# Learning To Detect Unseen Object Classes by Between-Class Attribute Transfer

Ahmed MAZARI Hafed RHOUMA

February 24, 2017

## Introduction

In this project, we study the paper [4] entitled "Learning to detect unseen object classes by between-class attribute transfer". The aim is to understand the output prediction methods developed by the authors. The study copes with the problem of object classification for computer vision tasks when training and test classes are disjoint. attribute-based classification technique is introduced to handle such tasks. The object detection relies on human-specified high-level description of target object. The description is based upon semantic attributes which allow to infer new classes without any additional training phase because the inner structure of the description has as property transfer knowledge.

## 1 Paper description

The paper [4] is written by C.Lampert, H.Nickisch and S. Harmeling. It tackles the problem of images classification when training and test classes are disjoint. Training classes are different from test classes.

The study attempts to answer the following questions :

- How to make prediction on unseen classes ?
- Is it possible to transfer knowledge ?
- What are the properties of semantic attributes?

To do so, we have as :

- Input : Images of different features
- Output : Unknow class (z)
- Features : color histogram, local self similarity, histogram of oriented gradients, rgSIFT descriptors, SIFT descriptors and SUFT descriptors

### 1.1 Attribute-based classification approach

The aim is to make prediction on unseen classes where no training examples is available. To do so, the authors suggested an approach which called $attribute-based\ classification$. This classification method is based upon semantic description of classes called attributes such as : color, shape, textures. However, these attributes are not directly visible but they are intrinsically related to visual information and the structure of the object. The attributes are characterized by a semantic representation that allows to make transfer learning doable. The figure 1 illustrates the high level attributes and how they can transfer knowledge between classes.

The target of attribute-based classification approach is to infer class labels by combining the predictions of many attributes. It satisfies the following two constraints by introducing a small set of high-level semantic per-class attributes :

- The amount of human effort to specify new classes should be minimal.
- Coupling data that requires only common knowledge


Figure 1: Description of high-level attributes and some classes

### 1.2 Learning problem

Let $(x_1, l_1)$,.., $(x_n, l_n) \subset$ X x Y be training samples where X is an arbitrary feature space and Y = $\{y_1, .., y_K\}$ consists of $K$ discrete classes.
The task is to learn a classifier $f$ : X $\rightarrow$ Z for a label set Z = $\{z_1, ..., z_L\}$ that is disjoint from Y.
To do so, given the situation of learning with disjoint training and test classes. If for each class z $\in$ Z and y $\in$ Y an attribute representation a $\in$ A is available, then we can learn a non-trivial classifier $\alpha$ : X $\rightarrow$ Z by transferring information between Y and Z through A.

The authors suggested two methods to do that : DAP (Direct Attribute Prediction) and IAP (Indirect Attribute Prediction)

### 1.3 Direct attribute prediction (DAP)

DAP uses in between layer of attribute variables to decouple the images from the layer of labels as it is illustrated in

figure 2. During the training, the output class label of each sample induces a deterministic labeling of the attribute layer. Consequently any supervised learning method can be used to learn per attribute parameters $\beta_m$. At the test time , attribute values can directly be inferred, and these imply output class label even for previously unseen classes.

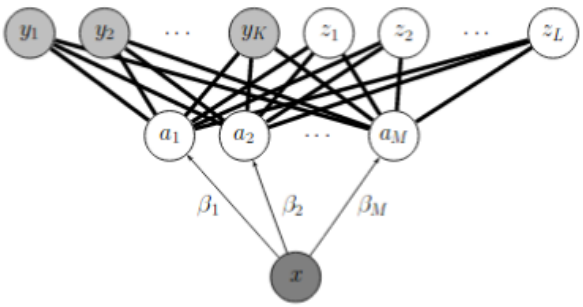


Figure 2: Direct Attribute Prediction (DAP)

They trained a non-linear SVM to predict each binary attributes $a_1, .., a_M$. The attribute of SVMs are all based on the same kernel, the sum of individual $X^2 - kernels$ for each feature, where the bandwidth parameters are fixed to the five times inverse if the median of the $X^2 - distances$ over the training samples [4].
The SVM's parameters $C$ is set to 10, which had been determined a priori by cross-validation on a subset of the training classes. Moreover, the SVM training contains 90% of the training samples and the remaining training is used as validation set for *Platt scaling* (transform the outputs of a classification model in to a probability distribution over classes) so that to obtain probability estimates.

#### 1.3.1 Probabilistic realization of DAP

Direct Attribute Classification can be implemented by combining a supervised classifier for the image-attribute or image-class prediction with a parameter free inference method to channel the information through the attribute layer [4]. The probabilistic model of the model illustrated in figure 2 is defined by the attribute representation $a^y = (a_1^y, .., a_m^y)$. We first learn $p(a_m|x)$ then we form a model for the complete image-attribute layer as :

$$p(a|x) = \prod_{m=1}^{M} p(a_m|x) \tag{1}$$

At the test time, we obtain a representation of the attribute-class layer. Then, we compute the posterior of a test class given an image :

$$p(z|x) = \sum_{a \in \{0,1\}^M} p(z|a)p(a|x) = \frac{p(z)}{p(a^z)} \prod_{m=1}^{M} p(a_m^z|x) \tag{2}$$

As a decision rule $f : \mathrm{X} \rightarrow \mathrm{Z}$ that assigns the best output class from all test classes $z_1, .., z_L$ to a test sample x, we use MAP prediction :

$$p(z|x) = f(x) = \underset{l=1,..,L}{argmax} \prod_{m=1}^{M} \frac{p(a_m^{z_l}|x)}{p(a_m^{z_l})} \tag{3}$$

### 1.4 Indirect attribute Prediction (IAP)

IAP as depicted in the figure 3, also uses the attributes to transfer knowledge between classes, but the attributes from a connecting layer between two layers of labels, one for classes that known at training time and one for classes that are not. IAP is multi-class logistic regression. At test time, the prediction for all training classes induce a labeling of the attribute layer, from which a labeling over the test classes can be inferred. They used, $L^2$ regularized-

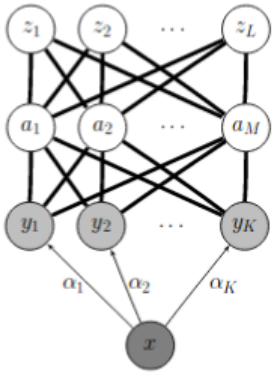


Figure 3: Indirect Attribute Prediction

multi-class- logistic regression with the same features representation as DAP. This give directly the estimate of class posteriors $P(y_k|x)$. The regularization parameter C is determined by five-fold-cross-validation

#### 1.4.1 Probabilistic realization of IAP

IAP can be implemented by combining a supervised regressor for the image-class prediction with a parameter free inference method to channel the information through the attribute layer.
In order to implement IAP, we only modify the image-attribute stage, then we compute the feature-attribute prediction model as follow :

$$p(a_m|x) = \sum_{k=1}^{K} p(a_m|y_k)p(y_k|x) \tag{4}$$

Afterwards, we carry on in the same way as in for DAP using equation 2.

## 2 Dataset description

### 2.1 Animals with attributes dataset

The dataset consists of 30475 images of 50 animals classes as illustrated in figure 4 with six pre-extracted feature representations for each image as depicted in figure 5. The animals classes are aligned with Osherson's classical class/attribute matrix [9, 10], thereby providing 85 numeric attribute values for each class. Using the shared attributes, it is possible to transfer information between different classes. The *dataset* [1] can be downloaded from the website of the authors.

[1] http://attributes.kyb.tuebingen.mpg.de/

| | | | | |
|---|---|---|---|---|
| skunk | polar bear | beaver | giraffe | *leopard* |
| lion | killer whale | bobcat | wolf | *pig* |
| fox | grizzly bear | collie | tiger | *hippopotamus* |
| ox | chihuahua | otter | cow | *seal* |
| mole | dalmatian | antelope | weasel | *persian cat* |
| sheep | spider monkey | hamster | mouse | *chimpanzee* |
| horse | blue whale | squirrel | buffalo | *rat* |
| bat | siamese cat | elephant | moose | *humpback whale* |
| zebra | rhinoceros | rabbit | walrus | *giant panda* |
| deer | german shepherd | dolphin | gorilla | *raccoon* |

Figure 4: 50 different animals classes. The last column represents the test classes (10) and the remaining are the 40 training classes

We have Animal/attribute matrix for 50 animal categories and 85 attributes. They are in the same order as in the text files "classes.txt" and "predictes.txt". The Missing values are marked by -1.

| | | | | | |
|---|---|---|---|---|---|
| black | toughskin | tail | bipedal | stalker | mountains |
| white | bulbous | horns | active | skimmer | water |
| blue | lean | claws | inactive | cave | newworld |
| brown | flippers | tusks | nocturnal | fierce | oldworld |
| gray | hands | smelly | hibernate | arctic | timid |
| orange | hooves | flies | agility | coastal | smart |
| red | longleg | hops | fish | desert | group |
| yellow | pads | swims | meat | bush | solitary |
| patches | paws | tunnels | plankton | plains | nestspot |
| spots | longneck | walks | vegetation | forest | domestic |
| stripes | chewteeth | fast | insects | fields | |
| furry | meatteeth | slow | forager | jungle | |
| hairless | buckteeth | strong | grazer | tree | |
| big | strainteeth | weak | hunter | ocean | |
| small | quadrapedal | muscle | scavenger | ground | |

Figure 5: 85 semantic attributes of the animals with attributes dataset

### 2.2 Description of images, classes and features

The authors have collected 180 000 images for 50 Osherson/Kemp animals classes from the Internet. They manually processed to remove outliers and duplicates. The remaining collection consists of 30475 images of at least 92 images for any class.

The Animals dataset, formed by combining the collected images with the semantic attribute table (matrix of 50 by 85 such that rows represent the animals and the column represent the 85 features ) that can serve as a testbed for the task of incorporating human knowledge into an object detection system. They have selected 10 test classes : chimpanzee, giant panda, hippopotamus, humpback whale, leopard, pig, raccoon, rat, seal.

They have also selected six different feature types : color histogram, local self similarity, histogram of oriented gradients, rgSIFT descriptors, SIFT descriptors and SUFT descriptors.

## 3 Description of shogun library and code

### 3.1 Shogun library

Shogun is a free, open source machine learning toolbox with a focus on large scale kernel methods and especially on support vector machine (SVM) for regression and classification problem. It's also offer a full implementation of Hidden Markov Models. It's written is $C++$ and offers interfaces for Python, Matlab, Octave, R, JAVA, Lua, Ruby and C#. It operates under Linux, Cygwin (or windows) and Mac Os operating system [1].

It support miscellaneous learning algorithms which are as follow :

- Support vector machine for classification and regression
- Dimensionality reduction algorithms such as Kernel PCA, Kernel Local Tangent Space Alignment, Multidimensional Scaling.
- Online learning algorithms such as SGD-QN (stochastic gradient descent-quasi Newton)
- Clustering : GMM, K-means
- Kernel Ridge Regression, Support Vector Regression
- Hidden Markov models
- K-Nearest Neighbors
- Linear discriminant analysis
- Kernel Perceptrons

  Originally, Shogun was developed for bioinformatics applications in order to deal with huge datasets consisting up to 10 million samples. It supports kernels for specific data (DNA sequences ) : Spectrum, Weighted Degree, Weighted Degree with Shifts [2].

  Shogun has an advantage of easily combines multiple data representation, algorithm classes that allow rapid prototyping of data pipelines and extensibility in term of new algorithm. The algorithms implemented in shogun can be executed on single machines thanks to its optimization routines [3].

  Setup steps: A.

  In order to understand the methods developed in [4] we relied on shogun toolbox [1] and the *code*[2] provided along with the paper and dataset.

### 3.2 Practical constraints and code

First of all, it's important to consider the content of the file "README-code.txt" provided along with the code. It's mentioned that the code was originally constructed to run on their own computer server and will most likely not run out-of-the-box on other installations.

[2] http://attributes.kyb.tuebingen.mpg.de/

The difficulties encountered are associated mainly with shogun toolbox which is the core of the implementation and related packages (too many) in terms of stability, dependency and adaptability to recent versions. The two methods developed by the authors are implemented in two different languages : Python for DAP (direct attribute prediction) and Matlab for IAP (Indirect attribute prediction). We tried to debug the code source file by file with different shogun's and gcc version, the error remains the same, even after making change with respect to the version of shogun. We succeeded to execute the code but we get errors from a *ccp* file from shogun. Despite the errors the code executes for 6 hours in average but at the end it doesn't return anything, The DAP directory remains empty. After that, we tried a new file proposed by the authors *"new-attributes.py"* [3] as a recent version of their code adapted to recent versions of shogun but it remains unsolved.

Despite the fact that we followed the instruction defined by the authors in "README-DAP.txt" and "README-attributes.txt" in order to run the code correctly, we didn't manage to run it successfully.

First of all, we executed "preprocess_data.py" that converts *ASCII* features from Animals with attributes features into Python Pickle format (written to ./feat directory) and it worked successfully. Secondly, we run "attributes.sh" that calls "attributes.py" after changing permission rights executing chmod as follow :
chmod u+w attributes.sh attributes.py

"attributes.py" trains attribute classifiers on train classes and put the result in ./DAP directory. It trains all the attributes classifier using all the possible features : color histogram, local self similarity, histogram of oriented gradients, rgSIFT descriptors, SIFT descriptors and SUFT descriptors. This code takes in average 6 hours to execute in a machine characterized by :

- acer Aspire V
- 6 GB DDR3 L Memory
- 128 GB SSD
- Intel HD graphics 520, up to 3148 MB Dynamic Video Memory
- Intel $Core^{TM}$ i5-6200U 2.3GHz with Turbo Boost up to 2.8 GHz

In order to identify the origin of error, we debugged the code "attributes.py" then we got an error related to "CombinedFeautures.cpp" line 149 from shogun library (/src/shogun/features/CombinedFeautures.cpp). After that, we noticed that the authors provided a new version of "attributes.py" named "new-attributes.py" which is supposed to be adapted to recent versions of shogun. After executing this new file, we encountered a problem of importing a module from shogun which is $Classifier$ that creates instance labels. Thus, instance labels are not created. Throughout the execution we identified also error related to LibSVM.cpp in shogun library and real features class function from shogun as depicted in figure 8.

Despite these errors, the code carry on its execution until the end but the directory ./DAPT remains empty, thus *DAP_eval.py* can't be executed since it has to use as input files stocked in ./DAP.
Furthermore, we got the idea with other group to try different versions of shogun. Unfortunately, it was unsuccessful.

[3] http://attributes.kyb.tuebingen.mpg.de/new-attributes.py

# 4 Results analysis

In order to evaluate the methods and results of the authors some graphs and curves such : ROC, AUC, Error Bar, Learning curve , Confusion matrix are indispensable. Thus, we planned to implement that tools on "DAP_eval.py" in order to evaluate the efficiency of DAP and for what extent DAP can be useful. However, we didn't manage to reproduce the results due to the reasons developed during previous section. Although, we describe the results provided by the authors.

They have trained the predictors for $P(a_m|x)$ on the training set $X, Y$, and used test classes attribute vector and the following equation to perform multi-class classification on test set such that classes $Z$ are different from those of $Y$.

$$f(x) = argmax_{l=1,..,L} \prod_{m=1}^{M} \frac{p(a_m^{z_l}|x)}{p(a_m^{z_l})} \tag{5}$$

The authors relies on confusion matrix to evaluate the performance of DAP and IAP. DAP results of 40.5% accuracy and IAP of 27.8% as depicted in Figure 6

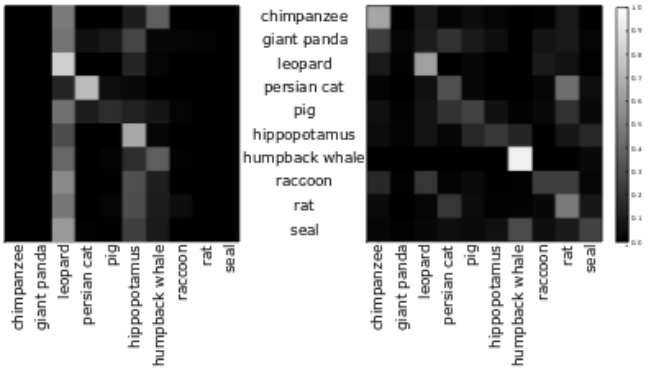


Figure 6: Confusion matrix for IAP (left), DAP (right) on Animals with attributes dataset. The diagonal represents the rate of predicted class equal to true class. The more a class has a light color the more it's true positive is high

If we compare DAP and IAP with a random classifier, we found that DAP accuracy 40.5% is drastically higher than the one of random classifier 10% (Because we have ten test classes so 10% per class).

The Figure 6 illustrates how well classifiers have been learned relying on ROC and AUC curves. We notice that all the classifier are better than random (AUC $> 0.5$) even

if several attributes performance is not much better than random. It turns out that it's important to have a large and diverse set to be able to learn more the inner structure of the objects. DAP learns also non visual attributes better than random classifier (such as *smelly*) because it's correlated with visual properties. However, according to 7 and 6 some classes contains errors since they are dark. This can be explained by the fact that these classes are more complex, then it needs sophisticated heuristics to learn the complex structure of the class, or that the attribute characterization itself is less informative for these classes.

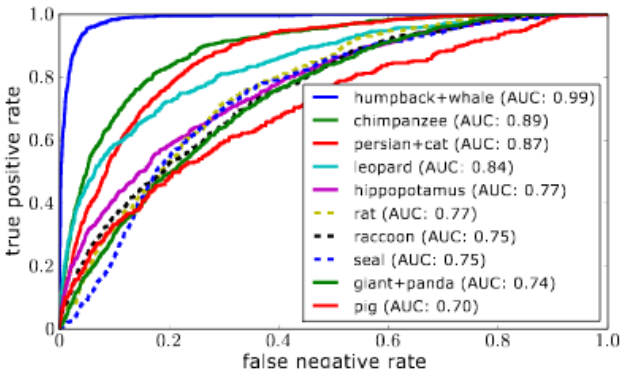


Figure 7: ROC and AUC curves for the 10 Animals with attributes test classes

## 5 Comparison with other methods

In order to compare DAP method with the state of art, the authors have first implemented One-shot learning. The accuracy of that classifier is poor. It's in the order of 14.3% for 1 training image and 18.9% for 10 training images. Secondly, they compared it to ordinary multi-class classification. It's results of 65.9% accuracy (better than 40.5% for DAP) because of the availability of more examples of the same class of test set. They split test class images into 50/50 for training and testing comparing that to the previous configuration.

## 6 Conclusion and perspective

The article comes up with an original idea which doesn't exist before 2009 for object detection with disjoint training and test classes. In standard machine learning algorithms, the aim is to generalize well on new examples unseen during the training but the classes for the training and test sets are the same. Thus, neither classical supervised learning algorithms nor clustering algorithms can be applied in this context. Unsupervised learning can't be applied since the target is to individually assign class label to test images, not only identify group sharing similarities.

DAP method needs to be improved. The first critics and improvement come in 2013 as developed in [5]. The method is called Attribute Label Embedding (ALE). It improves zero-shot recognition but the attribute prediction decreases. In [6] they improved zero-shot learning by developing two approaches : hierarchy based knowledge transfer and direct similarity-based-knowledge transfer. Then, in 2013 the same authors of the studied article come up with a novel idea to tackle zero-shot learning [7] which is the extension of [4].

Ziad Al-Halah and Rainer Stiefelhagen in 2016 [8] went further to cope with zero-shot learning by developing a hierarchical transfer model approach which has 3 aspects :

- Attribute label propagation
- Modeling the intra-attribute variations
- Guiding the transfer process to select and share the most relevant knowledge source for a novel class

To bottom line, in this project we learned how to tackle unseen class test by understanding the methods developed in the article, the provided codes and we explored related works and the state of art in order to be up-to-date with the development done since 2009. In perspective, we envisage to deepen our knowledge in zeo-shot learning and semantic transfer learning by following the state of art and reimplementing the related classifiers.

## References

[1] Sonnenburg, G. Rätsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. De Bona, A. Binder, C. Gehl and V. Franc: The SHOGUN Machine Learning Toolbox, Journal of Machine Learning Research, 11:1799−1802, June 11, 2010. In [6], the authors

[2] Shogun (toolbox). https://en.wikipedia.org/wiki/Shogun_(toolbox)

[3] Shogun website. http://shogun-toolbox.org/

[4] Christoph H. Lampert et al.Learning To Detect Unseen Object Classes by Between-Class Attribute Transfer.2009

[5] Zeynep Akata et al.Label-Embedding for Attribute-Based Classification.2013

[6] M. Rohrbach, M. Stark, and B. Schiele. Evaluating Knowledge Transfer and Zero-Shot Learning in a Large-Scale Setting. CVPR, 2011

[7] C. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. TPAMI, 2013.

[8] Ziad Al-Halah and Rainer Stiefelhagen. How to Transfer? Zero-Shot Object Recognition via Hierarchical Transfer of Semantic Attributes. arxiv, 1 April 2016

[9] D. N. Osherson, J. Stern, O. Wilkie, M. Stob, and E. E. Smith. "Default probability". Cognitive Science, 15(2), 1991

[10] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. "Learning systems of concepts with an infinite relational model". In AAAI, 2006

# Appendices

## A Installation of Shogun

In order to correctly setup-up shogun, a set of packages should be installed. For linux, follow the instruction below.

- sudo add-apt-repository ppa:shogun-toolbox/stable
  If this command doesn't work, try with these alternate commands https://github.com/shogun-toolbox/shogun/blob/develop/configs/shogun/Dockerfile
- sudo apt-get update
- sudo apt-get install libshogun17
- sudo add-apt-repository ppa:shogun-toolbox/nightly
- sudo apt-get update
- sudo apt-get install -y –no-install-recommends linux-image-extra-$(uname -r)
- sudo apt-get install -y –no-install-recommends linux-image-extra-virtual
- sudo apt-get install -y –no-install-recommends apt-transport-https ca-certificates curl software-properties-common
- curl -fsSL https://apt.dockerproject.org/gpg | sudo apt-key add -
- apt-key fingerprint 58118E89F3A912897C070ADBF76221572C52609D
- sudo apt-get update
- sudo apt-get -y install docker-engine
- apt-cache madison docker-engine
- sudo docker run hello-world
- sudo docker pull shogun/shogun-dev
- sudo docker run -it shogun/shogun-dev bash
- sudo apt-get install software-properties-common
- sudo add-apt-repository ppa:george-edison55/cmake-3.x
- sudo apt-get update
- sudo apt-get install cmake
- sudo apt-get upgrade
- sudo apt-get install exuberant-ctags
- sudo apt-get install swig

Once these above packages are installed, one could install correctly shogun as follow :

- git clone https://github.com/shogun-toolbox/shogun.git
- cd shogun
- git submodule update –init
- mkdir build
- cd build
- cmake -DENABLE_NLOPT=OFF -DPythonModular=ON ../
- make
- make install
- make clean

In your python execute the following command :

```
import shogun as sg
```

If shogun is not recognized, look at https://github.com/shogun-toolbox/docs/blob/master/INTERFACES.md

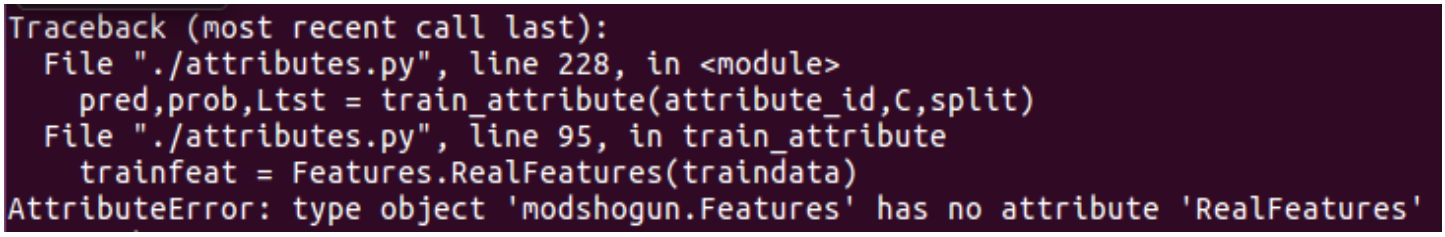


Figure 8: Modshoun features error from