

Learning to Reduce Annotation Load

Thèse N° 9074

Présentée le 18 janvier 2019

à la Faculté informatique et communications

Laboratoire de vision par ordinateur

Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

KSENIA KONYUSHKOVA

Acceptée sur proposition du jury

Prof. S. Süsstrunk, présidente du jury

Prof. P. Fua, Prof. R. Sznitman, directeurs de thèse

Prof. B. Caputo, rapporteuse

Prof. C. H. Lampert, rapporteur

Dr M. Rajman, rapporteur

2019

Acknowledgments

I do work. I gather sun rays for the cold, dark winter days.

Frédéric (Lionni [110])

The road towards the PhD is long and winding. The journey would not have been possible without the contributions and support of many people. First of all I would like to thank my supervisor Pascal Fua for the opportunity to join his lab and for the freedom he gave me in research. I cannot help admiring his energetic and productive manner of work. I would also like to thank my co-supervisor Raphael Sznitman for his enthusiasm and support. I would remember fondly our conversations over hot chocolate (special thanks goes to café de Grancy for the environment for stimulating discussions).

Next, I would like to express my gratitude to the members of my thesis jury: Martin Rajman, Christoph Lampert and Barbara Caputo, who took their time to carefully read my thesis. Thank you for the useful comments and insightful questions and discussion. I would also like to thank Sabine Süssstrunk, the jury president and my first advisor at EPFL, who also became my role model.

I was fortunate to be hosted at Google Research by Vittorio Ferrari and Jasper Uijlings. I would like to thank them for sharing their experience and providing feedback on my work. I am also thankful to the other members of the *hot chocolate club* and to all the cool people at Google.

It has been a pleasure to work in CVLab with eccentric and awesome people. They are experts in making snowmen (even in shorts!), great travel and conference companions, amateurs in baking delicious cakes, poetry-lovers, crazy athletes, supporters of tea and gardeners. There are also people who do not like green peas, people who believe or do not believe in free will and many others. I am proud to belong to this fantastic group!

My deep gratitude goes to my family: my parents Lidia and Peter and my grandmother Nina, who were always here for me during these years. Not to forget my brother Andrew, who inspired my interest in science. Finally, a special thanks goes to Lucas who has been my most strict reviewer and at the same time the biggest supporter of my ideas.

Lausanne, November 29, 2018

Ksenia

Abstract

Modern machine learning methods and their applications in computer vision are known to crave for large amounts of training data to reach their full potential. Because training data is mostly obtained through humans who manually label samples, it induces a significant cost. Therefore, the problem of reducing the annotation load is of great importance for the success of machine learning methods.

We study the problem of reducing the annotation load from two viewpoints, by answering the questions “*What to annotate?*” and “*How to annotate?*”. The question “*What?*” addresses the selection of a small portion of the data that would be sufficient to train an accurate model. The question “*How?*” focuses on minimising the effort of labelling each datapoint.

The question “*What to annotate?*” becomes particularly compelling if we can select data to be annotated in an iterative and adaptive way, a setting known as active learning (AL). The key challenge in AL is to identify the datapoints that are the most informative for the model at a given stage. We propose several techniques to address this challenge. Firstly, we consider the problem of segmenting natural images and image volumes. We take advantage of image priors, such as smoothness of objects of interest, and use them in a novel form of geometric uncertainty. Using this, we design an AL technique to efficiently annotate data that is tailored to segmentation applications. Next, we notice that no single manually-designed strategy outperforms others in every application and that often the burden of designing new strategies outweighs the benefits of AL. To overcome this problem we suggest learning an AL strategy from data by formulating the AL problem as a regression task that predicts the reduction in the generalisation error achieved by labelling each datapoint. This enables us to learn AL strategies from simulated data and to transfer them to new datasets. Finally, we turn towards non-myopic data-driven AL strategies. To this end, we formulate the AL problem as a Markov decision process and find the best selection policy using reinforcement learning. We design the decision process such that the policy can be learnt for any ML model and transferred to diverse application domains.

Abstract

Effectively addressing the question “*How to annotate?*” is of no less importance as large cost savings can be achieved by labelling each datapoint more efficiently. This can be done with intelligent interfaces that interact with a human annotator. We make two contributions towards answering the question “*How?*”. Firstly, we propose an efficient technique to annotate 3D image volumes for image segmentation. Annotating data in 3D is cumbersome and an obvious way to facilitate it is to select a subset of the data lying on a 2D plane. To find the optimal plane (*i.e.* the one containing the most informative datapoints) we design a branch-and-bound algorithm that quickly eliminates hypotheses about the optimal projection. Secondly, we propose an intelligent data annotation method to train object detectors. Instead of always asking the human annotator to draw bounding boxes in images, we detect automatically in which cases we can rely on the current detector and verify its proposal.

Keywords machine learning, active learning, classification, interactive learning, meta-learning, computer vision, segmentation, object detection

Résumé

Les méthodes modernes d'apprentissage automatique (en anglais : machine learning, ou ML) et leurs applications en vision par ordinateur sont connues pour avoir besoin de grandes quantités de données d'entraînement pour atteindre leur plein potentiel. Comme ces données d'entraînement sont principalement obtenues par un étiquetage manuel des échantillons, cela induit un coût important. En conséquence, le problème de la réduction de la charge d'annotation revêt une importance particulière pour le succès pratique des méthodes d'apprentissage automatique.

Nous étudions le problème de la réduction de la charge d'annotation de deux points de vue, en tentant de répondre aux questions “*Qu'annoter ?*” et “*Comment annoter ?*”. La question “*Quoi ?*” traite de la sélection d'un sous-ensemble des données suffisant pour entraîner un modèle précis. La question “*Comment ?*” se concentre sur la réduction du temps requis par un humain pour annoter chaque donnée.

La question “*Qu'annoter ?*” devient particulièrement prometteuse si nous pouvons sélectionner les données à annoter de manière itérative et adaptative : ce cas est communément appelé apprentissage actif (en anglais : active learning, ou AL). Dans ce cas, le défi principal est d'identifier les exemples les plus informatifs pour apprendre un certain modèle. Nous proposons plusieurs techniques pour relever ce défi. D'abord, nous considérons le problème de la segmentation d'images et de volumes d'images. Nous tirons parti des caractéristiques géométriques des images, telles que la régularité des objets représentés, et les utilisons dans une nouvelle mesure d'incertitude. A partir de cela, nous concevons une technique d'AL particulièrement adaptée aux applications de segmentation. Ensuite, nous remarquons qu'aucune stratégie conçue manuellement n'est meilleure que toutes les autres dans chaque application et que, souvent, l'effort nécessaire à la conception d'une nouvelle stratégie d'étiquetage l'emporte sur les avantages d'AL. Pour surmonter ce problème, nous proposons d'apprendre une stratégie AL à partir de données, en utilisant un modèle de régression qui prédit la réduction de l'erreur de généralisation à partir de l'étiquetage d'un point de donnée. Cela nous permet d'apprendre des stratégies AL à partir de données simulées et de les transférer sur de nouveaux domaines. Pour finir, nous nous tournons vers des stratégies AL qui essaient de minimiser le coût d'étiquetage à

long terme. Pour cela, nous formulons le problème d'AL comme un processus de décision markovien et trouvons la meilleure stratégie de sélection en utilisant l'apprentissage par renforcement. Nous formulons le processus de décision markovien de manière à ce que la stratégie puisse être apprise de façon générale pour tous les modèles ML et transférée de façon performante vers divers domaines applicatifs.

Il est tout aussi important de répondre à la question “*Comment annoter ?*”, car de grandes économies peuvent être réalisées en étiquetant chaque donnée plus efficacement. L'approche que nous considérons consiste à développer des interfaces intelligentes qui interagissent avec un annotateur humain. En particulier, nous apportons deux contributions pour répondre à la question “*Comment ?*”. Premièrement, nous proposons une technique efficace pour annoter des volumes d'image 3D pour la segmentation. L'étiquetage de données tridimensionnelles est lourd et complexe, et un moyen évident de le faciliter consiste à sélectionner un sous-ensemble des données qui se trouvent sur plan bidimensionnel. Pour trouver le plan optimal (c'est-à-dire celui qui contient les pixels les plus informatifs), nous concevons un algorithme par séparation et évaluation qui élimine rapidement les hypothèses non optimales. Deuxièmement, nous proposons une méthode d'annotation intelligente pour entraîner un détecteur d'objet. Au lieu de demander systématiquement à l'annotateur humain de manuellement délimiter les objets sur les images, nous parvenons à reconnaître automatiquement dans quels cas il est plus efficace de vérifier une proposition du détecteur.

Mots-clés apprentissage automatique, apprentissage actif, classification, apprentissage interactif, méta-apprentissage, vision par ordinateur, segmentation, détection d'objet

Contents

Acknowledgments	iii
Abstract / Résumé	v
1 Introduction	1
1.1 Motivation	1
1.2 Contributions of this thesis	3
1.3 What to annotate?	4
1.3.1 Active learning problem formulation	5
1.3.2 Evolution of active learning methods	6
1.3.3 Practical constraints and problem formulations	12
1.3.4 Connections to other fields	14
1.3.5 Contributions towards answering “ <i>What to annotate?</i> ”	15
1.4 How to annotate?	16
1.4.1 How to annotate in computer vision applications	17
1.4.2 Trends in design of annotation pipelines	20
1.4.3 Contributions towards answering “ <i>How to annotate?</i> ”	23
2 Geometry-Based Active Learning for Image Segmentation	25
2.1 Introduction	25
2.2 Related work	27
2.3 Approach	28
2.4 Geometry-based active learning	29
2.4.1 Uncertainty measures	30
2.4.2 Feature uncertainty (FUn)	32
2.4.3 Geometric uncertainty (GUn)	32
2.4.4 Combining feature and geometric uncertainties	34
2.5 Batch-mode geometry query selection	35
2.5.1 Parametrizing the search space	36

Contents

2.5.2	Searching for the best bisecting plane	38
2.5.3	Illustration of search procedure in 2D	41
2.6	Experiments	42
2.6.1	Setup and parameters	42
2.6.2	Multi-class classification with AL	46
2.6.3	Segmentation of volumetric data with AL	48
2.6.4	Segmentation of natural images with AL	55
2.6.5	Active learning or human intuition	58
2.7	Conclusion	59
3	Learning Active Learning from Real and Synthetic Data	61
3.1	Introduction	61
3.2	Related work	62
3.3	Towards data-driven active learning	62
3.3.1	Success, failure, and motivation	62
3.4	Monte-Carlo LAL	65
3.4.1	Independent LAL	65
3.4.2	Iterative LAL	67
3.5	Experiments	68
3.5.1	Implementation details	68
3.5.2	Baselines and protocol	70
3.5.3	Synthetic data	71
3.5.4	Real data	75
3.5.5	Analysis of LAL strategies and time comparison	80
3.6	Conclusion	82
4	Towards Data-Driven General-Purpose Active Learning	83
4.1	Introduction	83
4.2	Related work	84
4.3	Method	85
4.3.1	Approach	85
4.3.2	Formulating AL as an MDP	86
4.3.3	Policy learning using RL	88
4.4	Experiments	90
4.4.1	Baselines and Parameters	90
4.4.2	Transferability	91
4.4.3	Flexibility	96
4.4.4	Analysis	98
4.5	Conclusion	100

5	Learning Intelligent Dialogs for Bounding Box Annotation	101
5.1	Introduction	101
5.2	Related work	103
5.3	Problem definition and motivation	104
5.3.1	Why use intelligent annotation dialogs?	104
5.3.2	Problem definition	104
5.4	Methods	106
5.4.1	IAD by predicting probability of acceptance	106
5.4.2	IAD by reinforcement learning	109
5.5	Experiments	110
5.5.1	Experimental setup	110
5.5.2	IAD with a fixed detector	111
5.5.3	IAD with an iteratively improving detector	113
5.6	Examples of dialogs	116
5.7	Conclusion	123
6	Conclusions	125
6.1	Summary	125
6.2	Future work	126
6.3	Final remarks	128
Bibliography		129
Curriculum Vitae		

1 Introduction

1.1 Motivation

Modern machine learning (ML) techniques require large amounts of training data to reach their full potential. Because of the power that these techniques demonstrate, most recent advances in computer vision (CV) rely on them. Thus, modern CV methods are known to crave for large amounts of training data. Yet, annotated data is hard and expensive to obtain, in particular in vision tasks where the annotations require laborious human intervention. Beyond CV, the same problem arises in specialized domains such as biology, medicine, and high energy physics, where only experts (whose time is scarce and precious) can provide reliable labels. Thus, the combination of a large demand for annotated data and the significant cost of labelling makes the problem of reducing the annotation load a very important task for many ML applications.

In this thesis, we develop methods that help to annotate data in an intelligent way. We are faced with a situation where domain experts are at our disposal, but their time is limited and expensive. Therefore we would like to utilize it as effectively as possible. For this, we design intelligent methods that reduce the annotation load. Recall that ML methods demonstrate a great potential for prediction tasks. The key to annotating data intelligently is to use insights that ML models can provide already during the annotation stage, before labels of most datapoints are known.

There are two ways to reduce annotation load: We can either annotate less data, or we can annotate each datapoint with a smaller cost. Thus, we address two questions in our work: “*What to annotate?*” and “*How to annotate?*”. When deciding “*What to annotate?*”, we seek the smallest possible set of training samples that would be sufficient to train a predictive model. Some datapoints are more informative for ML algorithms than others. If we can focus training on informative examples, the model can be trained with less supervision. Intuitively, this should be possible to achieve. For example, imagine a support vector machine classifier. Its predictions depend only on datapoints that are chosen as

support vectors and a set of support vectors is typically much smaller than a training set. If we could identify (even imperfectly) the datapoints that could become support vectors prior to annotating data, the annotation effort could be reduced significantly. The challenge is to identify such datapoints before obtaining all the labels.

We can address the question “*How to annotate?*” with the help of intelligent annotation interface. We notice that various annotation modalities have different costs. For example, a binary answer to the question whether a given object is detected correctly is cheaper than manually drawing a bounding box; annotating 2D images is typically cheaper than annotating 3D image stacks. In many cases, the cheaper annotation modality has a sufficient amount of information for training an ML method and a more detailed modality is unnecessary, for example, a positive binary confirmation of a bounding box proposal contains as much information as a box drawn by hand. If we could identify in advance when a cheaper modality would still result in sufficient supervision for a given task, we would save annotation costs. Once again, the challenge is to understand this prior to annotating data.

To build data-annotation pipelines we experiment with two types of techniques: *manually-designed* and *data-driven*. First, we consider manual design. In this case, the methods we develop rely on our intuitions about the most efficient annotation strategy or the most convenient collaboration with the user. The resulting methods are usually intuitive, easy to implement and can leverage prior knowledge about the problem. However, their performance may vary significantly from one scenario to another. Besides, relying on the knowledge of an algorithm designer for every particular problem is not scalable and may be suboptimal.

Data-driven techniques for data annotation started gaining popularity with the rise of meta-learning methods in ML. In particular, this means that the selection strategies themselves are *learnt from data*. As a result, they are usually less interpretable and require prior training. However, once they are constructed, they can be applied to many problems and prior knowledge of a human algorithm designer is not needed any more. Besides, these techniques are very flexible and they can combine known selection strategies or design completely new strategies.

Table 1.1 – Organisation of this thesis in terms of “*What to annotate?*” and “*How to annotate?*” research questions and *hand-designed* and data-driven techniques.

Question / Technique	What to annotate?	How to annotate?
Manually-designed	Chapter 2	Chapter 2
Data-driven	Chapter 3, Chapter 4	Chapter 5

In this thesis we study how both types of techniques—manually-designed and data-driven—can be used to answer the two questions of data annotation: “*What?*” and “*How?*”. Table 1.1 shows the organisation of this work in terms of annotation questions and design techniques. While the answer to the question “*What?*” reduces the annotation load by minimizing the amount of data to be labelled, the answer to the question “*How?*” helps to reduce the time to label datapoints. In practice, each of these techniques has a cost-saving potential. Additionally, combining them together can help to reduce the annotation cost even further. While manually-designed techniques are beneficial for specific problems, the data-driven techniques are promising in a wide range of settings.

Research statement By adaptively selecting *what* data to annotate and by intelligently deciding *how* to annotate it, it is possible to reduce the annotation load in ML applications. ML techniques can help to design annotation methods either by supporting *human intuitions* or learning a strategy *from data*.

1.2 Contributions of this thesis

This thesis addresses the problem of reducing data annotation cost in various problems and settings. The applications we consider range from segmentation of 3D image volumes to detecting Higgs boson.

We start with manually-designed methods to address a particular problem. Chapter 2 discusses the question “*What to annotate?*” in the context of binary and multi-class image segmentation for 2D and 3D images. We use image smoothness priors to detect the most uncertain regions in images. For 3D image volumes we additionally propose a method that answers the question “*How to annotate?*” by reducing 3D annotation to 2D annotation. For this we introduce a branch-and-bound algorithm to find a 2D plane containing the most uncertain voxels. We conduct experiments in multiple challenging datasets where our methods demonstrate good performance at a lower cost compared to conventional methods.

Starting from Chapter 3, our attention shifts towards data-driven approaches. Instead of manually designing algorithms we learn them from previous interactions with data. We apply this technique to answer both “*What to annotate?*” and “*How to annotate?*”.

In Chapters 3 and 4 we propose data-driven meta-AL approaches for binary classification. First, in Chapter 3, we treat AL as a regression problem where we predict the reduction in the test error as a function of a current classification state and available datapoints. The meta-AL strategy is learnt on synthetic data and then applied to a new domain. Next, in Chapter 4, we model AL as an MDP and find a selection strategy with a reinforcement learning algorithm. For this, we formalise the objective of AL so as to achieve a pre-defined

quality with the smallest amount of annotations. The proposed data-driven approaches outperform standard strategies and bring us closer to a general-purpose learnt AL policy.

Lastly, in Chapter 5, a data-driven approach is applied to the question “*How to annotate?*”. We design an annotation method for bounding boxes in object detection. We combine several annotation modalities either with a learnt probabilistic model or with a reinforcement learning algorithm. Our intuition is that the annotation strategy should depend on properties of the image, class, and detector, and should be learnt from data from previous annotation experiences rather than modelled explicitly. The learnt policy is shown to perform consistently better than fixed combinations of annotation modalities.

In the remainder of this introductory chapter we formalise what we mean by the questions “*What?*” and “*How?*”. We attempt to cover these topics broadly as there is no survey reporting recent progress in a unified way. We start in Section 1.3 by introducing the active learning (AL) problem. We present a few selection strategies, focusing on the recent shift towards data-driven techniques. In Section 1.4 we review the progress in answering the question “*How to annotate?*” and in particular we investigate how it is applied to various tasks in CV. Detailed summaries of contributions of this thesis towards answering each question can be found at the end of the corresponding section.

1.3 What to annotate?

When we address the question “*What to annotate?*” our aim is to reduce the cost by reducing the amount of annotated data, but to keep the prediction quality of the trained model high. In this thesis we consider an interactive annotation setting where an oracle, who can provide a correct label for any given datapoint, is at our disposal. In such a scenario, active learning (AL) is the technique of choice.

AL seeks to find, iteratively and adaptively, a small set of training samples to be annotated for effective model training [164]. In practice, this means that instead of asking an oracle to annotate all the data, we carefully select which datapoints should be labelled next based on what we know so far. The intelligent selection of data to be annotated can help to reach a good model performance using fewer labels.

In this section we introduce AL problem both intuitively and formally. Then, we review the development of AL methods in terms of query selection algorithms starting from manually-designed strategies and moving towards the strategies that are learnt from data. In the literature review we mostly concentrate on empirical results in AL. Next, we enumerate various problem formulations that take into account realistic problem constraints. Then, we briefly outline related problems that put ideas from AL in a different perspective. Finally, we enumerate the contributions of this thesis towards answering the question “*What to annotate?*”.

1.3.1 Active learning problem formulation

We have a dataset $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$ and we would like to train a classifier f on a subset \mathcal{D} such that it will be efficient in predicting labels of unseen datapoints from the same distribution. A datapoint x_i is represented by a D -dimensional feature vector and $y_i \in \mathcal{Y}$ is its label, where there are Y possible labels. For example, we often study binary classification: $\mathcal{Y} = \{0, 1\}$. We consider the pool-based setting where all datapoints x_i are observed prior to the annotation procedure. We choose a classifier f that is iteratively trained on some $\mathcal{L}_t \subset \mathcal{D}$ to map features to labels: $f_t(x_i) = \hat{y}_i$, for example, by predicting the probability $p_t(y_i = y \mid x_i)$. Given a classifier and a pool of unlabelled data, the goal of AL is to select which datapoints should be annotated next in order to learn a classification model as quickly as possible. Schematically, the standard AL procedure is illustrated in Figure 1.1. Formally, the AL *episode* unfolds as follows.

The algorithm starts with a small labelled training dataset $\mathcal{L}_0 \subset \mathcal{D}$ and a large pool of unlabelled data $\mathcal{U}_0 = \mathcal{D} \setminus \mathcal{L}_0$. Then, the following steps are performed at iteration t :

1. A classifier f_t is trained using \mathcal{L}_t .
2. A query selection procedure picks an instance $x_t^* \in \mathcal{U}_t$ to be annotated at the next iteration.
3. x_t^* is given a label y_t^* by an oracle. The labelled and unlabelled sets are updated.
4. t is incremented.

The procedure terminates when the desired classification quality is achieved or the number of *iterations* reaches a predefined limit. In practice, achieving a pre-defined quality is determined by a user of the system, and in the experiments we use a validation dataset for this purpose.

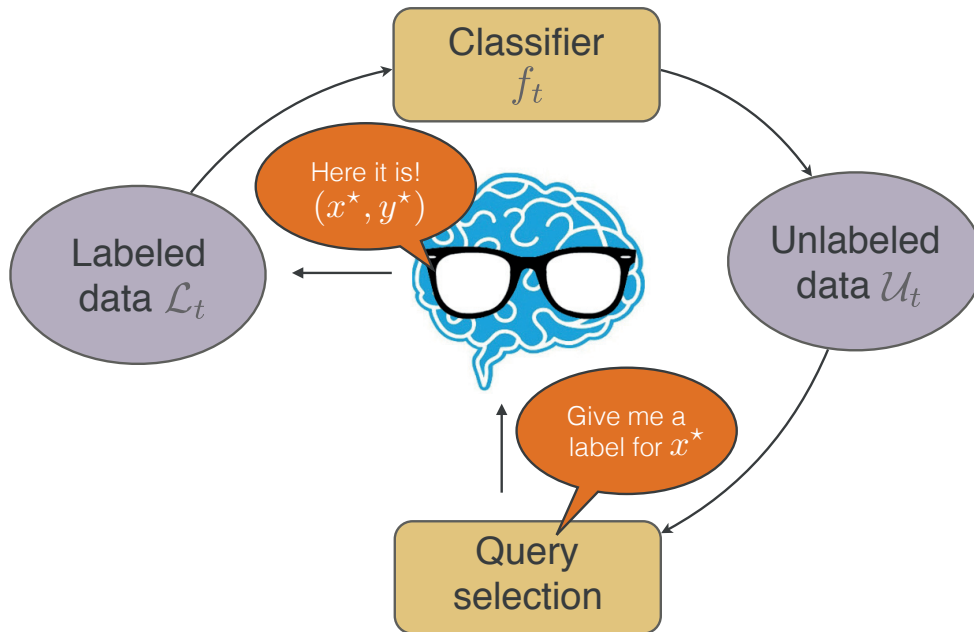


Figure 1.1 – Active learning (AL) procedure. AL aims to ease the data collection process by automatically deciding which instance (x_t^*) an oracle should label to train a classifier f_t as quickly and effectively as possible with the minimal amount of manual intervention.

1.3.2 Evolution of active learning methods

The extensive development of AL in the last decades has resulted in numerous strategies, varying in query selection policies, problem formulations, and practical restrictions. There are multiple ways to look at the kaleidoscope of existing methods. Many selection policies and practical considerations are covered by Settles [164]. Kovashka et al. [94] review AL strategies as part of crowdsourcing pipelines. In this thesis we take another perspective and review the development of methods from manually-designed heuristics to data-driven approaches.

Manually-designed methods

The driving force of manually-designed AL methods is the knowledge of an expert in ML. The strategies are motivated either by intuitions of the researcher (“the most uncertain datapoint should help the classifier the most” leads to uncertainty sampling strategy) or by approximations of theoretical objectives (“to minimize the loss we can decompose it into bias and variance and attempt to minimize the variance of the model” leads to variance minimization methods). These methods differ in their computational costs, theoretical guarantees, applicability with various classification schemas, etc., but they are unified by the fact that a human designer decides explicitly how the datapoints are selected. Here we describe a few representative methods from this family. We focus on

the methods that demonstrate empirical advantages in practice and that are important for the presentation of the next chapters.

Uncertainty sampling Among many selection strategies, uncertainty sampling (US) is both simple and computationally efficient. This makes it one of the most popular strategies in real applications, ranging from text classification [179] to predicting the 3D layout of rooms [122]. In short, it suggests labelling samples that are the most uncertain for the classifier, *i.e.* closest to the classifier’s decision boundary. US and its variants are reported to work remarkably well in numerous scenarios and settings despite their simplicity [115, 21, 64, 122, 172].

Three ways to define the most uncertain datapoint are presented in the book of Settles [164]: 1) maximum entropy of posterior probability distribution over classes, 2) minimal probability of selected class, and 3) minimal gap between the two most probable classes.

1. The most common way to estimate uncertainty is to compute the Shannon entropy H of the probability distribution over classes

$$H[p(y_i = \hat{y} | x_i)] = - \sum_{\hat{y} \in \mathcal{Y}} p(y_i = \hat{y} | x_i) \log p(y_i = \hat{y} | x_i), \quad (1.1)$$

and then to select a sample x^* that maximizes it:

$$x^* = \arg \max_{x_i \in \mathcal{U}} H[p(y_i = y | x_i)]. \quad (1.2)$$

We will refer to this strategy as *total entropy*.

2. Another way of selecting the most uncertain sample involves a datapoint with the smallest posterior probability for its most likely class b_1 , that is,

$$x^* = \arg \min_{x_i \in \mathcal{U}} p(y_i = b_1 | x_i) = \arg \min_{x_i \in \mathcal{U}} \max_{b \in \mathcal{Y}} p(y_i = b | x_i). \quad (1.3)$$

We will refer to this strategy as *minmax* because of its structure.

3. Uncertainty can also be measured by the difference in probability between the first and second most highly ranked classes b_1 and b_2 . The most uncertain sample is then taken to be

$$x^* = \arg \min_{x_i \in \mathcal{U}} \{p(y_i = b_1 | x_i) - p(y_i = b_2 | x_i)\}. \quad (1.4)$$

We will refer to this strategy as *min margin* because it minimizes the margin between the prediction and the second candidate for it.

In the case of binary classification, selection by *total entropy*, *minmax*, and *min margin* are strictly equivalent because the corresponding expressions (1.2), (1.3) and (1.4) are monotonic functions of each other and they select a datapoint with the probability to belong to any of the classes closest to 0.5. In the multi-class scenario, however, they are not equivalent and using one or the other can result in different behaviours [80, 114, 93]. According to Settles [164], *total entropy* is best suited for minimizing the expected logarithmic loss, while *minmax* and *min margin* are better for minimizing the expected 0/1-loss. There are many works relying on one of the above criteria or on their combination. This includes selection uncertainty [70], posterior distribution entropy [80, 202], the selected entropy or minimum margin criteria combined with exploration criteria [114], and all three strategies together [93].

Density-weighted methods The aim of the *density-weighted* query strategies is to account for the whole unlabelled set \mathcal{U} while selecting a datapoint. For example, *representative sampling* [105, 192] helps to avoid querying outliers and to select samples that are the most representative of the underlying distribution, and thus, once labelled, could influence the classifier the most. This approach is usually combined with other query selection strategies, for example, with uncertainty sampling. One variant could be:

$$x^* = \arg \max_{x \in \mathcal{U}} \left[\frac{1}{|\mathcal{U}|} \sum_{x_i \in \mathcal{U}} s(x, x_i) \right]^\alpha \cdot \gamma(x), \quad (1.5)$$

where $\gamma(x)$ is the informativeness measure according to some criteria (for example, *total entropy*), s measures how similar a datapoint is to another and a constant α controls the relative importance of the terms. Despite being very simple, density-based methods empirically demonstrate an advantage over other strategies in some applications [165]. Another way to incorporate the data distribution into the selection strategy is by enforcing the selected datapoints to be *diverse*, *i.e.* to be sufficiently different from already annotated datapoints in \mathcal{L} . This strategy is used in numerous applications, especially when a batch of samples needs to be selected [48, 107, 202].

Query-by-committee (QBC) *Query-By-Committee* (QBC) combines predictions of different classifiers and queries datapoints on which they disagree the most [55, 17]. This approach is inspired by the Probably Approximately Correct (PAC) learning framework. The intuition behind this strategy is that the points from controversial regions (according to the classifier) are the most informative as they help to eliminate many wrong classification hypotheses at once. When constructing a QBC algorithm one needs to select a committee of classifiers which vote on which class y a datapoint x belongs to. The selection strategy is then to chose a datapoint maximizing the vote entropy. This query selection method is often employed in practice as it does not require class probability estimates [17].

Expected model output change (EMOC) Another intuitive selection strategy is *Expected Model Output Change* (EMOC) [82, 52]. The idea is to add a point x^* that has the greatest impact on the parameters of a model after retraining it with a new labelled dataset $\mathcal{L}_{t+1} = \mathcal{L}_t \cup x^*$. As the real label is unknown, the impact can be measured as the expectation over possible labels \mathcal{Y} .

Data-driven methods

Empirical studies [165, 12, 47] show that there is no single manually-designed AL strategy that consistently outperforms all others in all applications. Manual design has limited applicability when strategies are optimized for a particular application and/or problem setting. While they can achieve remarkable performance in specific applications, it is often challenging to predict in advance which strategy is the most desirable in a particular situation. Besides, they are limited to the ML researcher’s intuitions, and do not systematically explore the entire solution space.

To overcome these limitations, recent approaches tend to design a strategy in a data-driven fashion. Instead of hand-crafting a method to a particular problem at hand, meta-learning generates a method directly from data. In its simplest form, meta-AL learns to combine various manually-designed methods to account for properties of the dataset and to adapt to a changing classifier. A more flexible meta-AL approach goes beyond combining existing methods and it can propose a completely new selection strategy based on the properties of a dataset and a classifier.

Learning to combine AL strategies If a single manually designed method does not consistently outperform all the others, we might determine which of the strategies has the biggest potential for the problem and choose to apply it. Thus, early meta-AL strategies are concerned with how to dynamically combine several query strategies together [134, 12, 62, 35, 47]. In this case, a strategy is learnt on the fly while the data from the domain of interest is labelled by an oracle. In practice it means that all strategies from an ensemble of candidate strategies are applied to the data collection and their progress is evaluated during an AL run. Then, the strategy which selects the most useful datapoints should have more influence on the future selection. The progress can not be evaluated directly as there is not enough labelled data to measure the performance. Thus, introducing a criterion for a robust performance evaluation with little data becomes the main challenge for these approaches.

The fixed combination of several strategies is already present at the core of density-weighted methods. These strategies can be regarded as an exploration/exploitation type of approach. On the one hand, the uncertainty component “exploits” when it tries to refine the decision boundary. On the other hand, representativeness and diversity components “explore” by sampling in under-represented but dense regions of the feature space. A

natural step towards meta-AL is to dynamically determine the amount of exploration and exploitation needed at the current learning stage. Osugi et al. [134] suggest an approach where the amount of exploration is adjusted automatically based on its success in the previous iterations. The intuition is that if there is a significant change in the hypothesis caused by exploration, it is successful and its probability should be increased. On the contrary, if exploring does not modify the current hypothesis, its probability should be decreased and the strategy should concentrate on refining the boundary.

A more complex way to balance between exploration and exploitation is proposed by Ebert et al. [47]. They model the annotation procedure with a Markov decision process (MDP). In this case, not only can the amount of exploration be adjusted over time, but also the exploitation strategy can be chosen among competing uncertainty criteria (such as *total entropy* and *min margin*, for example). The states of MDP are various exploitation strategies with a fixed amount of exploration. An action is taken by switching between strategies or adjusting the amount of exploration. This approach can be extended by adding a strategy to find high-quality oracles in the multi-annotator scenario [114]. This is done by extending the state and action spaces with strategies that select an oracle according to some criteria.

It is possible to form a combination of strategies that is more flexible by selecting them from a pool and assigning some weights. This probabilistic blending does not only address the exploration/exploitation dilemma, but it can also combine arbitrary selection strategies. It can be done with the help of a multi-armed bandit algorithm [12, 62], where various AL heuristics are treated as arms of a bandit. During an AL episode, the strategies from a pool of candidates are assigned weights based on their past performance. The success of an exploration strategy is measured by the amount of change in the ML model [134, 47], but this does not perfectly align to the ultimate success of an AL strategy. The AL progress can be measured by the reduction in the test error, however, it is impossible to estimate it during an AL episode because only very limited annotated data is available. Baram et al. [12] propose to use the maximum entropy criterion to estimate the relative classification performance without labels. Hsu and Lin [62] continue this line of work. They introduce a new unbiased estimator of the test error: importance-weighted accuracy. Chu and Lin [35] go further and transfer the bandit-learned combination of AL heuristics between different problems. The weights from a previously learnt combination are used to initialize the combination weights when AL starts on a new problem.

There are two main limitations of approaches that combine AL strategies. First of all, as learning happens while performing AL, the success of a blend depends on the ability to estimate the classification performance from scarce annotated data. The second limitation is that this approach can only combine existing, human-designed strategies in an adaptive manner, but it cannot propose a truly new strategy.

Learning AL from data The most recent meta-AL works try to learn a free-form strategy from data that goes beyond combining existing techniques [9, 37, 149]. To learn an AL strategy from data, they formulate an AL process as an MDP where states are characterised by a classifier and a dataset and actions are the unlabelled samples. Then, given an annotated dataset, we can simulate AL episodes by interacting with a simulated oracle. Availability of labelled data makes tracking the progress easy and precise. Then, one can learn what kind of actions are the most beneficial for training a classifier in each state of the environment. Finally, an annotation policy maps the properties of the current learning state into a scoring function for each datapoint.

Liu et al. [113] learn a meta-AL strategy by imitation learning. This requires an algorithmic expert which can demonstrate what action is the best in a given situation. For this, they simulate labelling every datapoint from the unlabelled set and check by how much the accuracy is changed. They use the action (datapoint) that makes the biggest greedy progress in AL as supervision in imitation learning. A disadvantage of imitation learning is the way how an algorithmic expert generates ideal behaviour. As it looks only one step ahead, its supervision is greedy and a suboptimal policy might be learnt.

To learn a non-myopic policy, reinforcement learning can be employed. Non-myopic behaviour is achieved by setting the objective function to maximise a long-term AL reward. The method of Bachman et al. [9] is an extension of one- (or few-) shot learning [184] where a classification model is learnt from few samples after observing many related tasks. It learns jointly a data representation, a classifier, and an AL strategy. As a reward they consider the cumulative performance on a validation set at each iteration. The behavioural policy is learnt with a reinforcement learning method: a policy gradient method.

Fang et al. [50] study a stream-based AL setting where datapoints come one after another and a decision on whether to annotate each one or not should be made immediately before observing subsequent data. Their MDP has two possible actions: annotate or skip a datapoint. Then, a Q-learning based reinforcement learning algorithm is used to learn a policy that is represented by a neural network. The data representation is learnt jointly with the selection strategy and is used for state representation.

A limitation of existing meta-AL approaches is that their success on a new problem largely depends on the availability of similar annotated datasets. A typical set-up for these approaches involves many related tasks or classes. For example, the method of Bachman et al. [9] is applied to hand-written characters from different alphabets and recommending items to different users. Then, it is assumed that data from the related tasks (other alphabets or users) is already annotated and the question is how to annotate classes from a new task. The transfer of learnt AL strategies is only performed in the context of standard transfer learning where the tasks are very related, such as transfer between languages [50, 113] or cross-domain sentiment analysis [113].

1.3.3 Practical constraints and problem formulations

In practice, the annotation problems encountered in real-world applications are very far from vanilla AL settings due to a number of constraints. For example, there might be no perfect oracle, but multiple noisy oracles. Also, the cost of annotation of different datapoints can differ depending on properties of the data. When a direct application of existing policies is impossible, a specially tailored AL procedure is designed to take the constraints into account. In this section we review several real-life constraints that are common for AL pipelines.

Different forms of annotation Depending on the form of feedback from an oracle, the AL query selection strategy needs to be significantly modified. For example, if an oracle is given an opportunity to provide feedback on the attributes (for example, feedback on a landscape image used to train a scene classifier could be “this scene is too open to be a forest”), this feedback can be propagated to other images that share this attribute [21]. Then, the AL selection strategy takes into account the results of this label propagation. If annotators can provide feedback at different levels (for example, feedback on one level is scene classification and another is object classification), an AL method should additionally make a decision on what level to make a query [106]. A very different form of feedback is proposed by Huijser and van Gemert [68]. They suggest directly labelling the decision boundary. A generative model proposes instances that are a fusion of two classes along a 1D line (for example, images that gradually transform a shoe into a bag). Then, they ask the annotator to indicate where the transition between the classes is. This annotation translates to a point on a decision boundary back in the feature space. We discuss various forms of feedback in more detail when we discuss the question “*How to annotate?*” in Section 1.4.

Choosing a labeller A realistic annotation environment in large-scale problems involves a pool of annotators who differ in their level of expertise and motivation. A number of AL approaches consider a problem of selecting jointly a datapoint and an annotator for it [151, 64, 115, 114]. A special form of this problem is to choose one of two oracles, one of which provides “gold standard” labels for a high price and another one provides noisy labels for a small price [204]. Similar ideas can be applied in the context of multi-task learning [132] where an AL algorithm can decide whether it should learn from other tasks or query an oracle.

Abstaining oracles The assumption that every datapoint can be labelled is often unrealistic in practice. Some instances can be too difficult or ambiguous to label, or even irrelevant for a given classification task [82]. The selection of the best oracle for a particular datapoint should take into account the probability that it can actually provide a label [200]. Besides, the oracle might abstain from labelling difficult datapoints until he develops a good intuition about the class separation [67].

Cost-sensitive AL In practice, not all datapoints have equal annotation cost. We can easily imagine that an instance that is uncertain for a classifier is ambiguous for an oracle as well. Annotating an ambiguous datapoint might take more time than annotating a simple datapoint. This motivates cost-sensitive AL, where the cost of each label might be different [167, 183]. A special case of cost-sensitive AL is *auditing*, where only negative labels are costly [156]. This constraint comes from the domain of fraud detection, where an investigation of honest transaction is undesirable.

Bootstrapping AL If annotated data from related domains is available prior to AL, transfer or few-shot learning can be used to effectively initialise the AL procedure. In transfer learning [135] we have a source domain with a sufficient amount of annotated training data and a target domain with little or no annotated data. The source and target domains are considered to be related, but not identical, such that direct application of trained models does not lead to good results. The goal of transfer learning is to adapt a model or the dataset itself to account for the differences between domains. A logical extension is to use transfer learning as a way to bootstrap AL [191, 54] or learn transfer learning and AL jointly [33]. In few-shot (or one-shot) learning it is assumed that many annotated datasets are available for a related problem (for example, many known categories for object categorization) and a classifier for a new problem (for example, recognition of new object classes) is learnt from a small set of examples [51, 184, 157, 45]. A natural extension of few-shot AL is to allow for adaptive data selection as AL does. Few-shot learning inspired several methods of meta-AL [9, 37, 149] and other annotation tasks [198]. In this case, meta-AL is bootstrapped by data representation learnt on many related datasets.

Meta-learning in annotation/prediction tasks There are several sequential annotation/prediction problems that are naturally formulated as an MDP and can be approached by methods similar to meta-AL. Sometimes the authors refer to these problems as AL, but in this thesis we make a distinction between the settings. For instance, in one of the formulations of a sequential prediction/annotation task an agent needs to decide whether to predict or request a label [198]. Then the reward of an agent is positive for making a correct prediction, slightly negative for requesting a label and very negative for making a wrong prediction. This scenario is similar to stream-based AL, but the task of annotating data for training is not detached from the task of prediction on the unseen test dataset.

Contardo et al. [37] and Ravi and Larochelle [149] study another annotation/prediction task and introduce the problem of *static* data selection. This setting lies at the intersection between few-shot and active learning. The algorithm selects a single set of samples to be annotated after observing many related tasks. In contrast to few-shot learning, the algorithm is allowed to decide which datapoints it receives, but in contrast to AL it cannot adapt its selection across iterations. Contardo et al. [37] combine the classification

quality and the cost of labelling datapoints in the objective function. Ravi and Larochelle [149] model explicitly batch-mode selection. Additionally, their strategy can deal with distractors: datapoints that do not belong to any of the classes of interest. Querying a distractor datapoint is costly but does not bring any additional information to the classifier.

Limitations of AL Despite extensive research in AL, it has not been widely adapted in practice [8]. This can be explained by several difficulties of applying AL to real tasks. First of all, the iterative annotation pipeline requires model re-training after every new sample or batch of samples. It is usually quite expensive for models that do not support efficient incremental retraining [163]. Next, sampling bias is a known artefact in AL and it implies the need for special treatment [163, 11, 42]. Besides, human annotators are usually considered to be perfect oracles and the cost of every label is uniform, which is clearly not the case in practice. In Section 1.4 we discuss how to deal with annotator modelling. Finally, it is hard to foresee the performance of AL algorithm on a new application [164, 12, 47] and in some cases the active data collection result in worse performance than passive sampling [31, 11].

1.3.4 Connections to other fields

Suppose now that all the data is already annotated. Can the ideas from AL still be useful in this situation? It turns out that yes, similar intuitions can be used, for example, for scheduling training procedures, teaching humans to classify images or saving computational time. Two general (and sometimes overlapping) approaches consist of determining the order of training samples, and selecting or re-weighting a subset of data for training.

Choosing the order In *curriculum* and *self-paced* learning, the task is to select the order of samples in such a way that the optimisation procedure used for learning a model converges faster and possibly avoids getting stuck in local minima. The intuition behind these methods is related to human learning, which is more successful when it starts from easy concepts and continues toward more elaborate ones. While in curriculum learning [18] the order is mainly determined by the teacher from prior knowledge, in self-paced learning [97] the student has a direct impact on determining which samples are important. Jiang et al. [76] unify and extend curriculum and self-paced learning. Curriculum and self-paced learning are also applied to other scenarios, such as determining the order of tasks in multi-task learning [144], determining the order of samples used in training deep neural networks [194, 77], or training an RL agent of increasing complexity [38].

Selecting or re-weighting data *Machine teaching* studies the following problem: A teacher knows precisely the underlying model and tries to transfer it to a student with the minimal amount of training examples. Research in machine teaching ranges from theoretical results on teaching dimension [56]—the minimum size of the training set that

allows to distinguish classes of the problem—to practical approaches that teach humans to distinguish images representing certain fine-grained classes [78, 123]. *Hard-negative mining* is a standard way to help training in applications with an overwhelming amount of negative datapoints [169, 27]. For example, in object detection all bounding boxes that do not overlap sufficiently with an object of interest are the negative samples. Instead of sampling negative datapoints uniformly, hard-negative mining concentrates on the datapoints that are the most difficult for a classifier. Apart from the special case of dominating negative datapoints, AL ideas can be used for selecting training data in more general settings. For example, it is shown that neural networks can be trained more accurately by using the most uncertain datapoints in stochastic gradient descent [32]. The *prioritised replay* technique [159] in reinforcement learning also serves to amplify some of the training samples – those on which the temporal-difference errors are the biggest. Data selection can also be done with a purpose of reducing the computational cost. For example, consider a CRF model with a high cost to initialise the potentials [152]. The computational cost can be reduced by deciding which of the potentials are sufficient for an efficient inference.

1.3.5 Contributions towards answering “*What to annotate?*”

Geometry-aware active learning for image segmentation [89, 91] Training an efficient image segmentation algorithm requires significant amount of pixel-wise annotations, which are known to be very time-consuming to produce. AL techniques are attractive in this case. However, most AL techniques used in computer vision are designed for general classification tasks. As such, these methods do not account for the specific difficulties or exploit the opportunities that arise when annotating individual pixels in 2D images and 3D voxels in image volumes. To remedy this, we introduce the concept of *geometric* uncertainty which can be combined with the more traditional *feature* uncertainty. Our basic insight is the following. Inconsistent predictions in a small region of an image are quite unlikely when an object of interest is expected to be smooth. If an algorithm produces such a prediction in an image region, it is a sign that this region is difficult for the algorithm. So, if an image patch is assigned a label that is different than those of its neighbours, it ought to be considered more carefully in annotation than patches that are assigned the same labels. Then, AL draws the attention of the oracle to the regions that are uncertain both in feature space and geometry space. Both high feature and high geometric uncertainty indicate that a datapoint is challenging for the algorithm, and thus, once annotated, it can influence the segmentation the most. We express both types of uncertainties in terms of entropy so that they can be combined in a principled way. Next, we introduce a novel definition of uncertainty for *multi-class* AL, which involves entropy and can thus be combined with the geometric priors as well. We evaluate our approach on Electron Microscopy and Magnetic Resonance image volumes, as well as on natural images of horses and human faces. As a result, we demonstrate a substantial performance increase over state-of-the-art AL approaches.

Data-driven active learning through regression [90] Motivated by the fact that none of the proposed AL strategies clearly outperforms others in all cases, we propose a new data-driven AL approach that attempts to automatically select the best strategy that minimises the generalisation error of the classifier. We introduce Learning Active Learning (LAL), a method that treats the query selection strategy as a regression problem. Given a trained classifier and its output for a specific sample without a label, we predict the reduction in generalization error that can be expected by adding the label to that datapoint. In practice, we show that we can train this regression function on synthetic data by using simple features, such as the variance of the classifier output or the predicted probability distribution over possible labels for a specific datapoint. The features for the regression are not domain-specific and this enables us to apply the regressor trained on synthetic data directly to other classification problems. Furthermore, if a sufficiently large annotated set can be provided initially, the regressor can be trained on it instead of on synthetic data. The resulting AL strategy is then tailored to the particular problem at hand. We show that LAL works well on real data from several different domains such as biomedical imaging, economics, molecular biology, and high energy physics. This query selection strategy outperforms competing methods without requiring hand-crafted heuristics and at a comparatively low computational cost.

Data-driven active learning through reinforcement learning Finally, we continue the search of a general-purpose AL method. We characterise two properties that are missing in most data-driven AL approaches: *flexibility* to be applied with various ML models and *transferability* between various application domains. We present a new data-driven AL method that tackles the above limitations. We formalise the annotation process as Markov decision process (MDP). We design state and action spaces in such a way that they achieve *flexibility* and *transferability* and we design a reward function that achieves *flexibility* of the dual AL objective to attain a pre-defined quality with the smallest annotation cost. Compared to manually-designed AL strategies, the focus of this work moves from designing a selection strategy to designing a reward function to reflect the AL objective. The best non-myopic AL strategy is found with a reinforcement learning (RL) technique. We evaluate the learnt strategy on multiple unrelated domains and show that it outperforms the baselines in a wide range of problem settings.

1.4 How to annotate?

If we leave aside for a moment the question of *which* datapoints should be annotated, we can concentrate on the question of *how* to obtain each of the annotations at a minimal cost. Various schemas can reduce the cost of labelling each datapoint by changing the annotation modality, providing hints, or making the human-computer collaboration more convenient. ML techniques often help to design the fastest annotation pipelines.

While answering to the question “*What to annotate?*” often results in application-independent solutions, the question “*How to annotate?*” can be rarely answered without an application context. Computer vision is rich in applications that are good candidates for intelligent annotation pipelines. Possible approaches to answer the question “*How to annotate?*” for these applications are very diverse, ranging from eye tracking while watching stacks of biomedical images [101] to games asking annotators to guess bird species from blurred images [43]. In this section we describe the recent work on intelligent annotation methods aiming to reduce the load for a human annotator. We regard the diversity of labelling methods from two perspectives: applications and techniques. For various applications we investigate how image priors help in designing the best annotation schemas. Then we try to identify the trends in annotation techniques that are present across CV tasks. Finally, we summarise our contributions in answering the question “*How?*” for two CV applications: 3D image volume segmentation and object detection.

1.4.1 How to annotate in computer vision applications

Depending on the task in a CV application, such as image classification, semantic segmentation or object detection, training data needs to come in a particular form. The data modality determines the way how the annotations are collected. Here we discuss several annotation pipelines for common CV tasks.

Image classification Image classification or recognition is arguably one of the most fundamental tasks in CV. Massive datasets with class labels, such as ImageNet [154] enabled the deep learning revolution in CV [96]. So, the question of label collection is of great importance in image classification. Apart from providing a class label for each image, alternative forms of feedback are often used for label collection. *Group-based labelling* reduces the labelling cost by assigning a class label to a group of several images at once. Groups can be formed with hierarchical clustering [196]. Then, by selecting the size of the group to annotate the method balances between the annotation cost and the expected accuracy. The trade-off depends on the granularity of the target class. Feedback in the form of “how close a given image is to various sets of images” can be used to build a classifier [188]. This form of feedback is practical in fine-grained recognition where classes are difficult to distinguish. Moreover, feedback on so-called mid-level attributes, such as “openness” of a scene or “furriness” of an object can help to efficiently annotate images for fine-grained classification [141, 21]. For example, the algorithm can convey its current guess on the class label to the annotator [21]. If the guess is wrong, the annotator indicates the correct class along with the attribute that is discriminative for the mistake. This form of feedback enables to quickly annotate many negative images by attribute propagation. Additionally, asking the annotators which regions of an image are the most discriminative of a certain class is useful for fine-grained classification [43]. The classifier learns to identify these regions to guide the predictions. The annotation of these regions can be made enjoyable by involving annotators into an interactive game.

Segmentation Image segmentation, both background-foreground and semantic segmentation, is another fundamental problem in CV. Human-computer collaboration is the core component of interactive segmentation methods [23, 153]. The annotator’s effort can be minimised by combining annotation modalities of varying coarseness, such as bounding box, sloppy contour, and tight polygon. This can be done by predicting the least expensive modality that is sufficient to initialise the segmentation algorithm [71]. For example, when an image is simple, a cheap bounding box would work well, but when an image is complex, a tight polygon is necessary.

Training data for the learning-based segmentation algorithms usually comes in a form of pixel-wise masks which are known to be very tedious to produce. To understand how the cost of the annotation influences the final segmentation, Zlateski et al. [208] study the performance of a convolutional neural network depending on the amount and coarseness of the training labels. In order to reach the same prediction quality, the number of annotations can be traded for their precision. However, the performance improves when more time is spent on annotations.

Instead of pixel-wise masks some works try to adapt cheaper data modalities. Scribbles (sparsely provided annotated pixels) are known to be very user-friendly to annotate images and video [108, 199, 133]. Scribbles annotations can be propagated from labelled to unlabelled pixels using a graphical model [108]. In video segmentation, scribbles are propagated through the video while preserving its consistency [133]. Another cheap annotation modality for segmentation is point-clicks [13, 15, 190]. Point-clicks on the object of interest can be incorporated into a weakly-supervised CNN with a special form of loss function [13]. If an algorithm computes many hypotheses of the segmentation, the annotator can click on the object boundaries to eliminate wrong hypotheses [73]. Polygons and pixel-wise masks are complementary label modalities: one-to-one correspondence between them can be easily established by assigning a mask to the area inside a polygon or by approximating the borders of a mask by a polygon. Then, the segmentation can be obtained either by predicting a class of pixels or by predicting the vertices of polygon with supervised [30] or reinforcement learning [2]. The polygon prediction task can involve the annotator to correct wrongly predicted vertices.

Weakly-supervised learning method deal with the question of how approximate annotation forms can be efficiently incorporated into the training procedure. For example, bounding boxes can be used by initialising segmentation masks with the region proposal candidates and iteratively refining them [39], or by incorporating the imprecision of the labels directly into a classification model [206]. Image-level labels are often available as a cheap annotation modality. Segmentation masks can be obtained by formulating the problem as Multiple Instance learning [146] or by solving a constrained optimisation of CNNs [142]. In some cases it is impossible to learn a correct segmentation mask from image-level label alone, for example, when two objects systematically occur together (*i.e.* trains and rails). In this case, Kolesnikov and Lampert [88] propose to use mid-level features of

CNN to discover types of objects assigned to each class and ask a human annotator to indicate false objects among them.

Object detection The data for training object detectors usually comes in the form of bounding boxes around objects of interest. In a standard interface [171] the annotator first clicks on the upper-right and then on the bottom-left corner of the imaginary bounding box. In addition to this, this procedure usually requires some adjustments for objects of complex shapes. Surprisingly large cost savings can be made at no quality loss by employing the following simple procedure [139]. The annotator clicks on the top-most, right-most, bottom-most and left-most points of the object. It defines exactly the same rectangle but it is much faster as the cognitive load for the annotator is lower. Furthermore, the object detectors themselves can help in the annotation task [203, 138]. The task is simplified if the annotator needs to correct hypotheses instead of drawing bounding boxes from scratch [203] or to identify wrong hypotheses [138].

Approximate forms of labels help to reduce the annotation cost. For example, eye tracking data [137] helps to extract the position and size of an object through a learnt mapping between eye-movements and bounding boxes. Point-clicks in the center of an imaginary bounding box can be used to train an object detector with multiple-instance learning [140]. Moreover, if two people perform this task, the distance between their clicks provides an additional cue on the size of the object. A similar approach is used for the spatio-temporal localisation of actions in videos [127].

Video annotation Annotating video data can be even more tedious than annotating image data. Nevertheless, video annotation can be treated with the same techniques as image annotation in tasks such as action recognition [127], gesture recognition and segmentation [189], score assignments [158], and clustering of human activities [87]. As eye-tracking data is handy for videos segmentation [101], action recognition [125], and action localization [127]. Vondrick et al. [187] provide an overview of interactive video annotation schemas.

In contrast to image annotation, video annotation can benefit from temporal priors. For example, when video frames are annotated with scribbles, motion cues help to detect trajectories of annotated pixels [133]. The observations from the detected trajectories can further guide the annotation process. Motion boundaries can be used to compute the segmentation hypothesis [73]. The hypotheses are presented to the annotator who decides to label additional boundaries to eliminate erroneous hypotheses.

Biomedical imaging Biomedical imaging applications are different from most other CV tasks because of the specialised data modalities (like isotropic or non-isotropic 3D image stacks) and the scarce availability of oracles (as only busy biomedical experts can provide reliable annotations). To deal with these problems many creative ways to annotate data are proposed. For example, to work with busy annotators, data for 3D

image segmentation can be annotated with a pedal or through voice commands to free the hands of the clinician during an operation [46]. Hands-free annotation is possible when an expert answers binary questions, such as whether a voxel lies inside the object of interest or not. Another way to deal with the shortage of annotators is to attract non-experts to these tasks. For example, non-experts' input can be provided with eye-tracking [101] or gamification [5, 65].

While we might avoid expensive label modalities during the annotation, we necessarily have to deal with them when proofreading the results. Proofreading is crucial in biomedical tasks as these applications are very sensitive to the quality of predictions. As an example, consider connectome reconstruction where segmentation is an important step. It has been shown that it is much easier to indicate which regions should be merged than how to split the regions [148]. Then, given an over-segmented image, the annotator indicates which regions should be joined together to obtain the correct segmentation mask and the algorithm guides the annotator to concentrate on the regions that are easier to correct and that have high impact on the final connectome. Proofreading is studied in other biomedical tasks, for example, the delineation of biomedical curvilinear structures such as neurons and blood vessels [131].

1.4.2 Trends in design of annotation pipelines

Normally, annotation in every CV application requires specialised methods. Still, similar ideas are encountered throughout various scenarios, for example: alternative query forms, such as eye-tracking and point-clicks, or using algorithmic guidance to simplify labelling by hypothesis correction. In this subsection we enumerate several techniques that are efficient in a broad range of application.

Batch-mode annotation Many interactive annotation pipelines suffer from long model update times which are necessary at every iteration. Moreover, modern CV algorithms require large training datasets and many AL iterations are needed if only one datapoint is added at each iteration. Therefore, batch-mode selection has become a standard way to increase efficiency by asking the expert to annotate more than one sample at every iteration [163, 61, 166, 201, 122, 48, 59]. This procedure amortises the total retraining time over many annotations and enables to annotate data in parallel by several annotators. Besides, in some situations it is easier for humans to provide labels to groups of examples [79]. Density-based AL strategies from Section 1.3.2 deal with the question how to form batches that ensure the diversity of the selection [48, 59]. Moreover, batches can be formed with hierarchical clustering [196] and annotator's cognitive efforts can be taken into account [187].

Crowdsourcing Crowdsourcing techniques have a long history in large-scale annotations and they are applied to nearly all CV tasks [94]. Although crowdsourcing does not

reduce the annotation time of each datapoint, it allows to collect large databases faster by getting many annotators to perform the task in parallel. Yet, additional questions need to be answered in crowdsourcing, such as how to ensure high annotation quality with noisy labels [140, 168, 195], how to estimate the expertise of annotators [25, 181, 195, 115, 114] or how to discover groups of annotators from various “schools of thought” [195]. Besides, we can integrate the disagreement into a classification scheme [168, 181, 195] or decide when to stop labelling a particular image [25].

Alternative query forms Many solutions to reducing the annotation cost rely on collecting data in a *weaker* form than the prediction task supports. For example, one can collect bounding boxes instead of masks for segmentation or image-level labels instead of bounding boxes for object detection. *Weakly supervised learning* methods integrate the weak labels into the optimisation framework. To compensate for weak supervision these methods usually incorporate image/video priors. As a result, weakly-supervised methods make it possible to solve the semantic segmentation task with point-clicks [13, 15, 190, 73, 140, 127], scribbles [108, 199, 133] or eye-tracking [101]; object detection with eye-tracking [137] or point-clicks [140, 127]; action recognition with eye-tracking [125]. Other pipelines collect annotations in the form of feedback on attributes. At first sight, it is a more time-consuming annotation modality. However, once attributes are collected, it could be possible to classify images into classes that do not have any training examples [100]. Besides, once attribute feedback propagates labels to unlabelled data, many samples get their labels assigned automatically and the average annotation time goes down. The benefits of learning these attributes are demonstrated in image clustering [98] and image classification [21].

It has been shown in the psychology literature that feedback in the form of comparisons is very natural for humans and thus, easier to produce. A way to incorporate comparisons into image classification is to ask the annotator to estimate how close a given image is to the various sets of images [188]. Then, the data in the form of similarity comparisons is used to train a classifier. Moreover, this form of feedback is especially beneficial to annotate data for fine-grained recognition where the classes are very close to each other. Besides, the feedback in the form of comparison is the only possible solution when the task is intrinsically ambiguous, such as assigning a numerical score to judge a skill or a status of a patient [158].

Assisted annotation In interactive annotation pipelines, the predictions by the algorithm can be presented to the user to facilitate the annotation process. These techniques are a part of many interactive annotation pipelines [25], for example, in image classification [21], object detection [203, 138] and gesture recognition in videos [189]. Assisted annotation helps to reduce a tedious annotation modality to an easier form, such as to accept or reject predictions instead of drawing bounding boxes [138] or to merge segmentation regions instead of drawing or splitting them [148].

Combining and choosing annotation modalities Instead of fixing the annotation modality prior to labelling, it can be adaptively chosen during the labelling process depending on the data [155, 71] and/or on the expertise of an annotator [181]. If fine-grained image classification is performed with a hierarchical taxonomy of the labels, the annotator can choose a particular depth of taxonomy depending on his confidence [181]. In interactive image segmentation, an algorithm can choose an annotation modality (bounding box, sloppy contour, tight polygon) that would be sufficient to initialise the segmentation [71]. In object detection the annotation modality (such as box verification, naming the object, or naming the image) can be adaptively selected for a given set of constraints (such as budget or precision) [155]. Also, it can be cheaper to use human assistance to convert one annotation modality (for example, bounding boxes) into another (for example, parts of objects) than to collect data of the target modality from scratch [24].

Gamification Gamification is used in attempt to attract more annotators and save financial costs by transforming a tedious process into an enjoyable activity. Games motivate the annotators to provide correct and precise labels and at the same time they allow to detect unreliable annotators automatically. A game for collecting labels in image classification is proposed by von Ahn and Dabbish [185]. Two partners in a game are assigned randomly from a pool of players and they are shown an image. Their task is to assign a label to an image without communicating with each other such that their labels coincide. The peekaboom game [186] adapts these ideas to the collection of object locations in images. Again, two players are paired randomly. Only one player sees an image with a word that names an object in the image. He then clicks on a part of the image to reveal it to the other player whose task is to guess the associated word. The game Bubbles [43] has a similar interaction model applied to fine-grained image classification. A player needs to identify the class of a blurred image where he can progressively reveal the full-resolution image by selecting small areas. Revealing parts of an image is costly for the player, therefore he is motivated to reveal as little as possible. This game generates data that allows a CV algorithm to understand what regions are the most discriminative of the given class. Then, an algorithm is trained to identify the “bubbles” of important information which are used in classification. ReferItGame [85] is used to annotate referring expressions. Referring expressions need to identify an object in a scene uniquely by describing its properties and relationship to other objects. The first player sees an image with a highlighted object and he needs to write an expression to refer to this object. The second player sees the image with expression and he needs to click on the correct object to gain points. Biomedical annotation tasks usually require an expert for reliable annotations, however, some attempts are made to exploit the availability of many non-specialists. The big difficulty is to engage users in biomedical applications. Gamification formulates the biomedical tasks in accessible and appealing ways. For example, features of objects can be encoded in the form of visual stars and the game asks players to collect them while flying a plane [5]. Another game, SwifTree [65] supports the task of delineation of 3D tree structures in a game of navigation.

1.4.3 Contributions towards answering “*How to annotate?*”

Batch annotation in 3D segmentation [89, 91] We propose a method to facilitate annotating 3D image volumes with segmentation masks by a batch-mode selection that reduces the 3D task to a 2D one. To collect labels we ask an annotator to identify the class of a given supervoxel. Consider a naive implementation of batch selection in a 3D image stack. It would force the annotator to randomly view and label patches in the volume regardless of where they are, which is extremely cumbersome. Our approach avoids this by first selecting a planar patch of arbitrary orientation in the 3D volume and then allowing the user to quickly label positive and negative pixels within it. To achieve this, we develop an efficient algorithm that searches for a 2D plane containing a patch with the most informative samples. The key idea behind the efficiency of this algorithm is to evaluate entire subsets of the parameter space using a bounding function and progressively evaluating the best looking subsets. This streamlines the annotation process in 3D volumes so that annotating them is no more cumbersome than annotating ordinary 2D images. As a result, the labelling process is speeded up at no significant computational cost. Our method is tested on Electron Microscopy image stacks for the task of mitochondria segmentation and on Magnetic Resonance image stacks for brain tumour segmentation.

Intelligent dialogs for bounding box annotations [92] Another application where we reduce the cost of labelling is object detection. To this end, we introduce Intelligent Annotation Dialogs (IAD). Given an image, detector, and target class to be annotated, the aim of IAD is to automatically choose the sequence of annotation actions that results in producing a bounding box in the least amount of time. The possible annotation actions include manual drawing of bounding boxes, extreme clicking, and verification of boxes produced by a weak detector. We train the IAD agent to select the type of action based on previous experiences in annotating images. Our method automatically adapts to the difficulty of the image, the strength of the detector, the desired quality of the boxes, and other factors. This is achieved by modelling the episode duration as a function of problem properties. We consider two ways to do this, either a) by predicting whether a box proposed by weak a detector will be accepted by the user, or b) by directly predicting the duration of an annotation with a reinforcement learning agent. We evaluate IAD by annotating bounding boxes in the PASCAL VOC 2007 dataset in several scenarios. Our experiments demonstrate in all scenarios that thanks to its adaptive behaviour IAD speeds up box annotation compared to manual box drawing alone, or box verification series alone. Moreover, it outperforms any fixed combination of them in most scenarios. Finally, we demonstrate that IAD learns useful strategies in a complex realistic scenario where the detector is continuously improved with the growing amount of the training data.

2 Geometry-Based Active Learning for Image Segmentation

2.1 Introduction

Machine learning techniques are a key component of modern approaches to segmentation, making the need for sufficient amounts of training data critical. As far as images of everyday scenes are concerned, this is addressed by compiling large training databases and obtaining the ground truth via crowd-sourcing [115, 109], but at a high cost. By contrast, in specialized domains such as biomedical image processing, this is not always an option because only experts whose time is scarce and precious can annotate images reliably. This stands in the way of wide usage of many state-of-the-art segmentation algorithms, which require large amounts of annotated data for training. The problem is even more acute for multi-class segmentation, which requires even larger training sets and more sophisticated interfaces to produce them [81]. Thus, AL is particularly well suited for these problems.

However, most AL techniques [81, 84, 80, 182, 124, 121, 114], are inspired by earlier methods developed primarily for general tasks or Natural Language Processing [179, 103]. As such, they rarely account for the specific difficulties or exploit the opportunities that arise when annotating individual pixels in 2D images and 3D voxels in image volumes.

More specifically, 3D stacks such as those depicted by Figure 2.1 are common in the biomedical field and are particularly challenging, because it is difficult for users to quickly figure out what they are looking at and annotation tools are often cumbersome. In this chapter¹ we describe our approach to AL that is geared towards segmenting 3D stacks while accounting for geometric constraints of region shapes and thus making the annotation process convenient. Our approach can be applied both to background-foreground and multi-class segmentation of both ordinary 2D images and 3D image volumes. Our main contributions are as follows:

¹This chapter is based on Konyushkova et al. [89, 91]

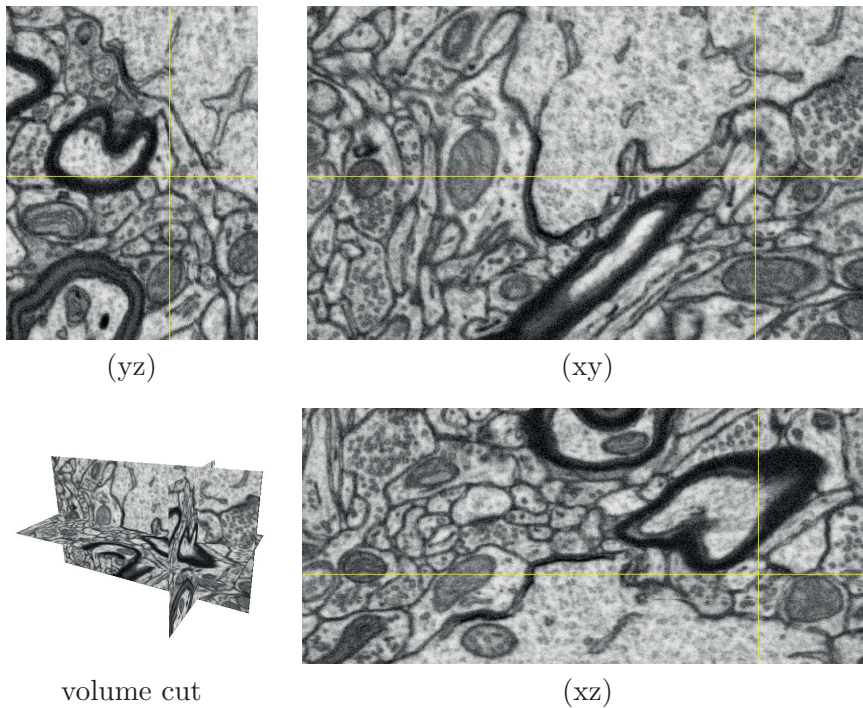


Figure 2.1 – Interface of the FIJI Visualization API [160], which is extensively used to interact with 3D image stacks. The user is presented with three orthogonal planar slices of the stack. While effective when working slice by slice, this is extremely cumbersome for annotating 3D organelles.

- We exploit geometric priors to select the image data for annotation more effectively, both for background-foreground and multi-class segmentation.
- We streamline the annotation process of 3D volumes so that annotating them is no more cumbersome than annotating ordinary 2D images, as depicted by Figure 2.2.

In the remainder of this chapter, we first discuss why current approaches to binary and multi-class AL are not the most effective when dealing with pixels and voxels (Section 2.2). Then, in Section 2.3 we give a short overview of our approach before discussing in details our use of geometric priors (Section 2.4) and how we search for an optimal cutting plane to simplify the annotation process (Section 2.5). Finally, in Section 2.6, we compare our results against state-of-the-art techniques in a few challenging cases. Additionally, we test our novel multi-class AL on image classification tasks. In conclusion, we provide experiments that illustrate the role of human intuition in the labelling procedure.

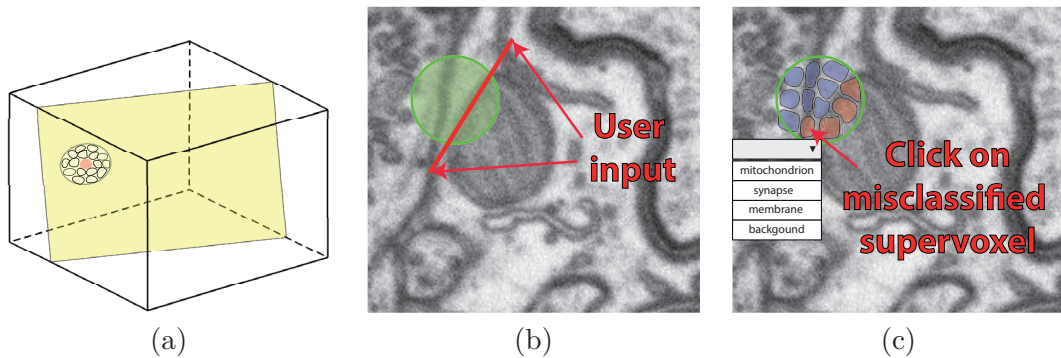


Figure 2.2 – Our approach to annotation. (a) The system selects an optimal plane in an arbitrary orientation and presents the user with a patch that is easy to annotate. The area to annotate is shown as part of the full 3D stack. (b) User interface, the planar patch the user would see. In case of two classes present in the patch, it could be annotated by clicking twice to specify the red segment that forms the boundary between the inside and outside of a target object within the green circle. (c) The other way to annotate data is to correct mistakes in the current prediction. Supervoxels predicted to be mitochondria are shown in red, background in blue. If a user clicks on the misclassified supervoxel he can select the correct class among proposed. Best viewed in color.

2.2 Related work

AL selection strategies are rarely designed to take advantage of image specificities when labelling individual pixels or voxels, such as the fact that a neighbourhood of pixels/voxels tends to have homogeneous labels. The segmentation methods presented in [104, 69, 205] do take such geometric constraints into account for classification purposes but not to guide AL, as we do.

Recently several authors realised the need to account for image properties in the AL selection for other computer vision tasks. For example, in human pose estimation the uncertainty depends on the spatial distribution of the detected body joints [111]. In brain connectome reconstruction, an algorithm of Plaza [148] can benefit from priors on how synapses can be situated in an image volume to result in a feasible reconstruction. Besides, his algorithm focuses on datapoints that have the largest influence on the final connectome. Some methods [72, 143, 59] account for the influence of neighbouring instances in AL selection by connecting datapoints in a graph as we do. However, the big difference to our approach is that they add edges between datapoints in a graph based on their feature similarity and not their geometric similarity.

As discussed in Section 1.4.2, batch-mode selection has become a standard way to increase efficiency by asking the expert to annotate more than one sample at a time [163, 61, 166, 48, 4]. But again, this has been mostly investigated in terms of semantic queries without due consideration to the fact that, in images, it is much easier for annotators to

quickly label many samples in a localized image patch than having to annotate random image locations. If samples are distributed randomly in a 3D volume, it is extremely cumbersome to label them using current image display tools such as the popular FIJI platform depicted by Figure 2.1. Thus, in 3D image volumes [104, 69, 57], it is important to provide the annotator with a patch in a well-defined plane, such as the one shown in Figure 2.2.

The technique of Top et al. [180] is the closest work to us as it asks users to label objects of interest in a plane of maximum uncertainty. Our approach has several distinctive features. First, the procedure we use to find the plane requires far fewer parameters to be set, as discussed in Section 2.5. Second, we search for the most uncertain patch in the plane and do not require the user to annotate the whole plane. Finally, our approach can be used in conjunction with an ergonomic interface that requires at most three mouse clicks per iteration when two classes are involved. Also, as we show in the result section, our method combined with geometric smoothness priors outperforms the earlier one.

2.3 Approach

We begin by broadly outlining our framework, which is set in a traditional AL context. That is, we wish to train a classifier for segmentation purposes, but have initially only few labelled and many unlabelled training samples at our disposal.

Since segmentation of 3D volumes is computationally expensive, supervoxels have been extensively used to speed up the process [6, 120]. In the remainder of this section and in Section 2.4, we will refer almost solely to supervoxels for simplicity but the definitions apply equally to superpixels when dealing with 2D images. We formulate the segmentation problem in terms of classifying supervoxels as a part of a specific target object. As such, we start by oversegmenting the image volume using the SLIC algorithm [1] and computing for each resulting supervoxel s_i a feature vector x_i . When dealing with ordinary 2D images, we simply replace the 3D supervoxels with 2D superpixels, which SLIC can also produce. Our AL problem thus involves iteratively finding the next set of supervoxels that should be labelled by an expert to improve segmentation performance as quickly as possible. To this end, our algorithm proceeds as follows:

1. Train a classifier on the labelled supervoxels \mathcal{L}_t and use it to predict the class probabilities for the remaining supervoxels \mathcal{U}_t with $t = 0$.
2. Score \mathcal{U}_t on the basis of a novel uncertainty function that we introduce in Section 2.4. It is inspired by the geometric properties of images in which semantically meaningful regions tend to have smooth boundaries. Figure 2.3 illustrates its behaviour given a simple prediction map: Non-smooth regions tend to be assigned the highest uncertainty scores.

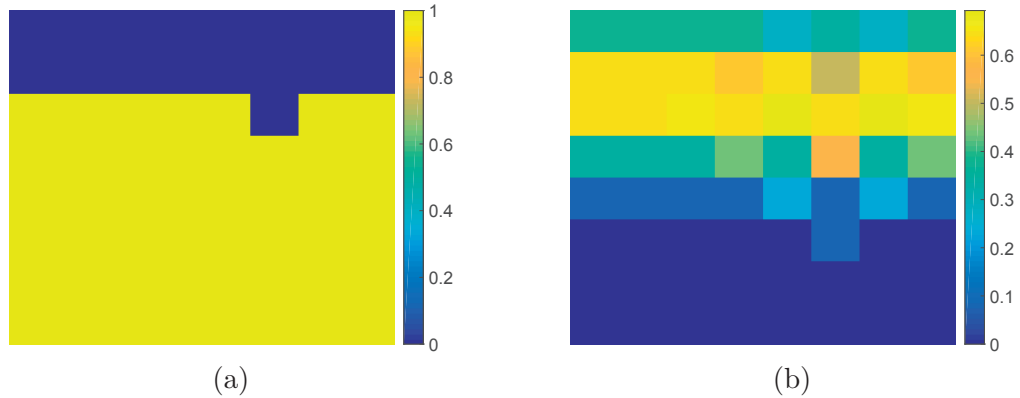


Figure 2.3 – Geometry-based uncertainty score. (a) Predicted binary classification map for an 8×8 image. In this example the classifier assigns the pixels coloured in yellow to class 1 with probability 1 and pixels coloured in blue to class 0, also with probability 1. Feature uncertainty has the lowest possible uncertainty value for all pixels as the classifier is certain of its predictions. (b) Geometric uncertainty score of Section 2.4.3. The area of transition between the two classes is given a high geometric uncertainty score. Its maximum is reached where the boundary is not smooth.

3. In volumes, select a 2D plane that contains a patch with the most uncertain supervoxels, as shown in Figure 2.2 and, in regular images, select a patch around the most uncertain superpixel. The expert can then effortlessly label an indicated 2D patch without having to examine the image data from multiple perspectives, as would be the case otherwise and as depicted by Figure 2.1. Furthermore, we can then design a simple interface that lets the user label supervoxel or superpixel batches with just a few mouse clicks, as shown in Figure 2.2 and described in Section 2.6.
4. Sets \mathcal{L}_t and \mathcal{U}_t are updated for $t = t + 1$ and the process is repeated until the segmentation quality is satisfactory.

2.4 Geometry-based active learning

Most AL methods were developed for general tasks and operate exclusively in feature space, thus ignoring the geometric properties of images and more specifically their geometric consistency. As discussed in Section 1.3.2, Uncertainty Sampling (US) is designed to focus the annotators' attention on samples for which image features do not yet provide enough information for the classifier to decide what label to assign them. It selects samples that are *uncertain* in feature space to be annotated first so that classifier is updated with the largest amount of information. In this chapter, we will refer to this family of approaches as *Feature Uncertainty (FU \mathbf{n})*. These methods are both effective

and computationally inexpensive, thus, they are chosen as a basis of our work. However, they do not account for image geometry to clue which samples may be mislabelled.

To remedy this, we first introduce the concept of *Geometric Uncertainty (GUn)* and then show how to combine it with **FUn**. Our basic insight is that supervoxels that are assigned a label different from that of their neighbours ought to be considered more carefully than those that are assigned the same label, as illustrated by Figure 2.3. In this 2D toy example, pixels near classification boundaries in the image space, as opposed to the feature space, are marked as being more uncertain and those near irregular parts of the boundary even more.

We express both kinds of uncertainties in terms of entropy so that we can combine them in a principled way. Using Shannon entropy of the prediction is often inefficient in multi-class classification, which we empirically demonstrate in experimental result. In order to combine these uncertainty measures in multi-class segmentation case, a new uncertainty criterion is needed.

2.4.1 Uncertainty measures

For each supervoxel s_i and each label y in a set \mathcal{Y} of possible labels, let $p(y_i = y | x_i)$ be the probability that its label y_i is y , given the corresponding feature vector x_i . In this section we are not concerned with the question of how this probability is obtained. For background-foreground segmentation, we take \mathcal{Y} to be $\{0, 1\}$. In the multi-class scenario, \mathcal{Y} is a larger set, such as {background, hair, skin, eyes, nose, mouth} for face segmentation.

We start with *total entropy*: a well known uncertainty measure defined by Shannon entropy of Equation (1.1). By definition, it is not restricted to the binary case and can be used straightforwardly in the multi-class scenario as well. For example, values of *total entropy* for 3-class classification are illustrated in Figure 2.4. Next, we introduce two novel uncertainty measures which are both entropic in nature, but account for different properties of the predicted probability distribution.

Selection entropy

When there are more than two elements in \mathcal{Y} , another way to evaluate uncertainty is to consider the label b_1 with highest probability against all others taken together. For $b_k \in \{b_1, \bar{b}_1\}$ this yields a probability distribution

$$p^s = p(y_i = b_k | x_i), \tag{2.1}$$

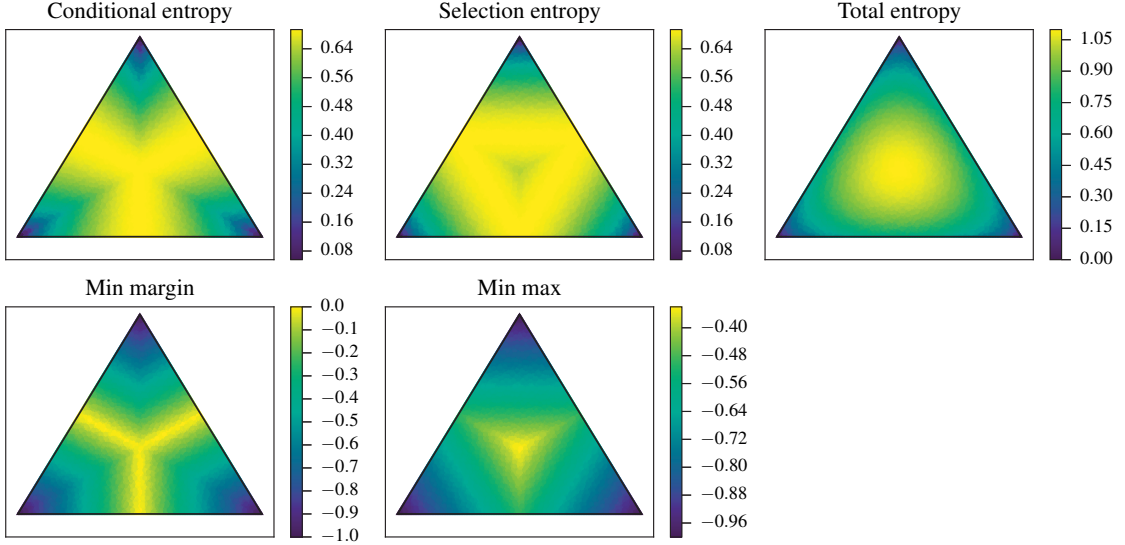


Figure 2.4 – Measures of Feature Uncertainty in a three-class problem. In each triangle the color denotes the uncertainty as a function of the three probabilities assigned to each class, which sum to 1. The three corners correspond to a point with probability 1 belonging to one of the three classes and therefore no uncertainty. By contrast, the center point can belong to any class with equal probability. For better comparison, we inverted some values such that yellow corresponds to higher uncertainty and dark blue to the lower uncertainty. Top: entropy-based measures of Section 2.4.1; Bottom: Measures proposed in the book of Settles [164].

such that $p(y_i = \bar{b}_1 | x_i) = \sum_{y \in \mathcal{Y} \setminus b_1} p(y_i = y | x_i)$. Then, we compute the entropy of the resulting probability distribution over two classes as *selection entropy* H^s

$$H^s = H(p^s). \quad (2.2)$$

This definition of uncertainty is motivated by our desire to minimize the number of misclassified samples by concentrating on the classifier’s decision output. The *selection entropy* uncertainty values for 3-class classification are depicted in Figure 2.4. Notice that selection entropy avoids choosing the datapoints with the equal probability assigned to every class when the number of classes is greater than two. This makes sense in practice because an example that is confused between all classes of the multi-class problem is likely to be an outlier.

Conditional entropy

Another way to evaluate uncertainty in a multi-class scenario is to consider how much more likely the top label candidate is than the second one. More precisely, let b_1 and b_2 be two highest ranked classes for a supervoxel s_i , with $p(y_i = b_1 | x_i) > p(y_i = b_2 | x_i) >$

$p(y_i = b_j | x_i), \forall b_j \neq b_1, b_2$. If we believe that one of them truly is the correct class, we can condition on this fact. For $\forall b_k \in \{b_1, b_2\}$ this yields

$$p^c = p(y_i = b_k | x_i, y_i^* \in \{b_1, b_2\}) = \frac{p(y_i = b_k | x_i)}{p(y_i = b_1 | x_i) + p(y_i = b_2 | x_i)}, \quad (2.3)$$

where y_i^* stands for the true class label. We then take the *conditional entropy* uncertainty to be the Shannon entropy of this probability distribution, which is

$$H^c = H(p^c). \quad (2.4)$$

This definition of uncertainty is motivated by the fact that the classifier is rarely confused about all possible classes. More typically, there are two classes that are hard to distinguish and we want to focus on those. For example, when trying to recognize digits from 0 to 9, it is unusual to find samples that resemble all possible classes with equal probability. If such a sample is found, it is likely to be an outlier and not very informative for the classifier. At the same time, there are many cases in which 3 and 5 are not easily distinguishable and such samples could help to improve the classifier. Recall that according to selection entropy, an example that is equally likely to be any of the digits should be avoided as a potential outlier. An example of *conditional entropy* uncertainty values for 3-class classification is shown in Figure 2.4.

2.4.2 Feature uncertainty (FUn)

In practice, we estimate $p(y_i = y | x_i)$ by means of a classifier trained using parameters θ and we denote the distribution probability by p_θ . Then, any of the uncertainty measures from Section 2.4.1 can be applied to the probability distribution $p_\theta(y_i = y | x_i) \forall y \in \mathcal{Y}$ resulting in *Feature Total Entropy* H from Equation (1.1), *Feature Selection Entropy* H^s from Equation (2.2) and *Feature Conditional Entropy* H^c from Equation (2.4). While all Feature Uncertainty measures are equivalent in the binary classification case, they behave quite differently in a multi-class scenario, as shown in the top row of Figure 2.4. Furthermore, even though our *selection entropy* and *conditional entropy* measures are in the same spirit as the *min margin* and *minmax* measures of Section 1.3.2 [164] (bottom row of Figure 2.4), their selection is still different. The main motivation behind these new uncertainty estimates is the fact that they enable the combination with geometric priors, as shown in Section 2.4.4. In the remainder of the chapter, we will refer to any one of these three uncertainty measures as the *Feature Uncertainty* H^θ .

2.4.3 Geometric uncertainty (GUn)

Estimating the uncertainty as described above does not explicitly account for correlations between neighbouring supervoxels. To account for them, we can estimate the entropy of

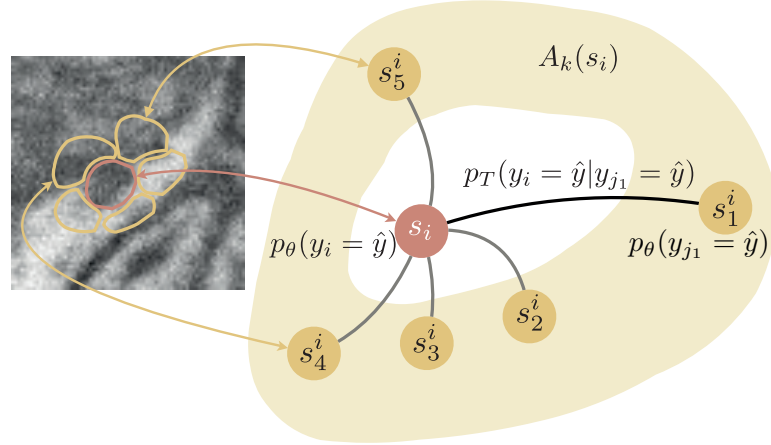


Figure 2.5 – Image represented as a graph. We treat supervoxels as nodes in the graphs and edge weights between them reflect the probability of transition of the same label to a neighbour. Supervoxel s_i has k neighbours from $A_k(i) = \{s_1^i, s_2^i, \dots, s_k^i\}$, $p^T(y_i = y | y_j^i = y)$ is the probability of node s_i having the same label as node s_j^i , $p^\theta(y_i = y | x_i)$ is the probability that y_i , class of s_i , is y , given only the corresponding feature vector x_i

a different probability, specifically the probability that supervoxel s_i belongs to class y given the classifier predictions of its neighbours and which we denote $p_G(y_i = y)$.

To this end, we treat the supervoxels of a single image volume as nodes of a directed weighted graph G whose edges connect neighbouring supervoxels, as depicted in Figure 2.5. We let $A_k(s_i) = \{s_1^i, s_2^i, \dots, s_k^i\}$ be the set of k nearest neighbours of s_i and assign a weight inversely proportional to the Euclidean distance between the voxel centers to each one of the edges. This simple definition makes most sense when the supervoxels are close to being spherical, which is the case when using an algorithm of Achanta et al. [1]. For each node s_i , we normalize the weights of all incoming edges so that their sum is one and treat this as the probability $p_T(y_i = y | y_j = y)$ of node s_i having the same label as node $s_j^i \in A_k(s_i)$. In other words, the closer two nodes are, the more likely they are to have the same label.

To define $p_G(y_i = y)$ we use a random walk procedure on G [117], as it reflects well our smoothness assumption and has been extensively used for image segmentation purposes [58, 180]. Given the transition probabilities $p_T(y_i = y | y_j = y)$, we can compute the probabilities p_G iteratively by initially taking $p_G^0(y_i = y)$ to be $p^\theta(y_j = y | x_j)$ and then iteratively computing

$$p_G^{\tau+1}(y_i = y) = \sum_{s_j \in A_k(s_i)} p_T(y_i = y | y_j = y) p_G^\tau(y_j = y). \quad (2.5)$$

Note that $p_\theta(y_j = y | x_j)$, $p_G^0(y_i = y)$ and $p_G^{\tau+1}(y_i = y)$ are vectors whose dimension is the cardinality of \mathcal{Y} , the set of all possible labels. The above procedure propagates the labels of individual supervoxels into their neighbourhood and the number of iterations, τ_{max} , defines the radius of the neighbourhood involved in the computation of p_G for s_i , thus encoding smoothness priors. Figure 2.3 shows the result of this computation for a simple 8×8 image with initial prediction of a classifier as shown on the left and $k = 4$ neighbours with equal edge weights. We apply $\tau_{max} = 4$ iterations and the resulting geometric uncertainty on the right shows how smoothness prior is reflected in the uncertainty: Non-smooth boundaries receive the highest uncertainty score.

Given these probabilities, we can use the approaches of Section 2.4.1 to compute the Geometric Uncertainty H^G for the probability distribution $p_G(y_i = y | x_i) \forall y \in \mathcal{Y}$ as *Geometric Total Entropy* H_G , *Geometric Selection Entropy* H_G^s and *Geometric Conditional Entropy* H_G^c , respectively.

2.4.4 Combining feature and geometric uncertainties

Finally, given a trained classifier, we can estimate both **FUn** and **GUn**. To use them jointly, we should ideally estimate the joint probability distribution $p_{\theta,G}(y_i = y | x_i)$ and the corresponding joint entropy. As this is not modeled by our classification procedure, we take advantage of the fact that the joint entropy is upper bounded by the sum of individual entropies H^θ and H^G . Thus, for each supervoxel, we take the *Combined Uncertainty (CUn)* to be

$$H^{\theta,G} = H^\theta + H^G \tag{2.6}$$

that is, the upper bound of the joint entropy. The same rule can be equally applied to the *total entropy* and entropy-based functions *selection entropy* and *conditional entropy*. This principled way to combine the uncertainties gives much better results than simple summing up of the scores of the methods discussed in the work of Settles [164] *min margin* and *minmax*. In practice, using this measure means that supervoxels that individually receive uncertain predictions and are in areas of non-smooth transition between classes will be considered first, as depicted by Figure 2.3. Note that the AL method of Mosinska et al. [130] that is based on Zhou’s propagation [205] is similar to the one we use as it takes into account the geometric location of datapoints in AL. However, it is adapted for the application of curvilinear structures and operates exclusively on H^G . We experimentally observed on our datasets that considering the upper bound on the joint entropy from Equation 2.6 results in a significant improvement in the learning speed.

In terms of computation cost of **CUn**, our MATLAB implementation of *Combined Total Entropy* on 10 volumes of resolution $176 \times 170 \times 220$ of the MRI dataset from Section 2.6.3 takes 1.4s per iteration (2.3 GHz Intel Core i7, 64-bit). The time performance is extremely important in the interactive applications.

2.5 Batch-mode geometry query selection

The simplest way to exploit the **CUn** from Section 2.4.4 would be to pick the most uncertain supervoxel, ask the expert to label it, retrain the classifier, and iterate. A more effective way is to find appropriately-sized batches of uncertain supervoxels and ask the expert to label them all at once before retraining the classifier. As discussed in Section 1.4.2, this is referred to as batch-mode selection, which usually reduces the time-complexity of AL. However, a naive implementation would force the user to randomly view and annotate several supervoxels in 3D volumes regardless of where they are. This would not be user friendly as they would have to navigate a potentially large volume at each iteration. In this section, we therefore introduce an approach to using the uncertainty measure described in the previous section to first select a planar patch in 3D volumes and then allow the user to quickly label supervoxels within it, as shown in Figure 2.2.

In practice, we operate on SLIC superpixels/supervoxels [1] that are roughly circular/spherical. We allow annotator to only consider circular regions within planar patches such as the one depicted in Figures 2.2 and 2.12. These can be understood as the intersection of a sphere with a plane of arbitrary orientation.

Recall from Section 2.4, that we can assign to each supervoxel s_i an uncertainty estimate $U(s_i)$ in one of several ways. Whichever one we choose, finding the circular patch of maximal uncertainty ρ^* can be formulated as finding

$$\rho^* = \arg \max_{\rho} \sum_{s_j \in \rho} U(s_j), \quad (2.7)$$

where the summation occurs over the voxels that intersect the plane and are within the sphere.

Since Equation (2.7) is linear in $U(s_j) \geq 0$ for any given voxel s_i , we design a branch-and-bound approach to finding the plane that yields the largest uncertainty. It recursively eliminates whole subsets of planes and quickly converges to the correct solution. Whereas an exhaustive search would be excruciatingly slow, our current MATLAB implementation on MRI dataset takes 0.024s per plane search with the same settings as in Section 2.4.4. This means that an efficient implementation of the entire pipeline could be real-time, which is critical for acceptance by users of such an interactive method.

As discussed above, in theory, this procedure could be used in conjunction with any one of the uncertainty measures defined in the previous section. In practice, as shown in Section 2.6, it is most beneficial when used in combination with the geometry-aware criterion of Section 2.4.4. We describe our branch-and-bound plane-finding procedure in more detail below.

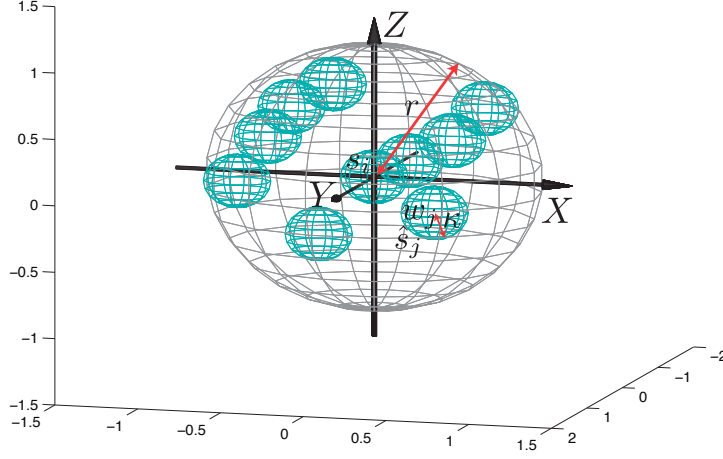


Figure 2.6 – Supervoxel approximation. Each supervoxel can be considered as a sphere of radius κ and center w_j . We are interested in the neighbourhood of supervoxel s_i defined by a sphere of radius r .

2.5.1 Parametrizing the search space

Let us consider a spherical volume centered at supervoxel s_i , such as the one depicted by Figure 2.6. Since the SLIC superpixels/supervoxels are always roughly circular/spherical, any supervoxel s_j can be well approximated by a spherical object of radius κ , set to a constant for a particular dataset, and its center w_j . We will refer to such an approximation as \hat{s}_j . Then, every $\hat{s}_j = (w_j, \kappa)$ is characterized by its center w_j and the common radius κ .

Let \hat{S}_i^r be the set of supervoxels within the distance r from \hat{s}_i , that is,

$$\hat{S}_i^r = \{\hat{s}_j = (w_j, \kappa) \mid \|w_j - w_i\| \leq r\}. \quad (2.8)$$

If we take the desired patch size to be r , we can then operate exclusively on the elements of \hat{S}_i^r . Let \mathcal{P}_i be the set of all planes bisecting it at the center of \hat{s}_i . As we will see below, our procedure requires defining planes, area splits of approximately equal size, and supervoxel membership to certain areas and planes. To make this easy to do, we parametrize planes in \mathcal{P}_i as follows.

Let us consider a plane $\rho \in \mathcal{P}_i$, such as the one shown in yellow in Figure 2.7. It intersects the XY plane along a line characterized by a vector \vec{v}_1 , shown in blue. Without loss of generality, we can choose the orientation of \vec{v}_1 so that its X coordinate is positive and denote by ϕ the angle between the negative component of axis $-Y$ and \vec{v}_1 . Similarly, let us consider the intersection of ρ with YZ plane and characterize it by the vector \vec{v}_2 with

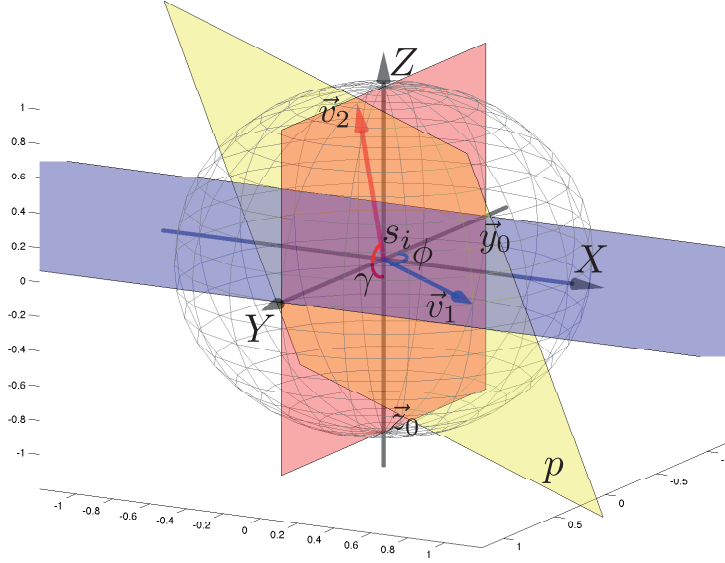


Figure 2.7 – Coordinate system for plane selection. A circular patch is defined as the intersection of a plane with a sphere. Plane ρ_i (yellow) is parametrised by two angles, ϕ and γ ; ϕ is the angle between the negative component of axis $-Y$ and plane intersection with XY (blue), similarly, γ is the angle between $-Z$ and plane intersection with YZ (red). Best seen in color.

a positive Y coordinate and shown in red. Now let γ be the angle between $-Z$ and \vec{v}_2 . We can now parametrize the plane ρ by the two angles $\phi \in [0, \pi)$ and $\gamma \in [0, \pi)$ because there is one and only one plane passing through two intersecting lines. We will refer to (ϕ, γ) as the plane's angular coordinates. Finally, let $\mathcal{C}_i^r(\rho)$ be the set of supervoxels $\hat{s}_j \in \hat{S}_i^r$ lying on ρ , that is,

$$\mathcal{C}_i^r(\rho) = \{\hat{s}_j \in \hat{S}_i^r \mid d(\rho, w_j) \leq 2\kappa\}, \quad (2.9)$$

where d is the distance from a supervoxel center to the plane.

The set \mathcal{P}_i can be represented by the Cartesian product $[0, \pi) \times [0, \pi)$ of the full ranges of ϕ and γ . Let $\Phi = [\phi_{\min}, \phi_{\max})$ and $\Gamma = [\gamma_{\min}, \gamma_{\max})$ be two angular intervals. We will refer to a set of planes with angular coordinates in $\Phi \times \Gamma$ as the *corridor* $\Omega = \Phi \times \Gamma$, as illustrated by Figure 2.8. The boundaries of this corridor are defined by planes $\rho_1 = (\phi_{\min}, \gamma_{\min})$, $\rho_2 = (\phi_{\min}, \gamma_{\max})$, $\rho_3 = (\phi_{\max}, \gamma_{\min})$ and $\rho_4 = (\phi_{\max}, \gamma_{\max})$.

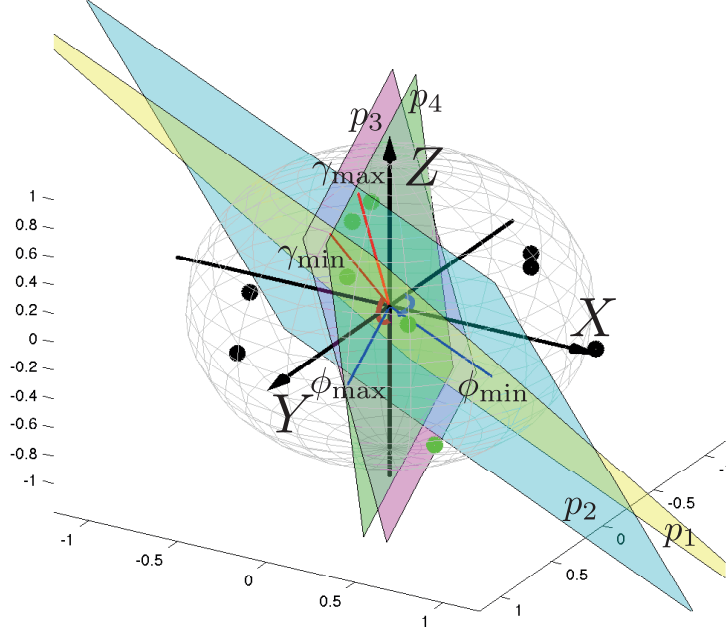


Figure 2.8 – A corridor is a union of the areas between planes ρ_1 and ρ_4 as well as between ρ_2 and ρ_3 . The green points depict supervoxels included in corridor Ω while black points depict supervoxels outside of it. Best seen in color.

2.5.2 Searching for the best bisecting plane

Uncertainty of planes and corridors

Recall that we assign to each supervoxel \hat{s}_j an uncertainty value $U(\hat{s}_j) \geq 0$. We take the uncertainty of plane ρ to be

$$U(\rho) = \sum_{\hat{s}_j \in C_i^+(\rho)} U(\hat{s}_j). \quad (2.10)$$

Finding a circular patch ρ^* of maximum uncertainty then amounts to finding

$$\rho^* = (\phi^*, \gamma^*) = \arg \max_{\rho \in \mathcal{P}_i} U(\rho). \quad (2.11)$$

Similarly, we define the uncertainty of a corridor as the sum of the uncertainty values of all supervoxels lying between the four planes bounding it, between ρ_1 and ρ_4 , and between ρ_2 and ρ_3 as depicted by Figure 2.8. We therefore write

$$U(\Omega) = \sum_{\hat{s}_j \in C_i^+(\Omega)} U(\hat{s}_j), \quad (2.12)$$

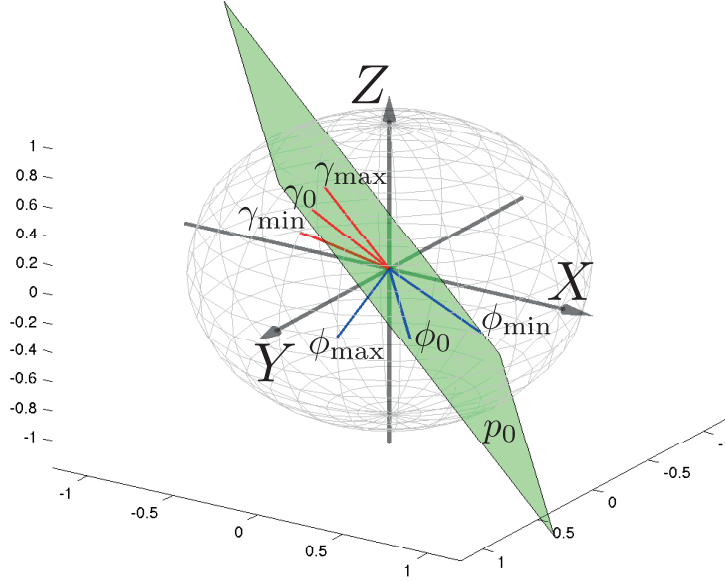


Figure 2.9 – Bounding function and corridor splitting procedure. The score of the plane p_0 is less or equal to the score of all the points included between two planes p_{\min} and p_{\max} : $U(\rho_0) < U(\Omega)$. We split the corridor Ω into corridors $[\phi_{\min}, \phi_0) \times [\gamma_{\min}, \gamma_0)$, $[\phi_{\min}, \phi_0) \times [\gamma_0, \gamma_{\max})$, $[\phi_0, \phi_{\max}) \times [\gamma_{\min}, \gamma_0)$ and $[\phi_0, \phi_{\max}) \times [\gamma_0, \gamma_{\max})$ and evaluate their uncertainty values. Among all available sectors we select a sector with the highest value to be split next. Best seen in color.

where $\mathcal{C}_i^r(\Omega)$ represents the supervoxels lying between the four bounding planes. In practice, a supervoxel is considered to belong to the corridor if its center lies either between ρ_1 and ρ_4 or between ρ_2 and ρ_3 , or is no further than κ away from any of them. When the angles are acute, this is easily decided by checking that the dot product of the voxel coordinates with the plane normals have the same sign, provided that these normals orientations are chosen so that they all point inside the corridor.

Branch and bound

To solve Equation 2.11 and find the optimal circular patch, we use a branch-and-bound approach. It involves quickly eliminating entire subsets of the parameter space $\Phi \times \Gamma$ using a bounding function [99, 26], a recursive search procedure, and a termination criterion, which we describe below.

Bounding function Let us again consider the corridor $\Omega = [\phi_{\min}, \phi_{\max}) \times [\gamma_{\min}, \gamma_{\max})$ bounded by the four planes ρ_1 to ρ_4 . Let us also introduce the plane $\rho_0 = (\alpha_1 \phi_{\min} + \beta_1 \phi_{\max}, \alpha_2 \gamma_{\min} + \beta_2 \gamma_{\max})$, where $\alpha_1 + \beta_1 = 1, \alpha_2 + \beta_2 = 1$ depicted by Figure 2.9. Given that $U(\hat{s}_j) \geq 0$ and that Equation 2.11 is linear in $U(\hat{s}_j)$, the uncertainty of ρ_0 will always be less or equal to that of Ω . This allows us to bound the uncertainty of any plane

Chapter 2. Geometry-Based Active Learning for Image Segmentation

from above and to search for the solution only within the most promising parameter intervals, as follows.

Search procedure As in work of Lampert et al. [99], we maintain a priority queue L of corridors. At each iteration, we pop the corridor $\Omega_{\max}^j = [\rho_1^j, \rho_2^j, \rho_3^j, \rho_4^j]$ with the highest uncertainty $U(\Omega_{\max}^j)$ according to Equation 2.12 and process it as follows.

We introduce two new angles $\phi_0^j = (\phi_{\min}^j + \phi_{\max}^j)/2$ and $\gamma_0^j = (\gamma_{\min}^j + \gamma_{\max}^j)/2$ and split the original parameter intervals into two, as shown in Figure 2.9. We compute the uncertainty of corridors $[\phi_{\min}, \phi_0] \times [\gamma_{\min}, \gamma_0)$, $[\phi_{\min}, \phi_0] \times [\gamma_0, \gamma_{\max})$, $[\phi_0, \phi_{\max}] \times [\gamma_{\min}, \gamma_0)$ and $[\phi_0, \phi_{\max}] \times [\gamma_0, \gamma_{\max})$ and add them to the priority queue L .

Note, that we always operate on acute angles after the first iteration with initialization $[0; \pi)$, which allows us to compute the uncertainty scores of corridors as discussed before.

Termination condition The search procedure terminates when the bisector plane $\rho_0 = (\phi_0, \gamma_0)$ of the corridor $\mathcal{C}_i^r(\Omega_{\max}^j)$ touches all the supervoxels from the corridor. To fulfil this condition it is enough to ensure that the distance d from any point in the corridor to a bisector plane is within the offset 2κ , that is,

$$d(\rho_0, \hat{s}_l) \leq 2\kappa, \forall \hat{s}_l \in \Omega_{\max}^j, \quad (2.13)$$

Since $U(\rho_0)$ is greater than the uncertainty of all the remaining corridors, which is itself greater than that of all planes they contain as discussed above, ρ_0 is guaranteed to be the optimal plane we are looking for.

Global optimization

Our branch-and-bound search is relatively fast for a single voxel, which we set to be a center in Section 2.5.1. However, it is not fast enough to perform for all supervoxels in a stack. Instead, we restrict our search to m most uncertain supervoxels in the volume.

We assume that the uncertainty scores are often consistent in small neighbourhoods, which is especially true for the geometry-based uncertainty of Section 2.4.3. By doing so it enables us to find a solution that is empirically observed to be close to the optimal one with a low value of m . In this way, the final algorithm first takes all supervoxels S with uncertainty U and selects the top m locations. Then, we find the best plane for each of the top m supervoxels and choose the best plane among them.

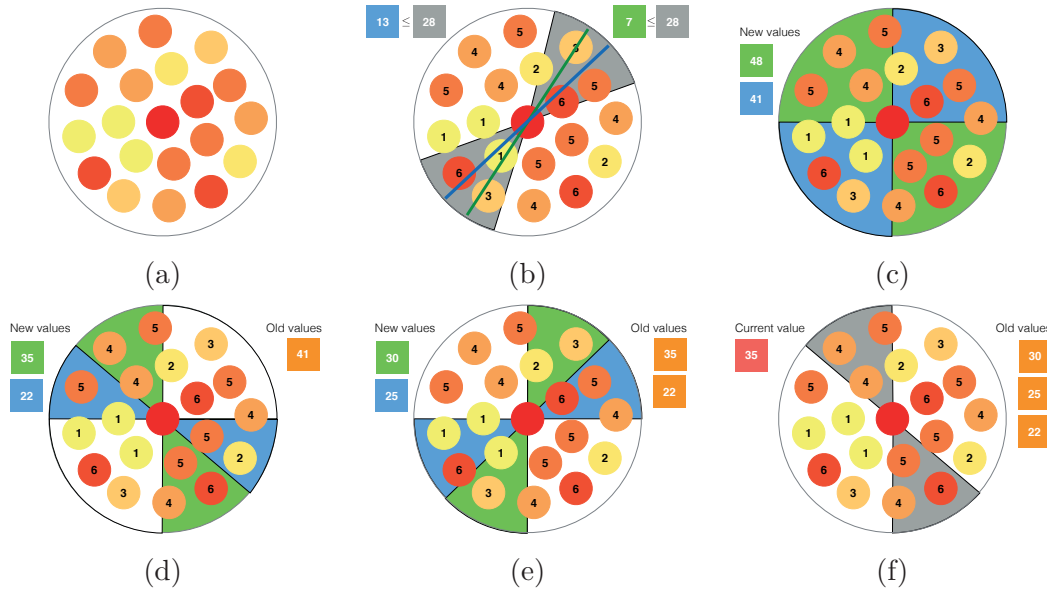


Figure 2.10 – Illustration of our branch-and-bound algorithm in 2D. a) The 2D-equivalent of searching for a plane in sphere is searching for a line in circle. b) The bounding function states that value of a sector is not smaller than the value of a line inside it. c) The procedure starts with splitting the whole parameter interval into 2 sectors. We compute the value of each sector and keep a priority queue of them. d) At each step of the procedure we divide the sector with the highest uncertainty in two new sectors by a bisector plane. e) and f) The procedure continues by splitting the sector with the highest value.

2.5.3 Illustration of search procedure in 2D

As it is difficult to represent graphically our branch-and-bound search procedure in 3D, for illustration purposes we describe it here on a 2D example. The 2D-equivalent of searching for a plane in sphere is searching for a line in circle.

In Figure 2.10(a) we show a circle with superpixels approximated by circles and where the color of each superpixel indicates how uncertain it is, with red being the most uncertain and yellow the least uncertain. Then, the task is to find a line of a maximum uncertainty, where the uncertainty of a line is defined as the sum of the uncertainties of the superpixels that it intersects. For example, Figure 2.10(b) demonstrates that the score of blue line is $6 + 1 + 6 = 13$ and the score of the green line is $3 + 1 + 3 = 7$. A corridor in 3D corresponds to a sector in 2D. An example of sector is shown in Figure 2.10(b) in grey with its score being $6 + 3 + 1 + 6 + 2 + 5 + 3 = 26$. Figure 2.10(b) also illustrates the bounding function condition: the score of any line inside the sector is no bigger than the score of the sector that includes this line, in our case, $13 \leq 26$ and $7 \leq 26$. The search procedure starts in Figure 2.10(c): We split the circle into blue and green sectors and compute their

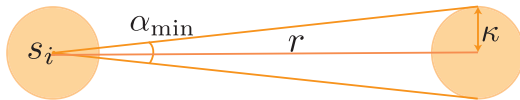


Figure 2.11 – Termination condition of our branch-and-bound procedure in 2D. The search terminates when a sector can fit at most one superpixel at the perimeter of the original circle. In this case, such a sector can be found as the one that exceeds the minimal angle α_{\min} .

scores (48 and 41). All the scores encountered during the search procedure are stored in a priority queue. At every iteration, the sector with the highest score is selected from the queue and split into two. For example, the green sector of Figure 2.10(c), whose score of 48 is the highest, is split into two new equal sectors resulting in 3 sectors depicted in Figure 2.10(d). The new uncertainty scores 35 and 22 are added to the priority queue. Next, the sector with the highest score is the blue sector of Figure 2.10(c). We split it into two new sectors in Figure 2.10(e). Next, we split the green sector of Figure 2.10(d) with the uncertainty score of 35. The procedure continues until the sector with the highest uncertainty can fit at most one superpixel at the perimeter of the original circle as shown in Figure 2.11.

2.6 Experiments

In this section, we evaluate our full approach on two different Electron Microscopy (EM) datasets and on one of Magnetic Resonance Imaging (MRI) dataset. We then demonstrate that **CUn** is also effective for natural 2D images. In multi-class MRI and multi-class natural 2D images of faces the extended version of our approach also results in enhanced performance.

2.6.1 Setup and parameters

For all our experiments, we used Boosted Trees selected by gradient boosting [177, 14] as our underlying classifier. This is a general-purpose classifier that can be trained fast, it provides probabilistic prediction and it extends naturally to the multi-class scenario. However, there exist no closed form solution for the optimization of Boosted Trees when new points are added. Thus, AL strategies such as expected model change or expected error reduction are not suitable to be applied with our classifier, because there is no efficient incremental update rule and re-training a classifier can take hours for one model update for a typical dataset size. Given that during early AL iterations rounds, only limited amounts of training data are available, we limit the depth of our trees to 2 to avoid over-fitting. Following standard practices, individual trees are optimized using 40%-60%

of the available training data chosen at random and 10 to 40 features are explored per split. The average radius of supervoxels κ is 4.3 in EM dataset and 5.7 in MRI dataset. We set the number k of nearest neighbours of Section 2.4.3 to be the average number of immediately adjacent supervoxels on average, which is between 7 and 15 depending on the resolution of the image and size of supervoxels. However, experiments showed that the algorithm is not very sensitive to the choice of this parameter. We restrict the size of each planar patch to be small enough to contain typically not more than 2 classes of objects and we explain what happens if this condition is not satisfied. To this end, we take the radius r of Section 2.5.1 to be between 10 and 15, which yields patches such as those depicted by Figure 2.12.

Baselines

For each dataset, we compare our approach against several baselines. The simplest is Random Sampling (**Rand**), which involves randomly selecting samples to be labelled. It can be understood as an indicator of how difficult the learning task is.

We also perform Uncertainty Sampling according to each of the three criteria described in Section 1.3.2. We will refer to *total entropy* uncertainty as **FEnt**, *minmax* strategy as **FMnMx** and *min margin* strategy as **FMnMar**. Notice that **FMnMx** and **FMnMar** cannot be easily combined with the geometric uncertainty because no upper-bound rule is applicable.

Proposed strategies

All entropy-based measures introduced in Sections 2.4.2, 2.4.3 and 2.4.4 can be used in our unified framework. Let H^F be the specific one we use in a given experiment. The strategy then is to select

$$x^* = \arg \max_{x_i \in \mathcal{U}} (H^F(x_i)). \quad (2.14)$$

Recall that we refer to the feature uncertainty **FUn** strategy that relies on standard *total entropy* as **FEnt**. By analogy, we will refer to those that rely on the *selection entropy* and *conditional entropy* of Equations 2.2 and 2.4 as **FEntS** and **FEntC**, respectively. Similarly, when using the combined uncertainty **CUn** of Section 2.4.4, we will distinguish between **CEnt**, **CEntS**, and **CEntC** depending on whether we use *total entropy*, *selection entropy*, or *conditional entropy*. For random walk inference, we set $\tau_{max} = 10$ in the multi-class case and $\tau_{max} = 20$ in the binary-segmentation one.

Any strategy can be applied in a randomly chosen plane, which we will denote by adding **p** to its name, as in **pFEnt**. Finally, we will refer to the plane selection strategy of Section 2.5 in conjunction with either **FUn** or **CUn** as **p*FEnt**, **p*FEntS**, **p*FEntC**,

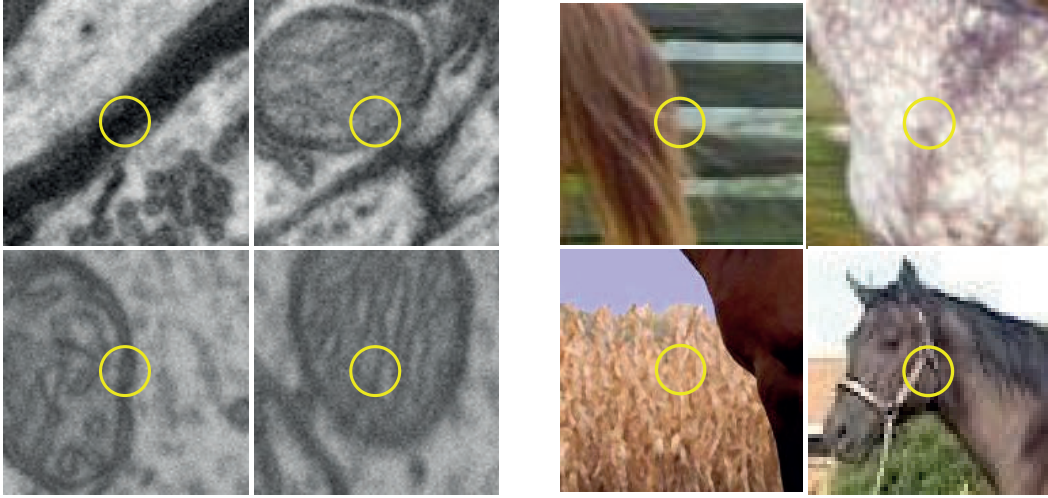


Figure 2.12 – Circular patches to be annotated by the expert highlighted by the yellow circle in Electron Microscopy and natural images. The patches can be annotated either with a line that separates 2 classes or by correcting the mistakes in the current prediction, as shown in Figure 2.2.

$\mathbf{p}^*\mathbf{CEnt}$, $\mathbf{p}^*\mathbf{CEntS}$ and $\mathbf{p}^*\mathbf{CEntC}$, depending on whether uncertainty from \mathbf{FEnt} , \mathbf{FEntS} , \mathbf{FEntC} , \mathbf{CEnt} , \mathbf{CEntS} , or \mathbf{CEntC} is used in the plane optimization. All plane selection strategies use the $m = 5$ best supervoxels in the optimization procedure. Further increasing this value does not yield any significant learning rate improvement.

Figures 2.2, 2.12 jointly depict what a potential user would see for plane selection strategies given a small enough patch radius. Given a well designed interface, it will typically require only a few mouse clicks to provide the required feedback, as depicted by Figure 2.2. The easiest way to annotate patches with only two classes is to indicate a line between them, and in situations when more than two classes co-occur in one patch, we allow users to correct mistakes in the current prediction instead. We will show that it does not require more than three corrections per iteration. For performance evaluation purposes, we therefore estimate that each user intervention for $\mathbf{p}^*\mathbf{FEnt}$, $\mathbf{p}^*\mathbf{CEnt}$, $\mathbf{p}^*\mathbf{FEntS}$, $\mathbf{p}^*\mathbf{CEntS}$, $\mathbf{p}^*\mathbf{FEntC}$, $\mathbf{p}^*\mathbf{CEntC}$ requires either two or three inputs from the user whereas for other strategies it requires only one.

Note that $\mathbf{p}^*\mathbf{FEnt}$ is similar in spirit to the approach of Top et al. [180] and can therefore be taken as a good indicator of how it would perform on our data. However, unlike in work of Top et al. [180], we do not require the user to label the whole plane and retain our proposed interface for a fair comparison.

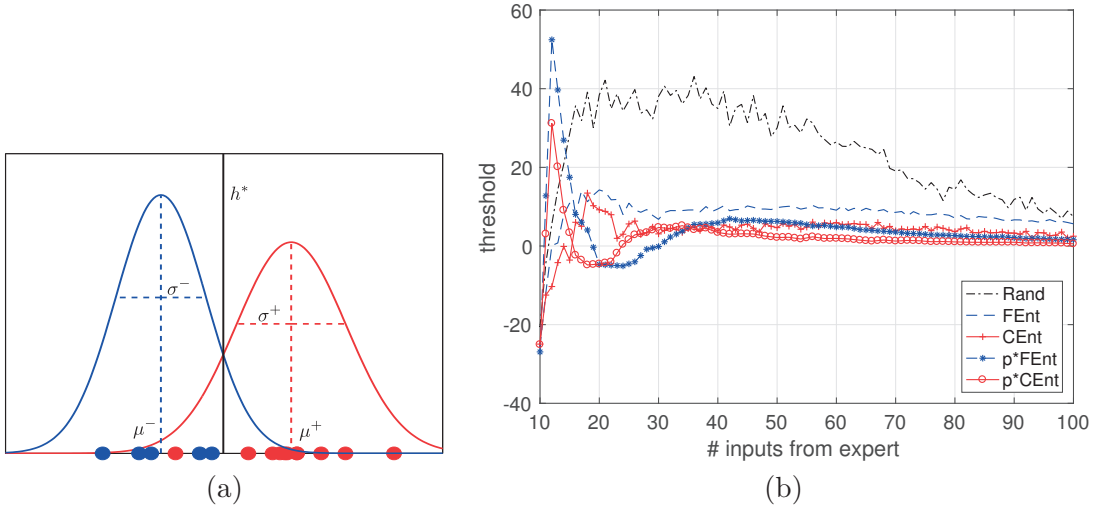


Figure 2.13 – Threshold selection. (a) We estimate mean and standard deviation for classifier scores of positive class datapoints (μ^+ and σ^+ , data is shown in red) and negative class datapoints (μ^- , σ^- , data is shown in blue) and fit 2 Gaussian distributions. Given their pdf, we estimate the optimal Bayesian error with threshold h^* . (b) Adaptive Thresholding convergence rate of classifier threshold for different AL strategies.

Adaptive thresholding for binary AL

The probability of a supervoxel belonging to a certain class from Section 2.4.2 is computed as

$$p_\theta(y_i = y | x_i) = \frac{\exp^{-2 \cdot (F_y - h_y)}}{\sum_{y_j \in Y} \exp^{-2 \cdot (F_{y_j} - h_{y_j})}}, \quad (2.15)$$

where $F = \{F_y | y \in Y\}$ is the classifier output and $h = \{h_y | y \in Y\}$ is the threshold (the multiplier 2 in the exponent is due to the use of exponential loss) [60]. Given enough training data, it can be chosen by cross-validation but this may be misleading or even impossible in an AL context. In practice, we observe that the optimal threshold value varies significantly for binary classification tasks and that the uncertainty measures are sensitive to it. By contrast, in multi-class scenarios, the threshold values remain close to 0 and our proposed entropy-based strategies are comparatively unaffected. In our experiments, we therefore take it to be 0 for multi-class segmentation and compute it as follows in the binary case. We assume that the scores of training samples in each class are Gaussian distributed with unknown parameters μ and σ . We then find an optimal threshold h^* by fitting Gaussian distributions to the scores of positive and negative classes and choosing the value that yields the smallest Bayesian error, as depicted by Figure 2.13(a). We refer to this approach as *Adaptive Thresholding* and we use it for all our experiments. Figure 2.13(b) depicts the value of the selected threshold as the amount

of annotated data increases. Note that our various strategies yield different convergence rates, with the fastest for the plane-based strategies, $\mathbf{p}^*\mathbf{FEnt}$ and $\mathbf{p}^*\mathbf{CEnt}$.

Experimental protocol

In all cases, we start with 5 labelled supervoxels from each class and perform AL iterations until we receive 100 simulated user inputs in the binary case and 200 in the multi-class case. Each method starts with the same random subset of samples and each experiment is repeated $N = 40 - 50$ times. We will therefore plot not only accuracy results but also indicate the variance of these results. We use half of the available data for independent testing and the AL strategy selects new training datapoints from the other half.

In our experiments we have access to fully annotated ground-truth volumes and we use them to simulate the expert’s intervention in our experiments. This ground truth allows us to model several hypothetical behaviours of human expert. We detail the specific features we used for EM, MRI, and natural images below.

2.6.2 Multi-class classification with AL

Recall from Section 2.4.1 that in multi-class scenarios, the different approaches to measuring \mathbf{FUn} yield different selection strategies, as shown in Figure 2.4. Therefore, even though some of these strategies derive from the similar intuition, they favour different points. For example, \mathbf{FMnMar} selects samples with small margin between the most probable classes irrespectively of the absolute values of the probabilities, whereas \mathbf{FEntC} allows for bigger margins for higher values. Selection entropy \mathbf{FEntS} tends to avoid samples that look like they can belong to any of the existing classes. This property can be useful to avoid querying outliers that look equally unlikely to belong to any class.

To study these differences independently of a full image segmentation pipeline, we first test the various strategies in a simple multi-class image classification task. We consider the three datasets depicted by Figure 2.14. *Digits* is a standard MNIST collection with 10 hand-written digits and we use raw pixel values as features. *Chinese* comprises 3 classes from the a dataset of of Chinese handwriting characters [112]. *Butterflies* dataset contains 5 classes from British butterfly images from a museum collection [78]. In the *Chinese* and *Butterflies* datasets[78] features are extracted using a ConvNet of Jia et al. [75].

We use a logistic regression classifier and test our various AL multi-class strategies including Expected Model Change (\mathbf{EMC}) [164, 176, 182, 82]. The results are shown in Figure 2.15. The strategies based on the *selection* and *conditional entropy* perform either better or at least as well as the strategies based on the standard measures of uncertainty. The performance of \mathbf{EMC} approach is not consistent and does not justify



Figure 2.14 – Sample images from the image-classification datasets *Chinese*, *Butterflies* and *Digits*.

a high computational cost: 45 and 310 seconds per iteration in *Chinese* and *Butterfly* datasets with 4096 samples with 359 and 750 features correspondingly, against 0.005 and 0.01 seconds by *conditional entropy*. The **EMC** execution time grows with the AL pool size and thus, we did not run experiments with more than 10 000 samples.

Many strategies exhibit a noticeable performance drop after a few iterations of AL. The reason for this is the imbalance of the class proportions in the training set. AL starts with a small balanced datasets. Then, it is highly likely that the first annotation iterations will make the training set unbalanced. With little data, the negative influence of the class imbalance on the classifier is bigger than the advantage of adding more data. This effect is quickly eliminated when more data is collected, thus we do not concentrate on it much in the further experiments.

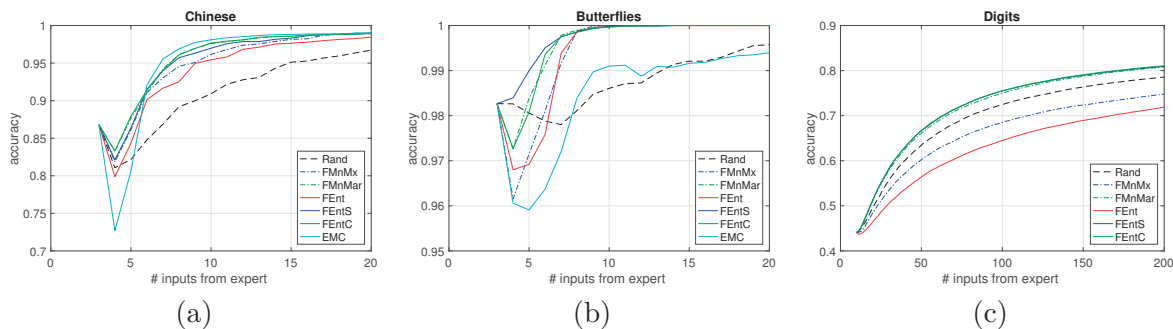


Figure 2.15 – Multi-class AL strategies applied to image classification tasks. Logistic regression is used as an underlying classifier. We compare standard multi-class AL criteria against the newly introduced entropy-based criteria on (a) *Chinese* dataset, (b) *Butterflies* dataset, and (c) *Digits* dataset.

2.6.3 Segmentation of volumetric data with AL

Results on EM data

First, we work with two 3D EM stacks of rat neural tissue, one from the *striatum* and the other from the *hippocampus* [118]. One stack of size $318 \times 711 \times 422$ ($165 \times 1024 \times 653$ for the hippocampus) is used for training and another stack of size $318 \times 711 \times 450$ ($165 \times 1024 \times 883$) is used to evaluate the performance. Their resolution is 5nm in all three spatial orientations. The slices of Figure 2.1 as well as patches in Figure 2.12(a) come from the striatum dataset. The hippocampus volume is shown in Figure 2.16. Since the image stacks have the same resolution in all dimensions, they can be viewed equally well in all orientations and specialized tools have been developed for such a use by neuroscientists [145].

The task is to segment mitochondria, which are the intracellular structures that supply the cell with its energy and are of great interest to neuroscientists. An example of one slice from hippocampus dataset with its ground truth is shown in Figure 2.17. It is extremely laborious to annotate sufficient amounts of training data for learning segmentation algorithms to work satisfactorily. Furthermore, different brain areas have different characteristics, which means that the annotation process must be repeated often. The features we feed to the Boosted Trees rely on local texture and shape information using ray descriptors and intensity histograms as in work of Lucchi et al. [120].

In Figure 2.18, we plot the performance of all the approaches in terms of the intersection over union (IoU) score [49] as a function of the annotation effort. The horizontal line at the top depicts the IoU scores obtained by using the whole training set, which comprises 276 130 and 325 880 supervoxels for the striatum and the hippocampus, respectively. **FEnt** provides a boost over **Rand** and **CEnt** yields a larger one. Any strategy can be

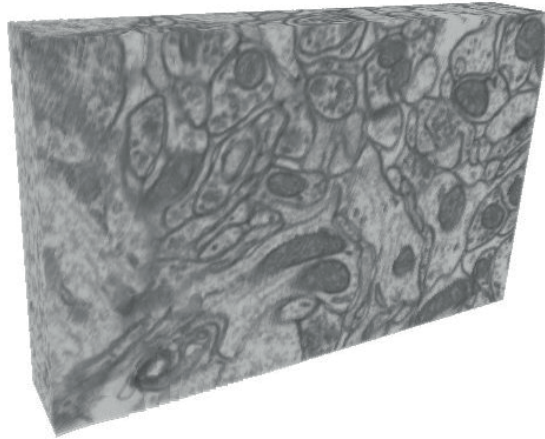


Figure 2.16 – Hippocampus image volume for mitochondria segmentation

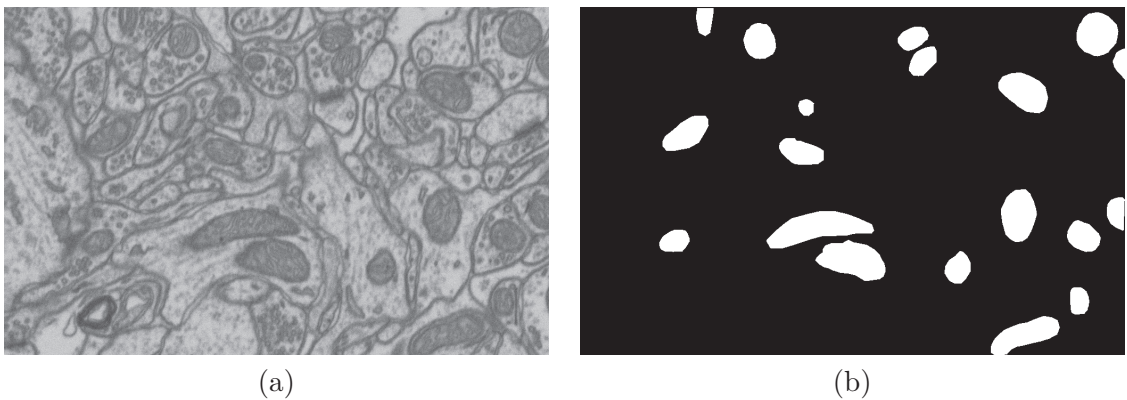


Figure 2.17 – An example of a slice (a) and its ground truth annotation (b) from *Hippocampus dataset*.

combined with a batch-mode AL that is done by selecting a 2D plane to be annotated. For example, strategies **pRand**, **pFEnt** and **pCEnt** present to the user a randomly selected 2D plane around the sample selected by **Rand**, **FEnt** and **CEnt**. Addition of a plane boosts the performance of all corresponding strategies, but further improvement is obtained by introducing the batch-mode geometry query selection with an optimal plane search by Branch-and-Bound algorithm in strategies **p*FEnt** and **p*CEnt**. The final strategy **p*CEnt** outperforms all the rest of the strategies thanks to the synergy of geometry-inspired uncertainty criteria and the selection of a batch.

Recall that these numbers are averaged over many AL episodes. In Table 2.1, we give the corresponding variances. Note that both using the **CUn** and the batch-mode with optimal plane selection tend to reduce variances, thus making the process more predictable.

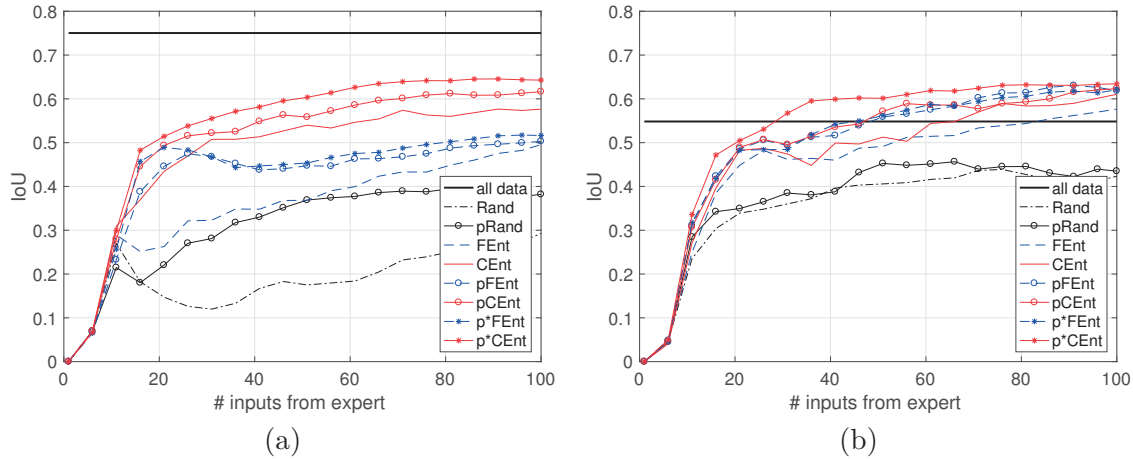


Figure 2.18 – Comparison of various AL strategies for (binary) mitochondria segmentation on (a) Striatum dataset, (b) hippocampus dataset.

Table 2.1 – Variability of results (in the metric corresponding to the task) by different binary AL strategies. 80% of the scores are lying within the indicated interval. **FUn** is more variable than **CUn**, batch selection is less variable than single-instance selection and the batch-selection with an optimal plane cut combined with geometry-inspired uncertainty is the least variable. The best result is highlighted in bold.

Dataset	FEnt	CEnt	pFEnt	pCEnt	p*FEnt	p*CEnt
Striatum	0.133	0.105	0.121	0.094	0.115	0.086
Hippoc.	0.117	0.101	0.081	0.092	0.090	0.078
MRI	0.076	0.064	0.078	0.074	0.073	0.048
Natural	0.145	0.140	0.149	0.124	—	—

Somewhat surprisingly, in the hippocampus case, the classifier performance given only 100 training data points is *higher* than the one obtained by using *all* the training data. In fact, this phenomenon has been reported in the AL literature [161] and suggests that in some cases a well chosen subset of datapoints can produce better generalisation performance than the complete set.

Note that the 100 samples we use are two orders of magnitude smaller than the total number of available samples. Nevertheless AL provides a segmentation of comparable quality. As an example of qualitative results, Figure 2.19 shows the segmentation obtained by the model trained with 1000 samples selected by **Rand** and the model trained with 100 samples selected by **CEnt**. We see that with 10 times less annotations the geometry-aware strategy results in the same or even better (on the boundaries) quality than the random strategy.

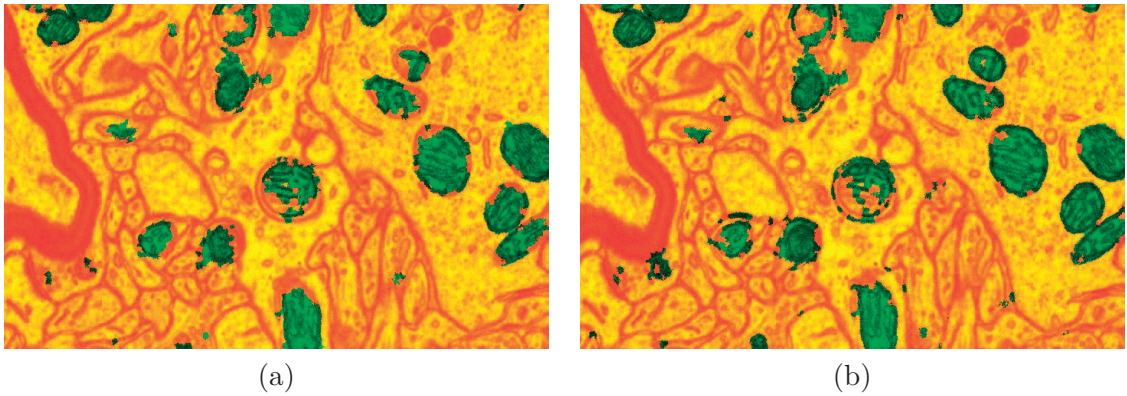


Figure 2.19 – A qualitative result on *Striatum dataset*. (a) Segmantation result with 1000 randomly sampled datapoints. (b) Segmentation result with 100 actively sampled datapoints according to strategy **CEnt**.

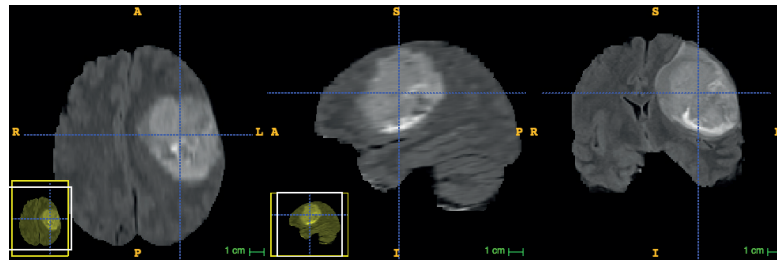


Figure 2.20 – An example from MRI dataset for tumour segmentation (Flair image).

Results on MRI data

In this section, we consider multi-modal brain-tumour segmentation in MRI brain scans (Figure 2.20). Segmentation quality critically depends on the amount of training data and only highly-trained experts can provide it. T1, T2, FLAIR, and post-Gadolinium T1 MR images are available in the BRATS dataset for each of 20 subjects [126]. We use standard filters such as Gaussian, gradient filter, tensor, Laplacian of Gaussian and Hessian with different parameters to compute the feature vectors we feed to the Boosted Trees.

Foreground-background segmentation We first consider segmentation of tumour versus healthy tissue. In Figure 2.21(a) we plot the performance of all the approaches in terms of the dice score [57] (a commonly used quality measure for brain tumour segmentation), as a function of the annotation effort and in Table 2.1, we indicate the corresponding variances. We observe the same pattern as in Figure 2.18, with **p*CEnt** again resulting in the highest score. Note that difference between **p*CEnt** and **pCEnt** is greater than between **p*FEnt** and **pFEnt** in all the experiments. This is the evidence

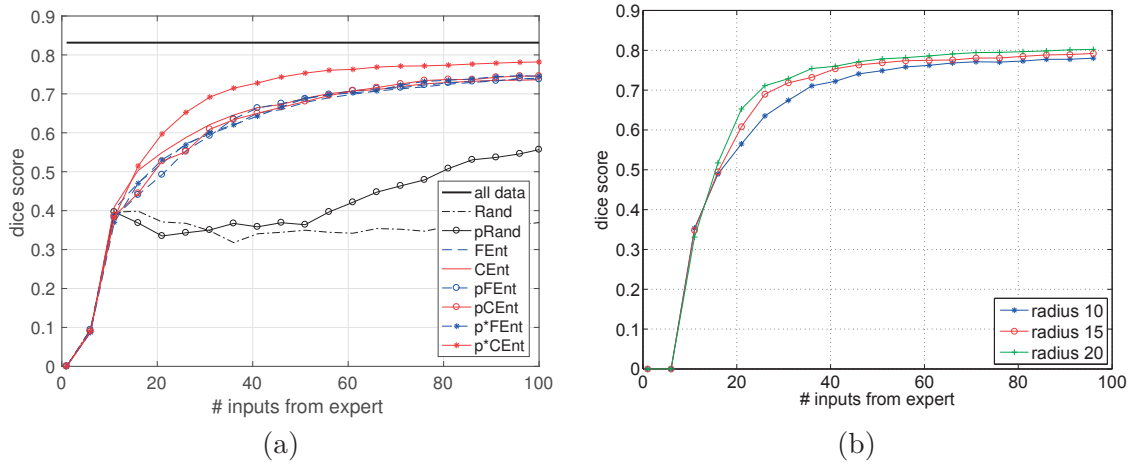


Figure 2.21 – Comparison of various AL strategies for MRI data for binary tumour segmentation. (a) Dice score for BRATS2012 dataset, (b) $\mathbf{p}^*\mathbf{CEnt}$ strategy with patches of different radius.

of the synergy brought by the *geometric* uncertainty and the batch selection based on the *geometry*.

The patch radius parameter r of Section 2.5.1 plays an important role in plane selection procedure. To evaluate its influence, we recompute the $\mathbf{p}^*\mathbf{CEnt}$ results 50 times using three different values for $r = 10, 15$ and 20 . The resulting plot is shown in Figure 2.21(b). As expected, with a larger radius, the learning curve is slightly higher since more voxels are labelled each time. However, as the patches become larger, it stops being clear that labelling can be done with small user effort and that is why we limit ourselves to radius sizes of 10 to 15.

Multi-class segmentation We test the multi-class approach on the full label set of the BRATS competition: healthy tissue (label 1), necrotic center (2), edema (3), non-enhancing gross abnormalities (4), and enhancing tumour core (5). Figure 2.22 shows a ground truth example for one of the volumes, where different classes are indicated in different colors.

Note that the ground truth is highly unbalanced: we have 4000 samples of healthy tissue, 1600 of edema, 750 of enhancing tumour core, 250 of necrotic center and 200 of non-enhancing gross abnormalities in the full training dataset. We use the protocol of the BRATS competition [126] to analyse the results. This involves evaluating how well we segment complete tumours (classes 2, 3, 4, and 5), core tumours (classes 2, 4, and 5), and enhancing tumours (class 5 only).

Figure 2.24 depicts the results of the proposed methods and the selected baselines on these three tasks. As before, the results clearly indicate that active selection provides

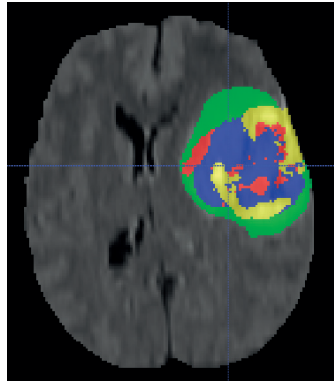


Figure 2.22 – Example of ground truth from multi-class brain-tumor segmentation. Necrotic center in red, edema in green, non-enhancing gross abnormalities in blue and enhancing tumor core in yellow. Best seen in color.

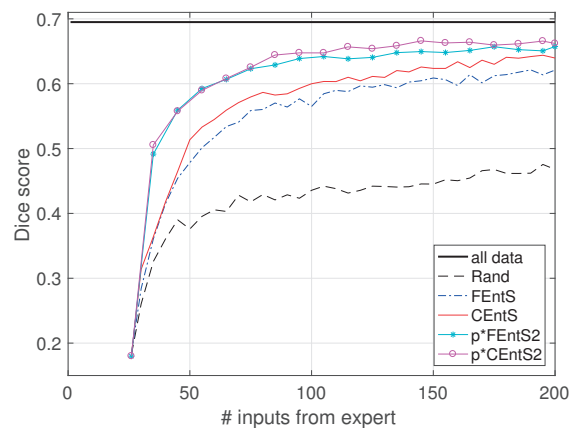


Figure 2.23 – Dice score for enhancing tumour segmentation. Performance of various strategies that have Selection Entropy at their basis.

a significant improvement over passive selection. Here we do not show all the variants of batch-mode query selection for the benefit of the figure clarity. Among the basic strategies, **FMnMar** gives the best performance in subtasks 1 and 2 and **FMnMx** in subtask 3. The proposed entropy-based uncertainty strategies **FEntS** and **FEntC** perform better or equivalent to the corresponding baselines **FMnMx** and **FMnMar** as in the task of image classification. Next, the **CUn** strategies **CEnt**, **CEntS** and **CEntC** outperform their corresponding **FUn** versions **FEnt**, **FEntS** and **FEntC**, where the improvement depends on the subtask and the strategy. Note that **FEntS** and **FEntC** as well as **CEntS** and **CEntC** perform equally well, thus, they can be used interchangeably. Further improvement is obtained when each of the strategies is combined with the optimal plane selection and we omit the random plane selection for the clarity of the figure.

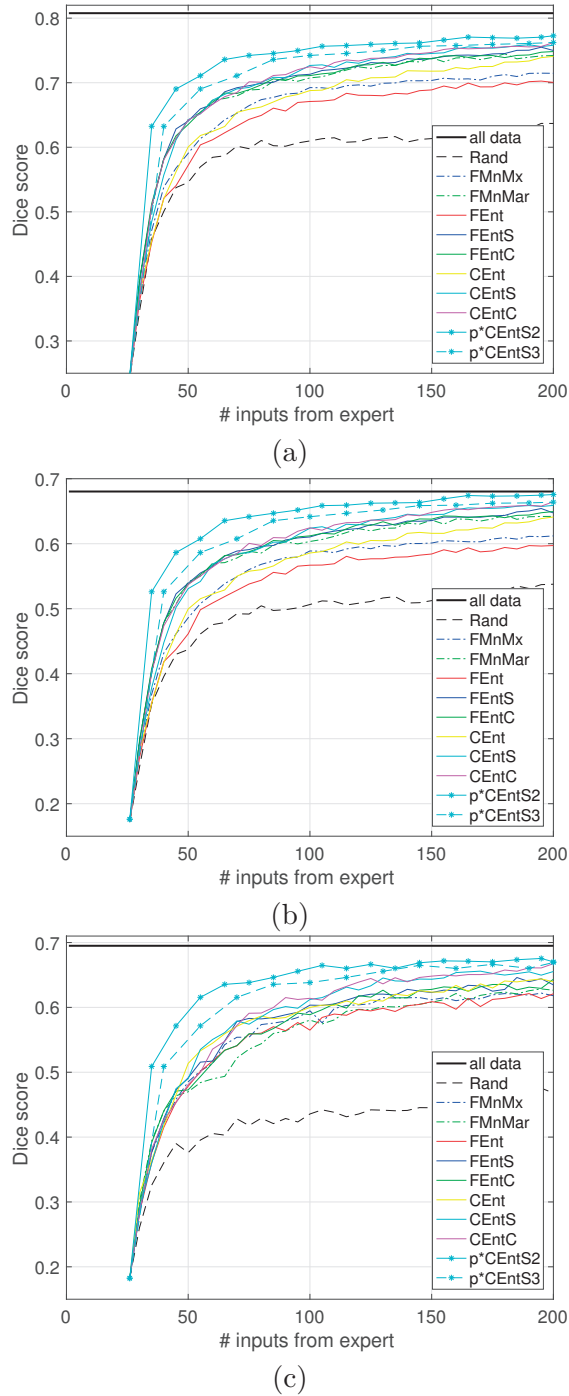


Figure 2.24 – Comparison of different AL strategies for multi-class MRI segmentation. Dice scores for three BRATS2012 tasks: (a) complete tumour, (b) tumour core, (c) enhancing tumour.

In this experiment we observe that around 43% of selected patches contain more than two classes. In such cases, simply finding a line separating two classes is not enough. To

handle such cases, we propose a different annotation scheme. The current prediction on supervoxels is displayed to the annotator who needs to correct the mistakes in the prediction. We count the number of misclassified samples throughout the experiments and on average there were no more than 10.42% errors in the supervoxel classes in all iterations, that is approximately 2.42 samples per iteration. Thus, we show the learning curves for the plane-based strategy and we count one annotation iteration as either two and or three inputs from the user, with both variants dominating simple strategies.

The difference between competing **CUn** strategies becomes negligible with a slight dominance of Selection Entropy **p*CEntS** in subtasks 1 and 2 and Total entropy **p*CEnt** in the last subtask. In seven of nine cases, the **CUn** in conjunction with the plane selection yields better results than **FUn** with plane selection and in two of nine, they perform equally well. For illustrative purposes, Figure 2.24 contains only the best performing learning curve of **p*CEntS** and Figure 2.23 shows the performance of all strategies based on the Selection Entropy in the third subtask.

2.6.4 Segmentation of natural images with AL

Finally, we turn to natural 2D images and replace supervoxels by superpixels. In this case, the plane selection reduces to a simple selection of patches in the image and we will refer to these strategies as **pFEnt** and **pCEnt** because they do not involve the branch-and-bound search for an optimal plane. In practice, we simply select superpixels with their 4 neighbours in binary segmentation and 7 in multi-class segmentation. Increasing this number would lead to higher learning rates in the same way as increasing the patch radius r , but we restrict it to a small value to ensure labelling can be done with two mouse clicks on average.

Foreground-background segmentation of horses We study the results of binary AL on the Weizmann horse database [22]. An example of an image with its ground truth is depicted in Figure 2.25. The results are depicted in Figure 2.26 and the corresponding variances are presented in Table 2.1. To compute image features, we use Gaussian, Laplacian, Laplacian of Gaussian, Prewitt and Sobel filters for intensity and color values, gather first-order statistics such as local standard deviation, local range, gradient magnitude and direction histograms, as well as SIFT features. The pattern is again similar to the one observed in Figures 2.18 and 2.21, with the difference between **CEnt** and **pCEnt** being smaller due to the fact that 2D batch-mode approach does not involve any optimization of patch selection. Note, however, that while the first few iterations result in the reduced scores for all methods, plane-based methods are able to recover from this effect quite fast.

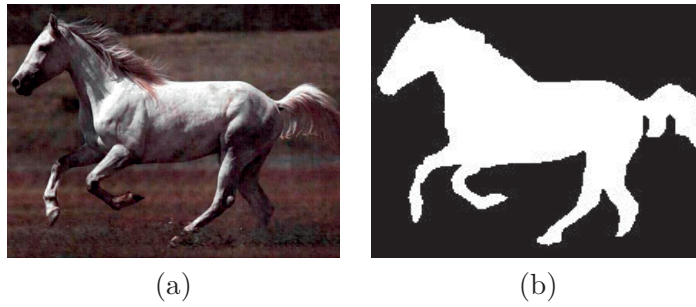


Figure 2.25 – An example of (a) an image and (b) ground truth annotation from *Weizmann horse dataset*.

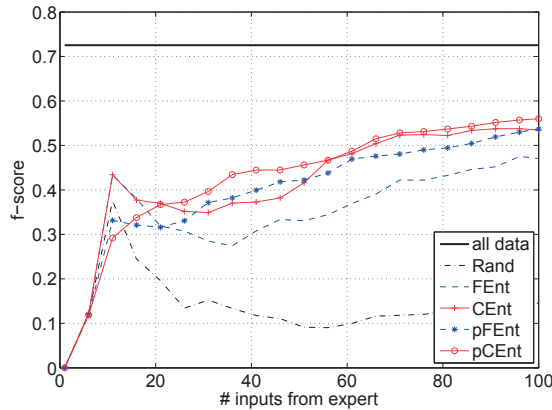


Figure 2.26 – Comparison of various AL strategies for binary segmentation of natural images of horses.

Multi-class segmentation of faces We apply the multi-class AL to the task of segmenting human faces [86]. We distinguish 6 classes: background, hair, skin, eyes, nose, and mouth. Figure 2.27 demonstrates an example of an image from the dataset with the corresponding ground truth annotation. Notice again that we deal with unbalanced problem, obviously classes ‘eyes’, ‘nose’, ‘mouth’ are a minority compared to ‘background’, ‘skin’ and ‘hair’. We use the same features as for the Weizmann horse database plus HOG features.

As in the case of multi-class MRI images, we must handle cases in which more than two classes are present in a single patch. However, this only happens in 0.84% of the selected patches because three classes do not often co-occur in the same small neighbourhood. Thus, we can still use the simple line separation heuristic depicted by Figure 2.2 in most iterations and leave a user an opportunity to use a standard brush for rare refined annotations.

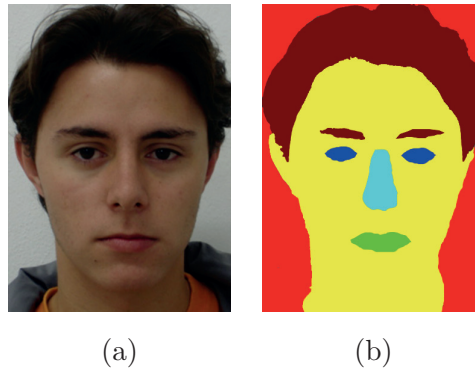


Figure 2.27 – Dataset for face segmentation (a) Example of an image from face segmentation dataset (b) Ground truth annotation for the given image. Different classes are indicated in different colors. Best viewed in color.

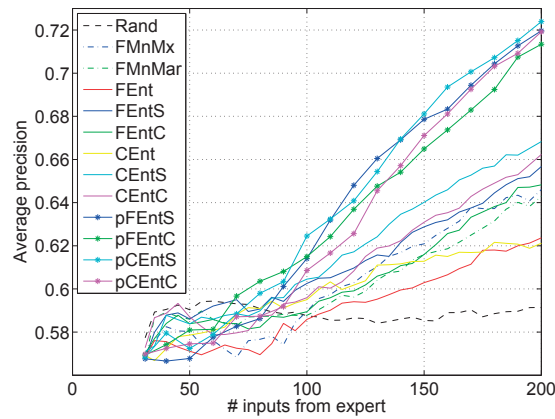


Figure 2.28 – Comparing several AL strategies for multi-class face segmentation.

In Figure 2.28 we compare our results to those of the baselines in terms of precision averaged over each one of the 6 classes. This measure was chosen because it is better suited for capturing the performance in smaller classes and, thus reflects better the performance in segmentation with unbalanced classes. At the same time we monitor the score of total precision (but omit the figure for conciseness), that performs in similar way for all AL strategies. This is done to ensure that performance on dominant classes is not sacrificed. Entropy-based algorithms **FEntS** and **FEntC** are better than the standard **FMnMx** and **FMnMar**, respectively. Moreover, selection that is based on the entropy allows for a combination with **CUn** and brings further improvement in average precision with the strategies **CEnt**, **CEntS** and **CEntC**. Next, each of the strategies can be used in conjunction with patch selection that allows for further growth of the learning rate. We show the patch selection results only for Selection Entropy and Conditional Entropy and skip Total Entropy as it performs poorly in total precision. As we can see, the

combination of plane selection with **CUn** strategies demonstrates better results at the end of learning experiments with the best result obtained by **pCEntS**.

2.6.5 Active learning or human intuition

Before we conclude the experiments, we would like to motivate why AL is important and why we cannot rely on human selecting the data for annotation manually. First advantage of AL is that it eliminates the cognitive cost for a human user who would otherwise need to decide which datapoint is informative for a classifier. For this, the human annotator would need to have a good understanding of the underlying classification algorithm. Besides, in the next experiment we show an example that demonstrates that not all human-intuitive strategies are useful for a classifier.

To design a human-intuitive selection strategy, we study distances to the closest class boundary for selected samples. For this purpose we count how many samples lie within radius of 10 pixels from the boundary for 2 strategies: **Rand** and **CEntS** in the face dataset. We observe that **CEntS** strategy samples 7.4% more datapoints in this area than **Rand**. More superpixels in this area illustrate the effect of geometric component that prefers regions in the non-smooth areas of the prediction. Then, an intuitive strategy could be to first label patches at the boundary between classes. We implemented the selection strategies that simulate such user behaviours and we refer to it as *boundary* strategy.

To design another human-intuitive strategy, we notice that as part of the AL query selection procedure, we predict the segmentation for the whole training volume at every iteration. Given this prediction, a human expert could manually identify patches that are worth labelling. For example, he might first correct the most obvious mistakes. Thus, we simulate such a strategy *max error* by selecting first the most confidently but wrongly classified samples.

We ran fifty trials using each of these two strategies on the face segmentation problem. Fig. 2.29 depicts the results. Surprisingly, the human strategies perform much worse than even passive data selection, that confirms the difficulty of the AL problem. The heuristics we proposed derive from our intuitive understanding of the problem. However, applying these intuitions is not straightforward for a human user. For example, it turns out that selecting samples that have the highest error leads to selecting outlier samples or those that have the most contradictory appearance. Thus, intelligent and automated query selection is necessary to determine how uncertain the classifier is and what smoothness prior should be used when selecting the next samples to be labelled.

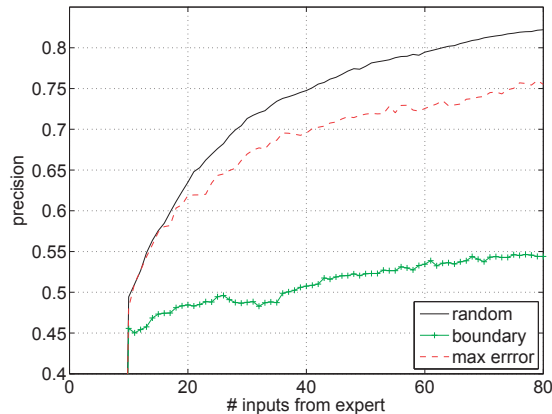


Figure 2.29 – Hypothetical human expert selection strategies. We demonstrate that strategies that are intuitive for a human annotator do not result in better performance than passive sampling.

2.7 Conclusion

In this chapter we introduce an approach to exploit image geometry priors to increase the effectiveness of AL in image segmentation application. We propose entropy-based uncertainty measures for multi-class classification that can be combined with geometric priors in a principled way. In the segmentation of 2D and 3D images, our approach leverages the uncertainty information on the prediction at an image patch and at its neighbours. For 3D image stacks, it adds an ability to select a 2D planar patch where annotations are easier to perform. We demonstrate the effectiveness of our approach on several datasets featuring MRI, EM, and natural images and both for foreground-background and multi-class segmentation.

We conclude that intuitions about geometrical properties of images are useful to answer the question “*What to annotate?*” in segmentation. Besides, by reducing the annotation task from cumbersome 3D annotations to 2D annotations, we provide one possible answer to the question “*How to annotate?*”. Moreover, we observe that addressing these two questions jointly can bring additional benefits to the annotation method. Finally, we notice that the human intuitions may not always result in a desirable behaviour.

Our hand-crafted annotation strategy brings us impressive cost savings in image segmentation. Yet, we realise that it would be impossible to design a selection strategy for every new problem at hand. Besides, our last experiment shows that not all human intuitions perform as expected. To overcome these limitation and systematically search in the space of possible strategies, in Chapters 3 and 4 we go beyond manually-designed AL strategies towards strategies learnt from data. In the same spirit, we move towards data-driven techniques in answering “*How to annotate?*” when we design a method to annotate data for object detection in Chapter 5.

3 Learning Active Learning from Real and Synthetic Data

3.1 Introduction

Over the years many AL strategies have been developed for various classification tasks, without any one of them clearly outperforming others in all cases. Consequently, a number of meta-AL approaches have been proposed to automatically select the best strategy. The examples include bandit algorithms [12, 62, 35] and reinforcement learning approaches [47]. A common limitation of many meta-AL methods is that they cannot go beyond combining pre-existing hand-designed heuristics. Besides, they require reliable assessment of the classification performance which is problematic because the annotated data is scarce. In this chapter¹ we introduce a method that overcomes these limitations thanks to its two features. First, we look at a whole continuum of AL strategies instead of combinations of pre-specified heuristics. Second, we bypass the need to evaluate the classification quality from application-specific data because we rely on experience from previous tasks and can seamlessly transfer strategies to new domains.

More specifically, we formulate Learning Active Learning (LAL) as a regression problem. Given a trained classifier and its output for a specific sample without a label, we predict the reduction in generalization error that can be expected by adding the label to that datapoint. We then ask an expert to annotate the datapoint that is expected to yield the largest performance increase. In practice, we show that we can train this regression function on synthetic data by using simple features, such as the variance of the classifier output or the predicted probability distribution over possible labels for a specific datapoint. The features for the regression are not domain-specific and this enables to apply the regressor trained on synthetic data directly to other classification problems. Furthermore, if a sufficiently large annotated set can be provided initially, the regressor can be trained on it instead of on synthetic data. The resulting AL strategy is then tailored to the particular problem at hand. We show that LAL works well on real data from several different domains such as biomedical imaging, economics, molecular biology and high

¹This chapter is based on Konyushkova et al. [90]

energy physics. This query selection strategy outperforms competing methods without requiring hand-crafted heuristics and at a comparatively low computational cost.

3.2 Related work

Among many proposed AL methods, uncertainty sampling (US) from Section 1.3.2 is both simple and computationally efficient. This makes it one of the most popular strategies in real applications. In short, it suggests labelling samples that are the most uncertain, i.e., closest to the classifier’s decision boundary. The above methods work very well in cases such as the ones depicted in Figure 3.1, but often fail in the more difficult ones depicted in Figure 3.2.

Among many AL methods introduced in the past, there is no one algorithm that consistently outperforms all others in all applications [165, 12, 47]. At the time of this work, meta-learning algorithms have been gaining in popularity [178, 157], but few of them tackle the problem of learning AL strategies [12, 62, 35, 47]. In Section 1.3.2 we discuss the development of meta-AL moving from combining several manually-designed strategies towards learning a strategy from data. However, at the time of this work, only strategies from the first group existed. Recall that they are limited because a) they are restricted to combining already existing techniques and b) their success depends on the ability to estimate the classification performance from scarce annotated data. The pioneering work of our data-driven approach LAL helps to overcome these limitations. Section 3.5 shows that it outperforms several baselines including those of Hsu and Lin [62] and Kapoor et al. [84].

3.3 Towards data-driven active learning

In this section we motivate why a data-driven approach can improve AL strategies and how it can deal with the situations where US fails. We select US as a representative method because it is popular and widely applicable, however the behaviour that we describe is typical for a wide range of AL strategies.

3.3.1 Success, failure, and motivation

We now motivate the need for LAL by presenting two toy examples. In the first one, US is empirically observed to be the best greedy approach, but in the second it makes suboptimal decisions. Let us consider simple two-dimensional datasets \mathcal{Z} and \mathcal{Z}' drawn from the same distribution with an equal number of points in each class (Figure 3.1a). The data in each class comes from a Gaussian distribution with a different mean and the same isotropic covariance. We can initialize the AL procedure with one sample from each class and its respective label: $\mathcal{L}_0 = \{(x_1, 0), (x_2, 1)\} \subset \mathcal{Z}$ and $\mathcal{U}_0 = \mathcal{Z} \setminus \mathcal{L}_0$. Here

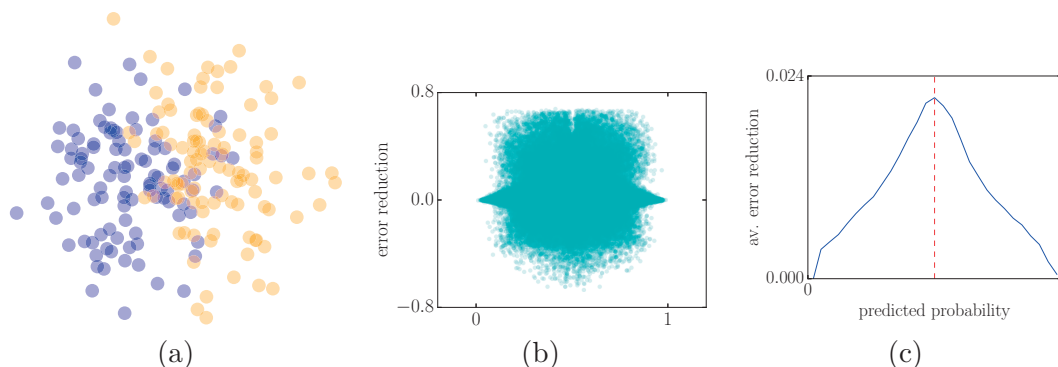


Figure 3.1 – (a) Two Gaussian clouds of the same size. (b) The test error reduction as a function of predicted probability of class 0, every point corresponds to one experimnt. (c) The test error reduction as a function of predicted probability of class 0 average over all experimnts.

we train a simple logistic regression classifier f on \mathcal{L}_0 and then test it on \mathcal{Z}' . If $|\mathcal{Z}'|$ is large, the test error can be considered as a good approximation of the generalization error: $\ell_0 = \sum_{(x',y') \in \mathcal{Z}'} \ell(\hat{y}, y')$, where $\hat{y} = f_0(x')$.

Let us try to label every point x from \mathcal{U}_0 one by one, form a new labelled set $\mathcal{L}_x = \mathcal{L}_0 \cup (x, y)$ and check what error a new classifier f_x yields on \mathcal{Z}' , that is, $\ell_x = \sum_{(x',y') \in \mathcal{Z}'} \ell(\hat{y}, y')$, where $\hat{y} = f_x(x')$. The difference between errors obtained with classifiers constructed on \mathcal{L}_0 and \mathcal{L}_x indicates how much the addition of a new datapoint x reduces the generalization error: $\delta_x = \ell_0 - \ell_x$. In Figure 3.1(b) we plot δ_x for the 0/1 loss function for each of 10 000 experiments as a function of the predicted probability p_0 and Figure 3.1(c) shows the averaged values. By design, US would select a datapoint with probability of class 0 close to 0.5. We observe that in this experiment, the datasample with p_0 closest to 0.5 is indeed the one that yields the greatest error reduction.

In the next experiment, the class 0 contains twice as many datapoints as the other class, see Figure 3.1(a). As before, we plot the average error reduction as a function of p_0 in Figure 3.1(c). We observe this time that the value of p_0 that corresponds to the largest expected error reduction is different from 0.5 and thus the choice of US becomes suboptimal. Also, the reduction in error is no longer symmetric for the two classes. The more imbalanced the two classes are, the further from the optimum the choice made by US is. In a complex realistic scenario, there are many other factors such as label noise, outliers and shape of distribution that further compound the problem.

Although query selection procedures can take into account statistical properties of the datasets and classifier, there is no simple way to foresee the influence of all possible factors. Thus, in this chapter, we suggest Learning Active Learning (LAL). It uses properties of classifiers and data to predict the potential error reduction. We tackle the query selection

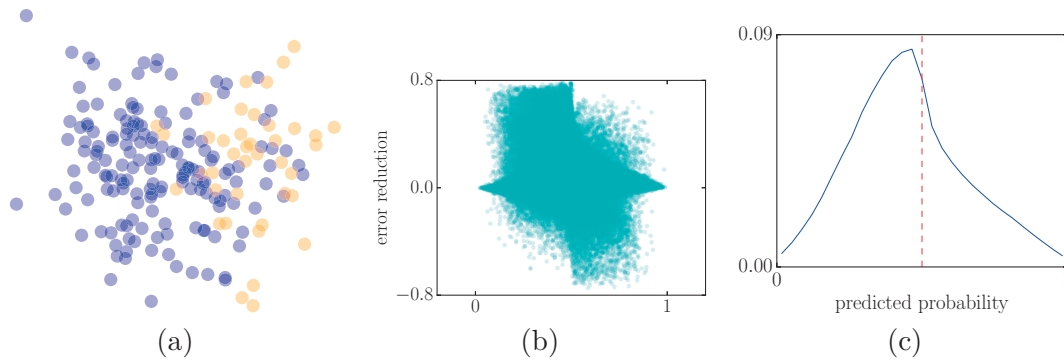


Figure 3.2 – (a) Two Gaussian clouds with the class 0 twice bigger than class 1. (b) The test error reduction as a function of predicted probability of class 0, every point corresponds to one experiment. (c) The test error reduction as a function of predicted probability of class 0 average over all experiments.

problem by using a regression model; this perspective enables us to construct new AL strategies in a flexible way. For instance, in the example of Figure 3.2 we expect LAL to learn a model that automatically adapts its selection to the relative prevalence of the two classes without having to explicitly state such a rule. Moreover, having learnt the error reduction prediction function, we can seamlessly transfer LAL strategy to other domains with very little annotated data. We will see in the results section that this is indeed one of the many things that LAL can handle, even in much more complex real-world cases.

The underlying assumption of LAL idea is the following. We assume that there exists a smooth function in the space of meta parameters that maps them into the potential generalisation performance of the updated classifier. Although the datasets for learning and datasets of interest do not share the same distribution in feature space, they share the distribution in meta parameters, that makes the learnt AL strategy transferable between datasets of different nature. To meet this requirement, we make the datasets for learning to cover different parts of meta parameter space and then we learn an AL strategy from a distribution of datasets.

Another scenario where learning a selection strategy from data is promising is *warm start*. It is largely overlooked in the literature but has a significant practical interest. In order to understand if a learning-based approach is applicable, some sufficient dataset \mathcal{D}_0 is made available prior to AL, but further improvement in classification can be annotation costly. In this situation we can benefit from the available training data \mathcal{D}_0 to learn a specialized LAL strategy for an application.

3.4 Monte-Carlo LAL

Our approach to AL is data-driven and can be formulated as a regression problem. Given a *representative* dataset with ground truth, we simulate an on-line learning procedure using a Monte-Carlo technique. We propose two versions of AL strategies that differ in the way how datasets for learning a regressor are constructed. When building the first one, LALINDEPENDENT, we incorporate unused labels individually and at random to retrain the classifier. Our goal is to correlate the change in test performance with the properties of the classifier and of newly added datapoint. To build the LALITERATIVE strategy, we further extend our method by a sequential procedure to account for selection bias caused by AL. We formalize our LAL procedures in the remainder of the section.

3.4.1 Independent LAL

Let the *representative* dataset² consist of a training set \mathcal{D} and a testing set \mathcal{D}' . Let f be a classifier with a given training procedure. We start collecting data for the regressor by splitting \mathcal{D} into a labelled set \mathcal{L}_τ of size τ and an unlabelled set \mathcal{U}_τ containing the remaining points (Algorithm 1 DATAMONTECARLO). We then train a classifier f on \mathcal{L}_τ , resulting in a function f_τ that we use to predict class labels for elements x' from the test set \mathcal{D}' and estimate the test classification loss ℓ_τ . We characterize the classifier state by K parameters $\phi_\tau = \{\phi_\tau^1, \dots, \phi_\tau^K\}$, which are specific to the particular classifier type and are sensitive to the change in the training set while being relatively invariant to the stochasticity of the optimization procedure. For example, they can be the parameters of the kernel function if f is kernel-based, the average depths of the trees if f is a tree-based method, or prediction variability if f is an ensemble classifier. The above steps are summarized in lines 3–5 of Algorithm 1.

Next, we randomly select a new datapoint x from \mathcal{U}_τ which is characterized by R parameters $\psi_x = \{\psi_x^1, \dots, \psi_x^R\}$. For example, they can include the predicted probability to belong to class y , the distance to the closest point in the dataset or the distance to the closest labelled point, but they do *not* include the features of x . We form a new labelled set $\mathcal{L}_x = \mathcal{L}_\tau \cup \{x\}$ and retrain f (lines 7–13 of Algorithm 1). The new classifier f_x results in the test-set loss ℓ_x . Finally, we record the difference between previous and new loss $\delta_x = \ell_\tau - \ell_x$ which is associated to the learning state in which it was received. The learning state is characterized by a vector $\xi_\tau^x = [\phi_\tau^1 \ \dots \ \phi_\tau^K \ \psi_x^1 \ \dots \ \psi_x^R] \in \mathbb{R}^{K+R}$, whose elements depend both on the state of the current classifier f_τ and on the datapoint x .

²The representative dataset is an annotated dataset that does not need to come from the domain of interest. In Section 3.5 we show that a simple synthetic dataset is sufficient for learning strategies that can be applied to various real tasks across various domains.

Algorithm 1 DATAMONTECARLO

- 1: **Input:** training set \mathcal{D} and test set \mathcal{D}' , classification procedure f , partitioning function SPLIT, size τ
 - 2: **Initialize:** $\mathcal{L}_\tau, \mathcal{U}_\tau \leftarrow \text{SPLIT}(\mathcal{D}, \tau)$
 - 3: train a classifier f_τ
 - 4: estimate the test set loss ℓ_τ
 - 5: compute the classification state parameters $\phi \leftarrow \{\phi_\tau^1, \dots, \phi_\tau^K\}$
 - 6: **for** $m = 1$ **to** M **do**
 - 7: select $x \in \mathcal{U}_\tau$ at random
 - 8: form a new labeled dataset $\mathcal{L}_x \leftarrow \mathcal{L}_\tau \cup \{x\}$
 - 9: compute the datapoint parameters $\psi \leftarrow \{\psi_x^1, \dots, \psi_x^R\}$
 - 10: train a classifier f_x
 - 11: estimate the new test loss ℓ_x
 - 12: compute the loss reduction $\delta_x \leftarrow \ell_\tau - \ell_x$
 - 13: $\xi_m \leftarrow \begin{bmatrix} \phi_\tau^1 & \dots & \phi_\tau^K & \psi_x^1 & \dots & \psi_x^R \end{bmatrix}$, $\delta_m \leftarrow \delta_x$
 - 14: $\Xi \leftarrow \{\xi_m\}$, $\Delta \leftarrow \{\delta_m\} : 1 \leq m \leq M$
 - 15: **Return:** matrix of learning states $\Xi \in \mathbb{R}^{M \times (K+R)}$, vector of reductions in error $\Delta \in \mathbb{R}^M$
-

To build an AL strategy LALINDEPENDENT we repeat the DATAMONTECARLO procedure for Q different initializations $\mathcal{L}_\tau^1, \mathcal{L}_\tau^2, \dots, \mathcal{L}_\tau^Q$ and T various labelled subset sizes $\tau = 2, \dots, T + 1$ (Algorithm 2 lines 4 and 5). For each initialization q and iteration τ , we sample M different datapoints x each of which yields classifier/datapoint state pairs with an associated reduction in error (Algorithm 1, line 13). This results in a matrix $\Xi \in \mathbb{R}^{(QMT) \times (K+R)}$ of observations ξ and a vector $\Delta \in \mathbb{R}^{QMT}$ of labels δ (Algorithm 2, line 9):

$$\begin{bmatrix} \phi_1(\mathcal{L}_2^1) & \dots & \phi_k(\mathcal{L}_2^1) & \psi_1(x_1^{12}) & \dots & \psi_r(x_1^{12}) \\ \phi_1(\mathcal{L}_2^1) & \dots & \phi_k(\mathcal{L}_2^1) & \psi_1(x_2^{12}) & \dots & \psi_r(x_2^{12}) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathcal{L}_\tau^q) & \dots & \phi_k(\mathcal{L}_\tau^q) & \psi_1(x_m^{q\tau}) & \dots & \psi_r(x_m^{q\tau}) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathcal{L}_T^Q) & \dots & \phi_k(\mathcal{L}_T^Q) & \psi_1(x_M^{QT}) & \dots & \psi_r(x_M^{QT}) \end{bmatrix}.$$

Our insight is that observations ξ should lie on a smooth manifold and that similar states of the classifier result in similar behaviours when annotating similar samples. From this, a regression function can predict the potential error reduction of annotating a specific sample in a given classifier state. Line 10 of the BUILDLALINDEPENDENT algorithm looks for a mapping $g : \xi \mapsto \delta$. This mapping is not specific to the dataset \mathcal{D} , and thus can be used to detect samples that promise the greatest increase in classifier performance in other target domains \mathcal{Z} . The resulting LALINDEPENDENT strategy greedily selects a datapoint with the highest potential error reduction at iteration t by taking the maximum

Algorithm 2 BUILDLALINDEPENDENT

-
- 1: **Input:** iteration range $\{\tau_{\min}, \dots, \tau_{\max}\}$, classification procedure f
 - 2: SPLIT \leftarrow random partitioning function
 - 3: **Initialize:** generate train set \mathcal{D} and test dataset \mathcal{D}'
 - 4: **for** τ **in** $\{\tau_{\min}, \dots, \tau_{\max}\}$ **do**
 - 5: **for** $q = 1$ **to** Q **do**
 - 6: $\Xi_{\tau q}, \Delta_{\tau q} \leftarrow \text{DATAMONTECARLO}(\mathcal{D}, \mathcal{D}', f, \text{SPLIT}, \tau)$
 - 7: $\Xi, \Delta \leftarrow \{\Xi_{\tau q}\}, \{\Delta_{\tau q}\}$
 - 8: train a regressor $g : \xi \mapsto \delta$ on data Ξ, Δ
 - 9: construct LALINDEPENDENT $\mathcal{A}(g)$:

$$x^* = \arg \max_{x \in \mathcal{U}_t} g[\xi_{t,x}]$$
 - 10: **Return:** LALINDEPENDENT AL strategy $\mathcal{A}(g)$
-

of the value predicted by the regressor g :

$$x^* = \arg \max_{x \in \mathcal{U}_t} g(\phi_t, \psi_x). \quad (3.1)$$

3.4.2 Iterative LAL

For any AL strategy at iteration $t > 0$, the labelled set \mathcal{L}_t consists of samples selected at previous iterations, which is clearly *not* random. However, in Section 3.4.1 the dataset \mathcal{D} is split into \mathcal{L}_τ and \mathcal{U}_τ randomly no matter how many labelled samples τ are available.

To account for this, we modify the approach of Section 3.4.1 in Algorithm 3 BUILDLALITERATIVE. Instead of partitioning the dataset \mathcal{D} into \mathcal{L}_τ and \mathcal{U}_τ randomly, we suggest simulating the AL procedure which selects datapoints according to the strategy learnt on the previously collected data (Algorithm 3, line 10). It first learns a strategy $\mathcal{A}(g_2)$ based on a regression function g_2 which selects the most promising 3rd datapoint when 2 random points are available. In the next iteration, it learns a strategy $\mathcal{A}(g_3)$ that selects 4th datapoint given 2 random points and 1 selected by $\mathcal{A}(g_2)$ etc. In this way, samples at each iteration depend on the samples at the previous iteration and the sampling bias of AL is represented in the data Ξ, Δ from which the final strategy LALITERATIVE is learnt.

The resulting strategies LALINDEPENDENT and LALITERATIVE are both reasonably fast during the on-line steps of AL: they just require evaluating the RF regressor. The off-line part, generating a datasets to learn a regression function, can induce a significant computational cost depending on the parameters of the algorithm. For this reason, LALINDEPENDENT is preferred to LALITERATIVE when an application-specific strategy is needed.

Algorithm 3 BUILDLALITERATIVE

```

1: Input: iteration range  $\{\tau_{\min}, \dots, \tau_{\max}\}$ , classification procedure  $f$ 
2: SPLIT  $\leftarrow$  random partitioning function
3: Initialize: generate train set  $\mathcal{D}$  and test dataset  $\mathcal{D}'$ 
4: for  $\tau$  in  $\{\tau_{\min}, \dots, \tau_{\max}\}$  do
5:   for  $q = 1$  to  $Q$  do
6:      $\Xi_{\tau q}, \Delta_{\tau q} \leftarrow$  DATAMONTECARLO ( $\mathcal{D}, \mathcal{D}', f, \text{SPLIT}, \tau$ )
7:      $\Xi_{\tau}, \Delta_{\tau} \leftarrow \{\Xi_{\tau q}, \Delta_{\tau q}\}$ 
8:     train regressor  $g_{\tau} : \xi \mapsto \delta$  on  $\Xi_{\tau}, \Delta_{\tau}$ 
9:     SPLIT  $\leftarrow \mathcal{A}(g_{\tau})$ 
10:  $\Xi, \Delta \leftarrow \{\Xi_{\tau}, \Delta_{\tau}\}$ 
11: train a regressor  $g : \xi \mapsto \delta$  on  $\Xi, \Delta$ 
12: construct LALITERATIVE  $\mathcal{A}(g)$ 
13: Return: LALITERATIVE

```

3.5 Experiments

3.5.1 Implementation details

The code for our experiments is made publicly available³. We test AL strategies in two possible settings:

- a) *cold start*, where we start with one sample from each of two classes and
- b) *warm start*, where a larger dataset of size $N_0 \ll N$ is available to train the initial classifier.

In *cold start* we take the representative dataset to be a 2D synthetic dataset where class-conditional data distributions are Gaussian and we use the same LAL regressor in all 7 classification tasks. Each class of the representative dataset comes from a Gaussian distribution with a randomly generated mean and variance. We set the size of its training and test subsets to 400 and 4000 respectively and the proportion of class 0 varies from 0.1 to 0.9. Each mean is drawn independently from a uniform distribution from 0 to 1 and the covariance is obtained by multiplying matrices whose entries are drawn uniformly between -0.5 and 0.5 with their transposes. Four examples of the representative dataset are shown in Figure 3.3.

The LAL data generation parameters of Section 3.4 are set to the following values for cold start experiments: $M = 100$, $T = 48$, $Q = 500$. For every new initialization we use a new representative dataset that insures that the learnt strategy can generalize to various problems. The quality of the learnt AL strategy could be sensitive to the choice of these

³<https://github.com/ksenia-konyushkova/LAL>

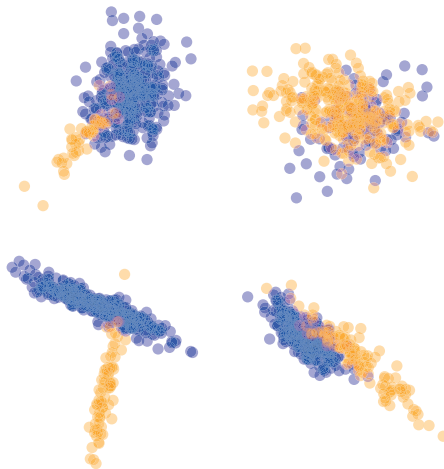


Figure 3.3 – 4 examples of the synthetic datasets with as a representative dataset and in the experiments with 2 Gaussian clouds.

parameters. For our selection we use a validation procedure, where the parameters are chosen such that the learnt AL strategy performs well on unseen datasets obtained with the same dataset generation procedure.

While we mostly concentrate on *cold start* scenario, we look at a few examples of *warm start* because we believe that it is largely overlooked in the literature, but it has a significant practical interest. Learning a classifier for a real-life application with AL rarely starts from scratch, but a small initial annotated set is provided to understand if a learning-based approach is applicable at all. While a small set is good to provide an initial insight, a real working prototype still requires much more training data. In this situation, we can benefit from the available training data to learn a specialized AL strategy for an application. In *warm start* experiments, we used 100 or 200 samples (in *Splice* and *Higgs* datasets correspondingly), out of which 40% were used to estimate the test error and 60% for collecting LAL data. Besides, we used multiple permutations of training and testing data to compensate for the limited amount of data (compared to the synthetic data). The LAL data generation parameters are the following. For *Splice* dataset, $Q = 100$ and $M = 10$, $\tau = 10, 14, \dots, 48$, $T = 12$. For *Higgs* dataset, $Q = 100$, $M = 10$ and $\tau = 50, 55, \dots, 110$, $T = 12$. The experiments show that is selected values are enough to interpolate between the learning states.

In most of the experiments, we use Random Forest (RF) classifiers for f and a RF regressor for g . The state of the learning process ξ_t at time t consists of the following features:

- a) predicted *probability* $p(y = 0 | \mathcal{L}_t, x)$;
- b) *proportion* of class 0 in \mathcal{L}_t ;
- c) *out-of-bag* cross-validated accuracy of f_t ;

- d) variance of *feature importances* of f_t ;
- e) *forest variance* computed as variance of trees' predictions on \mathcal{U}_t ;
- f) average *tree depth* of the forest;
- g) *size* of \mathcal{L}_t .

Notice that only a) depends on a datapoint and the other parameters are fixed for a particular iteration of AL. Thus, the information about a datapoint is identical to the information that uncertainty sampling has.

When we use GP as a classifier, we operate on the following features:

- a) predicted probability $p(y = 0 | \mathcal{L}_t, x)$;
- b) predicted variance by GP;
- c) variance and
- d) lengthscale of RBF kernel;
- e) kernel density estimation for x with respect to labelled and
- f) unlabelled samples;
- g) *size* of \mathcal{L}_t .

3.5.2 Baselines and protocol

We consider the three versions of our approach:

- a) **LAL-independent-2D**, LALINDEPENDENT strategy trained on a synthetic dataset of *cold start*;
- b) **LAL-iterative-2D**, LALITERATIVE strategy trained on a synthetic dataset of *cold start*;
- c) **LAL-independent-WS**, LALINDEPENDENT strategy trained on *warm start* representative data.

When it is clear which version of the algorithm is considered, we will refer to the first two strategies as **LAL-ind** and **LAL-iter**.

The LAL regressor is represented by RF regressor that requires a set of meta-parameters. Their values were set with a cross validation of a regression problem with the regression performance is measured by R squared metrics. The cross-validated parameters for the LAL strategies can be found in a Table 3.1

We compare them against the following 4 baselines:

- a) **Rs**, random sampling;
- b) **Us**, uncertainty sampling;

Table 3.1 – Cross-validated parameters of LAL strategies.

Strategy	Dataset	# trees	max depth of trees	max features per split
LAL-independent-2D	All	2000	40	6
LAL-iterative-2D	All	1000	30	7
LAL-independent-WS	<i>Splice</i>	500	10	6
LAL-independent-WS	<i>Higgs</i>	1000	40	7

- c) **Kapoor** [84], an algorithm that balances exploration and exploitation by incorporating mean and variance estimation of the GP classifier;
- d) **ALBE** [62], a recent example of meta-AL that adaptively uses a combination of strategies, including **Us**, **Rs** and that of Huang et al. [66] (a strategy that uses the topology of the feature space in the query selection). The method of Hsu and Lin [62] is chosen as our main baseline because it is a recent example of meta AL and is known to outperform several benchmarks.

In all AL experiments we select samples from a training set and report the classification performance on an independent test set. We repeat each experiment 50–100 times with random permutations of training and testing splits and different initializations. Then we report the average test performance as a function of the number of labelled samples. The performance metrics are task-specific and include classification accuracy, IOU [49], dice score [57], AMS score [3], as well as area under the ROC curve (AUC).

3.5.3 Synthetic data

Two-Gaussian-clouds experiments In this dataset we test our approach with two classifiers: RF and Gaussian Process classifier (GPC). Due to the the computational cost of GPC, it is only tested in this experiment. We generate 100 new unseen synthetic datasets of the form as shown in Figure 3.3 and use them for testing AL strategies. The details about the datasets are presented in Table 3.3. The results of AL experiments are presented in Figures 3.4 (a) and (b). In both cases the proposed LAL strategies select datapoints that help to construct better classifiers faster than **Rs**, **Us**, **Kapoor** and **ALBE**. We conclude that even in this simple task, when the classifier does not need much data to get close to the optimal performance, better decisions can be made by taking the learning state into account.

XOR-like experiments XOR-like datasets are known to be challenging for many machine learning methods and AL is no exception. It was reported in Baram et al. [12] that various AL algorithms struggle with tasks such as those depicted in Figure 3.5(a) and (b), namely *Checkerboard* 2×2 and *Checkerboard* 4×4 . Additionally, we consider *Rotated*

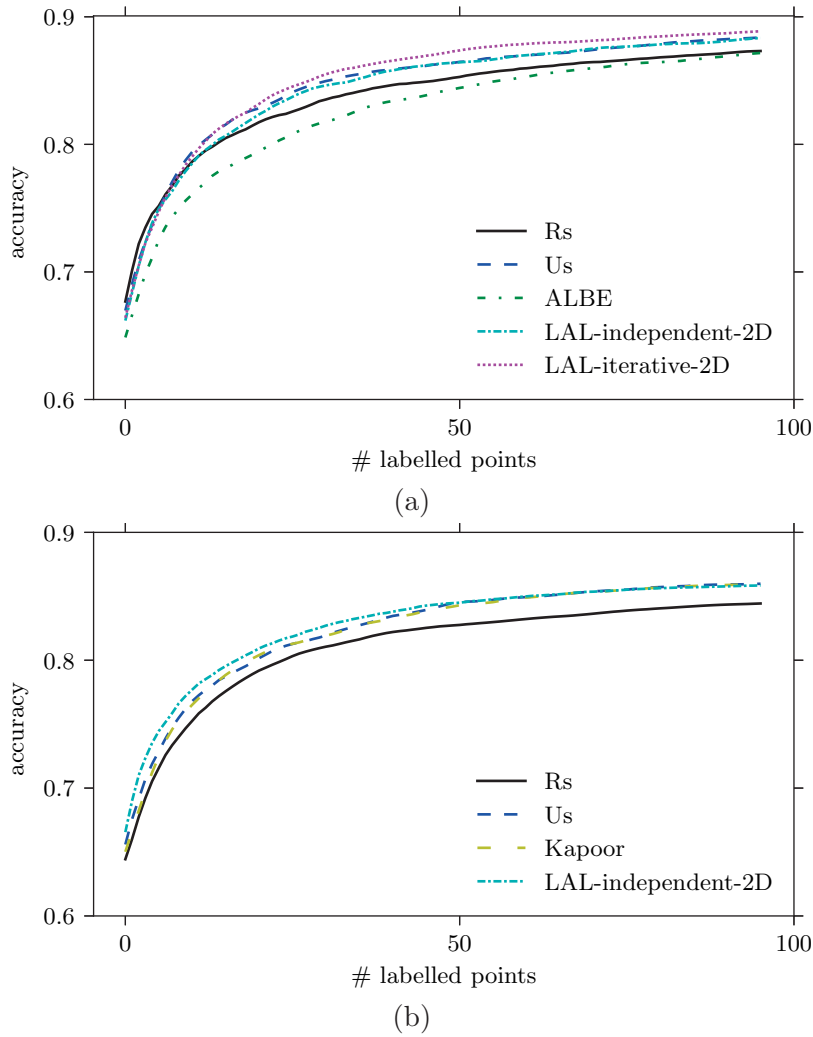


Figure 3.4 – Experiments on the synthetic data, 2 Gaussian clouds with (a) RF classifier (b) GP classifier.

Checkerboard 2×2 dataset (Figure 3.5c). Details about the datasets are presented in Table 3.3. The task for RF becomes more difficult in this case because the discriminating features are no longer aligned to the axis. Note that although these datasets are still synthetic, they do not resemble the data used to train LAL depicted in Figure 3.3.

The experimental results are shown in Figure 3.6. As previously observed [12], **Us** loses to **Rs** in these cases. **ALBE** does not suffer from such adversarial conditions as much as **Us**, but **LAL-iterative-2D** outperforms it on all XOR-like datasets. The standard deviation of the results of all synthetic experiments are presented in Table 3.2. There, we report the standard deviation at the last and middle iterations of each of the experiments.

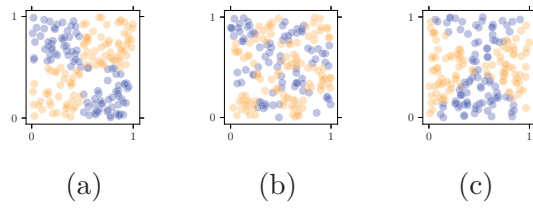
Figure 3.5 – An example of *Checkerboard* (a) 2×2 (b) 4×4 (c) *Rotated* 2×2 .

Table 3.2 – Standard deviation of 500 experiments in cold start and 100 experiments in warm start.

AL strategy	Rs	Us	ALBE	Kapoor	LAL-ind	LAL-iter
Cold start						
Dataset	last iteration					
<i>2 Gauss+RF</i>	0.0773	0.0810	0.0898	—	0.0851	0.0776
<i>2 Gauss+GP</i>	0.0683	0.0690	—	0.0708	0.0723	—
<i>Checkerboard</i> 2×2	0.0087	0.0473	0.0012	—	0.0362	0.0051
<i>Checkerboard</i> 4×4	0.0213	0.0410	0.0307	—	0.0347	0.0436
<i>Rotated checkerboard</i>	0.0146	0.0091	0.0096	—	0.0110	0.0098
	middle iterations					
<i>2 Gauss+RF</i>	0.0814	0.0945	0.1024	—	0.0909	0.0840
<i>2 Gauss+GP</i>	0.0777	0.0823	—	0.0865	0.0812	—
<i>Checkerboard</i> 2×2	0.0143	0.1142	0.0090	—	0.0569	0.0359
<i>Checkerboard</i> 4×4	0.0289	0.0475	0.0301	—	0.0400	0.0307
<i>Rotated checkerboard</i>	0.0217	0.0788	0.0187	—	0.0177	0.0340
	last iteration					
<i>Striatum</i>	0.0751	0.0255	—	—	0.0230	0.0389
<i>MRI</i>	0.1368	0.0182	—	—	0.0220	0.0250
<i>Credit card</i>	0.0422	0.0127	—	—	0.0113	0.0117
	middle iterations					
<i>Striatum</i>	0.1033	0.0318	—	—	0.0303	0.0497
<i>MRI</i>	0.1850	0.0277	—	—	0.0305	0.0330
<i>Credit card</i>	0.0446	0.0124	—	—	0.0122	0.0103
Warm start						
	last iteration					
<i>Splice</i>	0.0076	0.0061	0.0058	—	0.0058	—
<i>Higgs</i>	3.5590	2.7735	—	—	1.6883	—
	middle iteration					
<i>Splice</i>	0.0099	0.0077	0.0059	—	0.0081	—
<i>Higgs</i>	3.8875	3.7244	—	—	3.0531	—

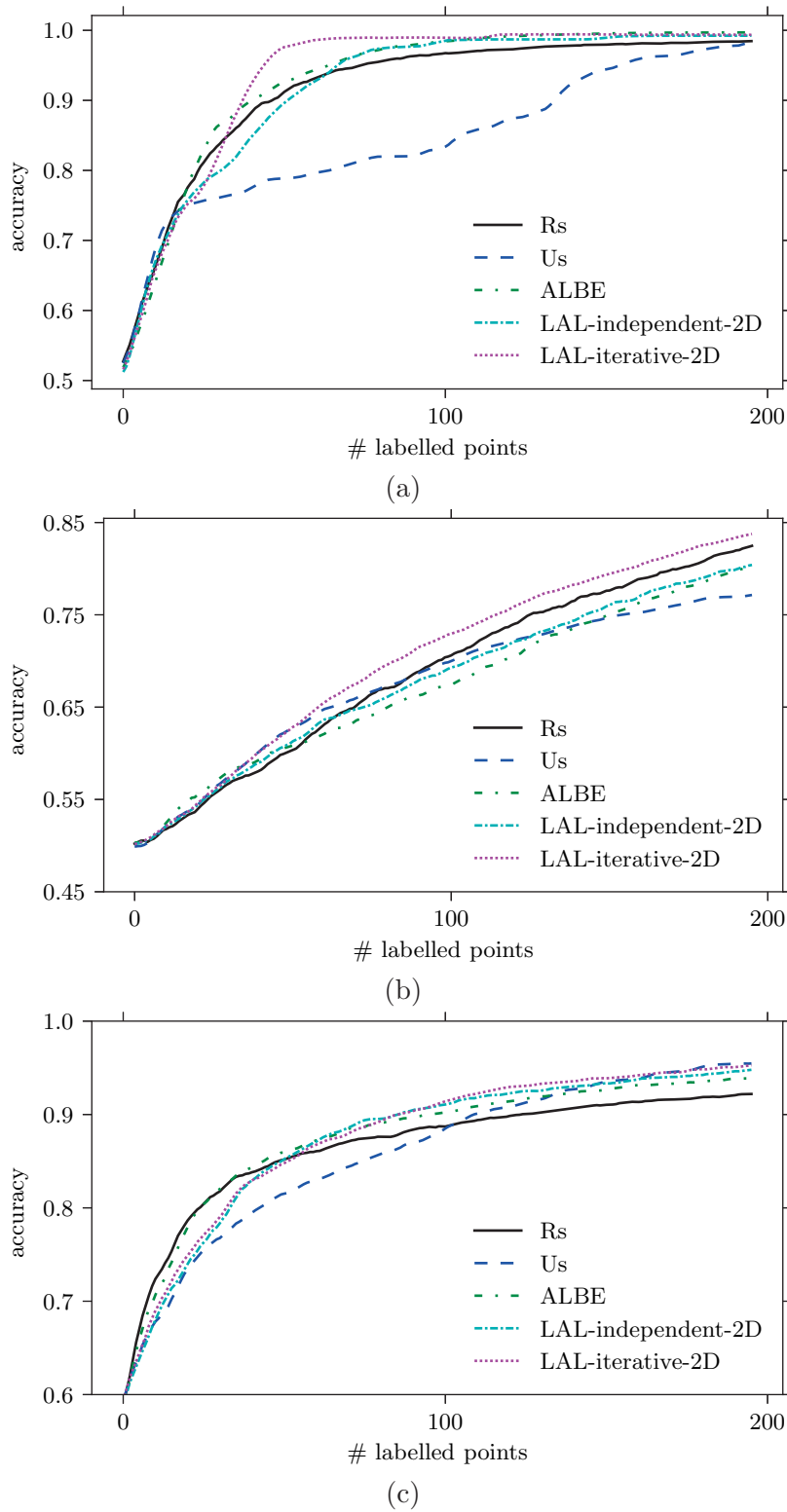


Figure 3.6 – Experiments on *Checkerboard* (a) 2×2 (b) 4×4 (c) rotated 2×2 .

3.5.4 Real data

We now turn to real data from domains where annotating is hard because it requires special training to do it correctly:

Striatum, 3D Electron Microscopy stack of rat neural tissue (Figure 2.1), the task is to detect and segment mitochondria [119], the protocol is the same as in Chapter 2;

MRI, brain scans (Figure 2.20) obtained from the BRATS competition [126], the task is to segment brain tumour in T1, T2, FLAIR, and post-Gadolinium T1 MR images, the protocol is the same as in Chapter 2;

Credit card, a dataset of credit card transactions made in 2013 by European cardholders [40], the task is to detect fraudulent transactions. We use 30 features are the result of PCA on the real features that are not provided due to the confidentiality issues. This is highly imbalanced dataset with only 0.17% of fraud transactions among normal transactions (see Table 3.3);

Splice, a molecular biology dataset with the task of detecting splice junctions between exons and introns in DNA sequences [116]. The sequences attributes are encoded numerically and a problem is formulated as a binary classification task;

Higgs, a high energy physics dataset that contains measurements simulating the ATLAS experiment [3]. The task to classify events into classes of tau tau decay of a Higgs boson and background noise. We preprocess the data by replacing missing feature values with the median of the corresponding feature.

The details about the above datasets including sizes, dimensionalities are presented in Table 3.3.

Cold start AL Figure 3.7 depicts the results of applying **Rs**, **Us**, **LAL-independent-2D**, and **LAL-iterative-2D** on the *Striatum*, *MRI*, and *Credit card* datasets and standard deviations are presented in Table 3.2. Both LAL strategies outperform **Us**, with **LAL-iterative-2D** being the best of the two. The best score of **Us** in these complex real-life tasks is reached 2.2–5 times faster by the **LAL-iterative-2D**. Considering that the LAL regressor was learned using a simple synthetic 2D dataset, it is remarkable that it works effectively on such complex and high-dimensional tasks. Notice that the *Credit card* dataset (Figure 3.7 c) is very imbalanced, and thus random sampling struggles to find at least one positive datapoint during 300 sampling iterations.

Due to the high computational cost of **ALBE**, we downsample *Striatum* and *MRI* datasets to 2000 datapoints (referred to as *Striatum mini* and *MRI mini*). Downsampling was not possible for the *Credit card* dataset due to the sparsity of positive labels (0.17%). We see in Figure 3.8 that **ALBE** performs worse than **Us** but better than **Rs**. We ascribe

Table 3.3 – Parameters of the datasets.

Dataset	Dimensions	# training samples	# test samples	positive class %
<i>2 Gauss clouds</i>	2	400	4000	50
<i>Checkerboard</i>	2	1000	1000	50
<i>Striatum</i>	272	276 130	294 496	11.59
<i>Striatum mini</i>	272	2000	2000	11.59
<i>MRI</i>	188	22 934	22 562	5.99
<i>MRI mini</i>	188	2000	2000	5.99
<i>Credit</i>	30	142 403	142 404	0.17
<i>Splice</i>	60	1000	2175	48.09
<i>Higgs</i>	30	125 000	125 000	34.26

this to the lack of labelled data, which **ALBE** needs to estimate classification accuracy (see Section 1.3.2).

Warm start AL Learning a strategy that is specific for a problem at hand is valuable for applications where task or feature distribution is so different from synthetic data that LAL strategy cannot be transferred, for example, for tasks where feature distributions contain missing or categorical variables. In Figure 3.9 we compare **LAL-independent-WS** on the *Splice* and *Higgs* datasets by initializing `BUILDLALINDEPENDENT` with 100 and 200 datapoints from the corresponding tasks. Notice that this is the only experiment where a significant amount of labelled data in the domain of interest is available prior to AL. We tested **ALBE** on the *Splice* dataset, however in the *Higgs* dataset the number of iterations in the experiment is too big. **LAL-independent-WS** outperforms other methods with **ALBE** delivering competitive performance—yet, at a high computational cost—only after many AL iterations. We conclude that the LAL approach can be indeed useful in this problem setting. Finally, the standard deviations of the results are presented in Table 3.2.

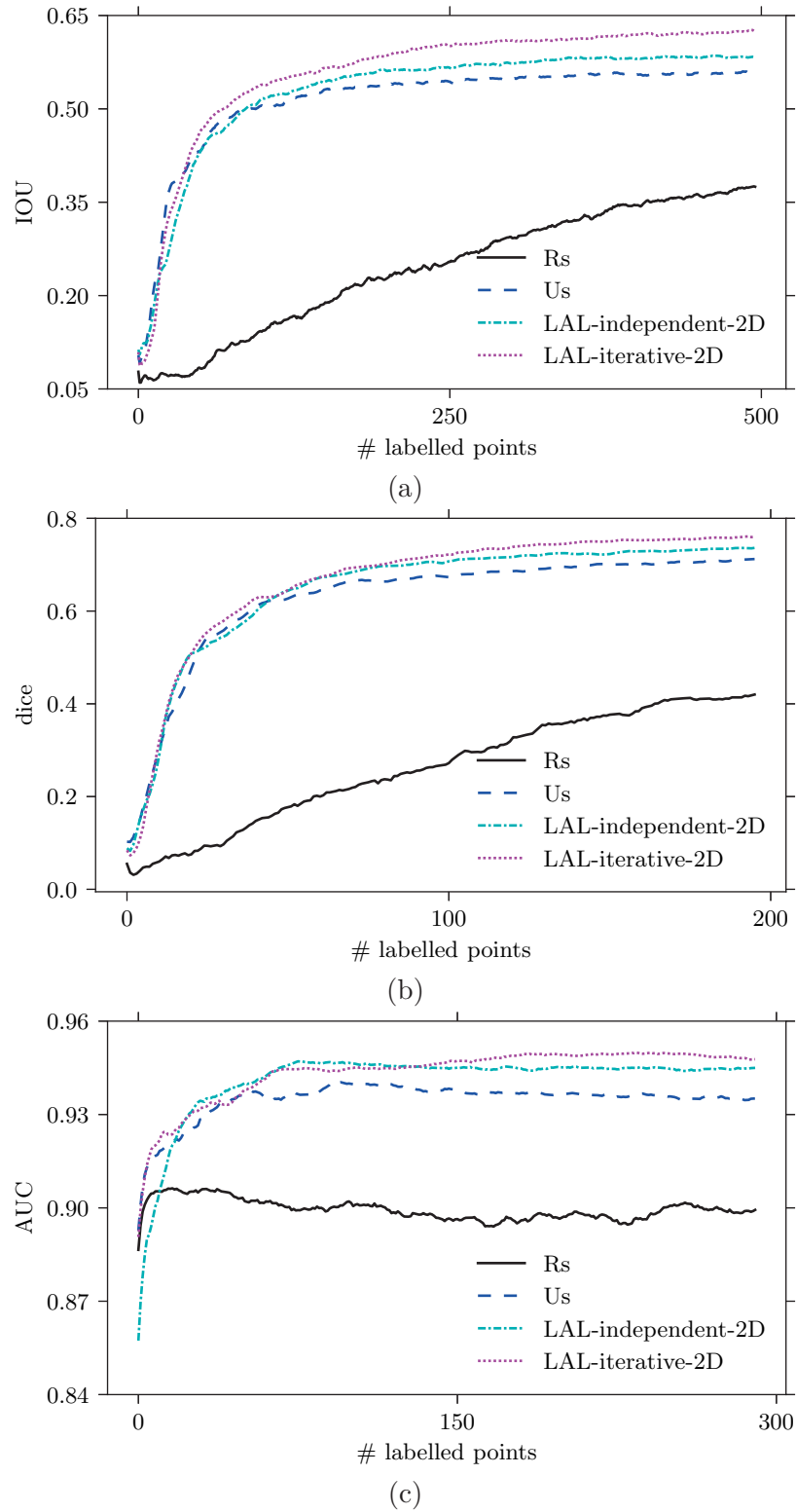
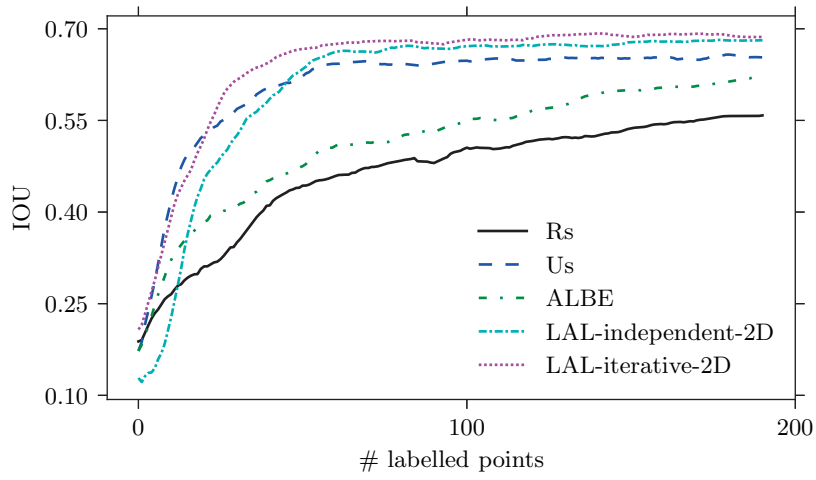
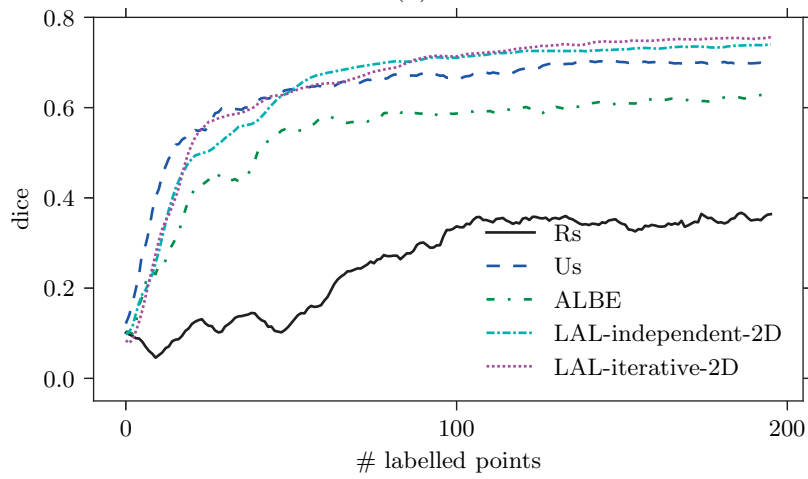


Figure 3.7 – Experiments on real data. (a) IOU for *Striatum*, (b) dice score for *MRI* and (c) AUC for *Credit card* as a function of a number of labelled points.



(a)



(b)

Figure 3.8 – Comparison with **ALBE** on (a) the *Striatum mini* and (b) *MRI mini* datasets.

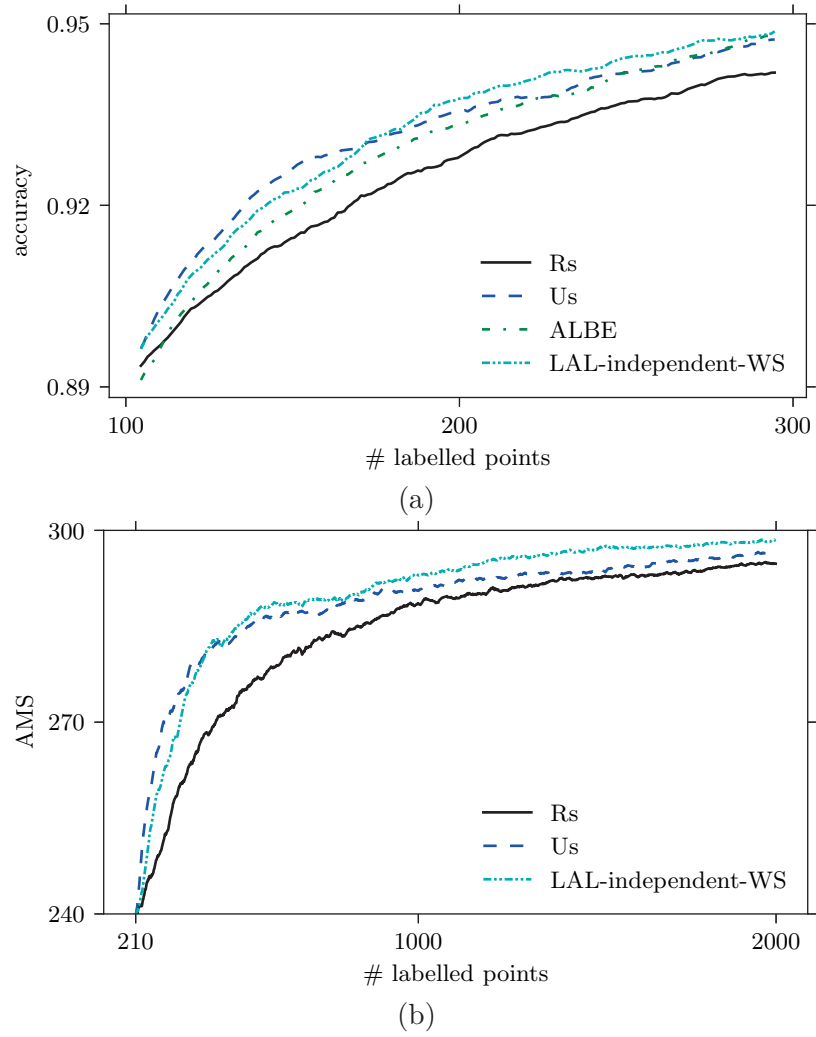


Figure 3.9 – Experiments on the real datasets in warm start scenario. (a) Accuracy for *Splice*, (b) AMS score for *Higgs*.

3.5.5 Analysis of LAL strategies and time comparison

To better understand LAL strategies, we show in Figure 3.10 the relative importance of the features of the regressor g for LALITERATIVE. We observe that both classifier state parameters and datapoint parameters influence the AL selection giving evidence that both of them are important for selecting a point to label.

In order to understand what kind of selection LALINDEPENDENT and LALITERATIVE do, we record the predicted probability of the chosen datapoint $p(y^* = 0 | \mathcal{D}_t, x^*)$ in 10 *cold start* experiments with the same initialization on the *MRI* dataset. Figure 3.11 shows the histograms of these probabilities for **Us**, **LAL-independent-2D** and **LAL-iterative-2D**. LAL strategies have high variance and modes different from 0.5. Not only does the selection by LAL strategies differ significantly from standard **Us**, but also the independent and iterative approaches differ from each other.

Computational costs While collecting synthetic data can be slow, it must only be done *once, off-line*, for all applications. Besides, Algorithm 1, 2 and 3 can be trivially parallelised thanks to a number of independent loops. Collecting data off-line for *warm start*, that is application specific, took us approximately 2.7h and 1.9h for *Higgs* and *Splice* datasets respectively. By contrast, the on-line user-interaction part is fast: it simply consists of learning f_t , extracting learning state parameters and evaluating the regressor g . The LAL run time depends on the parameters of the random forest regressor which are estimated via cross-validation. Run times of a Python-based implementation running on 1 core are given in Table 3.4 for a typical parameter set ($\pm 20\%$ depending on exact parameter values). Real-time performance can be attained by parallelising and optimising the code, even in applications with large amounts of high-dimensional data.

Table 3.4 – Time in seconds for one iteration of AL for various strategies and tasks.

Dataset	Dimensions	# samples	Us	ALBE	LAL
<i>Checkerboard</i>	2	1000	0.11	13.12	0.54
<i>MRI mini</i>	188	2000	0.11	64.52	0.55
<i>MRI</i>	188	22 934	0.12	—	0.88
<i>Striatum mini</i>	272	2000	0.11	75.64	0.59
<i>Striatum</i>	272	276 130	2.05	—	19.50
<i>Credit</i>	30	142 404	0.43	—	4.73

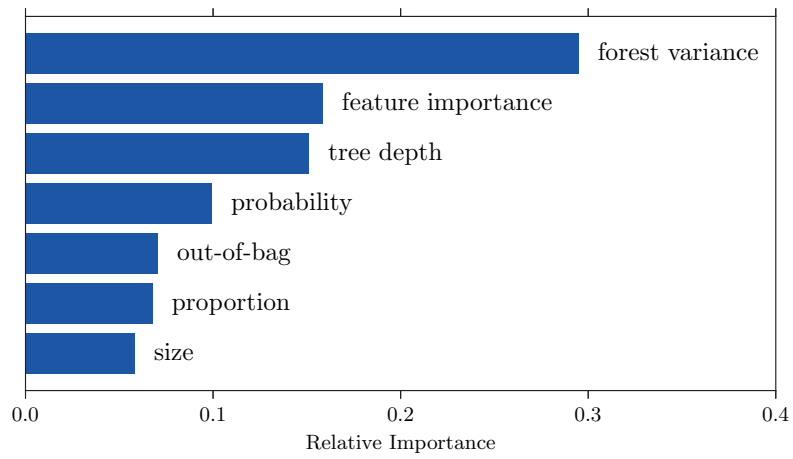


Figure 3.10 – Feature importances of the RF regressor representing LALITERATIVE strategy.

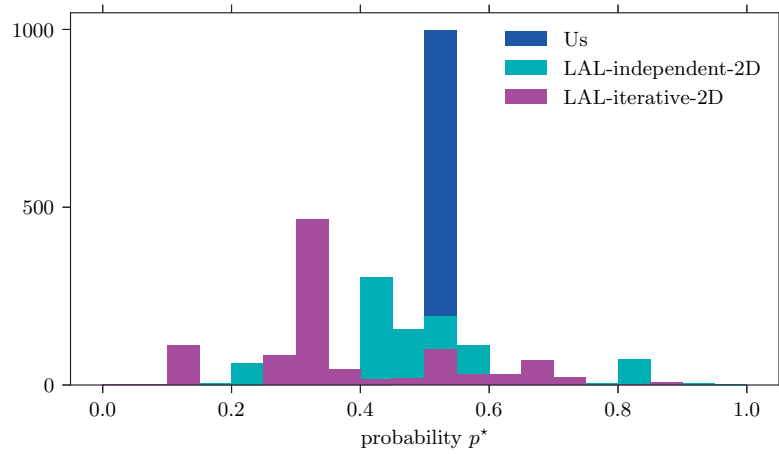


Figure 3.11 – Histograms of the selected probability for different AL strategies in experiments with *MRI* dataset.

3.6 Conclusion

In this chapter we introduce a new approach to AL that is driven by data: Learning Active Learning. We find out that learning a strategy from simple 2D data generalizes remarkably well to challenging new domains. Learning from a subset of application-specific data further extends the applicability of our approach. Finally, LAL demonstrates robustness to the choice of type of classifier and features.

The method described in this chapter suggests a solution to reducing annotation load by addressing the question “*What to annotate?*”. This is one of the first methods that suggests going away from hand-crafting selection strategies towards designing data-driven strategies. However, there are still a few areas for improvement in this approach. First of all, LAL makes greedy decisions on which datapoints to annotate by predicting the immediate reduction in generalisation error. Also, although it was demonstrated to work well with various classifiers, LAL still requires hand-crafting features for a particular classifier. In the next chapter we build on the meta-AL ideas of LAL, but we move towards more general-purpose AL by formulating the process as MDP and solving it with reinforcement learning. Among other advantages, it allows us to find non-myopic strategies and does not require hand-crafted features for every type of classifier.

4 Towards Data-Driven General-Purpose Active Learning

4.1 Introduction

In this chapter we propose a novel data-driven approach to AL that brings us one step closer to general-purpose AL. Data-driven approach to AL makes it possible to go beyond human intuition and potentially to discover completely new strategies by accounting for the state of the trained ML model when selecting the data to annotate. However, many of these methods are still limited to either learning from closely related domains [9, 50, 113], or using a greedy selection that may be suboptimal [90, 113], or relying on properties of specific classifiers [90, 9, 37, 149]. In short, even though data-driven AL methods have flourished recently, there is still no *general-purpose* non-myopic methods that transfer across different kinds of data and various ML models used in training. In this chapter, we introduce such a generic data-driven AL method that is applicable to heterogeneous datasets and to most ML models because it does not require hand-crafting model- or dataset-specific features.

More specifically, we reformulate AL as a Markov Decision Process (MDP) and use reinforcement learning (RL) to find AL strategy as an optimal MDP policy. To achieve the desired generality, we incorporate two important contributions into our approach. First, we take the AL objective to be minimizing the number of annotations required to achieve a given prediction quality, which is a departure from standard AL approaches that maximize the performance given an annotation budget. In this way, we optimise what the practitioners truly want, that is, the annotation cost, independently of the specific ML model and performance measure being used. To this end, we design the reward function of MDP to reflect our AL objective. Second, we propose a procedure that can learn an AL strategy from data coming from multiple unrelated domains for which annotations are already available. The strategy then applies to domains for which this is not the case. To this end, we defined generic MDP state and action representations that can be computed for arbitrary datasets and without regard to the specific ML model.

In our experiments we demonstrate the effectiveness of our approach for the purpose of binary classification by applying the learned strategies to previously unseen datasets from different domains. We show that they enable us to reach pre-defined quality thresholds with fewer annotations than several baselines, including recent meta-AL algorithms [62, 90]. We also analyse the properties of our strategies to understand their behaviour and how it differs from those of more traditional ones.

4.2 Related work

As we have discussed in Chapter 1, manually-designed AL methods differ in their underlying assumptions, computational costs, theoretical guarantees, and generalization behaviours. However, they all rely on a human designer having decided how the data points should be selected and the performance of any one of these strategies on a never seen before dataset is unpredictable, which makes it difficult to choose one over the other. If a single manually designed method does not consistently outperform all others, it makes sense to adaptively select the best strategy or to combine them. Still, this approach remains limited to combining existing strategies instead of learning new ones. Furthermore, strategy learning happens during AL and its success depends critically on the ability to estimate the classification performance from scarce annotated data.

Data-driven AL Recall that the most recent meta-AL works have turned to so-called *data-driven AL* approaches that learn AL strategies from annotated data [90, 9, 37, 149, 113, 50, 136]. They learn what kind of datapoints are the most beneficial for training the model given the current state of trained ML model. Then, past experience helps to eventually derive a more effective selection strategy. This has been demonstrated to be effective, but it suffers from a number of limitations. First, this approach is often tailored for learning only from related datasets and domains suitable for transfer or one-shot learning [113, 9, 50, 37, 149]. Second, many of them rely on specific properties of the ML models, be they standard classifiers [90] or few-shot learning models [9, 37, 149], which restricts their generality. Finally, in some approaches the resulting strategy is greedy—for example when supervised [90] or imitation learning [113] is used—that might lead to suboptimal data selection.

MDP formulation in data-driven AL is used both for *pool-based* AL, where datapoints are selected from a large pool of unlabelled data, and for *stream-based* AL, where datapoints come from a stream and AL decides to annotate a datapoint or not as it appears. In stream-based AL, actions—to annotate or not to—are discrete and Q-learning [193] is the RL method of choice to look for an op [198, 50]. By contrast, in pool-based AL, the action selection concerns all potential datapoints that can be annotated and it is natural to characterise them by continuous vectors that makes it not suitable for Q-learning. So, policy gradient [197, 174] methods are usually used [9, 37, 149, 136]. In this chapter

we focus on pool-based AL but we would like to reap the benefits of Q-learning that is, lower variance and better data-complexity thanks to bootstrapping. To this end, we take advantage of the fact that although actions in pool-based AL are continuous, their number is finite. Thus, we can adapt Q-learning for our purposes.

Most data-driven AL methods stipulate a specific objective function that is being maximised. However, the methods are not always evaluated in a way that is consistent with the objective that is optimized. Sometimes, the metric used for evaluation differs from the objective [9, 90, 136]. Sometimes, the learning objective may include additional factors like discounting [198, 50, 136] or may combine several objectives [198, 37]. By contrast, our approach uses our evaluation criterion—minimization of the time spent annotating for a given performance level—directly in the strategy learning process.

Among data-driven AL, the approach of Pang et al. [136] achieves generality by using multiple training datasets to learn strategies, as we do. However, this approach is more complex than ours, relies on policy-gradient RL, and uses a standard AL objective. By contrast, our approach does not require a complex state and action embedding, needs fewer RL episodes for training thanks to using Q-learning, and explicitly maximizes what practitioners care about, that is, reduced annotation cost.

RL for various tasks Traditionally, RL is used to enable an agent to act in an iterative environment and its direct application is in robotics [7, 128, 102]. However, RL methods are extended to be used in other tasks, such as active vision [29, 16, 74], learning architectures of neural networks [209, 10], visual question answering [63, 41], or image annotation [155, 2, 92].

4.3 Method

We formulate the AL process as a Markov decision process (MDP) and use reinforcement learning (RL) to find an optimal strategy. In this section, we first outline our design philosophy. We then formalize AL in MDP terms and finally describe our approach to finding an optimal MDP policy. For simplicity, we present our approach in the context of binary classification. However, an almost identical AL problem formulation can be used for other ML tasks and a separate selection policy can be trained for each one.

4.3.1 Approach

Our goal is to advance data-driven AL towards general-purpose strategy learning. Desirable strategies should have two key properties. They should be **transferable** across unrelated datasets and have sufficient **flexibility** to be applied in conjunction with different ML models. As in Chapter 3 we assume that transferability across different datasets is possible because they share meta-properties which are used by the AL strategy.

Our design decisions are geared towards learning such strategies. The iterative structure of AL is naturally suited for an MDP formulation: For every *state* of an AL problem, an *agent* takes an *action* that defines the datapoint to annotate and it receives a *reward* that depends on the quality of the model that is re-trained using the new label. An AL strategy then becomes an MDP *policy* that maps a state into an action.

To achieve seamless transferability and flexibility, our task is therefore to design the states, actions, and rewards to be generic. To this end, we represent states and actions as vectors that are independent from specific dataset feature representations and can be computed for a wide variety of ML models. For example, the probability that the classifier assigns to a datapoint suits this purpose because most classifiers estimate this value. By contrast, the number of support vectors in a support vector machine (SVM) or the number of layers of a neural network (NN) are not suitable because they are model-specific. Raw feature representations of data are similarly inappropriate because they are domain specific.

A classical AL objective is to maximize the prediction quality—often expressed in terms of accuracy, AUC, F-score, or negative squared error—for a given annotation budget. For flexibility’s sake, we prefer an objective that is not directly linked to a specific performance measure. We therefore consider the *dual* objective of minimizing the number of annotations required for a given *target quality* value. When learning a strategy by optimizing this objective, the AL agent only needs to know if the performance is above or below this target quality, as opposed to its exact value. Therefore, the procedure is less tied to a specific performance measure or setting. Our MDP reward function expresses this objective by penalizing the agent until the target quality is achieved. This motivates the agent to minimize its “suffering” by driving the amount of requested annotations down.

Having formulated the AL problem as a MDP, we can learn a strategy using RL. We simulate the annotation process on data from a collection of unrelated labelled datasets, that ensures the transferability to new unlabelled datasets. Our approach to finding the optimal policy is based on the deep Q-network (DQN) method of [129]. To apply DQN with pool-based AL, we modify it in two ways. First, we make it work with MDP where actions are represented by vectors corresponding to individual datapoints instead of being discrete. Second, we deal with the set of actions \mathcal{A}_t that change between iterations t as it makes sense to annotate a datapoint only once.

4.3.2 Formulating AL as an MDP

Let us consider an AL problem where we annotate a dataset \mathcal{D} . A test dataset \mathcal{D}' is used to evaluate the AL procedure. Then, we iteratively select a datapoint $\mathbf{x}^{(t)} \in \mathcal{D}$ to be annotated. Let f_t be a classifier trained on a subset \mathcal{L}_t that is annotated after

iteration t . This classifier assigns a numerical score $\hat{y}_t(\mathbf{x}_i) \in \mathbb{R}$ to a datapoint and then maps it to a label $y_i \in \{0, 1\}$, $f_t : \hat{y}_t(\mathbf{x}_i) \mapsto \hat{y}_i$. For example, if the score is the predicted probability $\hat{y}_t(\mathbf{x}_i) = p(y_i = 0 | \mathcal{L}_t, \mathbf{x}_i)$, the mapping function simply thresholds it at 0.5. If we wanted to perform a regression instead, $\hat{y}_t(\mathbf{x}_i)$ could be a predicted label and the mapping function would be the identity. In AL evaluation we measure the quality of classifier f_t by computing its empirical performance ℓ_t on \mathcal{D}' .

Then, we formulate AL procedure as an episodic MDP. Each AL run starts with a small labelled set $\mathcal{L}_0 \subset \mathcal{D}$ along with a large unlabelled set $\mathcal{U}_0 = \mathcal{D} \setminus \mathcal{L}_0$. The following steps are performed at iteration t .

1. Train a classifier f_t using \mathcal{L}_t .
2. A *state* s_t is characterised by f_t , \mathcal{L}_t , and \mathcal{U}_t .
3. The AL *agent* selects an *action* $a_t \in \mathcal{A}_k$ by following a policy $\pi : s_t \mapsto a_t$ that defines a datapoint $\mathbf{x}^{(t)} \in \mathcal{U}_t$ to be annotated.
4. Look up the label $y^{(t)}$ of $\mathbf{x}^{(t)}$ in \mathcal{D} and set $\mathcal{L}_{t+1} = \mathcal{L}_t \cup \{(\mathbf{x}^{(t)}, y^{(t)})\}$, $\mathcal{U}_{t+1} = \mathcal{U}_t \setminus \{\mathbf{x}^{(t)}\}$.
5. Give the agent the *reward* r_{t+1} linked to empirical performance value ℓ_t .

These steps repeat until a *terminal* state s_T is reached. In the case of *target quality* objective of Sec. 4.3.1, we reach the terminal state s_T when $\ell_T \geq q$, where q is fixed by the user, or when $T = |\mathcal{U}_0|$. The agent only observes s_t , r_{t+1} and a set of possible actions \mathcal{A}_t , while f_t , \mathcal{D}' and q are the parts of the environment. The agent aims to maximize the *return* of the AL run: $R_0 = r_1 + \dots + r_{T-1}$ by policy π that intelligently chooses the actions, that is, the datapoints to annotate. We now turn to specifying our choice for *states*, *actions*, and *rewards* that reflect the AL objective of minimizing the number of annotations while providing flexibility and transferability.

States We assume that there is a lot of unlabelled data at the start of an AL procedure. Without loss of generality, we can therefore set aside at the start of each AL run a subset $\mathcal{V} \subset \mathcal{U}_0$ and replace \mathcal{U}_0 by $\mathcal{U}_0 \setminus \mathcal{V}$. We use the classifier’s score \hat{y}_t on \mathcal{V} as a means to keep track of the state of the learning procedure. Then, we take the state representation to be a vector \mathbf{s}_t of sorted values $\hat{y}_t(\mathbf{x}_i)$ for all \mathbf{x}_i in \mathcal{V} .

Intuitively, the state representation is rich in information on, for example, the average prediction score or the uncertainty of a classifier. In Fig. 4.1, we plot the evolution of this vector for t using a policy defined by random sampling, uncertainty sampling, or our learnt strategy, all starting from the same initial state s_0 . Note that the statistics of the vectors are clearly different. Although their structure is difficult to interpret for a human, it is something RL can exploit to learn a policy.

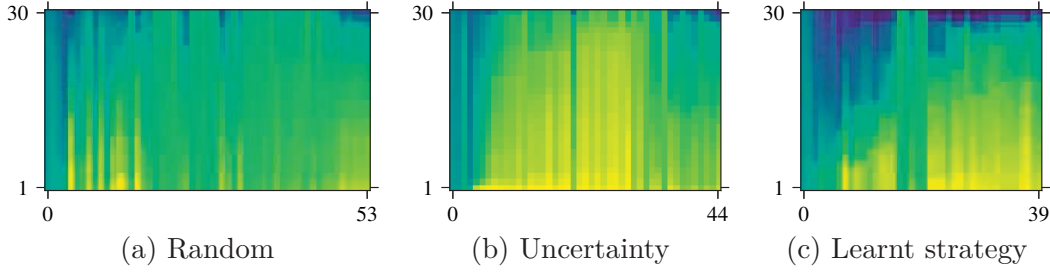


Figure 4.1 – The evolution of the learning state vector \mathbf{s}_t during an annotation episode starting from the same state for (a) random sampling, (b) uncertainty sampling, and (c) our learnt strategy. Every column represents \mathbf{s}_t at iteration t , with $|\mathcal{V}| = 30$. Yellow corresponds to values of \hat{y}_t that predict class 1 and blue – class 0.

Actions We design our MDP so that taking an action a_t amounts to selecting a datapoint $\mathbf{x}^{(t)}$ to be annotated. We characterize a potential action of choosing a datapoint \mathbf{x}_i by a vector \mathbf{a}_i which consists of the score $\hat{y}_t(\mathbf{x}_i)$ of the current classifier f_t on \mathbf{x}_i and the average distances from \mathbf{x}_i to \mathcal{L}_t and \mathcal{U}_t , that is $g(\mathbf{x}_i, \mathcal{L}_t) = \sum_{x_j \in \mathcal{L}_t} d(\mathbf{x}_i, x_j) / |\mathcal{L}_t|$ and $g(\mathbf{x}_i, \mathcal{U}_t) = \sum_{x_j \in \mathcal{U}_t} d(\mathbf{x}_i, x_j) / |\mathcal{U}_t|$, where d is a distance measure. So, at iteration t we choose an action a_t from a set $\mathcal{A}_t = \{\mathbf{a}_i\}$, where $\mathbf{a}_i = [\hat{y}_t(\mathbf{x}_i), g(\mathbf{x}_i, \mathcal{L}_t), g(\mathbf{x}_i, \mathcal{U}_t)]$ and $\mathbf{x}_i \in \mathcal{U}_t$. Notice, that \mathbf{a}_i is represented by the quantities that are not specific neither for the datasets nor for the classifiers.

Rewards To model our *target quality* objective of reaching the quality q in as few MDP iterations as possible, we choose our reward function to be $r_t = -1$. This makes the return R_0 of an AL run that terminates after T iterations to be $r_1 + \dots + r_{T-1} = -T + 1$. The fewer iterations, the larger the reward, thus the optimal policy of MDP matches the best AL strategy according to our objective. This reward structure is not greedy because it does *not* restrict the choices of the agent as long as the terminal condition is met after a small number of iterations.

4.3.3 Policy learning using RL

Thanks to our reward structure, learning an AL strategy accounts to finding an optimal (with the highest return) policy π^* of MDP that maps a state s_t into an action a_t to take, i.e. $\pi^* : s_t \mapsto a_t$. To find this optimal policy π^* we use DQN [129] method on the data that is already annotated. In our case, $Q^\pi(s_t, a_i)$ aims to predict $-(T - t)$: a negative amount of iterations that are remaining before a target quality is reached from state s_t after taking action a_i and following the policy π afterwards. Note that it is challenging to learn from our reward function because the positive feedback is only received at the end of the run, thus the credit assignment is difficult.

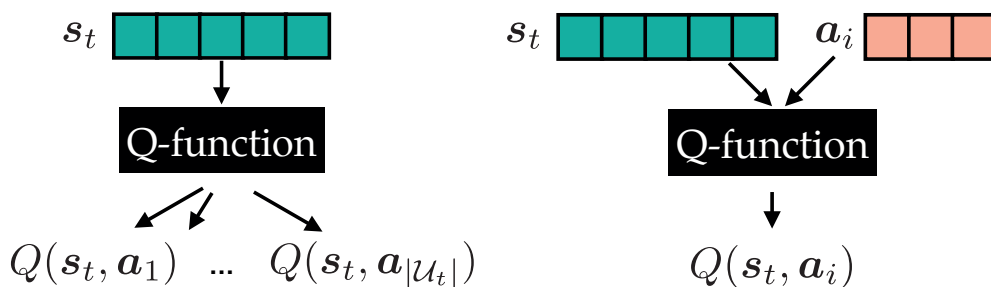


Figure 4.2 – Adapting the DQN architecture. Left: In standard DQN, the Q-function takes the state vector as input and yields an output for each discrete action. Right: In our version, actions are represented by vectors. The Q-function takes action and the state as input and returns a single value.

Procedure To account for the diversity of AL experiences we use a collection of Z annotated datasets $\{\mathcal{Z}_i\}_{1 \leq i \leq Z}$ to simulate AL *episodes*. We start from a random policy π . Then, learning is performed by repeating the following steps:

1. Pick a labelled dataset $\mathcal{Z} \in \{\mathcal{Z}_i\}$ and split it into subsets \mathcal{D} and \mathcal{D}' .
2. Use π to simulate AL episodes on \mathcal{Z} by initially hiding the labels in \mathcal{D} and following an MDP as described in Sec. 4.3.2. Keep the *experience* in the form of transitions $(\mathbf{s}_t, \mathbf{a}_t, r_{t+1}, \mathbf{s}_{t+1})$.
3. Update policy π according to the *experience* with the DQN update rule.

Even though the features are specific for every \mathcal{Z} , the experience in the form of transitions $(\mathbf{s}_t, \mathbf{a}_t, r_{t+1}, \mathbf{s}_{t+1})$ is of the same nature for all datasets, thus a single strategy is learned for the whole collection. When the training is completed, we obtain an optimal policy π^* .

In the standard DQN implementation, the Q-function takes a state representation \mathbf{s}_t as input and outputs several values corresponding to discrete actions [129], as shown in Fig. 4.2(a). However, we represent actions by vectors \mathbf{a}_i and each of them can be chosen only once per episode as it does not make sense to annotate the same point twice. To account for this, we treat actions as inputs to the Q-function along with states and adapt the standard DQN architecture accordingly, as shown in Fig. 4.2(b). Then, Q-values for the required actions are computed on demand for $\mathbf{a}_i \in \mathcal{A}_t$ through a feed-forward pass through the network. As our modified architecture is still suitable for Q-learning [193, 173], and the same optimization procedure as in a standard DQN can be used. Finding $\max_{\mathbf{a}_i} Q^\pi(\mathbf{s}_t, \mathbf{a}_i)$ still is possible because our set of actions is finite and the procedure has the same computational complexity as an AL iteration.

4.4 Experiments

In this section, we demonstrate the transferability and flexibility of our method, as defined in Sec. 4.3.1, and analyse its behaviour. The corresponding code is publicly available¹.

4.4.1 Baselines and Parameters

Baselines We will refer to our method as **LAL-RL** and compare it against the following 7 baselines. The first 3 are manually-designed. The next 3 are meta-AL algorithms with open source implementations. The final approach is similar in spirit to ours but no code is available on-line.

Rs, random sampling. The datapoint to be annotated is picked at random.

Us, uncertainty sampling [103], selects a datapoint that maximizes the Shannon entropy H over the probability of predictions: $\mathbf{x}^{(t)} = \arg \max_{\mathbf{x}_i \in \mathcal{U}_t} H[p(y_i = y | \mathcal{L}_t, \mathbf{x}_i)]$.

QUIRE [66], a query selection strategy that uses the topology of the feature space. This strategy accounts for both the informativeness and representativeness of datapoints. The vector that characterizes our actions is in the spirit of this representativeness measure.

ALBE [62], a recent meta-AL algorithm that adaptively combines strategies, including **Us**, **Rs** and **QUIRE**.

LAL-ind [90], a recent approach that formulates AL as a regression task and learns a greedy strategy that is transferable between datasets.

LAL-iter [90], a variation of **LAL-ind** that tries to better account for the bias caused by AL selection.

MLP-GAL(Te) [136], a recent method that learns a strategy from multiple datasets with a policy gradient RL method.

AL parameters We use logistic regression (LogReg) or SVM as our base classifiers for AL. We scale the feature vectors, but then we make no effort to tune the classifiers and use their sklearn python implementations with default parameters. This corresponds to a realistic scenario where there is no obvious way to choose parameters. For LogReg, they include l^2 penalty with regularization strength 1 and a maximum of 100 iterations. For SVM the most important parameters include rbf kernel and penalty parameter of 1. The distance measure d between datapoints is the cosine distance.

DQN implementation details RL with non-linear Q-function approximation is not guaranteed to converge, but in practice it still finds a good policy with a few tricks. We use separate target network and experience replay of [129], warm start, and prioritized

¹<https://github.com/ksenia-konyushkova/LAL-RL>.

replay of [159]. Besides, instead of reward normalisation we initialise the bias of the last layer to the average reward that an agent receives in warm start episodes. To compute $Q^\pi(\mathbf{s}_t, \mathbf{a}_i)$ we use NN where first \mathbf{s}_t goes in and a compact representation of it is learnt, then, \mathbf{a}_i is added to it and $Q^\pi(\mathbf{s}_t, \mathbf{a}_i)$ is the output. We use fully connected layers with sigmoid activations except for the final layer that is linear. We perform 1000 RL iterations, each of which consists of 10 AL episodes and 60 updates of the Q-function. As $\hat{y}(\mathbf{x}_i)$ we use $p(y_i = 0 | \mathcal{L}_t, \mathbf{x}_i)$. The size of \mathcal{V} is set to 30.

RL parameters Recall from Sec. 4.3.2, that our strategy is trained to reach the target quality q . For each dataset, we take q to be 98% of the maximum quality of the classifier trained on 100 randomly drawn datapoints, which is the maximum number of annotations we allow. We allow for a slight decrease in performance (98% instead of 100%) because AL learning curves usually flatten and our choice enables AL agents to reach the desired quality much quicker during the episode.

We use the *same* RL parameters in all the experiments. The RL procedure starts with 100 “warm start” episodes with random actions and 100 Q-function updates. While learning an RL policy, the Adam optimizer is used with learning rate 0.0001 and a batch size 32. To force exploration during the course of learning, we use ϵ -greedy policy π , which means that with probability $1 - \epsilon$ the action $a_t = \arg \max_a Q_\theta^\pi(s_t, a)$ is performed and with probability ϵ a random one is. The parameter ϵ decays from 1 to 0 in 1000 training iterations. We incorporate the following techniques: 1) separate target network [129] to deal with non-stationary targets (update rate 0.01), 2) replay buffer [129] (of size 10 000) to avoid correlated updates of neural network, 3) prioritized replay [159] to use the experience from the replay buffer with the highest temporal-difference errors more often (the exponent parameter is 3).

LAL baselines The baselines **LAL-iter** and **LAL-ind** are not *flexible* as they were originally designed to deal with Random Forest classifiers. In order we use them within our experimental setup with LogReg, we let them train 2 classifiers in parallel and use the hand-crafted by [90] features of RF in AL policy.

4.4.2 Transferability

We tested the transferability of **LAL-RL** on 10 widely-used standard benchmark datasets from the UCI repository [44]: 0-*adult*, 1-*australian*, 2-*breast cancer*, 3-*diabetes*, 4-*flare solar*, 5-*heart*, 6-*german*, 7-*mushrooms*, 8-*waveform*, 9-*wdbc*. We use LogReg and ran 500 trials where AL episodes run up to 100 iterations.

In Table 4.1 we report the average number of annotations required to achieve the desired target accuracy using either our method or the baselines, and in Table 4.2 we show the standard deviation of the results. In the 9 columns marked as *leave-one-out*, we test out

Chapter 4. Towards Data-Driven General-Purpose Active Learning

Scenario Dataset	test	leave-one-out								
	0	1	2	3	4	5	6	7	8	9
Rs	<i>50.78</i>	25.31	25.65	30.33	15.57	44.83	20.80	42.81	45.28	19.36
Us	<u>41.83</u>	13.53	27.07	<i>27.84</i>	15.50	<u>37.1</u>	15.60	15.6	<u>23.83</u>	7.25
QUIRE	58.33	30.02	33.33	37.12	9.02	57.58	20.30	42.9	<i>36.49</i>	15.45
ALBE	55.66	29.79	31.84	33.62	<u>10.91</u>	50.71	21.02	39.12	41.23	16.16
LAL-ind	59.39	<i>20.88</i>	<u>20.85</u>	26.63	15.31	44.14	<i>18.16</i>	<i>24.15</i>	39.13	<i>11.22</i>
LAL-iter	63.29	<i>20.24</i>	<u>21.79</u>	28.03	14.84	<i>40.38</i>	19.90	25.2	36.97	<i>10.39</i>
LAL-RL	37.52	14.15	18.79	26.77	<i>14.67</i>	32.16	15.06	<u>21.94</u>	20.91	7.09
notransf	—	15.01	16.14	24.40	—	23.26	14.65	16.47	18.06	7.14

Table 4.1 – Average number of annotations required to reach a predefined quality level.

Scenario Dataset	test	leave-one-out								
	0	1	2	3	4	5	6	7	8	9
Rs	23.69	22.90	24.40	24.25	12.47	26.73	23.43	22.13	24.93	16.70
Us	22.69	12.70	26.74	21.74	13.16	24.53	16.93	10.73	14.04	4.01
QUIRE	21.91	19.17	25.97	22.73	9.03	21.60	19.01	12.29	22.48	14.39
ALBE	22.79	21.96	25.03	23.74	10.01	25.48	21.86	14.73	23.49	14.50
LAL-ind	59.39	18.61	20.14	19.76	12.69	24.88	19.31	10.37	14.38	6.77
LAL-iter	63.29	15.62	19.61	20.99	11.82	22.72	21.20	11.29	14.38	6.29
LAL-RL	20.05	12.40	21.84	22.01	13.61	23.10	16.89	12.74	19.30	3.93
notransf	—	14.69	21.00	19.18	—	17.20	14.75	10.89	12.37	3.81

Table 4.2 – Standard deviation of number of annotations required to reach a predefined quality level.

method using a leave-one-out procedure, that is, training on 8 of the datasets selected from number 1 to number 9, and evaluating on the remaining one. In the course of this procedure, we never use dataset 0-*adult* for training purposes. Instead, we show in the column labelled as *test* the average number of annotations needed by all 9 strategies learnt in the leave-one-out procedure (the standard deviation is 2.34). In each column, the best number appears in **bold**, the second is underlined, and the third is printed in *italics*. We consider a difference of less than 1 to be insignificant and the corresponding methods to be *ex-aequo*.

LAL-RL comes out on top in 8 cases out of 10, second and third in the two remaining cases. As it has been noticed in the literature, **Us** is good in a wide range of problems [90, 136]. In our experiments as well, it comes second overall and, for the same level of performance, it saves 29.80% over **Rs** while **LAL-RL**, saves 34.71%. Table 4.3 reports similar results reaching 98% of the quality of a classifier trained with 200 and 500 random datapoints instead.

Unfortunately, we cannot compare **LAL-RL** to **MLP-GAL(Te)** in the same fashion for lack of publicly available code. They report results for 20 annotations, we therefore check that even if we also stop all our episodes that early, **LAL-RL** still outperforms

Baseline	Rs	Us	LAL-RL
LogReg-100	32.07	-28.80%	- 34.71%
LogReg-200	80.06	-29.61%	- 39.96%
LogReg-500	51.59	-31.49%	- 37.75%
SVM	30.87	-7.81%	- 28.35%

Table 4.3 – Increasing the number of annotations still using logistic regression (first three rows) and using SVM instead of logistic regression as the base classifier (fourth row). We report the average number of annotations required using **Rs** and the percentage saved by either **Us** or **LAL-RL**.

the strongest baseline **Us** in 90% of cases whereas **MLP-GAL(Te)** does so in 71% of the cases. Besides, we learn a policy using 5 times less data: 10 000 AL episodes instead of 50 000.

The individual durations of the episodes of 9 learned strategies **LAL-RL** on dataset *0-adult* are 38.80, 37.72, 36.74, 33.95, 34.58, 38.76, 37.46, 41.84, and 37.85. Fig. 4.3 shows the learning curves for all the baselines and for the 9 strategies. Some variability is present, but in 8 out of 9 cases **LAL-RL** outperforms all others baseline and once it shares the first rank with **Us** in terms of average episode duration. Fig. 4.4 shows additional learning curves that depict the performance of our baselines.

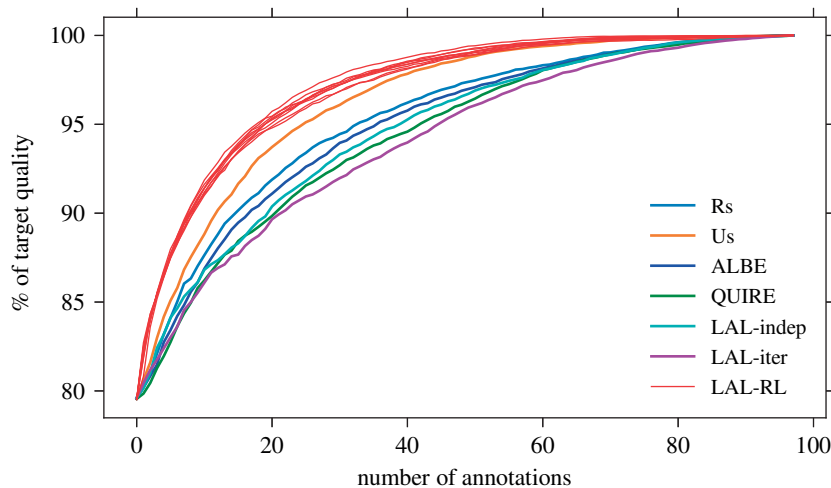


Figure 4.3 – Performance of all the strategies on *0-adult* dataset.

Transfer between related datasets Next, we study the transfer between related dataset as more traditional data-drive AL strategies do. We study two datasets: *digits-mnist* and *fashion-mnist* where the tasks are to distinguish between 10 hand-written digits or 10 images of pieces of cloth. So, we split classes from each dataset into 2

Strategy	digits-01234	digits-56789	fashion-01234	fashion-56789
Rs	21.52	36.91	41.91	38.70
Us	9.9	17.15	15.19	13.14
QUIRE	21.09	47.24	34.03	54.59
ALBE	20.4	42.25	35.08	45.01
LAL-ind	15.97	30.69	24.93	24.69
LAL-iter	13.86	27.38	21.89	28.08
LAL-RL	9.25	16.07	14.93	13.48

Table 4.4 – Average episode number of annotations to reach a predefined quality.

groups: classes 0-4 and classes 5-9. Then, the can learnt 4 **LAL-RL** strategies: one for each group. For example, we refer to **LAL-RL** learnt on classes 0-4 of digits dataset as **RLAL-d01234**. In this case we form a collection of datasets $\{\mathcal{Z}_i\}$ by sampling all possible pairs of classes 0-4 for *mnist-digits* dataset. Then, we study *intra-class* transfer when **LAL-RL** strategy is applied to the same dataset where it is learnt, but to different classes. The *related-dataset* transfer is performed when **LAL-RL** strategy is trained on *minst-digits* and applied to *minst-fashion* and vice versa. The result of both types of transfer are presented in the Table 4.4. Again, the best result is highlighted in **bold** and the difference of less than 1 is considered insignificant.

The *intra-class* transfer of **LAL-RL** strategy is successful in all cases 4 cases with **LAL-RL** holding the first rank. The second best performing strategy **Us** holds the first rank twice. The *related-dataset* transfer is analysed in Section 4.4.4. Notice, that in all cases **LAL-RL** outperforms the **Rs** strategy by a large margin and saves half of the annotation cost. To sum up, this experiment demonstrates that **LAL-RL** can perform well in traditional *intra-class* transfer even without learning dataset-specific action and state representations.

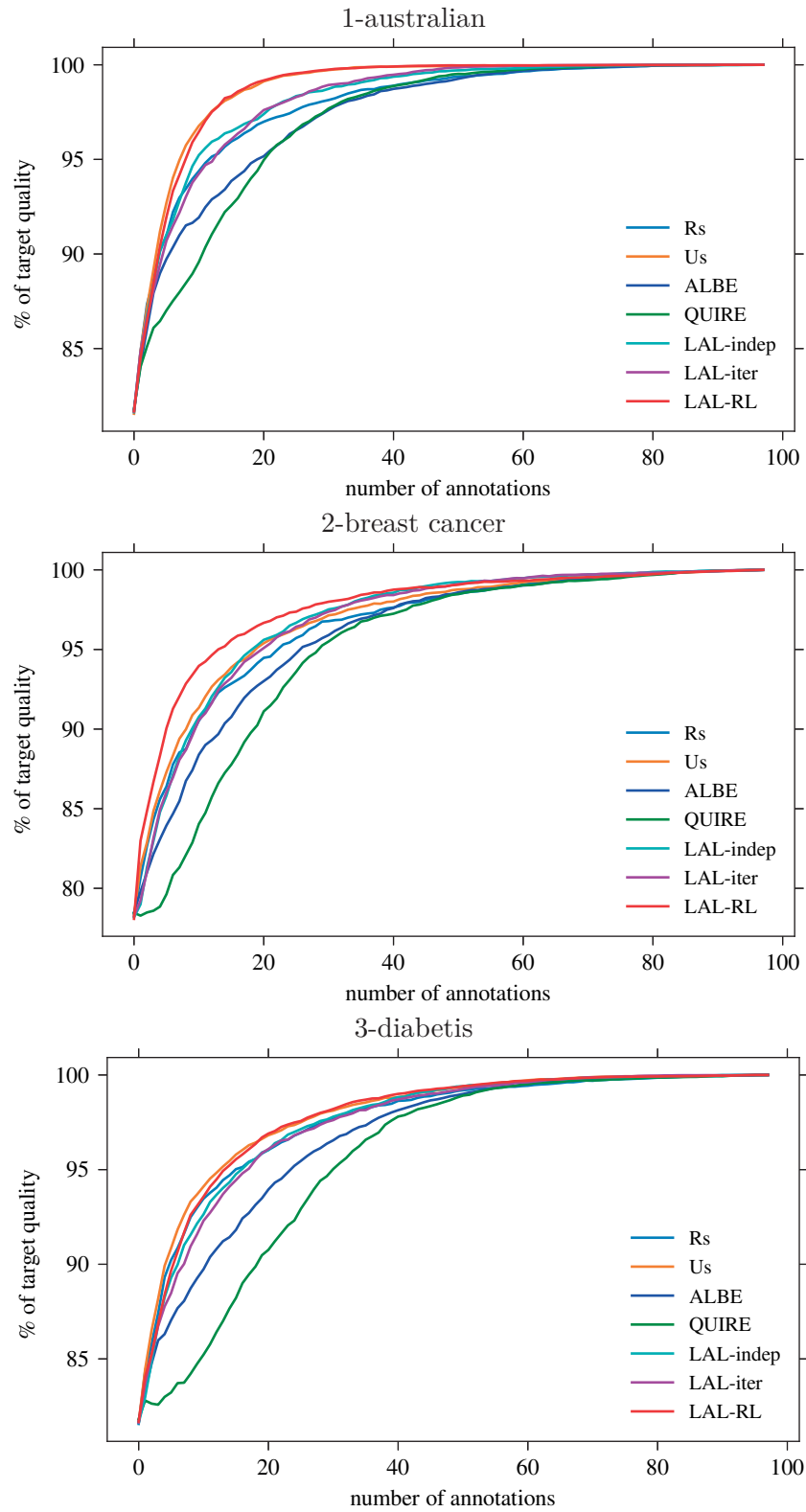


Figure 4.4 – Results of experiment from Sec. 4.4.2. Performance of all baseline strategies on 3 first datasets.

4.4.3 Flexibility

To demonstrate the flexibility of our approach now we repeat the experiments of Sec. 4.4.2 with our method, best baseline and random sampling using an SVM instead of LogReg and report the results in the last row of Table 4.3. Note that **Us** saves only 8% with respect to **Rs**, which is much less than in the experiments of Sec. 4.4.2 shown in rows 1 to 3. This stems from the fact that the sklearn implementation of SVMs relies on Platt scaling [147] to estimate probabilities, which biases the probability estimates when using limited amounts of training data. By contrast, **LAL-RL** is much less affected by this problem and delivers a 28% cost saving when being transferred across datasets.

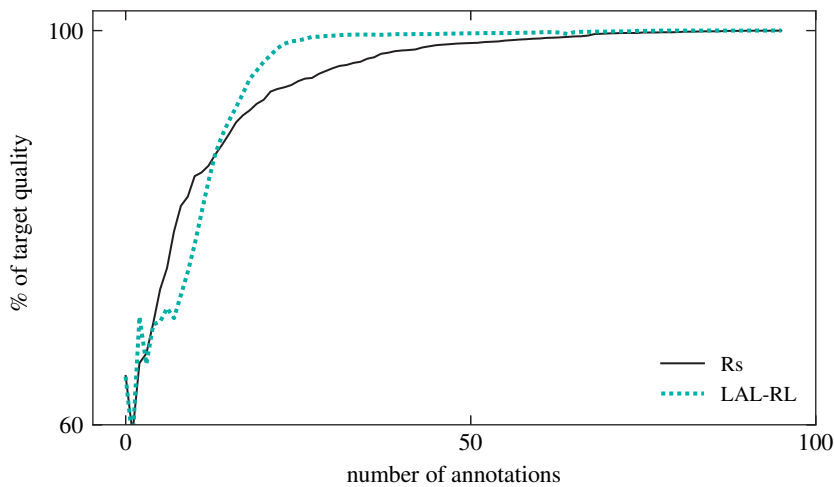


Figure 4.5 – Example of non-greedy behaviour of a learnt RL strategy

As predicted probabilities of SVM are unreliable during early AL iterations, greedy performance maximization is unlikely to result in good performance. It make this setting a perfect testbed to validate the non-myopic strategies *can* be learned by **LAL-RL**. In Fig. 4.5 we plot the percentage of the target quality reached by **Rs** and **LAL-RL** as a function of the number of annotated datapoints on one of the UCI datasets. The curve for **LAL-RL** demonstrates a non-myopic behaviour. It is worse than **Rs** at the beginning for approximately 15 iterations but almost reaches the target quality after 25 iterations, while it takes **Rs** 75 iteration to catch up.

Fig. 4.6 shows additional learning curves for the 2 methods that delivered the best performance on average in the experiments of Sec. 4.4.2 and random sampling. Note that the curve with dataset *5-flare solar* also clearly exhibits non-myopic behaviour.

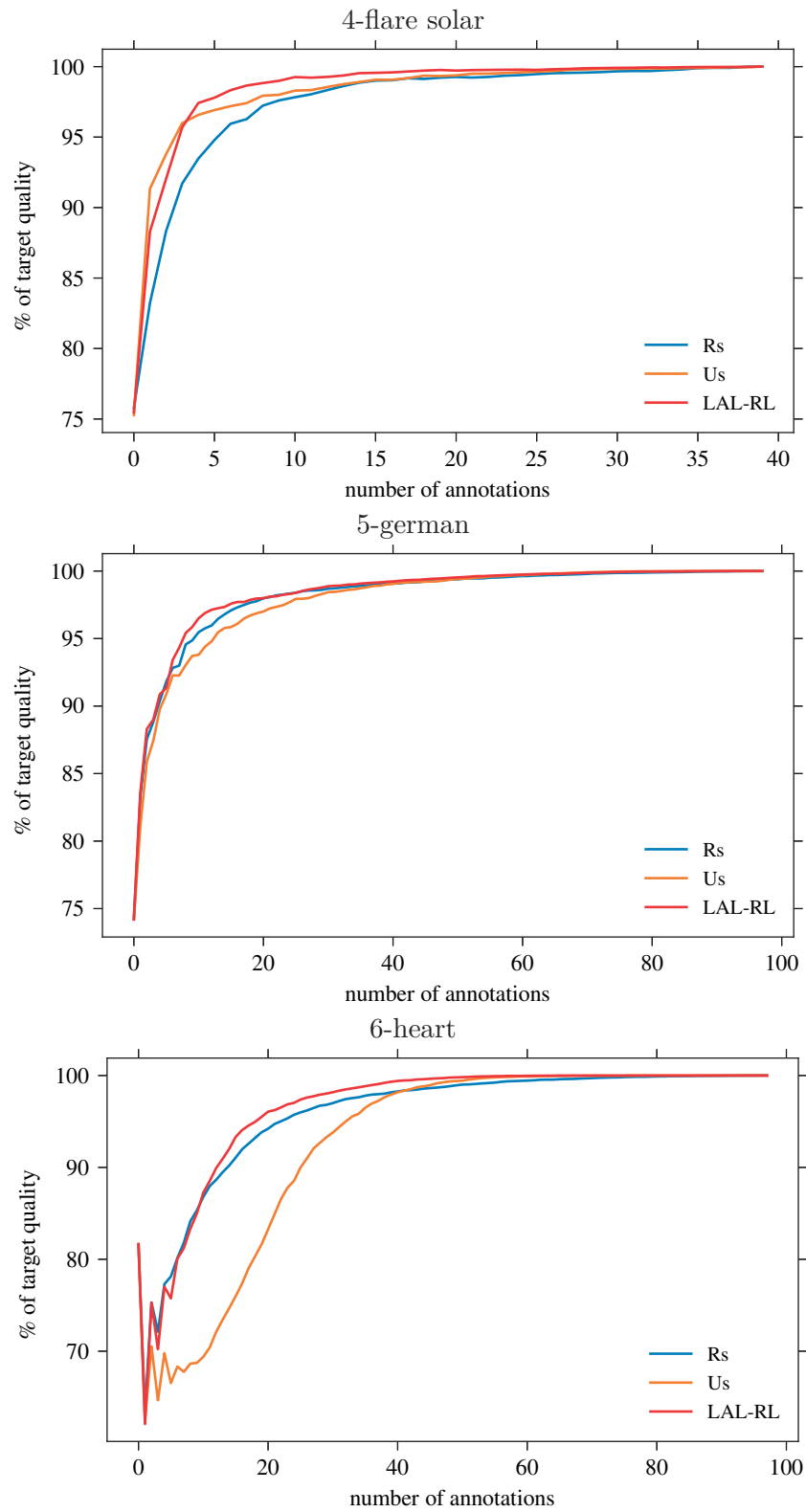


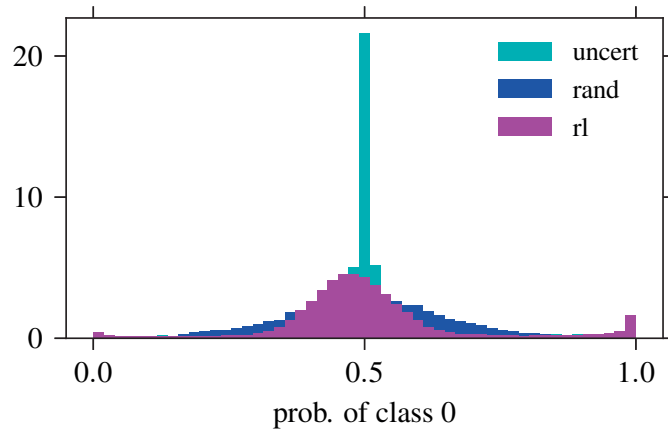
Figure 4.6 – Results of experiment from Sec. 4.4.3. Performance of 3 top strategies from experiment of Sec. 4.4.2 and a random sampling on the next 3 datasets.

4.4.4 Analysis

We now turn to analysing the behaviour of **LAL-RL** and its evolution over time. To this end, we ran additional experiments to answer the following questions.

What do we select? While performing the experiments of Sec. 4.4.2 we record $p_t = p(y^{(t)} = 0 | \mathcal{L}_t, x^{(t)})$. We show the resulting normalized histograms in Fig. 4.7(a) for **Rs**, **Us**, and **LAL-RL**. The one for **Rs** is very broad and it simply represents the distribution of available p_t in our data, while the one for **Us** is very peaky as it selects p_t closest to 0.5 by construction. Figs. 4.7 (b,c) depicts the evolution of p_t for **Rs** and **LAL-RL** for the time intervals $0 \leq t \leq 19$, $20 \leq t \leq 39$, $40 \leq t \leq 59$, $60 \leq t \leq 79$, $80 \leq t \leq 99$. The area of all histograms decreases over time as episodes terminate after reaching the target quality. However, while their shape remains roughly Gaussian in the **Rs** case, the shape changes significantly over time in case of **LAL-RL** strategy. Evidently, **LAL-RL** starts by annotating highly uncertain datapoints, then switches to uniform sampling, and finally exhibits a preference for p_t values close to 0 or 1. In other words, the **LAL-RL** demonstrates a structured behaviour.

Transfer or not? To separate the benefits of learning a strategy and the difficulties of transferring it, we introduce an artificial scenario **LAL-RL-notransfer** in which we learn on one-half of a dataset and transfer to the other half. In Table 4.1 **LAL-RL-notransfer** is better than **LAL-RL** in 3 case, much better in 2 and equal in 3 (we skip one small dataset). This shows that having access to the underlying data distribution confers a modest advantage to **LAL-RL**. Therefore, our approach still enables to learn a strategy that is competitive to having access to the underlying distribution thanks to its experience on other AL tasks. We also check how **LAL-RL-notransfer** performs on unrelated datasets, for example, learning the strategy on dataset 1 and testing it on datasets 2-9. The success rate in this case drops to around 40% on average, which again confirms the importance of using multiple datasets. As learning on one dataset to apply to another does not work well in general, we conclude that **LAL-RL** learns to distinguish between datasets to be successful across datasets.



(a)

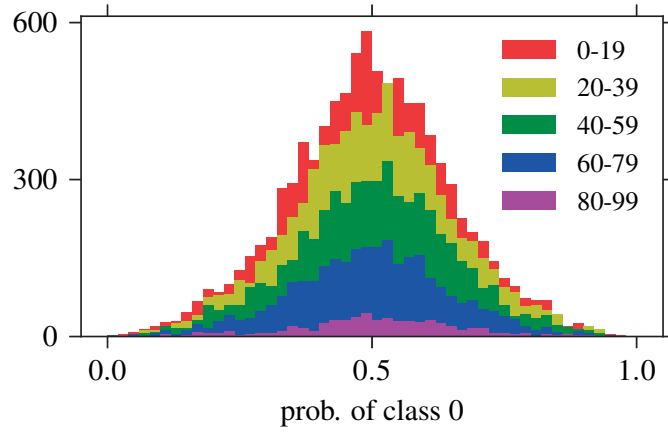
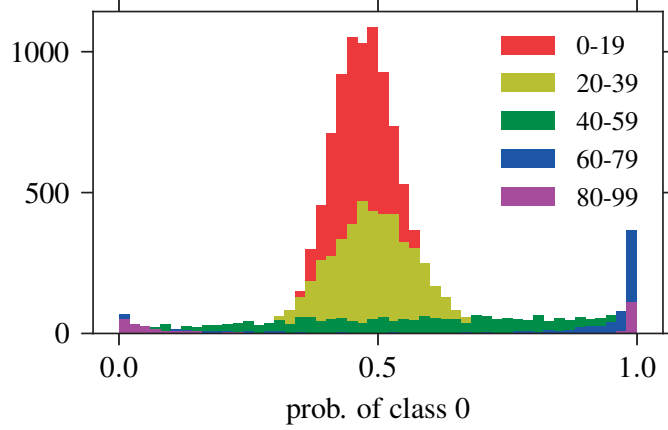
(b) **Rs**(c) **LAL-RL**

Figure 4.7 – Comparing the behavior of **Rs**, **Us** and **LAL-RL**. (a) Histogram of p_t for **Rs** in blue, **Us** in cyan, and **LAL-RL** in purple. (b) Evolution over time for random. (c) Evolution over time for **LAL-RL**.

4.5 Conclusion

In this chapter we present a data-driven approach to AL that is transferable and flexible. It can learn strategies from a collection of datasets and then successfully use them on completely unrelated data. It can also be used in conjunction with different base classifiers without having to take their specificities into account. The resulting AL strategies outperform state-of-the-art approaches. Our AL formulation is oblivious to the quality metric. In this paper, we have focused on the accuracy for binary classification tasks, but nothing in our formulation is specific to it. It should therefore be equally applicable to multi-class classification and regression problems. Thus, this new method brings us one step closer to general-purpose AL strategy.

This section concludes our study of “*What to annotate?*” question. Once again we use the data-driven approach to answer this question successfully. In the next chapter we continue exploring data-driven approaches, but we switch our attention to the question “*How to annotate?*”.

5 Learning Intelligent Dialogs for Bounding Box Annotation

5.1 Introduction

In this chapter¹ we present a method that attempts to address a question “*How to annotate?*” in the context of an important computer vision task: object detection. Many recent advances in computer vision rely on supervised machine learning techniques that are known to crave for huge amounts of training data. Object detection is no exception as state-of-the-art methods require a large number of images with annotated bounding boxes around objects. However, drawing high quality bounding boxes is expensive: The official protocol used to annotate ILSVRC [154] takes about 30 seconds per box [171]. To reduce this cost, recent works explore cheaper forms of human supervision such as image-level labels [19, 83, 207], box verification series [138], point annotations [127, 140], and eye-tracking [137].

Among these forms, the recent work on box verification series [138] stands out as it demonstrated to deliver high quality detectors at low cost. The scheme starts from a given weak detector, typically trained on image labels only, and uses it to localize objects in the images. For each image, the annotator is asked to verify whether the box produced by the algorithm covers an object tightly enough. If not, the process iterates: the algorithm proposes another box and the annotator verifies it.

The following observation is the core inspiration for our method: The success of box verification series depends on a variety of factors. For example, large objects on homogeneous backgrounds are likely to be found early in the series, and hence require little annotation time (Figure 5.1a). However, small objects in crowded scenes might require many iterations, or could even not be found at all (Figure 5.1b). Furthermore, the stronger the detector is, the more likely it is to correctly localize new objects, and to do so early in the series. Finally, the higher the desired box quality (*i.e.* how tight they should be), the lower the rate of positively verified boxes. This causes longer series, costing

¹This chapter is based on Konyushkova et al. [92]

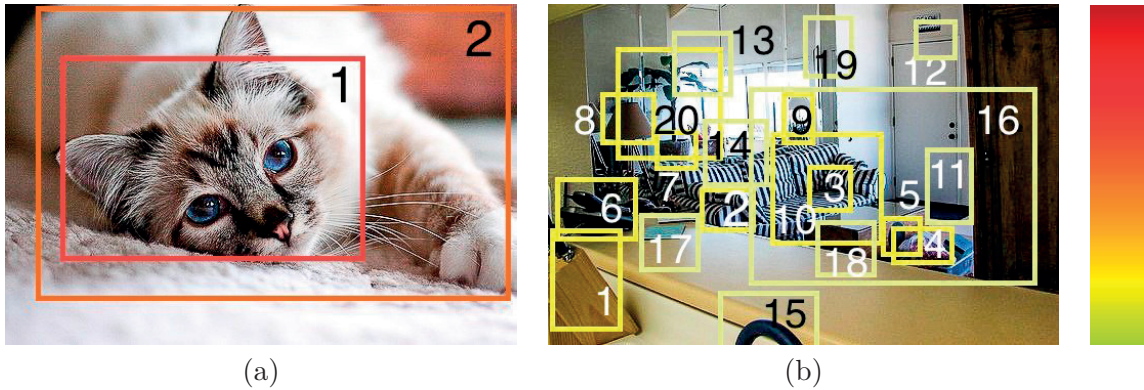


Figure 5.1 – (a) An image with a target class *cat*. The weak detector identified two box proposals with high scores. The best strategy in this case is to do a series of box verifications. (b) An image with a target class *potted plant*. The weak detector identified many box proposals with low scores. The best strategy is to draw a box.

more annotation time. Therefore, in some situations manual box drawing [139, 171] is preferable. While more expensive than one verification, it always produces a box annotation. When an annotation episode consists of many verifications, its duration can be longer than the time to draw a box, depending on the relative costs of the two actions. Thus, different forms of annotation are more efficient in different situations.

In this chapter we introduce Intelligent Annotation Dialogs (IAD) for bounding box annotation. Given an image, detector, and target class to be annotated, the aim of IAD is to automatically choose the sequence of annotation actions that results in producing a bounding box in the least amount of time. We train an IAD agent to select the type of action based on previous experience in annotating images. Our method automatically adapts to the difficulty of the image, the strength of the detector, the desired quality of the boxes, and other factors. This is achieved by modelling the episode duration as a function of problem properties. We consider two alternative ways to do this, either a) by predicting whether a proposed box will be positively or negatively verified (Section 5.4.1), or b) by directly predicting the episode duration (Section 5.4.2).

We evaluate IAD by annotating bounding boxes in the PASCAL VOC 2007 dataset [49] in several scenarios: a) with various desired quality levels; b) with detectors of varying strength; and c) with two ways to draw bounding boxes, including a recent method which only takes 7s per box [139]. In all scenarios our experiments demonstrate that thanks to its adaptive behaviour IAD speeds up box annotation compared to manual box drawing alone, or box verification series alone. Moreover, it outperforms any fixed combination of them in most scenarios. Finally, we demonstrate that IAD learns useful strategies in a complex realistic scenario where the detector is continuously improved with the growing amount of the training data.

5.2 Related work

Drawing bounding boxes Fully supervised object detectors are trained on data with manually annotated bounding boxes, which is costly. The reference box drawing interface [171] used to annotate ILSVRC [154] requires 25.5s for drawing one box. Recently, a more efficient interface reduces costs to 7.0s without compromising on quality [139]. We consider both interfaces in this paper.

Weak supervision for building object detectors Various forms of weak supervision for training object detectors are discussed in Sections 1.4.1. In this chapter we build on box verification series [138], where boxes are iteratively proposed by an object detector and verified by a human annotator. Experiments show that humans can perform box verification reliably (Figure 6 of Papadopoulos et al. [138]). Besides, the Open Images dataset [95] contains 2.5 Million boxes annotated in this manner, demonstrating it can be done at scale.

The closest work to ours proposes human-machine collaboration for bounding box annotation [155]. Given a repertoire of questions, the problem is modelled with a Markov decision process. Our work differs in several respects. (1) While Russakovsky et al. [155] optimizes the expected precision of annotations over the whole dataset, our method delivers quality guarantees on each individual box. (2) Our approach of Section 5.4.1 is mediated by predicting the probability of a box to be accepted by an annotator. Based on this, we provide a provably optimal strategy which minimizes the expected annotation time. (3) Our reinforcement learning approach of Section 5.4.2 learns a direct mapping from measurable properties to annotation time, while avoiding any explicit modelling of the task. (4) Finally, we address a scenario where the detector is iteratively updated (Section 5.5.3), as opposed to keeping it fixed.

Other works for annotating bounding boxes study box verification with the purpose of making the best prediction on a given dataset [34] or try to select images with the highest labelling cost that should help in learning a better model faster [203].

Reinforcement learning Reinforcement learning (RL) traditionally aims at learning policies that allow autonomous agents to act in interactive environments. Reinforcement learning has a long tradition e.g. in robotics [7, 128, 102]. In computer vision, it has mainly been used for active vision tasks [29, 16, 74], such as learning a policy for the spatial exploration of large images or panoramic images. Our use of RL differs from this, as we learn a policy for image annotation, not for image analysis. The learned policy enables the system to dynamically choose the annotation mechanism by which to interact with the user.

5.3 Problem definition and motivation

5.3.1 Why use intelligent annotation dialogs?

In this chapter we tackle the problem of producing bounding box annotations for a set of images with image-level labels indicating which object classes they contain. Consider annotating a *cat* in Figure 5.1 (a). The figure shows two bounding boxes found by the detector. We notice that: a) the image is relatively simple with only one distinct object; b) there are only few high-scored *cat* detections; c) they are big; d) we might know a-priori that the detector is strong for the class *cat* and thus detections for it are often correct. As box verification is much faster than drawing, the most efficient way to annotate a box in this situation is with a box verification series.

Now consider instead annotating a *potted plant* in Figure 5.1 (b). We notice that: a) the image is cluttered with many details; b) there are many low-scored *potted plant* detections; c) they are small; d) we might know a-priori that the detector is weak for this class and thus the detections for it are often wrong. In this situation, it is unlikely that the correct bounding box comes early in the series. Thus, manual box drawing is likely to be the fastest annotation strategy.

Even during annotation of one image-class pair, the best strategy may combine both annotation types: Given only one high-scored box for *cat*, the best expected strategy is to verify one box, and, if rejected, ask manual box drawing.

These examples illustrate that every image, class and detector output requires a separate treatment for designing the best annotation strategy. Thus, there is need for a method that can take advantage of this information to select the most time efficient sequence of annotation actions. In this chapter, we propose two methods to achieve this with the help of Intelligent Annotation Dialog (IAD) agents. In our first approach (Section 5.4.1) we explicitly model the expected episode duration by taking into consideration the probability for each proposed box to be accepted. Our second approach (Section 5.4.2) casts the problem in terms of reinforcement learning and learns a strategy from trial-and-error interactions without an intermediate modelling step.

5.3.2 Problem definition

We are given an image with image-level labels that indicate which object classes it contains. We treat each class independently, and we want to produce one bounding box given a single image-class pair. In particular, given that the image contains a set \mathcal{B}^* of object instances of the target class, we want to produce a bounding box \hat{b} of sufficient quality around one such object b^* . We measure the quality in terms of Intersection-over-Union (IoU) and we want to find \hat{b} such that there exists $b^* \in \mathcal{B}^* : \text{IoU}(\hat{b}, b^*) \geq \alpha$. More specifically, we want to automatically construct a sequence of actions which produces

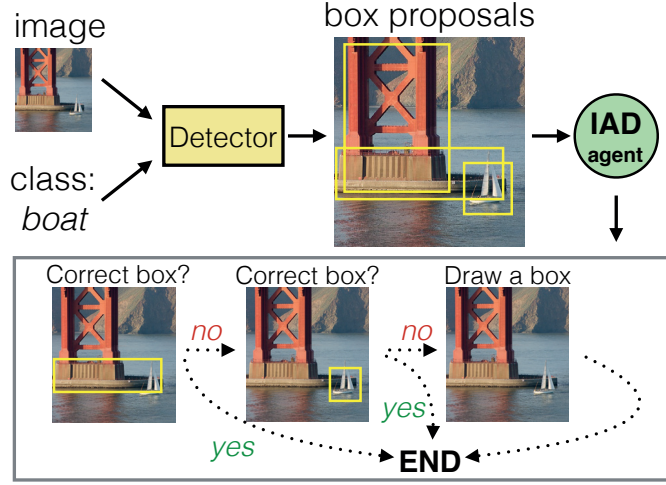


Figure 5.2 – Intelligent Annotation Dialog agent in action. For a given image and class *boat* the detector identifies a set of box proposals. IAD agent produces a planned dialog V^2D that means that the first two box proposals are verified and if none of them is accepted, manual box drawing is done. In reality, the annotation terminates after two box verifications.

\hat{b} while minimizing annotation time, choosing from two annotation actions: manual bounding box drawing D [139, 171] that takes t_D seconds and bounding box verification V [138] that takes t_V seconds.

We design the annotation dialog to end with a successfully annotated bounding box. Logically, the only possible planned sequence of actions which does this has the form V^mD . No sequence of verification V is guaranteed to produce a bounding box, so if m verifications fail to produce one, manual drawing is required. Conversely, manual drawing always produces a box and the dialog ends. Figure 5.2 illustrates how IAD agent produces a planned sequence of V^2D for the task of detecting a *boat* in the image with several detections. In reality, only a sequence of actions V^2 is executed because a boat is found at the second verification.

Verification questions are generated using an object detector. Papadopoulos et al. [138] present the highest scored detection to the annotator. Upon rejection, they remove boxes which highly overlap with the rejected one (this procedure is called *search space reduction*), after which they present the next box with the highest score. In this paper we assume the detector stays constant during a single annotation dialog, which means we can do search space reduction by non-maximum suppression (NMS). Let us denote by B_0 the sequence of detections followed by NMS. Because of NMS, we can assume boxes in B_0 to be independent for verification. Let \mathcal{S} be the set of all possible sequences of distinct elements in B_0 . Now our goal is to plan a sequence of actions $\pi = V^mD$ on a sequence $S_m = (s_1, \dots, s_m) \in \mathcal{S}$.

We can now formally define the optimization criterion for the IAD agent. Let $t(V^m D, S_m)$ be the duration of the episode when strategy $V^m D$ is applied to a sequence S_m and let us denote its expected duration as $T(V^m D, S_m)$. The task of IAD is to choose (1) the maximum number of verifications $m = k$ that will define a sequence of actions $V^k D$, and (2) a sequence of boxes $A_k = (a_1, \dots, a_k) \in \mathcal{S}$ such that the duration of the episode is minimized in expectation:

$$\begin{aligned} T(V^k D, A_k) &\leq T(V^m D, S_m), \\ m &\in \{0, \dots, n\}, \forall S_m \in \mathcal{S}. \end{aligned} \tag{5.1}$$

5.4 Methods

We now present our two methods to construct Interactive Annotation Dialogs (IAD).

5.4.1 IAD by predicting probability of acceptance

One way to minimize the expected duration of the episode is by estimating the probability that the proposed boxes will be accepted by the annotator. We can train a classifier g that will predict if the box $b_i \in B_0$ is going to be accepted or not as a function of various parameters of the state of the episode. By looking at the probability of acceptance $p(b_i)$ for every box, we can compute the expected duration of the episode $T(V^m D, S_m)$ for any $V^m D$ and S_m . Given this acceptance probability estimation, we show that there exists a simple decision rule that chooses m and S_m so as to minimize the expected episode duration.

Optimal strategy Suppose for now that we know the probabilities $p(b_i)$ for every box b_i to be accepted at a quality level α :

$$p(b_i) = \mathbb{P}[\max_{b^* \in B^*} \{\text{IoU}(b_i, b^*)\} \geq \alpha]. \tag{5.2}$$

Later in this section we will explain how to estimate $p(b_i)$ in practice.

Imagine for a moment that we have only one box proposal b_1 . In this case the only two possible sequences of actions are D and $V^1 D$. Let us compute the expected time until the end of the episode for both of them. The episode duration for strategy D is just the time required for manual drawing: $T(D) = t_D$.

For the second strategy $V^1 D$, the end of the episode is reached with probability $p(b_1)$ when a box proposal is accepted and with probability $q(b_1) = 1 - p(b_1)$ when manual drawing is done. Hence, the expected duration of the episode is

$$T(V^1 D, (b_1)) = t_V + q(b_1)t_D. \tag{5.3}$$

As we want to choose the strategy with the lowest expected duration of the episode, D is preferred to V^1D if $T(D) \leq T(V^1D, (b_1))$, *i.e.*

$$t_D \leq t_V + q(b_1)t_D \iff p(b_1) \leq t_V/t_D. \quad (5.4)$$

Now let us go back to a situation with a sequence of box proposals B_0 . We sort B_0 in the order of decreasing probability of acceptance $p(b_i)$, resulting in a sequence of boxes \bar{S}_n . Consider the following strategy (Algorithm IAD-Prob): Verify boxes from \bar{S}_n for which $p(b_i) > t_V/t_D$; if none of them is accepted, then do manual box drawing. We claim that the strategy produced by IAD-Prob is optimal, *i.e.* it minimizes the expected duration of the episode.

Algorithm IAD-Prob

- 1: **Input:** $B_0 = (b_1, \dots, b_n); p(b_1), \dots, p(b_n); t_V; t_D$
 - 2: $\bar{S}_n = (\bar{s}_1, \bar{s}_2, \dots, \bar{s}_n) \leftarrow \text{sort}(B_0)$ by $p(b_i)$
 - 3: $\pi = ()$
 - 4: $A_k = ()$
 - 5: **while** $p(\bar{s}_i) > t_V/t_D$ **do**
 - 6: $A_k \leftarrow A_k \cup \bar{s}_i$
 - 7: $\pi \leftarrow V^k D$
 - 8: **return** sequence of actions π , sequence of boxes A_k
-

Theorem 1. *If probabilities of acceptance $\{p(b_i)\}$ are known, the strategy of applying a sequence of actions $V^k D$ defined by IAD-Prob to a sequence of boxes A_k minimizes the annotation time, *i.e.* for all $m \in \{0, \dots, n\}$ and for all box sequences S_m :*

$$T(V^k D, A_k) \leq T(V^m D, S_m) \quad (5.5)$$

Sketch of the proof. The proof consists of two parts. First, we show that for any strategy $V^m D$, the best box sequence is obtained by sorting the available boxes by their probability of acceptance and using the first m of them. Second, we show that the number of verification steps found by IAD-Prob, k , is indeed the optimal one.

We start by rewriting the expected episode length in closed form. For a strategy $V^m D$ and any sequence of boxes, $S_m = (s_1, \dots, s_m)$, we obtain

$$\begin{aligned} T(V^m D, S_m) &= t_V + q(s_1)t_V + q(s_1)q(s_2)t_V + \dots \\ &\quad + q(s_1)q(s_2) \cdots q(s_{m-1})t_V + q(s_1)q(s_2) \cdots q(s_m)t_D \\ &= t_V \sum_{l=0}^{m-1} \prod_{j=1}^l q(s_j) + t_D \prod_{j=1}^m q(s_j). \end{aligned} \quad (5.6)$$

Our first observation is that Equation (5.6) is monotonically decreasing as a function of $q(s_1), \dots, q(s_m)$. Consequently, the smallest value is obtained by selecting the set of m

boxes that have the smallest rejection probabilities. To prove that their optimal order is sorted in decreasing order, assume that S_m is not sorted, i.e. there exists an index $l \in \{1, \dots, m-1\}$ for which $q(s_l) > q(s_{l+1})$. We compare the expected episode length of S_m to that of a sequence \tilde{S}_m in which s_l and s_{l+1} are at switched positions. Using Equation (5.6) and noticing that many of the terms cancel out, we obtain

$$\begin{aligned} & T(V^m D, S_m) - T(V^m D, \tilde{S}_m) \\ &= t_V(q(s_l) - q(s_{l+1})) \left(\prod_{j=1}^{l-1} q_j \right) > 0. \end{aligned} \quad (5.7)$$

This shows that \tilde{S}_m has strictly smaller expected episode length than S_m , so S_m cannot have been the optimal order.

Consequently, for any strategy $V^m D$, the optimal sequence is to sort the boxes by decreasing probability of rejection, *i.e.* increasing acceptance probability. We denote it by $\bar{S}_m = (\bar{s}_1, \dots, \bar{s}_m)$.

Next, we show that the number, k , of verification actions found by the IAD-Prob algorithm is optimal, *i.e.* $V^k D$ is better or equal to $V^m D$ for any $m \neq k$. As we already know that the optimal box sequence for any strategy $V^m D$ is \bar{S}_m , it is enough to show that

$$T(V^{m-1} D, \bar{S}_{m-1}) \geq T(V^m D, \bar{S}_m), \quad (5.8)$$

for all $m \in \{1, \dots, k\}$, and

$$T(V^{m-1} D, \bar{S}_{m-1}) \leq T(V^m D, \bar{S}_m). \quad (5.9)$$

for all $m \in \{k+1, \dots, n\}$. To prove these inequalities, we again make use of expression Equation (5.6). For any $m \in \{1, \dots, n-1\}$ we obtain

$$\begin{aligned} & T(V^m D, \bar{S}_m) - T(V^{m-1} D, \bar{S}_{m-1}) \\ &= t_V \prod_{j=1}^{m-1} q(\bar{s}_j) + t_D \prod_{j=1}^m q(\bar{s}_j) - t_D \prod_{j=1}^{m-1} q(\bar{s}_j) \\ &= \left(\prod_{j=1}^{m-1} q(\bar{s}_j) \right) (t_V + q(\bar{s}_m)t_D - t_D). \end{aligned} \quad (5.10)$$

For $m \in \{1, \dots, k\}$, we know that $p(\bar{s}_m) > t_V/t_D$ by construction of the strategy. As in Equation (5.4), this is equivalent to $t_V + q(\bar{s}_m)t_D - t_D \geq 0$. Consequently, Equation (5.10) is non-negative in this case, and inequality of Equation (5.8) is confirmed. For $m \in \{k+1, \dots, n\}$, we know $p(\bar{s}_m) \leq t_V/t_D$, again by construction. Consequently, $t_V + q(\bar{s}_m)t_D - t_D \leq 0$, which shows that Equation (5.10) is nonpositive in this case, confirming Equation (5.9). \square

Predicting acceptance probability To follow the optimal strategy IAD-Prob, we need the probabilities of acceptance $\{p(b_i)\}$ which we estimate using a classifier g . To obtain these probabilities we start with a (small) set Z_0 of annotated bounding boxes on a set of images I_0 . We apply a detector f_0 on I_0 to obtain a set of detections B_0 . Afterwards, we generate a feature vector ϕ_i for every box $b_i \in B_0$. The exact features are specified in Section 5.5.1 and include measurements such as detector scores, entropy, and box-size.

Next, we simulate verification responses for box proposals B_0 of every image-class pair with known ground truth. A box b_i gets label $y_i = 1$ if its IoU with any of the ground truth boxes is great or equal to α , otherwise it gets label 0. This procedure results in feature-label pairs (ϕ_i, y_i) that serve as a dataset for training a probabilistic classifier g .

Intuitively, the classifier learns that, for example, boxes with high detector’s score are more likely to be accepted than boxes with low detector’s score, bounding boxes for class *cat* are more likely to be accepted than bounding boxes for class *potted plant*, and smaller bounding boxes are less likely to be accepted than big ones.

5.4.2 IAD by reinforcement learning

The problem of finding a sequence of actions to produce a box annotation can be naturally formulated as a reinforcement learning problem. This approach allows us to learn a strategy directly from trial-and-error experience and to avoid the explicit modelling of Section 5.4.1. To construct an optimal strategy it does not need any prior knowledge about the environment. Thus, it is easily extensible to other types of actions or to stochastic environments with variable response time by an annotator.

Suppose that bounding boxes in an episode are verified in order of decreasing detector’s score given by B_0 . In an *episode* of annotating one image for a given target class, the IAD agent interacts with the *environment* in the form of the annotator. A *state* s_τ is characterised by the properties of a current image, detector and a current box proposal (as ϕ_i in Section 5.4.1). In each state the agent has a choice of two possible *actions* a : 1) ask for verification of the current box ($a = V$) and 2) ask for a manual drawing ($a = D$) The *reward* at every step τ is the negative time required for the chosen action: $r_\tau = -t_V$ and $r_\tau = -t_D$. If a box is positively verified or manually drawn, the episode terminates with a reward 0. Otherwise the agent finds itself in the next state corresponding to the next highest-scored box proposal in B_0 . The total *return* of the episode is the sum of rewards over all steps. Denoting the number of steps after which an episodes terminates by K , the return is $R = \sum_{\tau=1}^K r_\tau$. This is equal to $-(K-1)t_V - t_D$ if the episode finished with manual drawing, or $-Kt_V$ if it finished with box acceptance. By trying to maximise the return R , the agent learns a policy π that minimises the total annotation time. This

results in a strategy that consists of a sequence of actions π applied to a sequence of boxes B_0 .

Training the agent The agent can learn the optimal policy π from trial and error interactions with the environment. As in Section 5.4.1, we train on a small subset of annotated bounding boxes Z_0 . We learn a policy with *Q-learning* which learns to approximate *Q-function* $Q^\pi(a, s_\tau)$ that indicates what return the agent should expect at state s_τ after taking an action a and after that following a strategy π .

5.5 Experiments

5.5.1 Experimental setup

The code for our method is made publicly available². We evaluate the performance of the IAD approach on the task of annotating bounding boxes on the PASCAL VOC 2007 trainval dataset. In all experiments our detector is Faster-RCNN [150] using Inception-ResNet [175] as base network.

Annotator actions and timings We simulate the annotator based on the ground truth bounding boxes. When asked for verification, a simulated human annotator deterministically accepts a box proposal if $\text{IoU} \geq \alpha$ and it takes $t_V = 1.8$ seconds [138]. When the simulated annotator is asked to draw a box, we use the ground truth box. We consider two interfaces for drawing: the classical manual drawing M [171] and the new faster Extreme Clicking X [139]. We consider that it takes a simulated user $t_M = 25.5$ or $t_X = 7$ seconds to return a bounding box that corresponds to any of the objects b^* [171, 139].

Box proposal order The order of box proposals for verifications is set to be B_0 , i.e. in decreasing order of detector’s score (Section 5.3.2). Then, the optimality condition of strategy IAD-Prob assumes that a box with higher score is more likely to be accepted than a box with lower score. Empirically, we observe only rare cases when this assumption is violated, but even then, changing the order does not improve results. Thus, we keep the original order B_0 for computational efficiency and consistency with IAD-RL. The images come in the same fixed random order for all methods.

Box features When predicting the acceptance probability (Section 5.4.1) and during reinforcement learning (Section 5.4.2), we use the following features ϕ_i characterizing box b_i , image, detector, and target class:

- a) prediction score of the detector on the box: $d(b_i)$;

²https://github.com/google/intelligent_annotation_dialogs

- b) relative size of the box b_i in the image;
- c) average prediction score of all box proposals for the target class;
- d) difference between c) and $d(b_i)$;
- e) difference between the maximum score for the target class among all box proposals and $d(b_i)$;
- f) one-hot encoding of class.

IAD-Prob To predict box acceptance probabilities, we use a neural network classifier with 2 to 5 layers containing 5 to 50 neurons in each layer for predicting the acceptance of a box and these parameters are chosen in cross-validation (Section 5.4.1). We experimented with other types of classifiers including logistic regression and random forest and did not find any significant difference in their performance.

IAD-RL We learn a policy for the reinforcement learning agent with a method similar to [129] (Section 5.4.2). The function approximation of Q-values is a fully-connected neural network with 2 layers and 30 neurons at every layer. We learn it from interactions with simulated environment using experience replay. We use exploration rate $\epsilon = 0.2$, mini-batches of size 64 and between 500 and 1000 training iterations. A subset of training samples is reserved for validation: we use it for choosing parameter of neural network and for early stopping.

5.5.2 IAD with a fixed detector

Scenarios We evaluate our methods in several scenarios, by varying the following properties of the problem: a) the desired quality of boxes, b) the strength of the detector, and c) which interface is used to draw a box. Intuitively, different properties tend to prioritize different actions V or D . The higher the desired quality is, the more frequently manual box drawing is needed. When the detector is strong, box verification is successful more often and is preferred to drawing due to its small cost. Finally, using the fast Extreme Clicking interface, manual drawing is cheaper and becomes more attractive. Specifically, we consider the following three configurations, each for both quality levels:

1. Weak detector, slow drawing, varying quality

Classical, slow interface to draw boxes [171] with a weak detector. To train the detector (Section 5.5.1), we first produce bounding box estimates using standard Multiple Instance Learning (MIL, *e.g.* [20, 36, 170]). The first two columns of Table 5.1 report the average time per one annotation episode.

2. Weak detector, fast drawing, varying quality

The fast Extreme Clicking [139] for drawing boxes. We report the results in columns 3 and 4 of Table 5.1.

Drawing Detector Quality level	Slow drawing		Fast drawing			
	Weak detector		Weak detector		Strong detector	
	high	low	high	low	high	low
D (standard)	25.50 ± 0.00	25.50 ± 0.00	7.00 ± 0.00	7.00 ± 0.00	7.00 ± 0.00	7.00 ± 0.00
V^1D	23.01 ± 0.07	17.30 ± 0.07	7.62 ± 0.02	6.05 ± 0.02	3.45 ± 0.01	2.50 ± 0.01
V^2D	23.79 ± 0.06	16.67 ± 0.06	8.92 ± 0.02	6.67 ± 0.02	3.48 ± 0.01	2.45 ± 0.01
V^3D	24.67 ± 0.07	16.38 ± 0.07	10.21 ± 0.02	7.32 ± 0.03	3.65 ± 0.02	2.48 ± 0.01
V^*D (standard)	42.29 ± 0.07	17.37 ± 0.07	31.82 ± 0.11	11.46 ± 0.04	8.83 ± 0.09	3.18 ± 0.02
IAD-Prob	23.07 ± 0.23	12.64 ± 1.29	6.81 ± 0.02	5.86 ± 0.04	3.42 ± 0.18	2.73 ± 0.08
IAD-RL	23.62 ± 0.38	16.30 ± 0.09	6.83 ± 0.03	5.89 ± 0.05	3.60 ± 0.07	2.66 ± 0.06
lower bound	18.55 ± 0.05	10.23 ± 0.04	5.99 ± 0.01	4.66 ± 0.01	2.80 ± 0.01	2.19 ± 0.01

Table 5.1 – Average episode duration for *standard*, *fixed* and IAD strategies in scenarios varying in drawing speed, strength of detector and quality level. **Best fixed strategy** results are highlighted in bold. The **best result** of each scenario is indicated in yellow (multiple highlights if very close). The two IAD agents do approximately equally well.

3. Strong detector, fast drawing, varying quality

In many situations we have access to a reasonably strong detector before starting annotation of a new dataset. To model this we train f_0 on the PASCAL 2012 dataset train set which contains 16k boxes. The results are presented in the last two columns of Table 5.1.

Dataset We use PASCAL 2007 trainval [49], where we assume that image-level annotations are available for all images, whereas bounding boxes are given only in a small subset of images Z_0 . The task is to annotate the rest of the images Z' with bounding boxes. Z and Z' are set with 10-fold validation and the reported results are averages over them.

Standard strategies As baselines, we consider two standard annotation strategies. The first is to always do manual drawing (D). The second is to run box verification series, followed by drawing if all available boxes have been rejected (V^*D). This strategy is guaranteed to terminate successfully while being the closest to [138].

Fixed strategies We introduce a family of fixed strategies that combine the two actions V and D in a predefined manner, without adapting to a particular image, class and detector: V^1D , V^2D , and V^3D .

Lower bound We also report the lower bound on the duration of the annotation episode. If we knew which box (Section 5.3.2) in the proposal sequence B_0 is the first that will be accepted, we could choose a sequence of actions that leads to the lowest annotation cost. If accepted box is at the position k^* in sequence B_0 , then the strategy is the following. If the cost of k^* verifications is lower than the cost of a drawing, then

the verification series is done, otherwise, drawing is done. Note how this lower bound requires knowing the ground-truth bounding box. So, it is only intended to reveal the limits of what can be achieved by the type of strategies that we explore.

Results Table 5.1 shows that the scenario settings indeed influence the choice between V and D , along three dimensions: a) When annotations of higher quality are required, the best fixed strategy does fewer verifications, i.e. it resorts to manual drawing earlier in the series than when lower quality is acceptable (columns 1 vs. 2, 3 vs. 4, 5 vs. 6). b) When the detector is strong (columns 5 and 6), the best fixed strategy does more box verifications than with a weak detector (columns 3 and 4). c) When manual drawing is fast (columns 3 and 4), the best fixed strategy tends to do fewer box verifications than when drawing is slow (columns 1 and 2). The gap to the lower bound indicates how hard each of the scenarios is.

Importantly, both of our IAD strategies outperform any standard strategy in all scenarios. Moreover, IAD-Prob is significantly better than the best fixed strategy in three scenarios, equal in two, and worse in one. No single fixed strategy works well in *all* scenarios, and finding the best fixed strategy requires manual experimentation. In contrast, IAD offers a principled way to automatically construct an adaptive strategy that works well in *all* problem settings. Indeed, the consistent competitive performance of IAD demonstrates that it learns to adapt to the scenario at hand.

5.5.3 IAD with an iteratively improving detector

In realistic settings, the detector becomes stronger with a growing amount of annotations. Thus, to annotate bounding boxes with minimal cost, the object detector should be iteratively re-trained on previously annotated data.

Horizontal re-training One way to introduce detector re-training is suggested by the box verification series technique [138]. It starts with a given object detector f_0 , typically trained on image-level labels using MIL. In the first iteration, f_0 is applied to all images, and the highest scored detection b_1 in each image is sent for human verification. After this, the detector is re-trained on all accepted boxes, giving a new detector f_1 . In the second iteration, f_1 is applied to all images where a proposed box was rejected, attempting to localize the objects again as f_1 is stronger than f_0 (re-localization phase). Afterwards, these new detections are sent for verification, and finally the detector is re-trained again. The re-training, re-localization, and verification phases are iteratively alternated for a predefined number of iterations. We refer to this method as *V-hor* in our experiments. It essentially corresponds to the original method of Papadopoulos et al. [138].

Vertical re-training A different way to incorporate detector re-training is inspired by batch-mode active learning [164]. In this case, a subset of images I_1 (batch) is annotated

until completion, by running box verification series in each image while keeping the initial detector f_0 fixed. After this, the detector is re-trained on all boxes produced so far, giving f_1 , and is then applied to the next batch I_2 to generate box proposals. The process iteratively moves from batch to batch until all images are processed.

IAD with vertical re-training It is straightforward to apply vertical retraining to any fixed dialog strategy. However, re-training the detector on more data increases the advantage of V over D , so a truly adaptive strategy should change as the detector gets stronger. We achieve this with the following procedure. At any given iteration τ , we train dialog strategy IAD-Prob($I_\tau, f_{\tau-1}$) using boxes collected on I_τ and detector $f_{\tau-1}$. IAD-Prob($I_\tau, f_{\tau-1}$) is applied with detector f_τ to collect new boxes on the next batch $I_{\tau+1}$. Note that IAD-Prob($I_\tau, f_{\tau-1}$) is trained with the help of detector $f_{\tau-1}$, but it is applied with the box proposals of detector f_τ . This procedure introduces a small discrepancy, but it is not important when detectors f_τ and $f_{\tau-1}$ are sufficiently similar, which is the case in the experiments below. To initialize the procedure we set f_0 to be a weakly supervised MIL detector and we annotate I_1 by manual box drawing D .

We set the desired quality of bounding boxes to high (*i.e.* $\alpha = 0.7$) and we use Extreme Clicking for manual drawing. We perform 6 re-training iterations with an increasingly large batch size: $|I_1| = 3.125\%$, $|I_2| = 3.125\%$, $|I_3| = 6.25\%$, $|I_4| = 12.5\%$, $|I_5| = 25\%$, $|I_6| = 50\%$. This batching schedule is motivated by the fact that the gain in detector’s performance after re-training is more noticeable when the previous training set is considerably smaller.

Results Figure 5.3 (a) shows what proportion of boxes is collected as a function of total annotation time. We compare IAD-Prob against the strategy V -hor [138], and the standard fast drawing strategy X . IAD-Prob is able to annotate the whole dataset faster than any of the considered strategies. Figure 5.3 (b) shows the average episode duration in each batch. By design, the annotation time for strategy X is constant. For V -hor, after the first re-training iteration (from a weakly supervised to supervised detector) the average annotation cost grows because only difficult images are left to be annotated. On the contrary, annotation time for IAD decreases with every new batch because dialogs become stronger and box verifications become more successful.

Quality of boxes and resulting detector The data for training a detector and strategy in IAD includes both manually drawn boxes and boxes verified at $\text{IoU} \geq 0.7$. More precisely, IAD data collection results in 44% drawn boxes and 56% verified boxes. The quality of the verified boxes reaches 83% mIoU. The detector trained on the boxes produced by IAD reaches 98% of the mAP of the detector trained on ground-truth boxes.

Evolution of adaptive strategies To gain better understanding of adaptive behaviour of IAD, we study the composition of sequences of actions produced during

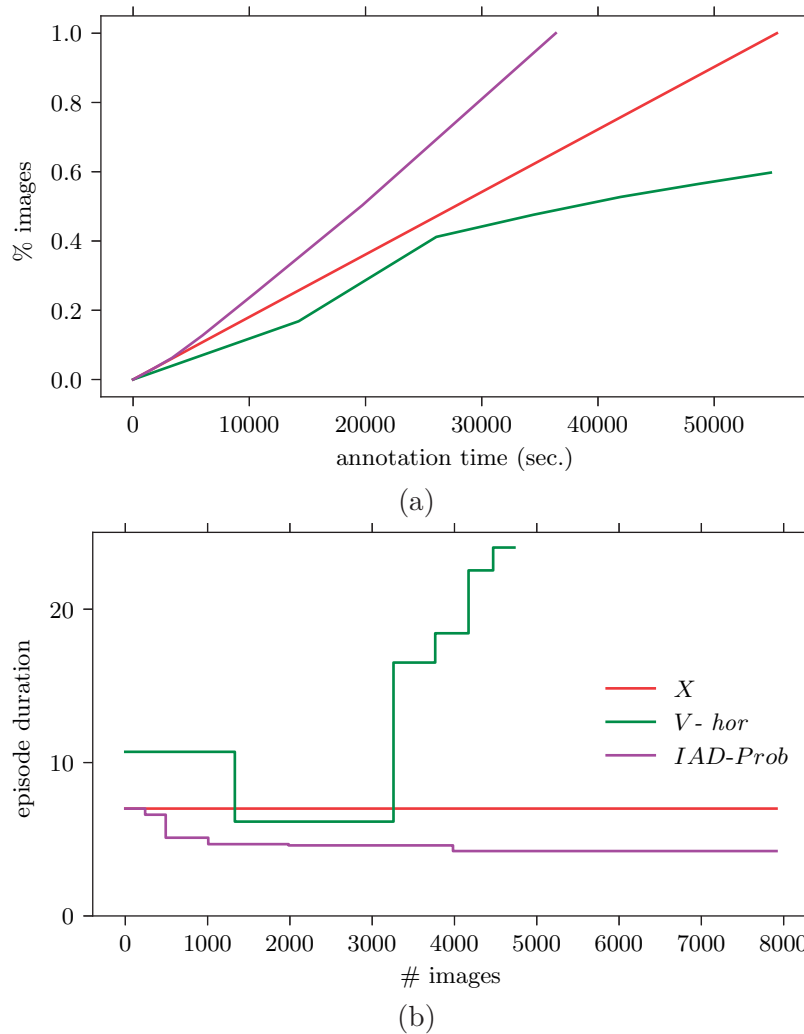


Figure 5.3 – (a) the proportion of annotated images as a function of annotation time for IAD-Prob and *standard* strategies. (b) average episode duration for various batches of data.

labelling of each batch. Figure 5.4 shows the proportion of images that are labelled by X , V , VX , VV and others sequences of actions. At the beginning of the process (batch 2), the vast majority of boxes is produced simply by asking for Extreme Clicking (X). It means that IAD learns that this is the best thing to do when the detector is weak. As the process continues, the detector gets stronger and IAD selects more frequently series composed purely of box verifications (V, VV), and mixed series with both actions (VX). This experiment demonstrates that IAD is capable of producing strategies that dynamically adapt to the change in problem property caused by the gradually improving detector. One cannot achieve this with any fixed strategy.

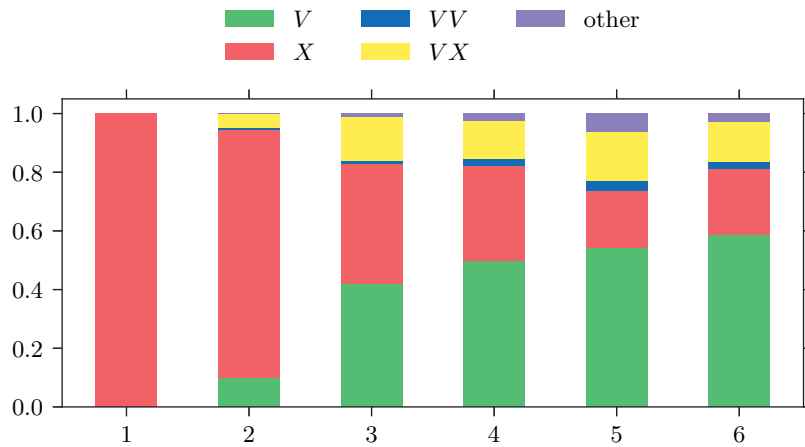


Figure 5.4 – The proportion of various annotation sequences in batches of data at 6 iteration.

5.6 Examples of dialogs

In this section we showcase the dialogs produced by IAD-Prob. We show several dialogs produced at iteration 5 of the experiment with a retrained detector (Section 5.5.3). Note that we illustrate examples of the dialogs’ *execution* and not *planned* dialogs. Questions to a human annotator are written at the top of images, box proposals for verification are shown in yellow, manually drawn bounding boxes are shown in blue and the annotator’s responses are at the bottom of each image.

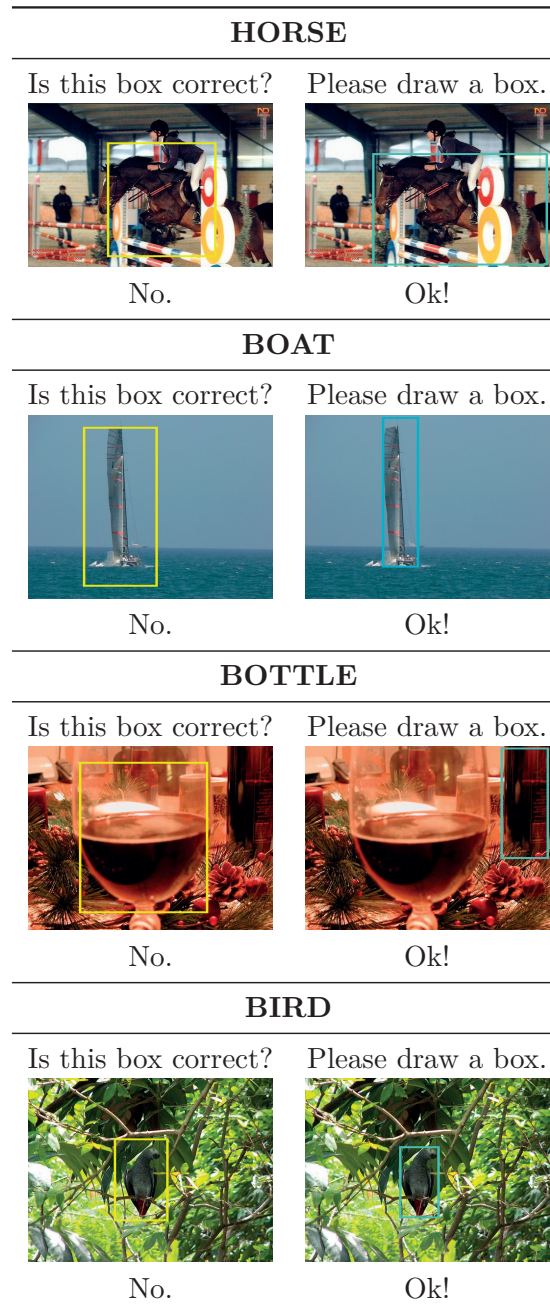


Figure 5.5 – If the detector’s output is strong on one bounding box, but this box is rejected (not tight enough, occluded part is missing, object belongs to another class), then box verification is followed by manual drawing.

MOTORBIKE

Is this box correct?



No.

Is this box correct?



Yes!

PLANE

Is this box correct?



No.

Is this box correct?



No.

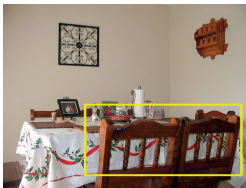
Is this box correct?



Yes!

DINING TABLE

Is this box correct?



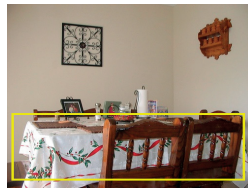
No.

Is this box correct?



No.

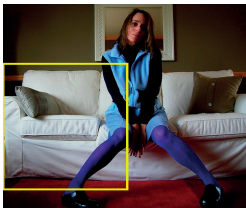
Is this box correct?



Yes!

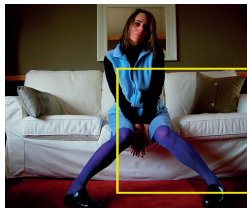
SOFA

Is this box correct?



No.

Is this box correct?



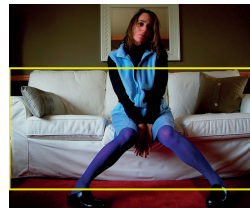
No.

Is this box correct?



No.

Is this box correct?



Yes!

Figure 5.6 – When the visual evidence for some object is strong, but the exact boundaries are hard to capture, a series of verifications can help to localise the object.

TRAIN

Is this box correct?



No.

Is this box correct?



No.

Please draw a box.



Ok!

DINING TABLE

Is this box correct?



No.

Is this box correct?



No.

Is this box correct?



No.

Is this box correct?



No.

Please draw a box.



Ok!

Figure 5.7 – When high-scored boxes for verification are exhausted, manual drawing is done.

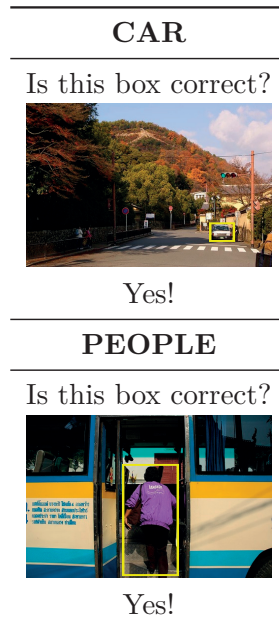


Figure 5.8 – Objects of classes for which the detector is strong are found with box verification even in complex scenes and configurations.

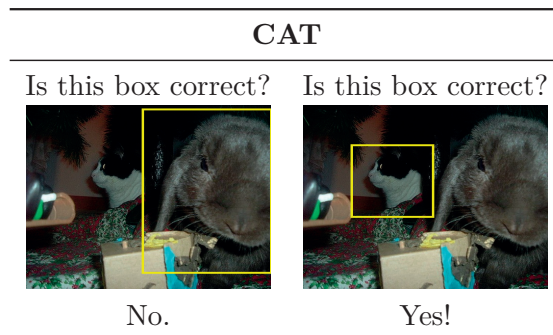



Figure 5.9 – When the detector’s output is strong on two objects in the scene, the correct bounding box is obtained with a series of box verifications. In this example the detector is confused between a rabbit and a cat.

CAT


Is this box correct?



Yes!

BICYCLE

Is this box correct?



Yes!

SHEEP

Is this box correct?



Yes!

TV MONITOR

Is this box correct?




Yes!

Figure 5.10 – Easily distinguishable objects on relatively uniform backgrounds are often found with a single verification for both big and small objects.

CHAIR


Please draw a box.



Ok!

POTTED PLANT


Please draw a box.



Ok!

BUS

Please draw a box.



Ok!

POTTED PLANT

Figure 5.11 – Small objects in cluttered scenes or objects without strong visual clues are annotated with manual drawing.

5.7 Conclusion

In this chapter we introduce Intelligent Annotation Dialogs for the task of bounding box annotation. IAD automatically chooses a sequence of actions $V^k D$ that results in time-efficient annotations. We present two methods to achieve this. The first method models the annotation time by predicting the acceptance probability for every box proposal. The second method skips the modelling step and learns an efficient strategy directly from trial-and-error interactions. In the extensive experimental evaluation IAD demonstrates competitive performance against various baselines and the ability to adapt to multiple problem properties.

This chapter addresses the question “*How to annotate?*” for the particular case of bounding box annotation for training object detectors. In contrast to Chapter 2 the annotation strategy is not designed by hand (we do not determine the best dialog for every dataset), but it is learnt from data. Properties of datasets, the current strength of the detector, the difficulty of an image and the class help the algorithm to construct an effective dialog.

6 Conclusions

Modern machine learning and computer vision methods have recently achieved unprecedented success in many applications. However, hidden behind this success are immense amounts of data used for training supervised algorithms and in fact, machine learning algorithms are usually restricted to the domain on which they have been trained. To move forward towards lifelong learning we need to look for efficient ways to collect sufficient training datasets with the minimal amount of human intervention.

This work studies two questions related to efficient data collection: “*What to annotate?*” and “*How to annotate?*”. On the one hand, by addressing the question “*What to annotate?*”, we seek to obtain small training datasets which enable to learn a model efficiently. The challenge is to identify the most informative data before obtaining their labels. On the other hand, addressing the question “*How to annotate?*” entails developing an efficient annotation method to obtain the label of a datapoint. Here the difficulty is to foresee which annotation method is going to be the most efficient without knowing what the annotation outcome is going to be. We have access to domain experts, however their time is scarce and expensive and we would like to minimise the burden of manual data labelling.

6.1 Summary

Our journey towards answering “*What to annotate?*” and “*How to annotate?*” starts with image segmentation applications. We manually design a selection strategy that finds the most informative datapoints and an annotation procedure that reduces 3D annotation to 2D. Our AL strategy and 2D plane annotation method are tailored for the particular problem at hand and perform remarkably well in this application. However, designing tailored strategies by hand does not scale, especially in the context of lifelong learning.

Then, we move towards designing selection strategies and annotation methods from data using meta-learning. First, we do it to address the question “*What to annotate?*” and

propose two data-driven AL strategies. In both cases we aim to obtain general-purpose AL techniques that can be applied to a wide variety of applications. To this end, we learn strategies either from synthetic data or from unrelated datasets in various application domains. In our search for the best strategy, we go from a greedy strategy towards a non-myopic one, and from a classifier-specific strategy towards an application-independent one.

Finally, we turn again to the question *"How to annotate?"* and suggest an approach that is also learnt from data, in the context of an application to object detection. To annotate data efficiently, we construct intelligent dialogs that learn which annotation modality to apply to data, through properties of the dataset, classifier, image and class.

To conclude, we consider two complementary questions *"What to annotate?"* and *"How to annotate?"* that contribute to reducing the annotation load. The keys to answer both them are 1) to make use of machine learning methods already during the data annotation stage and 2) to interactively collaborate with humans. Our work can be seen as a small step towards the goal of lifelong learning.

6.2 Future work

Methods to answer the questions *"What to annotate?"* and *"How to annotate?"* are a part of an exciting field, where rapid development in the last years brought many interesting solutions and new challenges. We are still far from having a universal solution that suits all applications and problem settings and many open questions remain.

Synergies between *"What to annotate?"* and *"How to annotate?"* Both questions are important for designing efficient labelling procedures. If they are used together, the resulting method might benefit from the synergy of the two methods, as was the case in Section 2. However, combining these methods can also introduce additional difficulties and their interaction should be taken into account. For example, in Chapter 5 we concentrated on the question of how every image should be annotated with the smallest cost and assumed that images come in a random order. If we could select the most difficult images first and annotate them (at a high cost) at the beginning of the procedure, it is possible that the box verification series would be much more successful later on and that, in total, more annotation effort would be saved. One way to explore this avenue might be to jointly learn a selection strategy and an intelligent annotation interface. By using a target quality objective, we can design a strategy that would minimise the annotation effort in complex scenarios by taking both the informativeness and cost of label and modelling the interactions between them.

Truly general-purpose data-driven AL The approach of Chapter 3 is limited because the hand-crafted features of the classifier need to be designed from scratch for

every new classification model. In Chapter 4 we become independent of the classifier, but the state and action representation are still partly hand-crafted, although suited for many more domains and classifiers. Ideally, further developments in data-driven AL will lead to methods that learn both state and action representation that are completely independent of the dataset and classifier. Some steps towards this is done by Pang et al. [136], however, they still require hand-crafted features (histograms of data features) augmented by heuristics specific to the classifier to learn a state representation.

Preventing forgetting in AL Currently, all state and action representations of AL are forgetful: they only know what happens at the current iteration and completely forget about the previous states of the classifier, actions of AL and to what consequences they led. Some authors try to overcome this limitation by using LSTM as a Q-function predictor [37, 9, 198]. However, the root cause of the problem lies in the MDP formulation: the markovian assumption implies that the action selection depends only on the current state. To remain in the MDP framework while still preventing forgetting, we can augment the states and actions of AL with temporal information. Similar modifications could also be beneficial for methods that learn how to annotate data (for example, dialogs of Chapter 5).

Jointly learning model parameters and hyper-parameters A well-known challenge in AL is to tune the hyper-parameters of the classifier based on limited data. Normally, this is done using cross-validation, which is impossible at the early stages of AL. Common solutions include using default (suboptimal) hyper-parameter values or keeping an additional validation set to find the best parameters. None of these two solutions is satisfactory: The first one sacrifices prediction quality and the second one is unrealistic in most cases where AL is useful (*i.e.* when labelled data is costly). Some meta-learning techniques have been proposed to learn how to find the best architectures of ML models [209, 10]. We can use similar techniques to adjust the hyper-parameters of the ML model on the fly as the amount of data grows. In the simplest form, **LAL-RL** can include additional actions for adjusting the classifier.

Never ending learning In the field of meta-AL, two main directions have been explored: 1) learning *during* every annotation episode from interactions with the domain of interest and 2) learning *before* the annotation episode from interactions with other domains. By restricting ourselves to just one of these two approaches, we miss the opportunity to use the information that is already present in the problem. Some works tried to combine these directions, but so far it was done only in the context of learning combinations of pre-existing strategies [35]. Data-driven methods can benefit from continuous learning, where they first learn a strategy from abundant data from other domains and then adjust it during the interaction with the user during annotation on the domain of interest. This can be particularly interesting if the on-line adjustment is done with a real human annotator: In this case the strategy might be able to adapt to

strength and weaknesses of a particular user and model the context switches. Besides, these ideas are valid for addressing the question “*How to annotate?*” as well.

From application-specific to data-driven and back Although we believe that it is important to develop general-purpose AL methods, in reality, domain specific knowledge can still help in some case. Still, a better way to integrate this domain-knowledge is by adjusting a general-purpose AL to a particular problem at hand. For example, this happened to our data-driven approach of Section 3 that was recently adapted to be used for action localisation [28].

Data annotation for deep learning AL can be used with most ML models, however, it has proved to be particularly challenging with deep learning models. The difficulties arise from 1) the large amount of data required to see the effect of training, 2) the lack of reliable probabilistic estimates, 3) the computational cost of retraining a new model with more data. Selecting data points one by one does not make much sense in this case and only batch-mode selection with a large batch size is practical. With deep learning techniques, the core question that needs to be addressed becomes: What makes a good batch? There have been several attempts to apply AL to deep learning [162, 53], but there is still ample room for improvement.

Better model from less data It was demonstrated in the literature [161] and in some of our experiments (Chapter 2) that occasionally it is possible to train a model that generalizes better using a small subset of data selected with AL than with all the data. This counter-intuitive phenomenon can be explained, for example, by the fact that we might avoid selecting outliers using AL. However, a more systematic understanding of this phenomenon is still missing and it is unclear if and how it can be exploited to train better models in general settings. To the best of our best knowledge, data-driven approaches were not tried so far in this context and we believe they might help to gain a better understanding.

6.3 Final remarks

We hope that this thesis convinced the reader that reducing the annotation cost is both important and challenging. The journey towards efficient annotation strategies is far from over as machine learning methods are unlikely to become less data-demanding. Thus, together with unsupervised and semi-supervised techniques, addressing the questions “*What to annotate?*” and “*How to annotate?*” will be essential for achieving the vision of lifelong machine learning.

Bibliography

- [1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012. [Cited on pages 28, 33, and 35]
- [2] D. Acuna, H. Ling, A. Kar, and S. Fidler. Efficient interactive annotation of segmentation datasets with polygon-RNN++. In *Conference on Computer Vision and Pattern Recognition*, 2018. [Cited on pages 18 and 85]
- [3] C. Adam-Bourdarios, G. Cowan, C. Germain, I. Guyon, B. Kégl, and D. Rousseau. The Higgs boson machine learning challenge. In *Workshop on High-energy Physics and Machine Learning at Advances in Neural Information Processing Systems*, 2015. [Cited on pages 71 and 75]
- [4] A. Al-Taie, H. K. Hahn, and L. Linsen. Uncertainty estimation and visualization in probabilistic segmentation. *Computers & Graphics*, 2014. [Cited on page 27]
- [5] S. Albarqouni, S. Matl, M. Baust, N. Navab, and S. Demirci. Playsourcing: A novel concept for knowledge creation in biomedical research. In *Large-Scale Annotation of Biomedical Data and Expert Label Synthesis workshop at Conference on Medical Image Computing and Computer Assisted Intervention*, 2016. [Cited on pages 20 and 22]
- [6] B. Andres, U. Köthe, M. Helmstaedter, W. Denk, and F. A. Hamprecht. Segmentation of SBFSEM volume data of neural tissue by hierarchical classification. In *DAGM Symposium on Pattern Recognition*, 2008. [Cited on page 28]
- [7] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, 1996. [Cited on pages 85 and 103]

Bibliography

- [8] J. Attenberg and F. Provost. Inactive learning? Difficulties employing active learning in practice. In *Conference on Knowledge Discovery and Data Mining Explorations Newsletter*, 2010. [Cited on page 14]
- [9] P. Bachman, A. Sordoni, and A. Trischler. Learning algorithms for active learning. In *International Conference on Machine Learning*, 2017. [Cited on pages 11, 13, 83, 84, 85, and 127]
- [10] B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcement learning. In *International Conference for Learning Representations*, 2017. [Cited on pages 85 and 127]
- [11] J. Baldridge and M. Osborne. Active learning and the total cost of annotation. In *Conference on Empirical Methods in Natural Language Processing*, 2004. [Cited on page 14]
- [12] Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 2004. [Cited on pages 9, 10, 14, 61, 62, 71, and 72]
- [13] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei. What’s the point: Semantic segmentation with point supervision. In *European Conference on Computer Vision*, 2016. [Cited on pages 18 and 21]
- [14] C. Becker, R. Rigamonti, V. Lepetit, and P. Fua. Supervised feature learning for curvilinear structure segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*, 2013. [Cited on page 42]
- [15] S. Bell, P. Upchurch, N. Snavely, and K. Bala. Material recognition in the wild with the materials in context database. In *Conference on Computer Vision and Pattern Recognition*, 2015. [Cited on pages 18 and 21]
- [16] M. Bellver, X. Giró-i Nieto, F. Marqués, and J. Torres. Hierarchical object detection with deep reinforcement learning. In *Workshop on Deep Reinforcement Learning at Advances in Neural Information Processing Systems*, 2016. [Cited on pages 85 and 103]
- [17] W. H. Beluch, T. Genewein, A. Nürnberger, and J. M. Köhler. The power of ensembles for active learning in image classification. In *Conference on Computer Vision and Pattern Recognition*, 2018. [Cited on page 8]
- [18] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *International Conference on Machine Learning*, 2009. [Cited on page 14]
- [19] H. Bilen and A. Vedaldi. Weakly supervised deep detection networks. In *Conference on Computer Vision and Pattern Recognition*, 2016. [Cited on page 101]
- [20] H. Bilen, M. Pedersoli, and T. Tuytelaars. Weakly supervised object detection with posterior regularization. In *British Machine Vision Conference*, 2014. [Cited on page 111]

-
- [21] A. Biswas and D. Parikh. Simultaneous active learning of classifiers & attributes via relative feedback. In *Conference on Computer Vision and Pattern Recognition*, 2013. [Cited on pages 7, 12, 17, and 21]
- [22] E. Borenstein and S. Ullman. Combined top-down/bottom-up segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008. [Cited on page 55]
- [23] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *International Conference on Computer Vision*, 2001. [Cited on page 18]
- [24] S. Branson, K. E. Hjørleifsson, and P. Perona. Active annotation translation. In *Conference on Computer Vision and Pattern Recognition*, 2014. [Cited on page 22]
- [25] S. Branson, G. Van Horn, and P. Perona. Lean crowdsourcing: Combining humans and machines in an online system. In *Conference on Computer Vision and Pattern Recognition*, 2017. [Cited on page 21]
- [26] T. M. Breuel. Fast recognition using adaptive subdivisions of transformation space. In *Conference on Computer Vision and Pattern Recognition*, 1992. [Cited on page 39]
- [27] M. Bucher, S. Herbin, and F. Jurie. Hard negative mining for metric learning based zero-shot classification. In *European Conference on Computer Vision*, 2016. [Cited on page 15]
- [28] F. Caba Heilbron, J.-Y. Lee, H. Jin, and B. Ghanem. What do I annotate next? An empirical study of active learning for action localization. In *European Conference on Computer Vision*, 2018. [Cited on page 128]
- [29] J. C. Caicedo and S. Lazebnik. Active object localization with deep reinforcement learning. In *International Conference on Computer Vision*, 2015. [Cited on pages 85 and 103]
- [30] L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a polygon-RNN. In *Conference on Computer Vision and Pattern Recognition*, 2017. [Cited on page 18]
- [31] G. C. Cawley. Baseline methods for active learning. In *Workshop on Active Learning and Experimental Design at Journal of Machine Learning Research*, 2011. [Cited on page 14]
- [32] H.-S. Chang, E. Learned-Miller, and A. McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. In *Advances in Neural Information Processing Systems*, 2017. [Cited on page 15]
- [33] R. Chattopadhyay, W. Fan, I. Davidson, S. Panchanathan, and J. Ye. Joint transfer and batch-mode active learning. In *International Conference on Machine Learning*, 2013. [Cited on page 13]

Bibliography

- [34] Y. Chen, H. Shioi, C. F. Montesinos, L. P. Koh, S. Wich, and A. Krause. Active detection via adaptive submodularity. In *International Conference on Machine Learning*, 2014. [Cited on page 103]
- [35] H.-M. Chu and H.-T. Lin. Can active learning experience be transferred? In *International Conference on Data Mining*, 2016. [Cited on pages 9, 10, 61, 62, and 127]
- [36] R. G. Cinbis, J. Verbeek, and C. Schmid. Multi-fold MIL training for weakly supervised object localization. In *Conference on Computer Vision and Pattern Recognition*, 2014. [Cited on page 111]
- [37] G. Contardo, L. Denoyer, and T. Artières. A meta-learning approach to one-step active-learning. In *CEUR International Workshop on Automatic Selection, Configuration and Composition of Machine Learning Algorithms*, 2017. [Cited on pages 11, 13, 83, 84, 85, and 127]
- [38] W. M. Czarnecki, S. M. Jayakumar, M. Jaderberg, L. Hasenclever, Y. W. Teh, S. Osindero, N. Heess, and R. Pascanu. Mix & match – agent curricula for reinforcement learning. In *International Conference on Machine Learning*, 2018. [Cited on page 14]
- [39] J. Dai, K. He, and J. Sun. BoxSup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *International Conference on Computer Vision*, 2015. [Cited on page 18]
- [40] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi. Calibrating probability with undersampling for unbalanced classification. In *IEEE Symposium Series on Computational Intelligence*, 2015. [Cited on page 75]
- [41] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Embodied question answering. In *Conference on Computer Vision and Pattern Recognition*, 2018. [Cited on page 85]
- [42] S. Dasgupta. Two faces of active learning. *Theoretical Computer Science*, 2011. [Cited on page 14]
- [43] J. Deng, J. Krause, and L. Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *Conference on Computer Vision and Pattern Recognition*, 2013. [Cited on pages 17 and 22]
- [44] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017. URL <http://archive.ics.uci.edu/ml>. [Cited on page 91]
- [45] Y. Duan, M. Andrychowicz, B. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. In *Advances in Neural Information Processing Systems*, 2017. [Cited on page 13]

-
- [46] F. Dubost, L. Peter, C. Rupprecht, B. Gutierrez Becker, and N. Navab. Hands-free segmentation of medical volumes via binary inputs. In *Large-Scale Annotation of Biomedical Data and Expert Label Synthesis workshop at Conference on Medical Image Computing and Computer Assisted Intervention*, 2016. [Cited on page 20]
- [47] S. Ebert, M. Fritz, and B. Schiele. RALF: A reinforced active learning formulation for object class recognition. In *Conference on Computer Vision and Pattern Recognition*, 2012. [Cited on pages 9, 10, 14, 61, and 62]
- [48] E. Elhamifar, G. Sapiro, A. Yang, and S. S. Sastry. A convex optimization framework for active learning. In *International Conference on Computer Vision*, 2013. [Cited on pages 8, 20, and 27]
- [49] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 2010. [Cited on pages 48, 71, 102, and 112]
- [50] M. Fang, Y. Li, and T. Cohn. Learning how to active learn: A deep reinforcement learning approach. In *Conference on Empirical Methods in Natural Language Processing*, 2017. [Cited on pages 11, 83, 84, and 85]
- [51] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006. [Cited on page 13]
- [52] A. Freytag, E. Rodner, and J. Denzler. Selecting influential examples: Active learning with expected model output changes. In *European Conference on Computer Vision*, 2014. [Cited on page 9]
- [53] Y. Gal, R. Islam, and Z. Ghahramani. Deep Bayesian active learning with image data. In *International Conference on Machine Learning*, 2017. [Cited on page 128]
- [54] E. Gavves, T. Mensink, T. Tommasi, C. Snoek, and T. Tuytelaars. Active transfer learning with zero-shot priors: Reusing past datasets for future tasks. In *International Conference on Computer Vision*, 2015. [Cited on page 13]
- [55] R. Gilad-Bachrach, A. Navot, and N. Tishby. Query by committee made real. In *Advances in Neural Information Processing Systems*, 2005. [Cited on page 8]
- [56] S. A. Goldman and M. J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 1995. [Cited on page 14]
- [57] N. Gordillo, E. Montseny, and P. Sobrevilla. State of the art survey on MRI brain tumor segmentation. *Magnetic Resonance Imaging*, 2013. [Cited on pages 28, 51, and 71]
- [58] L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006. [Cited on page 33]

Bibliography

- [59] M. Hasan and A. K. Roy-Chowdhury. Context aware active learning of activity recognition models. In *International Conference on Computer Vision*, 2015. [Cited on pages 20 and 27]
- [60] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001. [Cited on page 45]
- [61] S. C. H. Hoi, R. Jin, J. Zhu, and M. R. Lyu. Batch mode active learning and its application to medical image classification. In *International Conference on Machine Learning*, 2006. [Cited on pages 20 and 27]
- [62] W.-N. Hsu and H.-T. Lin. Active learning by learning. In *American Association for Artificial Intelligence Conference*, 2015. [Cited on pages 9, 10, 61, 62, 71, 84, and 90]
- [63] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko. Learning to reason: End-to-end module networks for visual question answering. In *International Conference on Computer Vision*, 2017. [Cited on page 85]
- [64] G. Hua, C. Long, M. Yang, and Y. Gao. Collaborative active learning of a kernel machine ensemble for recognition. In *Conference on Computer Vision and Pattern Recognition*, 2013. [Cited on pages 7 and 12]
- [65] M. Huang and G. Hamarneh. SwifTree: Interactive extraction of 3D trees supporting gaming and crowdsourcing. In *Large-Scale Annotation of Biomedical Data and Expert Label Synthesis workshop at Conference on Medical Image Computing and Computer Assisted Intervention*, 2017. [Cited on pages 20 and 22]
- [66] S.-J. Huang, R. Jin, and Z.-H. Zhou. Active learning by querying informative and representative examples. In *Advances in Neural Information Processing Systems*, 2010. [Cited on pages 71 and 90]
- [67] T.-K. Huang, L. Li, A. Vartanian, S. Amershi, and X. Zhu. Active learning with oracle epiphany. In *Advances in Neural Information Processing Systems*, 2016. [Cited on page 12]
- [68] M. Huijser and J. C. van Gemert. Active decision boundary annotation with deep generative models. In *International Conference on Computer Vision*, 2017. [Cited on page 12]
- [69] J. E. Iglesias, E. Konukoglu, A. Montillo, Z. Tu, and A. Criminisi. Combining generative and discriminative models for semantic segmentation of CT scans via active learning. In *Information Processing in Medical Imaging*, 2011. [Cited on pages 27 and 28]
- [70] P. Jain and A. Kapoor. Active learning for large multi-class problems. In *Conference on Computer Vision and Pattern Recognition*, 2009. [Cited on page 8]

-
- [71] S. D. Jain and K. Grauman. Predicting sufficient annotation strength for interactive foreground segmentation. In *International Conference on Computer Vision*, 2013. [Cited on pages 18 and 22]
- [72] S. D. Jain and K. Grauman. Active image segmentation propagation. In *Conference on Computer Vision and Pattern Recognition*, 2016. [Cited on page 27]
- [73] S. D. Jain and K. Grauman. Click carving: Segmenting objects in video with point clicks. In *Human Computation Workshop at American Association for Artificial Intelligence Conference*, 2016. [Cited on pages 18, 19, and 21]
- [74] D. Jayaraman and K. Grauman. Learning to look around: Intelligently exploring unseen environments for unknown tasks. In *Conference on Computer Vision and Pattern Recognition*, 2018. [Cited on pages 85 and 103]
- [75] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM international conference on Multimedia*, 2014. [Cited on page 46]
- [76] L. Jiang, D. Meng, Q. Zhao, S. Shan, and A. G. Hauptmann. Self-paced curriculum learning. In *American Association for Artificial Intelligence Conference*, 2015. [Cited on page 14]
- [77] L. Jiang, Z. Zhou, T. Leung, L.-J. Li, and L. Fei-Fei. MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, 2018. [Cited on page 14]
- [78] E. Johns, O. Mac Aodha, and G. J. Brostow. Becoming the expert – interactive multi-class machine teaching. In *Conference on Computer Vision and Pattern Recognition*, 2015. [Cited on pages 15 and 46]
- [79] J. Johnson, L. Ballan, and L. Fei-Fei. Love thy neighbors: Image annotation by exploiting image metadata. In *Conference on Computer Vision and Pattern Recognition*, 2015. [Cited on page 20]
- [80] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *Conference on Computer Vision and Pattern Recognition*, 2009. [Cited on pages 8 and 25]
- [81] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Scalable active learning for multiclass image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012. [Cited on page 25]
- [82] C. Käding, A. Freytag, E. Rodner, P. Bodesheim, and J. Denzler. Active learning and discovery of object categories in the presence of unnameable instances. In *Conference on Computer Vision and Pattern Recognition*, 2015. [Cited on pages 9, 12, and 46]

Bibliography

- [83] V. Kantorov, M. Oquab, M. Cho, and I. Laptev. ContextLocNet: Context-aware deep network models for weakly supervised localization. In *European Conference on Computer Vision*, 2016. [Cited on page 101]
- [84] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with Gaussian processes for object categorization. In *International Conference on Computer Vision*, 2007. [Cited on pages 25, 62, and 71]
- [85] S. Kazemzadeh, V. Ordonez, M. Matten, and T. L. Berg. ReferItGame: Referring to objects in photographs of natural scenes. In *Conference on Empirical Methods in Natural Language Processing*, 2014. [Cited on page 22]
- [86] K. Khan, M. Mauro, and R. Leonardi. Multi-class semantic segmentation of faces. In *International Conference on Image Processing*, 2015. [Cited on page 56]
- [87] M. Khodabandeh, A. Vahdat, G.-T. Zhou, H. Hajimirsadeghi, M. Javan Roshtkhari, G. Mori, and S. Se. Discovering human interactions in videos with limited data labeling. In *Conference on Computer Vision and Pattern Recognition Workshops*, 2015. [Cited on page 19]
- [88] A. Kolesnikov and C. H. Lampert. Improving weakly-supervised object localization by micro-annotation. In *British Machine Vision Conference*, 2016. [Cited on page 18]
- [89] K. Konyushkova, R. Sznitman, and P. Fua. Introducing geometry into active learning for image segmentation. In *International Conference on Computer Vision*, 2015. [Cited on pages 15, 23, and 25]
- [90] K. Konyushkova, R. Sznitman, and P. Fua. Learning active learning from real and synthetic data. In *Advances in Neural Information Processing Systems*, 2017. [Cited on pages 16, 61, 83, 84, 85, 90, 91, and 92]
- [91] K. Konyushkova, R. Sznitman, and P. Fua. Geometry in active learning for binary and multi-class image segmentation. arXiv:1606.09029v3, submitted to *Computer Vision and Image Understanding*, 2018. [Cited on pages 15, 23, and 25]
- [92] K. Konyushkova, J. R. R. Uijlings, C. H. Lampert, and V. Ferrari. Learning intelligent dialogs for bounding box annotation. In *Conference on Computer Vision and Pattern Recognition*, 2018. [Cited on pages 23, 85, and 101]
- [93] C. Körner and S. Wrobel. Multi-class ensemble-based active learning. In *European Conference on Machine Learning*, 2006. [Cited on page 8]
- [94] A. Kovashka, O. Russakovsky, L. Fei-Fei, and K. Grauman. *Crowdsourcing in Computer Vision*. Foundations and Trends in Computer Graphics and Vision, 2016. [Cited on pages 6 and 20]

-
- [95] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, A. Veit, S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik, D. Cai, Z. Feng, D. Narayanan, and K. Murphy. OpenImages: A public dataset for large-scale multi-label and multi-class image classification, 2017. URL <https://github.com/openimages/dataset>. [Cited on page 103]
- [96] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. [Cited on page 17]
- [97] M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, 2010. [Cited on page 14]
- [98] S. Lad and D. Parikh. Interactively guiding semi-supervised clustering via attribute-based explanations. In *European Conference on Computer Vision*, 2014. [Cited on page 21]
- [99] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Conference on Computer Vision and Pattern Recognition*, 2008. [Cited on pages 39 and 40]
- [100] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014. [Cited on page 21]
- [101] L. Lejeune, M. Christoudias, and R. Sznitman. Expected exponential loss for gaze-based video and volume ground truth annotation. In *Large-Scale Annotation of Biomedical Data and Expert Label Synthesis workshop at Conference on Medical Image Computing and Computer Assisted Intervention*, 2017. [Cited on pages 17, 19, 20, and 21]
- [102] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 2016. [Cited on pages 85 and 103]
- [103] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *ACM SIGIR proceedings on Research and Development in Information Retrieval*, 1994. [Cited on pages 25 and 90]
- [104] J. Li, J. M. Bioucas-Dias, and A. Plaza. Hyperspectral image segmentation using a new Bayesian approach with active learning. *IEEE Transactions on Geoscience and Remote Sensing*, 2011. [Cited on pages 27 and 28]
- [105] X. Li and Y. Guo. Adaptive active learning for image classification. In *Conference on Computer Vision and Pattern Recognition*, 2013. [Cited on page 8]

Bibliography

- [106] X. Li and Y. Guo. Multi-level adaptive active learning for scene classification. In *European Conference on Computer Vision*, 2014. [Cited on page 12]
- [107] L. Liang and K. Grauman. Beyond comparing image pairs: Setwise active learning for relative attributes. In *Conference on Computer Vision and Pattern Recognition*, 2014. [Cited on page 8]
- [108] D. Lin, J. Dai, J. Jia, K. He, and J. Sun. ScribbleSup: Scribble-supervised convolutional networks for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2016. [Cited on pages 18 and 21]
- [109] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, 2014. [Cited on page 25]
- [110] L. Lionni. *Frederick*. Pantheon Books, 1967. [Cited on page iii]
- [111] B. Liu and V. Ferrari. Active learning for human pose estimation. In *International Conference on Computer Vision*, 2012. [Cited on page 27]
- [112] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang. CASIA online and offline Chinese handwriting databases. In *IEEE International Conference on Document Analysis and Recognition*, 2011. [Cited on page 46]
- [113] M. Liu, W. Buntine, and G. Haffari. Learning how to actively learn: A deep imitation learning approach. In *Annual Meeting of the Association for Computational Linguistics*, 2018. [Cited on pages 11, 83, and 84]
- [114] C. Long and G. Hua. Multi-class multi-annotator active learning with robust Gaussian process for visual recognition. In *International Conference on Computer Vision*, 2015. [Cited on pages 8, 10, 12, 21, and 25]
- [115] C. Long, G. Hua, and A. Kapoor. Active visual recognition with expertise estimation in crowdsourcing. In *International Conference on Computer Vision*, 2013. [Cited on pages 7, 12, 21, and 25]
- [116] A. C. Lorena, G. Batista, A. de Carvalho, and M. C. Monard. Splice junction recognition using machine learning techniques. In *Brazilian Workshop on Bioinformatics*, 2002. [Cited on page 75]
- [117] L. Lovász. Random walks on graphs: A survey. In *Combinatorics, Paul Erdos in Eighty*, 1993. [Cited on page 33]
- [118] A. Lucchi, Y. Li, C. Becker, and P. Fua. Electron microscopy dataset. URL <https://cvlab.epfl.ch/data/em>. Accessed 25/07/18. [Cited on page 48]

-
- [119] A. Lucchi, Y. Li, K. Smith, and P. Fua. Structured image segmentation using kernelized features. In *European Conference on Computer Vision*, 2012. [Cited on page 75]
- [120] A. Lucchi, K. Smith, R. Achanta, G. Knott, and P. Fua. Supervoxel-based segmentation of mitochondria in EM image stacks with learned shape features. *IEEE Transactions on Medical Imaging*, 2012. [Cited on pages 28 and 48]
- [121] T. Luo, K. Kramer, D. B. Goldgof, L. O. Hall, S. Samson, A. Remsen, and T. Hopkins. Active learning to recognize multiple types of plankton. In *Journal of Machine Learning Research*, 2005. [Cited on page 25]
- [122] W. Luo, A. G. Schwing, and R. Urtasun. Latent structured active learning. In *Advances in Neural Information Processing Systems*, 2013. [Cited on pages 7 and 20]
- [123] O. Mac Aodha, S. Su, Y. Chen, P. Perona, and Y. Yue. Teaching categories to human learners with visual explanations. In *Conference on Computer Vision and Pattern Recognition*, 2018. [Cited on page 15]
- [124] J. Maiora and M. Graña. Abdominal CTA image analysis through active learning and decision random forests: Application to AAA segmentation. In *International Joint Conference on Neural Networks*, 2012. [Cited on page 25]
- [125] S. Mathe and C. Sminchisescu. Dynamic eye movement datasets and learnt saliency models for visual action recognition. In *European Conference on Computer Vision*, 2012. [Cited on pages 19 and 21]
- [126] B. H. Menze, A. Jacas, et al. The multimodal brain tumor image segmentation benchmark (BRATS). *IEEE Transactions on Medical Imaging*, 2015. [Cited on pages 51, 52, and 75]
- [127] P. Mettes, J. C. van Gemert, and C. G. M. Snoek. Spot on: Action localization from pointily-supervised proposals. In *European Conference on Computer Vision*, 2016. [Cited on pages 19, 21, and 101]
- [128] J. Michels, A. Saxena, and A. Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *International Conference on Machine Learning*, 2005. [Cited on pages 85 and 103]
- [129] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. In *Deep Learning Workshop at Advances in Neural Information Processing Systems*, 2013. [Cited on pages 86, 88, 89, 90, 91, and 111]
- [130] A. Mosinska, R. Sznitman, P. Głowacki, and P. Fua. Active learning for delineation of curvilinear structures. In *Conference on Computer Vision and Pattern Recognition*, 2016. [Cited on page 34]

Bibliography

- [131] A. Mosinska, J. Tarnawski, and P. Fua. Active learning and proofreading for delineation of curvilinear structures. In *Conference on Medical Image Computing and Computer Assisted Intervention*, 2017. [Cited on page 20]
- [132] K. Murugesan and J. Carbonell. Active learning from peers. In *Advances in Neural Information Processing Systems*, 2017. [Cited on page 12]
- [133] N. S. Nagaraja, F. R. Schmidt, and T. Brox. Video segmentation with just a few strokes. In *International Conference on Computer Vision*, 2015. [Cited on pages 18, 19, and 21]
- [134] T. Osugi, D. Kun, and S. Scott. Balancing exploration and exploitation: A new algorithm for active machine learning. In *International Conference on Data Mining*, 2005. [Cited on pages 9 and 10]
- [135] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 2010. [Cited on page 13]
- [136] K. Pang, M. Dong, Y. Wu, and T. Hospedales. Meta-learning transferable active learning policies by deep reinforcement learning. In *AutoML workshop International Conference on Machine Learning*, 2018. [Cited on pages 84, 85, 90, 92, and 127]
- [137] D. P. Papadopoulos, A. D. F. Clarke, F. Keller, and V. Ferrari. Training object class detectors from eye tracking data. In *European Conference on Computer Vision*, 2014. [Cited on pages 19, 21, and 101]
- [138] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller, and V. Ferrari. We don't need no bounding-boxes: Training object class detectors using only human verification. In *Conference on Computer Vision and Pattern Recognition*, 2016. [Cited on pages 19, 21, 101, 103, 105, 110, 112, 113, and 114]
- [139] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller, and V. Ferrari. Extreme clicking for efficient object annotation. In *International Conference on Computer Vision*, 2017. [Cited on pages 19, 102, 103, 105, 110, and 111]
- [140] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller, and V. Ferrari. Training object class detectors with click supervision. In *Conference on Computer Vision and Pattern Recognition*, 2017. [Cited on pages 19, 21, and 101]
- [141] A. Parkash and D. Parikh. Attributes for classifier feedback. In *European Conference on Computer Vision*, 2012. [Cited on page 17]
- [142] D. Pathak, P. Krähenbühl, and T. Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *International Conference on Computer Vision*, 2015. [Cited on page 18]

-
- [143] S. Paul, J. H. Bappy, and A. Roy-Chowdhury. Non-uniform subset selection for active learning in structured data. In *Conference on Computer Vision and Pattern Recognition*, 2017. [Cited on page 27]
- [144] A. Pentina, V. Sharmanska, and C. H. Lampert. Curriculum learning of multiple tasks. In *Conference on Computer Vision and Pattern Recognition*, 2015. [Cited on page 14]
- [145] T. Pietzsch, S. Saalfeld, S. Preibisch, and P. Tomancak. BigDataViewer: Visualization and processing for large image data sets. *Nature Methods*, 2015. [Cited on page 48]
- [146] P. O. Pinheiro and R. Collobert. From image-level to pixel-level labeling with convolutional networks. In *Conference on Computer Vision and Pattern Recognition*, 2015. [Cited on page 18]
- [147] J. C. Platt. Probabilistic outputs for SVMs and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 1999. [Cited on page 96]
- [148] S. M. Plaza. Focused proofreading to reconstruct neural connectomes from EM images at scale. In *Large-Scale Annotation of Biomedical Data and Expert Label Synthesis workshop at Conference on Medical Image Computing and Computer Assisted Intervention*, 2016. [Cited on pages 20, 21, and 27]
- [149] S. Ravi and H. Larochelle. Meta-learning for batch mode active learning. In *International Conference for Learning Representations Workshop Track*, 2018. [Cited on pages 11, 13, 14, 83, and 84]
- [150] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015. [Cited on page 110]
- [151] F. Rodrigues, F. Pereira, and B. Ribeiro. Gaussian process classification and active learning with multiple annotators. In *International Conference on Machine Learning*, 2014. [Cited on page 12]
- [152] G. Roig, X. Boix, R. de Nijs, S. Ramos, K. Kühnlenz, and L. Van Gool. Active MAP inference in CRFs for efficient semantic segmentation. In *International Conference on Computer Vision*, 2013. [Cited on page 15]
- [153] C. Rother, V. Kolmogorov, and A. Blake. “GrabCut” — interactive foreground extraction using iterated graph cuts. In *ACM International Conference & Exhibition on Computer Graphics and Interactive Techniques*, 2004. [Cited on page 18]
- [154] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015. [Cited on pages 17, 101, and 103]

Bibliography

- [155] O. Russakovsky, L.-J. Li, and L. Fei-Fei. Best of both worlds: Human-machine collaboration for object annotation. In *Conference on Computer Vision and Pattern Recognition*, 2015. [Cited on pages 22, 85, and 103]
- [156] S. Sabato, A. D. Sarwate, and N. Srebro. Auditing: Active learning with outcome-dependent query costs. In *Advances in Neural Information Processing Systems*, 2013. [Cited on page 13]
- [157] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, 2016. [Cited on pages 13 and 62]
- [158] A. Sarkar, C. Morrison, J. F. Dorn, R. Bedi, S. Steinheimer, J. Boisvert, J. Burggraaff, M. D'Souza, P. Kotschieder, S. Rota Bulò, L. Walsh, C. P. Kamm, Y. Zaykov, A. Sellen, and S. Lindley. Setwise comparison: Consistent, scalable, continuum labels for computer vision. In *ACM Conference on Human Factors in Computing Systems*, 2016. [Cited on pages 19 and 21]
- [159] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *International Conference for Learning Representations*, 2016. [Cited on pages 15 and 91]
- [160] B. Schmid, J. Schindelin, A. Cardona, M. Longair, and M. Heisenberg. A high-level 3d visualization API for Java and ImageJ. *BMC Bioinformatics*, 2010. [Cited on page 26]
- [161] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *International Conference on Machine Learning*, 2000. [Cited on pages 50 and 128]
- [162] O. Sener and S. Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference for Learning Representations*, 2018. [Cited on page 128]
- [163] B. Settles. From theories to queries: Active learning in practice. In *Active Learning and Experimental Design Workshop in Conjunction with International Conference on Artificial Intelligence and Statistics*, 2010. [Cited on pages 14, 20, and 27]
- [164] B. Settles. *Active Learning*. Morgan & Claypool Publishers, 2012. [Cited on pages 4, 6, 7, 8, 14, 31, 32, 34, 46, and 113]
- [165] B. Settles and M. Craven. An analysis of active learning strategies for sequence labeling tasks. In *Conference on Empirical Methods in Natural Language Processing*, 2008. [Cited on pages 8, 9, and 62]
- [166] B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. In *Advances in Neural Information Processing Systems*, 2007. [Cited on pages 20 and 27]

-
- [167] B. Settles, M. Craven, and L. Friedland. Active learning with real annotation costs. In *Workshop on Cost-Sensitive Learning at Advances in Neural Information Processing Systems*, 2008. [Cited on page 13]
- [168] V. Sharmanska, D. Hernández-Lobato, J. M. Hernández-Lobato, and N. Quadrianto. Ambiguity helps: Classification with disagreements in crowdsourced annotations. In *Conference on Computer Vision and Pattern Recognition*, 2016. [Cited on page 21]
- [169] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Conference on Computer Vision and Pattern Recognition*, 2016. [Cited on page 15]
- [170] P. Siva and T. Xiang. Weakly supervised object detector learning with model drift detection. In *International Conference on Computer Vision*, 2011. [Cited on page 111]
- [171] H. Su, J. Deng, and L. Fei-Fei. Crowdsourcing annotations for visual object detection. In *Human Computation Workshop at American Association for Artificial Intelligence Conference*, 2012. [Cited on pages 19, 101, 102, 103, 105, 110, and 111]
- [172] Q. Sun, A. Laddha, and D. Batra. Active learning for structured probabilistic models with histogram approximation. In *Conference on Computer Vision and Pattern Recognition*, 2015. [Cited on page 7]
- [173] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, 1998. [Cited on page 89]
- [174] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000. [Cited on page 84]
- [175] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Conference on Computer Vision and Pattern Recognition*, 2016. [Cited on page 110]
- [176] R. Sznitman and B. Jedynek. Active testing for face detection and localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010. [Cited on page 46]
- [177] R. Sznitman, C. Becker, F. Fleuret, and P. Fua. Fast object detection with entropy-driven evaluation. In *Conference on Computer Vision and Pattern Recognition*, 2013. [Cited on page 42]
- [178] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, 2016. [Cited on page 62]
- [179] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2001. [Cited on pages 7 and 25]

Bibliography

- [180] A. Top, G. Hamarneh, and R. Abugharbieh. Active learning for interactive 3D image segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*, 2011. [Cited on pages 28, 33, and 44]
- [181] G. Van Horn, S. Branson, S. Loarie, S. Belongie, and P. Perona. Lean multiclass crowdsourcing. In *Conference on Computer Vision and Pattern Recognition*, 2018. [Cited on pages 21 and 22]
- [182] A. Vezhnevets, J. M. Buhmann, and V. Ferrari. Active learning for semantic segmentation with expected change. In *Conference on Computer Vision and Pattern Recognition*, 2012. [Cited on pages 25 and 46]
- [183] S. Vijayanarasimhan and K. Grauman. Cost-sensitive active visual category learning. *International Journal of Computer Vision*, 2011. [Cited on page 13]
- [184] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016. [Cited on pages 11 and 13]
- [185] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *ACM Conference on Human Factors in Computing Systems*, 2004. [Cited on page 22]
- [186] L. von Ahn, R. Liu, and M. Blum. Peekaboom: A game for locating objects in images. In *ACM Conference on Human Factors in Computing Systems*, 2006. [Cited on page 22]
- [187] C. Vondrick, D. Patterson, and D. Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 2013. [Cited on pages 19 and 20]
- [188] C. Wah, G. Van Horn, S. Branson, S. Maji, P. Perona, and S. Belongie. Similarity comparisons for interactive fine-grained categorization. In *Conference on Computer Vision and Pattern Recognition*, 2014. [Cited on pages 17 and 21]
- [189] I. Wang, P. Narayana, J. Smith, B. Draper, R. Beveridge, and J. Ruiz. EASEL: Easy automatic segmentation event labeler. In *International Conference on Intelligent User Interfaces*, 2018. [Cited on pages 19 and 21]
- [190] T. Wang, B. Han, and J. Collomosse. TouchCut: Fast image and video segmentation using single-touch interaction. *Computer Vision and Image Understanding*, 2014. [Cited on pages 18 and 21]
- [191] X. Wang, T.-K. Huang, and J. Schneider. Active transfer learning under model shift. In *International Conference on Machine Learning*, 2014. [Cited on page 13]
- [192] Z. Wang, B. Du, L. Zhang, L. Zhang, M. Fang, and D. Tao. Multi-label active learning based on maximum correntropy criterion: Towards robust and discriminative labeling. In *European Conference on Computer Vision*, 2016. [Cited on page 8]

-
- [193] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 1992. [Cited on pages 84 and 89]
- [194] D. Weinshall, G. Cohen, and D. Amir. Curriculum learning by transfer learning: Theory and experiments with deep networks. In *International Conference on Machine Learning*, 2018. [Cited on page 14]
- [195] P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems*, 2010. [Cited on page 21]
- [196] M. Wigness, B. A. Draper, and J. R. Beveridge. Efficient label collection for unlabeled image datasets. In *Conference on Computer Vision and Pattern Recognition*, 2015. [Cited on pages 17 and 20]
- [197] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992. [Cited on page 84]
- [198] M. Woodward and C. Finn. Active one-shot learning. In *Deep Reinforcement Learning Workshop at Advances in Neural Information Processing Systems*, 2016. [Cited on pages 13, 84, 85, and 127]
- [199] J. Xu, A. G. Schwing, and R. Urtasun. Learning to segment under various forms of weak supervision. In *Conference on Computer Vision and Pattern Recognition*, 2015. [Cited on pages 18 and 21]
- [200] S. Yan, K. Chaudhuri, and T. Javidi. Active learning from imperfect labelers. In *Advances in Neural Information Processing Systems*, 2016. [Cited on page 12]
- [201] L. Yang and J. Carbonell. Buy-in-bulk active learning. In *Advances in Neural Information Processing Systems*, 2013. [Cited on page 20]
- [202] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 2015. [Cited on page 8]
- [203] A. Yao, J. Gall, C. Leistner, and L. Van Gool. Interactive object detection. In *Conference on Computer Vision and Pattern Recognition*, 2012. [Cited on pages 19, 21, and 103]
- [204] C. Zhang and K. Chaudhuri. Active learning from weak and strong labelers. In *Advances in Neural Information Processing Systems*, 2015. [Cited on page 12]
- [205] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, 2004. [Cited on pages 27 and 34]

Bibliography

- [206] J. Zhu, J. Mao, and A. Yuille. Learning from weakly supervised data by the expectation loss SVM (e-SVM) algorithm. In *Advances in Neural Information Processing Systems*, 2014. [Cited on page 18]
- [207] Y. Zhu, Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao. Soft proposal networks for weakly supervised object localization. In *International Conference on Computer Vision*, 2017. [Cited on page 101]
- [208] A. Zlateski, R. Jaroensri, P. Sharma, and F. Durand. On the importance of label quality for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2018. [Cited on page 18]
- [209] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *International Conference for Learning Representations*, 2017. [Cited on pages 85 and 127]

Ksenia Konyushkova

CONTACT INFORMATION	Avenue d'Ouchy 18 Lausanne 1006, Switzerland Currently a research assistant at CVLab, EPFL	✉ ksenia.konyushkova@epfl.ch 🌐 ksenia.konyushkova.com ☎ +41 78 860 28 89
RESEARCH INTERESTS	Active learning, computer vision, reinforcement learning, meta-learning, interactive learning	
EDUCATION	École polytechnique fédérale de Lausanne EPFL, Switzerland, School of computer and communication sciences	
	<i>PhD in computer science</i>	since September 2013
	<ul style="list-style-type: none">• Thesis topic: reducing annotation load in computer vision and beyond, advisors: Prof. Pascal Fua and Prof. Raphael Sznitman• <i>Course work</i>: Pattern Classification and Machine Learning, Computer Vision, Quantum Information Theory and Computation, Privacy Protection	
	University of Helsinki, Finland, Department of computer science	
	<i>MSc in algorithms and machine learning</i>	September 2011 – June 2013
	<ul style="list-style-type: none">• Thesis topic: ImSe: Instant Interactive Image Retrieval System with Exploration/Exploitation trade-off (grade: <i>eximia cum laude approbatur</i>, 5/5), advisors: Prof. Petri Myllymäki and Dr Dorota Glowacka; GPA: 4.8/5	
	National research institution, Higher School of Economics, Russia, Faculty of business informatics and applied mathematics	
	<i>BSc in business informatics</i>	September 2007 – June 2011
	<ul style="list-style-type: none">• Thesis topic: ontology based information retrieval algorithms of business data, advisor: Prof. Valery Kalyagin; GPA: 9/10	
ACHIEVEMENTS AND AWARDS	<ul style="list-style-type: none">• CVPR18 Doctoral Consortium travel award, NIPS17 travel award• Grand winner of LauzHack 2016 with project wshhh• Teaching award for contribution in a new course Pattern Classification and Machine Learning, 2014• Best paper award for paper “Directing Exploratory Search: Reinforcement Learning from User Interactions with Keywords”, Intelligent User Interfaces (IUI), Santa Monica, California, 2013• Google Anita Borg Memorial Scholarship finalist, 2012• Region Olympiads winner in Nizhny Novgorod region in astronomy and economics, 2005–2007	
PROJECTS AND WORK EXPERIENCE	École polytechnique fédérale de Lausanne EPFL, Switzerland	
	<i>Research assistant</i>	
	<i>CVLab, Prof. Pascal Fua</i>	since March 2014
	We improve Active Learning strategies by learning them with reinforcement or supervised learning, design special strategies for binary and multi-class semantic segmentation.	
	<i>IVRL, Prof. Sabine Süssstrunk</i>	September 2013 – January 2014
	We analyse a database of God(s) drawings by children from different cultural backgrounds and ages to identify developmental and cross-cultural patterns.	
	Google Research, Zurich, Switzerland	
	<i>Research intern</i>	August 2017 – November 2017
	<i>Machine Perception Team, Prof. Vittorio Ferrari</i>	
	We design a faster method to annotate objects in images with bounding boxes by learning the best annotation strategy from data. Learning intelligent dialogs for bounding box annotation.	
	Helsinki Institute for Information Technologies, Helsinki, Finland	
	<i>Research assistant</i>	March 2012 – July 2013
	<i>CoSCo, Prof. Petri Myllymäki</i>	
	We propose an approach for exploratory retrieval of scientific information or images.	
	Intel Corporation, Nizhny Novgorod, Russia	
	<i>Business analyst intern</i>	March 2011 – August 2011

SELECTED PUBLICATIONS	<p>Konyushkova K., Sznitman R., Fua P., Discovering General-Purpose Active Learning Strategies, arXiv:1810.04114, 2018.</p> <p>Konyushkova K., Uijlings J., Lampert C., Ferrari V., Learning Intelligent Dialogs for Bounding Box Annotation, Computer Vision and Pattern Recognition (CVPR), 2018.</p> <p>Konyushkova K., Sznitman R., Fua P., Learning Active Learning from Data, Neural Information Processing Systems (NIPS), 2017.</p> <p>Konyushkova K., Sznitman R., Fua P., Geometry in Active Learning for Binary and Multi-class Image Segmentation, under revision at CVIU.</p> <p>Konyushkova K., Sznitman R., Fua P., Introducing Geometry in Active Learning for Image Segmentation, International Conference on Computer Vision (ICCV), 2015.</p> <p>Konyushkova K., Arvanitopoulos N., Dandarova Robert Z., Brandt P.-Y., Süssstrunk S., God(s) Know(s): Developmental and Cross-Cultural Patterns in Children Drawings, arXiv:1511.03466, 2015.</p> <p>Konyushkova K., Glowacka D., ImSe: Exploratory Time-Efficient Image Retrieval System, Communications in Computer and Information Science, Information Retrieval, Springer, 2015.</p> <p>Konyushkova K., Glowacka D., Content-based image retrieval with hierarchical Gaussian Process bandits with self-organizing maps, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), 2013.</p> <p>Glowacka D., Ruotsalo T., Konyushkova K., Athukorala K., Kaski S. and Jacucci G., Directing Exploratory Search: Reinforcement Learning from User Interactions with Keywords, Intelligent User Interfaces (IUI), 2013. (best paper award)</p> <p>Ruotsalo T., Athukorala K., Glowacka D., Konyushkova K., Oulasvirta A., Kaipainen S., Kaski S. and Jacucci G., Supporting exploratory search tasks with interactive user modeling, Artificial Intelligence and Statistics (AISTATS), 2013.</p> <p>Konyushkova K., Kalyagin V., Social Choice and Algorithms of Ranking in the Internet, Collected articles of Higher School of Economics of Nizhny Novgorod, 2010.</p>
TALKS AND PRESENTATIONS	<ul style="list-style-type: none"> • Interview about Learning Intelligent Dialogs for CVPR Daily, Salt Lake City, USA, 2018 • Learning Active Learning, interview for TWIMLAI podcast, Long Beach, USA, 2017 • Active Learning for Segmentation in Neuroimaging, Junior Scientist Workshop on Machine Learning and Computer Vision, Janelia, USA, 2016 • Active Learning for Biomedical Image Segmentation, Swiss Machine Learning Day, 2014 • “Interactive Exploratory Search”, Seminar on Informational Retrieval and Natural Language Processing, Yandex, St. Petersburg, Russia, 2013 • Imse: Exploratory Image Retrieval with Hierarchies, PASCAL2 review meeting, Palma de Mallorca, Spain, 2013
CONFERENCE REVIEWING	<p>ICLR 19; NIPS 18 (30% of best reviewers); JMLR 18; ACML 18; LABELS@MICCAI 16-18; CVRSUAD@ECCV 16-17</p>
TEACHING	<p>Courses: Pattern Classification and Machine Learning, Probability and Statistics</p> <p>Student supervision:</p> <ul style="list-style-type: none"> • Semion Sidorenko, “Reducing Annotation Efforts in Biomedical Image Segmentation with Intelligent Interfaces”, 2017 • Natalija Gucevskaja, “Active Learning and Teaching for Image classification”, 2016 • Udaranga Wickramasinghe, “Active Learning for Multi-Class Image Segmentation”, 2015
SKILLS	<p>Python, Jupyter notebooks, Colab, TensorFlow, Matlab, Linux, Mac OS</p>
LANGUAGES	<p>English (fluent, Certificate in Advanced English, Cambridge), Russian (native), French (fluent), Finnish (basic)</p>
INTERESTS	<p>Photography and painting (watercolours and pastel), snowboard, ballet and rock’n’roll, sailing, quantum information theory and computation</p>

