Soft Comput (2017) 21:2923–2939
DOI 10.1007/s00500-017-2585-8

FOCUS

Deep net architectures for visual-based clothing image recognition on large database

Ju-Chin Chen[1] · **Chao-Feng Liu**[1]

Published online: 27 April 2017

Abstract In the Big Data era, there is a need for powerful visual-based analytics tools when pictures have replaced texts and become main contents on the Internet. Hence, in this study, we explore convolutional neural networks with a goal of resolving clothing style classification and retrieval tasks. To reduce training complexity, low-level and mid-level features were learned in the deep models on large-scale datasets and then transfer learning is incorporated by fine-tuning pre-trained models using the clothing dataset. However, a large amount of collected data needs huge computations for tuning parameters. Therefore, one architecture inspired from Adaboost is designed to use multiple deep nets that are trained with a sub-dataset. Thus, the training time can be accelerated if each net is computed in one client node in a distributed computing environment. Moreover, to increase system flexibility, two architectures with multiple deep nets with two outputs are proposed for binary-class classification. Therefore, when new classes are added, no additional computation is needed for all training data. In order to integrate output responses from multiple nets, classification rules are proposed as well. Experiments are performed to compare existing systems with hand-crafted features. According to the results, the proposed system can provide significant improvements on three public clothing datasets for style classifications, particularly on the large dataset with 80,000 images where an improvement of 18% in accuracy was recognized.

Keywords Deep Learning · Convolutional neural network · Clothing image recognition

1 Introduction

As the popularity of digital products and the wide development of sensor techniques, large sets of data that contain numbers, textures, images, and videos are produced. The authors predict the data amount collected globally will reach 35 trillion GB before 2020 (Gantz and Reinsel 2011). Some companies would like to collect this large volume of data and then extract useful information from it such as Facebook and Google. Formally, Big Data means that data are rapidly generated and added in four Vs (Chen and Lin 2014; Najafabadi et al. 2015), volume—the number of data amount, variety—the raw data is diverse and complex, velocity—the increasing rate of data, and veracity—the usefulness of results obtained from data analysis. However, the amount of data that cannot be analyzed using the traditional database techniques in storage, processing, or computation. Thus, dealing with Big Data has become a hot topic in recent years.

Not only storage and computation techniques need to be overcome for large data but also Big Data Analytics mining meaningful information from massive data to provide decision and prediction is a very important issue. To provide business analysis and decision guidance, some companies such as Google and Microsoft process data in exabyte proportions or larger. While social software companies such as Facebook and Twitter collect log records and use behaviors from billions of users, they also generate a very large quantity of data. However, Big Data Analytics faces a series of

Communicated by C.-H. Chen.

✉ Ju-Chin Chen
jc.chen@cc.kuas.edu.tw

Chao-Feng Liu
ggyy899770@gmail.com

[1] Department of Computer Science and Information Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan, ROC

challenges to dealing with billions of data containing thousands of dimensions, features, and category labels (Chen and Lin 2014). Therefore, extracting valuable knowledge from a massive data set is not an ordinary task and cannot be completed in a conventional machine learning scenario.

Recently, one machine learning technique, called Deep Learning, provides a solution to address the data analysis and learning problems for the Big Data. Deep Learning is to emulate the process of the sensorial areas of the neocortex in the human brain, which can automatically learn high-order hierarchical representation in deep architectures from the underlying data (Bengio et al. 2013; Arel et al. 2010). Thus through a hierarchical learning process, low-level features to complex high-level features can be extracted via the Deep Learning algorithm. In addition, Deep Learning algorithm can learn relational and semantic knowledge data representations from large unsupervised raw data at high-level layers (Hinton and Osindero 2006; Bengio et al. 2007) that conventional techniques that use shallow learning hierarchies fail to extract complex and nonlinear patterns efficiently. Now, the Deep Learning algorithm has been successful in some domains, for example, speech recognition (Dahl et al. 2012; Mohamed et al. 2012), natural language process (Socher et al. 2011) and computer vision (Ciresan et al. 2010; Krizhevsky et al. 2012). Google, Apple, Microsoft, and IBM (Jones 2014; Wang et al. 2011) have developed projects using Deep Learning to analyze collected data from users. For example, the virtual assistant in Apple cell phones, Siri, can provide various services such as weather reports, news updates, and order tickets, through voice (Efrati 2013). Thus, with the aid of Deep Learning algorithms, discriminative tasks such as semantic indexing, information retrieval, and discriminative modeling can be easily developed in Big Data Analytics. However, for some high-dimensional data such as images and videos factors that both increase data volume, learning processing with a hierarchical structure is slow and computationally expensive (Sukumar 2014). Therefore, the application of Deep Learning algorithms for Big Data Analytics involving high-dimensional data remains unexplored (Sukumar 2014).

Considering images becoming the main contents on the Internet and social media applications such as social networks, there has been a rapid increase in the size of digital images collected from online users. Additionally, more commercial benefits can obtain by analyzing those collected data to predict the consumers' behavior and then recommend products. Hence, in this study, we explore deep learning to analyze high-dimensional clothing images in an attempt to discover fashion style elements that can be used for consumption behavior prediction in the Big Data era. In the conventional computer vision field, lots of low-level features such as LBP (Ojala et al. 2002) and SIFT (Lowe 2004) have been developed to extract specific patterns from domain images. However, there is a large gap between these handcrafted features and the abstract meaning. Inspired from the learning ability for a large amount of data, we explore deep learning to mine low- and high-level features from fashion images in a hierarchical layered structure and further extend for the clothing retrieval task. Moreover, three modified deep architectures were developed by considering computational efficiency and the system flexibility. Different from conventional algorithms with shallow structure, the deep net contains many parameters to be optimized. A slow learning process for high-dimensional image data, particularly massive amounts of data collected from social applications or on-line shopping sites makes the system prohibitive. Therefore, inspired by the ideas of ensemble learning, one architecture is designed to use multiple deep nets and then each net is computed in one client node in a distributed computing environment. Thus, the training time can be accelerated. On the other hands, in order to increase system flexibility, two architectures with multiple deep nets with two outputs are proposed for binary-class classification. Therefore, when new classes are added, no additional computation is needed for all training data.

The remainder of this paper is organized as follows: in Sect. 2, we review the deep learning for computer applications and related studies in vision-based analysis for clothing images. In Sect. 3, the system framework for knowledge discovery of fashion style patterns and the learning process of deep net is introduced. In Sect. 4, three modifications of deep architectures are discussed to simulate the learning process of a large volume of data in the distributed computing environment. In Sect. 5, experiments are performed to evaluate the system performance and a conclusion is made.

2 Related work

This section introduces the history of Deep Learning and its usage in computer fields by examining studies pertaining to fashion related discuss.

2.1 Deep Learning

Deep Learning is a hierarchical model used to emulate computations similar to human learning. Different from conventional machine learning, which is a method best suited for simple structure small data volumes that incorporate shallow learning architectures, deep Learning can extrapolate abstract and complex data representations using a hierarchical layered structure where less abstract features are learned in the lower-level, while more abstract features are extracted in the higher-level. Deep learning can be used for a supervised problem if the labeled data is available. Moreover, Google and Stanford (Le et al. 2012) proposed a vast and

deep neural network to learn high-level features by using only unlabeled data. In Google's experimentation, they trained a nine-layered sparse autoencoder with 1 billion connections using ten million 200 × 200 images downloaded from the Internet (Sukumar 2014). The experiments were performed at a computational cluster of 1000 machines with 16,000 cores, and the training duration was of 3 days. Using the extracted high-level abstract features, the trained deep net can recognize 22,000 object categories from the ImageNet dataset. Since deep learning demonstrated the ability to extrapolate an abstract representation from large amounts of unlabeled/unsupervised/unseen data and outperformed than other conventional machine learning methods, deep learning becomes an attractive technique especially in the Big Data era.

In the deep net, the input of each layer is the output from the previous layer. In other words, data are sequentially passed from the first layer to the last output as the abstract representation. Using the layer structure, more abstract features are obtained by passing data through multiple nonlinear computations. For example, in face recognition, the first layer can learn the edges in different orientations. In the second layer, more complex features like the different parts of a face such as lips, nose, and eyes are learned. In the third layer, features like face shapes are learned. The final representations can be used for face recognition, semantic data indexing, or other classification problems. The idea of Deep Learning is that the input data is nonlinearly transformed in each layer (Sukumar 2014), which tries to extract underlying abstract factors in the data. Li et al. (LeCun et al. 1998) applied Deep Learning to develop the Microsoft Research Audio Video Indexing System (MAVIS) for speech recognition to provide a search service of audio and video files that use speech. In their study, the authors proposed convolutional neural networks (CNN) to scale up effective Deep Learning to high-dimensional data. In CNNs, the neuron units in the latent layers do not need to be connected to all of the nodes in the previous layer and the resolution of the high-dimensional image data can be reduced via sampling methods when feeding data into higher layers in the network (LeCun et al. 1998). Researchers have taken the advantage of CNNs to achieve impressive accuracy improvement in image classification on the ImageNet dataset, one of the largest databases for image object recognition with 1.2 million 256 × 256 RGB images (Krizhevsky et al. 2012). Additionally, CNNs are applied to many other computer vision applications such as object detection (Girshick et al. 2014), face recognition (Sun et al. 2015), and human attribute prediction (Zhang et al. 2014).

Recently, there have been improvements on Deep Learning including the usage of drop-out (Hinton et al. 2012) to prevent the optimization from overfitting. Also, more effective nonlinear activation functions are used such as rectified linear units (Glorot et al. 2011), max-outs (Goodfellow et al. 2013), and network-in-network (NiN) functions are proposed to reduce dimensionality (Lin et al. 2013). In addition, Socher et al. (2011) introduced recursive neural networks which can predict hierarchical tree structures for segmentation and annotation of complex image scenes and the performance can surpass other existing methods based on conditional random fields. Le et al. (2011) demonstrate that Deep Learning can be used for action scene recognition and video data tagging. Their approach outperforms other existing methods which have adopted hand designed features for images like SIFT and HOG on the video domain. These studies show the advantage of Deep Learning as an approach to extract data representations from different data types.

Based on the underlying assumption that the low- or mid-level features are common across different problem domains, the domain adaption or transfer with deep neural networks gains increased attention. The majority of existing approaches adapts to re-train the last few layers of the network using samples from the target domain, or performs fine-tuning of all the layers using backpropagation (Razavian et al. 2014; Oquab et al. 2014). Several works have shown that it is effective to transfer a Deep Learning model from a large-scale dataset, e.g., ImageNet, to other tasks (Chen et al. 2015b; Huang et al. 2015). However, these approaches usually require a relatively large training sample from the target domain to produce good results (Chen et al. 2015b). Thus, Chen et al. (2015b) proposed a new deep domain adaption approach using a double-path network to learn domain-invariant hierarchical features directly and transfer the domain information within intermediate layers to bridge the gap between the source and target clothing domain distributions. Alternatively, Huang et al. (2015) proposed a network architecture that learns effective features for measuring visual similarities across domains for cross-domain image retrieval.

2.2 Visual-based analytics on fashion images

Advances in mobile technology combined with the ubiquity of Internet service creates an environment where users are taking more pictures and posting them online. Because of the benefits realized by modeling and predicting consumer's consumption behavior from crawled data, there is a growing interest in popularity prediction based on online contents (Yamaguchi et al. 2014). As pictures become a core content type on the Internet, and human behavior migrates to social network sharing, Big Data increasingly generated from the login and browsing records attracts a lot of attention. For example, an online economic service can accept pictures as input and return corresponding recommendations for users. In Yamaguchi et al. (2014), the authors examine the social influence of clothing images

in an online fashion social network using vision-based analytics.

Because of the high revenue of these services and the huge impact for e-commerce applications, fashion pictures were selected as an influential target that would also be popular. There is a growing interest in methods for garment understanding or recognition (Yamaguchi et al. 2012, 2013), product suggestion (Kalantidis et al. 2013), outfit recommendation (Jagadeesh et al. 2014; Liu et al. 2012), clothing retrieval (Kalantidis et al. 2013), fashion style recognition (Chen and Liu 2015) and clothing attribute prediction (Bossard et al. 2013). Among applications, human parsing, namely partitions the human body into several clothing specific regions (e.g., hat, left/right leg, and upper-body clothes) gives each pixel in the input image a semantic label. This has drawn much attention in recent years (Yamaguchi et al. 2012, 2013), serving as the basis for other related applications such as clothing classification, and retrieval (Liu et al. 2014). The frameworks of pixel-based parsing are proposed to segment and classify simultaneously. Human parsing can be roughly divided into two parametric and nonparametric-based methods. For the parametric method, Yamaguchi et al. (2012) proposed an image parsing system that consisted of three parsers with different classification models to recognize the clothing classes for each pixel, and then combined all results from the parsers for final prediction. Yang et al. (2014) developed the human parsing system consisting of two sequential phases for image co-segmentation and region co-labeling to capture the correlations between different human images. Conversely, nonparametric methods build a pixel- (Liu et al. 2011) or hypothesis-level (Tung and Little 2014) to match a new image and annotated images in a dataset. The labels are then transferred from the annotated images to the new image. Instead, of combining multiple sequential steps, CNN-based methods have been proposed for image parsing (Farabet et al. 2013). Farabet et al. (2013) trained a multi-scale CNN from raw pixels to assign a label to each pixel. Long et al. (2014) proposed recurrent CNNs, the state-of-the-art scene parsing method, to speed up parsing time. However, these deep models cannot be easily updated when new semantic labels are incorporated (Liu et al. 2015). To increase flexibility of human parsing, Liu et al. (2015) proposed a matching convolutional neural network (M-CNN) to predict the matching confidence based on the k-nearest-neighbor (KNN) nonparametric framework. Different from the classic CNN architecture, the cross image matching filters are embedded into every convolutional layer to characterize the matching between the testing image and semantic region of KNN images. Then the output is fused by displacing the best matched region in the testing image for a particular semantic region in one KNN image.

Garment and clothing classification/retrieval is a hot issue as well. Song et al. (2011) predicted one's occupation by recognizing their clothing styles. Di et al. (2013) defined 12 fine-grained clothing classes: material, fastener types, collar types, folded collar, overcoat and jacket, and the hand-crafted features including LBP, SIFT, and histogram of gradient (HOG) for designation with SVM classifiers. In 2012, Bossard et al. (2013) proposed a clothing recognition system for 15 common classes with 78 attributes. The system identifies the upper-body region as region of interest (ROI) and extracts speeded up robust features (SURF), local binary patterns (LBP), and CIE $L^*a^*b^*$ color space as feature descriptors. Multi-class learning based on a Random Forest random is applied to recognize clothing styles. However, only 41% accuracy can be achieved. Different from previous works that focus on classifying or retrieving similar garment from images, Chen et al. (2015a) proposed a sparse-coding representation approach to discover the elements from ten fashion styles with color, LBP, and HOG features. However, only 68% accuracy rate was achieved. Although the authors define ten fashion styles by investigating color and texture statistics in their work, the definition of each fashion style is abstract and difficult. Conversely, Kiapour et al. (2014) designed an online competitive style rating mechanism to associate the style ratings of five style categories hipster, bohemian, pinup, preppy, and goth, for clothing based on reliable human judgments. Then, between- and within-class style classifications are performed by using the proposed style descriptor with linear kernel SVM classifiers. Although these issues are interesting, the ability to analyze them is limited for the existing methods are based on conventional machine learning techniques within a natural setting.

Rather than using conventional machine learning (Bossard et al. 2013; Chen et al. 2015a; Kiapour et al. 2014), the domain adaption or transfer with deep neural networks has been explored in the field of garment and clothing classification/retrieval. Khosla and Venkataraman (2015) applied CNNs with multiple VGGNet architectures to classify an input shoe image into one shoe category. Transfer learning is then applied in the VGGNet architecture, and the last fully connected layer from the trained model feature vector is input to retrieve the most similar five shoes in the data. A similar idea is applied in Bossard et al. (2013) where Random Forests are extended to be capable of transfer learning from different domains to reduce noise effects that exist in the crawled image data. Huang et al. (2015) proposed a dual attribute-aware ranking network (DARN) to address the gap between the user photo in a cluttered background and online images with a clean background in clothing retrieval problem. DARN consists of two subnetworks with similar structures for shopping scenarios and

street scenarios. In addition, an enhanced R-CNN detector is applied to localize the clothing area in images with cluttered backgrounds. Chen et al. (2015b) addressed the same problem of bridging the gap between two clothing domains by implementing a specific double-path deep neural network where each deep network is used to model one clothing domain and additional alignment layers have been placed to connect the two paths for the domain consistency. In Lin et al. (2015), the authors developed a hierarchical deep search framework for clothing retrieval in a recommendation system. Mid-level visual features were learned in one pre-trained network, and then the network was fine-tuned using the closing dataset. Note that a latent layer was added into the network and hash codes were learned. The authors conducted the experiments on the dataset with 15 clothing categories and 161,234 clothing images from Yahoo shopping websites. However, in the Big Data era, the flexibly and scalability of models are important issues. Chen and Liu (2015) proposed three modified deep net architectures for distributed computational environment. According to the experimental results, the classification rates for fashion style recognition were significantly improved based on the deep learning methods. Moreover, via the distributed computation, the classification rates are compatible with the original architecture and the flexibly of the deep learning would be improved.

3 System framework

Figure 1 depicts the CNN used to solve style recognition and clothing pattern discovery. Comprised of three components, the first component is the supervised pre-trained model on the large ImageNet dataset (Krizhevsky et al. 2012; Lin et al. 2015). Based on the assumption that the parameters of the low- and mid-level network layers can be re-used across domains (Oquab et al. 2014; Chen et al. 2015b; Khosla and Venkataraman 2015), domain transfer learning is performed in the second component by using clothing style images to fine-tune the pre-trained network with the latent layers containing multiple nodes (Lin et al. 2015). In the third component, the outputs of nodes in the latent layer from the re-trained network are used as feature vectors for the clothing retrieval task. Note that because three modifications are performed on this framework as a part of our work (introduced in the next section), the framework contains one deep net with multiple outputs represented as *Architecture 1*.

Figure 2 shows the deep convolutional net with eight layers which contain five convolution layers, two fully con-

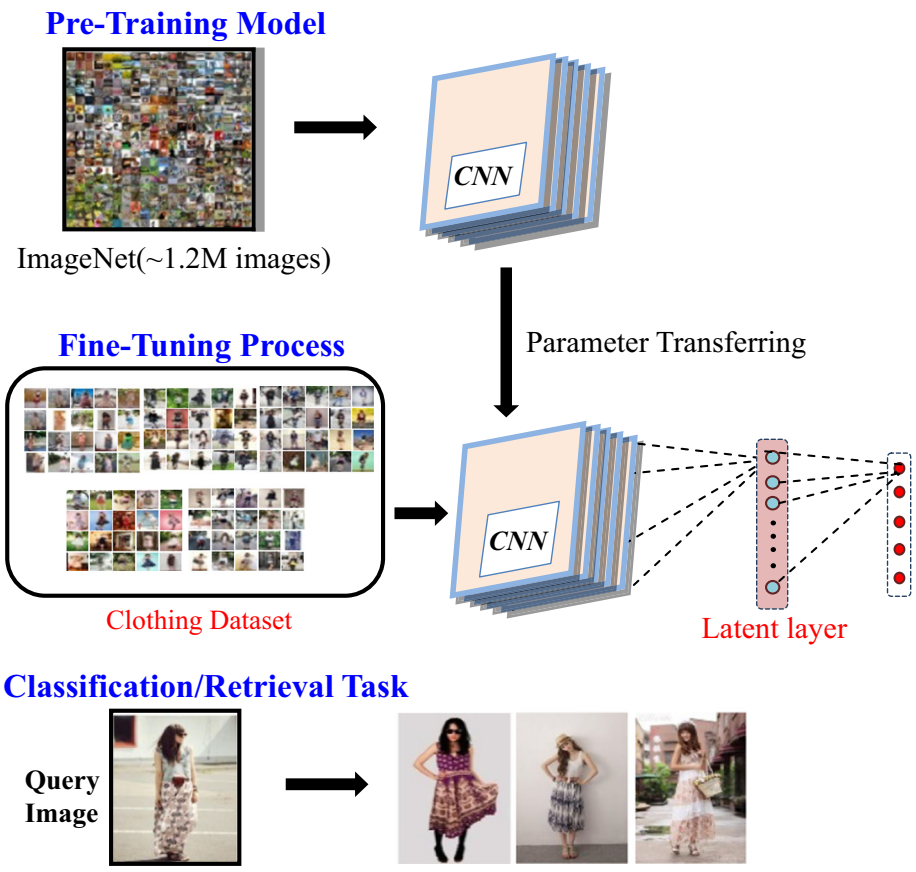

Fig. 1 System framework. Three components, pre-trained model, fine-tune process, and retrieval task. Note that *Architecture 1* contains one deep net with multiple outputs

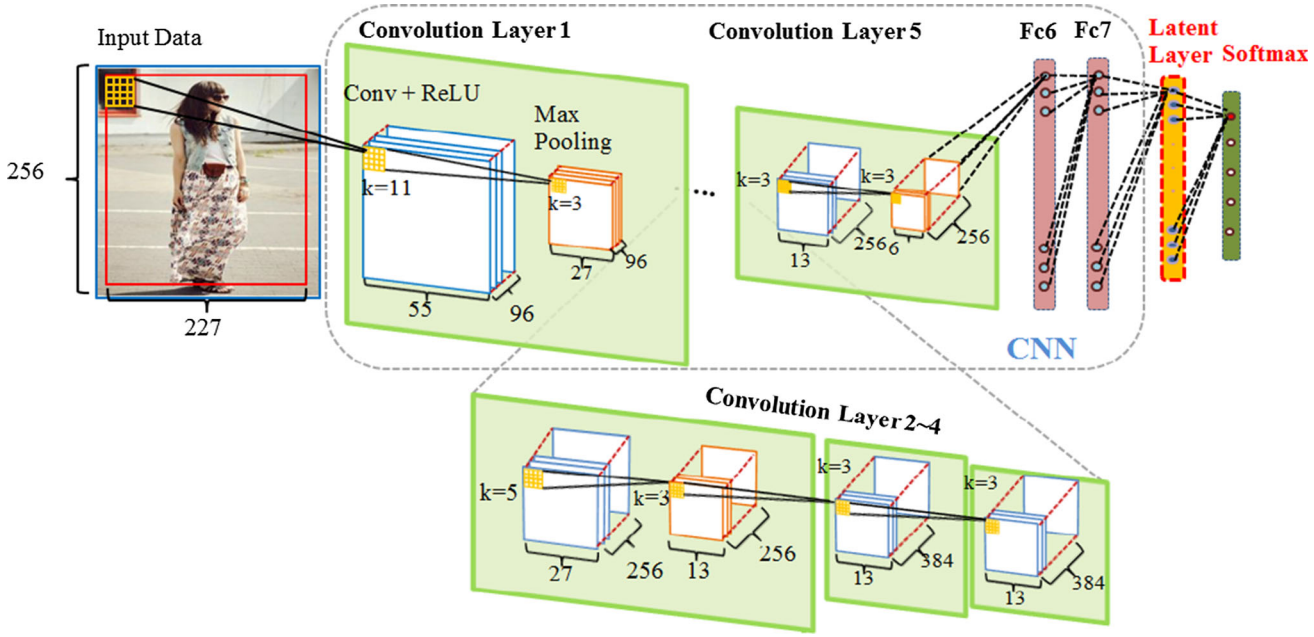

Fig. 2 Deep convolutional net with eight layers

nected layers, one latent layer, and the softmax output corresponding to the clothing styles. The first convolutional layer filters the $227 \times 227 \times 3$ input region (Fig. 2), randomly cropped from $256 \times 256 \times 3$, with 96 Gaussian kernels of size $11 \times 11 \times 3$ with a stride of four pixels. Then the rectified linear unit (ReLU) nonlinearity is applied and the responses are pooled with kernel size of 3×3 with a stride of two pixels, normalized, and padded as the output of the first convolution layer. The second convolutional layer takes the output of the first convolutional layer as input and filters it with 256 kernels of size $5 \times 5 \times 48$. Post-processing, the same as in the first convolutional layer, are performed before subsequent layers. The third convolutional layer has 384 kernels of size $3 \times 3 \times 256$ connected to the outputs of the second convolutional layer, and only the ReLU function is applied without pooling or normalization process. The fourth convolutional layer has 384 kernels of size $3 \times 3 \times 192$, and the fifth convolutional layer has 256 kernels of size $3 \times 3 \times 96$. The fully connected layer (fc6) containing 4096 nodes takes the output of the fifth convolution layers as input, and the pooling and dropout function is applied to reduce overfitting in the fully connected layer (Jagadeesh et al. 2014) by setting the output of latent neurons with a probability smaller than 0.5–0. Before passing to the output layer with the softmax function, a latent layer with 64, 128, or 256 nodes is added to extract features vectors for the retrieval. Because the memory requirements to train this network are huge, the batch size is set to 64 in the training process. After fine-tuning the network, the test data is fed into the network and the outputs of the softmax function become the classification results.

Figure 3 shows the detailed forward and backward computational propagation in the convolution layer. In general, the convolution layer comprises three functional elements, namely, convolution operator, active function, and pooling. Assuming the weight features of convolution filter to be $w = [\, w_1 \; w_2 \; \ldots \; w_n \,]$ where n is the length of w, the convolution operator is defined as

$$y = x \times w = [y_n], \quad \text{where} \quad y_n = \sum_{i=1}^{|w|} x_{n+i-1} w_i . \tag{1}$$

The active function was then applied to the convolution results; the commonly used active functions include sigmoid, Tanh, Maxout, and ReLU. In the proposed work, ReLU function was applied as

$$\hat{y} = \max(0, y_n) \tag{2}$$

In the forward propagation, the downsampling operator was applied to the pooling layer. The aggregate function g was applied to the subsample results from the previous layer to avoid overfitting. The input of function g is a vector, and the output is a scalar. Commonly used aggregation functions include mean pooling, max pooling, and L^p pooling. Here the max pooling is applied. With m as the size of pooling region, the max pooling is represented as

$$g\,(\tilde{y}) = \max\left(\hat{y}_n\right), \quad n = 1, \ldots, m \tag{3}$$

In the training process of the deep net, the stochastic gradient descent was applied to re-weigh the parameters in the back-

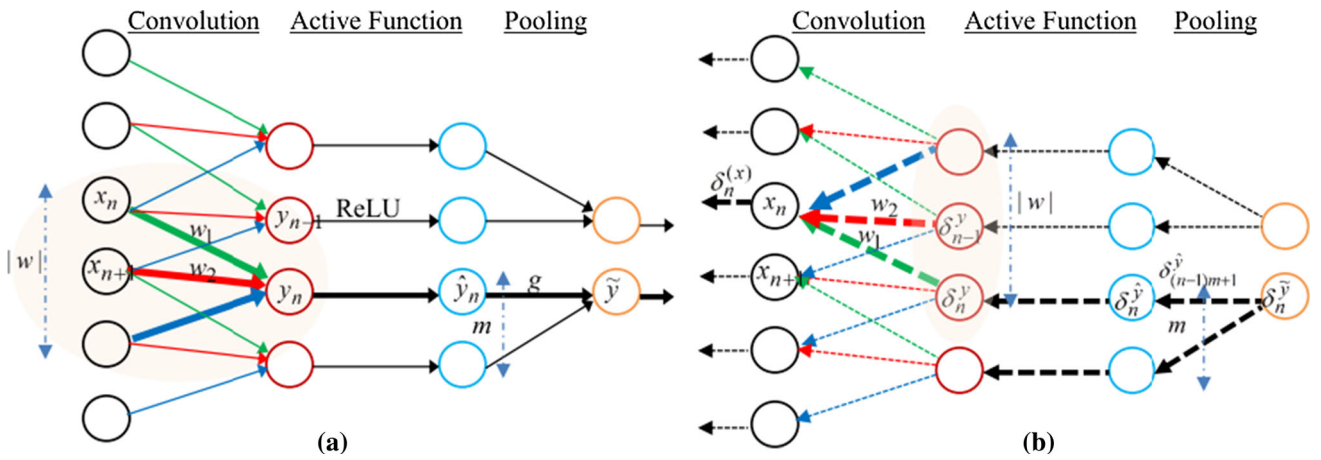

Fig. 3 **a** Forward and **b** backward propagation computation in the convolution layer

propagation process. As shown in Fig. 3b, the parameters in the pooling layer were firstly updated. From the pooling layer to the activation function, the error signals $\delta^{\hat{y}}_{(n-1)m+1:nm}$ for each training example were computed by upsampling using the following equation:

$$\delta^{\hat{y}}_{(n-1)m+1:nm} = \delta^{\tilde{y}}_n g'_n = \delta^{\tilde{y}}_n \frac{\partial g}{\partial \hat{y}_{(n-1)m+1:nm}} = \frac{\partial J}{\partial \tilde{y}_n} \frac{\partial \tilde{y}_n}{\partial \hat{y}_{(n-1)m+1:nm}} = \frac{\partial J}{\partial \hat{y}_{(n-1)m+1:nm}} \tag{4}$$

Note that the gradient of the max pooling function is $\frac{\partial g}{\partial z_i} = \begin{cases} 1 & \text{if } z_i = \max(z) \\ 0 & \text{otherwise} \end{cases}$, where z is a dummy variable. After obtaining the error signals propagated from the pooling layer, the error signals in the convolutional layer can be obtained by

$$\delta^y_n = \delta^{\hat{y}}_n \bullet f'(y_n) \tag{5}$$

where $f'(y_n)$ is the derivative of the active function. Then, the error signal δ^x_n propagated from the convolution layer can be obtained by the equation:

$$\delta^x_n = \frac{\partial J}{\partial x_n} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial x_n} = \sum_{i=1}^{|w|} \frac{\partial J}{\partial y_{n-i+1}} \frac{\partial y_{n-i+1}}{\partial x_n} = \sum_{i=1}^{|w|} \delta^y_{n-i+1} w_i \tag{6}$$

In other words, it can be viewed as the convolution of and the flip of w. The gradient of the error function with respect to w is

$$\frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial w_i} = \sum_{n=1}^{|x|-|w|+1} \frac{\partial J}{\partial y_n}\frac{\partial y_n}{\partial w_i} = \sum_{n=1}^{|x|-|w|+1} \delta^x_n x_{n+i-1} \tag{7}$$

Then in SGD process (Krizhevsky et al. 2012), the weight of filters could be updated by

$$\begin{aligned} v &= \gamma v + \alpha \nabla_w J \\ w &= w - v \end{aligned} \tag{8}$$

where v is the current velocity vector with the same dimension as w, α is the learning rate, and $\gamma \in (0, 1)$ determines how many iterations from the previous gradients are incorporated into the current update, and $\gamma = 0.5$ is set.

In the following section, more architectures are proposed by using either various training protocols or deep nets in the fine-tuned components. In addition, the feature extraction and classification rules in each architecture are introduced as well.

4 Deep net architectures for clothing recognition

In the Big Data era, millions of data is collected from the Internet. According to Glorot et al. (2011), the training time takes up to 3 days using 1600 computing cores on the millions of data. To utilize the advantage of parallel computing in the distributed environment, compared with *Architecture 1*, three additional architectures are proposed by using either various training protocols or deep nets in the fine-tuned components to accelerate the training process for deep nets through a distributed environment. In addition, the issue of how to efficiently re-train the model without re-computing all the training data when one new class is added is regarded in the proposed architectures.

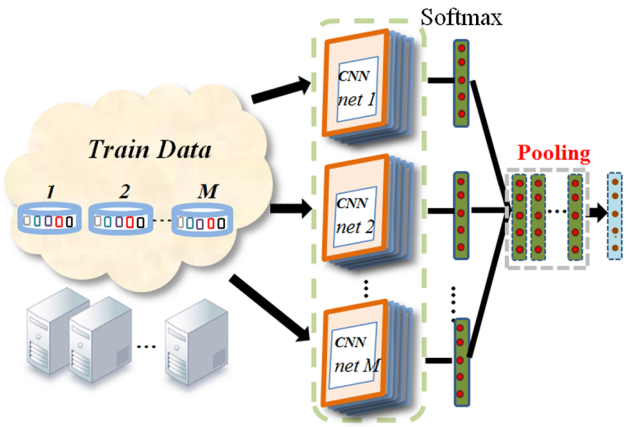

Fig. 4 Idea of *Architecture* 2 in the distributed computing environment

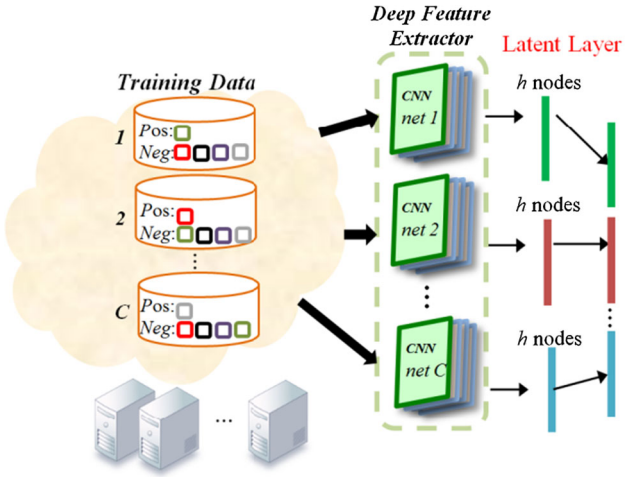

Fig. 5 Idea of *Architecture 3* with multiple nets of two softmax outputs for binary-class classification

4.1 Multiple nets for load saving

MapReduce (Dean and Ghemawat 2008) is the most famous framework for the distributed computing environment. The problem it addresses is divided into multiple independent sub-tasks, and then each sub-task can be distributed and processed in one client of cluster by the Map procedure. When each sub-task is completed, all of the sub-tasks are performed using a summary operation in the Reduce procedure. The key challenge is how to divide the problem into multiple independent sub-tasks that can be computed in parallel for each client node. For the deep net training, the input of each layer is the output of the previous layer; the relations between layers are highly dependent. Because the process is sequential, it is difficult to distribute the computation in each layer into different clients. On the contrary, the computation within a layer, i.e., convolution process, is adequate to be processed in each client. Dean (2012) developed a framework, DistBelief, which utilizes computing clusters with thousands of machines to accelerate the training process among two proposed algorithms Downpour SGD and Sandblaster L-BFGS. It is the first implementation of parallelism by partitioning large network architectures into several smaller structures, called blocks. Each block is assigned and calculated in one machine. However, boundary nodes whose edges belong to more than one partition require a data transfer between machines. Fully connected networks that have more boundary nodes demand higher communication costs resulting in performance benefits that are limited (Chen and Lin 2014).

Hence, rather than partitioning the network into smaller structures, the first modified architecture, *Architecture 2*. Inspired by the Random Forest (Bossard et al. 2013) machine learning technique, *Architecture 2* fine-tunes multiple deep nets. A large volume of training data is randomly divided into sub-data, and each sub-data is fed into one deep net that can be computed into one client node. Figure 4 demonstrates *Architecture 2* for a distributed computing environment. Note that each client node has one pre-trained model as depicted in Fig. 1, and the cross-domain transfer learning is applied with sub-data to fine-tune the corresponding deep net. Thus the low-, mid-, and high-level features can be learned in each deep net.

4.2 Multiple nets for new class and pattern mining

A problem faced in the classification task when a new class is added is that the trained model might need to be re-trained with all training data. Note that this is different from incremental learning where models are updated when more information is added to existing classes. However, it is not feasible to re-train or refine models with massive volumes of data, and even data are increasing everyday. Thus, inspired by ensemble algorithms, e.g., Adaboost (in which multiple weak models are trained and combined into the final strong one), *Architecture 3* is proposed to manage the problem of adding new classes. Figure 5 shows that *Architecture 3* consists of multiple deep nets with two outputs. This net architecture is the same as *Architecture 1*, except that there are only two softmax outputs in the last layer for the binary-class problem. Specifically, one node is for the positive class, i.e., one clothing style, and the other node is for the negative class, i.e., remaining clothing styles. In other words, each deep net can be processed using binary-class classification, and the number of deep nets in *Architecture 3* depends on the number of object classes. Hence, when one class is added, only one deep net is added and fine-tuned. Moreover, this architecture benefits from its data independence, and thus, each deep net can be fine-tuned at one client node in a distributed computing environment.

In *Architecture 4*, the training data of a negative class are reduced to investigate the performance tolerance within a

data volume. Note that the deep nets are the same as in *Architecture 3*, and the difference is that only the s part data of the negative class are trained with the positive class in each net. Here, in our simulation, $s = 1/4$. Because each net is applied to discover the discriminant features of the positive class from the remaining sets (complement set of the positive class), sufficient positive data are more important. Through the training of multiple nets, each net learns part of the discriminant features from the sub-data to save training time and storage of massive volumes of data.

4.3 Classification mechanism for deep net architectures

For clothing recognition and retrieval, different classification mechanisms are applied to obtain the final prediction results in the proposed four architectures, which can be categories into softmax layer and latent layer process.

4.3.1 Softmax layer process

For the net in *Architecture 1* and *Architecture 2*, the output of the cth node in the softmax layer, i.e. the last layer in the deep net, represents the posterior probability of the test data for the cth class. Hence, for *Architecture 1* that only one net in this architecture, the predicted label u^* for the test image I_q is assigned according to the class label which can give highest probability response. On the other hand, in order to combine the probability responses for multiple nets in *Architecture 2*, the pooling process is firstly applied to obtain the maximum probability response for each class k across all nets as

$$p(\text{Class}_k|I_q) = \max\left\{p\left(\text{Net}_k^1|I_q\right),\ldots,p\left(\text{Net}_k^m|I_q\right),\ldots,p\left(\text{Net}_k^M|I_q\right)\right\} \tag{9}$$

where $p(\text{Net}_k^m|I_q)$ is the kth node response in the softmax layer in the m deep net. Then the final predicted label k^* for the test image I_q is given by

$$k^* = \underset{k\in\{1,2,\ldots,C\}}{\arg\max}\ p(\text{Class}_k|I_q) \tag{10}$$

where C is the number of classes.

In *Architecture 3* and *Architecture 4*, multiple deep nets are consisted as well. However, the number of output nodes in the softmax layer is different from *Architecture 2*, and only two output responses where one is the posterior probability for the test image belonging to one specific class (positive class) and the other one is to other classes (negative class). Hence, the final predicted label k^* for the test image I_q is assigned by maximizing the posterior probability,

$$k^* = \underset{k\in\{1,2,\ldots,C\}}{\arg\max}\ p\left(\text{Net}_{k=1}^c|I_q\right) \tag{11}$$

where $p(\text{Net}_{c=1}^k|I_q)$ is the output response for the positive class in the kth net and C is the number of nets in the *Architecture 3* or *Architecture 4*.

4.3.2 Latent layer process

To extend the deep nets for various applications, in the proposed four architectures, the latent layer can be viewed as the feature extraction process, and traditional classification mechanism, either a nonparametric-based classifier (e.g., k-nearest-neighbor rule) or parametric-based mean vector or SVM can be applied. In *Architecture 1*, for each class c the mean vector f_c^{Mean} for the training data can be viewed as a kind of parametric model as

$$f_c^{\text{Mean}} = \frac{1}{N}\sum\nolimits_{y(I_n)\in c} L(I_n) \tag{12}$$

where $y(I_n)$ is the class label of the training data I_n, N is the number of training data belonging to class c, and $L(I_n)$ is the outputs of latent layer. Note that the length of $L(I_n)$ is 64, 128, or 256 in the proposed work. Then the predicted label k^* for the test image I_q is given by finding the minimum distance between the extracted feature vector $L(I_q)$ and f_c^{Mean} as

$$u^* = \underset{c\in\{1,\ldots,C\}}{\arg\min}\ \left\|L(I_q) - f_c^{\text{mean}}\right\|_2 \tag{13}$$

In addition, the k-NN, the nonparametric model, can be applied by comparing the distance between $\{L(I_n)\}_{n=1}^N$ and $L(I_q)$, where N is the number of training data.

On the other hand, because multiple nets are consisted in *Architecture 2*, for the training data I_n and the test image I_q the average pooling of the responses in the latent layer across all nets is first applied as

$$\tilde{L}(I_n) = \frac{1}{M}\sum_{m=1}^{M} L^m(I_n)$$
$$\tilde{L}(I_q) = \frac{1}{M}\sum_{m=1}^{M} L^m(I_q) \tag{14}$$

The mean vector $\tilde{f}_c^{\text{Mean}}$ can be obtained by

$$\tilde{f}_c^{\text{Mean}} = \frac{1}{N}\sum_{y(I_n)\in c}\tilde{L}(I_n). \tag{15}$$

Then the predicted label k^* for I_q can be estimated by the parametric model by finding the minimum distance between

the extracted feature vector $\tilde{L}(I_q)$ and $\tilde{f}_c^{\text{Mean}}$, or k-NN classification rule. Note that not only recognition, image retrieval results can be obtained as well.

However, the comparison process is not intuitive for *Architecture 3* and *Architecture 4* where the abstract meanings of the latent layer in all nets are different. In other word, each deep net in *Architecture 3* and *Architecture 4* the responses in the latent layer is the abstract properties for only one specific class. Hence, the feature vector for the training image I_n and the test image I_q is obtained by concatenating the response in each net,

$$\begin{aligned}\hat{L}(I_n) &= L^1(I_n) \oplus \cdots \oplus L^c(I_n) \oplus \cdots \oplus L^C(I_n)\\ \hat{L}(I_q) &= L^1(I_q) \oplus \cdots \oplus L^c(I_q) \oplus \cdots \oplus L^C(I_q).\end{aligned} \tag{16}$$

where $L^c(I_n)$ and $L^c(I_q)$ is the response of the latent layer in the cth net and $\oplus$ is the concatenation operation for I_n and I_q, respectively. Following that, the parametric model, i.e. the mean vector, can be obtained similar to Eq. (12) and the comparison can be applied similar to Eq. (13). Moreover, the classification results of k-NN by comparing between $\hat{L}(I_n)$ and $\hat{L}(I_q)$ can be used for recognition or retrieval applications.

5 Experimental results

In this section, we introduce the development environment and the three public clothing datasets. The performance is then compared with the existing systems in these datasets.

5.1 Development environment

Because the number of parameters need to be optimized is very large in the cNNs, the training process consumes a large amount of memory and uses much of the CPU/GPU. Among famous deep learning frameworks such as Caffe (Jia et al. 2014), cuda-convnet (Krizhevsky 2012), Decaf (Donahue et al. 2013), and OverFeat (Sermanet et al. 2014); Caffe is selected because it can provide CPU/GPU computations and pre-trained models which are suitable to demonstrate learning as our problem. Caffe (Jia et al. 2014) is a framework with the code written in C++, with CUDA used for GPU computation, and well-supported bindings to Python and MATLAB for training and deploying general purpose convolutional neural networks. It is developed and maintained by the Berkeley Vision and Learning Center (BVLC) and provides assistance for large-scale applications, and research prototypes in vision and multimedia. The experiments are performed on one PC with a 3.6 GHz and 4 cores CPU, 16 GB memory, an NVidia GTX 980 graphics card with 2048 CUDA core, and 4 GB of DDR5 memory.

5.2 Dataset and evaluation metric

We conduct the experiments on three public clothing datasets. The first dataset used in the study (Chen et al. 2015a), collects 800 images of ten style classes from fashion websites and online-shopping stores with sizes ranging from 102×62 to 540×150 pixels. The second data set (Kiapour et al. 2014), contains 1893 images with five different fashion styles: bohemian, goth, hipster, pinup, and preppy. Note that all the data images can be download but more information, like the similar degree of one clothing image related to one specific style provided by Internet user clicks, is not available. The third data set is the largest and contains 80,000 images with 15 classes of clothing and 78 attributes (Bossard et al. 2013). Figure 6 shows the examples of three datasets.

In order to analyze the proposed architectures, all three datasets were used in evaluating the classification performance. In addition, the second and third datasets were used for the performance evaluation of retrieving fashion style and clothing type, respectively. When an architecture is set, the retrieval task can be performed by feeding a query image I_q into the architecture and the outputs of the latent layers are treated as the feature vector f_q. The retrieval results can be obtained by selecting the top k images from the source images with a minimal Euclidean distance given by

$$s(I_q, I_n) = \left\| f_q - f_n \right\|_2, \tag{17}$$

Fig. 6 Examples of three datasets. *First row* (*left*) examples from the dataset (Chen et al. 2015a) and (*right*) from dataset (Kiapour et al. 2014). *Second row* examples from the dataset (Bossard et al. 2013)

where f_n is the feature vector of the nth source image in the pool. Specifically, in *Architecture* 2, all source images and the query image I_q are fed into M deep nets and the average pooling of the responses in the latent layer across all the nets are applied as Eq. (14) to obtain f_n and f_q. Note that the length of f_n and f_q equal the number of nodes in the latent layer (64, 125, or 256 used in the experiments). The training data can be used as the source images and the corresponding feature vectors can thereby be extracted off-line, whereas in *Architecture 3 and Architecture* 4, f_n and f_q are obtained by concatenating the response of the latent layer in each net as Eq. (16), i.e., the length of the feature vector f_n and f_q equal the number of classes C multiplied by the number of nodes in the latent layer. After extracting the feature vector of the query image, the top k images can be retrieved by Eq. (17). To evaluate the retrieval performance, the metric of a ranking-based criterion is then applied (Lin et al. 2015; Deng et al. 2011). The precision of the top k ranked images with respect to a query image is defined as

$$\text{pre}_k = \frac{\sum_{n=1}^{k} \text{RL}(n)}{k}, \tag{18}$$

where $\text{RL}(n)$ denotes the ground truth relevance between the query image and the nth ranked image and $\text{RL}(n) \in \{0, 1\}$ with the value of 1 for the query and the nth image with the same class label; otherwise, the value is 0.

5.3 Classification and retrieval results

The related clothing style recognition studies (Bossard et al. 2013; Chen et al. 2015a; Kiapour et al. 2014), used hand-crafted low- or mid-level features such as color features of RGB and Lab, texture features of LBP, HOG, SURF, and MR8 texture response. After extracting features, machine learning techniques such as Random Forest, SVM are applied for classification. However, because feature extraction and classifier learning are independent, the relations are ignored. Deep Learning merges these two processes into one training path. To compare with Chen et al. (2015a), numerous softmax output neurons in the deep net are set to 10 which correspond to ten clothing styles in the first dataset. Also, each training/test image is resized into256 × 256 pixels to keep the aspect ratio with zero padding. The deep net is trained with 10,000 iterations with a batch size of 64. The classification results are shown in Table 1. The best accuracy achieved (Chen et al. 2015a) is 68.2% which is obtained by using the spare representation of the feature vector which concatenates Lab color, HOG, and LBP features with 96, 1025, and 512-dimensions, respectively. Conversely, the accuracy rate of *Architecture 1* is 90.0 and 91.0% using Flicker or ImageNet dataset as the pre-trained dataset, respectively, with a significant improvement larger than 20%. The performance

Table 1 Accuracy on the first clothing dataset (Chen et al. 2015a) of ten classes

Method	Accuracy
Chen et al. (2015a): low-level features + sparse coding	0.68
Architecture 1: no fine-tune learning	0.68
Architecture 1: fine-tune learning with pre-trained model on Flicker dataset	0.90
Architecture 1: fine-tune learning with pre-trained model on ImageNet dataset	0.91

of *Architecture 1* without fine-tuning is evaluated. Not only is the training time exponentially increased but only a 68% classification rate is achieved. Thus, a fine-tuning mechanism is recommended.

In order to analyze system performance with different designs for latent layer, 64, 128 and 256 nodes are used in the latent layer for each net. Table 2 shows the accuracy rate with various node numbers and classification rules for *Architecture* 3 and *Architecture* 4 on the second dataset. In *Architecture* 3 and *Architecture* 4, we can observe that the effect of node number in the latent layer is not significant for different classification rules, and for each kind of classification rule, the difference of accuracy is about 1%. In addition, the accuracy using *Architecture* 3 is overall higher

Table 2 Performance comparison on the second clothing dataset with various nodes number in the latent layer and classification rules for *Architecture* 3 and 4

Architecture	Classification rule	Node number	Accuracy
Architecture 3	Softmax layer process	64	0.76
		128	0.77
		256	0.76
	Latent layer process: *parametric mean vector*	64	0.77
		128	0.76
		256	0.77
	Latent layer process: *nonparametric k-NN rule*	64	0.76
		128	0.76
		256	0.77
Architecture 4	Softmax layer process	64	0.73
		128	0.73
		256	0.72
	Latent layer process: *parametric mean vector*	64	0.71
		128	0.71
		256	0.72
	Latent layer process: *nonparametric k-NN rule*	64	0.70
		128	0.69
		256	0.69

Table 3 Performance comparison on the second clothing dataset with Kiapour et al. (2014) and the proposed method with different architectures and classification rule

Architecture	Classification rule	Accuracy
Kiapour et al. (2014)	SVM	0.76
Architecture 1	Softmax layer process	0.67
Architecture 2	Softmax layer process	0.72
	Latent layer process: *parametric mean vector*	0.72
	Latent layer process: *nonparametric k-NN rule*	0.68
Architecture 3	Softmax layer process	0.77
	Latent layer process: *parametric mean vector*	0.77
	Latent layer process: *nonparametric k-NN rule*	0.77
Architecture 4	Softmax layer process	0.73
	Latent layer process: *parametric mean vector*	0.72
	Latent layer process: *nonparametric k-NN rule*	0.70

Note that each deep net is fine-tuned learning with pre-trained model on ImageNet dataset

Table 4 Confusion matrix on the second dataset using *Architecture 3*

	Bohemian	Goth	Hipster	Pinup	Preppy
Bohemian	82.6	0.0	9.8	3.3	4.3
Goth	2.3	76.5	16.5	1.2	3.5
Hipster	4.0	1.2	56.0	1.3	26.7
Pinup	7.9	5.3	0.0	78.9	7.9
Preppy	6.9	8.0	15.0	1.1	69.0

than *Architecture* 4. It is because that only the 1/4 part data of the negative class are trained with the positive class in each net.

Additionally, we evaluate the system performance with different deep net architectures and compare with the work (Kiapour et al. 2014) on the second dataset. Note that in Kiapour et al. (2014), a parameter δ is used to determine the percentage of the data used in classification, and δ from 0.1 to 0.5 are set where $\delta = 0.1$ represents the top 10% of the images. Images, with a strong relation to one specific style, from each category are selected and the train-test process is executed 100 times with 9:1 train to test ratio. According to these results, the best accuracy rate is about 75% with $\delta = 0.2$ and worst rate is 70% with $\delta = 0.1$. Table 3 shows our results using all data with a 4:1 train to test ratio by different architectures and classification rules. The accuracy rate using *Architecture 1* is 67%, while a 77% accuracy rate can be obtained by using the proposed modification architecture,*Architecture 3*, with a 640-dimensional

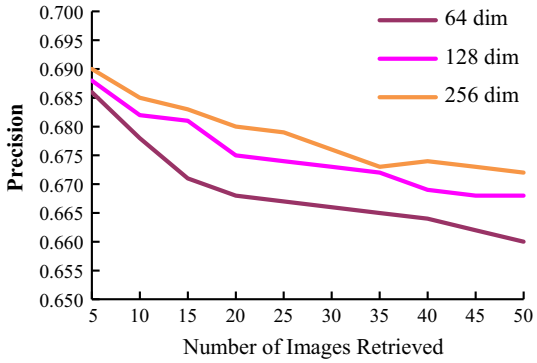

Fig. 7 Image retrieval precision of the second dataset for *Architecture 2*

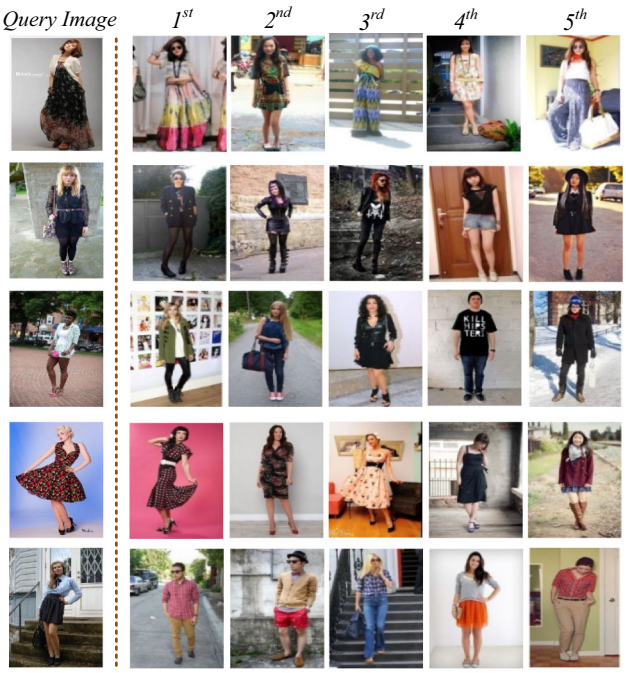

Fig. 8 Top 5 ranking images in the second datasets. From the *top* to *bottom row*, the query image is *Bohemian*, *Goth*, *Hipster*, *Pinup* and *Preppy* class

feature vector concentrated from 128 outputs in the latent layer of each net. Note that five deep nets are used in *Architecture 3*. It is hard to judge the improvement compared with the work (Khosla and Venkataraman 2015) because the number of training/test data is not the same. However, it is noted that no prior segmentation pre-process is performed in our work. Also, *Architecture 2* and *Architecture 4* test the same dataset for the simulation in the distributed computing environment. The difference between these two architectures is that both the number of training data for the positive and negative classes are reduced in *Architecture 2*, while in *Architecture 4* only the data of negative class is reduced and the number of positive data is the same as used in *Architecture 3*. It is observed that compared with

Table 5 Performance comparison on the third clothing dataset with various node numbers in the latent layer and classification rules for *Architecture* 2 and 3

Architecture	Classification rule	Node number	Accuracy
Architecture 2	Softmax layer process	64	0.55
		128	0.56
		256	0.56
	Latent layer process: *parametric mean vector*	64	0.50
		128	0.51
		256	0.52
	Latent layer process: *nonparametric k-NN rule*	64	0.54
		128	0.54
		256	0.54
Architecture 3	Softmax layer process	64	0.58
		128	0.58
		256	0.58
	Latent layer process: *parametric mean vector*	64	0.59
		128	0.58
		256	0.53
	Latent layer process: *nonparametric k-NN rule*	64	0.59
		128	0.58
		256	0.49

Table 6 Performance comparison on the third clothing dataset with Bossard et al. (2013) and the proposed method with different architectures and classification rule

Architecture	Classification Rule	Accuracy
Bossard et al. (2013)	SVM (baseline)	0.35
	Random Forest	0.41
Architecture 1	Softmax layer process	0.59
Architecture 2	Softmax layer process	0.56
	Latent layer process: *parametric mean vector*	0.52
	Latent layer process: *nonparametric k-NN rule*	0.54
Architecture 3	Softmax layer process	0.58
	Latent layer process: *parametric mean vector*	0.59
	Latent layer process: *nonparametric k-NN rule*	0.59

Architecture 1, *Architecture 2* can provide a higher accuracy rate, about 5% improvement, when classification rule of comparing the mean vector with Euclidean distance is applied. However, for *Architecture 4*, there is a 5–7% accuracy decrease compared with *Architecture 3*. It is predictable because the overfitting problem may occur when less training data for negative is used for large parameter estimation. In contrast, when sub-data are used for both positive and negative applications, data might not cause server degradation for the learning of discriminant features because multiple nets can compensate each other to obtain better classification performance. Moreover, Table 4 shows the confusion matrix using *Architecture 3*. It can be observed that Bohemian class is more easily recognized. However, hipster and preppy classes, which are more abstract and harder to be defined, result in a low recognition rate and are easily confused.

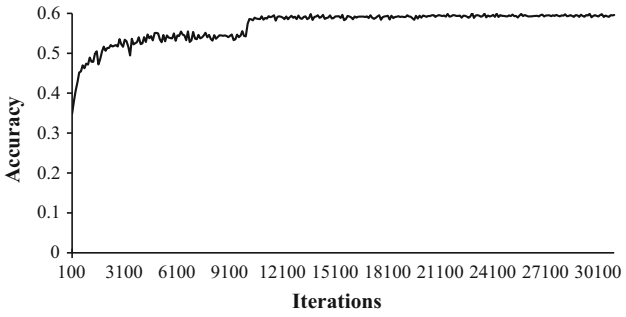

Fig. 9 The learning curve with various training iterations with *Architecture* 1

In order to investigate the retrieval performance of the fashion style, we used all the test data as the query image to obtain the precision among the top k ranking images in the second dataset. Note that we have conducted an exhaustive search for the query and database images based on the L_2-norm distance between the concatenating responses of the latent layer in each CNN net as shown in Eq. (17). Then, the precision among the top k ranking images can be obtained. Figure 7 shows the precision results of using the various lengths of latent layer for *Architecture 2*. Figure 8 shows examples of the clothing retrieval. Given a query image, the corresponding top 5 images were retrieved by *Architecture 2*.

Since the numbers of images in the first and second dataset are <2000, in order to evaluate the proposed architectures for the larger dataset, the third dataset (Bossard et al. 2013), which contains more than 80,000 clothing images with 15 classes, was used. According to the number of classes, the deep net is with 15 neurons in the output layer. Also, 100,000 iterations with 100 batches are set in the training process. Firstly, we evaluate the effects of node number for the system performance and the numbers of nodes in the latent layer were set to 64, 128, and 256. Table 5 shows the accuracy with different classification rules and node numbers in the latent layer for *Architecture* 2 and 3. We can observe that for each kind of classification rule, the difference of accuracy rates with different node numbers is not significant for the *Architecture* 2. However, for the *Architecture* 3, with increasing the node number, the accuracy rates decrease when the k-NN classification rule or the computation of the Euclidean distance with the mean vector was applied. It is because the

Table 7 Confusion matrix on the third dataset using *Architecture 2* with 128 nodes in the latent layer and the test data is classified based on the softmax layer output

	Blouses	Cloak	Coat	Jacket	Long	Sport shirt	Robe	Shirt	Short	Shit	Sweater	T-shirt	Upper	Uniform	Waistcoat
Blouses	11.6	8.4	11.6	12.0	11.6	0.9	8.0	3.1	7.6	0.4	10.2	0.4	13.9	0.4	0.0
Cloak	0.3	47.6	8.3	4.6	16.8	0.1	6.3	0.0	4.0	1.0	4.5	0.3	4.5	1.8	0.0
Coat	0.1	3.3	58.3	15.9	5.5	0.0	4.1	0.1	0.8	5.7	2.3	0.3	1.7	1.8	0.1
Jacket	0.1	3.2	14.2	52.6	4.1	0.2	2.4	0.6	1.4	11.9	4.5	0.2	1.8	2.3	0.5
Long	0.1	7.1	3.8	2.2	64.1	0.2	6.3	0.1	5.5	2.0	1.5	0.1	5.5	1.3	0.2
Sport shirt	0.0	6.1	6.6	8.7	5.1	36.2	4.6	2.6	1.5	3.1	5.6	13.8	2.6	3.1	0.5
Robe	0.1	7.6	9.6	5.4	13.5	0.0	49.9	0.1	2.0	2.8	2.8	0.8	2.8	2.8	0.0
Shirt	1.7	3.6	9.7	17.8	5.3	3.1	5.0	33.6	1.9	3.6	5.6	3.9	2.8	2.2	0.3
Short	0.5	5.7	6.2	3.5	25.9	0.0	3.3	0.4	39.1	0.7	2.5	0.3	11.4	0.6	0.1
Shit	0.0	2.6	6.3	8.2	3.1	0.1	1.9	0.3	0.3	72.1	1.1	0.2	1.9	1.8	0.3
Sweater	0.2	6.5	5.8	8.6	4.8	0.5	3.4	0.3	1.2	2.1	59.3	1.5	4.4	1.7	0.0
T-shirt	0.6	5.6	5.6	5.3	6.7	4.2	5.6	2.8	1.9	1.1	7.5	44.7	7.5	1.1	0.0
Upper	0.1	3.5	2.7	2.9	10.2	0.0	3.4	0.1	4.0	1.2	2.5	1.1	67.2	1.1	0.1
Uniform	0.0	3.6	9.0	7.5	3.0	0.4	4.9	0.5	0.8	4.6	2.1	0.8	2.3	60.3	0.2
Waistcoat	1.1	4.7	11.0	20.9	5.2	0.0	4.2	0.5	2.1	11.0	4.2	0.0	5.8	4.7	24.6

Table 8 Confusion matrix on the third dataset using *Architecture 3* with 128 nodes in the latent layer and the test data is classified based on the softmax layer output

	Blouses	Cloak	Coat	Jacket	Long	Sport shirt	Robe	Shirt	Short	Shit	Sweater	T-shirt	Upper	Uniform	Waistcoat
Blouses	25.3	8.0	6.2	8.4	9.8	1.3	5.3	4.4	9.8	0.9	8.4	0.9	9.8	0.9	0.4
Cloak	0.9	54.5	6.1	4.6	12.0	0.3	5.8	0.6	4.4	0.9	4.5	0.45	3.1	1.5	0.5
Coat	0.4	3.9	59.5	14.7	3.8	0.0	4.1	0.4	1.5	5.4	2.1	0.3	1.3	1.9	0.7
Jacket	0.8	3.4	14.1	53.1	3.1	0.2	2.6	0.9	1.9	11.1	4.0	0.4	1.6	1.8	1.2
Long	0.4	9.3	3.7	2.4	61.1	0.1	6.5	0.2	6.7	1.8	1.2	0.3	5.1	0.6	0.4
Sport shirt	1.0	3.6	2.6	5.6	3.6	46.9	4.6	3.1	2.0	2.0	6.6	13.8	2.0	2.0	0.5
Robe	0.6	9.2	7.0	5.2	10.8	0.6	54.3	0.6	1.9	2.3	2.3	0.8	2.4	2.3	0.3
Shirt	3.9	3.9	7.2	11.7	3.9	3.6	2.5	45.6	2.8	2.5	2.8	5.3	2.8	1.7	0.0
Short	1.1	7.1	4.5	4.2	20.6	0.1	3.0	0.7	43.8	1.5	2.1	0.7	9.4	1.0	0.4
Shit	0.1	2.2	5.8	7.8	4.1	0.2	2.7	0.5	0.8	71.3	0.3	0.1	1.5	2.0	0.6
Sweater	0.5	7.4	4.8	8.3	3.0	0.3	3.6	0.9	1.0	1.5	63.6	1.6	2.7	0.7	0.2
T-shirt	1.1	5.6	2.5	3.9	4.4	3.3	5.0	5.0	1.9	1.7	8.3	50.0	5.3	1.4	0.6
Upper	0.4	4.7	1.4	2.7	9.0	0.3	3.1	0.3	4.8	1.0	1.6	0.9	68.6	0.8	0.4
Uniform	0.2	3.2	7.5	7.5	2.0	0.6	4.5	1.0	0.7	3.8	1.3	0.6	1.5	65.3	0.4
Waistcoat	0.0	6.8	7.9	18.9	4.7	0.5	1.6	0.5	1.6	7.3	2.6	0.5	6.8	3.1	37.2

variations for each class in the third dataset are larger than the first and the second dataset, and the risk of overfitting occurrence is higher when the larger length of feature vectors was used. Note that for *Architecture* 3, the length of feature vector as shown in Eq. (16) is the number of classes multiplied the number of nodes in the latent layer. In other words, the length is 1920 and 3840 for the node number 128 and 256, respectively. Hence, the classification rule based on the softmax layer is recommended for *Architecture* 3.

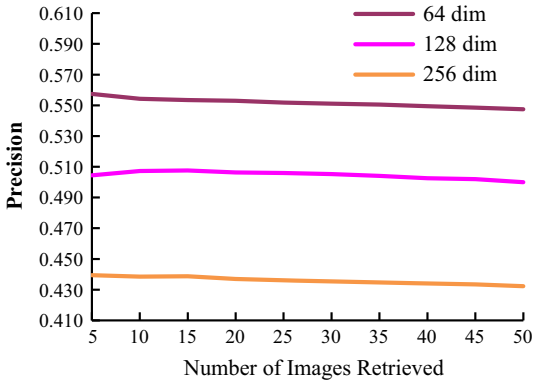

Fig. 10 Image retrieval precision of the third dataset for *Architecture 3*

Table 6 shows the comparison with the study (Bossard et al. 2013), and the accuracy rate with the best parameter, i.e. node number, was listed. Note that each deep net is fine-tuned with pre-trained model on ImageNet dataset. In Bossard et al. (2013), the authors applied Random Forest to be capable of transfer learning from different domains and provide a 41.38% average accuracy higher than 35.07% with SVM baseline. The deep net with *Architecture* 1 can provide accuracy of 59.5%, with an 18% improvement than (Bossard et al. 2013). Figure 9 shows the learning curve with various training iterations with *Architecture* 1. It can be observed that the system is stable after 30,000 iterations. Moreover, Tables 7 and 8 shows the confusion matrices using*Architecture 2* and *Architecture 3* with 128 nodes in the latent layer and the test data is classified based on the softmax layer output, respectively. From the classification results, it can be observed that the classification results for blouses and waistcoat class are worse and easily confused with other classes in both cases. The classification results for the shirt and upper class are better. Additionally, the number of classes that have at least 50% classification rate is 8 and 10 in Tables 7 and 8, respectively. The result is consistent that *Architecture* 3 has higher classification rate than *Architecture* 2 in Table 5.

In addition, we randomly selected 2000 images as the query image to obtain the precision of the top k ranking images in the third dataset. The experimental settings which include the distance measurement and feature vectors were the same as the settings in the second dataset. Figure 10 shows the precision results of using the various length of the latent layer for *Architecture 3*, and Fig. 11 shows the examples of clothing retrieval for each class. Given a query image, the corresponding top five images were retrieved.

Fig. 11 Top 5 ranking images in the third datasets. From the *top* to *bottom row*, the query image is *blouses*, *cloak*, *coat*, *jacket*, *long*, *sport shirt*, *robe*, *shirt*, *short*, *suit*, *sweater*, *t-shirt*, *upper*, *uniform* and *waistcoat class*

6 Conclusion

Inspired by the robust classification ability of Deep Learning and its ability to analyze a massive volume of data in the Big Data era, we applied the deep net theory to clothing style recognition and proposed three modifications of deep architectures for the distributed computing environment. Compared with the existing approaches that use hand-crafted low-level features and machine learning algorithms with shallow structure, the proposed method provided more promising results on the three public clothing datasets, particularly on the large dataset with 80,000 images where an improvement of 18% in accuracy was recognized. Also, the proposed architectures, *Architecture 2* and *Architecture 4*, using sub-training data either on positive or negative classes might not cause server accuracy degradation. Thus these architectures could provide one feasible way to apply the Deep Learning tool for Big Data Analytics in the distributed computing environment.

Acknowledgements This work is supported by National Science Council (NSC), Taiwan, under Contract of MOST 104-2221-E-151-028. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

Compliance with ethical standards

Conflict of interest The authors declare that they have no competing interests.

References

Arel I, Rose DC, Karnowski TP (2010) Deep machine learning—a new frontier in artificial intelligence research. IEEE Comput Intell Mag 5(4):13–18

Bengio Y, Courville A, Vincent P (2013) Representation learning: a review and new perspectives. IEEE Trans Pattern Recog Mach Intell 35(8):1798–1828

Bengio Y, Lamblin P, Popovici D, Larochelle H (2007) Greedy layer-wise training of deep networks. In: International conference on neural information processing systems, pp 153–160

Bossard L, Dantone M, Leistner C, Wengert C, Quack T, Gool LV (2013) Apparel classification with style. In: Asia conference on computer vision, vol 7727, pp 321–335

Chen XW, Lin X (2014) Big data deep learning: challenges and perspectives. IEEE Access 2:514–525

Chen JC, Liu CF (2015) Visual-based deep learning for clothing from large database. In: ASE BigData & SocialInformatcis

Chen JC, Xue BF, Lin Kawuu W (2015a) Dictionary learning for discovering visual elements of fashion styles. In: CEC workshop

Chen Q, Huang J, Feris R, Brown LM, Dong J, Yan S (2015b) Deep domain adaptation for describing people based on fine-grained clothing attributes. In: IEEE conference on computer vision and pattern recognition, pp 5315–5324

Ciresan DC, Meier U, Gambardella LM, Schmidhuber J (2010) Deep big simple neural nets excel on handwritten digit recognition. Neural Comput 22(12):3207–3220

Dahl GE, Yu D, Deng L, Acero A (2012) Context-dependent pretrained deep neural networks for large-vocabulary speech recognition. IEEE Trans Audio Speech Lang Process 20(1):30–41

Dean J (2012) Large scale distributed deep networks. In: International conference on neural information processing systems, pp 1232–1240

Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. ACM Mag 51(1):107–113

Deng J, Berg AC, Li FF (2011) Hierarchical semantic indexing for large scale image retrieval. In: IEEE conference on computer vision and pattern recognition, pp 785–792

Di W, Wah C, Bhardwaj A, Piramuthu R, Sundaresan N (2013) Style finder: fine-grained clothing style recognition and retrieval. In: IEEE conference on computer vision and pattern recognition workshops, pp 8–13

Donahue J, Jia Y, Vinyals O, Hoffman J, Zhang N, Tzeng E, Darrell T (2013) DeCAF: a deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531

Efrati A (2013) How deep learning works at Apple. Information. https://www.theinformation.com/How-Deep-Learning-Works-at-Apple-Beyond

Farabet C, Couprie C, Najman L, LeCun Y (2013) Learning hierarchical features for scene labeling. IEEE Trans Pattern Recog Mach Intell 35(8)

Gantz J, Reinsel D (2011) Extracting value from chaos. IDC iView. https://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf

Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: IEEE conference on computer vision and pattern recognition, pp 580–587

Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier networks. In: International conference on artificial intelligence and statistics, pp 315–323

Goodfellow IJ, Warde-Farley D, Mirza M, Courville A, Bengio Y (2013) Maxout networks. arXiv preprint. arXiv:1302.4389

Hinton G, Osindero S (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554

Hinton G, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov R (2012) Improving neural networks by preventing coadaptation of feature detectors. arXiv:1207.0508

Huang J, Feris RS, Chen Q, Yan S (2015) Cross-domain image retrieval with a dual attribute-aware ranking network. arXiv preprint arXiv:1505.07922

Jagadeesh V, Piramuthu R, Bhardwaj A, Di W, Sundaresan N (2014) Large scale visual recommendations from street fashion images. In: ACM SIGKDD International conference on knowledge discovery and data mining, pp 1925–1934

Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Caffe DT (2014) Caffe: convolutional architecture for fast feature embedding. In: International conference on multimedia, pp 675–678

Jones N (2014) Computer science: the learning machines. Nature 505(7482):146–148

Kalantidis Y, Kennedy L, Li LJ (2013) Getting the look: clothing recognition and segmentation for automatic product suggestions in everyday photos. In: ACM international conference in multimedia retrieval, pp 105–112

Khosla N, Venkataraman V (2015) Building image-based shoe search using convolutional neural networks. In: CS231n course project reports

Kiapour MH, Yamaguchi K, Berg AC, Berg TL (2014) Hipster wars: discovering elements of fashion styles. In: European conference on computer vision, pp 472–488

Krizhevsky A (2012) Cuda-convnet. https://code.google.com/p/cuda-convnet/

Krizhevsky A, Sutskever I, Hinton G (2012) Imagenet classification with deep convolutional neural networks. In: International conference on neural information processing systems, pp 1106–1114

Le QV, Zou WY, Yeung SY, Ng AY (2011) Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In: IEEE conference on computer vision and pattern recognition, pp 3361–3368

Le Q, Ranzato M, Monga R, Devin M, Chen K, Corrado G, Dean J, Ng A (2012) Building high-level features using large scale unsupervised learning. In: International conference on machine learning, pp 81–88

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. IEEE Proc 86(11):2278–2324

Lin M, Chen Q, Yan S (2013) Network in network. In: International conference on learning representations. arXiv:1312.4400

Lin K, Yang HF, Liu KH, Hsiao JH, Chen CS (2015) Rapid clothing retrieval via deep learning of binary codes and hierarchical search. In: ACM international conference in multimedia retrieval, pp 499–502

Liu C, Yuen J, Torralba A (2011) Nonparametric scene parsing via label transfer. IEEE Trans Pattern Recog Mach Intell 33(12):2368–2382

Liu S, Feng J, Song Z, Zhang T, Lu H, Xu C, Yan S (2012) Hi, magic closet, tell me what to wear! In: International conference on multimedia, pp 619–628

Liu S, Feng J, Domokos C, Xu H, Huang J, Hu Z, Yan S (2014) Fashion parsing with weak color-category labels. IEEE Trans Multimedia 16(1):253–265

Liu S, Liang X, Liu L, Shen X, Yang J, Xu C, Lin L, Cao X, Yan S (2015) Matching-CNN meets KNN: quasi-parametric human parsing. arXiv:1504.01220

Long J, Zhang N, Darrell T (2014) Do convnets learn correspondence. In: International conference on neural information processing systems, pp 1601–1609

Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91110

Mohamed A, Dahl G, Hinton G (2012) Acoustic modeling using deep belief networks. IEEE Trans Audio Speech Lang Process 20(1):14–22

Najafabadi MM, Villanustre F, Khoshgoftaar TM, Seliya N, Wald R, Muharemagic E (2015) Deep learning applications and challenges in big data analytics. J Big Data 2(1):1–21

Ojala T, Pietikainen M, Maenpaa T (2002) Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. TPAMI 24(7):971–987

Oquab M, Bottou L, Laptev I, Sivic J (2014) Learning and transferring mid-level image representations using convolutional neural networks. In: IEEE conference on computer vision and pattern recognition, pp 1717–1724

Razavian AS, Azizpour H, Sullivan J, Carlsson S (2014) CNN features off-the-shelf: an astounding baseline for recognition. In: IEEE conference on computer vision and pattern recognition workshops, pp 512–519

Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun Y (2014) Overfeat: integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229

Socher R, Huang EH, Pennington J, Ng AY, Manning CD (2011a) Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In: International conference on neural information processing systems, pp 801–809

Socher R, Lin C, Ng A (2011b) Parsing natural scenes and natural language with recursive neural Networks. In: International conference on machine learning, pp 129–136

Song Z, Wang, Hua MX, Yan S (2011) Predicting occupation via human clothing and contexts. In: International conference on computer vision, pp 1084–1091

Sukumar SR (2014) Machine learning in the big data era: are we there yet? In: ACM SIGKDD conference on knowledge discovery and data mining: workshop on data science for social good

Sun Y, Wang X, Tang X (2015) Deeply learned face representations are sparse, selective, and robust. In: IEEE conference on computer vision and pattern recognition. arXiv:1412.1265

Tung F, Little JJ (2014) Collage parsing: nonparametric scene parsing by adaptive overlapping windows. ECCV 8694:511–5252

Wang Y, Yu D, Ju Y, Acero A (2011) Voice search. In: Language understanding: systems for extracting semantic information from speech, pp 119–146

Yamaguchi K, Kiapour MH, Ortiz LE, Berg TL (2012) Parsing clothing in fashion photographs. In: IEEE conference on computer vision and pattern recognition, pp 3570–3577

Yamaguchi K, Kiapour MH, Berg TL (2013) Paper doll parsing: retrieving similar styles to parse clothing items. In: International conference on computer vision, pp 3519–3526

Yamaguchi K, Berg TL, Ortiz LE (2014) Chic or social: visual popularity analysis in online fashion networks. In: ACM conference on multimedia, pp 773–776

Yang W, Luo P, Lin L (2014) Clothing co-parsing by joint image segmentation and labeling. In: IEEE conference on computer vision and pattern recognition, pp 3182–3189

Zhang N, Paluri M, Ranzato M, Darrell T, Bourdev L (2014) PANDA: pose aligned networks for deep attribute modeling. In: IEEE conference on computer vision and pattern recognition, pp 1637–1644