CHAPTER 8

Visual similarity-based fashion recommendation system

Betul Ay and Galip Aydin
Firat University Computer Engineering Department, Elazig, Turkey

8.1 Introduction

Visual similarity search systems have become one of the most popular application areas of image retrieval systems in recent years. Content-based image retrieval (CBIR) [1] aims to search images based on their contents such as shapes, objects, colors, local geometry, or texture rather than the metadata associated with the image file such as file names, descriptions, or keywords. Image retrieval systems have been used in a large array of application areas such as search engines, personalized recommendation systems, art galleries management, retail systems, fashion design, and more commonly in e-commerce applications [2].

For any CBIR system, there are two major steps: finding the most important features of each image so that images can be described as feature vectors and calculating distances between images for similarity. Therefore, the success of a CBIR system relies heavily on the quality of the vector feature representation. The task of extracting high-quality, accurate, and efficient vector feature representation for each image is challenging due to the fact that images might have a wide variety of different properties such as content, size, resolution, etc. Also labeling large amounts of data is another major issue. Also supervised trainings with sufficient annotated data might limit the generalizability of the feature representations to novel classes. Recently, semisupervised and unsupervised learning approaches have gained popularity to overcome these difficulties.

One of the most interesting applications of deep learning in recent years is creating feature representations of images as vectors. CNNs have widely been used for this purpose and show great promise [3–6]. Another interesting approach for creating vector representations is generative adversarial networks (GANs) [7]. GANs are being utilized in semisupervised learning due to the fact that they can learn deep image representations from unlabeled data. GANs are created using two models: a generative model G and a discriminator model D. The generative network creates samples while the discriminative network tries to distinguish the generated samples from true data.

GANs are conceptually considered as a form of unsupervised learning because no labeled data is needed. In recent years, GANs have been one of the most popular fields

Generative Adversarial Networks for Image-to-Image Translation
https://doi.org/10.1016/B978-0-12-823519-5.00023-3
Copyright © 2021 Elsevier Inc.
All rights reserved.

of research due to their ability to learn high-dimensional and complex data distributions by taking advantage of the use of unlabeled data for model training. Furthermore, GANs can be leveraged to build powerful models in any domains including images, speech, and text.

Although GANs have been created for unsupervised learning, they have proven to be successful in semisupervised and reinforcement learning as well. GANs have successfully been used in various applications such as generating visually realistic images and style transfer however their effectiveness is not limited to these scenarios. Image-to-image translation (CycleGAN [8]), creating high-resolution images from low-resolution samples (SRGAN [9]), image generation from text (StackGAN [10]), learning to discover relationships between different domains such as fashion items (DiscoGAN [11]), transferring facial makeup from a reference image (Beautygan [12]), and other applications reviewed in Ref. [13] are some examples.

Extracting deep features from images is another area of use for GANs. The idea is that since GANs can generate realistic images, vector representations of these images can also be used to describe a given image. And since the discriminator pushes the generator to generate more realistic images with enough training, the system produces higher quality representations in comparison to CNNs such as VGG. For instance, Hou et al. [14] utilize GANs for extracting features from images. They utilize a pretrained CNN namely 19-layer VGGNet [15] network which was trained on ImageNet for feature extraction. The proposed system contains generator, VGGNet, and discriminator networks. The generator network does not feed the real and fake images directly to the discriminator, unlike traditional GANs. Instead, the real and generated fake images are first preprocessed using VGGNet and corresponding features are extracted. The extracted features are subsequently fed to the discriminative network. The authors evaluated the results via a web interface created for human evaluators and they report that the resulting face images cannot be distinguished from real face images easily and the proposed model generates more realistic images compared to DCGAN [16] and DFC-VAE [17].

Since GANs are able to provide us with accurate image representations as vectors, they can be employed in similar product search for e-commerce applications. Worldwide growth of e-commerce economy has resulted in many innovative solutions including recommender systems to be deployed. Traditionally recommendation systems make use of past customer interactions, clicks, and purchase history to recommend new and related products. Content-based, collaborative, or hybrid recommender systems create recommendations based on recorded customer behaviors and similar decisions but ignore the product image content. A new type of recommendation system being emerged in e-commerce sites is visual similarity recommendation which creates a list of visually similar items to the query image.

In this chapter, we present a visual recommendation system for e-commerce sites which utilizes GANs for image feature vector generation and a vector similarity search

library for fast and accurate querying similarity. The proposed GAN is trained on a large-scale shoe image dataset of 156,896 images. We also compare the precision and time performance of the proposed GAN with existing pretrained deep learning models on a standard fashion benchmark dataset, UT-Zap50K [18]. Since the proposed system does not require annotated image dataset, it is easy to extend for other types of fashion items other than shoes. We also prove this feature by extending the model with handbag images and conduct performance tests as well. The system as a whole presents a deep learning-based similar image recommendation solution. We also provide comparisons for several different GAN architectures for shoe similarity recommendation.

The rest of the chapter is organized as follows: In Section 8.2 we briefly discuss related literature and present background on GANs and CNNs. Section 8.3 presents the proposed fashion recommendation system architecture. Experimental results are discussed in Section 8.4 and conclusions are presented in Section 8.5.

8.2 Related works

The major research problem in this study is finding accurate vector representations for image features. Fast and accurate feature extraction from images allows us to build a robust and effective visual similarity recommendation system. Traditionally machine learning is used to recommend items for e-commerce customers [19–23]. However, in fashion domain image retrieval is subtle and subjective due to the fact that humans tend to have very different opinions on fashion items. Therefore, CBIR is an active area of research for e-commerce [24] and traditional recommender systems can be extended with visual similarity recommendations.

In Ref. [25] the authors presented an architecture for retrieving most similar images to the query image. AlexNet [26] and VGG-16 pretrained networks are used to extract local and deep features from the activation of the intermediate layers. Representations from the fc6 and fc8 layers are used as feature vectors. Hamming distance is used as the similarity metric. To evaluate the efficiency of the system 2399 women's fashion images obtained from Pinterest are collected and manually labeled into nine categories.

Kiapour et al. [27] proposed an architecture for street to shop image retrieval which aims to find similar clothing items to a given real world in an online shop. In Ref. [28] the authors have proposed a solution for cross-domain fashion product retrieval by retrieving similar clothing items from online shopping images. Shankar et al. [29] proposed a visual search and recommendation system for e-commerce. Similarly, they use CNN for generating image feature vectors for each fashion product. Images from the Fashionista dataset [30] and Flipkart catalog images are labeled to create a large annotated dataset.

This section introduces a theoretical overview of GANs [13]. We define the difference between Vanilla GAN and InfoGAN architectures depicted in Fig. 8.1 and highlight the strengths of the adversarial training process chosen for our recommendation task.

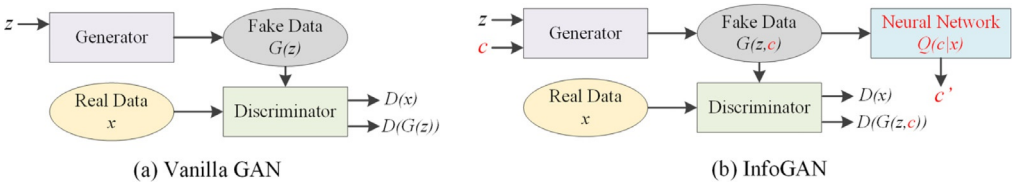

Fig. 8.1 Overview of GAN and InfoGAN architectures

Lastly, we provide an intuitive overview of state-of-the-art architectures based on CNNs.

8.2.1 Vanilla GAN

GAN architecture is comprised of two distinct networks: generator and discriminator. These networks are trained simultaneously and perform adversarial training by playing a two-player minimax game. The generator generates new data samples, while the discriminator decides whether each sample it receives belongs to the training dataset. For an image generation task, the generator network receives a random vector z which is sampled from a known distribution and generates a new fake image. Fake data generated from generator network and real data taken from the real dataset are fed into the discriminator. The discriminator takes account of all the data fed into it and returns a probability of whether an image is real or fake. More formally, this learning process is given in the following steps:

- The goal of generator G is to build mapping from a prior noise distribution where random noise $z \in \mathbb{R}^Z$, to a data space referred to as fake data $G(z)$.
- The goal of discriminator D is to estimate the probability of a sample coming from real data $x = \{x^1, \ldots, x^N\}$ and fake data $G(z)$, being real $D(x)$ or fake $D(G(z))$.
- The value function represents a two-player minimax game that tries to maximize its value with respect to D and minimize its value with respect to G, which is expressed by the following equation:
  $\min_G \max_D V_I(D, G) = \mathbb{E}_{x \sim P_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim P_{\text{noise}}}[\log(1 - D(G(z)))]$.
  Here, P_{data} and P_{noise} indicate real data distribution and noise distribution, respectively.
- While G captures the data distribution in the training set and tries to fool the D with minimization of $\mathbb{E}_{z \sim P_{\text{noise}}}[\log(1 - D(G(z)))]$ term, D network performs a binary classifier with the maximization of the both $\mathbb{E}_{x \sim P_{\text{data}}}[\log D(x)]$ and $\mathbb{E}_{z \sim P_{\text{noise}}}[\log(1 - D(G(z)))]$.

8.2.2 InfoGAN

Information maximizing generative adversarial networks (InfoGAN) described in Ref. [31] controls the different attributes of the generated images, unlike the other vanilla GAN

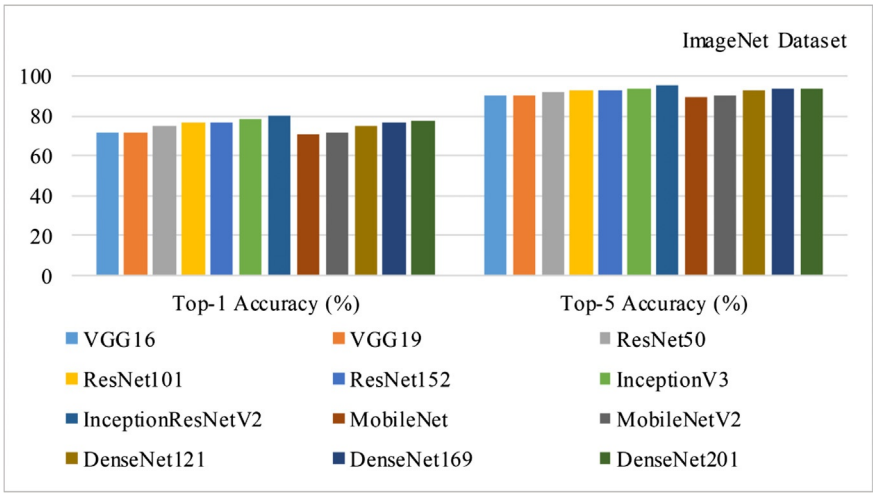

Fig. 8.2 Performance comparison of the state-of-the-art deeper CNNs.

architectures that control the generated images a little or not. The InfoGAN, an information-theoretic extension of GANs, uses information theory concepts so that the noise term is transformed into latent code, which provides systemic and predictable control on the output. It learns how to decompose the input noise vector into two parts; a source of incompressible noise z and a latent code c. It is trained to maximize the mutual information between c and the output of generator (generated image) $G(z, c)$. Fig. 8.1B depicts the architecture of InfoGAN. Information-regulated min-max objective of InfoGAN is formulated as follows by adding a constant regularization term with a hyperparameter λ:

$$\min_G \max_D V_I(D, G) = V(D, G) - \lambda I(c; G(z, c))$$

Here, $I(c; G(z, c))$ is the mutual information between c and $G(z, c)$. While Vanilla GAN formulation (1) uses a single unstructured noise vector z, the generator of InfoGAN takes the concatenated vector (z, c) where c represents structured semantic features of the data distribution [31]. For a given set of structured latent variables $c_1, c_2, \ldots, c_L$, the latent code c denotes the concatenation of all latent variables c_i, which calculated as $\prod_{i=1}^{L} P(c_i)$. It also uses a neural network $Q(c|x)$ that shares the same network structure with the discriminator D (except the last layer), by adding to the Vanilla GAN a negligible computation cost. The InfoGAN uses variational information maximization technique named as lower bounding mutual information, which reduces computationally complex of the mutual information calculation, to maximize the mutual information. The final objective function of InfoGAN with a variational lower bound $L_I(G, Q)$ of the mutual information under the condition $L_I(G, Q) \leq I(c; G(z, c))$ is defined as follows:

$$\min_{G,Q}\max_{D} V_{\text{InfoGAN}}(D, G, Q) = V(D, G) - \lambda L_I(G, Q)$$

8.2.3 CNN-based architectures

This section gives a review of CNN and dives deeper to explore the top CNN architectures which have proven themselves at visual tasks including image classification, object detection, and semantic segmentation. CNN emerged in the 1990s when Yann LeCun et al. [32] put forward new neural network architecture for classification of handwritten digits. Although the first CNN, known as LeNet, recognized digits of zip codes effectively, it could not cope with more difficult and complex data. Nevertheless, the work of these and many other researchers led to the development of larger and deeper CNNs. With the ImageNet visual recognition competition, the latest known CNN architectures have emerged. In 2012, the best invention was AlexNet architecture, developed by Alex Krizhevsky and his colleagues [26] at the University of Toronto. The first popular use of deep learning in computer vision began with the AlexNet architecture, which has 60 million parameters (over 1000 times higher than that of LeNet [33]), five convolutional layers and three fully connected layers.

Deeper CNNs, which appeared first on the issue of ImageNet classification, have been more efficient in coming up with solutions to classification problems. Accuracy comparison on ImageNet of most popular CNN architectures, also used in this study for quantitative comparison, are depicted in Fig. 8.2. These architectures are summarized below:

- *VGG:* This architecture was developed by Simonyan et al. in 2014 [15], based on the notion that deeper networks are stronger networks. The overall number of trainable parameters (around 140 million) is over 2.3 times higher than that of AlexNet. On the other hand, smaller filters have been used when compared with AlexNet. This architecture uses fixed 3×3 kernel filters in all convolution layers. Two versions of this architecture are available, VGG16 and VGG19, each with the layers 16 and 19. Both of the networks have been built with five blocks followed by a max-pooling layer and the blocks contain sequential convolutional layers.
- *Inception:* The architecture, known as GoogLeNet, was developed by Google researchers [34]. Although VGG architecture has better accuracy performance than AlexNet, it needs to use too much memory due to the number of parameters. On the other hand, Inception has been performed higher accuracy than AlexNet and VGG16 with fewer trainable parameters (about 5 million for InceptionV1 version). It has been built with inception blocks that contain convolutional layers with variable-sized filters of 1×1, 3×3, and 5×5. The architecture, continuously improving upon, has also different versions (InceptionV1, InceptionV2, InceptionV3, so on).
- *ResNet:* Researchers have noticed in the previous architectures that adding layers to deep architectures, while performance up to one point increased, a drop declined rapidly after one point. This problem, known as the vanishing gradient, arose during

network training with back propagation. More briefly, as the number of layer increases, gradient values decrease and approach zero. ResNet [35] solved the vanishing gradient problem by introducing a shortcut connection. The architecture was built with residual blocks consisting of convolutional layers and the shortcut connection that also named as skip connection that connects the first layer input to the last layer output. The shortcut connection also named as skip connection because the network with this connection can skip some of the convolutional layers. Different residual networks with different layers of 18, 34, 50, 101, and 152 have been proposed. The common feature of all ResNet architectures is that they have 7×7 convolutional layer followed by 3×3 max-pooling layer before residual blocks, and average pooling after the blocks (before fully connected output layer).

- *DenseNet:* Like ResNet, DenseNet [36] also focused on solving the vanishing gradient problem with fewer parameters. DenseNet architecture inspired from ResNet was constructed of dense blocks consisting of convolutional layers. Unlike ResNet, there are direct connections from the one convolutional layer to all sublayers. An input of any layer is the concatenated of the feature maps generated by all preceding layers. The architecture has been built with multiple dense blocks and it uses the same layers of ResNet before and after dense blocks. There are various versions of the network with different number of dense blocks such as DenseNet121, DenseNet169, DenseNet201, and DenseNet264.
- *MobileNets:* The main goal of this architecture [37] is to create light-weight and low-latency models, which can be used on limited-memory or resource-limited devices. The architecture (MobileNetV2) was built with inverted residual blocks and linear bottlenecks, where the input and output of the residual blocks are the bottlenecks layers [38]. When compared with previous architectures trained on ImageNet dataset, MobileNet have higher inference time and smaller model size, but it has worse classification performance (see in Fig. 8.2). This performance is acceptable given its ability to near real-time work on mobile devices. There are different versions such as MobileNet V1, V2, and V3.

The abovementioned architectures, which also have pretrained models, are still very popular today and are often used as benchmarks to compare new proposed architectures or make sure a new dataset is reliable.

8.3 Fashion recommendation system

The system architecture depicted in Fig. 8.3 consists of three major modules:

1. A deep neural network model for generating effective image representations or vectors
2. Feature vector database for querying image similarity
3. Web interface for interacting with the system

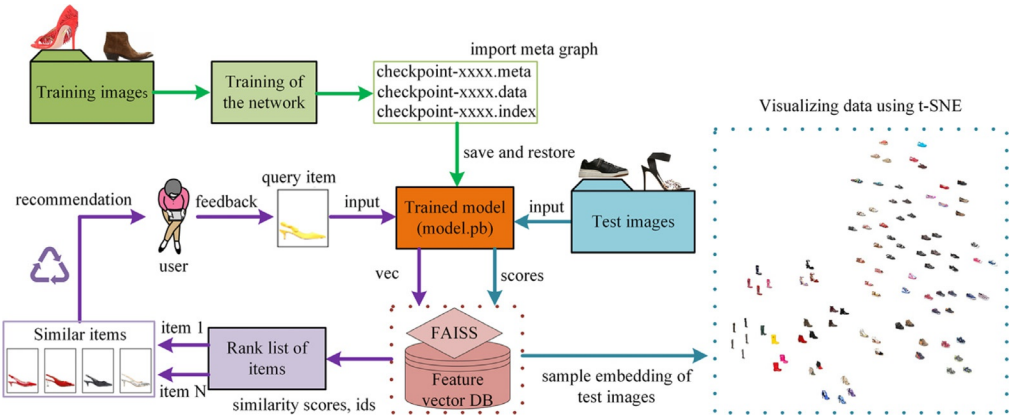

Fig. 8.3 Overview of fashion recommendation system.

Arguably, the most important part of any image retrieval system is a feature extraction module since the accuracy of the results depends on the quality of the extracted features. Feature extraction is basically the process of dimensionality reduction in which a given input is converted into more manageable values so that redundant information in the input are ignored and computational power required to process a large number of inputs is decreased. The quality end effectiveness of the extracted features leads to more successful learning and generalization steps. Extracted features from a given input are generally represented as feature vectors in which only the important or selected portions of the input are preserved. Almost all machine learning tasks require some form of feature extraction to deal with large datasets and hence several algorithms and approaches have been developed over the years. Sometimes experts use domain knowledge to extract important features from the data which is called feature engineering while several algorithms such as PCA (principal component analysis), Autoencoders, or LSA (latent semantic analysis) are used to extract important features. Traditionally, in image processing edge, corner or blob detection algorithms are used to extract features.

However, deep neural networks have lately paved the way for automatic feature extraction approaches. DNNs such as CNNs have proved to capture the significant properties of the images and create vector representations for later use. With little or no preprocessing, the images are fed into a pretrained DNN and vector representation or embedding of the image is obtained. Several studies show that the embedding obtained from DNNs can capture semantic and inherent features in the images and thus similar images are represented with closer vectors.

Although deep learning models such as CNNs give successful results for the general image similarity problem, it may not be possible to get sufficient quality results within the fashion domain. For instance, there are only a few general shoe types (bots, sneakers, oxfords, etc.) and products in one of these categories resemble each other. Therefore,

image similarity approaches in such domains require finer grain resolutions for successful results. In this study, we use GANs for generating vector representations of images.

Image similarity task is the process of retrieving similar images to the queried product image. In our case where we try to identify similar shoe images, similarity can be looked at three major properties: color, shape, and texture. The success of the similarity query must account for all three dimensions; hence we experimented with several GAN architectures and evaluated the results according to these three dimensions.

The second module is the vector database which stores the vector representations of images and returns the similar image ids for similarity queries.

With the widespread use of artificial intelligence models, the need to effectively query the vector representations of both text and image data has emerged. In the recommender systems, for example, each product or user is represented as a vector and for each vector, there needs to be a list of best recommendations generated. However, it can be quite expensive to generate such lists where the number of data is very large because traditional databases and algorithms are not suitable for storing and querying for similar vectors among hundreds of thousands or even millions of vectors. In recent years several similarity search libraries emerged such as FAISS, Annoy, and NMSLib. Most of these libraries generate a list of approximate nearest neighbors for a given query and employ several different types of indexes to perform this task effectively.

FAISS (Facebook AI Similarity Search) [39] is a library developed by Facebook AI Research group which is used for efficient searching of dense vectors or document embeddings. It contains many useful algorithms for searching arbitrary sizes of vectors. We employ FAISS in our architecture as a similarity search engine. FAISS implements some of the search algorithms on GPU and is extremely scalable in comparison with the traditional SQL database engines.

FAISS is best utilized with documents are represented as vectors and are identified by an integer id. Image embeddings or representation vectors are extracted using the trained model and are inserted into the FAISS index. FAISS can compare the vectors using L2 (Euclidean) distances, dot products or cosine similarity. When a query image is presented the system creates the vector embedding of the image and queries the FAISS index for similar images. FAISS returns the lowest L2 distances (or the highest dot product) with the query vector [39].

In our system, we first build a FAISS index with XXX shoe image vectors. When a query image is presented, the system first generates a vector of the image using the GAN model and queries the FAISS index for closest vectors. The FAISS library returns a list of image ids of which the L2 distances are smallest to the query vector.

The third and last module is the web interface which is used to interact with the system. We demonstrate selected shoe images and similar images.

Successful deep learning models require large amount of data for training. It is imperative to provide a sufficient amount of data so that after enough training the DNN can capture the essence of the domain and visually understand what is in it and what is not.

For a request image, the image is passed into the trained model and feature vector of the image is extracted. Similarity score is computed across all of the images stored in feature vector DB. The best similar images that have the lowest similarity score are ranked for the recommendation. Our goal is to achieve the optimal model as the inference time has to be fast.

8.3.1 Deep network architectures

We make an effort to build a recommender system based on visual similarity in this chapter. Firstly, we explore the best model, which learns the best visual features of fashion items. The best model in this study refers to the optimum neural network, which gives high accuracy with fast inference time. Since the various visual features such as colors, edges, corners, and other different patterns in the model are learned during the training process, deep learning is also referred to as feature learning. We present our feature learning experiments of following neural networks under this section.

8.3.1.1 Proposed network

The proposed architecture of discriminator and generator networks inspired by InfoGAN [31] is defined in Table 8.1.

The discriminator model D takes an image with tree color channels and 207×207 pixel in size and outputs a binary prediction as fake or real. Instead of binary prediction, we use the discriminator to extract a feature vector with a size of $1 \times 12{,}544$ after saving the trained discriminator model. Generator model G input is a concatenated 108-dimensional vector consisting of noise variable (100) and latent code (8) representing class information. We conduct unsupervised learning which we use no labeled data. We assume that our data consists essentially of eight different classes (heels, sandals, sports, boots, high boots, loafers, slippers, and flats), so the latent code value is set to eight. We change the latent code value to nine for adding handbag to further extend. We use batch normalization [40] in the models to stabilize the training. We also apply a regularization layer (dropout [41]) after all convolutional layers to solve the overfitting problem and memorization limitation. For D (Discriminator), all convolution layers use Leaky ReLU (LReLU) and the output layer using sigmoid activation is used to get the prediction score of the images over two classes as real (class = 1) and fake (class = 0). For G (generator), all transposed convolutional layers use ReLU and tanh activation function is used in the last layer, which is described in Ref. [16]. Similar to the InfoGAN, D and Q share the same network structure using convolutional layers except for the fully connected output layers. At the output layer, Q uses tanh activation function.

The discriminator and generator loss curves are depicted in Fig. 8.4. From the patterns in the loss curves, it can be seen that both discriminator loss and generator loss decrease up to about 2500th iteration. After about this iteration point, generator loss is increasing rapidly and discriminator losses are dropping, which mean that discriminator is getting too strong to distinguish the real and fake samples and generator is not able to generate better

Table 8.1 The networks used for training the shoe and handbag dataset.

Discriminator Model ***D***/Recognition Network ***Q***	Generator Model ***G***
Input 207x207 Color image	Input $\in \mathbb{R}^{108}$
3x3 conv2d. 16 LReLU. stride 2.	FC. 4x6x256
Dropout (.5)	3x3 conv2d_transpose. 128 ReLU. stride 1. batchnorm
3x3 conv2d. 32 LReLU. stride 1. batchnorm	Dropout (.6)
Dropout (.5)	3x3 conv2d_transpose. 64 ReLU. stride 1. batchnorm
3x3 conv2d. 64 LReLU. stride 2. batchnorm	Dropout (.6)
Dropout (.5)	3x3 conv2d_transpose. 32 ReLU. stride 1. batchnorm
3x3 conv2d. 128 LReLU. stride 1. Batchnorm	Dropout(.6)
Dropout(.5)	3x3 conv2d_transpose. 16 ReLU. stride 1. batchnorm
3x3 conv2d. 256 LReLU. stride 2. Batchnorm	Dropout(.6)
Dropout(.5)	3x3 conv2d_transpose. 16 ReLU. stride 1. batchnorm
3x3 conv2d. 16 LReLU. stride 1. batchnorm	Dropout(.6)
FC. 1 sigmoid for D (output layer for D)	3x3 conv2d_transpose. 3 Tanh. stride 1.
FC. 8 Tanh for Q (output layer for Q)	Dropout(.6)

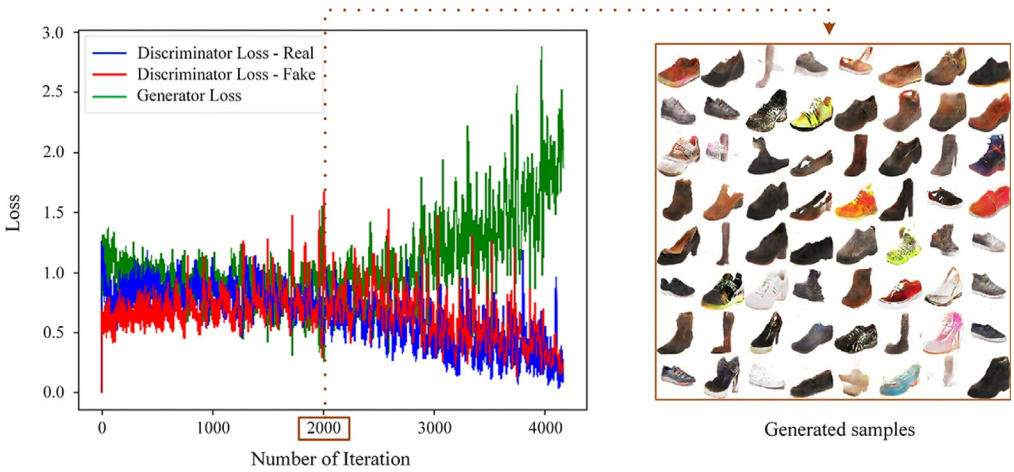

Fig. 8.4 Training results of the proposed network.

samples. We accept the 2000–2500 intervals are the ideal checkpoints for our network and the generator samples at this iterations confirm that the model has learned the shoe features well. We can observe from the Fig. 8.4 that the shoe samples generated by the proposed network have some basic shoe features such as color and class patterns including heels, sandals, sports, boots, high boots, loafers, slippers, and flats. The model also overcomes the image quality and diversity barriers, which the GAN models often suffer the lack of the diversity of generated images.

8.3.1.2 State-of-the-art CNNs

In this study, we retrain 12 pretrained models trained for the ImageNet classification (showed in Fig. 8.2) to use as feature extractor by removing the last (dense) layer. We remove the dense layer that has 1000 labels (classes) and add dropout and a new dense layer consisting of one label that represents shoe class. We leverage learning deep features from shoe images and extract the feature vectors of these new models trained on general shoe domain instead of the shoe classification task. In a nutshell, we transfer the pretrained features into shoe domain by using the power of transfer learning. Feature vectors containing the new visual features belong to shoe domain are used for computing a distance metric between similar shoe items. The process of extraction feature vectors for a given input is called inference. While the light-weight models have low inference time, the heavy models with large number of parameters are expensive for inference. The inference time has to be fast because the high inference time leads to negative user experience. Inference time or the response speed of the trained network is as important as the accuracy.

8.4 Experiments and results

To measure the performance of the aforementioned models and the overall architecture we have conducted several experiments. AI models created and employed in this study are developed using Tensorflow framework. We use a public standard benchmark dataset for measuring the performance of the models.

8.4.1 Experimental setup

The training experiments and performance tests have been conducted on a server that has 24-core Intel Xeon E5-2628L CPU, 256 GB RAM which runs Ubuntu Server 16.04 OS. 8 NVidia GTX 1080-Ti GPUs on the server have been used for training the models. The baseline framework is TensorFlow for all model experiments. The shoe dataset used for this study is collected from Turkish e-commerce sites: https://www.flo.com.tr, https://www.trendyol.com, and https://www.n11.com. We scraped the handbag data from various web sites: https://www.flo.com.tr, https://www.amazon.com, https://www.hepsiburada.com.tr, https://www.boyner.com.tr, https://www.trendyol.com,

https://www.ayakkabidunyasi.com.tr, https://www.morhipo.com, and https://www.n11.com. The overall training dataset consists of 156,896 shoe images and 130,540 handbag images. We conducted the performance tests of all models on 10,000 randomly selected shoe images from UT-Zap50K benchmark dataset.

8.4.2 Comparative results

Fig. 8.5 depicts the test results of all models on UT-Zap50K benchmark dataset. For each network architecture, Fig. 8.5A shows the precision results and Fig. 8.5B shows the inference times. It is known that the performance results of unsupervised learning hard to evaluate, hence no universally agreed performance metrics are available for visual recommendation applications. However, the return of similarity results from irrelevant classes for a query product of a particular class indicates that the model is working poorly. For example, when the customer clicks on a product belonging to the heel class, it is expected that similar products from the heel class will be recommended. Therefore, we firstly compare the models with the standard precision metric which is formulated as follows:

$$\text{Precision} = \frac{\#\text{of relevant items retrieved}}{\#\text{of retrieved items}}$$

Precision values for each model are calculated using eight classes in the UT-Zap50K dataset. For a given query image Zap50K class information is used as the ground truth. If the retrieved image is of the same class then the result is counted as relevant and marked irrelevant if otherwise.

Performance results shown in Fig. 8.5B shows that inference time increases linearly with the depth of the neural network. Large number of parameters (weights) makes the networks memory-inefficient and computationally expensive. Inference time for our proposed model is around 0.04 s which is significantly shorter than other pretrained models. Our model also provides higher precision rates. The proposed model performs the best in terms of precision and inference time among all models tested in this study.

Fig. 8.6 shows the results of all models tested in this study for a sample shoe image. It can be observed that all versions of DenseNet model have returned similarity results in irrelevant classes for a query image given in the type of sneakers.

Sample visual recommendation results with similarity scores have been displayed in Fig. 8.7 (woman shoes) and Fig. 8.8 (woman handbags).

8.4.3 Web interface for visual inspection

Success of image retrieval systems is hard to measure due to the fact that the concept of image similarity is highly subjective. We have created a web interface for visual inspection of similarity results as another performance evaluation. We have asked human

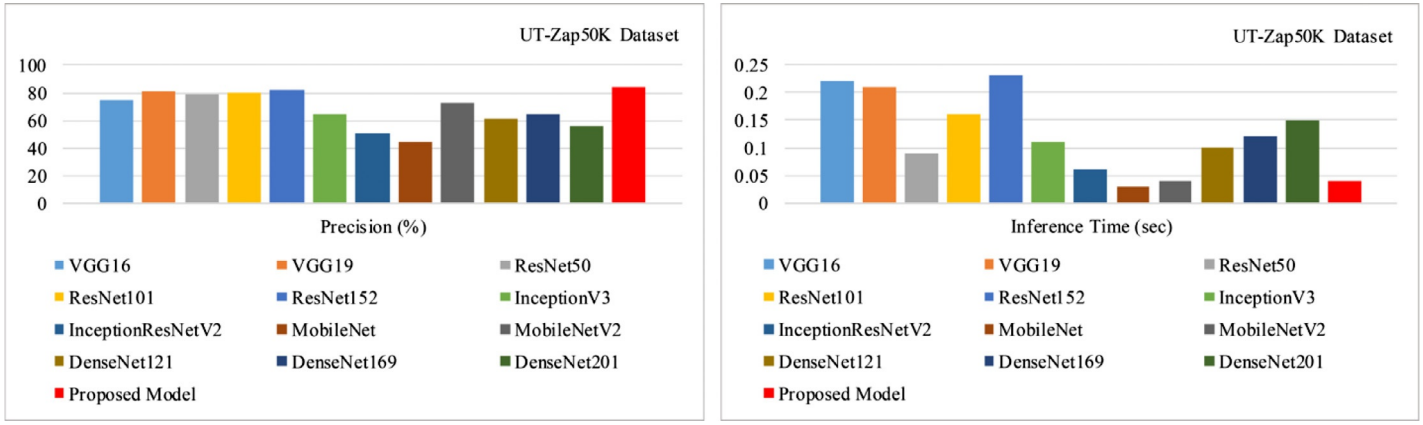

Fig. 8.5 Performance comparison for the models used in this study: (left) precision results and (right) inference time per model.

Fig. 8.6 Visual similarity search results for the proposed model and other pretrained models.

Fig. 8.7 Sample visual similarity results of the proposed model on unseen shoe images retrieved from beymen.com (Query image and top-5 similar images—the results taken from 7723 indexed images).

Fig. 8.8 Sample visual similarity results of the proposed model on unseen handbag images retrieved from beymen.com (Query image and top-5 similar images—the results taken from 3507 indexed images).

annotators to select the best results among alternative results. Fig. 8.9 shows the annotation interface. The annotator clicks a shoe image on the left and results of different images are shown on the right. Then the annotator selects the rows which he/she thinks contain the most similar images. We use this information to determine which model performs the best in terms of actual human feedback.

8.5 Conclusion and future works

In this chapter, we outlined our work visual similarity-based fashion recommendation system. This chapter is extended from our DeepML-2019 submission [42]. The system consists of a GAN-based image retrieval module and a high-performance image feature search library. We have collected a large set of shoe images from e-commerce to train the GAN and used the model we obtained from this training to create a CBIR system. We also created another set of shoe image from https://www.flo.com.tr which is used to create a web interface to demonstrate the results of our GAN model along with other models we tested in this study. The experimental results show that the proposed model achieved superior performance in terms of precision and query time. The results also show that the system can be used in real-world e-commerce platforms as well.

Based on the findings in this study, we plan to build an end-to-end online fashion recommendation system for e-commerce sites. The system will contain recommendation support for various fashion items such as shoes, clothes, and accessories. We also plan to explore other GAN architectures for generating successful image representation for different fashion categories.

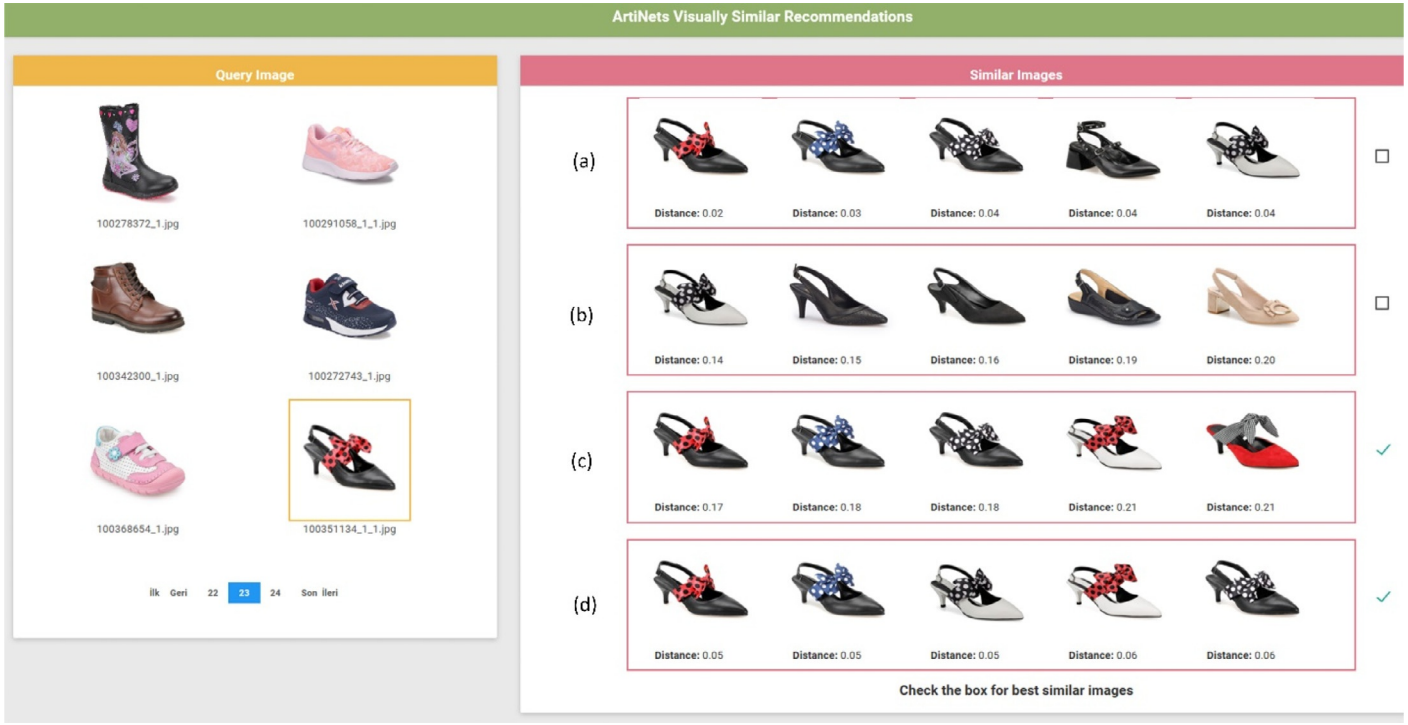

Fig. 8.9 Web interface checked human annotators to selection of the best model: (A) VGG19, (B) MobileNetV2, (C) ResNet152, (D) proposed model (modified InfoGAN).