2023 5th IEEE International Conference on Sustainable Technologies
for Industry 5.0 (STI), 09-10 December, Dhaka

AutoSimNet - A Visually Similar Product Recommendation System For E-commerce

Musanna Abdullah Bayezid, Asiqul Islam and Mr. M. M. Fazle Rabbi
Department of Computer Science and Engineering, Bangladesh University of Business and Technology (BUBT)
Email: musanna324@gmail.com, asiq.dev@gmail.com, rabbi@bubt.edu.bd

***Abstract*—In the dynamic landscape of modern E-commerce, the strategic role of visual similar recommendations has become paramount in enhancing user engagement and driving sales. With the proliferation of online shopping, consumers are increasingly reliant on digital platforms to discover and purchase products. Here, we proposed the AutoSimNet model, an effective e-commerce visually identical product recommendation methodology for visually comparable suggestions. In order to accomplish our goal, in this paper we proposed a two steps model. In the first stage, we extracted image features from the images and created embeddings using a deep learning approach called DAE (Denoising Auto-encoder). In the second stage, the KNN (K-Nearest Neighbor classification) algorithm was used to build an index tree. Additionally, we used a distance metric to collect the top-N products that are almost identical. Our âAutoSimNetâ model not only underscores the crucial significance of recommendation mechanisms in contemporary E-commerce but also introduces an innovative approach in the state-of-the-art to address the emerging need for visually guided product recommendations.**

***Index Terms*—Recommendation, Visually Similar, Clustering Algorithm, deep learning, fashion recommendation, Image similarity, E-commerce products, Denoising Auto-encoder, KNN.**

I. Introduction

With the increasing diversity of products available on e-commerce platforms, customers are confronted with the task of locating their preferred items. This issue has led to a significant interest in recommender systems as a potential solution to address this challenge[1]. Recommender systems are helpful guides for users when they're looking at lots of different products. These systems use a lot of information about the products and things that users do, like looking at items, buying things, or visiting web pages. They use all this information to figure out the best suggestions to give to customers, like when to show them, what to show them, and how to show them, all based on what the users are interested in. Prior research has shown that recommendation engines aid customers in making more accurate choices, reducing search time, and locating the best rates[2]. Recommender systems have developed into a crucial component of all major Internet merchants, accounting for up to 35% of Amazon sales [3] or more than 80% of Netflix material seen[4].

A significant share of e-commerce sales comes from the fashion and lifestyle sector, encompassing clothing, shoes, bags, accessories, and more [5]. As a result, enhancing the user's exploration journey for fashion items becomes a crucial area of attention.One distinctive feature of the fashion sector is that a user's choice to buy is mainly guided by the visual appeal of the product. Therefore, it's pivotal for the primary discovery methods, namely "Search" and "Recommendations," to consider the visual characteristics present in the images linked to these items[6], [7]. Visual similarity recommendations work by showing users products that look similar to the ones they're interested in. This can be really helpful because sometimes people might not know the exact name or description of what they're looking for, but they can recognize something similar when they see it. By suggesting visually similar items, the recommendation system helps users explore a wider range of options that align with their tastes, making their product search experience more enjoyable and efficient. This paper introduces a distinctive, data driven methodology where consumer choices are autonomously retrieved from the accessible customer information. To be more precise, In this work, we concentrate on fashion items and create a system that just needs one input image to produce a ranked list of recommendations for similar style clothing. The suggested recommendation approach performs in a two phase mode. In the initial phase, extract image features into latent space using Denoising Autoencoder (DAE)[8] to reduce the dimensionality of input data. In the second step, the extracted features serve as inputs for the ranking system (KNN). Although we focus on fashion products in this research, similar recommendation systems can be used for various other kinds of products.

The primary contributions of the proposed system are outlined below:

- **Innovative Fusion of Techniques:** Our paper introduces a pioneering blend of Denoising Autoencoder (DAE) and KNN ranking to create a product recommendation system, enhancing relevance and accuracy.
- **Enhanced Visual Feature Extraction:**The DAE component effectively transforms raw images into a compact, meaningful latent space representation, facilitating nuanced visual similarity detection.
- **Personalized Ranking Precision:** The DAE component effectively transforms raw images into a compact, meaningful latent space representation, facilitating nuanced visual similarity detection.
- **Latent representation:** The DAE component effectively transforms raw images into a compact, meaningful latent space representation, facilitating nuanced visual similarity detection.

The rest of the document is organized in the following manner: In Sect. 2 describes the related literature. The proposed method of the AutoSimNet is discussed in Sect. 3 where problem formulations, overview and motivations are included. Experimental analysis, dataset and performance evaluations are discussed in Sect. 4. Finally, the Discussion is in Sect. 5 and The study's conclusions are outlined in Section 6.

II. Related Work

Recommenders typically generate suggestions using two distinct approaches: content-based filtering and collaborative filtering[9].The Content-based recommendations assess the characteristics linked to both the products and the customers to determine suitable recommendations [10]. The widely used approach is user-based, referred to as collaborative filtering (CF), wherein customers' past actions such as purchases and product ratings are utilized to uncover the connections among consumers and items.[11]. Collaborative filtering recommends items to consumers based on items purchased by other customers who have similar interests, as well as considering the customers' past behaviors. It doesn't require specific product expertise and generates recommendations by utilizing underlying characteristics of both customers and products. The majority of collaborative filtering models encounter challenges with the "cold start" issue, as new products or customers lack connections to other existing products or customers. [12]. For this reason, CF-based strategies have been preferred to content-based ones. Singular Value Decomposition (SVD) is commonly employed in collaborative filtering; as the user-item matrix becomes large, it becomes difficult to fill in the missing values and requires a lot of memory space. Hence, LFM (Latent Factor Model) is used in recent work modeling on such observed data [13], this technique breaks down the user matrix from the user-item matrix and the item matrix, aiming to reduce the discrepancies in observed ratings. The neural collaborative filtering method is based on deep learning that is proposed in [14], which extends the widely employed matrix factorization approach found in various collaborative filtering methods. In the [15], use a combination strategy that combines a deep learning model that extracts both visual and latent non-visual user features with a matrix factorization-based predictor. These approaches are constrained by the ranking system recommendations since MF methods demand product ratings and labeling. This paper [16] contains a current comprehensive analysis of recommender systems based on deep learning.

Due to advancements in deep learning, it's possible to extract latent features from product images or descriptions [17]. Product images can be transformed into features using Convolutional Neural Networks (CNN), aiding the recommender system in identifying correlations between two products in paper [18]. In [19] a CNN is employed to classify fashion classes, involving 4 stages: apparel type, attribute analysis, retrieval, and object identification. The paper [20] presents a two-stage fashion recommendation system that employs a CNN for image classification and a ranking mechanism to provide recommendations. The approach for ranking is based on k-nearest neighbors (k-NN) for recommendations.

Some scholars have directed their attention towards generative models [21], [22]. Denoising Autoencoder can be considered a type of generative model, which reconstructs clean data from noisy or corrupted input [23]. In the majority of instances, this technique is applied to enable the network to learn robust features and patterns by filtering out noise during training. Within our approach, we employ a Denoising Autoencoder to reveal underlying connections among various categories of clothing items. We demonstrate that recommendation systems that just depend on visual criteria are appropriate since that is able to provide quite visual fashion product recommendations for items with a similar design. This could also be advantageous for new customers, especially when there's no existing historical user data to rely upon.

III. Methodology

Building an image-based product recommendation system involves extracting visual features from images and calculating similarity distances between an input image and a set of images. This process is complex and encompasses steps such as data collection, image feature extraction, dimensionality reduction, similarity calculation, user input handling, recommendation ranking, personalization, performance evaluation, and deployment. Here we are proposing an E-commerce similar image network (AutoSimNet) model for visual similar product recommendations, as shown in Fig:1.

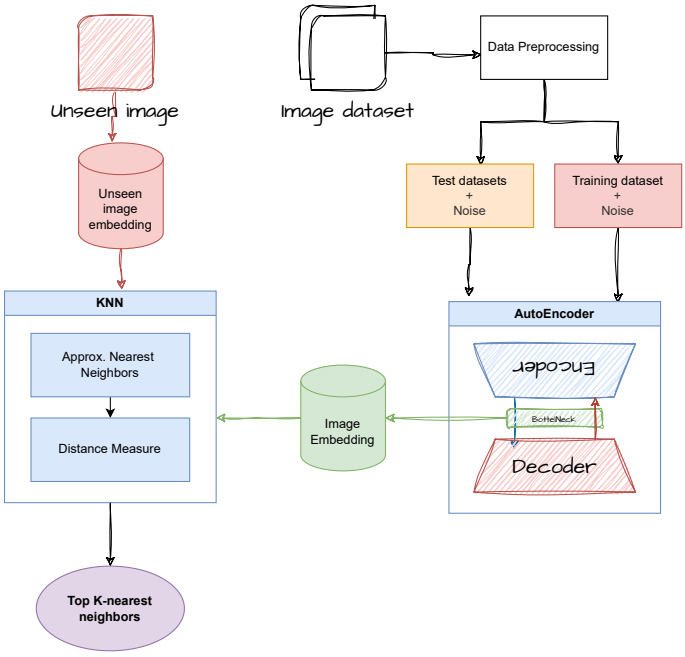

Fig. 1: AutoSimNet architecture

Our AutoSimNet chose the autoencoder network approach for feature extraction and embedding before the clustering stage. What we have done here is, adopted an CNN based denoising autoencoder to do feature engineering and to get

embedding using the encoder when the autoencoder is trained. In order to keep our model simple and lightweight enough we constructed our own model architecture.fig2. We used three layers for Encoder and decoder to build a model. As we can see, we introduced a 3×3 size Kernel in the convolutional layer and 2×2 Max Pooling layer and a Batch Normalization After every layer Table I.

Layer	Output Shape	Kernel size	Filters	strides	Parameter
InputLayer	(64, 64, 1)				0
Conv2D	(64, 64, 32)	3×3	32		320
MaxPooling2D	(64, 64, 32)	2×2	32		0
BatchNormalization	(32, 32, 32)		32		128
Conv2D	(32, 32, 32)	3×3	32		9248
MaxPooling2D	(16, 16, 32)	2×2	32		0
BatchNormalization	(16, 16, 32)		32		128
Conv2D	(16, 16, 32)	3×3	32		9248
MaxPooling2D (Bottleneck)	(8, 8, 32)	2×2	5		0
Conv2DTranspose	(16, 16, 32)	3×3	32	2	9248
Conv2DTranspose	(32, 32, 32)	3×3	32	2	9248
Conv2DTranspose	(64, 64, 32)	3×3	32	2	9248
Conv2D (DecoderOutput)	(64, 64, 3)	3×3	1	2	289
Total params: 47,105 **Trainable params:** 46,977 **Non-trainable params:** 128					

TABLE I: DAE Architecture.

A. Dataset

We worked with both Fashion Product Images Dataset [24] and Fashion MNIST [25] which are publicly available.
For the analysis, the 'Fashion Product Images Dataset' contains 36 subcategories and 44k products images. We splited it into 80% and 20% for training and testing purposes.Fig:2

Dataset	Train	Test
Fashion Products image Datasets	40K	4K
Fashion MNIST Dataset	40K	10K

TABLE II: Test Train Split.

Fig. 2: Fashion Product Images Dataset sample

And from Fashion MNIST 40000 images have been used for training and another 10000 were used for testing.Fig:3 The resolution of each sample image was reduced into 64Ã64Ã1.Also some preprocessing was done like reshaping, into grayscale, normalization, adding noise etc.

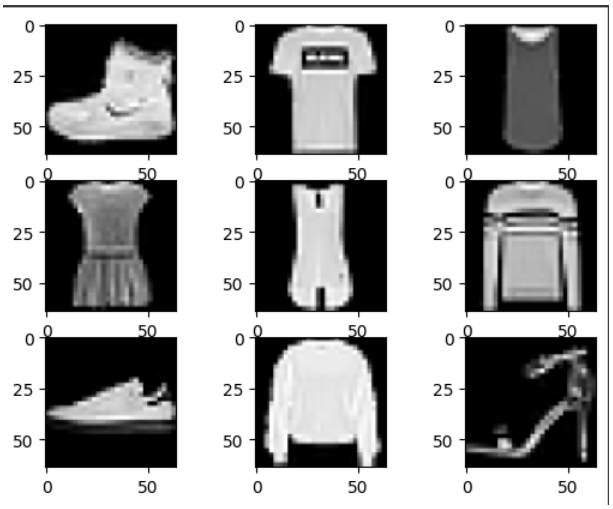

Fig. 3: Fashion MNIST Dataset sample

B. Feature extraction

Feature extraction occurs in convolutional layers of a CNN. These layers learn pixel relationships to capture shape, color, and texture details. Input image data $(h \times w \times d)$ undergoes convolution with kernels $(fh \times fw \times d)$ to produce feature maps. Each kernel detects specific patterns, enhancing the understanding of visual attributes. Activation functions introduce non-linearity, and stacking layers capture hierarchical features. Convolution enables CNNs to uncover essential image characteristics for tasks like object recognition. Then batch normalization used to stabilize and accelerate the training of neural networks by normalizing the intermediate activations within each mini-batch during training.

How batch Normalization works using mini-batches:

1) Input: Mini-batch of activations $X = \{x(1), x(2), \hat{a}\check{S}, x(m)\}$ from a layer in the network.Learnable parameters γ (scale) and β (shift) for each feature.

2) Compute Mean and Variance: Compute the mean μ and variance σ^2 of the mini-batch activations:
$$\mu = \frac{1}{m}\Sigma_{i=1}^{m} x^i, \quad \sigma^2 = \frac{1}{m}\Sigma_{i=1}^{m}(x^i - \mu)^2$$

3) Normalize activations: Normalize the activations using the mean and variance
$$x\prime^i = \frac{x^i - \mu}{\sqrt{\sigma^2 + \epsilon}}$$
where ϵ is a small constant added for numerical stability.

4) Scale and Shift: Apply learnable scaling and shifting to the normalized activations:
$$y^i = \gamma.x' + \beta$$

5) Output and Update: The batch-normalized activations $y^{(i)}$ are used as the output of the layer.
The parameters γ and β are learned during training using gradient descent.

Using the Leaky Rectified Linear Unit (Leaky ReLU) activation function is a common choice for introducing non-linearity in neural networks, especially when dealing with multi-labeled datasets like fashion products.

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \sigma x, & \text{otherwise} \end{cases}$$

Here, σ is a small positive constant (usually a small value like 0.01). The inclusion of the σx term for negative inputs ensures that gradients can flow even for negative values, preventing the neuron from becoming dormant.After this to do downsampling operation that reduces the spatial dimensions of the feature maps while retaining important information. Pooling is commonly used after convolutional layers to decrease the computational complexity of the network, enhance translation invariance, and control overfitting. There are different types of pooling layers, with max pooling and average pooling being the most common.(Show Three Fig of polling).

C. k-nearest neighbors:

The k-nearest neighbors (KNN) algorithm is a simple and intuitive machine learning algorithm used for classification and regression tasks [26]. It makes predictions based on the majority class or average of target values of the k nearest data points in the feature space. In this model we used KNN to get k-nearest data points from our latent space. Let X be the training dataset consisting of m samples, each with n features: $X = \{x^i, x^2, ..., x^m\}$, where $x^i = [x_1^i, x_2^i, ..., x_n^i]$.

For classification tasks, let Y be the corresponding class labels for each sample: $Y = [y^1, y^2, ..., y^m]$, where y^i is the class label of x^i.

For regression tasks, let Y be the corresponding target values for each sample: $Y = [y^1, y^2, ..., y^m]$, where y^i is the target value o x^i

Given a new input data point x_{new}, the KNN algorithm proceeds as follows:

1) Distance Calculation:
Calculate the distance between x_{new} and each trainin sample x^i using a distance metric d. We used Euclidea as distance metrics:

$$distance(x_{new}, x^i) = d(x_{new}, x^i)$$

2) Nearest Neighbors Selection:
Select the k training samples with the smallest distances to x_{new} . These k samples are the nearest neighbors.

3) Classification Task:
If the task is classification, determine the majority class among the k nearest neighbors: predicted class=majority($[y^{i1}, y^{i2}, ..., y^{ik}]$)

4) Regression Tasks:
If the task is regression, compute the average target value among the k nearest neighbors
predicted target value = $\frac{1}{k}\Sigma_{j=1}^{k} y^{(ij)}$

The key components of the KNN algorithm are the distance calculation, selection of nearest neighbors, and the prediction process. The algorithm leverages the distances and labels/target values of training samples to make predictions for new input data points.
We calculated euclidean distance for new data point x_{new} with existing x data points and returned k-nearest with out doing any classification/ranking.

IV. Experimental Analysis

We trained our model using fashion product data set and fashion mnist data set separately. And also combined both data sets but the training accuracy was very similar around 91% in training and 89% in the testing phase.
We also trained our model using smaller sized data like 3000 images and also through bigger data around 43000 images and applied 100 epochs for training.

A. Main Results

Our main result is shown in Fig: 4 and 5

Fig. 4: User input

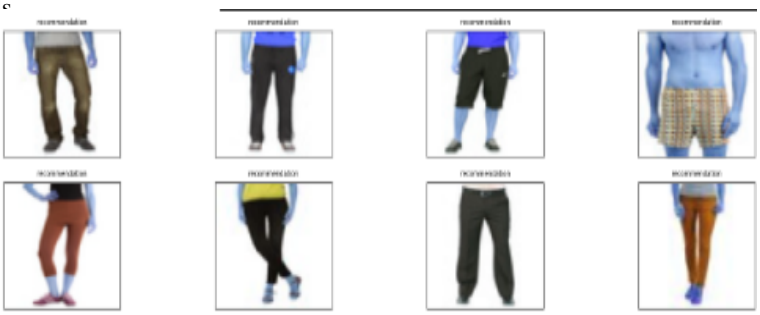

Fig. 5: AutoSimNet recommendation based on user's input

B. Performance Evaluation

We trained our model using fashion product data set and fashion mnist data set separately. And also combined both data sets but the training accuracy was very similar around 91We also trained our model using smaller sized data like 3000 images and also through bigger data around 43000 images and applied 100 epochs for training.Fig:6 7

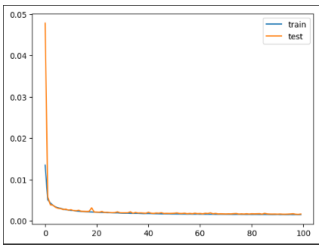

Fig. 6: Learning curve for Fashion Products Datasets

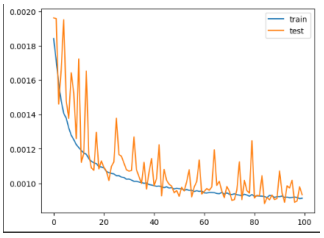

Fig. 7: Learning curve for Fashion MNIST Dataset

Our model learned very quickly while training on fashion product image dataset and learned in an unstable manner while training on a fashion MNIST dataset. But after 100 epochs our learning algorithm achieved almost the same accuracy, so we can say our model performed well on both of this dataset.

V. Discussion

From the existing fashion product recommendation system we can say that, there are not many methods which can be sthatimply used by any smaller organizations. So we decided to build a new method which can be used by those who have little processing power or simply can not afford to possess a heavy complex system. We wanted to create such a system where only images of products are available as a dataset. And also our system will work or recommend similar products when only one image is searched by a user. Therefore We took data sets called Fashion product images and Fashion MNIST [24]. And used only images. To get latent space we feed that data into an Autoencoder. After getting embedded data we passed it into a KNN classifier and recommended those images which are most nearest to our input image.

VI. Conclusion

Visual recommendation systems have gained considerable traction in recent times, offering valuable insights into product interactions through images and other visual content. However, it's crucial to recognize that textual-based recommendation systems remain equally important. They extract sentiments and ratings from user reviews, providing essential information about product perception. In this paper, we propose a deep learning-based model for visual recommendation, leveraging AutoSimNet as a similarity metric. This approach holds promise for improving visual recommendation accuracy and relevance. Looking ahead, we envision combining the strengths of AutoSimNet with textual recommendation systems to create a comprehensive recommendation framework. This integration will consider both visual and textual aspects of user profiles, leading to more personalized and informative product recommendations.

References

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-based systems*, vol. 46, pp. 109–132, 2013.

[2] G. Häubl and K. B. Murray, "" double agents": Assessing the role of electronic product recommendation systems," 2013.

[3] I. MacKenzie, C. Meyer, and S. Noble, "How retailers can keep up with consumers," *McKinsey & Company*, vol. 18, no. 1, 2013.

[4] S. Chhabra, "Netflix says 80 percent of watched content is based on algorithmic recommendations," *Mobile Syrup, August*, vol. 22, 2017.

[5] H. Hwangbo, Y. S. Kim, and K. J. Cha, "Recommendation system development for fashion retail e-commerce," *Electronic Commerce Research and Applications*, vol. 28, pp. 94–101, 2018.

[6] S. Bell and K. Bala, "Learning visual similarity for product design with convolutional neural networks," *ACM transactions on graphics (TOG)*, vol. 34, no. 4, pp. 1–10, 2015.

[7] C. Lynch, K. Aryafar, and J. Attenberg, "Images don't lie: Transferring deep visual semantic features to large-scale multimodal learning to rank," pp. 541–548, 2016.

[8] S. Gu, X. Liu, L. Cai, and J. Shen, "Fashion coordinates recommendation based on user behavior and visual clothing style," pp. 185–189, 2017.

[9] G. Askalidis and E. C. Malthouse, "The value of online customer reviews," pp. 155–158, 2016.

[10] Y. H. Cho, J. K. Kim, and S. H. Kim, "A personalized recommender system based on web usage mining and decision tree induction," *Expert systems with Applications*, vol. 23, no. 3, pp. 329–342, 2002.

[11] H. Wang, X. Shi, and D.-Y. Yeung, "Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[12] M.-F. Balcan, A. Blum, and S. Vempala, "Efficient representations for lifelong learning and autoencoding," pp. 191–210, 2015.

[13] T. Iwata, S. Watanabe, and H. Sawada, "Fashion coordinates recommender system using photographs from fashion magazines," 2011.

[14] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," pp. 173–182, 2017.

[15] R. He and J. McAuley, "Vbpr: visual bayesian personalized ranking from implicit feedback," vol. 30, no. 1, 2016.

[16] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM computing surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.

[17] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[18] R. J. Mooney and L. Roy, "Content-based book recommending using learning for text categorization," pp. 195–204, 2000.

[19] B. Lao and K. Jagadeesh, "Convolutional neural networks for fashion classification and object detection," *CCCV 2015 Comput. Vis*, vol. 546, pp. 120–129, 2015.

[20] H. Tuinhof, C. Pirker, and M. Haltmeier, "Image-based fashion product recommendation with deep learning," pp. 472–481, 2019.

[21] H. Wang, X. Shi, and D.-Y. Yeung, "Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[22] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[23] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, and D.-R. Liou, "Autoencoder for words," *Neurocomputing*, vol. 139, pp. 84–96, 2014.

[24] P. Aggarwal, "Fashion product images," 2022. [Online]. Available: https://www.kaggle.com/datasets/paramaggarwal/fashion-product-images-dataset

[25] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[26] K. Taunk, S. De, S. Verma, and A. Swetapadma, "A brief review of nearest neighbor algorithm for learning and classification," in *2019 international conference on intelligent computing and control systems (ICCS)*. IEEE, 2019, pp. 1255–1260.