Reverse Image Search for the Fashion Industry Using Convolutional Neural Networks

2020 IEEE 5th International Conference on Signal and Image Processing (ICSIP) | 978-1-7281-6896-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICSIP49896.2020.9339350

Kudzai Felix Mawoneke, Xin Luo, Youqin Shi
School of Computer Science and Technology
Donghua University
Shanghai, China
e-mail: kfmawo@gmail.com , xluo@dhu.edu.cn, yqshi@dhu.edu.cn

Kenji Kita
Faculty of Engineering
Tokushima University
Tokushima, Japan
e-mail: kita@is.tokushima-u.ac.jp

***Abstract*—A.I. is becoming a hot topic in the rapidly developing digital world of today. Its applications are extensive and ever yet further growing as our understanding of it increases. Reverse image search is another interesting upcoming topic, of which its applications make our lives significantly easier in a number of business sectors and even in our day to day lives. Its implementation is when instead of text, pictures are used as search input, and similar images are returned as the result to help users of a system easier find what they are looking for. In this paper we trained our own Convolutional Neural Network(CNN), and designed an image search engine for use in the fashion industry.**

Keywords- machine learning; reverse image search; distance metrics; fashion; Convolutional Neural Networks(CNNs)

I. INTRODUCTION

With the dawn of online shopping, it is very easy these days to find anything you wish to buy, even from the comfort of your home. You can easily complete a purchase, and have the goods delivered to you. However, even though all these products are readily available at the stores, sometimes, it can be difficult locating them on the websites to initiate the purchases. Part of the problem inherently lies in the manner through which our search mechanisms have conventionally been carried out. Up until now, the most common means we have used to search for goods is one in which we input text into a search box and then the appropriate listings accompanied by pictures of the products are returned to us as output. Notwithstanding, we sometimes inevitably find ourselves in situations where it is difficult to precisely describe the particular item we wish to find in few enough words to fit into a text field search box [1]. This is especially true in the field of fashion. Consider an example of when trying to procure a blazer, or a dress of a particular design. There are so many nuances and subtleties that differentiate that blazer from the plethora of others already in existence. Consider trying to describe these all in a text field search box. You may though, happen to have a similar image to one that you intend to buy, already saved on your phone or laptop. It would be of great convenience, to be able to simply upload this image to the online platform where you are doing your shopping, then have all of the most similar images (in this case, blazers) available on the database of that system, returned to you for you to choose from, with the simple clicks of a few buttons. Another practical example of the need for a feature like this is if one were looking through a fashion magazine or if they saw someone throughout their day wearing something they fancied, but did not know the particular name of the clothing item. Quickly snapping and saving a picture which could then be used to search for similar products on online stores would then help this person procure this product for themselves eventually as well.

This is the field of computer vision, and it has been extensively explored since the early 1990s [2]. It still attracts a great amount of attention today, and recent efforts in research and integration of using image search technology has played a big role in improving the efficiency of online shopping in present day. Early implementations of this particular reverse image search functionality were implemented in websites such as TinEye [3], which is now one of the largest and most popular reverse image search websites presently, with also one of the largest databases of indexed images. Other examples of websites which utilize this capability are Ditto, Snap Fashion, ViSenze, Cortica, Google Image search [4], Bing Image Search [5], Flickr, Yahoo Images, and Getty Images to name a few. Although not all of the aforementioned websites are particularly online stores, their successful implementation of this feature gives it practical validation. A great example of a website that does in effect integrate reverse image searching for online shopping is Taobao [6], the largest Chinese online shopping company, grossing over billions of dollars in profits each year.

Furthermore, with cutting edge technology increasing at an exponential rate and showing no signs of slowing down, high definition photos are becoming commonplace, and companies are in no delay to leverage them. Online stores heavily rely on using images for advertising and showcasing their products to their customers in as much detail as possible. With this rapid increase in the number of existing images on databases and servers, it becomes increasingly difficult and expensive as well to physically annotate every new image that comes to existence. Annotation and tagging are the manual addition of descriptive text and tags to images for retrieval through textual searches [7]. Recent popular social websites, such as Facebook, Instagram etc., make it possible for users to submit tags with the photos that they upload.

This provides a means of indexing, and allows for the photos to be found later in a much easier fashion. Unfortunately, one of the largest shortcomings of this is that plenty of the user submitted tags are biased to the individuals' sentiments, as in an example of a person uploading a picture of a cat, and adding a tag that says 'cute'. Although this tag is relevant to the user submitting it, it is not necessarily helpful to another person who may be browsing the system looking for pictures of cats. Hence from all this we can see, it is with ever increasing importance that effective systems and technologies capable of automatically sifting through and organizing images, using more sophisticated technologies, become readily available as well [8].

II. Related Work

In the technical world of computing, reverse image search has been referred to as Content Based Image Retrieval(CBIR) [9]. This simply means that images are searched for and retrieved, using only the contents of the image, without the assistance of any external labels or metatags. A picture is usually broken down into its defining features, sometimes referred to as feature vectors. The measure of level of feature extraction determines the image content level. The information inherently obtained from an image can be categorized into three levels [8].

- Low level features - Shape, texture, color, spatial information and motion are considered as low level.
- Middle level features - Organization of different types of objects, scenes and roles represent this category.
- High level features - Include objects or scenes with emotional or religious significance impressions, meaning associated with the combination of perceptual visual contents.

Such features are obtained by applying mathematical algorithms to the image, and as such there are a number of choices as to which one may use. Generally speaking, any image search system can be broken down into four important and distinctive components [10], which are;

1) Defining which feature you want to extract from the images, e.g. shape, color, texture etc. This also entails defining the algorithm you will use to do so.
2) Saving and indexing your features – after you have chosen which algorithm to use, you must apply it to all of the images in your dataset, and them index and save them in the appropriate database, as Fig.1 shows.
3) Fig.2 shows the core components of an image search system. Application of a distance metric – Now that the list of features is obtained, we must further apply another mathematical function to determine which features are closest in similarity, and hence this ultimately reveals which images are most similar. Some examples of distance metrics are hamming distance, cosine distance [11], Euclidean distance, k means clustering [3] etc.
4) Returning and displaying the best matches to the user – after the most similar features are determined, the images that they belong to are then retrieved once again and displayed back to the user in an orderly fashion, starting with the most relevant. The accuracy, speed and efficiency of a system highly depend on the types of features and algorithms chosen for its implementation [4].

Figure 1. Feature extraction.

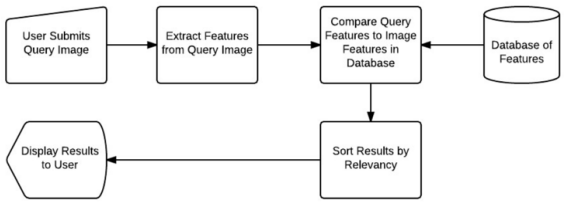

Figure 2. Core components of an image search system.

Most CBIR systems have commonly utilized low level features such as color, texture or shape [12] individually for their functionality, while others make have utilized a combination of some or all of them. Padmashree Desai et al. proposed an image retrieval system using combined approach wavelets and local binary pattern [8], to extract shape, color and texture. They were able to build an image search system which attained an overall high success rate at retrieving images from 10 classes as a benchmark test, with high accuracy. P. S. Hiremath and Jagadeesh Pujari [13] also proposed a method that combined color, texture and shape for an image retrieval system. One of the shortcomings of conventional low level CBIR systems however, is that they fail to pick up on similarities of objects which may look completely different while yet still belonging to the same class. Take for instance tables. Tables may be circular or square, tall or short, and yet all of them are still basically tables. This is true also in the field of fashion, clothing items may look completely different, but essentially still belong to the same class. To be able to detect similarities between these, we need a system more intricate, and thanks to the dawn of artificial intelligence, this becomes possible. Another disadvantage of some conventional CBIR systems was that color histograms did not preserve spatial information [14], which some researchers attempted to address using Color Correlograms [15]. One of the many advantages of Convolutional Neural Networks (CNNs) is that spatial information is preserved [16].

Artificial intelligence is a relatively new area in the field of computing but its popularity is growing at an unprecedented rate due to its power [17], unrivalled ability at automating tasks for humans, and the practicality of its implementation. Convolutional Neural Networks (CNNs) are a subsector of Machine Learning. The very first time an architecture of a neural network was originally proposed, was by Kunihiko Fukushima [16], who discovered that this

structure could preserve patterns, and most importantly 'self-organize', which is the ability to learn without a teacher. The idea of the ConvNet(CNN) was then introduced by Yann Le Cun et al [11].

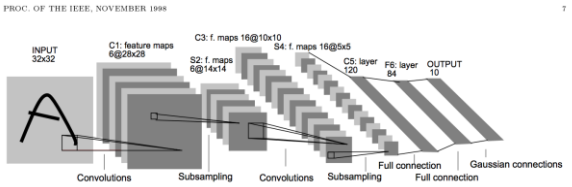

Figure 3. Architecture of LeNet, a Convolutional Neural Network

Fig. 3 shows the first ever CNN called LeNet, which was used to identify and classify handwritten digits, ranging from one to ten. CNNs were architecturally modelled and designed after the human neuron [18] and similarly as well for the manner in which they were designed to function. Presently, they have found their use in simple image classification, object detection, automatic colorization, self-driving cars, face detection [19], image analysis in health etc. In the field of computer vision, the majority of the work that has been done has been for image classification and detection.

Domonkos Varga and Tam´as Szir´anyi proposed a fast content-based image retrieval using Convolutional Neural Network and hash function [7]. In their work, they made use of a hash function as a distance metric, and conducted experiments querying images in comparison with other pre-existing hash algorithms.

III. Our Approach

In our work we designed and trained a CNN, for the sake of obtaining the vector features of the images, using a custom dataset. The general architecture of the model we used is represented by Fig. 4, and we elaborate further on the configurations we used specific to our network as well. In our work, we utilized the ability of CNNs to classify objects i.e. systems which upon being given an image are able to deduce to which class an item in the picture belongs to, e.g. cat, dog, airplane etc., and extended upon it for the purposes of visual image search and retrieval. This is because during training, a CNN learns how to break down an image and automatically learns which features are important to extract. All CNNs can be divided into a number of important building blocks. Here below they are listed and explained in more detail along with the configurations that we used for our network.

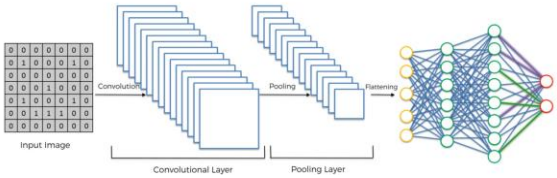

Figure 4. Core components a CNN network.

- Convolutional Layer – a convolution operation is mathematically defined by,

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau \tag{1}$$

and how it is implemented in practice is illustrated in Fig. 5.

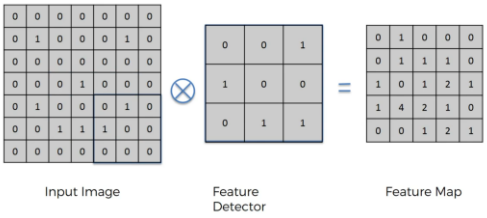

Figure 5. CNN feature detector.

A number of feature detectors are passed over the image, and a set of feature maps obtained as a result. These are the actual image pixels of an image, hence in effect the size of the image is reduced allowing for less computational expense, and while yet still preserving the important information and spatial information of an image. This also allows for the defining features of specific classes to still be detected even if those objects were to appear in different positions within an image, e.g. bottom left corner or top right corner. In our network we used 32, 3x3 feature detectors. When using CNNs, there is no need to manually define the features to look for, this is automatically determined by the network throughout its training.

- Pooling Layer – this layer further reduces the size of the feature maps, further discarding irrelevant information from the image where features are not detected and selecting only the regions with the highest feature occurrences. This also assists in accounting for slight distortions or variances amongst different objects of the same class, i.e. different objects of the same class will have approximately similar pooled feature maps. An illustration of the pooling process is shown in Fig. 6.

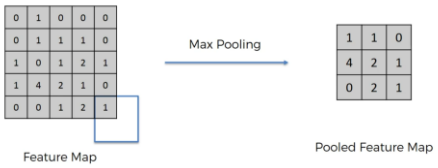

Figure 6. Max Pooling operation.

For the CNN architecture, we used a pooling filter of size 2x2.

- Flattening – after the pooled feature map is obtained from the previous layer, its output is taken, row by row and flattened, i.e. stacked into a single column vector so that it may be fed into the input layer of the ANN, as can be seen from the section with connected circular nodes in Fig. 4.

- Artificial Neural Network (ANN) – made up of individual elements called perceptrons, also sometimes referred to as neurons. Perceptrons are governed by the formula,

$$a^{(1)} = \sigma(Wa^{(0)} + b) \quad (2)$$

where a^1 is the output, σ is the activation function, W is the weight applied to the sum of the inputs coming into the perceptron, $a^{(0)}$, and b is the bias. The collection of these makes the ANN. Networks where all the perceptrons are connected to each other may also be referred to as Deep Neural Networks (DNN). These layers may also be referred to as fully connected layers. The configuration that we used for the network is shown in Fig. 7.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 78, 58, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 39, 29, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 37, 27, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 18, 13, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 11, 64)        36928
_________________________________________________________________
flatten (Flatten)            (None, 11264)             0
_________________________________________________________________
dense (Dense)                (None, 128)               1441920
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 45)                2925
=================================================================
Total params: 1,509,421
Trainable params: 1,509,421
Non-trainable params: 0
```

Figure 7. Our CNN model parameters and architecture.

Saurabh Karsoliya [20] concluded that is very difficult to determine the topology to assign to a network i.e. number of hidden layers and number of neurons, just from the ***number*** of inputs and outputs. He also mentions that increasing the number of hidden layers increases accuracy for any network, but too many layers increase the chances of overfitting. Therefore, for the network, we utilized three dense layers, the first one containing 128 neurons, the second one containing 64, and the final one containing 45 neurons, as the output layer; one of each of the 45 neurons in the final layer to represent each label in the dataset. Optimal parameters are hence truly subject to experimentation, and since the model obtained satisfactory accuracy during training, we proceeded.

The manner in which CNNs are trained can be categorized into unsupervised, supervised, or re-enforcement learning. The method we used falls under supervised learning, since the correct predictions for the dataset are known and available to the network all ***throughout*** training, in which the network checks if it calculated the prediction correctly, if not then the error is sent back throughout the network, updating the weights in a process known as backpropagation. For the error to be known within the network, it must be calculated by means of a loss function. For the network, we chose the categorical cross entropy loss function. A single iteration of this, i.e. information being passed through the network then the error being back propagated, is called an epoch. We were able to attain the high degree of accuracy shown in Figure 9, within six epochs, after which training had to be interrupted to prevent overfitting.

- Output Layer – for the output layer, we used the softmax function, which calculates probabilities at the output and therefore decides what class each image belongs to. It is given by the function below,

$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}} \quad (3)$$

Given sample vector input x and weight vectors {wi}, the predicted probability of y = j, where here y is the detected class.

After a CNN model is successfully trained, the vector features of any image that is fed into the network may be easily attained from the dense layers, by a means of a function that utilizes the model. For training the network, we used a dataset of fashion images called the Fashion Product Images Dataset [21]. This dataset contains over 46,000 color images, of the shape 80x60, of many of the normally worn items of clothing by people in their normal day to day lives. From this dataset, we chose to train our CNN model using the option of sub-category labels of 45 classes which included 'top wear', 'bottom wear', 'shoes', 'socks', 'watches' etc. Our CNN model was trained on 38,000 images, and the remaining 6446 were used for testing to evaluate its performance. Using the architecture and configuration we previously described, we attained a high level of accuracy as shown in Fig. 8 and 9. The relatively low number which appears is a result of a tradeoff that we had to make of apportioning the greater number of the images to our training set and only having a few left for testing, since our dataset was not very large, especially accounting for the large number of classes that we had.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1
1	0.14	0.50	0.22	2
2	0.70	0.59	0.64	73
3	1.00	0.12	0.22	8
4	0.97	1.00	0.99	114

Figure 8. CNN model evaluation header snippet.

40	0.95	0.98	0.97	1085
41	0.88	0.98	0.93	100
42	0.91	0.89	0.90	45
43	0.95	0.93	0.94	432
44	0.92	0.90	0.91	151
accuracy			0.93	6446
macro avg	0.65	0.60	0.60	6446
weighted avg	0.93	0.93	0.93	6446

Figure 9. CNN model evaluation snippet.

The metrics shown are some of the most commonly used evaluation metrics, and from the snippets it can be seen that the classes which had more images belonging to them for the purposes of training the network, were able to attain higher

degrees of accuracy and precision. From Fig.8 and 9, the macro average can be seen, which means that each class is given equal significance regardless of how many occurrences are within it. This explains why it attained a lower accuracy score than the consequent weighted average, which aggregates the contributions of all classes to compute the average metric. Hence a higher accuracy is obtained for the weighted average, since the number of images were not distributed evenly among all of the 45 classes. Some classes contained significantly more images for testing and training than other classes. Therefore, this gave us a high confidence level and assurance that our model was relevantly working, which in turn allowed us to proceed and trust the features obtained as output from this model, and use them for similarity search with distance metrics.

For the distance metric i.e. measuring the distances between the vector ***features***, we utilized Euclidean distance which measures the shortest distance between two vector points in a space of (a, b…n) dimensions. We apply the appropriate Euclidean formula for ***a, b***, where if $\mathbf{a} = (a_1, a_2)$ and $\mathbf{b} = (b_1, b_2)$ then the distance is given by

$$d(a, b) = \sqrt{(a1 - a2)^2 + (b1 - b2)^2} \quad (4)$$

We also utilized cosine distance, which is given by,

$$\text{Similarity}(p,q) = \cos\theta = \frac{p \cdot q}{\|p\|\|q\|} = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i^2}\sqrt{\sum_{i=1}^{n} q_i^2}} \quad (5)$$

WHERE ***p, q,*** are the vectors of the image features of the input query images and features of the images in our database/ storage respectively.

IV. Prepare Application/ Software Experiment Results

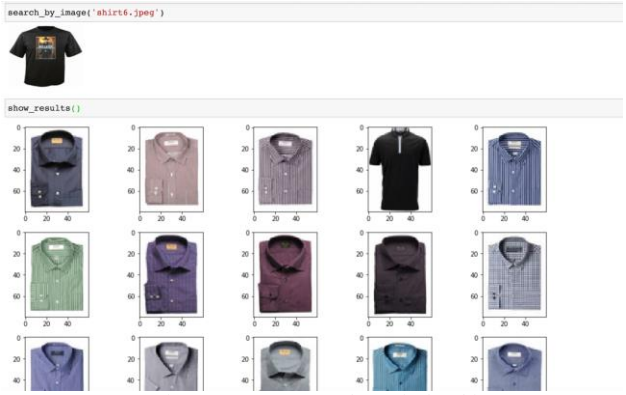

Figure 10. Image search results by shirt.

The implementation of our application is as shown below. From our dataset we selected 10 classes, some of which had the relatively highest number of sample training images. For each of these classes, we then performed tests by using 10 different images and then calculating an accuracy score, e.g. 10 searches within the top wear class using 5 T-shirts and 5 shirts. Below are some snippets of the image search results, and the subsequent table of results using Euclidean distance and cosine distance respectively.

Fig. 10 shows an instance of us testing the software by using an image, 'shirt6.jpeg', which was not part of the original test/ train dataset. This image, along with all the other images used during this application testing process, was randomly selected from the internet. It is used as an example of the query image that a user would input into the system. The shirts displayed below that under the 'show results ()' section are the images which have been ranked by our system based on closest in similarity.

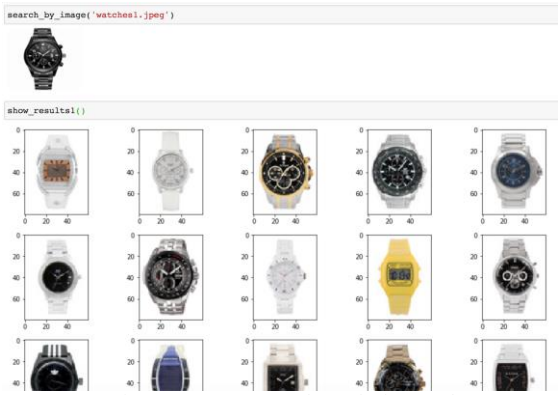

Figure 11. Image search results by watch.

Here Fig. 11 is another example of a random image used to test the system. In Figure 10 we were testing the 'topwear' class. Fig. 11 is an instance of the search results obtained from a test in the 'watches' class.

TABLE I. Table of Results Using Euclidean Distance

CATEGORY	1 %	2 %	3 %	4 %	5 %	6 %	7 %	8 %	9 %	10 %	AVERAGE %
BAGS	0.96	0.00	0.12	1.00	1.00	0.00	0.25	0.8	0.16	0.8	0.509
TOP WEAR	0.96	0.6	0.32	1.00	1.00	1.00	1.00	0.48	0.68	0.04	0.708
BOTTOM WEAR	1.00	1.00	1.00	1.00	1.00	0.96	1.00	1.00	0.08	1.00	0.908
WATCHES	1.00	0.6	1.00	1.00	1.00	0.36	0.84	0.00	1.00	0.00	0.68
BELTS	0.92	0.00	0.00	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.188
SHOES	0.04	0.12	0.04	0.12	0.00	0.00	1.00	0.36	0.52	1.00	0.320
SOCKS	0.00	0.00	0.00	0.00	0.00	0.32	0.00	0.00	0.36	0.04	0.072
DRESSES	0.08	0.12	0.00	0.96	0.68	0.28	0.44	0.68	0.24	0.12	0.360
SAREE	0.00	0.00	0.00	0.00	0.04	0.50	0.28	0.28	0.00	1.00	0.190
JEWELLERY	0.92	0.12	0.00	0.12	0.88	0.16	0.04	0.76	0.32	0.00	0.320

Table I above shows the percentage accuracy obtained for each iteration. As can be seen, each class was tested for 10 iterations, using a different query image belonging to the class each time. For this table, the distance metric that was

used to rank the similarity of the images is Euclidean distance.

TABLE II. TABLE OF RESULTS USING COSINE DISTANCE

	1 %	2 %	3 %	4 %	5 %	6 %	7 %	8 %	9 %	10 %	AVERAGE %
CATEGORY											
BAGS	0.96	0.00	0.12	1.00	1.00	0.00	0.25	0.8	0.16	0.8	0.509
TOP WEAR	0.96	0.6	0.32	1.00	1.00	1.00	1.00	0.48	0.68	0.04	0.708
BOTTOM WEAR	1.00	1.00	1.00	1.00	1.00	0.96	1.00	1.00	0.08	1.00	0.908
WATCHES	1.00	0.6	1.00	1.00	1.00	0.36	0.84	0.00	1.00	0.00	0.68
BELTS	0.92	0.00	0.00	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.188
SHOES	0.04	0.12	0.04	0.12	0.00	0.00	1.00	0.36	0.52	1.00	0.320
SOCKS	0.00	0.00	0.00	0.00	0.00	0.32	0.00	0.00	0.36	0.04	0.072
DRESSES	0.08	0.12	0.00	0.96	0.68	0.28	0.44	0.68	0.24	0.12	0.360
SAREE	0.00	0.00	0.00	0.00	0.04	0.50	0.28	0.28	0.00	1.00	0.190
JEWELLERY	0.92	0.12	0.00	0.12	0.88	0.16	0.04	0.76	0.32	0.00	0.320

Table II above shows the results of the same experimental procedure as from Table I. All the test images that were used remained the same. Here, the only change that was made was the distance metric; the algorithm by which the similarity of images is ranked. It also shows the percentage accuracy of results per test image, and also by class. The last columns for both tables show the average score of accuracy attained by the system for that particular class.

V. CONCLUSION

We were able to define and train our own Convolutional Neural Network (CNN) using a custom dataset, and using our own configurations. After successfully training the network and attaining a relatively high degree of accuracy, we were able to apply Euclidean and Cosine distances to implement a reverse search engine. From the tables above, it is clear to see that the results between the different distance metrics that we utilized are nearly entirely identical. From this we can draw a conclusion that the greater part of the accuracy and precision of an image search system depends on the accuracy of the image features obtained. In our case of using CNNs, this means the accuracy of the system heavily depends on the accuracy of the trained CNN model.

We also made an observation concerning the accuracy of the CNN. The categories that had a highest number of images belonging to their classes during training were able to generally attain a higher degree of accuracy during image search experimentation. However, there were still few instances anyhow where the classes that did not have many images belonging to them still performed excellently. This leads us to conclude that having a very large number of training images per each class is very helpful to prevent the CNN model from overfitting during training.

VI. FURTHER WORK AND IMPROVEMENTS

Further work can still be done on this topic at a later time. The configurations of the CNN used may be adjusted, and tested for further refinement. A dataset with more images may also be used to improve the robustness of the model. One of the shortcomings we encountered was that some of the image classes did not contain enough images for adequately training the CNN model for a high degree of accuracy for that particular class. Also another set of experiments that may be conducted is one in which, after satisfactory training of the CNN, other distance metrics, such as hamming distance, k nearest neighbors etc. may be implemented to complete to complete the similarity search component, and then tested against each other using the same images, in order to determine a most effective distance metric.

REFERENCES

[1] Wengang Zhou, Houqiang Li, and Qi Tian, “Recent advance in Content-based Image Retrieval: A literature survey,” in Fellow, IEEE, Sept. 2017.

[2] A. Nodari, M. Ghiringhelli, A. Zamberletti, M. Vanetti, S. Albertini, and I. Gallo, “A visual search application for content based image retrieval in the fashion domain,” University of Insurbia, June 2012.

[3] TinEye. www.tineye.com

[4] Google image search: http://images.google.com/.

[5] Bing Image Search: https://www.bing.com/images/.

[6] Taobao: www.taobao.com

[7] Xian-Sheng Hua, Meng Wang, Microsoft Research Asia, Hong-Jiang Zhang, “Image retagging,” ACM 978-1-60558-933-6, Oct. 2010.

[8] Desai, Padmashri and Pujari, Jagadeesh and Kinnikar, Anita, “An image retrieval using combined approach wavelets and local binary pattern,” B.V.B. College of Eng. and Tech. Hubli, Karnataka, Aug. 2016.

[9] Datta, Ritendra and Joshi, Dhiraj and Li, Jia and Wang, James, “Image retrieval: Ideas, influences, and trends of the new age,” ACM Comput. Surv, vol. 40, pp 1–60, Jan. 2008.

[10] Karthik Kumar, Yamini Nimmagadda, and Yung-Hsiang Lu, “Energy conservation for image retrieval on mobile systems,” ACM Transactions on Embedded Computing Systems, 2012, pp 66.

[11] Yann Le Cun, Leon Bottou, Yoshua Bengio, and Patrick Haffner, “Gradient based learning applied to document recognition," IEEE, Nov. 1998.

[12] Hasan Rashaideh, Habes Alkhraisat, and Alaa T. Al Ghazo, “Building a context image-based search engine using multi clustering Technique,” Computer Science Dept. Al-Balqa Applied University Salt, Jordan, 2015.

[13] P. S. Hiremath and Jagadeesh Pujari, “Content based image retrieval based on color, texture and shape features using image and its complement,” Dec. 2007, International Journal of Computer Science and Security, vol. 1: Issue 4.

[14] Ruofei Zhang, Zhongfei Zhang, “Addressing CBIR efficiency, effectiveness, and retrieval subjectivity simultaneously,” Dept. of Computer Science, SUNY, Binghamton, NY, Jan. 2003.

[15] Jing Huang, S. Ravi Kumar et al, “Image indexing using color correlograms”, Proc. of the IEEE CVPR, Puerto Rico, Jun. 1997.

[16] Fukushima Kunihiko, “Neocognitron: A self organising neural netwrk model for a mechanism of pattern recognition unaffected by shift in position,” Acta neurochirurgica, Supplementum, vol. 40, pp 51-67, Feb. 1987.

[17] Hyeok-June Jeong, Kyeong-Sik Park, and Young-Guk Ha, “Image preprocessing for efficient training of YOLO deep learning networks,” Jan. 2018, pp 635-637, doi:10.1109/BigComp.2018.00113.

[18] Keiron O'Shea, and Ryan Nash, "An introduction to convolutional neural networks," Department of Computer Science, Aberystwyth University, Ceredigion School of Computing and Communications, Lancaster University, Lancashire, Nov. 2015.

[19] E. Wang Yang, Zheng Jiachun, "Real-time face detection based on YOLO," Aug. 2018, Electronic Publication: doi:10.1109/ICKII.2018.8569109.

[20] Gaurang Panchal , Amit Ganatra , Y. P. Kosta and Devyani Panchal, "Behavior analysis of multilayer perceptrons with multiple hidden neurons and hidden layers," International Journal of Computer Theory and Engineering, Vol. 3, No. 2, April 2011.

[21] https://www.kaggle.com/paramaggarwal/fashion-product-images-dataset