Mobile image detection using features extracted from Convolutional Neural Network

Yusuf Yaslan
Istanbul Technical University
yyaslan@itu.edu.tr

Payam V. Azad
Istnabul Techinical Unviersity
vakil@itu.edu.tr

Abstract

The purpose of this research is to solve detecting images problem when we have very few training sample (just one in our case). We have attempt solving it using Convolutional Neural Networks and used features extraced from different layers of network. We also tried Transfer Learning and Augmentation to enhance the results.

1. Introduction

Computer Vision and Machine Learning is the most active field in Computer Science field and day by day computers power in solving various problems is immensely improving. By advent of Deep Learning from 2012 [9] we percieved a huge improvement in vision problems. These days CNNs surpass human performance in lots of tasks and it still getting better. For getting the best out of deep learning we need big number of train sample [20]. But training models with rare samples is still unsolved problems that we tried to solve it.

1.1. DataSet

As a dataset we have used Stanford Mobile Visual Search Dataset [3]. This dataset contains different catagories like CD Covers, DVD Covers, Book Cover and Paintings. From each catagory there is about 100 sample that is our Reference samples provided in high quality. As a test data for each catagory we have equal number of pictures taken by different mobile phones(one picture per item). We are suppose to train our model over Reference images (one-shot) and try to detect the each picture taken by mobile belongs to which object.

1.2. Background Check

We started our research based on Onur Celikaya's Master Thesis [2]. He have solved this problem using Vocabulary Trees and extracted features using SIFT [12] Based on a work from Nister et al [13].

As we have mentioned solving this problem with Deep Learning is pretty difficult task because of small size of training dataset but there have been several attacks specially in recent years for similar datasets using Deep Learning. A neat solution suggested by Held et al [7] using Augmentation and increase the size of dataset that we have used it. But without augmentation also DeepMind team from google have done precious works like [20][14][15] that have been published very recently. We also took some of our ideas from them.

1.3. Hypothesis

As though Deep Learning works based on extracting features inside itself we have proposed that we can use these features extracted from midlle layers of our Deep Learning network for instance detection instead of classic SIFT or SURF features. We want to use these features as a input to custome developed detection function that is working based on consine distance and Pearson Correlation between test image and reference image.

2. Methods

Nueral Networks has been out as a general solver for more than 40 years but due to some problems like difficulty of training and optimizing, being so computational intensive and vanishing gradient problem never gain this much fame. But in recent years there have been a big lean toward Deep Nueral Networks after new methods like backpropagation, convolutional neural networks had emerged and also huge improves happened in the hardware field like using highly capable GPU's for training complex networks.

2.1. Convolutional Neural Network

Convolutional Neural Netowrks are type of Deep Nueral networks that imply small filters over each layer as convolution layer and convolut these tiny filters over input of the layer (that in the case of first layer it is pixels). These filters is been learned during the procedure of foreward-backward propagation. The first succusful network has been proposed by Alex Krizhevsky et al [10] at 2012 that is called AlexNet and was been a leap in solving complex problem of ImageNet. Also CNNs had earlier appearences in Academia by Yann Lacun and his team [11]. This is succuss still continues and every years new models is emerging that are superior to their succesors and there is no other algorithm can be competing with CNNs in object detection.

2.1.1 Networks

New CNN network models has been created every year and it is getting smarter and also more complex. Now we have extremely complex and also immensely powerful network from LeNet created by Yann Lacun at 1995 [11]. This network just had 7 layers and except being first working CNN had not achieve any succuss. In comparison 18 layered AlexNet [9] was a great succuss. This success and complexity pattern goes on with VGGNet [17], GoogLeNet [18] and now we have Residual Network [6] with 150 layers. At this research we have used GoogleNet because of it is combination of impressive power and not being extremely heavy like ResNet or even newer Inception-v3. Also we have used AlexNet and VGGNet that both had significantly poorer results than GoogLeNet.
We have not used ResNet or Inception-v3 because they need intense computational power that we was shy of it so we thought GoogLeNet would be the optimum network for us.

2.2. Transfer Learning

As we have discussed before CNNs constructed from multiple layers of filters that we need to learn these filters during training process. Using small datasets for this training lead to drastic overfitting. And also training a network over a reasonable size dataset may takes more than weeks using powerful GPU servers. For solving these problems Transfer Learning proposed [5] [16].
In Transfer Learning we can train our model over a huge dataset like ImageNet that compound of millions of pictures with 1000 labels and then fine tune it for other datasets by changing last layers. The reason that it is working is that first layers of CNN tries to extract general features of images like colors, edges and general patterns. It is the last layers that tries to extract specefic patterns of dataset.
So by implementing Transfer Learning we can use previously trained networks over ImageNet that is done by extremely powerful machines and just fine-tune 2-3 buttom layers for our dataset that will took immensely less time and computation.

2.3. Augmentation

Data Augmentation in case that train data is small in images is pretty widespread approach. [7] In This approach we try to mimic test sets and by distorting reference image trying to create new images and extend our train set.
We also used this technique and done 4 augmentation. First rotate 45 degrees, then rotate 135 degrees and Second we tried to randomly add a small number to color channels. We have done it twice so from one reference image totally we created 4 augmented images.
In this process we tried to create images similar to images taken by mobile phone that is possible to take from different angle and there can be color distortions.

2.4. Detection

For detecting each instance we created our own method. Input parameter of this method is two feature vector derived from reference and test image and we try to find the most similar images by this method. For calculating similarity we are calculating combination of cosine distance of two feature vector, euclidean distance of histograms and Pearson Correlation between these vectors. And by using a ranking algorithm give them each a point based on our hyper parameters that is been extracted by different manual tests.

2.4.1 Feature Vector

The feature vector used for calculating similarities of each image is derived from different layers of networks. After several tests the best vector that we could found was contatination of features created by 2nd, 3rd and last layers.

2.5. Tools

For running and fine-tuning CNN we have used Google's TensorFlow library. [1] There are few well-known libraries for Deep Learning including Caffe[8], TensorFlow[1], Theano[19] and Torch[4] that each have its cons and pros. We have started using caffe for our tests because it has the richest resource of pre-trained models but migrated to TensorFlow because of its extremely well structure, ease of use and power. And converted pre-trained models from caffe into TensorFlow.
TensorFlow is written by python in interface but it has C++ and Cuda as its backbone. It is been developed by Google and used in Google labs so it is highly up-to-date and efficient. It support multi GPU servers and also multiple GPU servers. The other big advantage of TensorFlow is ease of switch between GPU processing and CPU processing that is done by just a flag without any headache and end-user can use its model either in CPU or GPU without any effort.

3. Results

3.1. Without Augmentation

3.2. With Augmentation

4. Discussion

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] O. Calikus. Yerel znitelikler kullanilarak grnt indeksleme ve eleme, 2016.

[3] V. R. Chandrasekhar, D. M. Chen, S. S. Tsai, N.-M. Cheung, H. Chen, G. Takacs, Y. Reznik, R. Vedantham, R. Grzeszczuk, J. Bach, et al. The stanford mobile visual search data set. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 117–122. ACM, 2011.

[4] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[5] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, pages 647–655, 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[7] D. Held, S. Thrun, and S. Savarese. Deep learning for single-view instance recognition. *arXiv preprint arXiv:1507.08286*, 2015.

[8] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[11] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, et al. Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks*, volume 60, pages 53–60, 1995.

[12] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[13] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 2161–2168. IEEE, 2006.

[14] D. J. Rezende, S. Mohamed, I. Danihelka, K. Gregor, and D. Wierstra. One-shot generalization in deep generative models. *arXiv preprint arXiv:1603.05106*, 2016.

[15] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*, 2016.

[16] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

[17] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[19] T. T. D. Team, R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[20] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.