POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Politecnico
di Torino

Master's Degree Thesis

Explaining black-box models in deep active learning in the context of image classification

Supervisors

Prof. Tania Cerquitelli

Dr. Salvatore Greco

Candidate

Manuele Macchia

A.Y. 2020/2021

Contents

Chapter 1

Introduction

In recent years, artificial intelligence and machine learning shaped the way we interact with the world. Many systems that we use daily are built upon machine learning models and algorithms. Personal virtual assistants on our phones can understand our speech and instantly provide relevant information to answer our questions. Spotify's recommendation algorithm suggests songs and creates custom playlists tailored to our taste. Our mail client may offer suggestions to auto-complete our sentences and help write a response faster.

Currently, artificial intelligence is driving innovation in numerous fields. Carmakers are building autonomous vehicles by training machine learning models on data collected over billions of kilometers of human driving and simulations. This new generation of vehicles will change the way we commute and travel. In medicine, artificial intelligence applications can detect anomalies in patient scans and predict the likelihood of a given disease based on the particular patient's history and those of millions of other patients.

One of the most promising and most widely used machine learning techniques is deep learning. Deep neural networks can model and extract patterns from unstructured data such as images, videos, and text and perform predictions on new data points with high accuracy. Deep networks stack multiple representation layers that ideally correspond to different levels of abstraction. This multi-layer architecture enables machines to build complex concepts out of simpler concepts [1]. Deep learning techniques attained outstanding results in various complex tasks, such as semantic segmentation [2, 3] and sentiment analysis of textual data and machine translation [4].

This work focuses on a supervised learning problem, namely multi-class

image classification. Simpler feed-forward neural networks such as multilayer perceptrons (MLPs) do not achieve high performance on this sort of task. Instead, a specialized kind of deep neural architecture had success in benchmarks and practical applications. Convolutional neural networks (CNNs) [5] are a particular kind of neural network that is especially useful for processing data with a known grid-like topology, such as image data [1]. In the last decade, CNNs have become the dominating approach for image classification [6], steadily improving on benchmarks such as the ImageNet image classification challenge [7]. Some notorious CNN architectures are AlexNet [8], VGG [9], and ResNet [10]. An essential requirement for training deep neural networks such as CNNs is a large amount of labeled data. The high cost of manual labeling thousands of samples can prevent the usage of deep learning in many fields, particularly those where labeling requires a large degree of professional knowledge [11], such as medicine.

Active learning limits manual labeling costs. It selects a limited amount of samples from a large pool of unlabeled data in an iterative fashion. At each step, active learning aims to query the unlabeled samples that maximize the performance gain of the model. Naturally, active learning may be applied in the context of deep learning to limit the amount of data required by deep models. The goal of deep active learning (DAL) is to reduce labeling costs while retaining the exceptional performance of deep neural networks. Many works have successfully applied this technique to real-world problems where obtaining labels is expensive, such as medical image analysis [12, 13] and classification of hyperspectral images [14]. However, combining active learning and deep learning poses some specific challenges [11] that we should consider when applying this technique in practice. We will introduce a typical DAL framework and discuss the related research in Section 2.1.

Deep neural networks are inherently black-box, i.e., their decision-making process is opaque. As machine learning models become more widespread for high-stakes applications, the ability to explain the reasons behind a given prediction and understand how the model operates globally is of fundamental importance. The goal of explainable AI (xAI) research is to provide tools to solve this problem, thereby helping to increase trust in the model and encourage its conscious adoption. We will focus on xAI techniques and state-of-the-art approaches in Section 2.2.

Ideally, xAI can provide insights into the inner workings of the model. In the context of image classification, xAI may help clarify why a model misclassifies a given sample or whether it is biased towards a specific class.

By injecting this knowledge into the DAL framework, it may be possible to improve the performance of a model more quickly. Our work focuses on the design of such a system, based on the information provided by EBAnO [15, 16], an explanation framework able to analyze the decision-making process of CNNs in image classification through the mining the knowledge contained in convolutional layers in a completely unsupervised manner. EBAnO is suitable for our purposes, as it does not generate surrogate models to explain predictions. Instead, it can directly access network knowledge, which we can exploit to understand its points of weakness. We query the samples that expose these pitfalls and use them to correct and improve the model based on this information.

This work aims to design a query strategy that exploits the rich information provided by EBAnO in an unsupervised manner to improve the performance of DAL with respect to classic DAL baselines. In Chapter 3, we present our experimental setup and describe several novel query strategies. In Chapter 4, we present and discuss the results of our experiments. In Chapter 5, we draw conclusions and propose future research directions.

Chapter 2

Related work

In this chapter, we present the DAL framework and discuss classic approaches and recent research in query strategies. We describe different types of xAI techniques and discuss state-of-the-art approaches. Finally, we outline the possibilities and challenges of integrating xAI into the proposed DAL framework.

2.1 Deep active learning

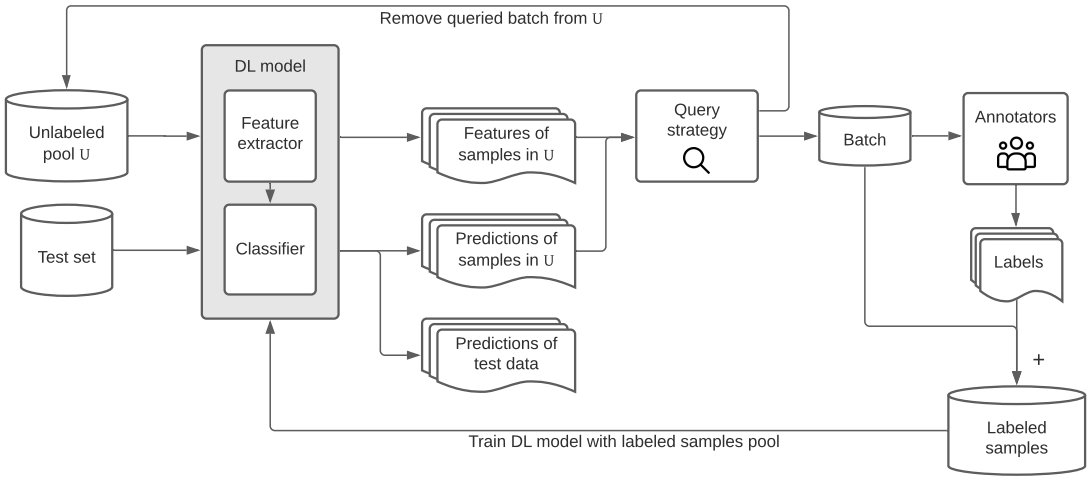

Figure 2.1: DAL framework. The diagram shows the main components and flow of operations.

The purpose of DAL is to train a deep neural network by labeling as few samples as possible. Figure 2.1 presents a typical DAL framework. DAL iteratively selects the most informative samples to annotate from a large pool of unlabeled data based on a specific query strategy. The first step consists of training a deep learning model on a set of labeled data. If no data is readily available, the oracle, i.e., one or more domain experts, may choose a subset of the unlabeled pool to annotate. At each iteration, DAL queries a batch of samples from the unlabeled pool. Annotating examples is a costly procedure: the larger the batch, the higher the cost incurred. Therefore, choosing a suitable query strategy is critical to maximizing model performance as quickly as possible. The procedure ends when the labeling budget ends, or the model reaches a satisfactory performance. Choosing appropriate stopping criteria is still an open problem in active learning literature [17].

Undoubtedly, the key component of DAL is the query strategy. As a first naive approach, one may manually select the best samples from the unlabeled pool. The authors of [18] train a CNN via a DAL loop in the context of medical image classification. At each iteration, they perform manual quality assurance on all predictions for samples in the unlabeled pool. The annotators select the worst-classified images based on their knowledge, label them, and add them to the training set. This procedure is repeated until predictions for unlabeled samples are satisfactory or there are no more unlabeled samples.

The approach adopted by [18] requires a double effort by annotators. Experts first analyze model predictions for all samples in the unlabeled pool and then annotate the queried instances. This strategy quickly becomes cost-prohibitive, especially when the unlabeled pool is large. Moreover, it may not be obvious to a human annotator which are the samples that maximize the model performance. Humans may introduce sampling bias into the model, leading to poor predictive performance and other unexpected consequences. In the next section, we will consider query rules that operate in an unsupervised manner to limit resources and not waste budget.

Given the importance of choosing appropriate query rules, the research in query strategies for active learning is rich. According to [11, 19], there are three different approaches to active learning querying.

Membership query synthesis The learner, i.e., the deep learning model, can request to query the label of any sample in the input space. In other words, the learner can query the label of any sample generated by it.

Stream-based selective sampling The learner evaluates whether each sample in the stream needs to be labeled by the oracle.

Pool-based sampling The learner chooses the best query sample, or batch of samples, based on the evaluation and ranking of the entire unlabeled dataset.

Pool-based sampling is the most promising approach when combined with deep learning methods, as neural networks rely on batch training [19]. Our work focuses on pool-based techniques in the context of image classification. The following sections will discuss classic and state-of-the-art query strategies based on different takes on the characteristics of the ideal set of samples to query at each iteration.

2.1.1 Uncertainty-based methods

The approach adopted by uncertainty-based query strategies is to select the unlabeled samples closest to the classifier's boundary, i.e., the most uncertain ones. The intuition is that the more uncertain a sample prediction is, the more information we gain by including the ground truth for that sample in the training set [19].

The critical aspect of uncertainty-based sampling is to choose an appropriate uncertainty measure for unlabeled samples. In the context of deep learning, a common uncertainty measure is the softmax output of the neural network. The softmax function provides a normalized score to each class of a given sample, which we can interpret as a probability distribution over classes. Given the softmax output for all unlabeled samples, we can adopt different strategies to measure their uncertainty.

The classic uncertainty sampling methods which use the softmax output of the network are least confident sampling, margin sampling, and entropy sampling. The least confident strategy [20] queries the instances whose predicted output is the least confident. A drawback of this approach is that it ignores the rest of the softmax distribution. Margin sampling [21] is based on the output prediction margin. It considers the difference between the first and second most likely predictions of a given instance, i.e., its prediction margin. The basic intuition is that examples with smaller margins are more ambiguous than instances with wider margins. Therefore, instances with small margins are more difficult to classify and provide more information during model training. However, similarly to least confident sampling,

this approach does not take into account most of the output distribution. Entropy [22] is a measure of a variable's average information content or uncertainty. We can interpret the entropy of the network output as an uncertainty measure for the sample. The higher the entropy, the more uncertain the sample is. This measure, unlike least confident and margin sampling, takes into account the complete class probability distribution.

CEAL [23] is based on the classic uncertainty sampling methods. In a complementary manner, they propose to use high-confidence samples by including them into the training set without human supervision, i.e., using pseudo-labels. This technique allows the model to receive a higher number of labeled samples more cost-effectively.

The authors of [24] exploit the DAL framework to perform in situ plankton classification. They use the classic uncertainty-based query strategies: least confidence, margin sampling, and entropy. The confidence of predictions is measured via the softmax output of the network. They compute the top k most informative samples from the unlabeled pool and train the model on the whole labeled dataset at each step. The authors also include high-confidence samples in the training set via pseudo-labels, as in [23].

A medical work on histopathological image classification [25] uses entropy-based sampling and considers the softmax output layer of the network to extract a probability distribution over classes. The authors include high-confidence samples in the training set, auto-labeled by the network from the previous DAL iteration. Similarly, [26] considers the softmax output with classic uncertainty-based sampling strategies to find the top-k uncertain samples. This work includes pseudo-labels for high-confidence samples in the training set.

The work in [13] tackles an unbalanced image classification problem in a medical setting using DAL. The softmax output of the network is used to calculate weighted information entropy, which is utilized as an uncertainty score for querying additional data at each iteration.

Authors of [27] use the softmax network output to obtain a probability distribution over classes. An entropy-based query strategy selects the top k most uncertain data points. Besides the standard DAL procedure, this work implements a joint loss that minimizes cross-entropy for labeled data and entropy for unlabeled data.

Many works that use uncertainty-based query strategies exploit the softmax output of the final layer of the network to obtain a probability distribution over classes for each sample in the unlabeled pool. However, the

softmax response may be an unreliable measure of uncertainty, as it is often too confident [11]. In other words, the softmax layer output may be high for a given label even when the network is uncertain about it.

Alternatively, we may exploit the Bayesian setting to address the problem of measuring the uncertainty of predictions. In Bayesian CNNs (B-CNNs) [28], network weights are described by probability distributions computed as Bayesian posteriors rather than simple point estimates. Conveniently, dropout can be used to approximate Bayesian inference in B-CNNs. We refer to this technique as Monte Carlo (MC) dropout. More specifically, the authors of [28] train a model using dropout before every layer with weights and use dropout at inference time to sample from the posterior distribution of the weights.

DBAL (Deep Bayesian Active Learning) [29] applies B-CNNs to the active learning framework. The authors evaluate the performance of multiple baseline query strategies, namely maximum entropy, BALD [30], variation ratios, maximum standard deviation, and random acquisition, and compare them to their non-deterministic counterparts that rely on B-CNN uncertainty. They find that their uncertainty measurement attains higher accuracy in fewer active learning steps. Deep Ensemble Bayesian Active Learning (DEBAL) [31] improves on Bayesian active learning methods introduced by [29] by proposing a stochastic ensemble of multiple MC dropout models to alleviate the mode collapse problem suffered by DBAL. Similarly, the authors of [32] consider the softmax output distribution obtained via MC dropout and deep ensembles of five networks of identical architecture but different random initializations. They find that deep ensembles lead to better performance of uncertainty-based acquisition functions.

Classic uncertainty-based query strategies find the top k elements according to a given metric. Each element is chosen regardless of other elements inside the queried batch. According to [11], this may constitute a problem. If we only consider information-rich samples, i.e., the most uncertain ones, we risk wasting labeling resources as we may end up with a set of samples very similar to one another. These samples collectively provide less information than a more diverse, albeit less uncertain, set of samples.

The author of [33] warns that querying strategies based exclusively on uncertainty may be subject to sampling bias. The supervised learner queries and learns only the most uncertain samples, i.e., samples close to the current classifier's boundary. Consequently, the classifier shifts from the underlying data distribution as it only sees very uncertain samples.

These issues stem from the same fundamental problem of purely uncertainty-based strategies. The main drawback of these methods is not considering the similarity of selected samples. Therefore, they might select redundant samples to annotate, thereby wasting labeling resources [24]. To alleviate this problem, we can use hybrid query strategies. Hybrid methods take into account both sample uncertainty and sample diversity.

2.1.2 Diversity-based methods and hybrid strategies

The main difference between DAL and classic AL is that DAL uses batch-based sample querying instead of the one-by-one query method in classic active learning [11]. Uncertainty-based techniques described in Section 2.1.1 are not optimized for sampling batches. Instead, they query batches of size k by taking the top-k samples according to the associated metrics. This approach does not consider that these k samples may contain redundant information, and therefore labeling resources may go to waste.

Query techniques based on diversity (or representativeness) address the problem of uncertainty-based techniques that may select batches composed of excessively similar samples. Query strategies that balance the uncertainty and diversity of samples are often referred to as hybrid query strategies.

The work in [34] considers both informativeness, as measured by prediction uncertainty, and representativeness of the queried batch. Given the batch size k and a pre-filter factor β, to increase the diversity of examples, the algorithm pre-filters the top-βk informative samples and clusters them via K-means to k clusters. The k different samples closest to the cluster centers are selected. This process is repeated until the budget ends. The author finds that this approach outperforms uncertainty-based sampling strategies.

MedAL [35] and O-MedAL [12] apply the active learning framework to medical image analysis. Samples are queried via a hybrid strategy that includes a diversity-based metric, taking into account the distance between feature embeddings. MedAL generates embeddings $\psi(x)$ for a given sample x by extracting the activations of specific layers of the network. As a first step, MedAL computes the entropy of predictions $H(x)$ for all samples in the unlabeled pool $x \in \mathcal{U}$. It selects the subset of k samples with the highest entropy and constructs a new set $\mathcal{U}_e$. To ensure a diverse and representative batch of data, the hybrid strategy evaluates the average distance between each unlabeled sample $x \in \mathcal{U}_e$ and each labeled sample $x_s \in \mathcal{S}$. Only unlabeled samples belonging to the highest-entropy subset are considered.

The distance between sample representations $\psi(x)$ is measured using the Euclidean distance. The algorithm computes the average distance between each sample in $\mathcal{U}_e$ and all samples in the labeled pool $\mathcal{S}$, and selects the sample that maximizes this metric. Then, it adds the chosen sample to the labeled set $\mathcal{S}$, removes it from $\mathcal{U}_e$, and recomputes the scores. It repeats this procedure until the desired number of query samples is reached.

In [14], authors use a new algorithm, WI-DL, to select samples that maximize information density, following the idea that the most informative instances should not only be uncertain but should also be representative of the underlying data distribution. WI-DL proposes a similarity measure and an uncertainty measure to apply a query strategy based on information density.

Batch Active learning by Diverse Gradient Embeddings (BADGE) [36] considers predictive uncertainty and sample diversity in batch selection. At each DAL iteration, BADGE computes the loss gradient for each sample in the unlabeled pool. The basic intuition is that the larger the norm of the gradient embedding, the higher the uncertainty associated with the sample. Then, BADGE samples a random subset using the K-means seeding algorithm to obtain a batch of the desired size. This process is repeated until the model reaches a satisfactory performance. This algorithm does not require any manual hyperparameter tuning to balance uncertainty and diversity of examples, which allows for a more straightforward application to real-world problems.

WAAL (Wasserstein Adversarial Active Learning) [37] proposes a training loss that considers network parameter optimization and batch selection, based on Wasserstein distance. WAAL considers active learning as a distribution matching problem. According to the authors, DAL is a process that follows a given distribution $\mathcal{Q}$ different from the underlying data distribution $\mathcal{D}$. WAAL balances between uncertainty and diversity of the queried batch, and is regarded as a hybrid strategy.

BatchBALD [38] is a deep Bayesian active learning framework based on BALD [30]. BatchBALD extends BALD to the batch-mode setting of DAL by estimating mutual information between a set of multiple samples and network parameters, rather than considering each sample individually. As expected, by considering the diversity of samples within the queried batch, BatchBALD obtains better results than standard BALD on multiple benchmark datasets.

2.1.3 Density-based methods

The key concept of density-based techniques is the idea of core-sets [39]. The authors define the active learning problem as a core-set selection problem. They aim at sampling the available dataset such that a model trained on the sample and a model learned on the complete training set produce similar outputs. Their work focuses on defining a loss function that incorporates the core-set objective.

The work in [40] applies the idea of core-sets to palmprint recognition, a real-world image classification problem that may be tackled via active learning to limit manual labeling costs. The basic idea is to model the true data distribution of the whole palmprint dataset using the labeled set only. To do so, authors train a binary classifier to distinguish between samples that belong to the current training set and those that belong to the unlabeled set instead. Their goal is to minimize the difference between the two distributions. Therefore, at each DAL iteration, the authors first train the discriminator on the two sets and then query the samples which the classifier predicts as belonging to the unlabeled set with the highest confidence.

It should be noted that, while [39] shows that the query strategy based on core-sets outperforms DAL baselines on several benchmark datasets, the authors of [41] cannot reproduce these results. Instead, [41] finds that the core-set approach does not improve on the baselines evaluated in the paper.

2.2 Explainable AI

Understanding the reasons behind the decisions taken by deep neural networks is a challenge due to their black-box nature. Explainability techniques developed by researchers in xAI provide tools to explain black-box models in a post-hoc manner.

According to [42, 43], model understanding is a fundamental problem that should be addressed when developing AI applications. Understanding model behavior facilitates the detection of human biases and prejudices that may be inherited from the massive amount of data on which machine learning models are trained. Model understanding helps provide recourse to individuals who are negatively affected by model predictions. Moreover, transparency of AI systems is enforced with laws such as the General Data Protection Regulation (GDPR) and the right to explanation [44, 43].

Defining the concept of model explanation is not straightforward, and

different researchers address the problem from a different perspective [43, 45]. Authors of [42] define explanation as an interpretable description of the model behavior and provide two criteria for the validity of an explanation: it must be faithful to the model and understandable by the end-user.

There are several types of xAI techniques that operate in different ways. Most importantly, we can distinguish between global and local explanation methods. Global explanations provide an overview on the complete model behavior, while local explanations focus on the prediction of a particular instance [42].

Our focus lies on local post-hoc explainability techniques, as we are interested in understanding the reasons behind specific predictions. In particular, we aim to explain the model behavior in the local neighborhood of samples belonging to the unlabeled pool in the DAL setting. The main explainability techniques that fall under this category are based on feature importance, rules, saliency maps, prototypes, counterfactual explanations, and adversarial examples.

2.2.1 Feature importance

Explainability techniques based on feature importance aim to understand the sample features that contribute the most to a given prediction. The basic approach is to perturb selected features in the input and evaluate the change in model prediction caused by the perturbation.

LIME (Local Interpretable Model-agnostic Explanations) [46] is a model-agnostic local explainability technique. It works by training an interpretable surrogate model locally around the prediction. LIME employs interpretable data representations x' that are understandable to humans and are generally different from original representations x. In the context of image classification, an interpretable feature may be a binary indicator mapped to image pixels, indicating the presence of a concept inside the image, e.g., a specific object. LIME learns the local behavior of the original model by perturbing the original input and obtaining the label for the perturbed samples. LIME is a customizable framework, allowing it to be adaptable to any domain. However, high customizability translates to a high number of hyperparameters to optimize. Potentially, tuning hyperparameters may allow for arbitrary changes to explanations, which can be considered a drawback of this method [42].

SHAP (SHapley Additive exPlanations) [47] is a different approach to

explainability that considers feature importance, based on the concept of Shapley values [48]. SHAP computes the marginal contribution of each feature towards the model prediction, averaged over all possible permutations.

2.2.2 Rule-based techniques

Rule-based explainability techniques work by defining sets of rules that describe model behavior.

Anchors [49] aim to identify the local conditions under which the classifier predicts the same class. Anchors take the form of sets of rules that are sufficient for predicting a particular label with high probability for a specific sample. Notably, anchors are model-agnostic. Similarly to LIME [46], anchors consider an alternative interpretable representation of the input.

In the context of image classification, anchors are defined by fixing a set of interpretable features and superimposing different images over the pixels not belonging to the interpretable feature set. The images obtained with this procedure describe the sufficient conditions for obtaining the prediction $f(x)$, providing insights on the characteristics required for predicting the class and possibly highlighting weaknesses in the model.

2.2.3 Saliency maps

Saliency maps, or sensitivity maps, are a particular kind of explainability technique based on feature importance.

Grad-CAM (Gradient-weighted Class Activation Mapping) [50] produces visual explanations for CNN-based models. Grad-CAM uses the gradients of any target concept to produce a heatmap that highlights the most important regions in the image for predicting the concept. SmoothGrad [51] is a simple technique that helps to visually sharpen saliency maps, such as those extracted via Grad-CAM. The experiments described by the authors suggest that small perturbations applied to samples can sharpen gradient-based saliency maps.

DeepLIFT (Deep Learning Important FeaTures) [52] is a method to obtain contribution scores to the inputs of the network for a given output. This method calculates the importance in terms of difference from a reference state, which usually represents a neutral input, depending on the specific domain of the task at hand. This method addresses the limitations caused by rectifying gradients during backpropagation, as done by Grad-CAM [50].

2.2.4 Prototypes

Prototypes or examples-based explainability techniques are based on selecting a set of instances, namely prototypes, that are representative of the training dataset [44]. Influential instances are data points from the training set that influence the determination of model parameters and, consequently, the prediction.

A possible approach to find a set of prototypes is to rank samples belonging to the training set based on their influence on the test loss. This method allows the end-user to understand if the model relies on the correct semantic type of features to determine the class of interest [42].

Influence function is a classic tool used in robust statistics for assessing the effect of a sample on regression parameters [53]. To assess the importance of a sample of interest, instead of fitting the model from scratch using all training data except the sample under analysis and evaluating it, we can use Cook's distance as an analytical alternative. The authors of [54] extend Cook's distance insights to modern machine learning.

Activation maximization approaches identify samples that strongly activate a function of interest. In the context of image classification, the function may be a neuron in the CNN under analysis. Samples may be synthetic, i.e., created ad-hoc to strongly activate the function of interest, or natural, i.e., belonging to a specified set of real data points. The work in [55] explores activation maximization to understand how CNNs build up their understanding of the underlying concepts in images over their layers.

TracIn [56] computes the influence of training samples of interest by keeping track of the path of the loss during the training procedure. The work in [57] proposes representer points from the training set as a set of points to explain the predictions of a deep neural network for a given test instance.

A limitation of prototype-based techniques is that it is difficult to understand the cut-off between influential and non-influential samples [44]. A possible definition could be based on the distance of samples from the decision boundary in a classification setting.

2.2.5 Counterfactuals

As machine learning models are increasingly deployed for high-stakes decisions, it becomes essential to provide recourse to affected individuals [42]. To address this problem, we can use counterfactual explanations.

Counterfactual-based explainability techniques assess how and by how much the features of a sample of interest need to change in order for the model prediction to change, e.g., flipping from a negative to a positive outcome.

Counterfactual generation algorithms generally differ on two points: the choice between candidate counterfactuals and how much access is needed to the underlying predictive model [42]. Each counterfactual explanation considers a different perspective on reaching a prediction of interest, i.e., there may be multiple ways to change the model prediction. It is non-trivial to choose the best one [44].

One of the problems to tackle when generating a counterfactual is to produce feasible recourses. Simply put, the features to change for the model prediction to flip should be actionable. The authors of [58] define an optimization framework that includes a cost function to choose between candidate feasible actions, i.e., to select the best counterfactual within a constrained set of feasible counterfactuals. Notably, feasibility is defined by the end-user and is domain-specific. A limitation of the approach in [58] is that it applies to linear classifiers only.

When assessing the feasibility of recourses, it should be considered that changing one feature without affecting other features may be impossible [42]. The work in [59] addresses this issue by preserving causal relationships among input features via structural causal models. The authors propose a proximity loss based on a causal graph that can be added to any counterfactual generation method. When assessing the practical applicability of their approach, they find that a partial causal graph works as well. Moreover, when it is impossible to express feasibility with simple constraints, they propose a method that learns feasibility via user feedback on counterfactuals.

2.2.6 EBAnO

EBAnO (Explaining BlAck-box mOdel) [16, 15][1] is an engine for explaining the decisions of a particular class of black-box algorithms in the context of image classification. EBAnO analyzes the knowledge contained in the convolutional layers of CNNs in an unsupervised manner. Notably, EBAnO does not employ surrogate models. While EBAnO can produce

[1]The description of EBAnO refers to a new unpublished version and differs from the one provided by the cited papers.

both local and global explanations, our interest mainly lies in locally explaining model predictions. EBAnO computes local explanations for a given *class-of-interest*, showing what parts of the original image impact the prediction of the class under analysis. Figure 2.2 describes the explanation process of EBAnO.

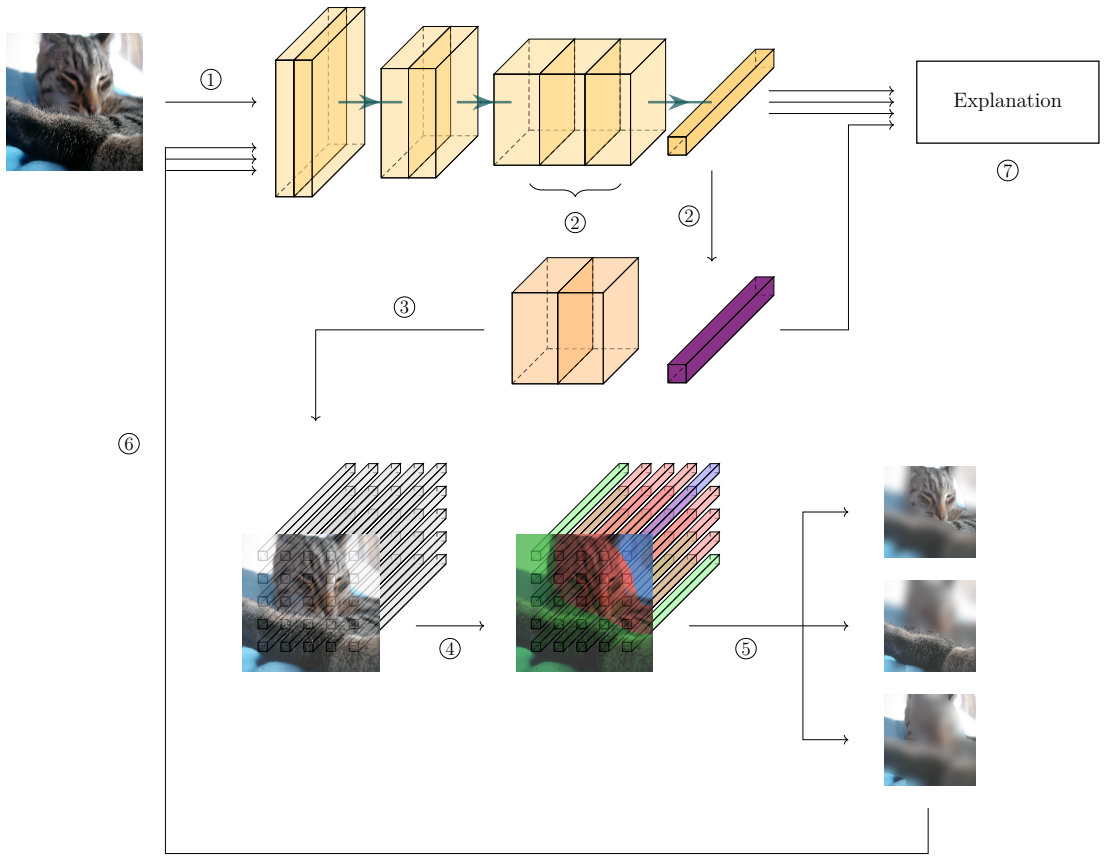

Figure 2.2: Local explanation process of EBAnO. An image is fed to a CNN ① and its output is collected along with the activations of specific layers of interest ②. Hypercolumns are extracted from the activations ③ and are processed in an unsupervised manner to extract interpretable features ④. Each interpretable feature is perturbed to produce a new set of perturbed images ⑤. The set of perturbed images is fed to the CNN ⑥ and the output of the network is processed to produce local explanations ⑦.

First, an image of interest is fed to the black-box CNN model under analysis ①. The output of the network, i.e., the probability distribution

over classes for the sample, is collected. Moreover, the activations of specific layers of the CNN are extracted ② in order to produce hypercolumns ③. Hypercolumns [60] are a vector containing the activations of the model for a specific sample. Each input pixel is mapped to a hypercolumn representing the activations of the units above that pixel. To generate a hypercolumn for each pixel, the extracted convolutional filters are scaled via bilinear interpolation to fit the original input image size. Hypercolumns are extracted from a subset of the convolutional layers of the network. The choice of layers to analyze is a hyperparameter to tune.

EBAnO indentifies meaningful sections of the image under analysis via unsupervised clustering of hypercolumns ④. These image portions are called *interpretable features* and represent concepts that are easily interpretable by humans and reflect the inner knowledge of the network with high fidelity. Clustering of hypercolumns is performed using the K-Means algorithm [61]. Since the number of clusters, i.e., interpretable features, is unknown at evaluation time, clustering is repeated multiple times for different k. Given the set of k candidate explanations, EBAnO chooses the most informative one.

To measure the influence of each interpretable feature $f \in F$ on the prediction, EBAnO generates a set of $k = |F|$ perturbed images ⑤. In particular, each interpretable feature is perturbed while the rest of the image remains unchanged. Each perturbed image is fed to the CNN ⑥ and predictions p_f are collected. At this stage, for each perturbation, three scenarios may happen regarding the prediction of the class-of-interest $ci \in C$: the network does not change its prediction, the probability of the predicted class decreases, or the probability of the predicted class increases.

Using this information, EBAnO defines two indices that measure the influence of each interpretable feature on the prediction, namely the *Normalized Perturbation Influence Relation* (nPIR) and the *Normalized Perturbation Influence Relation Precision* (nPIRP) ⑦. The nPIR measures the impact of an interpretable feature on the prediction of the class-of-interest for a given sample. The nPIRP measures the precision of the impact of an interpretable feature over class labels, i.e., it measures if an interpretable feature f influences the prediction of only one or more than one class. Figure 2.3 qualitatively describes the meaning of different values of nPIR and nPIRP indices.

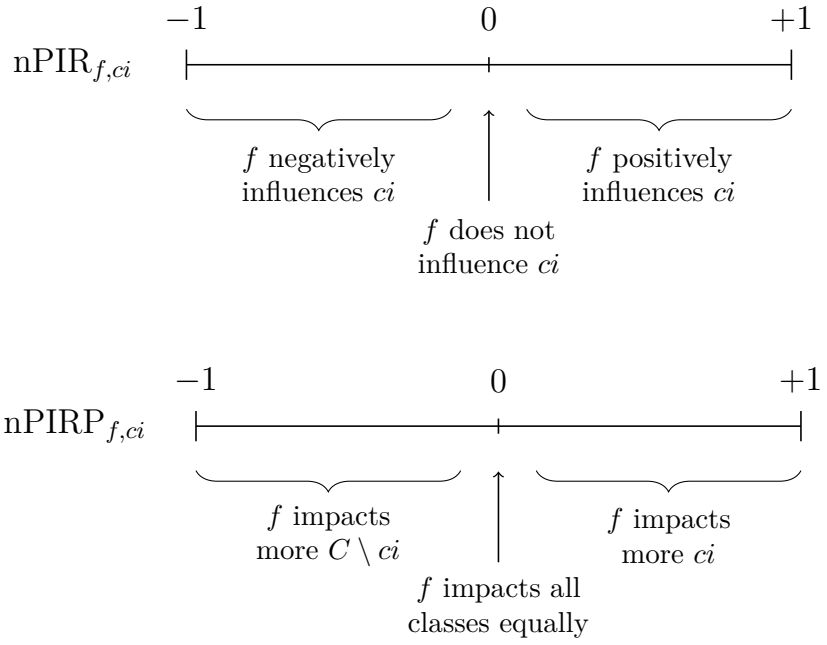

Figure 2.3: Qualitative description of the meaning of nPIR and nPIRP indices for a particular feature of interest $f \in F$ and class-of-interest $ci \in C$. Lower nPIR values correspond to negative influence of f on the class-of-interest, i.e., by perturbing f, the predicted probability for ci is higher, and vice versa for higher nPIR values. Lower nPIRP values correspond to f impacting more other classes $C \setminus ci$ rather than the class-of-interest, while higher nPIRP values correspond to f impacting precisely only on the prediction of the class-of-interest.

The current implementation of EBAnO for local explanations is EBAnO-Express[2]. A major drawback of this implementation consists in its time inefficiency. We will discuss this problem and present a novel implementation of EBAnO in Section 3.2.

[2] `https://github.com/EBAnO-Ecosystem/EBAnO-Express`

2.3 Explainability techniques in deep active learning

Recently, researchers began to integrate model explainability techniques in the active learning paradigm. There are two different approaches to this research field. One approach focuses on developing human-AI interfaces and providing tools to improve model teaching experience [17]. The other approach is to directly integrate information provided by explainability techniques into the DAL framework, either in a supervised or unsupervised manner.

The strategy adopted by [62] is both model-agnostic and explainer-agnostic. The authors use LIME [46] to generate explanations for examples queried at each DAL iteration. Their algorithm, CAIPI, takes as input the set of labeled samples $\mathcal{S}$, the set of unlabeled instances $\mathcal{U}$, and the iteration budget. At each DAL step, the framework applies LIME to explain the predictions of the current model for the queried samples. Besides query samples and relative predictions, CAIPI presents the generated explanations to the annotator. The user provides the true label and, eventually, the correction to the explanation c. In case the model is right for the wrong reasons, as determined by the annotator, the algorithm asks the end user to provide an explanation correction to c. The explanation correction is explained back to the learner by employing counterfactuals, effectively augmenting the training dataset at each active learning step. The authors apply their framework to an image classification task using deep neural networks. Via their experimental results and a user study, they find that their technique improves model performance and, most importantly, encourages trust in the model via explainability techniques.

The authors of [63] introduce ALEX (Active Learning based Enhancement of a Model EXplainability). They propose to train an explainer model, i.e., a surrogate model, in addition to the classifier model during each active learning iteration. The purpose of the explainer model is to query the instances where, on average, a different part of the data is used to explain the predicted class. The authors aim at labeling the instances that are most difficult to explain, by measuring the divergence between each instance in the unlabeled pool and the explanation vectors estimated for labeled samples contained in the training set. By selecting the instances with the highest divergence, the authors find that the performance of ALEX is mostly above

baseline methods.

Research in DAL is rich, and many query strategies with different approaches have been developed in recent years. However, the usage of local and global xAI techniques in the DAL context is still an emerging field of study. While many works focus on implementing xAI to improve the experience of annotators [17], to the best of our knowledge, no existing work injects knowledge extracted from explanations into a DAL query strategy in an unsupervised manner. Our work positions itself in this research gap.

Chapter 3

Methodology

This chapter describes the components of our experimental setup, discusses the proposed methodology for applying EBAnO to the DAL framework, and presents our novel query strategies.

3.1 Envisioned explainability framework

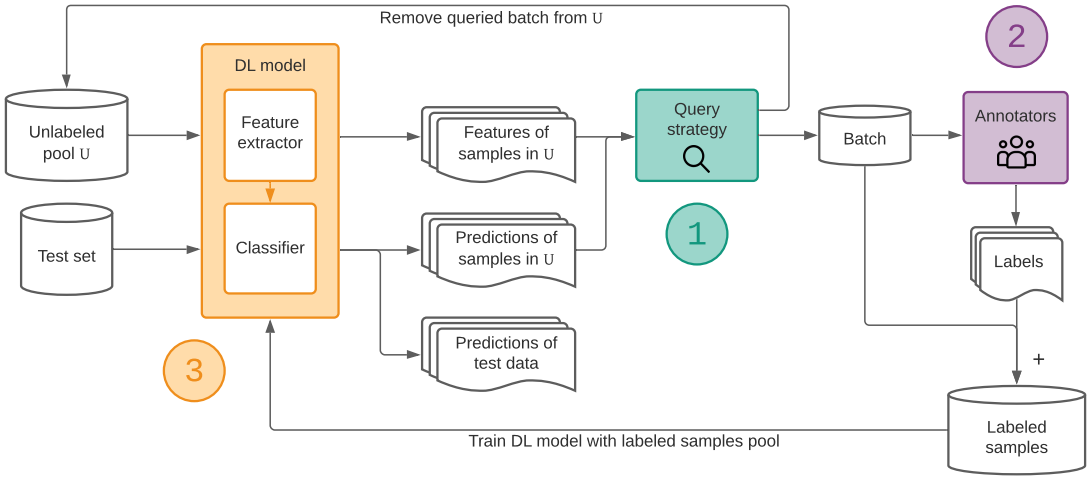

Figure 3.1: Our vision for a complete explainability framework. Explainability methods can be applied to different components of the DAL framework: unsupervised query strategies ①, annotation of queried samples ②, and deep neural network training procedure ③. This work focuses on component ①.

We propose a complete explainability framework applicable to different parts of the DAL procedure, as shown in Figure 3.1. We envision this framework to be composed of three main parts.

① Application of an unsupervised query strategy, exploiting the information provided by EBAnO on the unlabeled pool to select the samples that maximize the performance gain of the model at each iteration.

② Generation of global and local explanations via EBAnO at each iteration to validate the correctness of the current model and evaluate whether additional training steps are required.

③ Implementation of a training procedure for deep neural networks, exploiting information provided by EBAnO on the training samples, possibly under the form of a custom loss function.

Our work focuses on the first part of the proposed explainability framework. This thesis aims to design and implement a query strategy that selects samples in the training pool that maximize the model improvement, based on the explainability indices provided by EBAnO, in an unsupervised manner.

3.2 Explainability engine

EBAnO is the most fundamental component in our custom query strategy. As such, it is necessary to optimize its performance. The DAL framework works iteratively. Therefore, we must apply EBAnO to the whole unlabeled pool at each step. The current implementation of EBAnO for images that provides local explanations, EBAnO-Express, processes one image input at a time. This behavior severely limits the explanation throughput and becomes a problem when we need to fetch explanations for tens of thousands of images.

To tackle this issue, we propose BatchEBAnO, an optimized version of EBAnO for images that increases throughput by processing multiple images in batch mode. BatchEBAnO fundamentals are similar to the EBAnO-Express implementation. However, we apply some measures to obtain comparable results in a shorter amount of time. A fast and efficient implementation of EBAnO is of fundamental importance to provide explanations as

quickly as possible during DAL while retaining a good quality of explanations.

3.2.1 BatchEBAnO

The EBAnO framework for local explanations comprises five fundamental components: an image input pipeline, a hypercolumn extractor, a perturbation procedure, an interpretable feature extractor, and a numerical and visual explanation procedure. BatchEBAnO efficiently implements these components and speeds up the entire process.

Our optimized engine is based on *numpy*, a library for efficiently manipulating multi-dimensional array objects [64]. Most operations exploit the preallocation of contiguous arrays in memory to allow faster reads and writes without incurring the overhead cost of Python objects.

The image input pipeline of EBAnO-Express consists of feeding a single image at a time to a deep neural network model to extract the activations of a subset of convolutional layers and obtain output predictions. BatchEBAnO speeds up this step by feeding a batch of images to the network instead of a single image, exploiting the parallelization power of GPUs and radically decreasing the time required to extract activations at the expense of higher GPU memory usage.

Our engine performs hypercolumn extraction in a similar way to EBAnO-Express. Convolutional layer activations are resized to match the resolution of the input image so that each pixel is mapped to a hypercolumn. Each hypercolumn has a fixed number of features, depending on the number of convolutional layers extracted and their respective kernels. We analyze the last three convolutional layers of our network.

Hypercolumns often contain redundant information that can be expressed via a lower number of dimensions with a smaller memory footprint. At this stage, the main difference between the two engines is the dimensionality reduction technique they employ. EBAnO-Express reduces the number of dimensions of hypercolumns via principal component analysis (PCA), while our engine uses truncated singular value decomposition (T-SVD).

BatchEBAnO fits T-SVD to a subset of hypercolumns. In other words, we choose a random subset of $n < h \cdot w$ hypercolumns, with h the image height, w the image width, and $h \cdot w$ the total number of hypercolumns. This technique significantly speeds up the dimensionality reduction phase

while retaining a good overall quality of hypercolumns, as we show in Section 3.2.2. Here, n is a hyperparameter and must be tuned for different network architectures. The choice of using T-SVD instead of PCA is due to faster computation on our particular image format and number of features. At this stage, each image must be processed separately. Therefore, even a minimal speed-up will compound over hundreds of thousands of images.

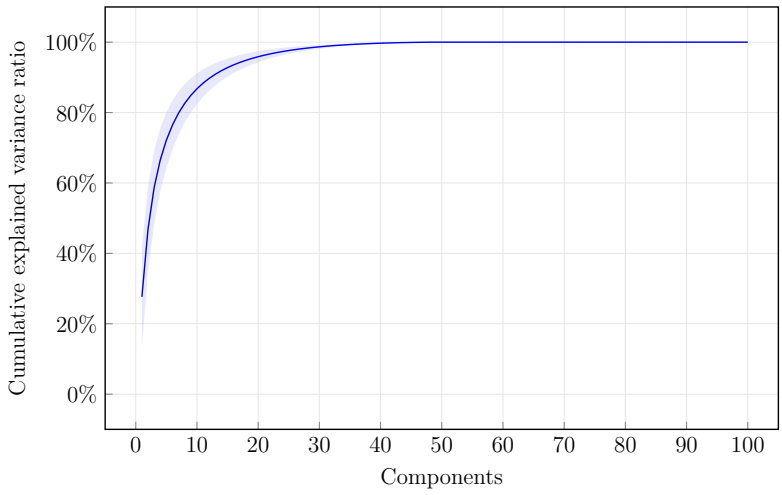

Figure 3.2: Mean and standard deviation of the cumulative percentage of variance explained by each of the 100 selected components over a random subset of ImageNet composed of 5000 images, with five images per class. Hypercolumns are extracted via ResNet-50, using its last three convolutional layers. As we can see, on average, the variance ratio explained by additional components quickly decreases after 10.

To further optimize the performance of hypercolumn extraction, we fine-tune the number of hypercolumn features retained after dimensionality reduction. Figure 3.2 shows the average cumulative percentage of variance explained by a hundred components on a subset of ImageNet. We find that ResNet-50 requires only ten components to explain most of the variance inside the hypercolumns.

EBAnO-Express performs an unsupervised clustering analysis of hypercolumns via the K-means algorithm provided by *scikit-learn* [65]. Instead, BatchEBAnO uses Faiss, a library for efficient similarity search and clustering of dense vectors [66] developed by Facebook AI. This provides a significant speed-up in clustering the reduced hypercolumns.

During the image perturbation phase, EBAnO masks every interpretable feature of each image independently and produces a set of perturbed images. While EBAnO-Express feeds one masked image at a time, BatchEBAnO exploits GPU parallelization capabilities to compute predictions for the whole set of perturbed images at once.

BatchEBAnO trades off memory to significantly speed up computations. While EBAnO-Express requires constant memory as it processes one image at a time, the memory footprint of BatchEBAnO increases linearly with the batch size. The number of images inside batches is a hyperparameter that must be tuned depending on the resources available on each specific machine. Notably, the batch size does not affect the quality of explanations.

In the following section, we compare the processing time of our optimized engine against the original implementation and compare the quality of the interpretable features extracted by EBAnO-Express and BatchEBAnO.

3.2.2 Systematic comparison to EBAnO-Express

We estimate that fine-tuned BatchEBAnO reduces ten-fold the time required to process the same amount of images on ResNet-50. Figure 3.3 shows the results of an experiment that processes six batches of images of size 1, 8, 16, 32, 64 and 128 respectively. BatchEBAnO consistently outperforms EBAnO-Express.

The clustering quality retained by BatchEBAnO in our setting is similar to the one provided by the original implementation. To systematically compare the quality of the extracted interpretable features, we consider two different metrics: *adjusted rand-index* (ARI) and *mean silhouette score*. Our experiments are carried out on a balanced sample of five thousand images from our dataset of choice, ImageNet [7]. Chapter 4 describes the dataset in more detail.

As a first step, we compare the similarity between the clusterings of the two engines by measuring the ARI. Given two cluster assignments c_a and c_b, ARI is a measure of similarity between c_a and c_b. We compute the ARI for each image. Notably, it may happen that while EBAnO-Express puts a given label on an interpretable feature, our optimized engine chooses a different label. The rand-index is suitable for our purpose as it ignores label permutations.

Figure 3.4 shows the distribution of ARI values obtained for each particular number of clusters k, i.e., for each fixed number of interpretable features.

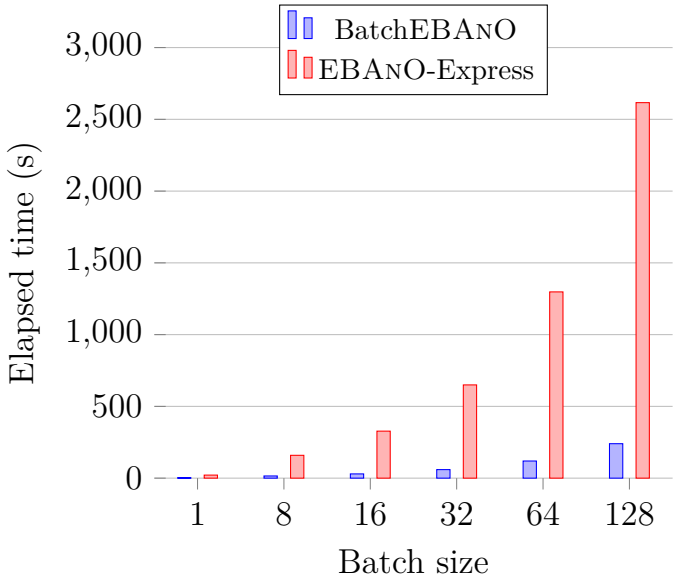

Figure 3.3: Comparison of elapsed time between EBAnO-Express and BatchEBAnO to process a batch of images of variable size. BatchEBAnO decreases the time requirements ten-fold, while retaining good cluster quality. The histogram shows the mean over three independent experiments performed on the same machine. The mean elapsed time to process a single-image batch is 21.22 seconds via EBAnO-Express against 2.63 seconds via BatchEBAnO.

Moreover, it presents the results considering only the best clustering, i.e., the best number of features k, independently chosen by each engine.

To further assess the quality of clusters, we consider the mean silhouette coefficient (or silhouette score) of each image. The silhouette score helps validate the consistency of interpretable features within an image. Given an image I and its set of hypercolumns $h \in I$, the silhouette score $s(h)$ considers the following metrics.

1. The mean distance $a(h)$ between h and all other hypercolumns inside the same feature cluster, i.e., the mean dissimilarity between samples inside the same cluster.

2. The mean distance $b(h)$ between h and all other hypercolumns in the nearest cluster, i.e., the lowest average dissimilarity between h and any other cluster.

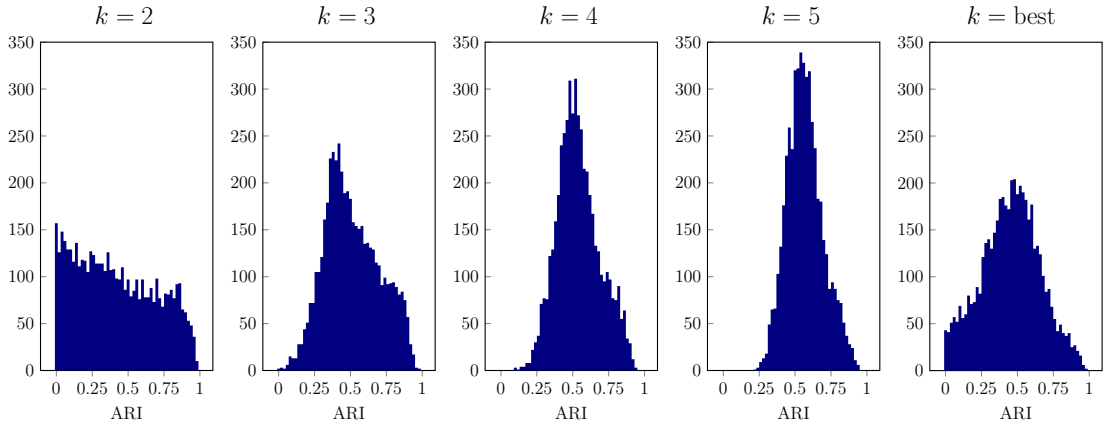

Figure 3.4: Adjusted rand-index distribution over a sample of 5000 images. Each image is clustered by EBAnO-Express and by BatchEBAnO. In the first four plots, we fix the number of interpretable features k and show the results for the whole sample. On average, the adjusted rand-index is higher for larger k. In the last plot on the right, only the best clustering is considered.

The distance between hypercolumns is measured as the Euclidean distance. The silhouette coefficient $s(h)$ is defined as follows. It ranges from -1 to 1, with -1 being the worst score and 1 the best value.

$$s(h) = \frac{b(h) - a(h)}{\max(\{a(h), b(h)\})}$$

For each individual image I, we compute the mean silhouette score of clusters generated by BatchEBAnO $\text{ms}_B(I)$ and the mean silhouette score of clusters generated by EBAnO-Express $\text{ms}_E(I)$. The mean silhouette scores are computed independently for each engine. Notably, the hypercolumns computed for an image differ from one engine to another. Only the best clustering from each engine is considered.

To quantitatively assess the difference between the clustering produced by EBAnO-Express and the one by BatchEBAnO, we compute the difference ms_{diff} between the scores associated to each image. Formally, the difference is defined as follows.

$$\text{ms}_{\text{diff}}(I) = \text{ms}_B(I) - \text{ms}_E(I)$$

Figure 3.5 shows the distribution of $\text{ms}_B(I)$ and $\text{ms}_E(I)$ over all samples in our ImageNet sample, separately for the two engines under analysis.

Moreover, it shows the distribution of their difference ms_{diff}.

Note that, to speed up computations, the mean silhouette score for any image I is measured on a random subset of 5000 hypercolumns $h \in I$, i.e., using only 10% of the available data. Nonetheless, the results are similar to those obtained when using the whole set of hypercolumns. On a random subset of 100 images, the mean difference between the silhouette score obtained using a random subset of hypercolumns from BatchEBAnO and the silhouette score computed on the whole set of hypercolumns is $2.728 \cdot 10^{-5} \pm 0.003$.

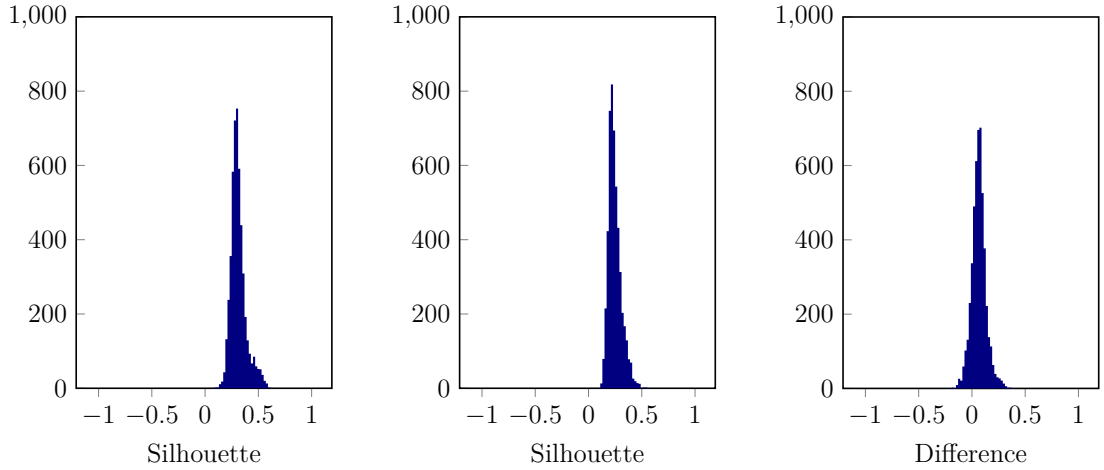

Figure 3.5: Distribution of the mean silhouette score over a sample of 5000 images. On the left, mean silhouette computed by BatchEBAnO. In the center, mean silhouette computed by EBAnO-Express. On the right, we present the distribution of the difference between silhouette scores obtained for the same image.

As highlighted by the ARI score, the clusterings performed by EBAnO-Express and BatchEBAnO on the same image are not perfectly similar. This behavior is due to the fact that the information contained in hypercolumns extracted by BatchEBAnO is less precise than EBAnO-Express, due to the lower number of features retained. However, Figure 3.5 shows that, on average, the quality of clusters extracted by BatchEBAnO is comparable to the quality of clusters of the original engine, as measured by the mean silhouette score. These results highlight that while there is a difference between clusterings of the same image by the two engines, the quality of clusters extracted by BatchEBAnO is perfectly suitable for our purposes.

As shown by these experiments, BatchEBAnO significantly reduces the

time required to extract explanations from images while retaining a good quality of interpretable features. We believe that BatchEBAnO will be helpful in each part of our envisioned explainability framework, and especially for extracting information from data in an unsupervised manner in a DAL setting.

3.3 Deep active learning

DAL aims to reduce the cost of training a deep neural network by annotating the set of samples that maximize the model performance as quickly as possible. The DAL framework is described in Algorithm 1.

Algorithm 1 Incremental DAL framework

Require: Initial labeled set $\mathcal{S}_0$, unlabeled pool $\mathcal{U}_0$, test set $\mathcal{T}$, iterations n, query strategy Q, batch size q

$\theta_0 \leftarrow$ TrainNetwork$(\mathcal{S}_0, \theta_{\text{rand}})$ ▷ Train the initial network

for $i \leftarrow 1, ..., n$ **do**

  $\mathcal{B}_i \leftarrow Q(\mathcal{U}_0, q)$ ▷ Select q samples from $\mathcal{U}_0$

  $\mathcal{L}_i \leftarrow$ AnnotateSamples$(\mathcal{B}_i)$

  $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \cup (\mathcal{B}_i, \mathcal{L}_i)$ ▷ Add queried samples to $\mathcal{S}_i$

  $\mathcal{U}_i \leftarrow \mathcal{U}_{i-1} \setminus \mathcal{B}_i$ ▷ Remove queried samples from $\mathcal{U}_i$

  $\theta_i \leftarrow$ TrainNetwork$(\mathcal{S}_i, \theta_{i-1})$

  $e_i \leftarrow$ EvaluateNetwork$(\mathcal{T})$

end for

The initial network is trained on the labeled training set $\mathcal{S}_0$. At the beginning of the first iteration, the network predicts the labels $\tilde{\mathcal{Y}}_0$ of samples in the initial unlabeled pool $\mathcal{U}_0$. The predicted labels and, eventually, additional information on the unlabeled pool are fed to the query strategy Q that selects the most informative batch of examples $\mathcal{B}_0$ of size q according to predefined metrics. Annotators label the chosen batch of samples. This procedure is costly: the larger the batch, the higher the cost incurred. The new batch of labeled samples is added to the labeled training set of the following iteration, $\mathcal{S}_1$. At the same time, samples belonging to $\mathcal{B}_0$ are removed from the unlabeled pool before the next iteration to form $\mathcal{U}_1$. Network parameters are updated with S_1. The network is evaluated on a test set $\mathcal{T}$ and the process is repeated from the beginning until it reaches a stopping condition.

Notably, choosing appropriate stopping criteria is still an open problem in active learning literature [17]. If no labels are available at the beginning of training, the annotators may choose a random subset of images from the unlabeled pool $\mathcal{U}$ to form $\mathcal{S}_0$, while removing the labeled samples from the pool to form $\mathcal{U}_0$.

There are two main strategies to train the network on newly annotated samples: fine-tuning the current model or retraining from scratch [19]. In our work, we fine-tune the network using all the available labeled samples. Fine-tuning allows us to train the network for fewer epochs at each iteration, reducing computational costs.

3.3.1 Baselines

We compare our proposed query methods with four different baseline strategies. We test the performance of three classic uncertainty sampling methods: least confident sampling, margin sampling, and maximum entropy sampling. Moreover, we include the random sampling baseline as a training lower bound.

Due to the usage of neural networks, an essential requirement in the DAL setting is that the query strategy must query a batch of samples, rather than a single example, at each iteration. Formally, the query strategy Q selects the batch $\mathcal{B}^*$ that maximizes a score among all possible batches from the unlabeled pool U.

$$\mathcal{B}^* = \underset{\mathcal{B} \subseteq \mathcal{U}}{\operatorname{argmax}}\, Q(\mathcal{B})$$

Uncertainty-based strategies require a probability distribution $p(y|x)$ over classes for each sample x in the current unlabeled pool $\mathcal{U}$. The softmax output of the network ϕ provides a proxy for this probability distribution.

Uncertainty-based strategies

The least confident query strategy [20] selects the instances whose predicted output is the least confident. Confidence is measured as the probability of the predicted class. Formally, we consider the probability of the most likely class $max_j p(y = j|x)$, i.e., the class with the highest softmax output, and select the samples $x^* \in \mathcal{U}$ with the lowest probability for the most likely class.

$$x^* = \underset{x \in \mathcal{U}}{\operatorname{argmin}}\, \max_j p(y = j|x)$$

Margin sampling [21] queries the examples with the lowest prediction margin. The prediction margin is defined as the difference between the highest output probability $p(y = j_1|x)$ and the second-highest one $p(y = j_2|x)$. The lower this difference, the higher the uncertainty associated with the sample prediction.

$$x^* = \underset{x \in \mathcal{U}}{\operatorname{argmin}}\, p(y = j_1|x) - p(y = j_2|x)$$

The entropy-based [22] method selects the samples with the highest entropy of output probabilities $H(x)$. The higher the entropy of the network output $\phi(x)$, the more uncertain the prediction for x. This method takes into account all class predictions.

$$H(x) = -\sum_j p(y = j|x) \log p(y = j|x)$$

$$x^* = \underset{x \in \mathcal{U}}{\operatorname{argmax}}\, H(x)$$

In every uncertainty-based baseline, we continuously select the best sample from the unlabeled pool $\mathcal{U}$ and remove it from $\mathcal{U}$ until the desired batch size is reached.

Random sampling

While these baselines apply a specific strategy to query a batch of samples, it is important to consider random sampling as well. In fact, random sampling has been regarded as a strong baseline [11] compared to other active learning strategies, and is a good proxy for a lower bound on accuracy.

3.3.2 Proposed query strategies

The goal of our work is to exploit the information provided by EBAnO to improve the classic uncertainty-based query strategies. To this aim, we study the nPIR and nPIRP indices extracted from a sample of ten thousand images using our CNN of choice, pre-trained on the full dataset from which examples were extracted, ImageNet [7]. We will describe the dataset and the network in more detail in Chapter 4. Based on our findings, we develop two novel query strategies: EBAnO-*Uncertainty* and EBAnO-*Augment*.

Analysis of indices

EBAnO works by computing an explanation for a class-of-interest $ci \in C$. Naturally, we do not have ground truth information for samples in the unlabeled pool. Therefore, we consider the predicted class $\text{argmax}_j\, p(y = j|x)$ to be the class-of-interest of sample x. Assuming that adding more uncertain or misclassified samples to the training set increases the performance of the network more rapidly, we want to understand if the indices extracted by EBAnO regarding the predicted class can give us information on whether a sample was classified correctly or not, and how certain the network is about its decision.

As a first step, we analyze the distribution of nPIR indices associated with the interpretable feature with the biggest influence on the prediction, i.e., the feature $f_{\max}$ with maximum nPIR, and the distribution of the nPIRP associated with the same feature. As we discussed in Section 2.2.6, the nPIRP index of $f_{\max}$ measures how precisely the most influent interpretable feature impacts the prediction. Figure 3.6 shows the distribution of these indices over all samples.

The distribution of the maximum nPIR of misclassified samples is similar to correctly classified samples. For most samples, our explanation engine extracts a single interpretable feature that strongly influences the prediction. An interesting behavior we observe is that the precision of $f_{\max}$ of misclassified samples is, on average, lower than in correctly classified samples.

Explanations are based on information extracted from the last layers of the network, which we hypothesize contain specialized information for the prediction. Therefore, we expect that features with high nPIR should only contain information that is precisely relevant to the predicted class. If the nPIRP of $f_{\max}$ is low for a given sample x, the network is uncertain about the most influential concept contained in that sample. This problematic sample may highlight a point of weakness of the network, which may be corrected by adding the problematic sample, along with its ground truth, to the training set.

Assuming the softmax output is a good proxy for the uncertainty of classification, our intuition is supported by the fact that, on average, a low nPIRP associated with $f_{\max}$ corresponds to an uncertain prediction. This behavior is highlighted in Figure 3.7. However, EBAnO provides additional information on the behavior of the network on each specific sample. Therefore, we believe that integrating EBAnO into an uncertainty-based query strategy

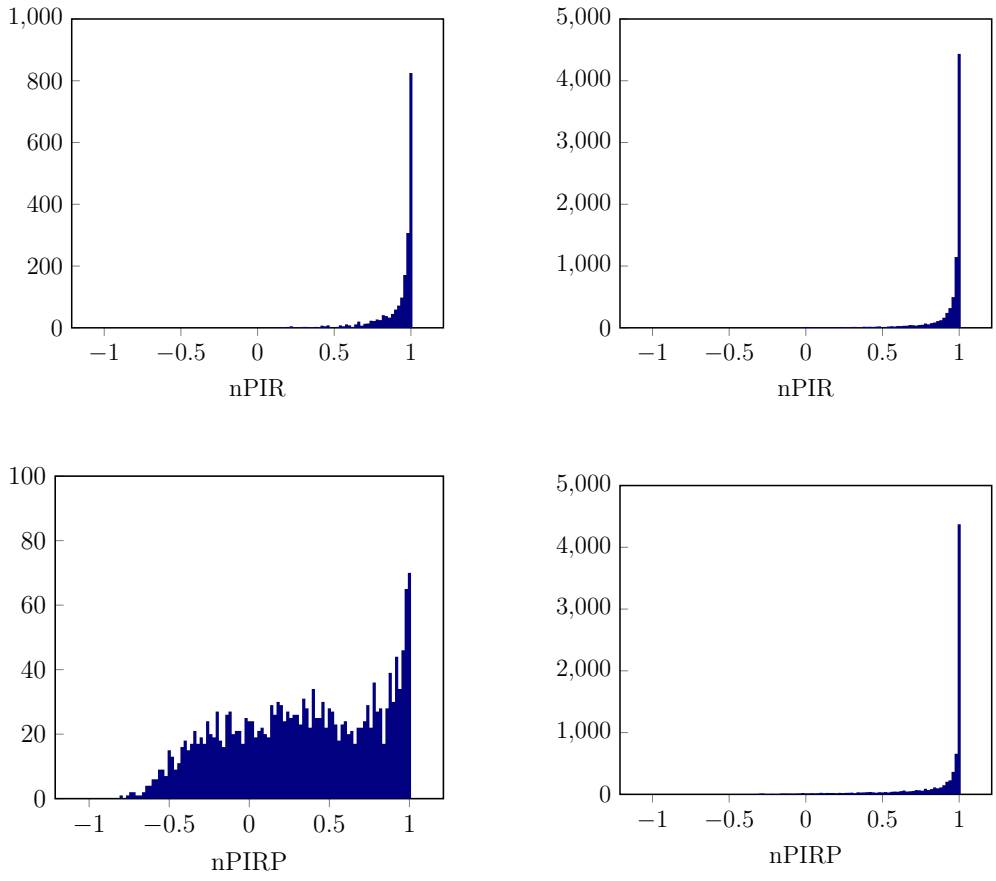

Figure 3.6: Distribution of nPIR associated to the most influent interpretable feature of each sample and nPIRP of the same feature. While there is no clear difference in the distribution of nPIR between correctly classified samples and misclassified ones, the precision of the most influent features is generally lower for misclassified samples.

will positively impact network performance over multiple DAL iterations.

EBAnO-Uncertainty query strategy

We introduce EBAnO-Uncertainty, a novel query strategy based on EBAnO. Our method selects the samples whose most influential interpretable feature has low precision. Formally, for each unlabeled sample x, EBAnO extracts a set of interpretable features F. We only consider the feature $f_{\max} \in F$ with maximum nPIR and measure its precision via the associated nPIRP

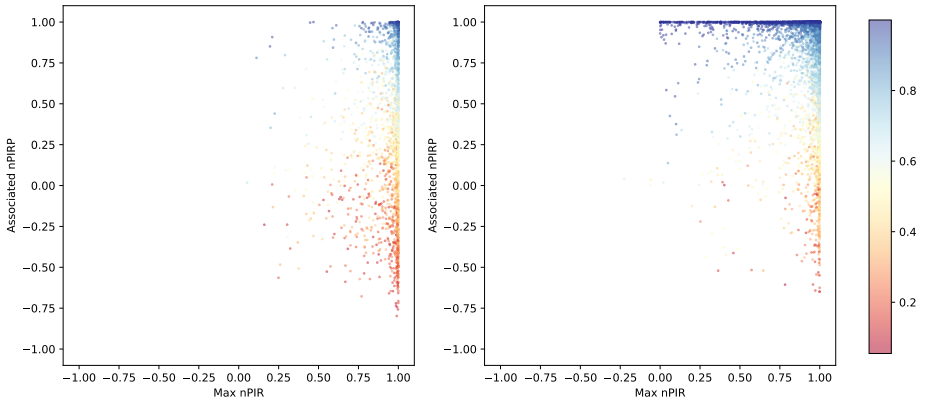

Figure 3.7: Scatterplot comparing maximum nPIR and associated nPIRP, i.e., the indices of the most influent interpretable feature, with the highest softmax output of the network. The plot on the left shows only misclassified samples, while the plot on the right shows correctly classified samples. Each point corresponds to an image inside the analyzed ImageNet subset. There is high correlation between the nPIRP index and the maximum softmax output.

index.

Let $\mathcal{Q}_\epsilon$ be the subset of samples of the unlabeled pool whose nPIRP associated with $f_{\max}$ falls below a given threshold ϵ. As a first approach, we may randomly take samples from $\mathcal{Q}_\epsilon$ until we reach the desired query batch size q. However, selecting an appropriate threshold ϵ is a non-trivial problem. Moreover, this strategy does not consider the magnitude of the influence of $f_{\max}$, nor the impact of $f_{\max}$ on other classes.

To address these issues, we introduce a ranking function m that assigns a score to each sample in the unlabeled pool. This function evaluates the difference between the nPIR of the most influential interpretable feature and its associated nPIRP. The larger this difference, the more influent $f_{\max}$ is on the prediction and the less focused it is on the predicted class. As both nPIR and nPIRP are bounded to the interval $[-1, 1]$, this metric provides a fair balance between these two concepts.

$$\mathrm{m}(x) = \mathrm{nPIR}_{f_{\max}}(x) - \mathrm{nPIRP}_{f_{\max}}(x)$$

The EBAnO-Uncertainty query strategy selects the samples x^* with the

highest score $\mathrm{m}(x^*)$ until the desired query batch size is reached. We exclude samples with a score under $\eta = 0.5$. This lower bound is a hyperparameter that may be fine-tuned depending on the specific task.

If EBAnO-Uncertainty did not query enough samples to reach the desired query batch size q, or if we reach an arbitrary query limit for our custom strategy $l_{\mathrm{U}} < q$, we include additional samples via highest-entropy sampling. The choice of using entropy as a complementary strategy stems from the intuition that, while EBAnO primarily focuses on a precise concept regarding the most influent interpretable feature of the explanation corresponding to the predicted class, entropy sampling considers the whole prediction probability distribution over all classes. We believe that applying both strategies in a complementary manner will yield a more diverse batch.

To limit the computational cost of evaluating a huge number of samples via BatchEBAnO, we consider only a random subset of κ samples from the unlabeled pool at each DAL iteration. Randomly sampling from the pool in a uniform fashion helps to select a representative sample of unlabeled data points.

The basic assumption is similar to the one made by classic uncertainty-based query strategies: more uncertain samples probably include information that is more relevant to the model during training. However, classic uncertainty-based methods only look at the softmax output of the model. We believe that EBAnO can extract more granular and richer information from the network and provide a better measure for sample uncertainty.

EBAnO-Augment query strategy

The second query strategy we present is EBAnO-Augment. This strategy is based on EBAnO-Uncertainty and queries samples using the same algorithm. However, it differs from the previous method as it perturbs a subset of queried images and interleaves them into the current training set.

EBAnO-Uncertainty queries the samples whose most influential interpretable feature $f_{\max}$ is not precisely focused on the class-of-interest, i.e., the predicted class. The network has not learned that the concept contained in $f_{\max}$ refers precisely to the ground truth. Our goal is to teach the network to recognize the most influential concept as relevant to the true class associated with the sample, with the intent of better modeling the true class and lowering the generalization risk.

To do so, for each sample x selected via EBAnO-Uncertainty, we preserve

the most influential interpretable feature $f_{\max}$ while perturbing the rest of the image via Gaussian blur. This procedure is similar to the one described in Section 2.2.6, albeit using an inverse perturbation mask. Figure 3.8 shows some examples of perturbed images generated from queried samples. The perturbed image is effectively a counterfactual example that attempts to correct a misbelief of the network or reinforce a correct belief. We impose a hard limit $l_{\mathrm{A}} \leq l_{\mathrm{U}} < q$ on the number of perturbed images to add to the training set at each iteration. Note that l_{A} is generally different from l_{U}.

EBAnO-Augment may seem similar to the CAIPI algorithm [62], an active learning algorithm based on integrating counterfactual explanations into the training set. However, unlike the algorithm described in [62], our method is entirely unsupervised and does not require explanation corrections from end-users.

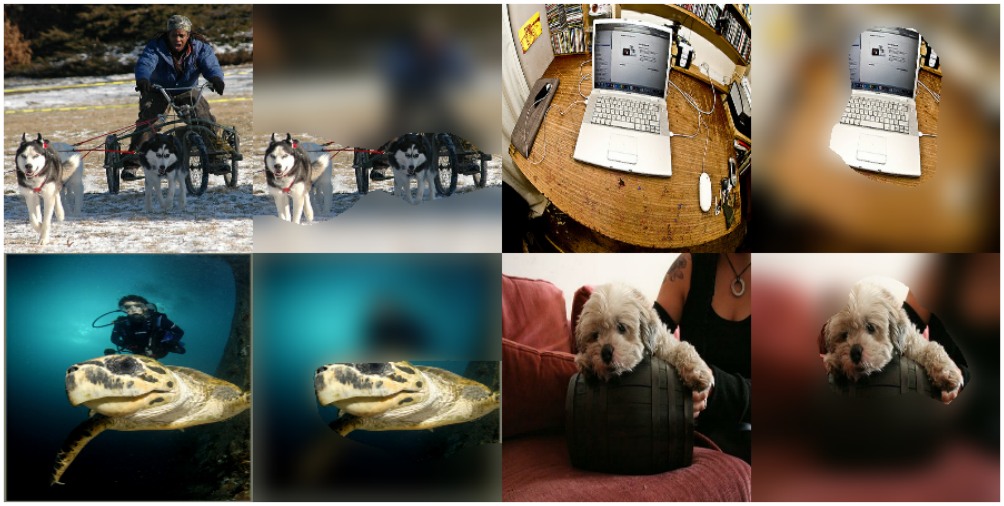

Figure 3.8: Examples of perturbed images produced by EBAnO-Augment. The original image is perturbed via Gaussian blur except for the portion corresponding to the most influential interpretable feature.

We should note that the current version of EBAnO-Augment has two potential drawbacks.

1. Samples with more than one subject may generate perturbed samples with incorrect labels. This problem may happen if the most influential interpretable feature for the predicted class is unrelated to the ground

truth. For instance, consider an image with ground truth *sub* containing two concepts: a sea turtle and a sub. If the most influential interpretable feature for the predicted class corresponds to the sea turtle, the perturbed image will contain a sea turtle with incorrect ground truth *sub*. This issue may be corrected by re-computing the explanation using the ground truth obtained after the annotation stage, in which case the network should choose the portion of the image containing the sub as the most influential feature and correctly perturb the rest of the image, including the sea turtle. We will tackle this problem in a future work.

2. Introducing perturbed images may cause a shift of the training set from the underlying data distribution. This effect has not been empirically quantified and will be considered in a future work.

Chapter 4

Experiments

This chapter describes the details of our experimental setup and presents the results of our tests.

4.1 Dataset

We perform our experiments on subsets of the widely-used ImageNet dataset [7]. ImageNet is a benchmark in object category classification and detection on hundreds of object categories and millions of images. More specifically, the ImageNet dataset contains 1000 object classes and a total of 14197122 annotated images organized by the semantic hierarchy of WordNet.

We use three non-overlapping subsets of ImageNet, with varying complexity and richness of content. The first subset contains 25 classes (ImageNet-25), the second subset is made up of 100 classes (ImageNet-100), and the third one contains 250 classes (ImageNet-250). The smallest subset is especially useful during the prototyping phase. It allows us to perform multiple tests on different query strategies over a short time frame. The other, larger subsets are useful for understanding whether our techniques are suitable for querying large data pools. The list of classes we used in each subset is available in the Appendix.

The DAL framework requires a further dataset subdivision. Each dataset must provide a dedicated split for training the initial model, a split that simulates the unlabeled pool, a test set for model evaluation at each iteration, and a validation set to monitor model performance during training. The initial training set is composed of 300 examples per class. The validation set is taken from the original ImageNet validation split. Unfortunately,

Dataset	$\lvert\mathcal{S}_0\rvert$	$\lvert\mathcal{U}_0\rvert$	Validation	Test
ImageNet-25	7500	22500	1250	2500
ImageNet-100	30000	90000	5000	10000
ImageNet-250	75000	225000	12500	25000

Table 4.1: Number of samples in each subset for ImageNet-25, ImageNet-100 and ImageNet-250. The initial training set is denoted by $|\mathcal{S}_0|$ while the unlabeled pool is $|\mathcal{U}_0|$. These figures refer to the initial DAL setting, i.e., iteration 0.

ImageNet does not provide labels for its test set. Thus, we hold out 100 images per class for testing. The remaining 900 samples per class simulate the unlabeled pool, of which the model does not know the labels unless they were queried previously. Table 4.1 provides a detailed summary of the components of each ImageNet subset.

We resize all images to a square box of size 224×224 using bilinear interpolation. This format is accepted as input by our chosen CNN architectures and reduces the memory footprint of the dataset at the expense of limited quality loss. These images retain a sufficiently high resolution, which is a crucial characteristic to test the performance of our custom query strategy powered by EBAnO. The EBAnO engine works by extracting several interpretable features from input images and perturbing them. Therefore, high-resolution images provide more details to EBAnO, which can extract relevant features. This characteristic should reflect positively on the performance of our custom approach.

4.2 Deep convolutional neural network

The backbone of our experiments is ResNet-50, a deep CNN that employs residual connections to address the vanishing gradient problem of very deep networks [10]. Figure 4.1 shows the basic building block of this residual network. As highlighted by the authors, residual connections provide a more stable training procedure and are easier to optimize than VGG-16 [9], which the authors of EBAnO used for their preliminary experiments in [16]. ResNet-50 reaches higher accuracy on ImageNet benchmarks with fewer parameters than VGG-16. We use the network implementation provided by

Keras [67].

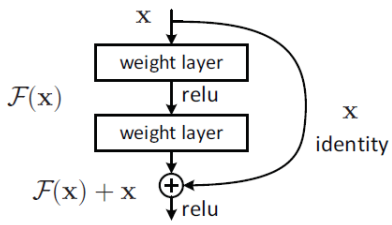

Figure 4.1: A residual shortcut inside ResNet-50, a fundamental building block of deep residual learning. The authors of [10] show that it is easier to optimize the residual mapping $\mathcal{F}(x) + x$ than to optimize the original mapping $\mathcal{F}(x)$.

The stable training of ResNet-50 allows us to compare different queries over multiple active learning iterations fairly. Figure 4.2 shows the mean training loss and validation loss on a random sampling baseline. The network can learn new information provided by additional training samples without incurring convergence problems.

All experiments on the same dataset share the same training hyperparameter setting to ensure a fair comparison. The goal of our experiments is not to obtain the best possible network accuracy per se but rather to show which query strategy performs best. Therefore, we are not interested in optimizing hyperparameters for each specific experiment.

4.3 Experimental setup

Initially, we produce a base model for each ImageNet subset. At this stage, the model follows the training procedure detailed in [10]. Notably, we optimize our model using stochastic gradient descent (SGD) with a mini-batch size of 256 and weight decay of 0.0001. We train each model for 400 epochs. The initial learning rate is 0.1 and decreases by a factor of 0.1 when the validation loss plateaus, for a maximum of two times. More specifically, the learning rate decays when the validation loss has not improved for 50 epochs. To limit computational cost and time requirements, the only data augmentation procedure we employ during training is a random horizontal flip with probability 0.5. The base model is shared across all experiments performed

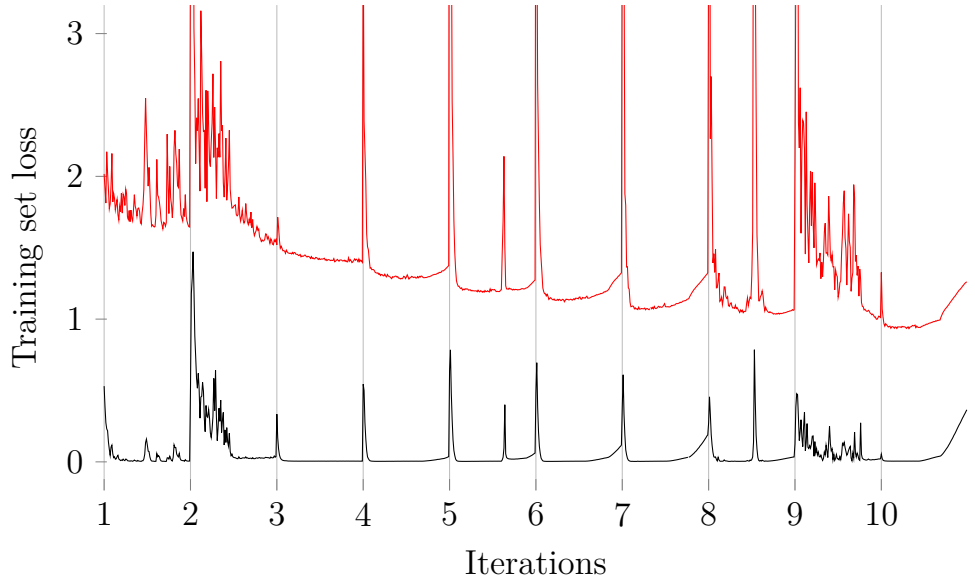

Figure 4.2: Training set loss (in black) and validation set loss (in red) over 1000 epochs and ten active learning steps with random sampling on ImageNet-25. Each active learning step trains the network for 100 epochs. The periodical loss spikes correspond to a new active learning iteration. However, the loss quickly recovers, and normal training resumes. Training on ImageNet-100 and ImageNet-250 shows a similar trend.

on the respective dataset to ensure fair comparison among different query strategies.

We perform τ active learning iterations. During DAL, we fine-tune the best network from the last iteration instead of retraining the model from scratch. The best model is defined as the network that minimized the validation loss in the previous training iteration. The training set is composed of all initial training samples and annotated samples until that point. The learning rate is set to 0.01 at the beginning of each active learning iteration. It decays by a factor of 0.1 when the validation loss does not improve for more than p epochs, for a maximum of two times. The network is trained for a total of e epochs per iteration. Table 4.2 presents the hyperparameter setting we employ for each ImageNet subset.

The number of queried samples q at each iteration is different for each dataset. In general, the larger the dataset, the higher the number of samples queried. At the same time, we simulate fewer DAL loops and train the

Dataset	Iterations τ	Epochs per iteration e	Patience p
ImageNet-25	10	100	25
ImageNet-100	8	75	20
ImageNet-250	6	60	20

Table 4.2: Network training hyperparameter setup for ImageNet-25, ImageNet-100, and ImageNet-250.

network for fewer epochs for larger datasets to limit computational costs of network training, as we describe in Table 4.2.

The baseline query strategies against which we compare our custom methods are described in Section 3.3.1. We consider *Random* (random sampling), *Margin* (largest margin sampling), *LC* (least confidence sampling), and *Entropy* (highest entropy sampling).

We test our custom query strategies EBAnO-Uncertainty and EBAnO-Augment with a set of four experiments for each ImageNet subset.

EBAnO-Uncertainty early mix The first $\tau/2$ query iterations are performed via EBAnO-Uncertainty, while the remaining $\tau/2$ iterations are carried out via highest entropy sampling. During the first $\tau/2$ iterations, we limit the number of samples queried via EBAnO-Uncertainty to $l_{\mathrm{U}} < q$ and fill the remaining slots via highest entropy sampling. The minimum score $m(x)$ is set to $\eta = 0.5$. Before applying EBAnO-Uncertainty, we take a subset of size κ from the unlabeled pool via random sampling.

EBAnO-Uncertainty late mix The first $\tau/2$ query iterations are performed via highest entropy sampling, while the remaining $\tau/2$ iterations are carried out via EBAnO-Uncertainty. The parameter configuration is the same one described in early mix.

EBAnO-Augment early mix The first $\tau/2$ query iterations are performed via EBAnO-Augment, while the remaining $\tau/2$ iterations are carried out via highest entropy sampling. During the first $\tau/2$ iterations, we limit the number of samples queried via EBAnO-Augment to $l_{\mathrm{U}} < q$ and fill the remaining slots via highest entropy sampling. We limit the number of perturbed images to $l_{\mathrm{A}} \leq l_{\mathrm{U}} < q$. Before applying EBAnO-Augment, we randomly sample a subset of size κ from the unlabeled

pool.

EBAnO-Augment late mix The first $\tau/2$ query iterations are performed via highest entropy sampling, while the remaining $\tau/2$ iterations are carried out via EBAnO-Augment. The parameter configuration is the same one described in early mix.

Early mix and late mix allow us to limit the computational cost of EBAnO-based query strategies and understand the effect of EBAnO at different stages of model performance. In particular, we expect that clusters generated at earlier stages of model training to be less precise than those generated at later stages, when the model has learned more information and performs better.

Dataset	Query size q	Limit l_{U}	Limit l_{A}	Subset size κ
ImageNet-25	1500	1000	500	All
ImageNet-100	3000	2000	1000	20000
ImageNet-250	7500	5000	3000	25000

Table 4.3: Query strategy hyperparameter configuration for EBAnO-Uncertainty and EBAnO-Augment. The configuration differs for each ImageNet subset.

Table 4.3 summarizes the query strategy configuration we adopted for each ImageNet subset. Note that the subset selection of κ samples is only applied to EBAnO-Uncertainty and EBAnO-Augment strategies, while baseline query strategies may exploit the whole unlabeled pool. This is done to avoid unfairly penalizing baseline strategies.

Notably, every initial training dataset is balanced, i.e., each class has the same amount of samples. However, the training subset may become more unbalanced over time as DAL queries samples without any knowledge of the class contents of the chosen batch.

4.4 Results

This section describes the setup of our experiments and presents the results. We report model accuracy on the test set at each DAL iteration, repeating every experiment three times for better statistical significance.

Firstly, we consider the four baselines: Random, Margin, LC, and Entropy. Figures 4.3, 4.4, and 4.5 show the results obtained on datasets ImageNet-25, ImageNet-100, and ImageNet-250 respectively. The base model performance is already good for ImageNet-25, while ImageNet-100 and ImageNet-250 models start from a lower accuracy level. The initial accuracy discrepancy between base models is due to the task difficulty being higher for larger datasets. In other words, the model has to discriminate between a higher number of classes while the number of images per class in the initial training set remains constant for all three types of experiments.

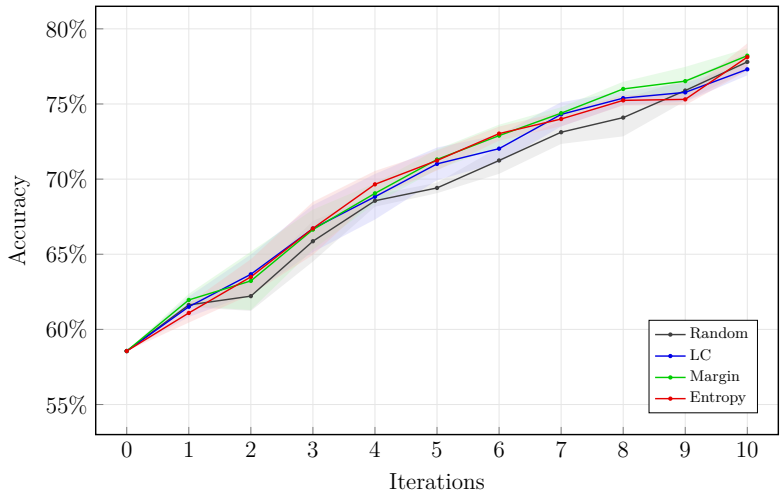

Figure 4.3: Accuracy of ImageNet-25 baselines: Random (random sampling), LC (least confident sampling), Margin (largest margin sampling) and Entropy (highest entropy sampling). At each iteration, 1500 samples from the unlabeled pool are annotated and added to the training set. The best performing baselines are Entropy and Margin.

The baseline plots of ImageNet-25 and ImageNet-100 show a steady upwards trend of model accuracy as soon as the active learning loop begins. This accuracy increase means that the models learn additional information from the queried batches. On the other hand, experiments carried out on ImageNet-250 show a mostly constant trend until the third iteration. However, model accuracy increases significantly from the fourth iteration onwards and keeps increasing until all iterations are performed. The initial constant trend may be due to the fact that the model requires a larger batch of data than the one we provide to converge to a higher accuracy point.

The experiments carried out on ImageNet-25 reach a high accuracy of 76%-78% after only ten iterations, i.e., querying a total of two-thirds of the unlabeled pool. For this type of experiment, we do not use the last third of never-queried data to highlight the performance gap between different methods. In fact, we expect all methods to converge to the same accuracy when all available data has been queried. On the other hand, we query a smaller amount of data for experiments on ImageNet-100 and ImageNet-250 due to the high computational cost of training on these datasets. A lower amount of training data translates to a lower final model accuracy of 55%-57%, which is still higher than the base model performance.

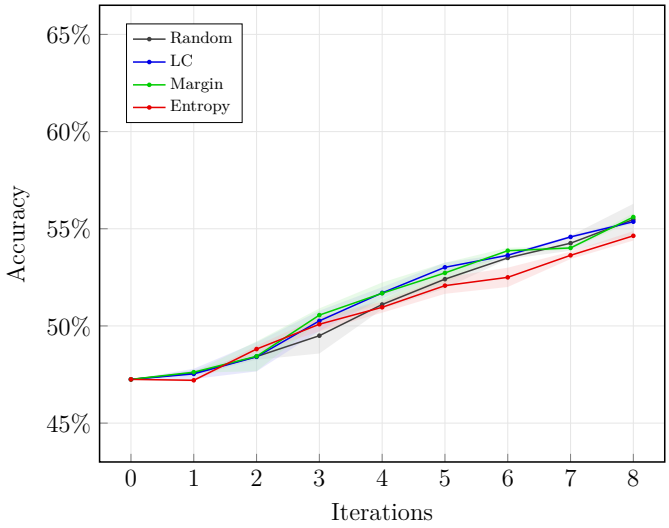

Figure 4.4: Accuracy of baselines on ImageNet-100. At each iteration, 3000 samples from the unlabeled pool are annotated and added to the training set. The best performing baselines are LC (least confident sampling) and Margin (largest margin sampling).

The goal of active learning is to reduce the amount of data needed for a model to reach a satisfactory level of performance. In other words, we want the accuracy to increase as fast as possible by querying the smallest possible subset of the unlabeled pool. Analyzing the performance of the baseline methods, we can see that, on average, the model performance increase is slightly higher for uncertainty-based techniques when compared to random sampling. This gap is especially evident in the experiments carried out on ImageNet-25. Unexpectedly, the Entropy baseline performs on par with or

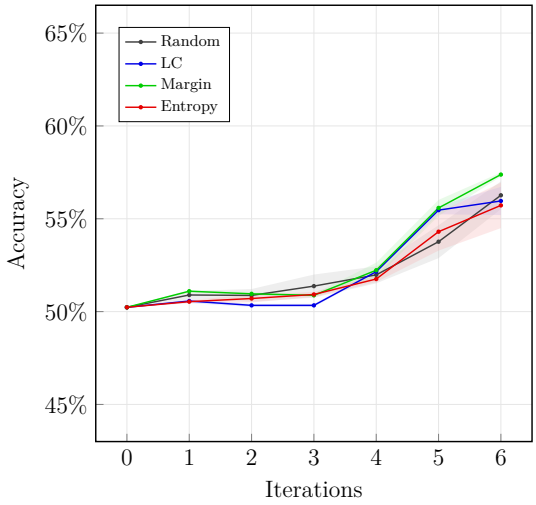

Figure 4.5: Accuracy of baselines on ImageNet-250. At each iteration, 7500 samples from the unlabeled pool are annotated and added to the training set. The best performing baselines are LC (least confident sampling) and Margin (largest margin sampling), albeit all strategies are similar to one another.

worse than Random on ImageNet-100 and ImageNet-250, while LC and Margin are, on average, above the other two baselines. We speculate that this is due to the lower initial performance of the model, which provides less relevant information on the rest of the class distribution. Nonetheless, these results show that the random baseline is a strong method to beat, as the performance gap between it and other methods is relatively small.

We now analyze the performance of our custom query strategies on the datasets under analysis. Figures 4.6, 4.7, and 4.8 show EBAnO-Uncertainty and EBAnO-Augment strategies against the two best baselines for each dataset. Margin and Entropy are the two best performing baselines on ImageNet-25, while LC and Margin are the best baselines on ImageNet-100 and ImageNet-250.

We perform each experiment based on EBAnO using two types of mixed query strategies. *Early mix* shows the performance of EBAnO during earlier stages of model training, while *late mix* shows its performance during the last steps of the DAL framework. Notably, we present the results of two sets of experiments on ImageNet-25, using a non-limited and a limited approach, in Figure 4.6. The limited approach imposes a limit on the maximum number

of samples queried by EBAnO l_{U} and a limit on the number of samples to add to the augmented set l_{A}. The limited set of experiments is denoted with *(L)*.

We find that EBAnO-Uncertainty early mix on ImageNet-25 performs slightly better than baselines, especially during the first stages of training, and retains the slight performance advantage over successive iterations. This is a promising result. On the other hand, we find that the performance of EBAnO-Uncertainty late mix on ImageNet-25 is similar to Entropy throughout the incremental training procedure. We hypothesize that this is because the earlier stages of training are those where better performing active learning strategies provide the most advantage to model accuracy. EBAnO-Uncertainty late mix cannot exploit the advantages provided by EBAnO early on, as the first five steps are performed using Entropy. The performance similarity at later stages can be explained by the fact that once most of the unlabeled pool is annotated, the differences between models will decrease.

Interestingly, we find some discrepancies between EBAnO-Uncertainty (L) late mix and Entropy performance at the earlier stages of DAL. Both methods use the same query strategy in the first five steps of active learning, namely Entropy. This difference underlines the high variability of our experiments, which is mainly due to the nature of the training procedure employed by deep neural networks. This issue does not arise in EBAnO-Augment (L) late mix.

Regarding EBAnO-Augment on ImageNet-25, we find that, unfortunately, there is no real advantage to using this custom strategy, regardless of the approach we use. This method appears to be on par with classic uncertainty-based query strategies. The performance gap to our other strategy, EBAnO-Uncertainty, may be due to the current limitations of EBAnO-Augment we discussed in Section 3.3.2. Nonetheless, the approach produces better results than Random and shows some promise.

We find that the accuracy values obtained at the end of training on ImageNet-100 and ImageNet-250 by baseline strategies and EBAnO-based query methods are limited by the smaller number of DAL iterations due to high computational cost, as we discussed before. This problem is especially evident on ImageNet-250. Nonetheless, we can see some interesting trends in the current results.

On ImageNet-100, EBAnO-Augment early mix shows more promising results, as it performs slightly better than the baselines during the first

four iterations. On the other hand, the late mix strategy follows the low-performing trend of Entropy during the first four iterations. Therefore, EBAnO-Augment starts at a disadvantageous point with respect to the best-performing baselines. However, our custom strategy with data augmentation promptly increases the accuracy of the network from steps four to five. This increase suggests that EBAnO-Augment is correcting the bad performance of the network inherited from the Entropy strategy. On ImageNet-250, our custom strategies are mostly on-par with baseline strategies. The similarity among different methods may be due to the fact that there is no real difference in performance to be gained from different sets of data from the unlabeled pool at this stage of training. In other words, any additional data batch will increase the model performance in a similar manner.

Overall, the margins between baseline and EBAnO-based query strategies are small in most of our experiments. The low difference between DAL strategies may be due to the nature of the ImageNet dataset, a challenging benchmark that contains diverse images within the same category. We hypothesize that the high diversity of data points and low redundancy in ImageNet translates to queried data batches that always contain a high amount of information, regardless of the type of strategy employed to select data from the unlabeled pool. Therefore, selecting samples with low uncertainty, as measured by the softmax output of the model, will still yield a highly informative set of data which increases model performance similarly to a batch of data queried via a custom query strategy. However, this effect is limited, as uncertainty-based strategies still beat random sampling, albeit by a small margin. Future works may explore different datasets to validate further the performance of our custom query strategies.

Notwithstanding its limitations, our results show that the performance of our custom query strategies is promising. Moreover, further hyperparameter tuning of our custom query strategies may increase the performance gap to classic uncertainty-based baselines on the datasets under analysis.

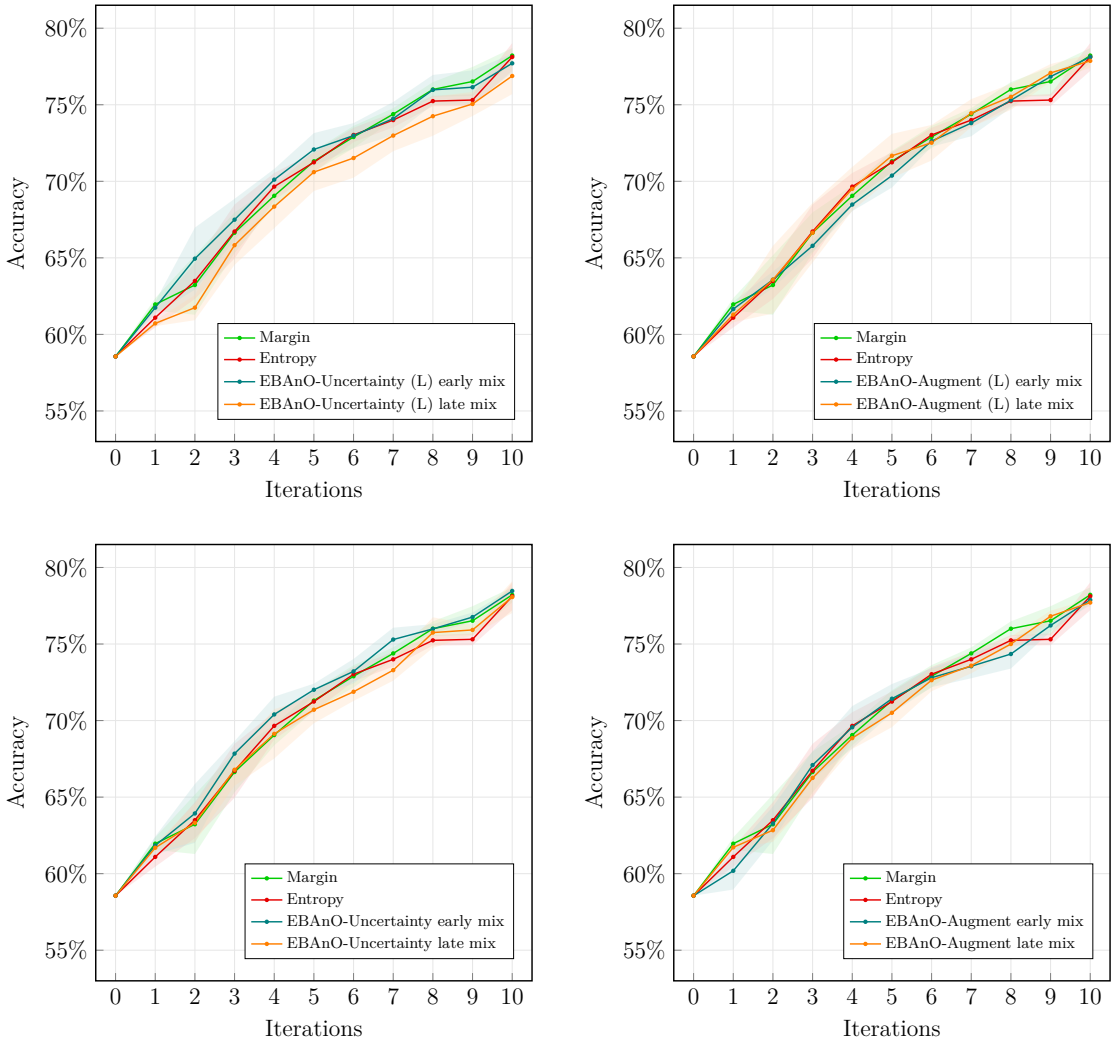

Figure 4.6: EBAnO-Uncertainty (on the left) and EBAnO-Augment (on the right) against Margin and Entropy on ImageNet-25. At each iteration, 1500 samples are added to the training set. If EBAnO strategies do not fill the whole batch, additional samples are queried via Entropy. The top row shows limited (L) approaches. A maximum of $l_{\mathrm{U}} = 1000$ data points are queried via EBAnO-Uncertainty, while the remaining samples are selected via Entropy. For EBAnO-Augment (L), the augmentation set is composed of a maximum of $l_{\mathrm{A}} = 500$ samples. Non-limited approaches, shown in the bottom row, do not impose any hard limit on EBAnO-based queries. EBAnO-Uncertainty early mix remains above baselines across all iterations, on both the limited and non-limited approaches, albeit the accuracy difference is slight.

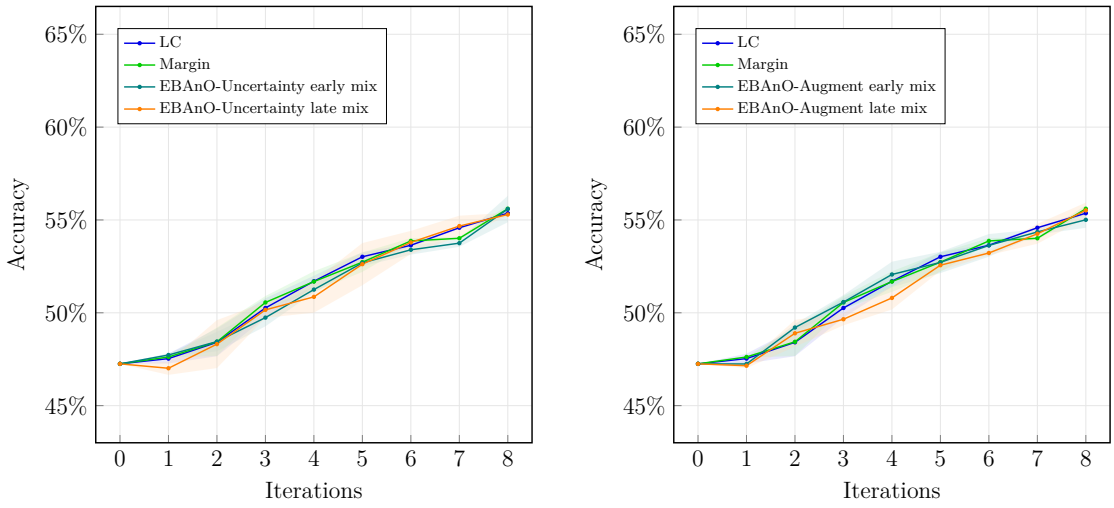

Figure 4.7: EBAnO-Uncertainty (on the left) and EBAnO-Augment (on the right) against LC and Margin on ImageNet-100. At each iteration with EBAnO, 3000 samples are added to the training set. Of these, a maximum of $l_{\mathrm{U}} = 2000$ data points are queried via EBAnO, while the remaining samples are selected via Entropy. The augmentation set is composed of a maximum of $l_{\mathrm{A}} = 1000$ highest-ranked samples queried by EBAnO. EBAnO-Augment early mix performs slightly better than baselines in the early stages. On the other hand, EBAnO-Augment late mix performs worse in the first stages, following the trend of Entropy, but promptly picks up performance in the last steps.

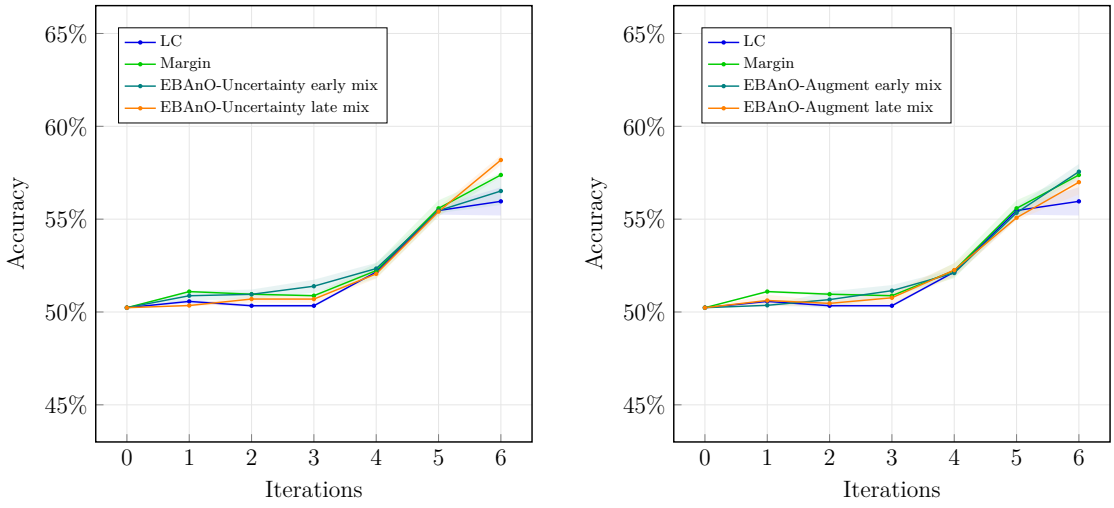

Figure 4.8: EBAnO-Uncertainty (on the left) and EBAnO-Augment (on the right) against LC and Margin on ImageNet-250. At each iteration with EBAnO, 7500 samples are added to the training set. Of these, a maximum of $l_{\mathrm{U}} = 5000$ data points are queried via EBAnO, while the remaining samples are selected via Entropy. The augmentation set is composed of a maximum of $l_{\mathrm{A}} = 3000$ highest-ranked samples queried by EBAnO. Both strategies have comparable performance to baselines.

Chapter 5

Conclusions

In our work, we explore how the DAL framework helps to reduce manual labeling costs by selecting a limited amount of samples from a large pool of unlabeled data in an iterative fashion using a ranking function, namely a query strategy. We focus on the multi-class image classification task with CNNs, a specialized class of black-box deep learning models.

We envision a complete explainability framework applicable to different parts of the DAL procedure. The main goal of this work is to design an active learning query strategy that exploits information provided by EBAnO, an explainability method that provides insights into the inner workings of black-box CNNs, to improve model performance more quickly than classic uncertainty-based DAL baselines. To the best of our knowledge, no existing study injects knowledge extracted from explanations into a query strategy in an unsupervised manner.

We introduce BatchEBAnO to efficiently integrate EBAnO into the DAL loop. This engine is an optimized version of EBAnO that delivers a tenfold decrease in execution time on ResNet-50 while retaining good cluster quality. Based on an empirical study on the behavior of EBAnO, we develop two novel query strategies, EBAnO-Uncertainty and EBAnO-Augment, that select samples whose most influential interpretable feature is not precisely focused on the predicted class, as measured by indices directly extracted from samples in the unlabeled pool. Additionally, EBAnO-Augment partially obfuscates a subset of queried images and interleaves them into the current training set. Our custom query strategies provide a ranking on the uncertainty of the samples, based on the contrast between the influence of the most influential interpretable feature on the prediction

and its precision over classes.

We perform multiple experiments on three ImageNet subsets of varying sizes to compare our custom query strategies against classic uncertainty-based methods and random sampling. Our custom strategies show promising results on the evaluated benchmarks and seem to be effective for improving the performance of uncertainty-based baselines in some settings. However, further hyperparameter tuning and testing on different datasets and models are required to assess the validity and effectiveness of EBAnO-based methods.

5.1 Limitations and future work

The baseline query strategies we evaluated and our custom query methods exclusively focus on the uncertainty of samples. They do not consider the diversity of queried batches, as done by state-of-the-art hybrid query strategies. Moreover, as discussed in Section 2.1, the softmax output of the network is not always a good proxy for model uncertainty on classes. Therefore, future work may explore hybrid query strategies and exploit different uncertainty measurements to increase model performance further.

Our custom EBAnO query strategies focus only on the most influential interpretable feature and ignore the rest of the distribution of nPIR and nPIRP indices. There may be additional helpful information in the unused indices that we can exploit in our custom query strategies in a future version.

While BatchEBAnO drastically reduces the time required to generate explanations, it does not scale well to huge data pools. This limitation is intrinsic to the nature of the explainability procedure of EBAnO. In this work, to avoid processing an excessively large amount of data at each DAL iteration, we sample a limited number of images from the unlabeled pool in a random fashion. Future work may explore different strategies to pre-select a manageable subset of the unlabeled pool to process via BatchEBAnO.

As discussed before, the ImageNet dataset is a challenging benchmark for image classification, and any batch of data contains relevant information that significantly increases model performance, regardless of the query strategy. Future work may focus on performing experiments on other datasets with different characteristics to assess the performance of our custom query strategies in a different setting.

5.2 Acknowledgments

Computational resources were kindly provided by SmartData@PoliTO[1] and by HPC@POLITO[2] at Politecnico di Torino.

[1] https://smartdata.polito.it/

[2] http://www.hpc.polito.it

Appendix A

Datasets

Dataset	Labels
ImageNet-25	0, 217, 482, 491, 497, 566, 569, 571, 574, 701
ImageNet-100	3, 5, 7, 13, 18, 24, 28, 33, 45, 63, 66, 69, 76, 77, 109, 120, 144, 149, 220, 233, 235, 242, 244, 253, 265, 270, 289, 302, 310, 317, 332, 342, 344, 353, 356, 365, 371, 385, 415, 427, 432, 433, 450, 469, 470, 473, 477, 484, 495, 527, 538, 540, 546, 554, 562, 568, 570, 598, 601, 608, 617, 642, 647, 648, 659, 664, 685, 690, 694, 698, 700, 715, 725, 742, 750, 752, 763, 779, 784, 796, 802, 816, 820, 833, 840, 849, 866, 878, 883, 908, 913, 918, 936, 949, 954, 968, 970, 980, 995, 998

ImageNet-250	2, 6, 8, 12, 15, 16, 17, 22, 26, 30, 39, 41, 44, 48, 52, 56, 57, 64, 65, 67, 70, 75, 87, 88, 89, 100, 101, 106, 115, 117, 119, 125, 130, 132, 133, 136, 155, 156, 159, 169, 173, 176, 182, 184, 186, 187, 192, 198, 201, 205, 208, 209, 213, 227, 240, 245, 247, 248, 250, 255, 261, 266, 272, 273, 275, 280, 281, 285, 287, 290, 291, 293, 295, 304, 306, 313, 316, 320, 321, 322, 325, 326, 330, 334, 338, 347, 348, 349, 350, 355, 357, 362, 366, 368, 372, 375, 377, 378, 382, 384, 395, 396, 400, 405, 407, 411, 413, 414, 420, 422, 428, 442, 443, 445, 446, 447, 449, 452, 455, 457, 459, 460, 462, 467, 471, 479, 485, 487, 492, 493, 496, 498, 506, 510, 516, 518, 519, 523, 525, 529, 534, 541, 542, 544, 547, 552, 553, 561, 576, 586, 587, 591, 593, 594, 602, 606, 611, 615, 619, 620, 621, 625, 626, 629, 632, 640, 643, 645, 649, 650, 656, 660, 674, 683, 691, 692, 696, 703, 705, 713, 718, 732, 736, 741, 755, 756, 757, 760, 761, 762, 764, 768, 775, 777, 780, 792, 793, 794, 797, 799, 803, 814, 823, 828, 829, 830, 832, 835, 836, 843, 844, 845, 847, 856, 862, 870, 873, 874, 875, 880, 886, 887, 893, 898, 900, 905, 909, 911, 916, 922, 923, 928, 932, 942, 948, 951, 959, 960, 961, 964, 967, 971, 974, 976, 981, 983, 985, 988, 992, 996

Bibliography

[1] Ian Goodfellow and Yoshua Bengio. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[2] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. «V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation». In: *2016 Fourth International Conference on 3D Vision (3DV)*. Stanford, CA, USA: IEEE, Oct. 2016, pp. 565–571. ISBN: 978-1-5090-5407-7. DOI: 10.1109/3DV.2016.79. URL: http://ieeexplore.ieee.org/document/7785132/.

[3] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. «BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation». In: *arXiv:1808.00897 [cs]* (Aug. 2018). URL: http://arxiv.org/abs/1808.00897.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». In: *arXiv:1810.04805 [cs]* (May 2019). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[5] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, R E Howard, Wayne E Hubbard, and Lawrence D Jackel. «Handwritten Digit Recognition with a Back-Propagation Network». en. In: (1989), p. 9.

[6] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. «Bag of Tricks for Image Classification with Convolutional Neural Networks». en. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 558–567. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00065. URL: https://ieeexplore.ieee.org/document/8954382/.

[7] Olga Russakovsky et al. «ImageNet Large Scale Visual Recognition Challenge». en. In: *International Journal of Computer Vision* 115.3 (Dec. 2015), pp. 211–252. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-015-0816-y.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. «ImageNet classification with deep convolutional neural networks». en. In: *Communications of the ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386.

[9] Karen Simonyan and Andrew Zisserman. «Very Deep Convolutional Networks for Large-Scale Image Recognition». en. In: *arXiv:1409.1556 [cs]* (Apr. 2015). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep Residual Learning for Image Recognition». en. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Dec. 2015).

[11] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. «A Survey of Deep Active Learning». en. In: (Aug. 2020). URL: http://arxiv.org/abs/2009.00236.

[12] Asim Smailagic et al. «O-MedAL: Online active deep learning for medical image analysis». en. In: *WIREs Data Mining and Knowledge Discovery* 10.4 (July 2020). ISSN: 1942-4787, 1942-4795. DOI: 10.1002/widm.1353. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1353.

[13] Bogdan Kwolek, Michał Koziarski, Andrzej Bukała, Zbigniew Antosz, Bogusław Olborski, Paweł Wąsowicz, Jakub Swadźba, and Bogusław Cyganek. «Breast Cancer Classification on Histopathological Images Affected by Data Imbalance Using Active Learning and Deep Convolutional Neural Network». en. In: *Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions*. Ed. by Igor V. Tetko, Věra Kůrková, Pavel Karpov, and Fabian Theis. Vol. 11731. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 299–312. ISBN: 978-3-030-30492-8 978-3-030-30493-5. DOI: 10.1007/978-3-030-30493-5_31. URL: http://link.springer.com/10.1007/978-3-030-30493-5_31.

[14] Peng Liu, Hui Zhang, and Kie B. Eom. «Active Deep Learning for Classification of Hyperspectral Images». en. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 10.2 (Feb. 2017), pp. 712–724. ISSN: 1939-1404, 2151-1535. DOI: 10.1109/JSTARS.2016.2598859. URL: http://ieeexplore.ieee.org/document/7568999/.

[15] Francesco Ventura and Tania Cerquitelli. «What's in the box? Explaining the black-box model through an evaluation of its interpretable features». In: (July 2019).

[16] Francesco Ventura, Tania Cerquitelli, and Francesco Giacalone. «Black-Box Model Explained Through an Assessment of Its Interpretable Features». In: *New Trends in Databases and Information Systems*. Ed. by András Benczúr, Bernhard Thalheim, Tomáš Horváth, Silvia Chiusano, Tania Cerquitelli, Csaba Sidló, and Peter Z. Revesz. Vol. 909. Cham: Springer International Publishing, 2018, pp. 138–149. ISBN: 978-3-030-00062-2 978-3-030-00063-9. URL: http://link.springer.com/10.1007/978-3-030-00063-9_15.

[17] Bhavya Ghai, Q. Vera Liao, Yunfeng Zhang, Rachel Bellamy, and Klaus Mueller. «Explainable Active Learning (XAL): An Empirical Study of How Local Explanations Impact Annotator Experience». en. In: *arXiv:2001.09219 [cs]* (Sept. 2020). arXiv: 2001.09219. URL: http://arxiv.org/abs/2001.09219.

[18] Jonathan Folmsbee, Xulei Liu, Margaret Brandwein-Weber, and Scott Doyle. «Active deep learning: Improved training efficiency of convolutional neural networks for tissue classification in oral cavity cancer». en. In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. Washington, DC: IEEE, Apr. 2018, pp. 770–773. ISBN: 978-1-5386-3636-7. DOI: 10.1109/ISBI.2018.8363686. URL: https://ieeexplore.ieee.org/document/8363686/.

[19] Samuel Budd, Emma C. Robinson, and Bernhard Kainz. «A Survey on Active Learning and Human-in-the-Loop Deep Learning for Medical Image Analysis». en. In: *arXiv:1910.02923 [cs, eess]* (Oct. 2019). URL: http://arxiv.org/abs/1910.02923.

[20] Burr Settles. «Active Learning». en. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.1 (June 2012), pp. 1–114. ISSN: 1939-4608, 1939-4616. DOI: 10.2200/S00429ED1V01Y201207AIM018.

[21] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. «Active Hidden Markov Models for Information Extraction». In: *Advances in Intelligent Data Analysis*. Ed. by G. Goos, J. Hartmanis, J. van Leeuwen, Frank Hoffmann, David J. Hand, Niall Adams, Douglas Fisher, and Gabriela Guimaraes. Vol. 2189. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 309–318. ISBN: 978-3-540-42581-6 978-3-540-44816-7. URL: http://link.springer.com/10.1007/3-540-44816-0_31.

[22] C E Shannon. «A Mathematical Theory of Communication». en. In: (1948), p. 55.

[23] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. «Cost-Effective Active Learning for Deep Image Classification». en. In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.12 (Dec. 2017), pp. 2591–2600. ISSN: 1051-8215, 1558-2205. DOI: 10.1109/TCSVT.2016.2589879. URL: http://ieeexplore.ieee.org/document/7508942/.

[24] Erik Bochinski, Ghassen Bacha, Volker Eiselein, Tim J. W. Walles, Jens C. Nejstgaard, and Thomas Sikora. «Deep Active Learning for In Situ Plankton Classification». en. In: *Pattern Recognition and Information Forensics*. Ed. by Zhaoxiang Zhang, David Suter, Yingli Tian, Alexandra Branzan Albu, Nicolas Sidère, and Hugo Jair Escalante. Vol. 11188. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 5–15. ISBN: 978-3-030-05791-6 978-3-030-05792-3. DOI: 10.1007/978-3-030-05792-3_1. URL: http://link.springer.com/10.1007/978-3-030-05792-3_1.

[25] Baolin Du, Qi Qi, Han Zheng, Yue Huang, and Xinghao Ding. «Breast Cancer Histopathological Image Classification via Deep Active Learning and Confidence Boosting». en. In: *Artificial Neural Networks and Machine Learning – ICANN 2018*. Ed. by Věra Kůrková, Yannis Manolopoulos, Barbara Hammer, Lazaros Iliadis, and Ilias Maglogiannis. Vol. 11140. Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 109–116. ISBN: 978-3-030-01420-9 978-3-030-01421-6. DOI: 10.1007/978-3-030-01421-6_11. URL: http://link.springer.com/10.1007/978-3-030-01421-6_11.

[26] Ya Li, Keze Wang, Lin Nie, and Qing Wang. «Face Recognition via Heuristic Deep Active Learning». en. In: *Biometric Recognition*. Ed. by Jie Zhou et al. Vol. 10568. Series Title: Lecture Notes in Computer

Science. Cham: Springer International Publishing, 2017, pp. 97–107. ISBN: 978-3-319-69922-6 978-3-319-69923-3. DOI: 10.1007/978-3-319-69923-3_11. URL: http://link.springer.com/10.1007/978-3-319-69923-3_11.

[27] Hiranmayi Ranganathan, Hemanth Venkateswara, Shayok Chakraborty, and Sethuraman Panchanathan. «Deep active learning for image classification». en. In: *2017 IEEE International Conference on Image Processing (ICIP)*. Beijing: IEEE, Sept. 2017, pp. 3934–3938. ISBN: 978-1-5090-2175-8. DOI: 10.1109/ICIP.2017.8297020. URL: http://ieeexplore.ieee.org/document/8297020/.

[28] Yarin Gal and Zoubin Ghahramani. «Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning». en. In: (2016), p. 10.

[29] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. «Deep Bayesian Active Learning with Image Data». en. In: *arXiv:1703.02910 [cs, stat]* (Mar. 2017). arXiv: 1703.02910. URL: http://arxiv.org/abs/1703.02910.

[30] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. «Bayesian Active Learning for Classification and Preference Learning». en. In: *arXiv:1112.5745 [cs, stat]* (Dec. 2011). arXiv: 1112.5745. URL: http://arxiv.org/abs/1112.5745.

[31] Remus Pop and Patric Fulop. «Deep Ensemble Bayesian Active Learning: Addressing the Mode Collapse issue in Monte Carlo dropout via Ensembles». In: (Nov. 2018). arXiv: 1811.03897. URL: https://arxiv.org/abs/1811.03897.

[32] William H. Beluch, Tim Genewein, Andreas Nurnberger, and Jan M. Kohler. «The Power of Ensembles for Active Learning in Image Classification». en. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, June 2018, pp. 9368–9377. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00976. URL: https://ieeexplore.ieee.org/document/8579074/.

[33] Sanjoy Dasgupta. «Two faces of active learning». en. In: *Theoretical Computer Science* 412.19 (Apr. 2011), pp. 1767–1781. ISSN: 03043975. DOI: 10.1016/j.tcs.2010.12.054. URL: https://linkinghub.elsevier.com/retrieve/pii/S0304397510007620.

[34] Fedor Zhdanov. «Diverse mini-batch Active Learning». en. In: *arXiv:1901.05954 [cs, stat]* (Jan. 2019). URL: http://arxiv.org/abs/1901.05954.

[35] Asim Smailagic, Hae Young Noh, Pedro Costa, Devesh Walawalkar, Kartik Khandelwal, Mostafa Mirshekari, Jonathon Fagert, Adrián Galdrán, and Susu Xu. «MedAL: Deep Active Learning Sampling Method for Medical Image Analysis». In: (Sept. 2018). arXiv: 1809.09287. URL: http://arxiv.org/abs/1809.09287.

[36] Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. «Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds». In: (Feb. 2020). URL: http://arxiv.org/abs/1906.03671.

[37] Changjian Shui, Fan Zhou, Christian Gagné, and Boyu Wang. «Deep Active Learning: Unified and Principled Method for Query and Training». en. In: (Feb. 2020). URL: http://arxiv.org/abs/1911.09162.

[38] Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. «BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning». In: *arXiv:1906.08158 [cs, stat]* (Oct. 2019). arXiv: 1906.08158. URL: http://arxiv.org/abs/1906.08158.

[39] Ozan Sener and Silvio Savarese. «Active Learning for Convolutional Neural Networks: A Core-Set Approach». en. In: *arXiv:1708.00489 [cs, stat]* (June 2018). arXiv: 1708.00489. URL: http://arxiv.org/abs/1708.00489.

[40] Xuefeng Du, Dexing Zhong, and Huikai Shao. «Building an Active Palmprint Recognition System». In: *2019 IEEE International Conference on Image Processing (ICIP)*. Taipei, Taiwan: IEEE, Sept. 2019, pp. 1685–1689. ISBN: 978-1-5386-6249-6. DOI: 10.1109/ICIP.2019.8803135. URL: https://ieeexplore.ieee.org/document/8803135/.

[41] Daniel Gissin and Shai Shalev-Shwartz. «Discriminative Active Learning». In: *arXiv:1907.06347 [cs, stat]* (July 2019). arXiv: 1907.06347. URL: http://arxiv.org/abs/1907.06347.

[42] Hima Lakkaraju, Julius Adebayo, and Sameer Singh. *Explaining Machine Learning Predictions*. 2020. URL: https://explainml-tutorial.github.io/.

[43] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. «A Survey of Methods for Explaining Black Box Models». en. In: *ACM Computing Surveys* 51.5 (Jan. 2019), pp. 1–42. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3236009. URL: https://dl.acm.org/doi/10.1145/3236009.

[44] Sheikh Rabiul Islam, William Eberle, Sheikh Khaled Ghafoor, and Mohiuddin Ahmed. «Explainable Artificial Intelligence Approaches: A Survey». In: *arXiv:2101.09429 [cs]* (Jan. 2021). arXiv: 2101.09429. URL: http://arxiv.org/abs/2101.09429.

[45] Finale Doshi-Velez and Been Kim. «Towards A Rigorous Science of Interpretable Machine Learning». en. In: *arXiv:1702.08608 [cs, stat]* (Mar. 2017). arXiv: 1702.08608.

[46] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. «"Why Should I Trust You?": Explaining the Predictions of Any Classifier». en. In: (Aug. 2016). arXiv: 1602.04938. URL: http://arxiv.org/abs/1602.04938.

[47] Scott M Lundberg and Su-In Lee. «A Unified Approach to Interpreting Model Predictions». In: *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, p. 10.

[48] L. S. Shapley. «Stochastic Games». en. In: *Proceedings of the National Academy of Sciences* 39.10 (Oct. 1953), pp. 1095–1100. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.39.10.1095.

[49] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. «Anchors: High Precision Model-Agnostic Explanations». en. In: *Proceedings of the AAAI conference on artificial intelligence*. 2018, p. 9.

[50] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. «Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization». en. In: *International Journal of Computer Vision* 128.2 (Feb. 2020), pp. 336–359. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://arxiv.org/abs/1610.02391.

[51] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. «SmoothGrad: removing noise by adding noise». en. In: (June 2017). URL: http://arxiv.org/abs/1706.03825.

[52] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. «Learning Important Features Through Propagating Activation Differences». In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 3145–3153. URL: https://proceedings.mlr.press/v70/shrikumar17a.html.

[53] R. Dennis Cook and Sanford Weisberg. «Characterizations of an Empirical Influence Function for Detecting Influential Cases in Regression». en. In: *Technometrics* 22.4 (Nov. 1980), pp. 495–508. ISSN: 0040-1706, 1537-2723. DOI: 10.1080/00401706.1980.10486199. URL: http://www.tandfonline.com/doi/abs/10.1080/00401706.1980.10486199.

[54] Pang Wei Koh and Percy Liang. «Understanding Black-box Predictions via Influence Functions». In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1885–1894. URL: https://proceedings.mlr.press/v70/koh17a.html.

[55] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. «Feature Visualization». In: *Distill* (2017). https://distill.pub/2017/feature-visualization. DOI: 10.23915/distill.00007.

[56] Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. «Estimating Training Data Influence by Tracing Gradient Descent». In: (Nov. 2020). URL: http://arxiv.org/abs/2002.08484.

[57] Chih-Kuan Yeh, Joon Sik Kim, Ian E. H. Yen, and Pradeep Ravikumar. «Representer Point Selection for Explaining Deep Neural Networks». In: *arXiv:1811.09720 [cs, stat]* (Nov. 2018). arXiv: 1811.09720. URL: http://arxiv.org/abs/1811.09720.

[58] Berk Ustun, Alexander Spangher, and Yang Liu. «Actionable Recourse in Linear Classification». In: *Proceedings of the Conference on Fairness, Accountability, and Transparency* (Jan. 2019), pp. 10–19. DOI: 10.1145/3287560.3287566. URL: http://arxiv.org/abs/1809.06514.

[59] Divyat Mahajan, Chenhao Tan, and Amit Sharma. «Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers». In: *arXiv:1912.03277 [cs, stat]* (June 2020). arXiv: 1912.03277. URL: http://arxiv.org/abs/1912.03277.

[60] Bharath Hariharan, Pablo Arbelaez, Ross Girshick, and Jitendra Malik. «Hypercolumns for Object Segmentation and Fine-Grained Localization». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[61] S. Lloyd. «Least squares quantization in PCM». en. In: *IEEE Transactions on Information Theory* 28.2 (Mar. 1982), pp. 129–137. ISSN: 0018-9448. DOI: 10.1109/TIT.1982.1056489. URL: http://ieeexplore.ieee.org/document/1056489/.

[62] Stefano Teso and Kristian Kersting. «"Why Should I Trust Interactive Learners?" Explaining Interactive Queries of Classifiers to Users». en. In: (May 2018). URL: http://arxiv.org/abs/1805.08578.

[63] Ishani Mondal and Debasis Ganguly. «ALEX: Active Learning based Enhancement of a Model's Explainability». en. In: *arXiv:2009.00859 [cs]* (Sept. 2020). arXiv: 2009.00859. URL: http://arxiv.org/abs/2009.00859.

[64] Charles R. Harris et al. «Array programming with NumPy». In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[65] F. Pedregosa et al. «Scikit-learn: Machine Learning in Python». In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[66] Jeff Johnson, Matthijs Douze, and Hervé Jégou. «Billion-scale similarity search with GPUs». In: *arXiv preprint arXiv:1702.08734* (2017).

[67] François Chollet et al. *Keras*. https://keras.io. 2015.