Adversarial Sampling for Active Learning

Christoph Mayer Radu Timofte
{chmayer,timofter}@vision.ee.ethz.ch
Computer Vision Lab, ETH Zürich, Switzerland

Abstract

This paper proposes ASAL, a new GAN based active learning method that generates high entropy samples. Instead of directly annotating the synthetic samples, ASAL searches similar samples from the pool and includes them for training. Hence, the quality of new samples is high and annotations are reliable. To the best of our knowledge, ASAL is the first GAN based AL method applicable to multi-class problems that outperforms random sample selection. Another benefit of ASAL is its small run-time complexity (sub-linear) compared to traditional uncertainty sampling (linear). We present a comprehensive set of experiments on multiple traditional data sets and show that ASAL outperforms similar methods and clearly exceeds the established baseline (random sampling). In the discussion section we analyze in which situations ASAL performs best and why it is sometimes hard to outperform random sample selection.

1. Introduction

The goal of Active Learning (AL) algorithms is to train a model most efficiently, i.e. achieving the best performance with as few labelled samples as possible. Typical AL algorithms operate in an iterative fashion, where in each AL cycle a query strategy selects samples that the oracle should annotate. These samples are expected to improve the model most effectively when added to the training set. This procedure continues until a predefined stopping criteria is met.

In this paper we will mainly focus on pool based active learning, because a pool of unlabelled samples is often available beforehand or can easily be built. Furthermore, annotating all pool samples serves as an ideal evaluation environment for active learning algorithms. It enables to train a fully-supervised model that establishes a performance upper bound on this data set. Similarly, randomly selecting instead of actively choosing samples establishes a lower bound. Then, the goal of an active learning algorithm is to approximate the performance of the fully supervised model with as few labelled samples as possible, while exceeding the performance of random sampling.

Uncertainty sampling is an effective query strategy that identifies samples that are more informative than random ones. The heuristic is, that samples for which the model is most uncertain contain new information and improve the model. Uncertainty sampling is the most commonly used AL strategy for Generative Adversarial Network (GAN) based AL methods [29, 8]. However, these related methods are designed for small and very simple datasets, cover only binary classification tasks and use Support Vector Machines (SVMs) for classification instead of CNNs. Generative Adversarial Active Learning (GAAL) [29] even fails to outperform random sample selection.

Our contributions are as follows:

- Adversarial Sampling for Active Learning (ASAL) is to the best of our knowledge the first *pool* based AL method that uses GAN to tackle multi-class problems.
- ASAL achieves sub-linear run-time complexity even though it searches the full pool in each AL cycle.
- We validate ASAL on full image data sets (MNIST, CIFAR-10, CelebA, SVHN, LSUN) compared to related methods that use much simpler challenges such as: two class subsets of MNIST, CIFAR-10 or SVHN.

2. Related Work

We review related work on AL especially on pool based uncertainty sampling, GAN based AL strategies and methods attempting to improve the run-time complexity.

Pool-based active learning methods select new training samples from a predefined unlabelled data set [7, 17, 26, 19, 27]. A common query strategy to identify new samples is uncertainty sampling [10, 25]. A well known uncertainty sampling strategy that is used to train SVMs, is minimum distance sampling [22, 3]. Minimum distance sampling requires a linear classifier in some feature space and assumes that the classifier is uncertain about samples in the vicinity of the separating hyper-plane. This strategy is mainly used for two class but can be extended to multi-class problems [9]. Joshi *et al.* [10] use information entropy to measure the uncertainty of the classifier for a particular sample.

Computing uncertainty with information entropy is suitable for two or multiple classes.

Jain *et al*. [9] propose two hashing based method to accelerate minimum distance sampling by selecting new samples in sub-linear time. These methods are designed to select the closest point (approximately) to a hyper-plane in a k-dimensional feature space, where the positions of the data points are fixed but the hyper-plane is allowed to move. Thus, these methods are limited to SVMs with fixed feature maps, because, if the feature map changes, the position of the samples become obsolete and need to be recomputed. Hence, the run time complexity is sub-linear for constant feature maps and linear otherwise. Unfortunately, CNN based methods update their feature maps during training. Thus, their methods are as efficient as exhaustive uncertainty sampling if CNNs are involved.

Zhu and Bento [29] propose GAAL, that uses a GAN to generate uncertain synthetic samples in each AL cycle. Generating instead of selecting uncertain samples leads to a constant run-time complexity because producing a new sample is independent of the pool size but requires training a GAN beforehand. Zhu and Bento [29] use the traditional minimal distance optimization problem but replace the variable x (denoting a pool sample) with the trained generator. Then, they use gradient descent to minimize the objective. The latent variable minimizing the objective results in a synthetic image close to the separating hyper-plane. They annotate the synthetic sample and use it for training. Zhu and Bento [29] demonstrate GAAL on subsets of MNIST and CIFAR-10 (two classes) using linear SVMs and DCGANs [18, 4]. However, GAAL performs worse than random sampling on both data sets, because it suffers from sampling bias and annotating is arbitrarily hard caused by sometimes poor quality of the synthetic uncertain samples. Note, that GAAL requires visually distinct classes (*horse* & *automobile*) to allow reliable annotations by humans.

Active Decision Boundary Annotation (ADBA) [8] is another GAN based AL strategy. The main contributions of ADBA are, training the classifier in the latent space and a new annotation scheme. Hence, it requires computing the latent state representation of each data sample using the pretrained GAN. Then, ADBA searches the most uncertain sample in the latent space and generates a line that is perpendicular to the current decision boundary and crosses the most uncertain sample. Then, the GAN generates images along this line such that the annotator can specify for which image in the sequence along this line the class label changes. ADBA shows that it outperforms uncertainty sampling in the latent space but misses to compare to uncertainty sampling in image space. Computing the latent space representation for each sample using a GAN is very costly and requires high quality GANs. Sampling lines in the latent space of multi-class problems might lead to many crossings. Such lines might be arbitrarily hard to annotate especially if many crossings are close and annotating a line is more costly than annotating a single image.

Thus, we propose ASAL that reuses the sample generation idea of Zhu and Bento [29] but we use information entropy as uncertainty score and directly extend it to multiple classes. Our main contribution is avoiding to annotate synthetic images by selecting the most similar samples from the pool with a newly developed sample matching method. We propose three different feature maps that we compute for each pool sample to fit a fast nearest neighbour model beforehand. During active learning, we compute the feature map of the synthetic sample and retrieve the most similar one from the pool in sub-linear time. Additionally, ASAL uses CNN based classifiers instead of linear SVMs. For the generator we train Wasserstein GANs beforehand [1].

3. Proposed Adversarial Sampling for Active Learning

ASAL allows using GANs for fast pool based active learning by generating samples and retrieving similar real samples from the pool, that will be labelled and added to the training set. Fig. 1 shows the main components of the proposed ASAL. (X^k, Y^k) denote the data set, used to train the classifier h with weights θ^k at active learning cycle k. Then, the trained classifier h_{θ^k} and the generator G enable producing uncertain samples $\tilde{x}$. The feature extractor F computes features that the nearest neighbour model uses to retrieve the most similar real samples from the pool. Finally, an oracle annotates the new real samples from the pool and adds them to the training set. Then, the new AL cycle $k+1$ starts.

In the remainder of this section, we introduce the adversarial sample generation and the sample matching strategy.

3.1. Adversarial Sample Generation using GANs

Instead of selecting uncertain samples from the pool, we follow Zhu *et al*. [29] and generate such samples using a trained GAN. Such a GAN enables to approximate the underlying data distribution of the pool. The discriminator D ensures that the samples drawn from the generator G are indistinguishable from real samples. At convergence, the generator produces the function $G : \mathbb{R}^n \to \mathcal{X}$ that maps the latent space variable $z \sim \mathcal{N}(\mathbf{0}_n, \mathbf{I}_n)$ to the image domain $\mathcal{X}$. The optimization problem that describes sample generation reads as follow:

$$\begin{aligned} \text{maximize} \quad & (H \circ h_{\theta^k})(x) \\ \text{subject to} \quad & x = G(z), \end{aligned} \tag{1}$$

where $H(q) := -\sum_{i=1}^{m} P(c = i|q) \log[P(c = i|q)]$ and m is the number of categories. Removing the constraint $x \in \mathcal{P}$ by including the generator simplifies the problem

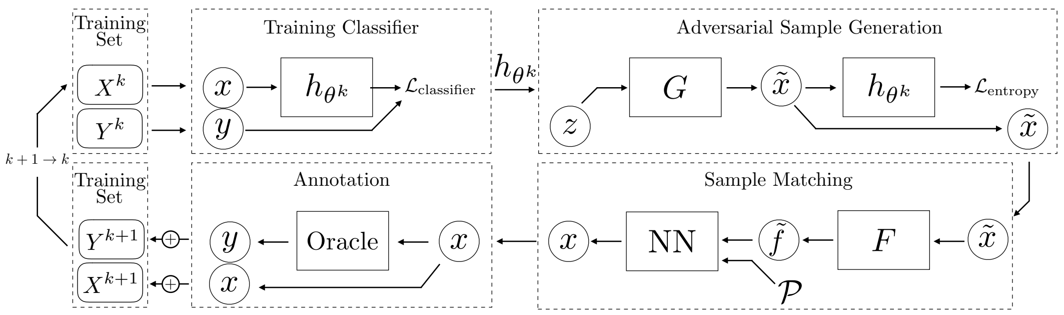


Figure 1: ASAL: (X^k, Y^k) with (x, y) is the training set at cycle k, h_θ is the classifier, z the latent variable, G the generator, $\tilde{x}$ the synthetic samples, F the feature extractor, $\tilde{f}$ the features, $\mathcal{P}$ the pool and NN the nearest neighbour method.

but changes its solution. New samples are no longer selected from the pool but are artificially generated. We solve the optimization problem in two steps: (i) we use the chain rule and gradient descent to minimize the objective with respect to z and (ii) we use G to recover a synthetic sample x from z. Thus, solving problem (1) has a constant run-time complexity $\mathcal{O}(1)$ because it is independent of the pool size. Note, that traditional uncertainty sampling achieves a linear run-time complexity $\mathcal{O}(n)$ (where $n = |\mathcal{P}|$ is the pool size) because it requires scanning each sample in the pool $\mathcal{P}$.

3.2. Sample Matching

Sample matching compares real pool samples to generated synthetic samples in a feature space and retrieves the closest matches. Therefore, we require a feature extractor $F : \mathcal{X} \to \mathcal{F}$, that maps data samples to a feature space, and a distance function $d : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_0^+$, that computes the distance between two data samples in this feature space.

Feature Extractors After the model generated meaningful features the task is selecting similar samples from the pool. In order to find suitable matches we need a feature space where nearby samples achieve a similar entropy (if entropy is the AL score used for sample generation). Naturally we would use the output of the $(l-1)$ layers of a CNN with l layers as features because entropy depends on these features : $F_{\text{CLS}}(x) = h_{\theta^k}^{l-1}(x)$. Unfortunately, the feature representation of each data sample becomes obsolete as soon as the weights θ of the CNN are updated. Thus, using the classifier to extract features for sample matching requires recomputing the feature representation of each data sample after each training iteration. This leads to a linear run-time complexity $\mathcal{O}(n)$. Thus, we propose to use feature extractors that are independent of the current weights θ^k such that we need to compute the features for each data sample only once in a pre-processing step. A feature space independent of the classifier does not guarantee entropy smoothness, *i.e.* samples with a nearby representation may have different entropy. However, perfect matches will have exactly the same entropy. Therefore, the closer the representations, the more likely they will score a similar entropy. However, this requires representative features for both: the real samples and the synthetic samples. Furthermore, we require, that the data set is sufficiently dense because for a sparse data set even the closest matches could be far away.

The image space is a simple feature space that uses the raw values of each pixel as one feature (RGB or gray values) $F_{\text{gray/rgb}}(x) = x$. The drawback is its large number of dimensions and that two visually close images with similar entropy that for example differ because of background intensity, small noise component, different scaling or small translations lead to far apart representations. Hence, we require a feature extractor that is mostly invariant to such perturbations. Thus, we propose to use the encoder of an auto-encoder to extract data set specific features. Furthermore, we can train these methods to extract features that are invariant to small perturbations in input images. We define the encoder and the decoder as follows $\phi : \mathcal{X} \to \mathcal{F}$ and $\psi : \mathcal{F} \to \mathcal{X}$ and minimize $\sum_{x \in X} \|x - (\phi \circ \psi)(x)\|_2^2$ to train the encoder and decoder. Thus, the feature extractor reads as: $F_{\text{auto}}(x) = \phi(x)$. Another feature spaces is defined by the features extracted by the discriminator. Training a GAN includes a discriminator that uses data-set specific features to solve the task of differentiating synthetic from real samples by assigning each input sample a probability how likely it is real or fake, $D : \mathcal{X} \to [0, 1]$. Thus, the features of the discriminator are not only suitable for real but also for synthetic samples and are data set specific. We propose to use the output of the $(j-1)$ layer of a discriminator with j layers as features: $F_{\text{disc}}(x) = D^{j-1}(x)$.

Efficient Feature Matching In order to find the best match we extract the features of the synthetic sample. Then, we retrieve the pool sample that has the most similar feature representation with respect to the distance function d. Sample matching reads as follows: $x = \arg\min_{x \in X} d(F(x), F(\tilde{x}))$, where $\tilde{x}$ is a synthetic sample. We propose to use the euclidean distance function $d(f_1, f_2) = \|f_1 - f_2\|_2$. This problem is equivalent to finding the nearest-neighbour of the synthetic sample in feature

Algorithm 1: ASAL

```
Input: Initialize the set $X, Y$ by adding random pool samples to $X^0$ and their labels to $Y^0$. Train the generator $G$ and the feature extractor $F$. Precompute the PCA, $\mu$ and the set $\mathcal{S} = \{F_{\text{PCA}}(x) \mid x \in X\}$.
Result: Trained Classifier $h_{\theta_k}$
repeat
    1. Train classifier $h_{\theta^k}$ to minimize empirical risk $R(h_{\theta^k}) = \frac{1}{|X^k|} \sum_{(x,y)\in(X^k,Y^k)} l(h_{\theta^k}(x), y)$.
    2. Generate synthetic samples $\tilde{x}$ with high entropy by solving Eq. (1).
    3. Compute the feature representations $\tilde{f}$ of the generated samples: $\tilde{f} = F_{\text{PCA}}(\tilde{x})$
    4. Retrieve real samples $x$ that match $\tilde{x}$ $x = \{p_i \in \mathcal{P} \mid i = \arg\min_{f\in\mathcal{S}} d(f, \tilde{f})\}$.
    5. Annotate the samples $x$ with labels $y$.
    6. Update the sets $X^{k+1} = X^k \cup \{x\}$, $Y^{k+1} = Y^k \cup \{y\}$.
until Labelling budget is exhausted;
```

space. Therefore, we use multi-dimensional binary search trees (k-d-trees) [2] for efficient nearest neighbour selection because their run-time complexity to search a nearest neighbour is sub-linear $\mathcal{O}(\log n)$ with respect to the pool size $n = |\mathcal{P}|$. However, the run time depends on the number of dimension of the feature space. Therefore, we achieve fast feature matching if $\dim(\mathcal{F}) \ll \dim(\mathcal{X})$. A property of auto-encoders is that they allow to compress samples from a high dimensional input spaces into a much lower dimensional latent space such that they enable fast sample matching. However, all other discussed feature spaces have typically a similar number of dimensions as the image space. Therefore, we propose Principal Component Analysis (PCA) to reduce the number of dimensions and to produce fewer features that contain most of the variance: $F_{\text{PCA}}(x) = \text{PCA}(F(x) - \mu)$, where $\mu = \frac{1}{|X|}\sum_{x\in X} F(x)$ and F is one of the previously introduced feature extractors. Then, the full sample matching reads as

$$x = \left\{ p_i \in \mathcal{P} \,\middle|\, i = \arg\min_{f\in S} d(f, F_{\text{PCA}}(\tilde{x})) \right\}, \qquad (2)$$

where $S = \{F_{\text{PCA}}(x) \mid x \in X\}$ can be precomputed before starting AL. We compress the features independent of the extractor to the same number of dimensions using PCA to ensure very similar absolute run-times of the nearest neighbour method. Alg. 1 shows a detailed description of the different steps of ASAL and Fig. 2 shows examples of synthetic samples with high entropy and their closest matches using the proposed matching with different features. We show more examples in the supplementary.

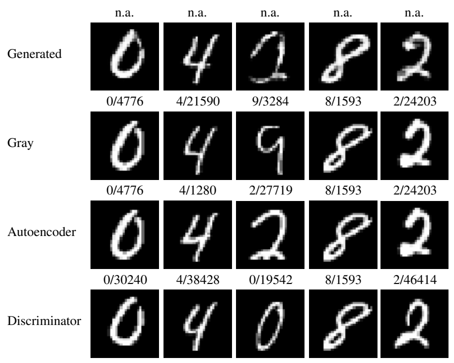


Figure 2: The rows show either generated or matched samples using different feature sets for *MNIST - ten classes*. The brackets denote (label id / sample id).

4. Experiments

4.1. Datasets

For the experiments we use five datasets: MNIST [14], CIFAR-10 [12], CelebA [15], SVHN [16] and LSUN Scenes [28]. The MNIST data set contains ten digits unevenly distributed. Each image has a resolution of 28×28 gray-scale pixels. The data set consists of 50k training, 10k validation and 10k testing samples. CIFAR-10 consists of 50k training and 10k validation 32×32 color images with uniformly distributed label categories. We use the validation set for testing. CelebA consists of roughly 160k training, 20k validation and 20k testing 64×64 color images and a list specifying the presence or absence of 40 face attributes for each image. SVHN consists of 73k training, 26k testing and 531k extra 32×32 color images with unevenly distributed label categories. We use the training and extra images to build the pool for AL. LSUN Scenes consists of roughly 10M training images with unevenly distributed labels. We split it into training and testing sets and centrally crop all the images to 64×64 color images.

For a fair comparison to GAAL, we follow Zhu and Bento [29] and construct the same binary data sets, consisting of the MNIST digits 5 & 7 and the CIFAR-10 classes *automobile* & *horse*. In addition we validate ADBA on the same MNIST data set. Furthermore, we validate ASAL on the full MNIST, CIFAR-10, SVHN and LSUN data sets and use four face attributes to build four different CelebA classification benchmarks. Each benchmark contains all 200k images, labelled according to the presence or absence of the attribute: *Blond_Hair, Wearing_Hat, Bangs, Eyeglasses*.

Table 1: Summary of all experiments. We run the experiment on CelebA four times but consider a different face attribute each time. *Budget* denotes the maximum amount of samples in the AL data set, *New* denotes the number of newly labelled samples in each AL cycle and *Initial* denotes the number of samples in the data set when AL begins. These three number allow computing the number of AL cycles.

Data set	Classes	Train Size	Classifier	GAN	Feature matching	Budget	New	Initial	AL Cycles	Seeds
MNIST	two	10k	Linear	WGAN-GP	Gray/Disc/Auto	500	10	50	45	5
CIFAR-10	two	10k	Linear	WGAN-GP	RGB/Disc/Auto	1000	10	50	95	5
MNIST	ten	50k	CNN	WGAN-GP	Gray/Disc/Auto	10k	50	100	198	5
CIFAR-10	ten	50k	CNN	WGAN-CT	RGB/Disc/Auto	30k	1000	1000	29	3
CelebA	two	160k	CNN	WGAN-GP	Auto-Encoder	2k	10	1000	190	5
SVHN	ten	604k	CNN	WGAN-GP	Auto-Encoder	50k	1000	1000	49	3
LSUN	ten	9604k	CNN	WGAN-GP	Auto-Encoder	30k	1000	1000	29	3

4.2. Experimental Settings

First, we produce different references to assess the performance of ASAL. (i) *Maximum-entropy sampling* (upper bound) because ASAL tries to approximate this strategy in sub-linear run-time complexity. (ii) *Random sampling* (lower bound, baseline) and (iii) the *fully supervised model* (upper bound). In addition we report for a subset of the experiments the results of Core-set based AL (MNIST & SVHN). We examine three different versions of ASAL using the previously introduced set of features: *ASAL-Gray/RGB*, *ASAL-Autoencoder*, and *ASAL-Discriminator*. For some settings we compare to *ASAL-CLS-Features* that uses the classifier features for matching. We reduce the dimension of the feature space to 50 using PCA. We experimentally verified that more dimensions only increase the run-time but lead to similar accuracy. Fig. 3d shows the test accuracy of *ASAL-Autoencoder* with three different number of PCA dimensions. To synthesize new samples we use Adam [11] and apply 100 gradient steps to maximize the entropy with respect to the latent space variable, see Eq. (1). We directly optimize for multiple latent space variables at the same time by embedding them in one batch. We always draw samples from the pool without replacement. We do not use data augmentation for any experiment except LSUN and train all models from scratch in each AL cycle. We run all experiments for five different runs with different random seeds and report the mean (solid line) except the computationally demanding experiments on *CIFAR-10 - ten classes* SVHN and LSUN, that we run for three random seeds. The shaded areas correspond to the maximum and minimum value for each operating point considering all random seeds. Please refer to the supplementary for the model architectures (classifiers, auto-encoders, and GANs), the training strategies and parameters. Tab. 1 summarizes all experimental setups. For the linear models ($h(x) = Wx + b$) we use directly the raw pixel values as input features for the classifier. We use Wasserstein GANs [1] with gradient penalty [6] and add a consistency term [24] for *CIFAR-10 - ten classes* because it produces synthetic samples with higher visual quality [24]. Note, that we use the same setup for CelebA for four different experiments but only change the target classification labels.

5. Results

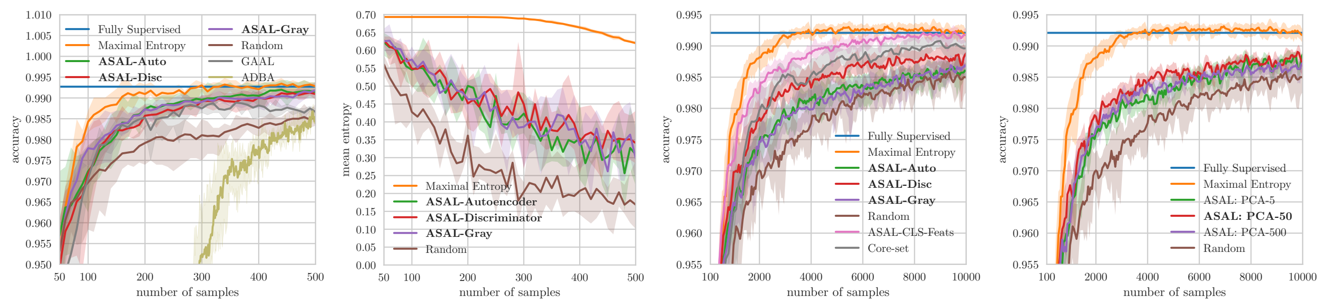
5.1. Linear Models

ASAL outperforms random sampling and approaches maximum entropy sampling quickly on *MNIST - two classes*. We observe, that all three proposed feature spaces for sample matching perform equally well. All ASAL strategies reach a classification accuracy of 98.5% with only 200 labelled samples, whereas random sampling requires 500 labelled samples, see Fig. 3a.

On *CIFAR-10 - two classes* only *ASAL-Autoencoder* exceeds the performance of random sampling. However, using auto-encoder features for sample matching reaches the classification accuracy of maximum entropy sampling already with 500 labelled samples, whereas random sampling requires approximately twice the amount of samples to reach a comparable performance, see Fig. 4a.

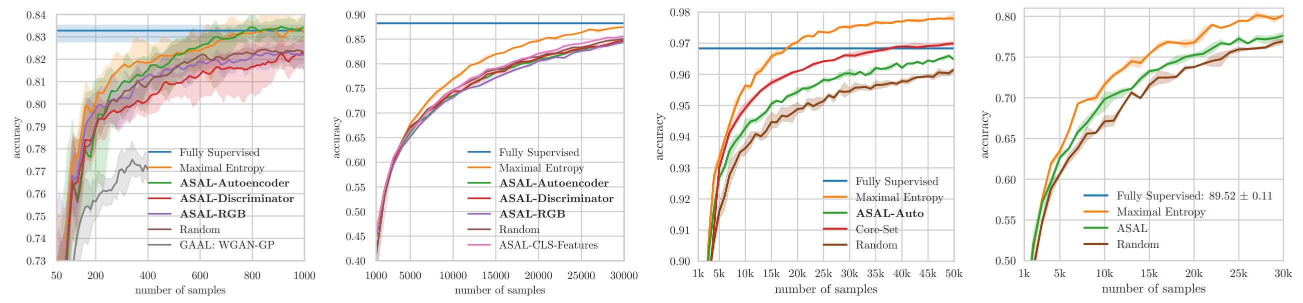
5.2. Convolutional Neural Networks

ASAL clearly outperforms random sampling on *MNIST - ten classes*. In contrast to the binary setting we used a CNN where the weights and thererfore the extracted features change in each AL cycle. Nonetheless, all three feature spaces, used for sample matching, exceed the performance of random sampling. However, the discriminator features lead to the highest classification accuracy, see Fig. 3c. Furthermore, we observe that *ASAL-discriminator* achieves almost similar test accuracy as core-set based AL but at a smaller run-time complexity. Unfortunately, ASAL performs similar to random sampling on *CIFAR-10 - ten classes* independent of feature space used for sample matching but classical uncertainty sampling exceeds the performance of random sampling, see Fig 4b. The four experiments using different target labels on CelebA emphasize, that ASAL outperforms random sampling and approaches



(a) Accuracy (MNIST - two classes) (b) Entropy of new samples. (c) Accuracy (MNIST - ten classes) (d) Small ablation study for PCA.

Figure 3: Test accuracy and entropy for different methods, data sets and benchmarks.



(a) CIFAR-10 - two classes (b) CIFAR-10 - ten classes. (c) SVHN - ten classes (d) LSUN - ten classes

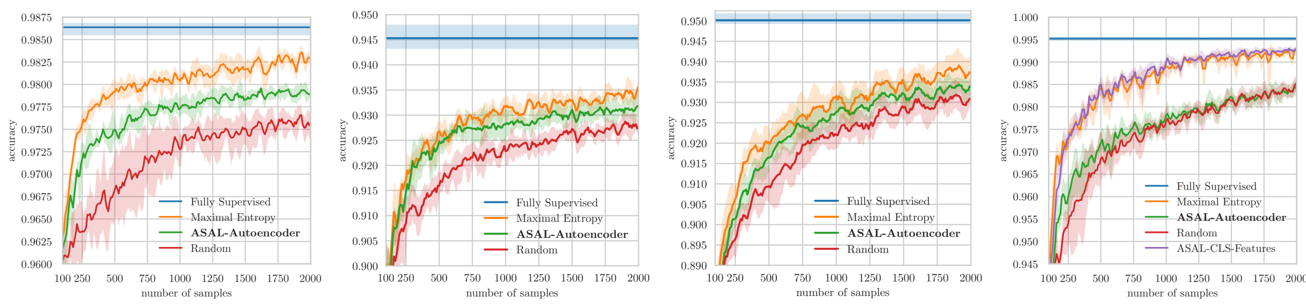
Figure 4: Comparison of classification accuracy between different methods on four different benchmarks.

uncertainty sampling with a better run-time complexity. However, for *CelebA - Eyeglasses* ASAL performs only marginally better than random sampling. ASAL exceeds random sampling during the first cycles but equals its performance when using more than 750 labelled samples, see Fig. 5. The results on SVHN in Fig 4c show that ASAL outperforms random sampling but achieves lower test accuracy than the more costly core-set based AL and uncertainty sampling. Similarly, Fig. 4d shows that ASAL outperforms random sampling on LSUN. We omit the comparison with core-set based AL on LSUN because it is demanding with respect to memory and leads to an even higher run time than uncertainty sampling. To summarize, ASAL outperforms random sampling on eight out of ten benchmarks. On *CelebA - Blond_Hair* and *CIFAR-10 - two classes* ASAL achieves almost the same performance as maximum entropy sampling. We will analyze the successful and the failure cases in Sec. 5.4 and give intuition when ASAL works.

5.3. Comparison between ASAL, GAAL and ADBA

The most similar method to ASAL is GAAL [29]. Even though Zhu & Bento [29] report that GAAL clearly performs worse than random sampling, we reproduced their results. For fairer comparison we replace their DCGAN with our Wasserstein GAN that we also use for ASAL and generate images with higher quality. Fig. 3a shows that GAAL achieves a higher accuracy than random sampling and an accuracy almost as high as ASAL at the beginning of AL. However, after adding more than 350 samples, the classification accuracy does not even saturate but drops and approaches the quality of random sampling. The reason for this decrease are generated samples where identifying the correct labels is very difficult such that the annotations get unreliable. Nonetheless, Fig. 2 shows that labelling synthetic samples is possible and therefore GAAL can work. Furthermore, we implement ADBA and use again the same GAN for sample and line generation. ADBA requires labeling transition points in generated lines of samples such that directly comparing the labeling effort is difficult. We count annotating one line as one annotation. Annotating one line leads to eleven labeled synthetic images. Thus, 500 samples in Fig. 3a correspond to 5500 labeled synthetic samples. The classifier is trained in the latent instead of the image space and requires much more annotations to achieve competitive results. Thus, we conclude that ADBA achieves worse performance than all other methods and is limited to binary classification with linear models in latent space. Hence, we omit further comparison with ADBA.

We reproduce GAAL on *CIFAR-10 - two classes* and observe that reliably labelling uncertain synthetic images is



(a) *Wearing_Hat* (b) *Blond_Hair* (c) *Bangs* (d) *Eyeglasses*

Figure 5: Test accuracy on four *CelebA* benchmarks. The target classes correspond to absence or presence of face attributes.

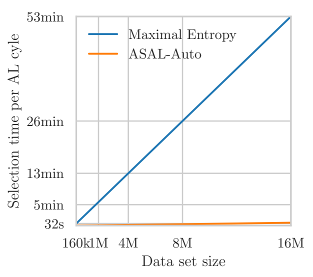
very difficult even with a state-of-the-art GAN. Fig. 4a reports the performance of GAAL on *CIFAR-10 - two classes*. We observe that GAAL performs clearly worse than random sampling and ASAL. We run GAAL only up to 400 labelled samples because the trend is clear and because manually labelling synthetic images is very costly and tedious. Thus, we conclude, that ASAL outperforms GAAL and ADBA.

5.4. Discussion

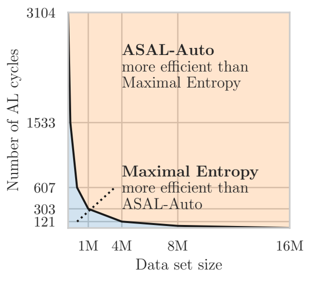
When designing ASAL we make several assumptions and approximations to enable a sub-linear run-time complexity. In this section we analyze the experiments of ASAL and investigate for the failure cases which of these assumptions hold and which do not. We assume that: *(i)* the GAN can generate synthetic images with high entropy that match the true data distribution, *(ii)* the data set is sufficiently large, such that there exists always a real sample that is close to each synthetic image, *(iii)* there exists a fixed feature space (independent of the classifier), where nearby representations have a similar entropy.

Fig. 3b shows that all three ASAL strategies retrieve on average samples with 63% higher entropy than random sampling on *MNIST - two classes*. We conclude that for this data set all assumptions hold. Especially the relatively large and dense data set with 10k samples that cover many variations of the digits enables reliable sample matching and leads to a well trained GAN. In Sec. 3.2 we described the feature extractor F_{CLS} that uses directly the CNN features. This feature space guarantees entropy smoothness such that nearby representations share a similar entropy. Using the best feature extractor F_{CLS} increases the run-time but avoids assumption (iii). Thus, if ASAL works only when using F_{CLS} we require a different feature extractor than the proposed. Furthermore, if ASAL fails with F_{CLS} features it indicates that the data set is too small such that training the GAN to generate uncertain samples with realistic features and matching is unfeasible. Indeed, Fig. 3c shows that ASAL on *MNIST - ten classes* using F_{CLS} approaches quickly the quality of maximum entropy sampling and verifies that F_{CLS} performs better than fixed features. It shows that using a non-optimal feature extractor reduces the performance but verifies that sample matching with the data set and synthetic samples works. We redo the same experiment on *CIFAR-10 - ten classes* and observe that using F_{CLS} only marginally exceeds the quality of random sampling. Therefore, our proposed sample matching and feature spaces are not the reason why ASAL fails but rather the small data set size. Hence, we expect that either the GAN generates synthetic samples with unrealistic characteristics or that the generated samples would be useful but close matches are missing in the data set. We redo the same experiments for the benchmark *CelebA - Eyeglasses* where ASAL fails. However, we already verified that ASAL works on three benchmarks on this data set and know that the quality of the synthetic uncertain images is sufficient. Fig. 5d shows that F_{CLS} achieves the same performance as maximum entropy sampling. Hence, the performance drop is caused by using the fixed instead of the varying features. Furthermore, the amount of images in the data set, that contain eyeglasses is very small such that the synthetic image might contain a face with an eyeglass and the matching retrieves a very similar face without eyeglasses. The issue is that the proposed feature extractor concentrates on many face attributes for matching but uncertainty depends only on a small subset.

SVHN is a less diverse data set than CelebA and CIFAR-10 but contains many more samples. Thus, the quality of generated samples is high and similar matching retrieves meaningful samples. Hence, all three assumptions hold. LSUN is the biggest tested data set in this paper but training a GAN to generate high quality samples is still challenging. Nonetheless, ASAL outperforms *random sampling*. Thus, the GAN is able to generate samples with features that help training and that are present in the data set too. Furthermore, matching is able to select similar samples because LSUN contains on average 1M samples per class. Thus, we conclude that ASAL can work with lower quality synthetic samples as long as they contain meaningful characteristics because we label matched real instead of generated samples.



(a) Timings for sample selection



(b) Transition point

Figure 6: Run-time of uncertainty sampling and ASAL to select 10 samples with respect to the data set size. The transition point denotes the number of AL cycles when ASAL gets more efficient than maximum entropy sampling.

5.5. Timings

In this section we report timings on CelebA and LSUN. For CelebA we first concentrate on the run time of one AL cycle with respect to different data set sizes. Next, we report the transition point after how many AL cycles ASAL gets more efficient than *uncertainty sampling* in case preprocessing time is taken into account. Finally, we report the run time of ASAL and other AL methods on LSUN including I/O-time. All measurements omit classifier training time because it is equivalent for all AL methods. We use a Nvidia TITAN X GPU and an Intel Xeon CPU E5-2680 v4.

Fig. 6a reports the time required to select ten new samples in each AL cycle with respect to data set size. We randomly augmented the original data set (160k) to create larger data sets containing up to 16M samples. Whereas it is possible to keep all images in memory for 160k (1.98GB) this is hardly possible for 16M images (198GB). For experiments on CelebA we omit including I/O-time. The sample matching proposed in ASAL stores only 50 float features per image (32MB for 160k and 3.2GB for 16M images). This saving enables to keep the features in memory and to build the nearest-neighbor model even for huge data sets.

ASAL has a sub-linear run-time complexity to select new samples. However, it requires several pre-processing steps such as training the GAN ($\sim$ 25h) and auto-encoder ($\sim$ 1.6h), extracting the features ($\sim$ 32s per 160k samples) and fitting the nearest-neighbor model ($\sim$ 5min for 16M samples). The sample selection time is $\sim$ 44s for 16M. Conversely, maximum entropy sampling avoids any pre-processing cost but has a much higher sample selection time: $\sim$ 53min for 16M. Fig. 6a shows that ASAL is much faster than uncertainty sampling but requires preprocessing. However, the time savings for ASAL in each AL cycle is large and allows to compensate for the initial pre-computation time when running ASAL for sufficiently many AL cycles. Fig. 6b shows the transition point, the point where ASAL achieves a higher efficiency than *maximum entropy sampling* depending on the data set size

Table 2: Run-time complexity and runtime for one AL cycle on LSUN including I/O-time ($n = |\mathcal{P}|$ refers to the pool size and $k = |X^k|$ to the number of labeled samples).

AL Method	Feature Extraction	Sample Selection	Runtime
Random	—	—	< 1s
Maximal-Entropy	—	$O(n)$	3660s
Learning-Loss [27]	—	$O(n)$	$\sim$ 3660s
Core-Set [19]	$O(n)$	$O(kn)$	$>$ 3660s
ASAL-Auto (ours)	$O(1)$	$O(\log n)$	471s

and number of AL cycles. Note, that the sample selection time for *uncertainty sampling* is independent of the number of selected samples but the run time of ASAL increases when selecting more samples. However, the sample selection time for ASAL is still much smaller than for uncertainty sampling even when querying much more samples. The reason is that we can generate many artificial samples within one batch at once. Note that selecting fewer samples reduces the risk of correlated uncertain samples.

Tab. 2 reports timings for AL methods including I/O time. We measure the time for ASAL, random and uncertainty sampling. For the other methods we predict the run time based on previous measurements: Learning-Loss [27] requires propagating each sample through the network each AL cycle and computing the learned loss. Hence, the runtime complexity is linear and the run time is similar to maximal entropy sampling. Similarly, Core-Set based AL[19] requires extracting the features for each sample every AL cycle to select new core samples. Note that in each AL cycle sample selection takes longer than training a classifier with less than 15k samples for methods with linear run time.

6. Conclusion

We proposed a new pool-based AL sampling method that uses sample synthesis and matching to achieve a sub-linear run-time complexity. We demonstrated, that ASAL outperforms random sampling on eight out of ten benchmarks. We analyzed the failure cases and conclude, that the success of ASAL depends on the structure and size of the data set and the consequential quality of generated images and matches. ASAL works exactly on the large data sets where it is most needed. There the sub-linear run-time compensates quickly for any pre-processing. ASAL is suitable for interactive AL where pre-processing is acceptable but small sampling times are required. In future research we propose to test ASAL for other AL scores such as the Learned-Loss [27] to accelerate their run time. Although auto encoder features work well for natural images we suggest to study VGG [20] or AlexNet [13] feature in future research.

Acknowledgments. This work was partly supported by ETH General Fund (OK) and by research grants from ASTRA and SBB.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. 2, 5

[2] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. 4

[3] C. Campbell, N. Cristianini, A. Smola, et al. Query learning with large margin classifiers. In *ICML*, pages 111–118, 2000. 1

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2

[5] I. Gulrajani. improved_wgan_training. \https://github.com/igul222/improved_wgan_training, 2018.

[6] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017. 5

[7] T. M. Hospedales, S. Gong, and T. Xiang. Finding rare classes: Active learning with generative and discriminative models. *IEEE transactions on knowledge and data engineering*, 25(2):374–386, 2013. 1

[8] M. Huijser and J. C. van Gemert. Active decision boundary annotation with deep generative models. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1, 2

[9] P. Jain, S. Vijayanarasimhan, and K. Grauman. Hashing hyperplane queries to near points with applications to large-scale active learning. In *Advances in Neural Information Processing Systems*, pages 928–936, 2010. 1, 2

[10] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *Conference on Computer Vision and Pattern Recognition*, pages 2372–2379. IEEE, 2009. 1

[11] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015. 5

[12] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009. 4

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 8

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4

[15] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015. 4

[16] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *Workshop on Deep Learning and Unsupervised Feature Learning, NIPS*, 2011. 4

[17] H. T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 79. ACM, 2004. 1

[18] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of the International Conference on Learning Representations*, 2015. 2

[19] O. Sener and S. Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018. 1, 8

[20] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR)*, pages 1–14, 2015. 8

[21] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *Proceedings of the International Conference on Learning Representations Workshops*, 2014.

[22] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001. 1

[23] X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang. CT-GAN. https://github.com/biuyq/CT-GAN, 2018.

[24] X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang. Improving the improved training of wasserstein gans: A consistency term and its dual effect. In *Proceedings of the International Conference on Learning Representations*, 2018. 5

[25] Y. Yang and M. Loog. Active learning using uncertainty information. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 2646–2651. IEEE, 2016. 1

[26] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113(2):113–127, 2015. 1

[27] D. Yoo and I. S. Kweon. Learning loss for active learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 8

[28] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 4

[29] J.-J. Zhu and J. Bento. Generative adversarial active learning. In *Advances in Neural Information Processing Systems Workshops*, 2017. 1, 2, 4, 6