Cats vs Dogs Image Classification

Mohamed Nabil
Department of Computing Science
mohamed180353@fsc.bu.edu.eg

Abstract

In this project, our task is to develop an algorithm to classify images of dogs and
cats, which is the Dogs vs. Cats competition from Kaggle. We mainly investigated two approaches to address this problem. The first one is a traditional pattern recognition model. We extracted some human-crafted features like color and Dense-SIFT, represented images using bag of words model, and then trained Support Vector Machines(SVMs) classifiers. For the second approach, we used Deep Convolutional Neural Networks (CNN) to learn features of images and trained Backpropagation(BP) Neural Networks and SVMs for classification. We tried various experiments to improve our performance on the test dataset, and finally got the best accuracy of 94.00% by the second approach.

1 Introduction

1.1 Motivation and Background

The Dogs vs. Cats competition from Kaggle is trying to solve the CAPTCHA[3] challenge, which relies on the problem of distinguishing images of dogs and cats. It is easy for humans, but evidence [3] suggests that cats and dogs are particularly difficult to tell apart automatically.

Many people has worked or are working on constructing machine learning classifiers to address this problem. In [3], a classifier based on color features got 56.9% accuracy on the Asirra dataset [2]. In [17], an accuracy of 82.7% was achieved from a SVM classifier based on a combination of color and texture features. And in [18], they used the SIFT (Scale-Invariant Feature Transform) [13] features to train a classifier and finally got an accuracy of 92.9%.

In our project, we also would like to solve this problem and achieve higher performance. We tried different strategies. For instance, we tried Dense-SIFT features and the combination of DenseSIFT and color features, and features learned from CNN. Also, we employed SVMs on the learned features and finally achieved our best classication accuracy of 94.00%.

1.2 Task Definition

Our basic task is to create an algorithm to classify whether an image contains a dog or a cat. The input for this task is images of dogs or cats from training dataset, while the output is the classification accuracy on test dataset.

The given dataset for this competition is the Asirra dataset provided by Microsoft Research. Our training set contains 25,000 images, including 12,500 images of dogs and 12,500 images of cats, while the test dataset contains 12,500 images. The average size for these images is around 350×500.

Our learning task is to learn a classification model to determine the decision boundary for the training dataset. The whole process is illustrated in Figure 1, from which we can see the input for the learning task is images from the training dataset, while the output is the learned classification model.

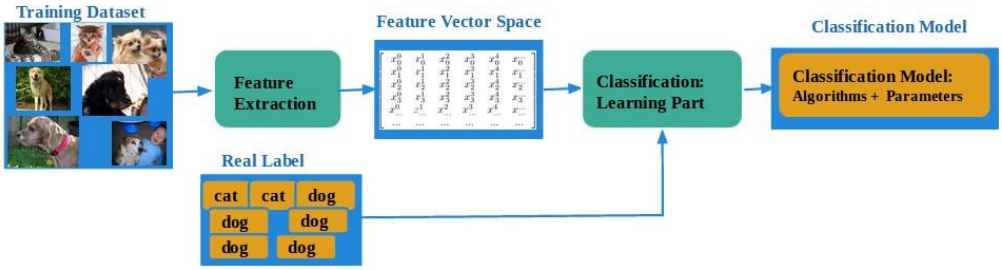

Figure 1: Architecture for Learning Task

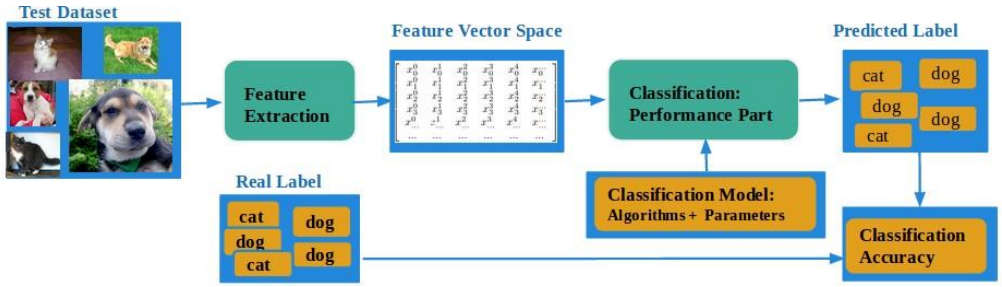

Figure 2: Architecture for Performance Task

Our performance task is to apply the learned classification model to classify images from the test dataset, and then evaluate the classification accuracy. As seen from Figure 2, the input is images from the test dataset, and the output is the classification accuracy.

1.3 Our Solution

In our solution, we mainly tried two different approaches. The first method is a traditional pattern recognition model, by which we learned the classification model from some human-crafted features, mainly including color feature, Dense-SIFT feature, and a combination of the two kinds of features. The second method is a trainable model, by which we applied a CNN to learn features. In terms of classifiers, we mainly chose SVMs and BP Neural Networks, considering the high dimensional feature space for images. We tried various experiments to achieve high accuracy on the test dataset, with different algorithms and parameter settings.

The outline of our paper is as follows. We introduce the first approach in section 2. The second approach is described in section 3. Finally we summarize our work and potential future work in section 4.

2 Method One: Using Human-Crafted Features

2.1 Human-Crafted Features

In typical image classification problems, we choose some fixed human-crafted features to use. There are many well studied features, such as SIFT, HoG[15], RGB or HSV color features, etc. In our project, the images are either dogs or cats. And we know that the shape and the priori probability of colors of dogs and cats are different. So we extracted local features descriptor SIFT and HSV color features to represent the original images.

The SIFT features are local and based on the appearance of the object, also invariant to image scale and rotations. The scale-invariant feature transform of a neighborhood is a 128 dimensional vector of histograms of image gradients, as shown in Figure 3. The region, at the appropriate scale and orientation, is divided into a 4×4 square grid, each cell of which yields a histogram with 8

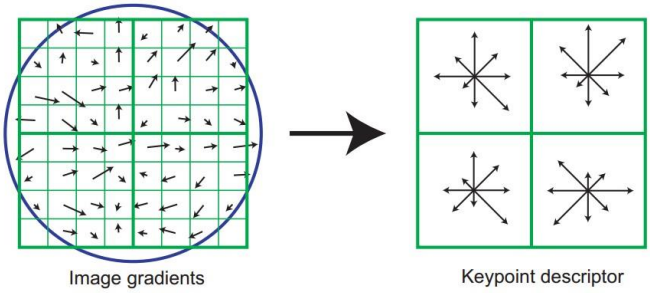

Figure 3: SIFT Descriptor [14]

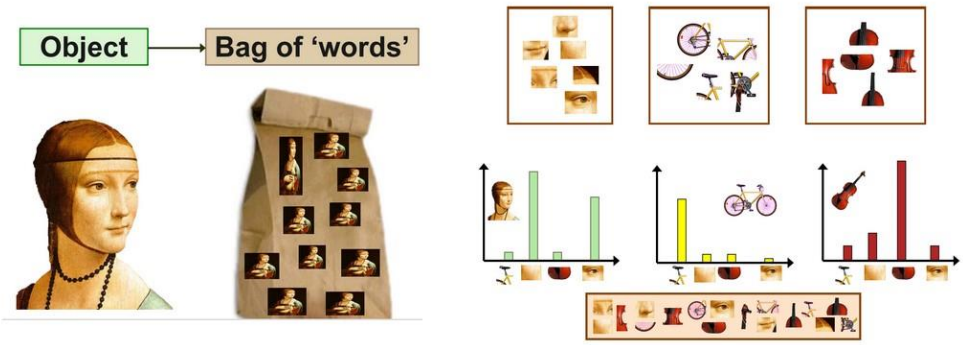

Figure 4: Bag of Words [16]

orientation bins. The Dense-SIFT is a fast algorithm for the calculation of a large number of SIFT descriptors of densely sampled features[12].

The HSV (hue, saturation, value) are the most common cylindrical-coordinate representations of points in an RGB color model, which rearranges the geometry of RGB in an attempt to be more intuitive and perceptually relevant. The reason why we chose it is that the HSV is closer to human perception of color and easy to interpret.

2.2 Our Model

After getting the Dense-SIFT and HSV features, we applied the bag of words model [16] to represent images. Figure 4 shows the bag of words model. It is a simplifying representation used in natural language processing and computer vision. In this model, a dictionary is formed by clustering the extracted features of training set using k-means algorithm. Every cluster is a word of this visual dictionary. Then images are represented by frequency vectors in which every dimention represents the proportion of features belong to a cluster. For classification, we trained SVM classifiers respectively on the Dense-SIFT feature, the HSV feature and the combination of the two. SVMs with suitable parameters have the ability to prevent overfitting, and experience shows that they usually can get good performance for classification.

Additionally, since the original images have various complicated backgrounds, we also tried the grab-cut [18] segmentation algorithm to cut the backgrounds of images.

2.3 Result and Analysis

When only using the Dense-SIFT features, the accuracy on the test dataset was only 67.60%. After combining with HSV feature, the accuracy increased to 71.47%. The performance is not good and here we analyzed some potential reasons. The most possible one is that the Dense-SIFT and color features of dogs and cats are quite similar. For instance, they both have one head, one tail and four legs. Also, their color has much in common.

For the image segmentation part, the classification performance turns out to decrease, due to the poor results of image segmentation. For example, in some images, the dogs or cats were even

cutted out. Also, some important information like tails or ears was removed during the segmentation process.

To further improve the performance, we can extract more distinctive features, use PCA to reduce dimension or try Deep Neural Network to extract features of images.

3 Method Two: Using Features learned by Convolutional Neural Network

Our second approach is to learn features by the CNN [5][6] and then use BP Neural Network or SVM classifiers to train these features. Different from human crafted features which are fixed features directly extracted from images, deep neural networks learn features from images, and discover multiple levels of representation, with higher-level features representing more abstract aspects of the data [4].

3.1 Deep Convolutional Neural Networks

The CNN is a kind of deep architecture which has achieved great performance in tasks like document recognition [5] and image recognition [6].

Different from traditional BP Neural Networks, which contains input layer, hidden layers and output layer, the CNN also contains Convolutional layers and Max Pooling layers.

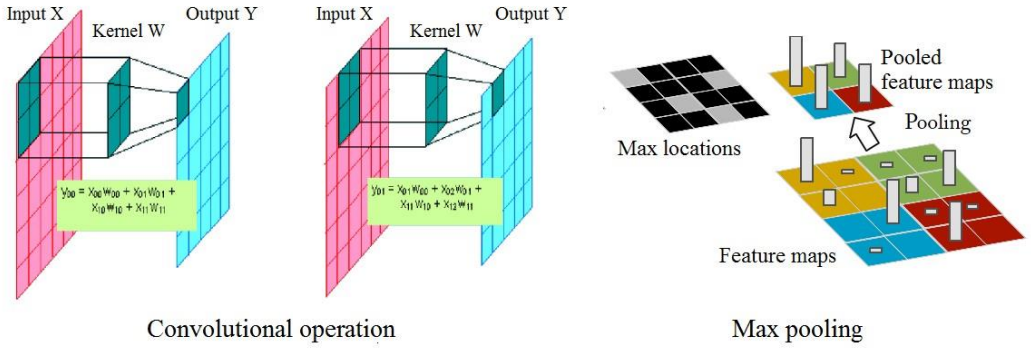

Figure 5: Convolutional Operation and Max Pooling in CNN[9][10]

Convolutional layers contain many feature maps, which are two dimensional hidden nodes. Every feature map owns a weight matrix called kernel, and different feature maps owns different kernels. Kernels do convolutional operation with every feature map in previous layer (layer j), then we sum them up and put it into sigmoid function. The output is the pixel value in layer j+1. With different kernels, we can learn different representations of data and the amount of parameters is not increased exponentially with the number of hidden nodes and layers.

Once a feature has been detected, its exact location becomes less important [5]. Only its approximate position relative to other features is relevant. Not only is the precise position of each of those features irrelevant for identifying the pattern, it is also potentially harmful because the positions are likely to vary for different instances of the pattern. Max pooling layers do sub-sampling operation on feature maps. For every four pixels, we only retain the max value, so that the size of feature maps will be half of the original size. Sub-sampling reduces the resolution of feature maps and reduces the sensitivity of the output to shifts and distortions, so that the model will be more robust. Max pooling operation can be incorporated into convolutional layers, and we don't need additional layers to do sub-sampling.

3.2 Our Model

In 2012, Krizhevsky and Hinton trained a CNN and achieved state of the art performance on the ImageNet 2012 classification benchmark [6]. Our model is based on the model of Krizhevsky. The original model contains 8 layers and the last three layers are two fully connected layers (same as hidden layers in BP Neural Network) and output layer. We cut off the last 2 layers and re-use the previous 6 layers to extract features from images. Recent work [8] proved that features extracted from the activation of a CNN trained in a fully supervised fashion on a large, fixed set of object recognition tasks can be re-purposed to novel generic tasks, which may differ

significantly from the originally trained tasks. And training a Deep Neural Network needs much experience and skill, and it is very time consuming. That is why we re-used the pre-trained network of Krizhevsky rather than training a deep architecture by ourselves.

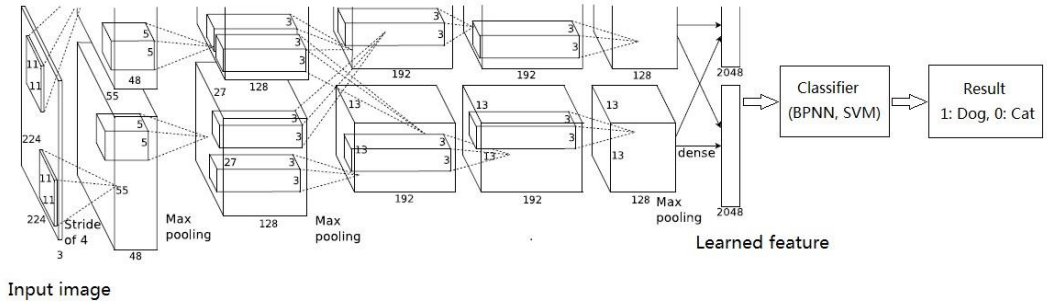

Figure 6: An illustration of the architecture of our model. The CNN was trained in parallel by two GPUs, and the figure explicitly shows the delineation of responsibilities between two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The networks input is 2242243 (RGB), and the number of neurons in the networks remaining layers is given by 55×55×96, 27×27×256, 13×13×384, 13×13×256, 4096. Then we train a classifier using the 4096 dimensional features and classify images as dog or cat. We refer to [6] for more details about the network.

The architecture of this model is illustrated by figure 6. First, input images are normalized into 224×224 (first normalize the short edge size to 224 and then choose the center part of images); then the first convolutional layer filters the 224×224×3 input image with 96 kernels of size 11×11×3 with a stride of 4 pixels (this is the distance between the receptive field centers of neighboring neurons in a kernel map). The second convolutional layer takes as input the pooled output of the first convolutional layer and filters it with 256 kernels of size 5× 5×48. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling layers. The third convolutional layer has 384 kernels of size 3×3× 192, and the fifth convolutional layer has 256 kernels of size 3×3×192. The fully-connected layers have 4096 neurons. Until now, the 6 layer CNN transforms an image into a feature vector of 4096 dimensions. Then we trained the BP Neural Networks and SVMs with 5 fold cross-validation on the extracted features. After that, we applied the learned models to classify the test set and evaluated their performance.

3.3 Result and Analysis

In this method, we finally got the best accuracy 94.00% from an SVM classifier (with RBF kernel and C = 10000), which is impressive. To clearly understand how and why the CNN work, here we look insight into the internal operation and behavior of the complex model.

To understand what every layers learned from images, we need to figure out what kind of inputs activate a feature map. A feature map is activated if its pixel value is 1. Zeiler [9] proposed a visualization technique using multi-layered Deconvolutional Network (deconvnet) [11] to reveals the input stimuli that excite individual feature maps at any layer in the model. It projects the feature activations back to the input pixel space. A deconvnet can be thought as a convent model that uses the same components (filtering, pooling) but in reverse, so instead of mapping pixels to features does the opposite. To examine a convnet, a deconvnet is attached to each of its layers, providing a continuous path back to image pixels. To examine a given covnet activation, we set all other

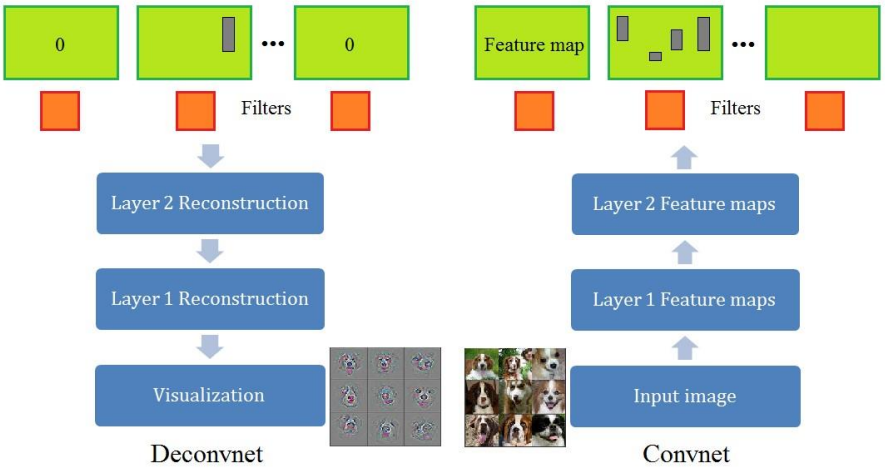

Figure 7: Projecting back from higher layers.

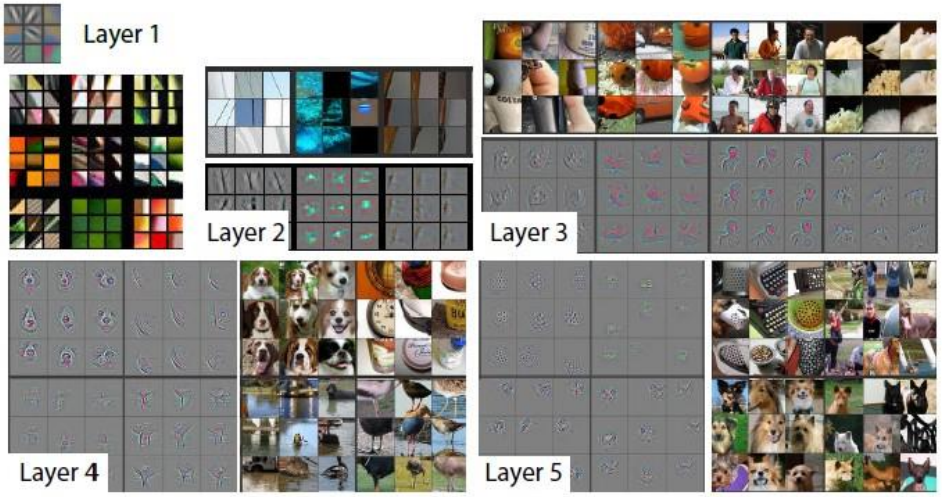

Figure 8: Feature visualization using deconvnet. Images are from the ImageNet2012 validation set[9]

activations in the layer to zero and pass the feature maps as input to the attached deconvnet layer. Then we reconstruct the activity in the layer beneath until input pixel space is reached. For more details about decovnet please refer to [9][11].

Figure 8 shows feature visualizations for different layers [9]. Instead of showing the single strongest activation for a given feature map, we show the top 9 activations. Projecting each separately down to pixel space reveals the different structures that excite a given feature map, hence showing its invariance to input deformations. Alongside these visualizations we show the corresponding image patches. These have greater variation than visualizations as the latter solely focus on the discriminant structure within each patch. For example, in layer 5, row 1, col 2, the patches appear to have little in common, but the visualizations reveal that this particular feature map focuses on the grass in the background, not the foreground objects.

The projections from each layer show the hierarchical nature of the features in the network. Layer 1 learns some basic edges and colors. Layer 2 responds to corners and other edge/color conjunctions. Layer 3 has more complex invariances, capturing similar textures or patterns. Layer 4 shows signif-

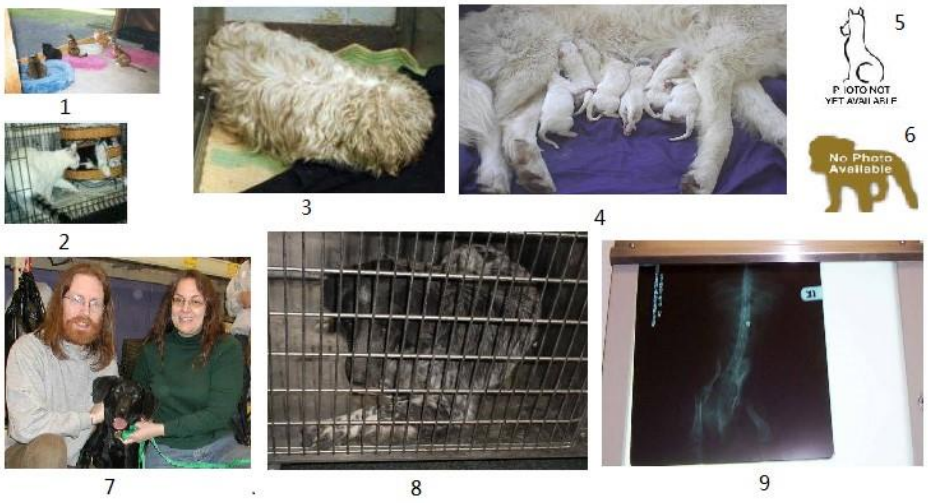

Figure 9: Some incorrectly classified images in the kaggle test set

icant variation, but is more class-specific: dog faces, birds legs. Layer 5 shows entire objects with significant pose variation, e.g. dogs and grass.

Now we get an insight into why deep architectures can achieve good performances. Features learned by Deep Neural Networks are hierarchical. With more hidden layers, the learned features become more high level and specific. Compared to human-crafted features such as Dense-SIFT or color features, they are more expressive, class-specific, and invariant to backgrounds. We can also figure out which layers are critical to recognition and change the parameters and architecture of deep neural network , e.g. increase the number of feature maps in some layers, to achieve better performance.

To know the remaining problems and achieve better performance, we investigated what kind of images were incorrectly classified. Figure 9 reveals some characters of images that have been failed to classify. Some images' resolutions are too low to recognize (1 and 2), and some images' critical characters, e.g. faces, are hidden (3 and 4). The test set also contains some cartoon images (5 and 6) that only contain simple shapes of dog or cat, which are hard to classify. Also, many wrongly classified images contain backgrounds (7 and 8) that are very complicated or similar to the forground animals. Finally, some images even cannot be recognized by humans (9).

From the properties of images that failed to recognize, we can try different strategies. We may try object localization to locate the animals in images, so that complicated backgounds can be elimited. We may also take advantage of extra training set to improve the ability of classification for special images such as cartoon dogs or cats. We can also combine features such as texture feature with features learned by Deep Neural Network, which may be helpful for the recognition of images that only contains the fur or partial of the animal's body.

4 Conclusion and Future Work

In this report, we first briefly explained our motivation of this project and showed some background materials. Then, we precisely illustrated our task, including the learning task and the performance task. After that, we introduced our solution in detail, mainly including two approaches.

The first approach is a traditional pattern recognition model, by which we learned the classification model from some human-crafted features, mainly including color feature, Dense-SIFT feature, and a combination of the two. To improve the performance, we also applied image segmentation approach to preprocess the data. However, due to poor segmentation result, we did not achieve any improvement. The best accuracy we got from the first method is only 71.47% (from an SVM classifier).

To achieve better performance, we implemented our second approach, which is a trainable model that applies the CNN to learn features. We also looked insight into what Deep Networks learned from images and explained why they achieve good performance. The highest accuracy of this approach is 94.00% (from an SVM classifier), which is also our best result and helps us rank 9th in 91 teams in the Kaggle competition.

In terms of classifiers, we mainly considered SVMs and BP Neural Networks, taking our high dimensional feature space into account. Various parameter settings were explored to improve classification accuracy on the test dataset. For example, for the BP Neural Networks, we tried different hidden layers and hidden units; for the SVMs, different kernel functions and C parameters were used. Table 1 illustrates the best results of each model and related parameters.

Table 1: Best Performance on the Test Dataset for Different Models

Feature	Classifier	Parameter Setting	Accuracy
Dense-SIFT	SVM	Linear kernel, C=1000	67.60%
Dense-SIFT+Colors	SVM	Linear kernel, C=5	71.47%
From Deep CNN	BP Network	1 Hidden layer, 30 Neurons	93.01%
From Deep CNN	SVM	RBF kernel, C = 10000	94.00%

In the future, we will explore more to achieve better performance. For instance, we will try to change the architecture and parameter settings of the Deep Neural Network based on the feature visualization of different layers' feature maps. Also, we will apply different parameter settings for SVMs and Deep Neural Networks. We may also try object localization to eliminate the influence of complicated backgrounds. Additionally, we would like to extract more features or try a combination of human-crafted features and learned features.

Acknowledgments

We would like to express our appreciation to Dr. Russ Greiner. Thanks for his time and efforts on guiding the whole process of our project. Also, we would like to thank Junfeng for being our co-coach and providing some suggestions. Apart from that, we would like to thank Dr. Mohamed Elgendi for his suggestions on image preprocessing.

References

[1] Kaggle DogVCat Competation: http://www.kaggle.com/c/dogs-vs-cats

[2] MSR Asirra: http://research.microsoft.com/en-us/um/redmond/projects/asirra/

[3] J. Elson, J. Douceur, J. Howell and J. Saul. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. Proc. of ACM CCS 2007, pp. 366-374.

[4] Bengio, Y. (2013). Deep learning of representations: Looking forward. arXiv preprint arXiv:1305.0445. [5] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[6] Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25 (pp. 1106-1114).

[7] Huang, F. J., & LeCun, Y. (2006, June). Large-scale learning with svm and convolutional for generic object categorization. In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on (Vol. 1, pp. 284-291). IEEE.

[8] 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.

[9] Zeiler, M. D., & Fergus, R. (2013). Visualizing and Understanding Convolutional Neural Networks. arXiv preprint arXiv:1311.2901.

[10] http://www1.i2r.a-star.edu.sg/ irkhan/conn2.html

[11] Zeiler, M. D., Taylor, G. W., & Fergus, R. (2011, November). Adaptive deconvolutional networks for mid and high level feature learning. In Computer Vision (ICCV), 2011 IEEE International Conference on (pp. 2018-2025). IEEE.

[12] Vedaldi, A., & Fulkerson, B. (2010, October). VLFeat: An open and portable library of computer vision algorithms. In Proceedings of the international conference on Multimedia (pp. 1469-1472). ACM.

[13] Lowe, D. G. (1999). Object recognition from local scale-invariant features. InComputer vision, 1999. The proceedings of the seventh IEEE international conference on (Vol. 2, pp. 1150-1157). Ieee.

[14] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints.International journal of computer vision, 60(2), 91-110.

[15] Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on (Vol. 1, pp. 886-893). IEEE.

[16] Bag of Words: http://cs.nyu.edu/ fergus/teaching/vision 2012/9 BoW.pdf

[17] Golle, P. (2008, October). Machine learning attacks against the Asirra CAPTCHA. In Proceedings of the 15th ACM conference on Computer and communications security (pp. 535-542). ACM.

[18] Parkhi, O. M., Vedaldi, A., Zisserman, A., & Jawahar, C. V. (2012, June). Cats and dogs. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on (pp. 3498-3505). IEEE.

[19] Dense SIFT: http://www.vlfeat.org/api/dsift.html