

In this method, the users are recommended clothing apparel based on the similarity of the image with its corresponding images from the database. This approach even recommends images to a new user, provided the user doesn't have any history, which makes the process unique and robust. Recommendations are based on cosine similarity between the images from the database which generates scores between one vs rest images. These scores are then sorted descendingly. Then only the top 10 scores are considered and an image corresponding to that score is provided to the user. The recommendations provided contain eclectic images throughout the database of clothing apparel.

Fig -1: Recommendation based on visual similarity

In figure 1, the first row shows an original image, and the second, third row are recommendations with a similarity score to the original image provided to the user.

Cosine similarity is a metric used to measure how similar two items are. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The output value ranges from 0–1. Where 0 means no similarity, whereas 1 means that both the items are 100% similar. The python Cosine Similarity or cosine kernel computes similarity as the normalized dot product of input samples X and Y.

$$\text{similarity}(A, B) = \frac{X \bullet Y}{||X|| \times ||Y||} = \frac{\sum_{i=1}^{n} X_i \times Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2} \times \sqrt{\sum_{i=1}^{n} Y_i^2}}$$

For extracting information from images, we have used a convolutional neural network [4]. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets can learn these filters/characteristics.

A kernel is a matrix, which is slid across the image and multiplied with the input such that the output is enhanced in a certain desirable manner. Instead of using manually made kernels for feature extraction, through Deep CNNs, we can learn these kernel values which can extract latent features. A fully connected layer connects every input with every output in his kernel term. For this reason, kernel size = n_inputs * n_outputs. It also adds a bias term to every output bias size = n_outputs. Usually, the bias term is a lot smaller than the kernel size so we will ignore it.

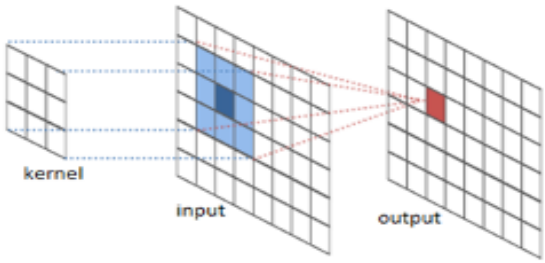

Fig -2: Working on the kernel in CNN

Stride is a component of convolutional neural networks or neural networks tuned for the compression of images and video data. Stride is a parameter of the neural network's filter that modifies the amount of movement over the image or video. For example, if a neural network's stride is set to 1, the filter will move one pixel, or unit, at a time. The size of the filter affects the encoded output volume, so stride is often set to a whole integer, rather than a fraction or decimal.

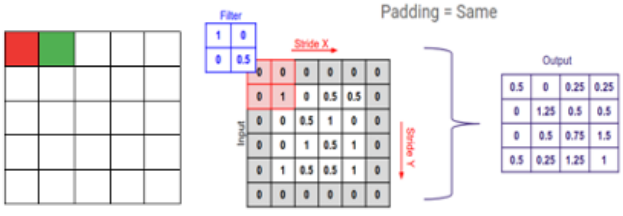

Fig -3: stride=1 (left) and padding in CNN (right)

Padding is a term relevant to convolutional neural networks as it refers to the number of pixels added to an image when it is being processed by the kernel of a CNN. For example, if the padding in a CNN is set to zero, then every pixel value that is added will be of value zero. If, however, the zero-padding is set to one, there will be a one-pixel border added to the image with a pixel value of zero. Padding works by extending the area in which a convolutional neural network processes an image. The kernel is the neural networks filter which moves across the image, scanning each pixel and converting the data into a smaller, or sometimes larger, format. To assist the kernel with processing the image, padding is added to the frame of the image to allow for more space for the kernel to cover the image. Adding padding to an image processed by a CNN allows for a more accurate analysis of images.

Transfer learning **[5]** is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

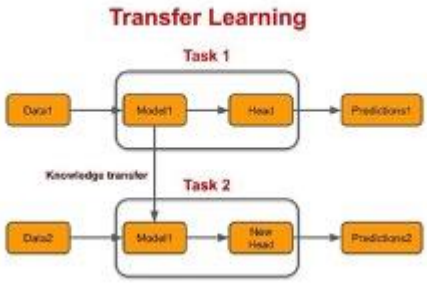

Fig -4: Transfer Learning method

It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in a skill that they provide on related problems. The pre-trained models have weights assigned so that we don't have to train the model again.

Weight is the parameter within a neural network that transforms input data within the network's hidden layers. A neural network is a series of nodes or neurons. Within each node is a set of inputs, weight, and a bias value. As an input enters the node, it gets multiplied by a weight value and the resulting output is either observed or passed to the next layer in the neural network. Often the weights of a neural network are contained within the hidden layers of the network.

3. METHODOLOGY

3.1. OBTAINING DATASET

To get eclectic clothing apparel, we used the unique dataset available at Kaggle [6] which contains fashion apparel such as shoes, heels, rings, clothes. This dataset had a range of fashion apparel, but we made use of only clothing apparel for males and females. The dataset comes with a text file containing fields like brand name, product name, product type, label, image file. By the use of column: product type, we eliminate those items rows that were not needed. After removing unnecessary fashion apparel, our database had cleaned and essential clothing images. We also used the Large-scale apparel dataset [7] which was fetched from DeepFashion Database. Four benchmarks concerning Attribute Prediction, Consumer-to-shop Clothes Retrieval, In-shop Clothes Retrieval, and Landmark Detection are developed using the Deep Fashion database [8]. The data and annotations of these benchmarks are employed as the training and test sets for the following computer vision tasks, such as Clothes Detection, Clothes Recognition, and Image Retrieval. This dataset contains diverse fashion images ranging from well-posed shop images which had 14 different female clothing apparel and 9 different male clothing apparel. Over 5000 images were collected and combined, separating female and male items.

Fig -5: Diverse images of male, female clothes from the dataset

3.2. DATA PREPROCESSING

The given images from our dataset are of different dimensions. So, it is required for every image to be of the same dimensionality while loading an image before training the model and, this step is performed using the Keras library [9]. And all the images are loaded and converted into one fixed dimension array (i.e. (height, width, channel) = (244,244,3)). Now we have our image array in three dimensions (i.e. (height, width, channel)), but for training our Deep learning model, there is a requirement of four dimensions.

Shapes are consequences of the model's configuration. Shapes are tuples representing how many elements an array or tensor has in each dimension. In Keras, the input layer itself is not a layer, but a tensor. It's the starting tensor sent to the first hidden layer. This tensor must have the same shape as your training data. For example: if you have 10 images of 224x224 pixels in RGB (3 channels), the shape of your input data is (10,224,224,3). Here the fourth dimension represents the batch format which is required by Keras. Deep learning models are trained depending upon batches created. Preparing each image with the corresponding dimensions is performed repeatedly for all images to form batches. Then these batch images are stacked in sequence vertically (i.e., row-wise) forming a single array. Now the dataset is preprocessed the way required by the Deep learning model. Figure 6 shows the data processing step for a single image. And the same procedure is applied for the 'n' number of images in the dataset.

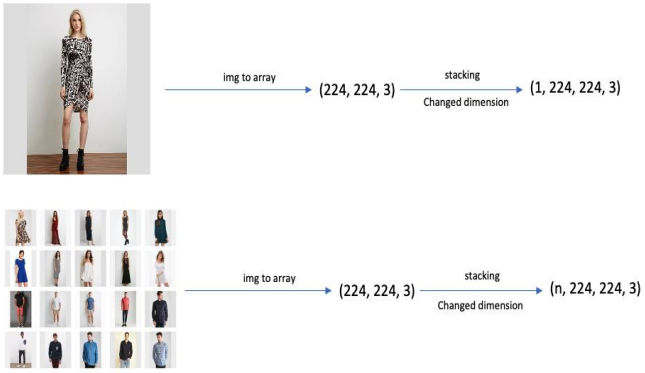

Fig -6: Data preparation process

3.3. FEATURE EXTRACTION MODEL

The feature extraction model will extract all rich information from the image as features. Our initial approach was to detect as many features as possible from a given image using CNN. Generally, more deep convoluted networks are required when more features have to be extracted. In our case, we wanted to extract most of the rich features detected for particular clothing apparel, so we decided to go with Transfer Learning. There was a choice between high precision or lesser training time while extracting features from a huge dataset, we wanted to elicit rich pieces of information from an image, hence choosing high precision. We choose a pre-trained VGG16

[10] model. It was also noticed that it performed well on large-scale data [10]. Considering VGG weights size and training time, it took a huge training time.

As we just wanted to extract rich features from a pre-trained model and not do prediction/classification problems with our dataset, VGG16's prediction layer was not needed (i.e., output layer). After removing the output layer from the pre-trained VGG16 model, the model summary is given in Figure 7.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312

Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0

Fig -7: model summary

Now the preprocessed images are passed through the prediction model, giving features detected by each image. Each prediction generated an array. This process took about two hours to train.

3.4. COSINE SIMILARITY MODEL

Cosine similarity is a metric used to determine how similar the documents are irrespective of their size. When vectors point in the same direction, cosine similarity is 1; when vectors are perpendicular, cosine similarity is 0; and when vectors point in opposite directions, cosine similarity is -1. In positive space, cosine similarity is the complement to cosine distance: cosine similarity = (1 – cosine distance). The Values range between -1 and 1, where -1 is perfectly dissimilar and 1 is perfectly similar.

$$\text{similarity}(A, B) = \frac{A \bullet B}{||A|| \times ||B||} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

We then use the concept of similarity on our predicated array obtained from the VGG16 model. A particular image's predicted array is cosine with the rest images array. This generates a 2D matrix containing similarity scores of one vs rest images. Then the cosine similarity matrix is converted into a data frame.

3.5. MODEL ARCHITECTURE

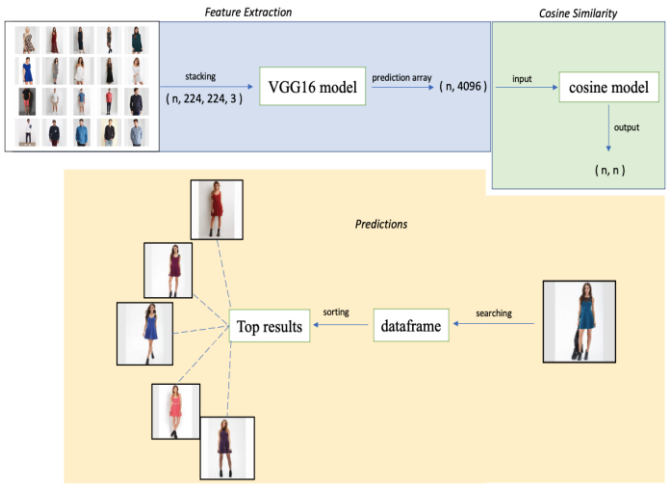

Fig -8: Model Architecture

The Visual Similarity model is implemented on "Web application" as shown in Figure. Once the image is passed through the data frame created by the "cosine similarity model", we get a 1D array containing similarity scores with respect to other images in the database. This 1D array is sorted descending, giving top scores at the beginning of an array. From this sorted 1d array, we choose the top 'n' results we want.

As shown in the Figure 8, after reaching a particular product elected by the user, it has generated top 5 results similar to the product. A detailed implementation of the model on the web application is explained in the next section.

In order to facilitate online purchase User Accounts and Guest Accounts are provided to the user. Our system is implemented using a 3-tier approach, web browser as our front-end client, Flask server as a middle tier, and backend database. This project allows viewing various products available and purchasing them, enables registered and

Guest users to purchase desired products instantly via various payment platforms.

4. IMPLEMENTATION

4.1. WEB APPLICATION

The Product Recommendation System was successfully implemented in a local virtual environment using a Flask python script and some HTML templates. Users are supposed to select the collection between male and female and once they choose a particular product the deep learning models working at the back end works to get similar product items as selected by a user Now by clicking on the "Similar Items" button to check out the recommendations for the user.

4.2. MAIL GENERATION

A mail is generated to send all the details of the user and the summary of the product purchased. It is auto-generated using Flask Mail and it is sent to a user after the successful completion of the payment through Razorpay API. It contains the following details.

4.2.1. Product Name along with the prices
4.2.2. The total amount of all the products purchased.
4.2.3. User personal details.
4.2.4. Estimated delivery date.

5. RESULT

The similarity score produced by the Deep learning model is shown in fig. Where index and column are the product images throughout the database, and each row indicating the similarity scores with respect to other images. Score 1 indicates that the item is completely similar whereas when the score approaches near 0, it indicates that the other items are not much similar to the current item. These scores determine how similar one image is to the rest images. We have chosen to give the top 'n' scores after the user selects a product on which they need similar search results from the database.

	../input/style-color-images/style/0_0_001.png	../input/style-color-images/style/0_0_002.png	../input/style-color-images/style/0_0_003.png	../input/style-color-images/style/0_0_004.png	../input/style-color-images/style/0_0_005.png
../input/style-color-images/style/0_0_001.png	1.000000	0.554047	0.558606	0.569193	0.521644
../input/style-color-images/style/0_0_002.png	0.554047	1.000000	0.352095	0.421485	0.522096
../input/style-color-images/style/0_0_003.png	0.558606	0.352095	1.000000	0.818573	0.382171
../input/style-color-images/style/0_0_004.png	0.569193	0.421485	0.818573	1.000000	0.416590
../input/style-color-images/style/0_0_005.png	0.521644	0.522096	0.382171	0.416590	1.000000

Fig -9: Similarity scores data frame

Once the user selects a particular product from the web application, the product item is searched throughout the dataframe, returning a single 1D array of all similarity scores compared with other products from the database. Then this array is sorted descending giving the highest scores at the beginning of the array. After sorting, the user is provided with a top 10 similarity score from the array, and corresponding to these scores are the product images which are then displayed on the web application as shown in Figure 10.

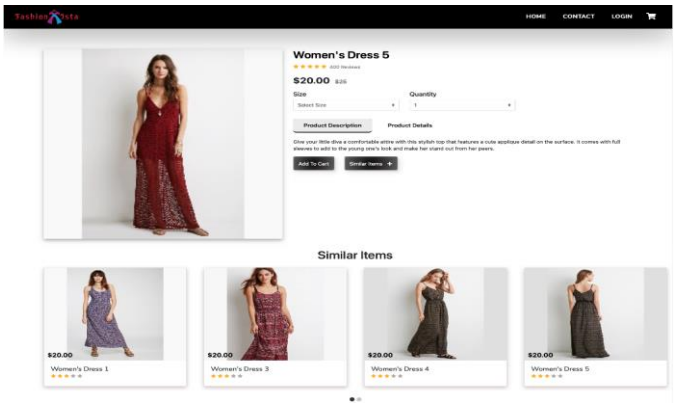

Fig -10: Similar items to the given product

6. CONCLUSION

An intelligent clothing recommendation system based on visual similarity deep learning network running on Web platform was designed and implemented in this paper. After collecting existing clothing images from various databases, and applying a deep learning network to build a recommender system. The proposed system for recommending clothing apparel to users using a visual similarity approach also proves to diminish the cold start approach in other filtering approaches. After using a random image sample for testing, the system was applied to give clothing recommendations, which had good performance (giving similarity scores above 80%) and reliability, and showed great potential in solving personalized recommendations.

ACKNOWLEDGMENT

We extend our gratitude towards our guide, Professor Ankit Khivasara for mentoring us in the mini-project, reviewing, and immensely supporting us during our implementation.

REFERENCES

[1] Pearl Pu, Li Chen, Rong Hu, (October 2011), "A user-centric evaluation framework for recommender systems", Proceedings of the fifth ACM conference on Recommender systems, https://doi.org/10.1145/2043932.2043962

[2] Hessel Tuinhof, Clemens Pirker, & Markus Haltmeier, (July 17, 2018), "Image Based Fashion Product Recommendation with Deep Learning", https://arxiv.org/pdf/1805.08694.pdf

[3] Pijush Kanti Dutta Pramanik, Avick Kumar Dey & Prasenjit Choudhury, (January 2021), "Recommender systems: an overview, research trends, and future directions", International Journal of Business and Systems Research, https://www.researchgate.net/publication/339172772_Recommender_Systems_An_Overview_Research_Trends_and_Future_Directions

[4] Saad Albawi; Tareq Abed Mohammed; Saad Al-Zawi,(2017), "Understanding of a Convolutional Neural Network", International Conference on Engineering and Technology (ICET), 10.1109/ICEngTechnol.2017.8308186

[5] Mahbub Hussain, Jordan J. Bird, and Diego R. Faria, (June 2018), "A Study on CNN Transfer Learning for Image Classification", UKCI 2018: 18th Annual UK Workshop on Computational Intelligence, https://www.researchgate.net/publication/325803364_A_Study_on_CNN_Transfer_Learning_for_Image_Classification

[6] Olga Belitskaya, "Style Color Images", https://www.kaggle.com/olgabelitskaya/style-color-images

[7] "Large-scale Fashion (DeepFashion) Database", http://mmlab.ie.cuhk.edu.hk/projects/DeepFashion.html

[8] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, Xiaoou Tang, (2016), "DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 10.1109/CVPR.2016.124

[9] Ekaba Bisong, (September 2019), "Building Machine Learning and Deep Learning Models on Google Cloud Platform", TensorFlow 2.0 and Keras (pp 347-399), Springer, 10.1007/978-1-4842-4470-8_30

[10] Ç.F. Özgenel and A. Gönenç Sorguç, (2018), "Performance Comparison of Pretrained Convolutional Neural Networks on Crack Detection in Buildings", 35th International Symposium on Automation and Robotics in Construction, https://www.iaarc.org/publications/fulltext/ISARC2018-Paper154.pdf