DeepSketch: Deep convolutional neural networks for sketch recognition and similarity search

Conference Paper · June 2015

DOI: 10.1109/CBMI.2015.7153606

3 authors:

Omar Seddati
Université de Mons
21 PUBLICATIONS **56** CITATIONS
SEE PROFILE

Stéphane Dupont
Université de Mons
155 PUBLICATIONS **2,051** CITATIONS
SEE PROFILE

Mahmoudi Saïd
Université de Mons
112 PUBLICATIONS **473** CITATIONS
SEE PROFILE

DeepSketch: Deep Convolutional Neural Networks for Sketch Recognition and Similarity Search

Omar Seddati
UMONS
Computer Science - TCTS Lab
Mons, Belgium
omar.seddati@umons.ac.be

Stéphane Dupont
UMONS
TCTS Lab
Mons, Belgium
stephane.dupont@umons.ac.be

Saïd Mahmoudi
UMONS
Computer Science Department
Mons, Belgium
said.mahmoudi@umons.ac.be

***Abstract*—In this paper, we present a system for sketch classification and similarity search. We used deep convolution neural networks (ConvNets), state of the art in the field of image recognition. They enable both classification and medium/high-level features extraction. We make use of ConvNets features as a basis for similarity search using k-Nearest Neighbors (kNN). Evaluation are performed on the TU-Berlin benchmark. Our main contributions are threefold: first, we use ConvNets in contrast to most previous approaches based essentially on hand crafted features. Secondly, we propose a ConvNet that is both more accurate and lighter/faster than the two only previous attempts at making use of ConvNets for handsketch recognition. We reached an accuracy of 75.42%. Third, we shown that similarly to their application on natural images, ConvNets allow the extraction of medium-level and high-level features (depending on the depth) which can be used for similarity search.[1]**

Keywords— Freehand sketch; DNN; ConvNets; Feature extraction; Sketch recognition

I. INTRODUCTION

Freehand sketches are commonly used by humans. They are a simple and powerful tool for communication. They are easily recognized across cultures and can be used to both describe static and dynamic information. As a consequence, sketch recognition starts to attract more and more interest in the research community.

Rencently, Deep Neural Networks (DNNs) have significantly improved performance in the field of image recognition. In contrast, sketch recognition state-of-the-art still relies mainly on hand crafted features and more traditional classification schemes. DNNs have not yet been effectively explored. One challenge comes from the fact that databases available for machine learning are very limited (in comparison to natural image recognition benchmarks), which may tame down the benefit from DNNs, which typically require very large data sets. Also, DNNs feature extractors trained on natural image database (such as ImageNet) are not suitable for use with sketches (that are indeed very different, containing no color or texture information).

In this work, we used deep Convolutional Networks (ConvNets), as particular structure if DNN. We first started by testing an architecture similar to the so-called AlexNet [13]. We trained our ConvNets from scratch on the TU-Berlin sketch recognition benchmark. We also tested the architecture proposed in [16], a modified version of the previous architecture to make it more suitable for sketches. Then, we compared and analysed the results from both, and proposed a new architecture that yields better results. Next, we also use this architecture to perform feature extraction from the sketch images. We show that a kNN method applied to these features is an appropriate approach for similarity search. We illustrate and discuss how features extracted from different layers (different depths) of the DNNs enable different facets of similarity (from more graphical from lower depth layers, to more semantic from higher depth layers) to be explored.

II. RELATED WORK

In this section, we review the work in the field of sketch recognition. Then, we briefly present some recent researches showing the power of ConvNets, as well as the two first attempts (to our knowledge) where ConvNets were used for sketch recognition.

Works on sketch recognition goes back to the development of SketchPad [5]. Ever since, different computer vision approaches were used in order to achieve better results in multiple domains of application. LaViola et al. [7] investigated recognition of mathematical sketches. Ouyang et al. [8] addressed chemical structure drawings recognition. Cao et al. [1] proposed a low dimensional symmetry aware flip invariant descriptor for sketches. Li et al. [3] exploited local feature representations (using star graph based ensemble matching strategy) and global structures to address local and global

[1] This work was partly supported by the Chist-Era project IMOTION with contribution from the Belgian Fonds de la Recherche Scientique (FNRS), contract no. R.50.02.14.F.

978-1-4673-6870-4/15/$31.00 ©2015 European Union

variations. They trained SVM using bag-of-features (BOF) to select the top *N* most similar sketch categories to the query.

In addition to the sparse and abstract nature of hand drawings (compared to natural images), there is also another challenge related to the lack of available databases for model training and benchmarking. Eitz et al. [2] defined a taxonomy of 250 object categories. They gathered 20000 unique sketches to form the first large scale dataset of human sketches (TU-Berlin benchmark). For sketch category recognition, they used local feature vectors (to encode distributions of sketches), bag-of-features representations [6], and multi-class support vector machines (SVM). Schneider et al. [4] modified the benchmark proposed by Eitz et al. [2] to make it more focused relevant aspect (what it look like) than person intention. Later on, they used SIFT, GMM based fisher vector and multi-class SVM to do sketch recognition.

In the related area of natural images computer vision, supervised learning with ConvNets achieved beyond state-of-the-art results [13, 14]. In general, ConvNets enable classification without requiring classic features extraction approaches, as the model trains itself to extract relevant features from pixels. So far, they were used only two times (in 2015) for sketch recognition. Sarvadevabhatla et al. [15] used two popular ConvNets architectures (ImageNet ConvNet and a modified LeNet ConvNet) and fine-tuned their parameters on the sketch database. They used a subset of TU-Berlin sketch benchmark by retaining only 160 categories and 56 sketches from each category (as done in [4]). In contrast, Yang et al. [16] trained a ConvNet from scratch instead of fine tuning existing ones. They show that the receptive fields of the first layer ConvNet filters had to be increased (compared to typical settings for natural images) to improve generalization performance.

III. Approach

Our work relies on the TU-Berlin benchmark too. Our system (illustrated in Fig. 1) uses a ConvNet for classification but also for features extraction and similarity search: the extracted features are used with kNN to find sketches that resemble to the query. To better understand the properties of those features, we use features extracted from three different layers (depth) of our ConvNets. When using the system, the user can draw a sketch in a query-by-example (QbE) mode. The ConvNet computes the features from the query and kNN searches for similar sketches in the database. Similarity is a ill-defined concept. In practice, a user may look for images (or in our case sketches) that are perceptually more similar, or else semantically/conceptually more similar (which in contrast may imply neighbors that are visually quite different in terms of color and textures). We believe DNNs trained in a supervised way provide an interesting approach to explore different facets of similarity, from lower-level (perceptual) through the features extracted from lower layers, to high-level (semantic) through features extracted in deeper layers, which have been shown indeed to compute more invariant features. In this paper, we want to illustrate and discuss how features extracted from different layers emphasized different facets of sketch similarity.

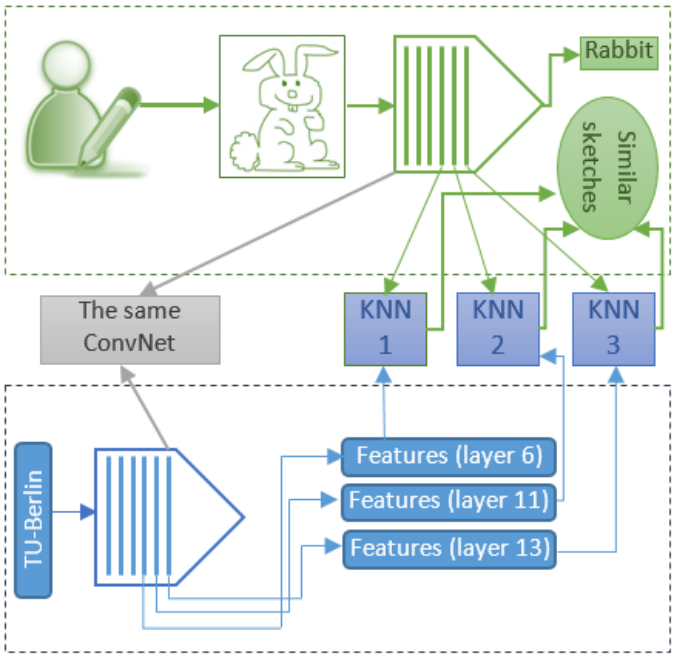

Figure 1: our system for classification and similarity search

IV. Model

In the area of natural image classification (object and scene recognition), and when ConvNets are used, most of the time, the baseline architecture starts from earlier work such as AlexNet [13]. Besides, visualization techniques of the filters towards which hidden layers converge is an interesting diagnostic tool. As the use of ConvNets for processing sketch images is quite recent [15, 16], there is however no previous work on trying to optimize the baseline architecture.

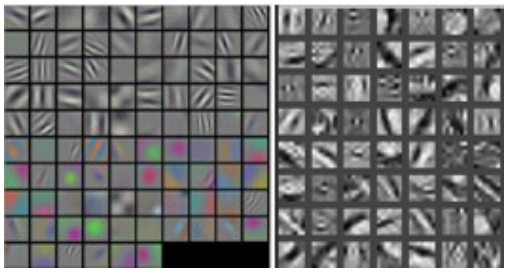

Figure 2: filters from the first layer (left, AlexNet. Right, our ConvNet).

In order to select a good design for our ConvNets, we started looking more closely at the two previous approaches [15, 16]. In [15], the authors retrained two different ConvNet models. The first choice was AlexNet (initially trained on ImageNet) because of the good results it achieved on different image databases. The second choice was LeNet (initially trained on MNIST) because of the similarity in the nature of the content to be processed (MNIST containing handwritten digits). They used those as feature extractors, coupled with a SVN trained to classify the sketches from the obtained features. Then they followed the same protocol as Schneider [4], enabling a comparative evaluation on the TU-Berlin corpus. In [16], the authors used similar architectures but they trained the

ConvNets on TU-Berlin data from scratch. Sketches are very different in nature that natural images. Therefore, there is small chance that ConvNet filters learned from natural images will be effective for sketches. They found on one side, that using filters with a larger receptive field (larger filter kernels) in the first layer is necessary for sketch classification, and on the other side that filters look significantly different that similarly trained filters on natural images, (in Fig. 2 we picture a similar observation from our models, showing the filters from the first layer of AlexNet, and those from one of our ConvNets trained on sketches from scratch). Another observation was made in [16]: contrary to what is expected, using more filters leads to better performance, despite the more abstract nature of sketches.

Studying both works however left us undecided about the choice of the proper architecture to start from, as indeed some contradictions can be observed. For example, AlexNet gave better results than LeNet even if the nature of sketches and MNIST images is closer. Also, the importance of using larger filters is stressed in [16], while [15] achieves interesting results even when using small filter kernels. We believe those contradictions are inherent to the preliminary nature of such works.

Finally, we decided to further test several variations of the proposed architectures (the one from [16] and the AlexNet). We however trained both from scratch. From our various trials, which results can only be summarized here given space constraints, we noticed the following:

- As expected, the filters are very different from what we find when training is done on natural images;
- The use of more and bigger filters in the first layer was an interesting idea, especially since there is an analogy to be made between what happens when we do so, and what happens inside the ConvNets when trained on natural images. We know that many filters from the first layer of ConvNets typically act as edge detectors. Hence, they compute feature maps (FMs) with similar properties than sketches. These FMs are numerous and pass through a layer of Max-pooling before reaching the next layer, makes the receptive of the next layer virtually larger. So, we also tried an alternative to the use of big kernels by reducing the size of the input, but the results have become worse. This is probably due to the abstraction of this kind of representation that makes the depth of the network essential for sketch recognition. However, after testing both approaches, the results were not conclusive enough to justify the use of larger filters. We also observed in our tests, this large filters still similar to the small ones learned with AlexNet (trained on sketches);

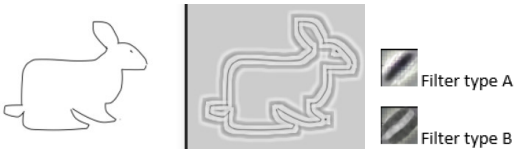

Figure 3: example shown the impact of using LRN with sketches

- Using larger kernels just before the classification (latest) layers of the ConvNets improves results. The reason might be that unlike natural images where we have additional information (like color, texture, etc.) that make object identification less abstract, in the case of sketches, in order to achieve accurate recognition, we need accurate identification of larger structures specific to each category. A good understanding of spatial arrangements can be achieved using those larger kernels;
- Using zero-padding of the original images helps to improve the results, which is understandable because it helps to preserve the information far from the center when multiple convolutions are performed. Also, unlike images, sketches are sparse, and filling unused areas with zeroes (white color as colors are reversed before training, cfr. Section V) has less dramatic consequences on the output;
- In [16], it was reported that Local Response Normalization (LRN) is not useful in the case of sketches. We believe the reason is that applying LRN to sketches results in some form of contour dilation, a "shadow" effect illustrated in Fig. 3. On sketches, this only creates redundancy. The shadow will result in the activation of neurons of *type B* in both sides of the original contour, while the contour itself will produce activations of neurons of *type A* for the same information. This will make the learning of the first layer filters harder (requiring more filters and hence more parameters), also affecting the sparsity of the network.

As a consequence, compared to previous works, we ended up using small and fewer kernels in the first layer. Firstly, it is understandable we don't need as much filters as for images which naturally present more varied content. Secondly, smaller kernels are appropriate as sketches typically exhibit sharp edges. We also use zero-padding before multiple convolution layers in such a way the size of the feature maps goes down only because of the max-pooling layer that follows. Finally, we use larger kernels in the latest convolution layers, those right before the classification layers of our ConvNets. We also used fewer filters in the last layers because for sketch recognition, what is needed is more powerful and abstract features extractors rather than more features for the classifiers. This also goes well with the plan to use these features for sketch retrieval using kNN.

V. IMPLEMENTATION DETAILS

Our system is based on the use of the publicly available Torch toolbox [17]. The details for the used architecture are listed in Table 1. We also used the kNN implementation from OpenCV [18].

Training - We start by rescaling the sketches from 1x1111x1111 to 1x180x180 and reversing the black and white colors. Then we remove the global mean and store all the sketches in RAM for fast access, training being performed on GPU. During training, at each iteration we randomly select 64 samples from 64 different sketches categories. Next, we create

64 matrixes of size equal to 1x224x224 filled with mines the global mean and add the mean-normalized sketches at random positions inside those matrixes, in order to improve invariance to translation. Later, we perform random rotations [-35:5:35] and mirroring. We also use the dropout technique to further favor generalization.

Table 1: details of our ConvNets parameters, for each of the 15 layers.

Ind	**Type**	**Filter**	**Filter**	**Stride**	**Pad**
1	Conv	7x7	64	2	0
2	ReLU	-	-	-	-
3	Maxpool	3x3	-	2	0
4	Conv	5x5	128	2	2
5	ReLU	-	-	-	-
6	Maxpool	3x3	-	2	-
7	Conv	3x3	256	1	1
8	ReLU	-	-	-	-
9	Conv	3x3	512	1	0
10	ReLU	-	-	-	-
11	Maxpool	3x3	-	2	-
12	Conv	5x5	4096	1	0
13	ReLU	-	-	-	-
14	Dropout	-	-	-	-
15	Conv	1x1	250	1	0

At the start, we set the learning rate at 0.1 and a momentum equal to 0.9. After 80 epochs (epoch = 13056 samples were presented to the ConvNet) we start reducing gradually those values in order to catch more details.

Testing - During testing, for each sketch we create ten 1x224x224 matrices filled with the global mean. We then generate ten variants of the sketch (by placing the original one in the center and on the four corners + mirroring). All these are passed them through our ConvNet that will estimate for each sketch the probability of each object class. Next, we compute the final vector of class probabilities as a product of the ten different estimates. Finally, we hence obtain a vector with 250 probabilities (one per class) and the winner class is the one that has the highest probability. This process of creating variants of the test images and "averaging" probability estimates is typical when using ConvNets.

VI. EXPERIMENTS AND EVALUATION

In this section, we summarize our results. As already mentioned, we used the TU-Berlin sketch benchmark. It contains 250 classes and for each class we have 80 instances (binary images). In total, this amounts to 20,000 sketches with resolution: 1x1111x1111.

A. Classification

As in [16], we followed the guidelines suggested by [2]. We prepared three different splits. In each split we use 67% of the original data (53 sketches) for training and the 33% remaining (27 sketches) for test. Three ConvNets were trained separately on the three splits, using the exact same architecture as presented in the previous section. The classification results are presented in Table 2. In table 4, we compare our results with those of previous state of the art. We have been able to improve the classification accuracy by more than 3% absolute compared to the best previously published result.

Table 2: our results for the three splits of TU-Berlin benchmark

	Accuracy (%)
ConvNet1 + split1	77.25
ConvNet2 + split2	74.19
ConvNet3 + split3	74.80
mean	75.42

B. Similarity Search

We then switched to a subjective analysis of the interest of features extracted with the ConvNets for sketch similarity search. We began by using bottlenecks to reduce dimensionality of features coming from different layers. This means extracting features from the selected three layers (6, 11 and 13). To train one layer (for each of the three layers) with N_l inputs (N_l = number of features in the output of layer l) and n outputs. In our case we set n to 1024. Once the training is done, we use these layers to build shorter vectors and pass them to the kNN system.

In Figure 4, we illustrate the results obtained with our system with an example. We notice that applying kNN with features from different layers leads to sensibly different and interesting results. Searching similar sketches using kNN applied on features from layer 6 leads to retrieved sketches exhibit local structures that are similar to the example query, suggesting that layer mainly captures local details, such as some patterns found on the body of the sketched zebra. This hence leads to a lot of confusion with sketches from other object classes but having those patterns in common. When using features from layer 11, the global shape of the animal's body starts to become more predominant, and combined with the previous features (layer 6), the retrieved results become more relevant. Finally, with layer 13, the resemblance is no longer based only on perceptual criteria such as the global shaper and the local hand-drawn patterns and it is clear that a more conceptual (semantic) feature representation has been learned. Hence, searching neighbors of the query sketch with kNN applied to layer 13 features retrieves not only similar sketches but also sketches representing the same concept. For example, a zebra head is retrieved as the third best match even though it looks visually quite different. As can also be observed, when using those higher level features, there is only one retrieved sketch that does not represent an animal with a striped coat out 20 results (while with layer 11 we had 3, and layer 6, we had 18). We have done other tests that lead to the same conclusions. For example, we used kNN and majority vote (MV) method with the test sketches of the first split of the TU-Berlin sketch benchmark and we obtained results that confirm what has been noticed. In addition, we have the classification results obtained with the bottlenecks. Finally, to avoid the effect of adding a layer when using the bottleneck approach. We used principal component analysis (PCA) to get shorter vectors (1024 features) to use with kNN and majority vote. All results are reported in the table 3. These results show an interesting analogy with hierarchies of feature

representations built when training ConvNets on natural images.

Table 3 accuracy comparison using different approaches

	Accuracy (%)		
Method	Layer 6	Layer 11	Layer 13
Bottleneck + kNN + MV	57.78	70.4	73.18
Bottleneck + classification	62.7	73.88	75.18
PCA + kNN + MV	2.37	5.53	70.41

In Figure 5, there is an example that shows the benefits of using features coming from first layers. It is clear that the closest result to the query is the one with the blue box. We can see that using deeper features makes the closest result appear further away even if the concept remains good. Further work will include objective as well as user studies to assess how a QbE-based sketch search system can benefit from these properties.

Table 4: results comparison

Method	Feature	mAP (%)
[2]	SIFT-variant + BoF + SVM	56
[16]	Deep neural networks	72.2
Ours	Deep neural networks	75.42
Human	-	73

VII. CONCLUSION AND FUTURE WORK

In this paper, we first reviewed approaches that have been introduced recently in the field of sketch recognition using ConvNets. Then, we proposed a ConvNet architecture that is more accurate in terms of classification rate, and also less expensive (fewer and smaller kernels) than previous attempts. We also provided rationales for the various changes that we made in relation to what has been done before. Then, we showed for the first time (in the field of sketches) that ConvNets learn concepts that are becoming increasingly abstract with the depth of the network, and when coming closer to the last classification layers. This is very useful especially for sketch recognition, where the drawn representations are sometimes very abstract. We finally showed that the features learned by those ConvNets are useful for similarity search.

In the future, we plan to further improve our system and also provide an objective assessment of the contribution of our similarity search approach to sketch retrieval, using proper evaluation metrics. We will also investigate the problem of sketch-based image retrieval, hence the possibility to retrieve natural images (or videos) from hand-drawn sketch queries.

REFERENCES

[1] CAO, Xiaochun, ZHANG, Hua, LIU, Si, et al. Sym-fish: A symmetry-aware flip invariant sketch histogram shape descriptor. In: Computer Vision (ICCV), 2013 IEEE International Conference on. IEEE, 2013. p. 313-320.

[2] EITZ, Mathias, HAYS, James, et ALEXA, Marc. How do humans sketch objects?. ACM Trans. Graph., 2012, vol. 31, no 4, p. 44.

[3] LI, Yi, SONG, Yi-Zhe, et GONG, Shaogang. Sketch recognition by ensemble matching of structured features. In: In British Machine Vision Conference (BMVC). 2013.

[4] SCHNEIDER, Rosália G. et TUYTELAARS, Tinne. Sketch Classification and Classification-driven Analysis using Fisher Vectors. ACM Transactions on Graphics (TOG), 2014, vol. 33, no 6, p. 174.

[5] SUTHERLAND, Ivan E. Sketch pad a man-machine graphical communication system. In: Proceedings of the SHARE design automation workshop. ACM, 1964. p. 6.329-6.346.

[6] SIVIC, Josef et ZISSERMAN, Andrew. Video Google: A text retrieval approach to object matching in videos. In: Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on. IEEE, 2003. p. 1470-1477.

[7] LAVIOLA JR, Joseph J. et ZELEZNIK, Robert C. MathPad 2: a system for the creation and exploration of mathematical sketches. In: ACM SIGGRAPH 2007 courses. ACM, 2007. p. 46.

[8] OUYANG, Tom Y. et DAVIS, Randall. ChemInk: a natural real-time recognition system for chemical drawings. In: Proceedings of the 16th international conference on intelligent user interfaces. ACM, 2011. p. 267-276.

[9] EITZ, Mathias, HILDEBRAND, Kristian, BOUBEKEUR, Tamy, et al. Sketch-based image retrieval: Benchmark and bag-of-features descriptors. Visualization and Computer Graphics, IEEE Transactions on, 2011, vol. 17, no 11, p. 1624-1636.

[10] EITZ, Mathias, HILDEBRAND, Kristian, BOUBEKEUR, Tamy, et al. An evaluation of descriptors for large-scale image retrieval from sketched feature lines. Computers & Graphics, 2010, vol. 34, no 5, p. 482-498.

[11] HU, Rui, BARNARD, Mark, et COLLOMOSSE, John P. Gradient field descriptor for sketch based retrieval and localization. In: ICIP. 2010. p. 1025-1028.

[12] HU, Rui et COLLOMOSSE, John. A performance evaluation of gradient field hog descriptor for sketch based image retrieval. Computer Vision and Image Understanding, 2013, vol. 117, no 7, p. 790-806.

[13] KRIZHEVSKY, Alex, SUTSKEVER, Ilya, et HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. In : Advances in neural information processing systems. 2012. p. 1097-1105.

[14] SZEGEDY, Christian, LIU, Wei, JIA, Yangqing, et al. Going deeper with convolutions. arXiv preprint arXiv:1409.4842, 2014.

[15] SARVADEVABHATLA, Ravi Kiran et BABU, R. Venkatesh. Freehand Sketch Recognition Using Deep Features. arXiv preprint arXiv:1502.00254, 2015.

[16] YANG, Yongxin et HOSPEDALES, Timothy M. Deep Neural Networks for Sketch Recognition. arXiv preprint arXiv:1501.07873, 2015.

[17] Collobert, R., Kavukcuoglu, K., & Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. In BigLearn, NIPS Workshop (No.EPFL-CONF-192376).

[18] OPENCV, Learning. Computer vision with the OpenCV library. GaryBradski & Adrian Kaebler-O'Reilly, 2008.

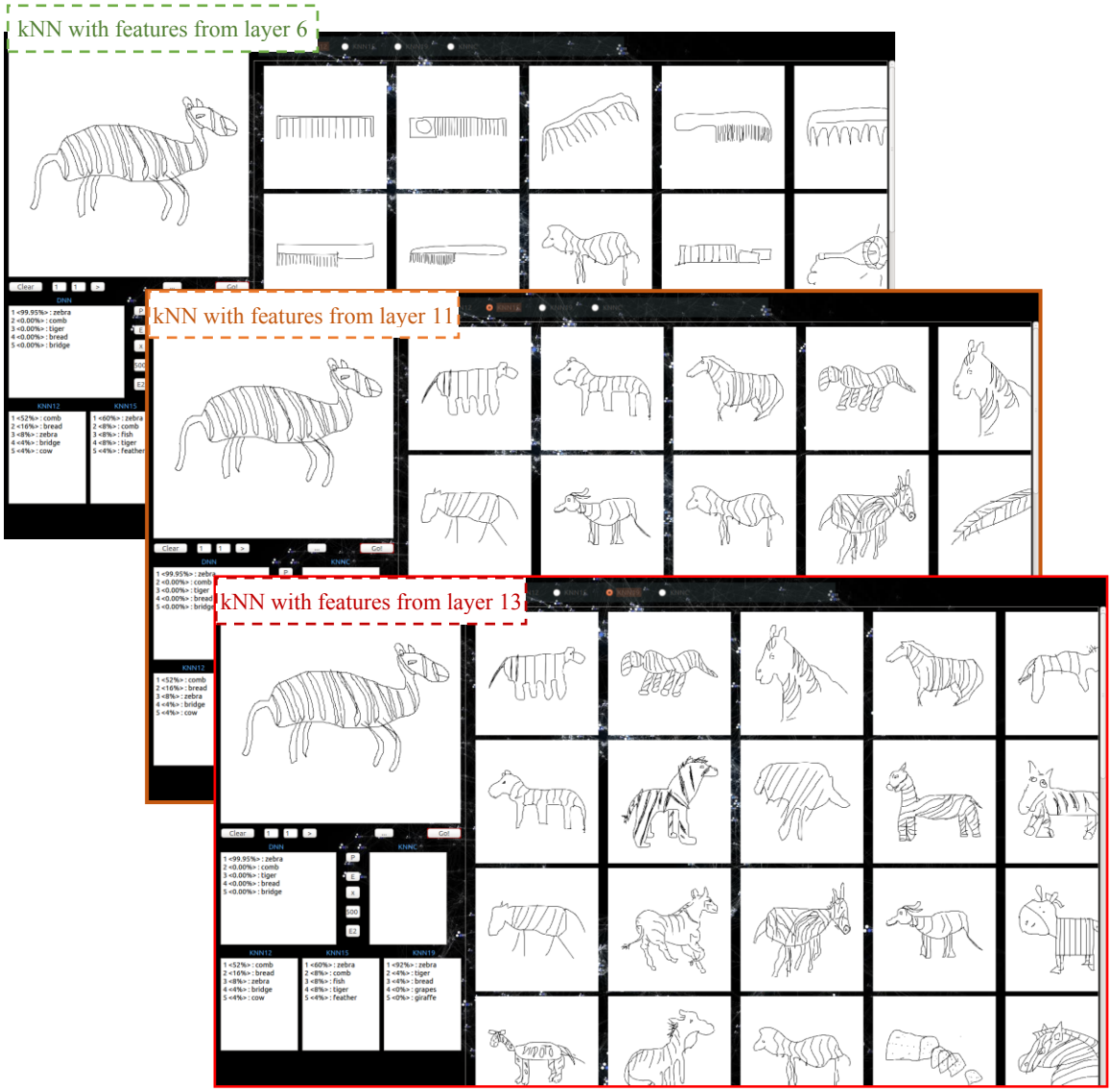

Figure 4 example of results obtained with our system for similarity search

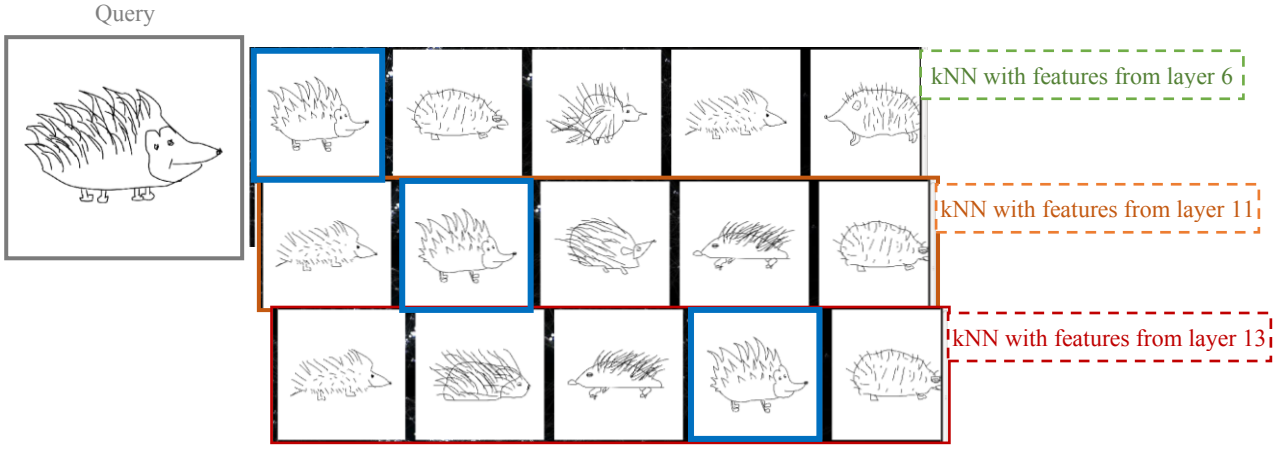

Figure 5 an example that shows the relationship between depth and details conservation