



Internship Report
May 19th- August 10th, 2014

Learning a Prior for Lifelong Visual Object Categorization

ROYER Amélie
amelie.royer@ens-rennes.fr
perso.eleves.ens-rennes.fr/~aroyer

Academic year 2013–2014
ENS Rennes, 2^{ème} année
Département Informatique et Télécommunications

Advisor :
LAMPERT Christoph
CV&ML Group



Institute of Science and Technology

Institute of Sciences and Technology,
Vienna, Austria

Abstract. When facing the task of classifying object images into categories, it can be useful to consider the context of the past seen queries to infer knowledge on the future inputs, rather than only using the immediate visual information of the object. In this work, we propose two combined algorithms bringing together a state-of-the-art image classifier and an online-learning system which gradually learns the intrinsic context of an input sequence. To evaluate these algorithms, we design three methods to generate realistic sequences of queries; by “realistic”, we mean the object images are not uniformly sampled but rather part of a joint context and semantically related. Our results establish that when dealing with such a realistic sequence of images, this combined visual-contextual approach outperforms the original classifier, by reducing the ambiguity on the classes.

Keywords: object recognition, image classification, domain adaptation, semantic context, online learning, context modelling.

1 Introduction

Image classification is a major task in the field of machine learning, with applications in computer vision; given a set of object categories, also called *labels* or *classes*, and a set of images, the goal is to design a system able to correctly determine the (unique) category an image belongs to. Here we consider the multi-class form of the task, i.e. classifying images into strictly more than two categories, as opposed to binary classification (only 2 classes). In recent years, the number of object categories, as well as the size of image databases have been rapidly increasing. For example, the ILSVRC classification challenge¹ is based on a subset of the ImageNet database and contains 1.2 million training images for 1000 classes. Despite this challenging large-scale setting, current state-of-the-art classifiers, such as convolutional neural networks [8], yield extremely good accuracies.

In order to evaluate and compare such systems, one usually computes the number of wrong classifications made by the classifier on a uniformly random sample of images extracted from a subset of the image database (namely, the *testing set*). However, in a real-life setting, the data is not necessarily uniformly distributed, but rather the queries are part of a joint visual context. We can illustrate this fact by imagining a sequence of images generated by a mobile robotic system trying to categorize each object it encounters; assuming the two last seen objects are a desk and a chair, a human being could conclude that the current environment is an office, and that the next objects will probably be related; thus categories such as “computer desk” or “cupboard” have a reasonable chance to appear soon, whereas “canoe” or “giant panda” seem less likely. It is the same principle as guessing the sense of a word based on the context of a conversation.

However, current classifiers do not make this kind of contextual inference. The main objective of this work is to incorporate this principle into an existing classification system, in order to avoid predicting irrelevant classes. In fact, a classifier generally only uses visual information to make its decision; by providing it with this additional contextual material, we aim to prevent mistakes that would seem incongruous for a human being aware of the context (see Fig. 1).

Given a “realistic” query sequence, we want to learn its underlying context, and combine it with the visual information provided by a classic image classifier. Formulated in this way, the problem is strongly related to the task of domain adaption: it deals with adapting a classification system when the queries at training time (also called *sources*) are not sampled from the same distribution as the queries received during testing time (also called *targets*). For example, for spam filtering, the e-mail examples used for training (source) might be entirely different from the ones received by a particular user (target); the goal of domain adaptation is to adapt the classifier to this user without having to entirely re-train it on this new target dataset. More specifically, in our framework, we assume the images in the training set were uniformly distributed (thus the source distribution is uniform), while the unknown target distribution, \mathbb{P}_t , represents the context of the realistic sequence we use at testing time.

Our goal is to gradually model this target distribution, or at least an approximation of it, as the classifier observes new instances. This fits the framework of online learning: the classifier predicts a class for the current input image, receives a feedback, and based on this information, updates its parameters. We first tackle the fully-supervised setting of online learning, where the feedback received is the true label of the input; secondly, we propose extensions of our algorithms in the reinforcement and unsupervised settings, where the feedback is limited, or even worse, non-existent.

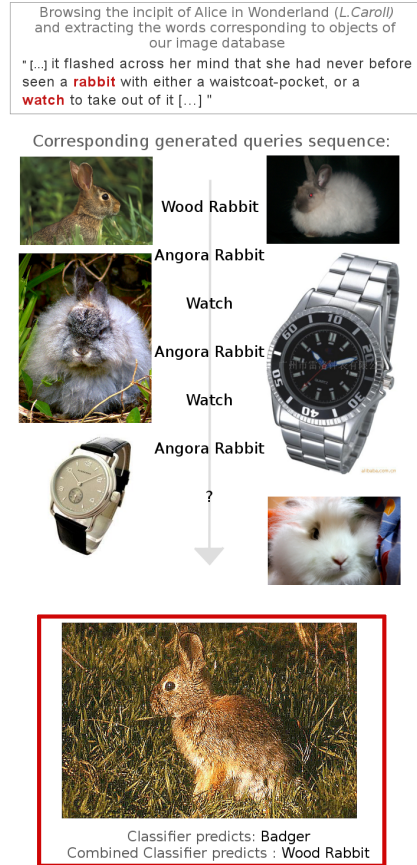


Fig. 1: Example of benefitting from context inference.

¹ ImageNet Large Scale Visual Recognition Challenge (ILSVRC), <http://image-net.org/challenges/LSVRC/2014/>.

In this work, our main contributions are two online learning algorithms that gradually build some knowledge about the context only using the past seen labels, and then associate it with a standard image classifier. They are explained in details in the third section. In the fourth section, we propose three different methods to generate “realistic” sequences of object categories, and use them for evaluating our algorithms. Finally, the last section contains experimental results and conclusions.

2 Related Work

The task we deal with is part of the general framework of online learning [2][9]. As for the specific problem of learning a context, it is related to *online sequence prediction* [4][10], or more generally *statistical language models* [18], where the objective is to learn the probability of appearance of a word in a text (or word sequence). However many of these algorithms assume a particular structure on the data, which is not the case in this work. For example, Marti et al. [11] combine a handwriting recognition system with an offline statistical natural language model in order to improve the word recognition accuracy; this is suitable when the words to recognize are organized in a regular sentence structure from the considered language.

In the framework of object recognition, several works have shown that adding a notion of semantic context in object categorization tasks helps improve the classification accuracy by reducing the ambiguity on the classes. For instance, it is applied in the task of image segmentation where the goal is to attribute a label to each pixel in order to separate the different objects appearing in a picture. In [17], semantic context information is incorporated in the image feature extraction step, which allows the classifier to use conceptual relations between the object categories appearing in the image to make better decisions (e.g. an object lying on water is correctly categorized as a boat with this method, while a non-contextual approach classifies it as a building).

In the domain of classification, recent works make use of semantic-hierarchical databases. For example, Jia et al. [5, 6] use the ImageNet database, where object categories are linked by conceptual-semantic relations in a hierarchy tree; given a small set of images of unknown categories, they infer the subtree of the hierarchy they belong to (e.g. infer “dog” from images of different dog breeds). Identifying the correct subtree reduces the ambiguity and improves the classification accuracy on these inputs. Contrary to our approach, it is not part of the online-learning framework, since they directly infer a global context (the subtree) from a set of queries, while we are interested in gradually building a model of the context when receiving the queries one at a time. Furthermore, they always assume a hierarchical structure of the database.

Finally, Shimada et al. [20] propose a combined method for hand shape recognition which is similar to our approach: the authors associate an offline recognition algorithm with an online classifier, which personalizes the system for a given user without having to retrain the original offline recognition algorithm; however the online learning algorithms used as well as the combination with the original classifier differ from ours.

3 Learning a Context

3.1 Problem Formulation

Denoting by \mathcal{X} a set of images, and \mathcal{Y} a set of classes (here we are in a multi-class setting, thus $|\mathcal{Y}| > 2$), we define a classification system as a function $f : x \in \mathcal{X} \mapsto f(x) = (f_1(x), \dots, f_{|\mathcal{Y}|}(x)) \in \mathbb{R}^{|\mathcal{Y}|}$. Given an instance $(x, y) \in \mathcal{X} \times \mathcal{Y}$, the classifier outputs a score $f_{\hat{y}}(x)$ for each class \hat{y} in \mathcal{Y} ; a higher score means the classifier considers the class is more likely to be the correct category of the input x . The final decision of f is then defined as $f^*(x) = \arg \max_{\hat{y}} f_{\hat{y}}(x)$, the goal being to predict the right class (i.e. optimally, $f^*(x) = y$). In this work, we are initially provided with an image classifier, f , and we assume the training set used to learn f follows an uniform distribution; from a domain adaptation point of view, it corresponds to the source distribution.

At test time, we consider a realistic sequence of N images and their ground truth labels, $S = (x_i, y_i)_{i=0}^{N-1} \in (\mathcal{X} \times \mathcal{Y})^N$. We call a sequence realistic if the images, and therefore their labels, share some semantic-contextual

relations (for example a class A and a class B belonging to the same context would tend to often appear together). Note that these relations do not need to represent reality: for instance, in the example from *Alice in Wonderland* in Fig. 1, the context of the sequence is composed of watches and rabbits, which is generally not a logical association for a human being, but it is considered a realistic sequence in this work because it is consistent in the context of *Alice in Wonderland*.

Because of this “realism” hypothesis, we believe the queries are not uniformly distributed but follow some unknown target distribution, \mathbb{P}_t , that we informally refer to as the context of the sequence. More precisely, we work in the setting of prior probability shift [16, Chapter 1]: this means the distribution of the labels between training time and testing time changes, but the class conditional densities of the images are the same. Formally, $\forall x \in \mathcal{X}, \forall y \in \mathcal{Y}, \mathbb{P}_s(y) \neq \mathbb{P}_t(y)$ and $\mathbb{P}_s(x|y) = \mathbb{P}_t(x|y)$. Finally, we do not make any additional parametric assumption about \mathbb{P}_t : the shape of the distribution is not restricted, because this would not fit the framework of realistic query sequences.

We aim to create a combined classifier, g , which incrementally learns the underlying context in S , and associates this information with the output of the original classifier, f . Our algorithms follow a typical online learning scheme: the combined classifier, g , receives the queries one at a time, and updates its knowledge of the context at each round. The online learning literature often distinguishes three different feedback settings (fully-supervised, reinforcement, unsupervised) that we describe in Fig. 2 below. Note that even though our context model is updated and changes over time, the true context itself (\mathbb{P}_t) is fixed.

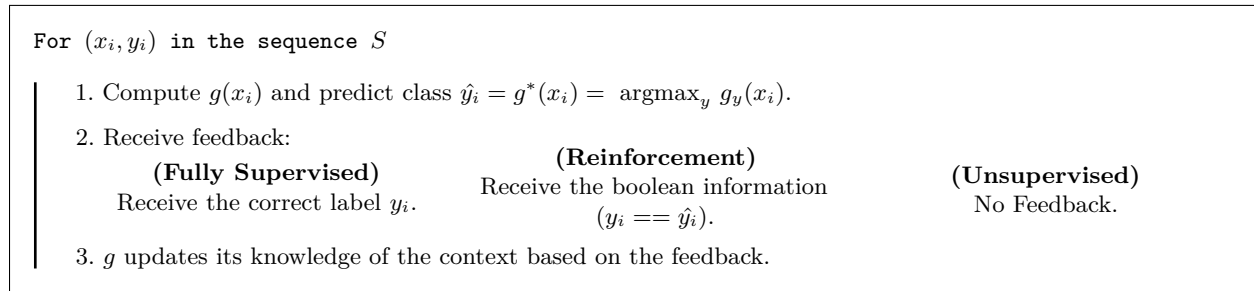


Fig. 2: Feedback scenarios for online learning

Finally, we introduce the notion of top- k accuracy, which we use to evaluate the classification quality. It assesses the fact that the correct label is among the k first labels output by the system. Formally, the top- k accuracy of a classifier f on the sequence $S = (x_i, y_i)_{i=0 \dots N-1}$ is defined as:

$$\operatorname{acc}_k(f) = \frac{1}{N} \sum_{i=0}^{N-1} \ell_k(x_i, y_i), \tag{1}$$

where $\ell_k(x_i, y_i) \triangleq \begin{cases} 1 & \text{if } y_i \text{ is among the } k \text{ labels with highest scores in } f(x_i), \\ 0 & \text{otherwise.} \end{cases}$

We conjointly define the top- k error rate which is simply the dual notion, i.e. $\operatorname{err}_k = 1 - \operatorname{acc}_k$.

In the next subsection, we propose a model that learns a representation of the context only using the past seen labels, as well as a reinforcement and unsupervised variant. Next, we introduce a more general algorithm (and its reinforcement counterpart), which is inspired from classic online learning methods, and uses the output of the classifier f to learn a confidence weight for each class; the last subsection contains two extensions of the previous models. For each method, we first introduce the general framework, then we detail how we learn context information, and finally we show how to combine it with the original classifier, f , to obtain the final context-sensitive classifier g .

3.2 A Static Probabilistic Model

Motivation. Our first approach to tackle the problem is to define a context as a probability distribution over the set of labels \mathcal{Y} , and to gradually estimate it by using the information of the past seen labels: given a past sequence of labels \mathbf{y}_0^{n-1} , what is the chance of seeing some label y at time n ?

Formally, we build a probabilistic “context model”: $\theta : \mathcal{Y}^* \rightarrow [0, 1]^{|\mathcal{Y}|}$; for each label y , it outputs the probability of seeing y after the current past sequence of labels, i.e. $\theta_y(\mathbf{y}_0^{n-1}) = \mathbb{P}(y | \mathbf{y}_0^{n-1})$. Since we build θ through an online learning procedure, we denote by θ^n the context model at round n , but for simplicity we often keep this index implicit: for example, $\theta(\mathbf{y}_0^{i-1})$ implicitly refers to the context model at round i . The same yields for the combined classifier g .

Combining a Context Model with a Classifier. In this paragraph, we assume the initial classifier f has probabilistic output, i.e. $f_y(x) = \mathbb{P}(y|x) \in [0; 1]$. There exists several classification algorithms with probabilistic outputs, or simply real-valued outputs that we can transform into probabilities, so this hypothesis is not too restrictive in practice.

Let $(x_i, y_i)_{i=0\dots(n-1)}$ be the sequence of queries received up to round n , and x_n be the next input image. We assume we have a classifier f and a context model θ , both with probabilistic outputs. Using Bayes’ rule, we can link $f_y(x) = \mathbb{P}_s(y|x)$ with the estimated probability of seeing a given label in the sequence, $\mathbb{P}_t(y)$, which is exactly the output of the context model.

The original classifier, f does not use the context to make its decision, it just assumes the labels are uniformly distributed, i.e. they follow the source distribution $\mathbb{P}_s(y) = \frac{1}{|\mathcal{Y}|}$; hence applying Bayes’ rule yields:

$$\forall y \in \mathcal{Y}, f(x_n)_y = \mathbb{P}_s(y|x_n) = \frac{\mathbb{P}_s(x_n|y) \times \frac{1}{|\mathcal{Y}|}}{\mathbb{P}_s(x_n)} \quad (2)$$

However, in reality, the context of the sequence of queries at testing time is represented by the target distribution, \mathbb{P}_t . Assuming our context model is an approximation of the real context of the query sequence, we have $\mathbb{P}_t(y) \approx \theta_y(\mathbf{y}_0^{n-1})$. Therefore Bayes’ rule yields the following:

$$\forall y \in \mathcal{Y}, \mathbb{P}_t(y|x_n) = \frac{\mathbb{P}_t(x_n|y) \times \theta_y(\mathbf{y}_0^{n-1})}{\mathbb{P}_t(x_n)} \quad (3)$$

As mentioned in the beginning of the section, we are in the prior probability shift setting of domain adaptation, which means $\forall x \in \mathcal{X}, \forall y \in \mathcal{Y}, \mathbb{P}_s(y) \neq \mathbb{P}_t(y)$ and $\mathbb{P}_s(x|y) = \mathbb{P}_t(x|y)$. As for the distributions of the images themselves, $\mathbb{P}_s(x)$ and $\mathbb{P}_t(x)$, they do not matter here, since we are only interested in the class that maximizes the conditional probability according to the label y . Finally, by combining (2) and (3), and getting rid of the terms that do not depend on y , we obtain :

$$\forall y \in \mathcal{Y}, \mathbb{P}_t(y|x_n) \propto f_y(x_n) \times \theta_y(\mathbf{y}_0^{n-1})$$

This conditional probability distribution over the labels in the target setting defines the combined classifier at round n , $g : x \in \mathcal{X} \mapsto (\mathbb{P}_t(y|x_n))_{y \in \mathcal{Y}} \in \mathbb{R}^{|\mathcal{Y}|}$; it depends both on the initial image classifier and on the context model. In the three next paragraphs, we propose an algorithm to build the context model θ for the three settings of online learning: fully-supervised, reinforcement, unsupervised.

Fully Supervised framework. We build a probabilistic context model θ by using a very natural estimation rule; that is, to deduce the probability of seeing a label from its past frequency. Formally, we define θ at round n as:

$$\theta_y(\mathbf{y}_0^{n-1}) = \mathbb{P}(y|\mathbf{y}_0^{n-1}) \triangleq \frac{w_{n-1}(y) + \varepsilon}{n + \varepsilon|\mathcal{Y}|}$$

w_n is a vector counting the number of appearances of each label up to round n . The ε constant is a smoothing term, which prevents giving a 0-weight to labels that have yet to appear in the sequence. Finally, we obtain a probability distribution from the vector w_n by dividing it by the sum of its components. We refer to this approach as *Multinomial Model*. It is summed up in Fig. 3.

In practice we take $\varepsilon = \frac{1}{2}$; first because it achieves good experimental results, and secondly because in this case the algorithm is a multi-class extension of the Krichevsky-Trofimov estimator [7], for which there exists theoretical bound results [2, Chapter 9].

From a probabilistic point of view, this definition of θ is equivalent to saying that the labels are distributed according to a multinomial distribution of parameter \mathbf{p} , which we update at each round by maximizing the likelihood of a Dirichlet prior distribution, whose parameters are the number of appearances of each label.

Data: $S = (\mathbf{x}_0^{N-1}, \mathbf{y}_0^{N-1})$, classifier f , smooth term ε
init: $\forall y, w_{-1}(y) \leftarrow 0$;
for $n \leftarrow 0$ **to** $N - 1$ **do**
 $\theta_y(\mathbf{y}_0^{n-1}) \leftarrow \frac{w_{n-1}(y) + \varepsilon}{n + \varepsilon|\mathcal{Y}|}$;
 predict
 $\hat{y}_n = g^*(x_n) = \operatorname{argmax}_y (\theta_y(\mathbf{y}_0^{n-1}) \times f_y(x_n))$;
 receive y_n ;
 $\forall y, w_n(y) \leftarrow \begin{cases} w_{n-1}(y) + 1, & \text{if } y = y_n \\ w_{n-1}(y), & \text{otherwise} \end{cases}$
end

Fig. 3: Supervised Multinomial Model

Reinforcement Framework. We extend the previous context model to the reinforcement setting of online learning. In this case, the algorithm does not receive the correct label y_n , but only knows if its prediction, $\hat{y}_n = g^*(x_n)$, was correct or not. Thus the previous multinomial update rule is not possible, because it requires to know the true label, while here the system only has this information in the case where its prediction is correct ($y_n == \hat{y}_n$). This remark leads to the following update rule:

predict $\hat{y}_n = g^*(x_n) = \operatorname{argmax}_y (\theta_y(\mathbf{y}_0^{n-1}) \times f_y(x_n))$;
receive $correct_n = (y_n == \hat{y}_n)$;
if $correct_n$ **then**
 $\forall y, w_n(y) \leftarrow \begin{cases} w_{n-1}(y) + 1, & \text{if } y = y_n \\ w_{n-1}(y), & \text{otherwise} \end{cases}$
else
 $\forall y, w_n(y) \leftarrow \begin{cases} w_{n-1}(y), & \text{if } y = \hat{y}_n \\ w_{n-1}(y) + \frac{1}{|\mathcal{Y}| - 1}, & \text{otherwise} \end{cases}$
end

Fig. 4: Reinforcement Multinomial Model

- If the prediction \hat{y}_n is equal to the correct label y_n , apply the ordinary update rule of the multinomial model.
- Otherwise, we only know that the predicted label \hat{y}_n is wrong, and we assume all other classes have an equal chance to be the correct one; therefore we increase the weights of every class, apart from \hat{y}_n , by the same amount.

Note that contrary to the fully supervised case, this update rule depends on the original classifier f (because it uses the prediction $\hat{y}_n = g^*(x_n) = \operatorname{argmax}_y f_y(x_n) \times \theta_y(\mathbf{y}_0^{n-1})$ for the update). Yet, the better the classifier f , the greater the chance that \hat{y}_n is correct and that the update is the same as the multinomial model. Therefore when the original classifier f has good accuracies, the reinforcement learning yields closer results to the multinomial model. The new update rule is described in Fig. 4.

Unsupervised Framework. In this last setting, the classifier does not receive any information from the environment after its prediction. We choose to assume the model is always correct, i.e. we always update with the predicted label. The new update rule is described in Fig. 5.

predict $\hat{y}_n = g^*(x_n) = \operatorname{argmax}_y g_y(x_n)$;
 $\forall y, w_n(y) \leftarrow \begin{cases} w_{n-1}(y) + 1, & \text{if } y = \hat{y}_n \\ w_{n-1}(y), & \text{otherwise} \end{cases}$

Fig. 5: Unsupervised Multinomial Model

Finally, we can make the same remark as for the reinforcement setting: the better the original classifier f , the smaller the difference in results with the fully supervised setting.

3.3 A More General Online Learning Approach

In this subsection we present a different approach for learning a context and building the combined classifier g . The previous algorithm uses the frequencies of the past seen queries to model an approximation of the unknown target distribution of the labels.

The drawback of this method is that it is, in a sense, “static”: if the label sequences we consider were simply generated by independently sampling labels from the target distribution $\mathbb{P}_t(y)$ fixed during the whole sampling process, then the multinomial model would approximate it perfectly when the size of the sequence tends to infinity (by the laws of large numbers, the relative frequency of a label y converges to its exact probability $\mathbb{P}_t(y)$). However, we do not make any assumption about the generative process of our realistic query sequences. For example in the case of a Markovian generative process, we cannot assume the labels distribution $\mathbb{P}_t(y)$ is fixed during the whole generative process, because the probability of seeing a given label depends on the state of the Markov chain we are currently in.

Therefore, we propose a more general context modelling algorithm which, contrary to the multinomial model, does not have such a static behaviour, and should therefore be better fitted when the generative process of the sequence is not fixed over time.

This second approach tackles the problem as a task of “expert weighting”. This refers to a wide category of problems where we are provided with a set of experts (i.e. a set of functions which output a certain decision) and the goal is to take the best decision given the experts advice. A commonly used algorithm is the “weighted majority algorithm”: it aims at learning a weight for each expert and the final decision is a weighted linear combination of the experts’ output.

In our framework, we define experts as follow: given an image x , for each class k , there is an expert E_k which votes for class k with score $f_k(x)$ (the output of the original classifier for class k), and votes for other classes with score 0 (i.e. no preference). If w is the weight vector learned by the algorithm, then the linear combination of the experts’ outputs yields the following:

$$g(x) = \sum_{k=1}^{|\mathcal{Y}|} w_k E_k(x) = w \odot f(x), \quad \text{where } \odot \text{ is the component-wise multiplication of two vectors.}$$

With this point of view, the weights we learn represent how much confidence we put in an expert given the sequence of past seen queries. Note that each expert gives a non-zero score to its own class only; intuitively, to fill in these zero entries we would need an information on the similarity between two classes, which we do not have (and which, in a way, would correspond to a global semantic context between the object categories).

Fully Supervised. In order to learn the weight vector w , we draw inspiration from the family of multiplicative update algorithms (we update w by multiplying its components at each round). More precisely, our algorithm resembles WINNOWER [9]: it is conservative, i.e. we only make an update when the classifier makes a mistake, and we only modify the weights of the true label y_n and the wrongly predicted label \hat{y}_n . However, apart from this, the algorithm differs from the classic WINNOWER.

The intuition of our update rule is that when a mistake is made, the context knowledge (*here represented by the weight vector w*) should “disagree” with the decision of the original classifier f (*here represented by the experts*), to show that the visual information was not sufficient. More precisely, suppose the classifier g makes a mistake ($\hat{y}_n \neq y_n$); it means that the score of the predicted label is greater than the one of the true label: $g(\hat{y}_n) = w_{\hat{y}_n} f_{\hat{y}_n}(x) > g(y_n)$.

```

Data:  $S = (\mathbf{x}_0^{N-1}, \mathbf{y}_0^{N-1}), f, \alpha$ 
init:  $\forall y, w_{-1}(y) \leftarrow \frac{1}{|\mathcal{Y}|}$ ;
for  $n \leftarrow 0$  to  $N - 1$  do
   $g(x_n) \leftarrow w_n \odot f(x_n)$ ;
  predict  $\hat{y}_n = \operatorname{argmax}_y g_y(x_n)$ ;
  receive  $y_n$ ;
  if  $y_n \neq \hat{y}_n$  then
     $\forall y, w_n(y) \leftarrow$ 
     $\begin{cases} w_{n-1}(y) \times e^{\alpha(1-f_y(x_n))}, & \text{if } y = y_n \\ w_{n-1}(y) \times e^{-\alpha(1-f_y(x_n))}, & \text{if } y = \hat{y}_n \\ w_{n-1}(y), & \text{otherwise} \end{cases}$ 
  else
     $w_n = w_{n-1}$ 
  end
end

```

Fig. 6: Supervised Weighting Model

First we need to update the weight of the true label; there are two cases: if the original classifier score $f_{y_n}(x_n)$ was low, this means we must strongly increase the weight w_{y_n} , so that it counterbalances the low score given by f . Conversely, if $f_{y_n}(x_n)$ was already high, we do not need to increase the weight w_{y_n} as much, since it means the classifier f was quite confident in this class and thus it was not “too wrong”.

The same kind of reasoning applies for the update of the weight $w_{\hat{y}_n}$ of the predicted label. We finally observe that the update is always in the “opposite direction” of the decision of the original classifier, and that its amplitude depends on the score of the classifier. To express this, we take $e^{\alpha(1-f_y(x_n))}$ as update coefficient. $\alpha > 0$ is called the *learning rate*; in practice we simply take $\alpha = 1$. We call the resulting model *Weighting model*; it is summed up in Fig. 6.

Reinforcement Approach. A drawback of the Weighting model is that we need both the predicted and true label to update it; for this reason it is not possible to extend it to the unsupervised framework, as we always lack the information of the true label. For the reinforcement approach, we keep the same update rule as in the fully-supervised setting but we only do the positive update (update of the true label) when we are correct, and the negative update (update of the wrongly predicted label) otherwise. This update rule is presented in Fig. 7.

```

receive  $correct_n = (\hat{y}_n == y_n)$ ;
if  $correct_n$  then
     $\forall y, w_n(y) \leftarrow \begin{cases} w_{n-1}(y) \times e^{\alpha(1-f_y(x_n))}, & \text{if } y = y_n \\ w_{n-1}(y), & \text{otherwise} \end{cases}$ 
else
     $\forall y, w_n(y) \leftarrow \begin{cases} w_{n-1}(y) \times e^{-\alpha(1-f_y(x_n))}, & \text{if } y = \hat{y}_n \\ w_{n-1}(y), & \text{otherwise} \end{cases}$ 
end

```

Fig. 7: Reinforcement Weighting Model

Comparison to the Multinomial Model. As we stated previously, the main difference with the Multinomial model is that the Weighting update rule is conservative: when there is no mistake, the model doesn’t change its parameters. Moreover, this update depends on the original classifier f directly, and we penalize the predicted label in the case of a wrong prediction. Furthermore, note that contrary to the Multinomial approach, the original classifier f does not need to have probabilistic outputs. Apart from this, we can unify both methods; in fact, recall that we defined the combined classifier g for the Multinomial model in Sec. 3.2 as follow:

$$\forall y \in \mathcal{Y}, g_y(x_n) = \mathbb{P}_t(y|x_n) \propto f_y(x_n) \times \theta_y(\mathbf{y}_0^{n-1}); \text{ which we rewrite in: } \boxed{g(x_n) = w \odot f(x_n)} \quad (4)$$

The second equation exactly corresponds to the definition of the combined classifier g in our second model. Therefore a combined classifier with a probabilistic context model θ (our first approach) is a sub-case of the combined approach with a weighting algorithm (our second approach); and in this case $w_y^n = \theta_y^n(\mathbf{y}_0^{n-1})$.

3.4 Extensions

In this subsection we present two extensions to the previous models which are more specific to the databases we use in our experiments.

Unlearning Multinomial. As we mentioned, a drawback of the multinomial model is its “static” behaviour: in practice the labels seen at the very beginning of the sequence have the same influence in the current context model as newer labels, because the update rule is static. However this behaviour is harmful if, for example, the label sequence was generated by a Markovian process, since in this case only the last seen labels are important. In the *Unlearning Multinomial Model*, we add an “unlearning” step when computing the labels’ frequencies, in order to “forget” the oldest labels. In practice, we introduce a sliding window of a certain length L on the sequence: we keep the same update rule as the Multinomial model, but we only consider the frequencies of the labels among the last seen L labels (in practice $L = 100$), thus gradually suppressing the influence of the oldest labels.

Using the Hierarchy as Additional Information. In the introduction, we gave the example of “desk”, “chair” and “computer”, which are three linked categories in real life, if the environment is an office for example. However, up to now, we have not used such real life assumptions about object categories to model a context. In this paragraph, we assume our image databases are hierarchically structured, which means the labels are connected to each other by semantic relations; we incorporate this real life hierarchy information in the Multinomial and Weighting models.

The Hierarchical Multinomial model uses the same update rule as the fully-supervised Multinomial, but when updating the weight w_x for a label x , it propagates it to all labels y , with some coefficient $p(x, y)$. Figure 8 sums up how we define this coefficient: in the up-phase, we recursively compute a weight between x and its parents depending on their number of children. When reaching the lowest common ancestor of x and y , we begin the down phase, which is just a multiplication, and this finally yields $p(x, y)$. These coefficients are precomputed for efficiency reasons.

For the Hierarchical Weighting model, we develop an idea close to the COMMITTE algorithm [12]. We use the same update rule as the fully-supervised Weighting model, with an additional update for all other labels. The idea is to increase their weight if they are close to the true label, and to down-weight it if they are close to the wrongly predicted one. For this, we use our propagation weight $p(x, y)$ as a similarity measure between the labels (although not necessarily symmetric). This motivates the following supplementary update rule: $\forall y \notin \{y_n, \hat{y}_n\}, w_{n-1}(y) \times e^{\beta(p(y_n, y) - p(\hat{y}_n, y))}$. In our experiments, we take $\beta = 6$ because it achieves the best results, however finding an optimal value would require a more rigorous model selection.

4 Generating “Realistic” Sequences

We aim to show that when the sequence of query images sent to a classification system is somewhat structured in a joint semantic context, then introducing a prior knowledge over the label distribution shows better results than only relying on the visual information provided by the image classifier.

To highlight this fact, we first need to generate “realistic” ordered label sequences, *i.e.* sequences such that two close labels are semantically related. For each label, an image from the corresponding class can then be sampled, which results in a realistic sequence of input images. We now propose three generation processes (1 real life + 2 synthetics) for such sequences.

4.1 The ImageNet Database

In practice, our databases are subsets of the hierarchical ImageNet database, in which the object categories are linked together by semantic-conceptual relations. More precisely, each dataset contains approximately 2000 object categories: 1000 of them are used as labels for the classification task and are leaves in the hierarchy tree (*i.e.* $|\mathcal{Y}| = |\{\text{leaves}\}| = 1000$), while the rest is only used to shape the tree structure. Because of the semantic relations, lower nodes are sub-categories of higher nodes (*e.g.* “pea” is a sup-category of “green pea”), and the higher a node in the tree, the vaguer the category (the root node is “entity”). For the same reason, two nodes close in the hierarchy are conceptually related.

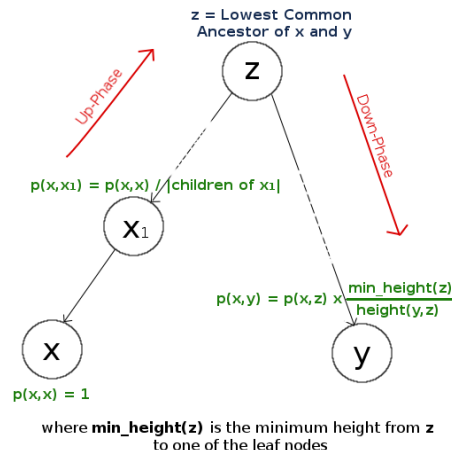


Fig. 8: Definition of $p(x, y)$

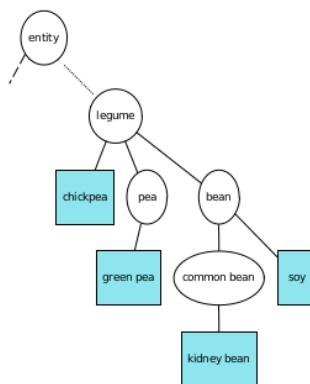


Fig. 9: Example of Subtree in the hierarchy. Categories for the classification task (*i.e.* leaves) are represented by blue boxes.

4.2 Generative Processes for Realistic Label Sequences

TXT Database. We generate the first dataset from real-life data: we browse a set of English books (classic literature taken from the Project Gutenberg²), and sequentially extract each word corresponding to an object category of the classification task. Beforehand, we apply a part-of-speech tagger, using the Python Natural Language toolkit [1, Chapter 5], in order to only keep the nouns in the text. In fact, object categories in classification tasks are, to our knowledge, grammatically nouns; this preprocessing step prevents us from extracting unwanted homonyms as object categories (e.g. to watch / a watch, he saw / a saw).

The main drawback of this generation method, is that the number of retrieved object categories depends on the level of details of the classification task. For instance, categories such as “black-footed ferret” do not appear often in usual books, while larger categories like “dog” are easier to retrieve. We make use of the semantic hierarchical structure of our databases to counteract this disadvantage. If the text contains a word corresponding to one of the high-level nodes of the hierarchy tree, we extract it and randomly sample one of the leaves of the corresponding subtree (e.g. when encountering the word “legume”, we randomly choose one of the leaves in the corresponding subtree, such as “soy” for example). Furthermore, each object category is given in the form of a synset, i.e. a list of semantically related words (for example [‘wood rabbit’, ‘cottontail’, ‘cottontail rabbit’]), which provides us with more words to retrieve for each class. This hierarchical structure is not necessary for the label sequence generation, but here it helps us generating longer sequence of words, containing object categories which usually do not often appear in texts.

KS Database. For the two next generation processes, we assume we are provided with a semantic-hierarchical image database (in our case, the ImageNet database). Taking advantage of this hierarchy, we define a distance over labels as follow :

$$d(y_1, y_2) = \text{height}(\text{lca}(y_1, y_2))$$

where $\text{height}(x)$ is the maximum path length from the node x to any of the labels (leaf nodes) in the hierarchy, and $\text{lca}(x, y)$ is the lowest common ancestor of x and y . The labels are then projected on a 2D-grid structure using Kernelized Sorting [15], so that their placement respects this distance measure (see Fig. 10). In practice we generate label sequences with a random walk on the grid (with 20% probability to go up, down, left, right or stay in the current position).

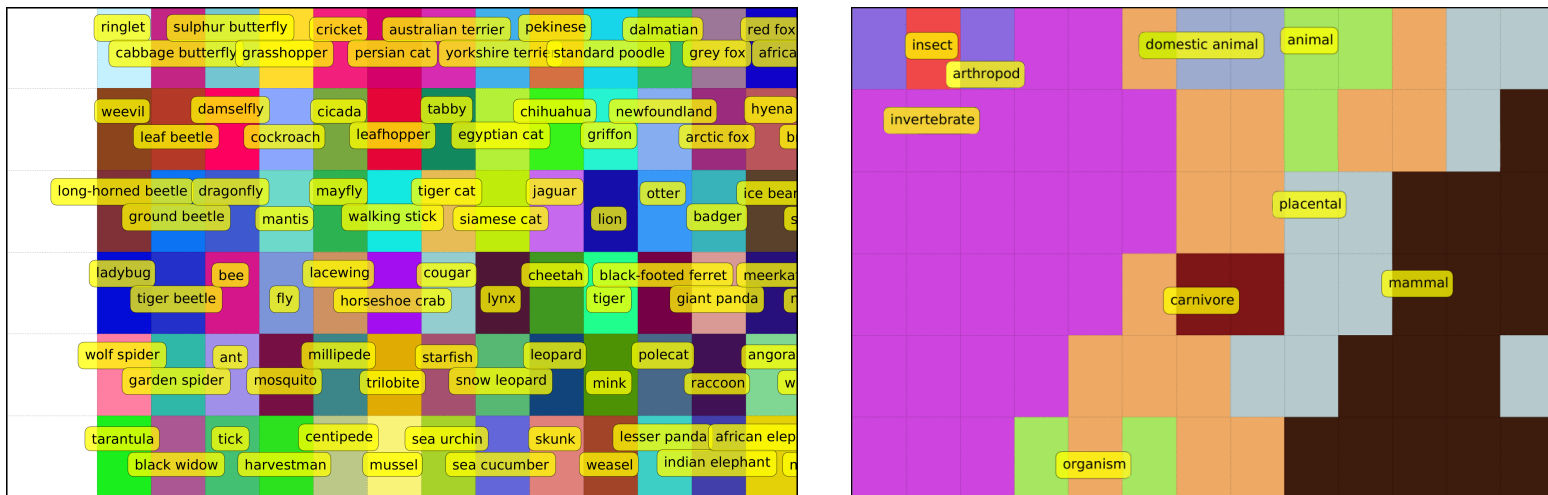


Fig. 10: A subset example of a 2D-grid we obtained (*on the left*) and the corresponding colored clustering for labels sharing the same subtree (of height 4) (*on the right*)

² Project Gutenberg : Free e-books, http://www.gutenberg.org/wiki/Main_Page.

MDS Database. This dataset is based on the same idea as the previous one, but uses a different projection method, namely Multi Dimensional Scaling [3] (Scikit-learn implementation [13]), which is a classic dimension reduction method focusing on respecting a distance measure. As a result, each class is associated to a point in \mathbb{R}^2 and a label sequence is generated by doing a random walk on the k nearest neighbours of a point (for the experiments we took $k = 9$ neighbours, including the current point itself in the next position possibilities). See Fig. 11 for an example.

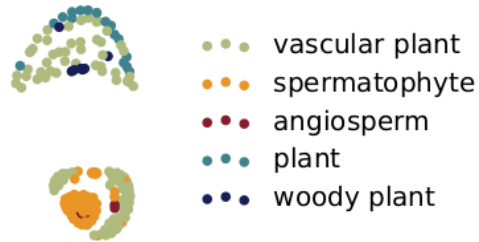


Fig. 11: A subset example of a projection with color clustering for labels sharing the same subtree (of height 4)

4.3 Structure of Typical Sequences

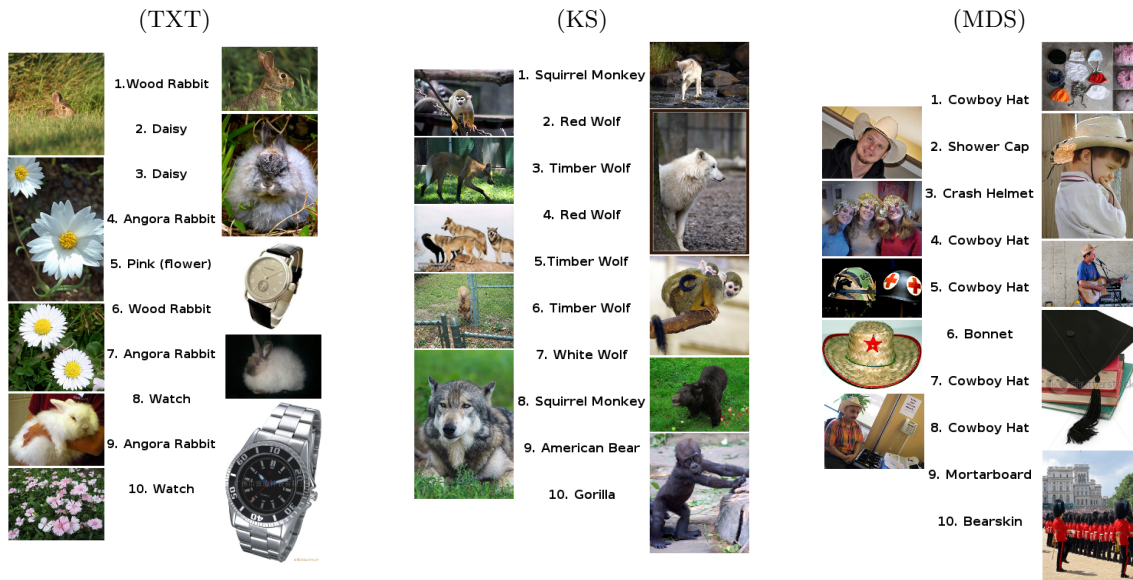


Fig. 12: Typical sequence structure for TXT, KS and MDS

From the structure of typical generated sequences (see Fig. 12), we observe each database has its own characteristic. The MDS generated sequences tend to stay in the neighbourhood of one category (in the example, hats): this comes from the fact that the 2D plot is usually very clustered, and the clusters themselves are far away from one another. Therefore, when browsing the plot by nearest neighbours we generally stay around the same point. On the contrary, the KS generative process can browse the whole 2D-grid. Therefore the generated sequences tend to be “locally realistic” (seeing many images of wolves at some point for example), but taken as a whole, the changes in categories are more frequent (in the example, we switch from monkey to wolf to bear and back to monkey). Finally, the TXT database is a sort of trade-off between those two characteristics: it also has a limited vocabulary, but is not restricted to one “theme” like the MDS database. Furthermore we observe words burstiness which is a natural structure in texts: it means that once a word appears in a text, it tends to appear a lot in the future, even if on average it is a rare word (in the example, the word “rabbit” has such a behaviour).

To have a better understanding of the generated sequences’ behaviour, we plot the perplexity of the Multinomial context model on the different databases. The *perplexity* is a measure of a language model performance often used in information theory [19], it assesses how well such a model represents a word sequence and how confident it is in its predictions. Transposing this notion in our framework where words are

object categories and texts are label sequences, we define the perplexity of a probabilistic context model θ on a sequence \mathbf{y}_0^{n-1} as:

$$\text{Perp}(\mathbf{y}_0^{n-1}) \triangleq \sqrt[n]{\frac{1}{\mathbb{P}(\mathbf{y}_0^{n-1})}} = \left(\prod_{i=0}^{n-1} \mathbb{P}(y_i | \mathbf{y}_0^{i-1}) \right)^{-\frac{1}{n}} = \left(\prod_{i=0}^{n-1} \theta_{y_i}^i(\mathbf{y}_0^{i-1}) \right)^{-\frac{1}{n}}$$

Intuitively, a good context model should assign a high (if not the highest) probability to the correct next label in the sequence. Therefore, a lower perplexity means the model successfully grasps the structure of the sequence; a perfect model would always give a probability of 1 to the correct next label, and thus achieve a perplexity of 1. On the contrary, a non-informative model does not make any assumptions and models an uniform distribution; in this case the perplexity is equal to $|\mathcal{Y}|$. It is the starting point of our Multinomial context model. We first present the curves for the TXT and MDS databases (Fig. 13); for both, the perplexity decreases very fast in the beginning, and more slowly afterwards. This shows that the context model gets better over time, until reaching an “equilibrium state”.

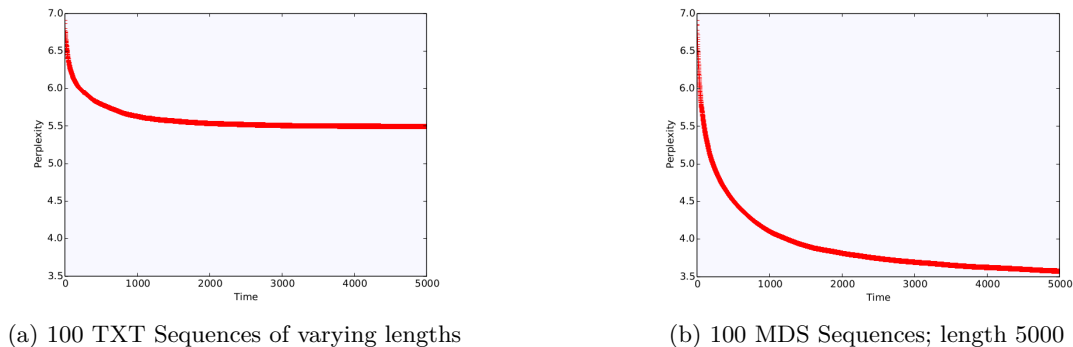


Fig. 13: Mean Log-Perplexity on for the Multinomial Model TXT and MDS sequences

However for the KS database (Fig. 14 (a)), the perplexity initially decreases but then starts growing. We reckon it comes from the fact that KS sequences are generated from a random walk on a grid, and may drift far away from the initial point; thus the labels on the beginning of the sequence are not really part of the current context, yet the Multinomial model still counts them with the same weight as the others. To tackle this issue, we use the Unlearning Multinomial which only consider a subset of the last seen labels (sliding window of size 100). As shown in Fig. 14 (b), its perplexity has a better curve progression.

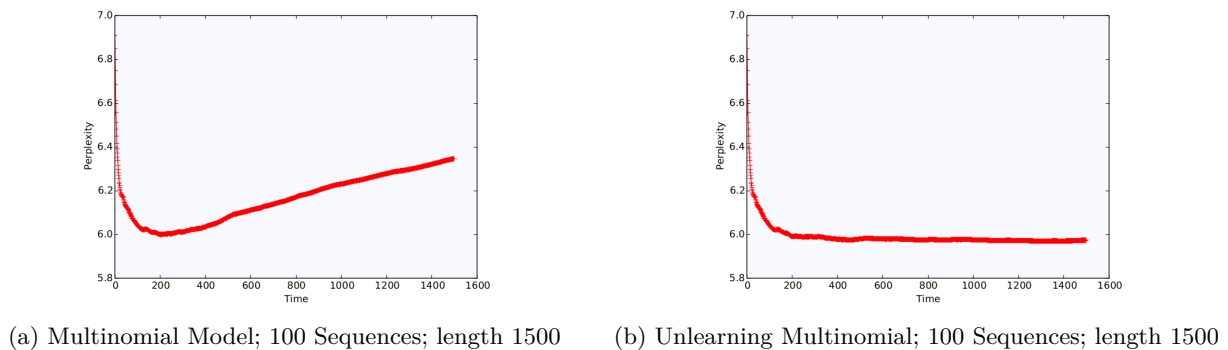


Fig. 14: Mean Log-Perplexity on KS sequences

5 Experiments

5.1 Experimental Setting

We conducted the experiments on two image databases that are subsets of the ImageNet hierarchy (ILSVRC2010 and ILSVRC2012). The first step is to choose the original classifier f . We do the experiments in two settings to show that the results do not depend on the chosen classifier: first, we use the CCV³ toolkit and its pretrained ImageNet classifiers. This system uses convolutional neural networks and already achieves state-of-the-art accuracies.

Secondly, we use the JSKD⁴ toolkit to train a classifier on the databases. Using the training set (1.2 million images), we learn a multi-class SVM (Support Vector Machine) classifier: for each label y we train a SVM binary classifier f_y that separates the class y from all the other classes. The corresponding multi-class classifier is given by $f : x \in \mathcal{X} \mapsto (f_y(x))_{y \in \mathcal{Y}}$. However, SVMs do not output probabilistic scores, which we need to build the Multinomial model; therefore we apply Platt Scaling [14], using 50K training images that were left out during the training step as a cross-validation set: for each one of the binary classifiers f_y , we learn the parameters of a Sigmoid function which, when applied to the scores output by this classifier, transform them into probabilities.

As for the classification task itself, we use 4 databases for label sequences: from the realistic label sequences generation methods presented above, we use 100 TXT sequences (lengths vary approximately from 400 to 20000), 100 KS sequences of length 1500 and 100 MDS sequences of length 1500. The fourth set of sequences contains 100 random label sequences of length 1500; it is denoted by RND and is used to show the effect of our approach when there is no explicit context information in the image sequence.

The images we use come from the validation dataset (50K images) of the corresponding image database. For each label y_i we encounter in the labels sequence, we randomly sample an image x_i among the images of the corresponding class. This generates the testing sequence $S = (x_i, y_i)$.

5.2 Results

In this section, we present and analyze the results obtained with the aforementioned experimental settings. In Tab. 5, we present the top-1 and top-5 error rates (lower is better) for the CCV classifier combined with the fully-supervised algorithms (Multinomial, Weighting and their extensions: Unlearning Multinomial, Hierarchy Multinomial, Hierarchy Weighting) on the ILSVRC2012 validation dataset. As defined in Sec. 3.1, the top- k error rate is the mean number of times the correct label of an image was *not* in the k labels with highest score output by the classifier.

Table 1: Mean Top-1 and Top-5 Error Rates(%) (ILSVRC2012 dataset; CCV classifier; fully-supervised)

Classifier \ Seq. Datab.	RND 1-Err	TXT 1-Err	KS 1-Err	MDS 1-Err	RND 5-Err	TXT 5-err	KS 5-Err	MDS 5-Err
Original (f)	38.5 ± 1.3	42.6 ± 3.0	38.9 ± 3.5	39.1 ± 9.2	16.5 ± 0.9	19.5 ± 1.8	17.0 ± 2.7	16.8 ± 6.4
Multinomial	43.0 ± 1.2	33.5 ± 3.0	36.5 ± 3.2	25.3 ± 6.5	18.8 ± 1.0	12.9 ± 1.8	14.6 ± 2.5	5.5 ± 2.4
Weighting	41.3 ± 1.4	35.4 ± 3.3	36.2 ± 3.0	28.3 ± 6.3	17.8 ± 1.0	14.2 ± 2.0	14.6 ± 2.2	6.8 ± 2.3
Unlearning Multinomial	40.1 ± 1.2	35.2 ± 3.2	32.7 ± 2.9	27.3 ± 6.5	17.2 ± 0.9	14.5 ± 1.6	12.3 ± 2.0	7.1 ± 2.6
Hierarchy Multinomial	40.5 ± 1.3	34.9 ± 2.6	34.9 ± 3.2	25.3 ± 6.3	17.6 ± 0.9	14.0 ± 1.5	13.8 ± 2.4	5.4 ± 2.1
Hierarchy Weighting	42.1 ± 1.4	36.5 ± 3.7	34.9 ± 2.7	30.5 ± 6.9	18.4 ± 1.0	15.0 ± 2.3	13.3 ± 1.8	8.1 ± 3.2

We first notice that the original CCV classifier alone already achieves state-of-the-art results (around 18% top-5 error rate), thus acting as a strong baseline. For the RND database (where no clear context exists), our different context-sensitive approaches make slightly more errors than the original classifier, which is to

³ “*ccv: A Modern Computer Vision Library (ConvNet, Deep Convolutional Networks)*”, <http://libccv.org/doc/doc-convnet/>.

⁴ “*JSKD: SGD for large-scale classification.*”, <http://lear.inrialpes.fr/src/jsgd/>.

be expected since we are enforcing a semantic context information which does not make sense in this case. However the difference is very small (between +0.7 % and +2.3% error rate for top-5).

As for the realistic label sequences, we observe that all our algorithms yield similar results, and they all outperform the original classifier. Furthermore, the improvement is often higher on the top-1 error (-13.8% errors on MDS top-1 error). Overall the Multinomial model performs the best, especially on the TXT and MDS databases. On the contrary, the best results on KS are achieved by the Unlearning Multinomial, which shows that adding an unlearning step to reduce the influence of oldest labels does improve the results in this case; in fact, by browsing the KS grid we may drift far away from the starting point, thus the oldest labels are not representative of the context anymore.

In [Tab. 2](#), we present the top-1 and top-5 error rates in the same setting but for the reinforcement scenarios. As we stated in [Sec. 3.2](#), because of the originally good performance of the CCV classifier, the reinforcement models yield results very similar to the fully-supervised ones (approximately +2% error rate compared to fully-supervised), and they outperform the original classifier on every realistic sequences dataset.

Table 2: Mean Top-1 and Top-5 Error Rates(%) (ILSVRC2012 dataset; CCV classifier; reinforcement setting)

Classifier \ Seq. Datab.	RND 1-Err	TXT 1-Err	KS 1-Err	MDS 1-Err	RND 5-Err	TXT 5-err	KS 5-Err	MDS 5-Err
Original (f)	38.5 ± 1.3	42.6 ± 3.0	38.9 ± 3.5	39.1 ± 9.2	16.5 ± 0.9	19.5 ± 1.8	17.0 ± 2.7	16.8 ± 6.4
Reinforced Multinomial	41.0 ± 1.2	35.3 ± 2.9	38.2 ± 3.4	28.7 ± 8.2	17.7 ± 1.0	14.5 ± 1.7	16.0 ± 2.6	8.6 ± 4.6
Reinforced Weighting	39.2 ± 1.2	37.7 ± 2.1	37.7 ± 3.4	32.8 ± 6.9	16.8 ± 0.9	15.9 ± 1.6	16.4 ± 2.6	10.8 ± 4.6

Finally, [Tab. 3](#) contains the results for the Unsupervised Multinomial. In terms of applications, the unsupervised scenario is very interesting, because we do not always have access to all information about the queries at testing time, thus the fully-supervised feedback setting is not possible. As for the reinforcement setting, the Unsupervised model yield accuracies close to the fully-supervised Multinomial model (difference of roughly 6%), and it still outperforms f on realistic sequences, apart from the KS sequences, which probably comes from the fact that the fully-supervised Multinomial Model does not perform so well on this database (contrary to the Unlearning Multinomial), thus its unsupervised counterpart does not yield good results either. For better visualization of these error rates (ILSVRC2012; CCV), [Appendix 1](#) contains the corresponding whisker plots.

Table 3: Mean Top-1 and Top-5 Error Rates(%) (ILSVRC2012 dataset; CCV classifier; unsupervised setting)

Classifier \ Seq. Datab.	RND 1-Err	TXT 1-Err	KS 1-Err	MDS 1-Err	RND 5-Err	TXT 5-err	KS 5-Err	MDS 5-Err
Original (f)	38.5 ± 1.3	42.6 ± 3.0	38.9 ± 3.5	39.1 ± 9.2	16.5 ± 0.9	19.5 ± 1.8	17.0 ± 2.7	16.8 ± 6.4
Unsupervised Multinomial	45.5 ± 1.3	40.0 ± 3.5	42.8 ± 3.7	32.3 ± 9.8	20.0 ± 1.0	17.3 ± 2.4	18.4 ± 3.2	10.6 ± 6.2

Lastly, in order to show these results do not depend on the initial classifier f , we present the error rates obtained on the ILSVRC2012 dataset using JSGD in [Tab. 4](#). Since the original JSGD classifier is not as good as the CCV one, the improvement is often greater (-30% errors on MDS top-5 error). On the other hand, because the original classifier is less accurate, the reinforcement and unsupervised models usually perform worse than with CCV (roughly +15% error rate from fully-supervised Multinomial to unsupervised here). Apart from this, the conclusions are generally the same as before: the Multinomial model yields the best results on the TXT and MDS databases (although it is outperformed by the hierarchical Multinomial on MDS, but only by 0.1%). As for the KS sequences, the Hierarchy Weighting perform better. However, among the models who do not use this hierarchy information, Weighting performs better than Multinomial on KS. This show that the simple Multinomial is not so well fit for the generative process of KS sequences.

Finally, [Appendix 2](#) contains the error rates for the ILSVRC2010 dataset using the CCV and JSGD classifiers; since they are very similar to these tables, we do not present them here.

Classifier \ Seq. Datab.	RND 1-Err	TXT 1-Err	KS 1-Err	MDS 1-Err	RND 5-Err	TXT 5-err	KS 5-Err	MDS 5-Err
Original (f)	72.4 \pm 1.4	70.0 \pm 2.2	72.7 \pm 2.6	71.7 \pm 8.0	52.3 \pm 1.4	50.1 \pm 2.6	52.6 \pm 3.0	51.3 \pm 9.5
Multinomial	78.1 \pm 1.1	60.3 \pm 4.2	71.1 \pm 2.6	52.8 \pm 11.5	57.2 \pm 1.5	38.0 \pm 4.3	48.4 \pm 3.2	23.1 \pm 10.8
Weighting	77.4 \pm 1.3	61.4 \pm 4.7	66.5 \pm 2.3	54.9 \pm 9.8	56.2 \pm 1.5	39.8 \pm 4.3	43.6 \pm 2.6	24.1 \pm 8.9
Reinforced Multinomial	74.8 \pm 1.1	63.6 \pm 3.9	73.2 \pm 2.7	60.6 \pm 12.0	54.1 \pm 1.4	43.7 \pm 3.9	52.2 \pm 3.3	36.9 \pm 13.2
Unsupervised Multinomial	86.0 \pm 1.1	74.6 \pm 5.5	85.3 \pm 2.6	69.9 \pm 12.9	65.0 \pm 1.5	54.8 \pm 6.4	64.4 \pm 3.8	47.4 \pm 16.4
Reinforced Weighting	73.9 \pm 1.3	64.2 \pm 3.6	72.4 \pm 2.8	64.7 \pm 11.9	53.4 \pm 1.4	45.2 \pm 3.8	52.2 \pm 3.2	44.3 \pm 14.3
Unlearning Multinomial	75.5 \pm 1.3	61.3 \pm 4.5	64.7 \pm 2.8	54.7 \pm 9.9	54.5 \pm 1.4	41.0 \pm 4.0	41.7 \pm 3.0	27.3 \pm 9.1
Hierarchy Multinomial	76.1 \pm 1.2	61.1 \pm 3.6	69.7 \pm 2.7	53.0 \pm 11.0	55.0 \pm 1.5	40.4 \pm 3.2	46.9 \pm 3.1	23.0 \pm 10.3
Hierarchy Weighting	78.3 \pm 1.2	63.0 \pm 5.1	63.9 \pm 2.6	55.9 \pm 9.0	57.2 \pm 1.4	41.5 \pm 4.8	38.7 \pm 2.6	23.8 \pm 8.2

Table 4: Mean Top-1 and Top-5 Error Results on the ILSVRC2012 dataset using a JSGD classifier (%)

Conclusion

In this work, we studied image classification in the situation where the images to be classified come in a semantically meaningful order. We showed that incorporating knowledge of the semantic context into the task of classifying images improves the accuracy results in this setup. Context modelling can be seen as assigning a weight to each label, which represents how likely a class is in the given context. We use contextual knowledge to “guide” the visual classifier when needed. We have created three generative processes for realistic sequences, each with their own characteristic. Finally, we have proposed two online learning algorithms for joint context modelling and classification, and extend them to reinforcement and unsupervised settings. In every setting we experimented, our approach outperforms the original classifier on the task of classifying images ordered in a realistic sequence. Furthermore, when using an initial state-of-the-art classifier, the reinforcement and unsupervised approaches yield very close results to the fully-supervised setting.

Possible future works include investigating more specific context modelling methods (e.g. prediction suffix trees) or exploring situations where the data has a known particular structure (for example query sequences generated by a Markov chain). Finally, it could be interesting to analyze these algorithms from a more theoretical point of view, in the framework of online learning, in order to better understand the impact and limits of this approach. More particularly, it is possible to investigate the trade-off between an accurate context model, and a context model which combines well with a classifier. In fact, in this task we do not only need the model to be a good approximation of the context, but also that the scores output by the model do not influence the original visual classifier too much when multiplying them (because generally the visual information is still more accurate than the contextual one, therefore the combination should not favor the contextual decision).

Acknowledgements

First, I would like to thank my advisor, Professor Christoph Lampert, for his great help and availability during this internship, as well as Hervé Jégou for recommending this internship to me. I also thank IST Austria for the excellent working (and living) environment provided, and Elisabeth Hacker for her administrative help.

I am very thankful to the whole Computer Vision team and friendly people at IST for their warm welcome, great moments of fun, and thrilling table soccer games.

Finally I would like to thank Mathias Fleury, Alix Trieu and Nathanaël Cherièrè for reviewing earlier versions of this report.

References

- [1] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.
- [2] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. 2006.
- [3] T. F. Cox and M.A.A. Cox. *Multidimensional Scaling, Second Edition*. Chapman and Hall/CRC, 2 edition, 2000.
- [4] O. Dekel, S. Shalev-Shwartz, and Y. Singer. Individual sequence prediction using memory-efficient context trees. *IEEE Transactions on Information Theory*, 55(11):5251–5262, 2009.
- [5] Y. Jia, J. T. Abbott, J. L. Austerweil, T. L. Griffiths, and T. Darrell. Visual concept learning: Combining machine vision and bayesian generalization on concept hierarchies. In *NIPS*, 2013.
- [6] Y. Jia and T. Darrell. Latent task adaptation with large-scale hierarchies. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.
- [7] R. E. Krichevsky and V. K. Trofimov. The performance of universal encoding. *IEEE Transactions on Information Theory*, 27(2):199–206, 1981.
- [8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*. 2012.
- [9] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Mach. Learn.*, 2(4):285–318, April 1988.
- [10] Y. Lomnitz and M. Feder. A universal probability assignment for prediction of individual sequences. In *ISIT*, 2013.
- [11] U.-V. Marti and H. Bunke. Hidden markov models. chapter Using a Statistical Language Model to Improve the Performance of an HMM-based Cursive Handwriting Recognition Systems, pages 65–90. 2002.
- [12] C. Mesterharm. A multi-class linear learning algorithm related to WINNOWER with proof. In *NIPS*, 2000.
- [13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [14] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classifiers*, 2000.
- [15] N. Quadrianto, L. Song, and A. J. Smola. Kernelized sorting. In *NIPS*. 2009.
- [16] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009.
- [17] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. 2007.
- [18] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? pages 1270–1278, 2000.
- [19] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–, july, october 1948.
- [20] K. Shimada, R. Muto, and T. Endo. A combined method based on SVM and online learning with HOG for hand shape recognition. *JACIII*, 16(6):687–695, 2012.

Appendices

Whisker Plots (CCV ILSVRC2012, Top-5)

This appendix contains whisker plots for the top-5 error rates on the ILSVRC2012 dataset using the pretrained CCV classifier, with the setting presented in [Sec. 5.1](#).

Here, red lines are the median values, limits of the boxes are the first and third quartiles, and the extreme points are the minimum and maximum values.

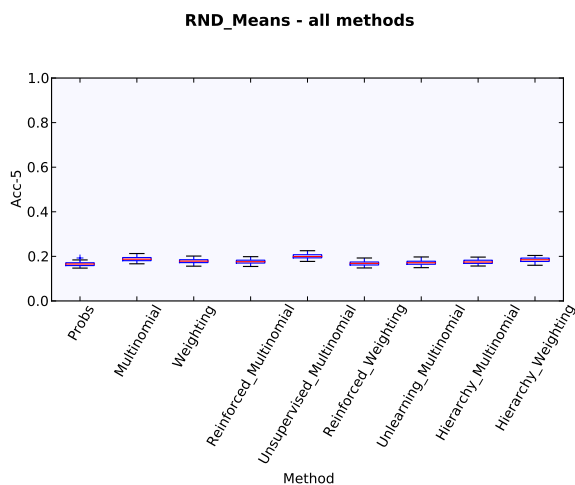


Fig. 15: Mean Top-5 Error Rates on RND sequences using CCV as initial classifier on ILSVRC2012

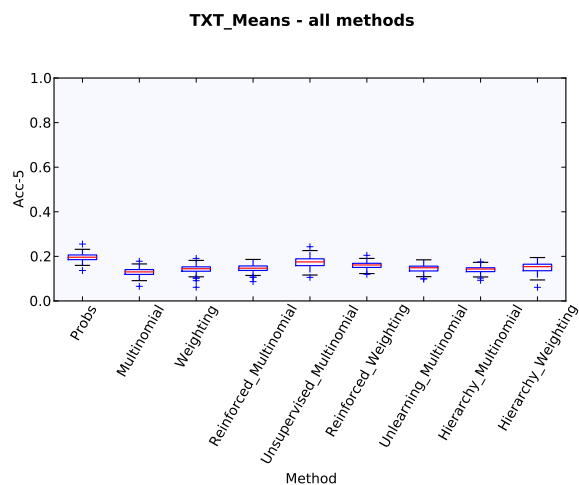


Fig. 16: Mean Top-5 Error Rates on TXT sequences using CCV as initial classifier on ILSVRC2012

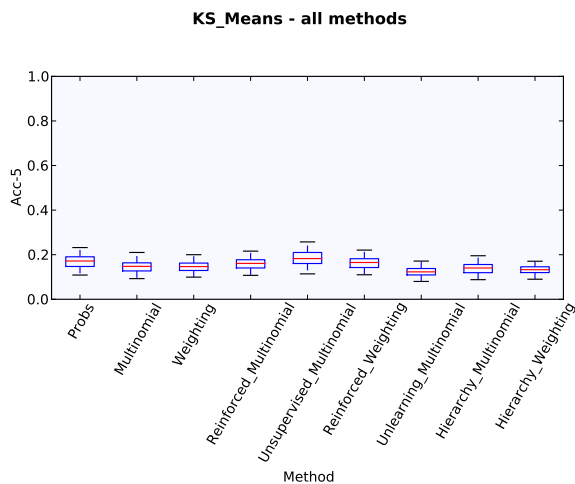


Fig. 17: Mean Top-5 Error Rates on KS sequences using CCV as initial classifier on ILSVRC2012

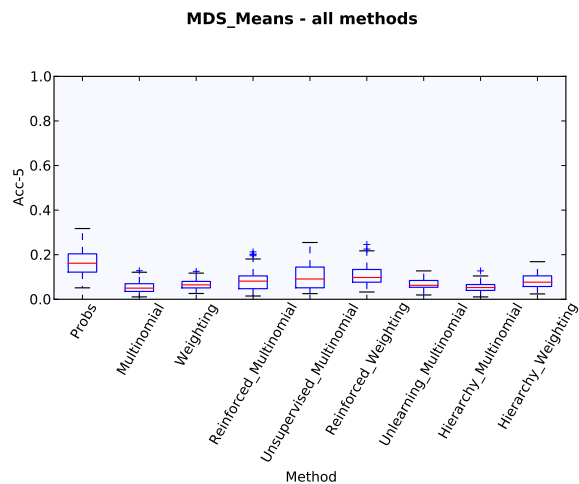


Fig. 18: Mean Top-5 Error Rates on MDS sequences using CCV as initial classifier on ILSVRC2012

Additional Error results (CCV and JSGD, ILSVRC2010)

Table 5: Mean Top-1 and Top-5 Error Rates(%) (ILSVRC2010 dataset; CCV classifier; all settings)

Seq. Datab.	RND 1-Err	TXT 1-Err	KS 1-Err	MDS 1-Err	RND 5-Err	TXT 5-err	KS 5-Err	MDS 5-Err
Classifier								
Original (f)	34.5 ± 1.2	34.7 ± 2.5	34.7 ± 2.5	34.0 ± 6.9	14.6 ± 0.9	14.3 ± 1.5	14.5 ± 1.9	13.8 ± 5.2
Multinomial	38.4 ± 1.2	26.3 ± 3.1	32.9 ± 2.2	23.5 ± 7.6	16.3 ± 0.9	9.5 ± 1.7	12.5 ± 1.7	6.2 ± 5.0
Weighting	36.7 ± 1.2	27.7 ± 3.4	33.0 ± 2.0	26.3 ± 7.4	15.5 ± 0.9	10.4 ± 1.8	12.8 ± 1.6	7.4 ± 4.6
Reinforced Multinomial	36.8 ± 1.2	27.5 ± 3.0	34.3 ± 2.3	26.1 ± 8.6	15.5 ± 0.9	10.4 ± 1.8	13.7 ± 1.9	8.4 ± 6.6
Unsupervised Multinomial	40.4 ± 1.3	30.6 ± 3.2	37.8 ± 2.6	29.4 ± 10.3	17.3 ± 1.0	11.6 ± 1.9	15.3 ± 2.2	10.1 ± 7.9
Reinforced Weighting	35.0 ± 1.1	30.7 ± 2.1	33.8 ± 2.4	30.1 ± 6.7	14.7 ± 0.9	11.7 ± 1.5	14.0 ± 1.9	10.2 ± 5.6
Unlearning Multinomial	35.9 ± 1.2	27.6 ± 3.0	29.9 ± 2.1	25.2 ± 7.5	15.1 ± 0.9	10.5 ± 1.5	10.8 ± 1.5	7.5 ± 5.0
Hierarchy Multinomial	36.4 ± 1.1	26.7 ± 2.6	31.4 ± 2.1	23.3 ± 7.4	15.4 ± 0.9	10.0 ± 1.4	11.8 ± 1.6	6.0 ± 4.8
Hierarchy Weighting	37.4 ± 1.2	28.8 ± 3.5	31.9 ± 1.9	27.7 ± 7.6	15.9 ± 0.9	11.1 ± 2.0	11.9 ± 1.4	8.2 ± 5.2

Table 6: Mean Top-1 and Top-5 Error Rates(%) (ILSVRC2010 dataset; JSGD classifier; all settings)

Seq. Datab.	RND 1-Err	TXT 1-Err	KS 1-Err	MDS 1-Err	RND 5-Err	TXT 5-err	KS 5-Err	MDS 5-Err
Classifier								
Original (f)	65.4 ± 1.2	61.1 ± 2.7	65.2 ± 2.6	63.3 ± 10.3	44.4 ± 1.2	39.3 ± 2.7	44.1 ± 2.6	42.6 ± 10.9
Multinomial	68.6 ± 1.1	48.6 ± 4.8	62.1 ± 2.7	45.0 ± 11.1	47.7 ± 1.2	29.2 ± 3.8	40.0 ± 2.9	18.0 ± 8.9
Weighting	68.0 ± 1.3	50.0 ± 4.8	59.8 ± 2.3	47.2 ± 10.1	47.0 ± 1.3	30.5 ± 3.8	37.3 ± 2.3	19.2 ± 7.9
Reinforced Multinomial	66.3 ± 1.1	50.8 ± 4.9	64.0 ± 2.7	50.4 ± 12.5	45.4 ± 1.3	31.9 ± 3.8	42.9 ± 2.8	27.1 ± 11.8
Unsupervised Multinomial	74.6 ± 1.6	58.5 ± 5.9	72.5 ± 3.5	56.5 ± 15.1	52.9 ± 1.4	37.7 ± 5.0	50.6 ± 3.5	32.4 ± 15.2
Reinforced Weighting	65.5 ± 1.1	52.5 ± 3.9	63.9 ± 2.6	54.9 ± 12.8	44.7 ± 1.3	33.4 ± 3.6	43.2 ± 2.6	33.1 ± 14.6
Unlearning Multinomial	66.4 ± 1.2	50.2 ± 4.6	57.4 ± 2.4	47.1 ± 10.0	45.6 ± 1.3	31.0 ± 3.5	34.7 ± 2.2	22.1 ± 7.7
Hierarchy Multinomial	67.0 ± 1.1	49.6 ± 4.2	60.9 ± 2.8	44.8 ± 11.0	46.2 ± 1.2	30.1 ± 3.1	38.8 ± 2.9	18.0 ± 8.6
Hierarchy Weighting	68.6 ± 1.3	51.2 ± 4.9	57.1 ± 2.3	47.6 ± 9.8	47.9 ± 1.4	31.5 ± 4.0	33.2 ± 2.3	18.0 ± 7.5