Kemal Berk Kocabağlı

Summer'17 Research Summary

Four Eyes Lab at UCSB

Advisor: Professor Matthew Turk
Mentor: Jedrzej Kozerawski

Github: berk94
kberkkocabagli@gmail.com

Abstract

This paper explores the problem of one-shot-one-class learning, which is an object-recognition task from a single positive input image. We have a source dataset independent from the testing domain. We use feature sharing to detect similarity between the novel image and the already existing classes in our source dataset. Then, considering the similarity information, we use the top N most similar images in model regression networks to learn a transformation between small sample and large sample classifiers and the rest of the classes in building a weak soft support vector machine (SVM) decision boundary for the novel image. Finally, we transform the weak SVM into a more generalized decision boundary using the weights learned by the model regression networks.

Before I begin

Prior to my summer research at UC Santa Barbara, I had not taken any convolutional neural networks, computer vision or graphics classes. I was at the more statistical side of machine learning, had mainly processed text. Therefore, I had to read and watch a lot before I could form the slightest idea about my new project: one-shot-one-class object recognition. My preparation entails a myriad of blogs I scrolled through (like this one **here**), papers (Wang&Hebert, Koch) I took notes on and videos (especially CS231N at Stanford) I paused and played intermittently. Getting familiar with the research jargon and actually fully understanding the papers I read was not easy and I still have a lot to learn.

Contents

1 Introduction

Humans are amazing at detecting objects upon just a single shot. However, computers require hundreds if not thousands of images to learn a feature representation of a certain class and use it for prediction. Gathering and labelling thousands of images is very difficult, or in certain cases, not possible. Take a robot in a completely alien environment, for instance, who is expected to recognise any object that it has seen once. Although how we do it still remains an enigma, there is a believable theory behind it, which involves our use of past experience or information and generation of meta-data to apply to new domains.

When we abstract the task of object recognition to a generic level, we can say that every object shares some properties such as; being darker in color under less amount of light, having an orange tint under sunlight and a blue one under moonlight. All of the objects are composed of geometrical shapes whose transformations in 6 degrees of freedom could be learned. These transformations could be applied to new objects, who are again composed of geometrical shapes, to guess how might the new object look like after a certain transformation, or under certain light conditions.

In our method, by observing classifiers provided with different amounts of positive training data, we aim to learn a transformation between them. Hence, after this transformation is applied to a basic SVM classifier with a linear kernel built on the one positive image, we will get an estimation of how a classifier with more training data (which we do not have) would look like. Implicitly, we are using the intra-class variance of other classes to have an idea of that of our novel class.

2 Problem Statement

In one-shot-one-class setting, we are only given one positive-labeled image and no negative instances.
The task is finding a classifier for the novel image so that we will be able to recognise it in further encounters.

An alternative version of this problem is where we are given a support set $S = \{(x_i, y_i)|i \in [1, N]\}$ which contains one example per each novel class y_i and the purpose is classifying a new testing image as one of the classes in the support set. However, in our setting, we are not assuming that the test object should necessarily belong to one of the classes in the support set.

3 Proposed Solution

In our first meeting with my mentor Jedrzej Kozerawski and advisor Professor Matthew Turk on July 3rd, 2017, we discussed the ways I could contribute to what Jedrzej had already done.

In his paper, Jedrzej [2017] uses a source dataset (ILSVRC2012, ImageNet) independent from the test domain to pick an auxiliary set, which he considers to be negative examples for the novel class. He then creates a weak SVM decision boundary, w^0, from the one positive image and the negative images in the auxiliary set. Finally, he transforms this weak classifier into a better, more generalized one through transfer learning ($w^* = \lambda_0 * T_s(T_0(w^0)) + \lambda_1 * T_0(w^0) + \lambda_2 * w^0$), where T_0 and T_s are learned with two model regression networks (one for each) and correspond to the transitions between one-shot to few-shot and few-shot to many-shot classifiers respectively.

Throughout the whole process, Jedrzej uses the fc6 layer of AlexNet (Krizhevsky [2012]) -trained on ILSVRC2012 - for feature extraction, which gives a vector of length 4096 for each image. On top, he applies linear discriminant analysis (LDA) to further reduce the dimensionality to 500 (so, his linear SVMs will have 501 parameters with the bias).

A complication Jedrzej told me about was related to the granularity of the test set. The resulting classifier after the linear combination might generalize or overfit too much and we need to find a balance depending on how fine-grained the test set is. See the figure below.

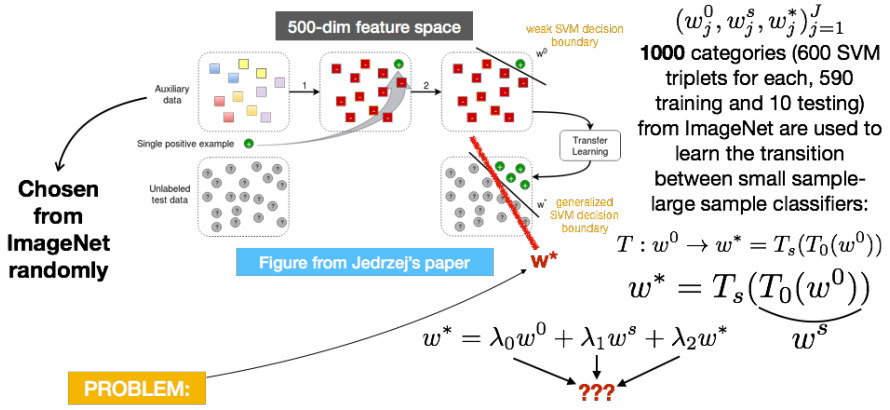

Figure 1: Deep Transition Network workflow. AlexNet fc6 is used for feature extraction with LDA resulting in 500-dim feature space.

I first considered estimating the parameter k as in k-means clustering as (Hamerly, Elkan) using statistical tests or an information criterion (such as Bayesian) and then finding a relationship between this k and the λs, but it seemed to be too intractable. Intuitively, the higher the k, the

more the granularity and the less the decision boundary diameter.

Then I looked into Siamese Neural Networks (Koch, Zemel, Salakhutdinov [2015]) to infer a granularity parameter based on similarity learning, however I abandoned that idea as well since it would not be feasible to complete in short amount of time during my research.

Finally, I came up with a method I call informed DMT, or iDMT, which is an improvement over Jedrzej's framework with a parameter N that could be used to adjust for different granularity.

4 My contribution

When I thought on how humans in general classify objects, I realized that while we do use our general knowledge of all objects, we weigh things that are most similar to the new object more than the rest. My approach involves first finding the top N most similar classes from our source data to the new object and then utilising this information.

Therefore, I made two changes to Jedrzej's methodology:

1. Instead of picking the auxiliary set randomly from the source data set, pick those negative images only from classes that are **not** in the top N most similar classes.
2. Instead of using the same classifier transitions T_0 and T_s for all novel objects, apply fine-tuning on top of the network weights learned from the entire dataset taking into account only the top N most similar classes to weigh the effect of similar classes more on the transitions.

Assumption: The source dataset should be labeled for iDMT.

4.1 Similarity Judge Module

4.1.1 Implementation

To find the most similar N classes to our novel image, we choose a deep convolutional neural network (CNN) trained on as many different classes as possible so that its layers hold more precise feature representations. A good candidate is **AlexNet**, since it is trained on 1000 classes from the ILSVRC2012 dataset. Note that if our test domain is very high-grained, we would need both our source data and the training data for our feature extraction model to be high-grained for best performance.

For feature extraction, we pick an intermediate layer rather than the last layer, because as we go towards the end of the network, only the most discriminative features regarding the dataset the CNN was trained on remain, which might not be useful to describe novel images. The intermediate layer should not come too early in the network, as the features would then be too general to discriminate anything (edges, for example. All objects have edges.)

After that, for each class C^i in our source dataset, we pick K images randomly and find the average cosine similarity between the feature vector of our novel image (call it FV_{novel}) and the feature vectors FV_j of the randomly selected images from class C^i. The average cosine similarity can be written as:

$$CS^i_{average} = \frac{\sum_{j=1}^{K} CS(FV_{novel}, FV_j)}{K}$$

where

$$CS(FV_{novel}, FV_j) = \frac{FV_{novel}.FV_j}{||FV_{novel}||_2||FV_j||_2}$$

Applying argsort on the list of the average cosine similarities in descending order and selecting the first N values gives us the indices of the top N most similar classes. For instance, if we get [0, 21, 32, 5, 8] as the resulting list for $N = 5$, it would mean that class 0 (first class) is the most similar to our novel image, followed by class 21 (22nd class), followed by class 32 (33rd class) and so on.

For experimental purposes, I conducted a sanity check using fc6 and fc8 layers of AlexNet to measure how well similarityJudge performs. First, I picked one random image per class in the source dataset and found the list of classes from most to least similar for each. To illustrate, for dog class I would expect the index of the dog class to come first in the returned list, since we know that the image actually belongs to the dog class. Then, for each class C^i I found the index of C^i in the most to least similar list. Ideally, we would like this value to be 0 for all classes as each class is most similar to itself. I counted the number of classes where the index of C^i was in top-1, top-3 and top-5. fc6 performed better than fc8 in this test.

Then, to make a fair comparison, I did the same experiment but this time took the first image of each class instead of a random image. Again, fc6 performed better than fc8. The sanity check script (simJudge_sanity_check.py) may need to be modified for different feature extraction models.

4.1.2 Results

The source dataset I used was Caltech-256.

Table 1. Performance upon random image selection from each class

Feature extraction model	Layer	Top-1 count	Top-3 count	Top-5 count
AlexNet	fc6	126	168	185
Alexnet	fc8	118	155	172

Table 2. Performance upon selecting the first image of each class

Feature extraction model	Layer	Top-1 count	Top-3 count	Top-5 count
AlexNet	fc6	152	190	206
Alexnet	fc8	134	189	204

Intuitively, fc6 should also perform better on novel images since it contains more general features that are less biased to discriminate only the training set of AlexNet compared to fc8. I could only test it with eyeballing, but here are some examples:

Figure 2: Input image = centaur
Most similar 10 classes (highest to lowest):
121.kangaroo-101
105.horse
254.greyhound
134.llama-101
065.elk
181.segway
159.people
229.tricycle
028.camel
114.ibis-101

Figure 3: Input image = statue
Most similar 10 classes (highest to lowest):
144.minotaur
121.kangaroo-101
112.human-skeleton
111.house-fly
049.cormorant
134.llama-101
219.theodolite
084.giraffe
066.ewer-101
151.ostrich

Figure 4: Input image = dragon
Most similar 10 classes (highest to lowest):
205.superman
111.house-fly
144.minotaur
228.triceratops
053.desk-globe
003.backpack
179.scorpion-101
066.ewer-101
146.mountain-bike
076.football-helmet

Figure 5: Input image = werewolf
Most similar 10 classes (highest to lowest):
090.gorilla
038.chimp
168.raccoon
121.kangaroo-101
009.bear
134.llama-101
056.dog
186.skunk
060.duck
085.goat

4.2 Workflow of iDMT

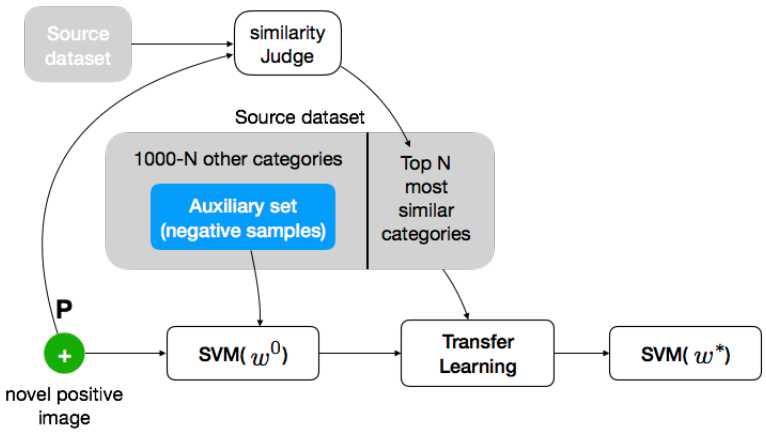

Figure 6: Informed Deep Transition Network workflow. As in DMT, AlexNet fc6 is used for feature extraction. The extension is the similarityJudge module which gives us extra information about the source dataset so that we make an informed selection when building the auxiliary set and applying transfer learning.

From getting the novel input image to outputting a classifer for it, the steps of the algorithm can be summarized as:

1. Feed the novel image and the source dataset to the similarityJudge module to receive the list of top N most similar classes.

2. Give the top N classes to the SVMTripletGenerator module to receive $S * N$ SVM classifier triplets: $(w_j^0, w_j^s, w_j^*), j \in [1, S * N]$

3. Use the (w_j^0, w_j^s) pairs to train (or fine-tune)[1] the first Model Regression Network (MRN1) to find T_0

4. Use the (w_j^s, w_j^*) pairs to train (or fine-tune)[2] the second Model Regression Network (MRN2) to find T_s

5. Generate the auxiliary set for the novel image by random sampling from classes in the source dataset that are not in the top N most similar classes.

6. Generate a weak SVM decision boundary w^0 using the positive novel image and negative samples from the auxiliary set.

7. Feed w^0 into MRN1=T_0 to get w^s

8. Feed w^s into MRN2=T_s to get w^*

9. The final classifier to be returned will be the affine combination

$$w = \lambda_0 * w^0 + \lambda_1 * w^s + \lambda_2 * w^*, \sum_{i=0}^{2} \lambda_i = 1$$

[1,2]: if N is small and there isn't enough classifiers for training the model regression networks from scratch, use fine-tuning on networks pre-trained on classifier triplets from all classes in the source dataset (in this case, we have to pre-train both T_0 and T_s).

5 Conclusion

Since I could not complete all of the code for iDMT, I could not test and compare it to DMT on the same target datasets. I will work on the code once I get back to Turkey to finish and refine it.

The link for the GitHub project repository is:
https://github.com/berk94/One-Shot-One-Class-Learning

I would be happy to receive pull requests from Jedrzej or anyone (other undergraduate or graduate researchers) who is interested in the same problem and wants to use/develop my code.

6 Future work

One-shot-one-class(OSOC) learning is a very intriguing area and demands further investigation. While I was building the similarityJudge module, I also had the chance to think on how feature sharing can be used between different convolutional networks and which layers are better for what purposes. Just the comparison of the performance of layers within a network at certain tasks could be a good contribution to the community.

Many other concepts could be integrated into one-shot-learning such as; ensemble learning, recurrent neural networks and reinforcement learning. In all cases, the boundaries of the OSOC problem should be clear. For example, if we are given the image of a Volvo and told that it represents class X, should we associate class X with specifically a Volvo or just a car? At what point do objects cease being the same object?

7 General Remarks

When I look back into the summer, I feel very lucky to be a part of the Four Eyes Lab and experience the life of a graduate student firsthand. In undergraduate, we are usually guided in classes and the expectations of our teachers are straightforward. Yet, in research, although our advisors give us hints and ideas, we have to find our own direction.

I want to specifically thank Professor Matthew Turk for providing me such an opportunity, Jedrzej Kozerawski for guiding me through the process and all my colleagues in the Four Eyes Lab including Benjamin Nuernberger, Martin Hering, Adam Ibrahim, Hilda He and Nikolas Chaconas. I also would like to thank John O'Donovan and Tobias Hollerer for their valuable discussions.

8 Bibliography

[1] Hamerly, G. and Elkan, C. 2003. Learning the k in k-means. In Advances in Neural Information Processing Systems (NIPS).

[2] Krizhevsky, Alex; Sutskever, Ilya and Hinton Geoffrey E. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097?1105, 2012.

[3] Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition."ICML Deep Learning Workshop. Vol. 2. 2015.

[4] Kozerawski, Jedrzej: DMT: Deep Model Transition for One-Shot One-Class SVM Classification. 2017.

[5] Luo, Hengliang: Siamese Network: Architecture and Applications in Computer Vision. 2014.

[6] Wang, Yu-Xiong and Hebert,Martial. Learning to learn: Model regression networks for easy small sample learning. In European Conference on Computer Vision, pages 616?634. Springer, 2016.