Pacific Graphics 2019
C. Theobalt, J. Lee, and G. Wetzstein
(Guest Editors)

Volume 38 (2019), Number 7

Interactive Curation of Datasets for Training and Refining Generative Models

Wenjie Ye[1,2] Yue Dong[2] Pieter Peers[3]

[1]Tsinghua University
[2]Microsoft Research Asia
[3]College of William & Mary

(a) Rough Stone (b) Rusted Metal (c) Bedrooms with Twin Beds (d) Happy faces

Figure 1: *Examples of Generative Adversarial Networks (GANs) learned from user-curated training sets of (a) rough stone textures, (b) rusted metal textures, (c) images of bedrooms with twin beds, and (d) portraits of happy faces. Each of these GANs are trained on datasets curated using our interactive system in approximately* 12 *(stone) to* 35 *(face) minutes starting from a larger more general dataset of stones, metals, bedrooms, and faces respectively.*

Abstract

We present a novel interactive learning-based method for curating datasets using user-defined criteria for training and refining Generative Adversarial Networks. We employ a novel batch-mode active learning strategy to progressively select small batches of candidate exemplars for which the user is asked to indicate whether they match the, possibly subjective, selection criteria. After each batch, a classifier that models the user's intent is refined and subsequently used to select the next batch of candidates. After the selection process ends, the final classifier, trained with limited but adaptively selected training data, is used to sift through the large collection of input exemplars to extract a sufficiently large subset for training or refining the generative model that matches the user's selection criteria. A key distinguishing feature of our system is that we do not assume that the user can always make a firm binary decision (i.e., "meets" or "does not meet" the selection criteria) for each candidate exemplar, and we allow the user to label an exemplar as "undecided". *We rely on a non-binary query-by-committee strategy to distinguish between the user's uncertainty and the trained classifier's uncertainty, and develop a novel disagreement distance metric to encourage a diverse candidate set. In addition, a number of optimization strategies are employed to achieve an interactive experience. We demonstrate our interactive curation system on several applications related to training or refining generative models: training a Generative Adversarial Network that meets a user-defined criteria, adjusting the output distribution of an existing generative model, and removing unwanted samples from a generative model.*

CCS Concepts

• ***Computing methodologies*** → *Active learning settings;* ***Graphics systems and interfaces;*** • ***Human-centered computing*** → *Interactive systems and tools;*

1. Introduction

One of the grand goals of computer graphics is to support users in generating novel imagery. Often this process focuses on creating a unique exemplar such as a texture, portrait, or scene. However, creating intricate and detailed visuals from scratch is time-consuming and requires skill and practice. To alleviate this difficulty, artists often start from an exemplar or borrow parts from a set of exemplars. This idea has resulted in many example-driven authoring aids such as texture synthesis [RDDM17], Poisson image editing [PGB03], etc. Generative models take this paradigm a step further, and aim to not just create a single exemplar, but to provide an automated process for generating any number of exemplars that meet a set of user-defined criteria. Recent advances in deep learning have enabled users to specify a generative model by providing a (large) set of exemplars. Such deep learning based approaches can be seen as learning a distribution (characterized by the training dataset), and generating an exemplar can be seen as drawing a sample from the learned distribution. A popular class of deep learning based generative strategies are *Generative Adversarial Networks* (GANs) [GPAM*14], which can learn to generate high quality samples with impressive accuracy from just an *unlabeled* training set of exemplars. However, the accuracy and quality of the learned distribution greatly depends on the characteristics of the training dataset. While gathering a general non-discriminative training dataset can be automated via web crawlers with relatively little effort, sifting through the collected data to eliminate unsuited and corrupted exemplars and/or select exemplars that meet user-defined criteria is arduous and time-consuming.

In this paper, we endeavor to empower the user to easily curate training datasets such that the resulting GANs better match the users' expectations. We a priori assume that the user has access to a large general dataset of possible training data; we do not assume this database has been pre-screened nor do we assume that it only contains the target distribution. By enabling the user to tailor the training dataset, we empower the user to indirectly control the output distribution of the generative model. The source from which the user selects the training set does not need to be in the form of an image dataset, and it can also be in the form of samples drawn from another, more general, generative model, effectively allowing the user to refine the output distribution of an existing generative model. However, the user's desired output distribution might be difficult to explicitly express concisely in "rules" or they might be subjective. Hence, it will be essential for the user to directly participate in the selection process of the training dataset.

We propose an interactive system for curating datasets driven by, possibly subjective, user-defined criteria for training or refining generative models. At its core, our system regresses a classifier that can discriminate desired from undesired training exemplars. Learning such a classifier requires knowledge of the user's preferences in the form of labeled exemplars, which are obtained via a novel interactive curation framework that builds on *active learning* [Set10] to minimize user labeling effort. Typically, active learning proceeds in an iterative fashion where in each round, the user is asked to label a small set of the candidates which are selected based on information learned from previously labeled candidates. Once a classifier is obtained, we filter the large input dataset and extract the exemplars that match the user's intent, which are then subsequently used to train or refine the generative model.

A key design constraint in our interactive system is that we rely on a physical user's feedback. This has two important consequences. First, due to the high training cost, deep neural network classifiers are best refined with batches of data as opposed to exemplar by exemplar as in classic (non-neural network) active learning. Consequently, we will rely on batch-mode active learning and present a batch of candidates to the user to label. The candidates in each batch are selected based on each candidate's potential to improve the classifier. However, a too large batch will be too time-consuming and too overwhelming for the user to label in one pass. Hence, a smaller batch size is more amendable for users to process. However, the smaller the batch of candidates, the more important the diversity among the candidates becomes in addition to their potential for improving the classifier. In this paper we encourage a diverse set of candidates using a novel disagreement metric suited for small batches. A second consequence is that we cannot assume that the user is a perfect oracle that can always make a binary decision (i.e., "meets" or "does not meet" the selection criteria) for each presented candidate, either because it is a difficult case (i.e., the user is uncertain) or because the user does not care whether the sample is included or not in the training set and thus output distribution of the generator. We therefore allow the user to not only mark positive examples (i.e., that meet the selection criteria), but also mark samples as "undecided". As a consequence, care needs to be taken when selecting new candidates to be presented to the user. We will employ a novel non-binary query-by-committee strategy [SOS92] to differentiate between the uncertainty of the user's intent versus the uncertainty of the classifier.

We demonstrate the efficacy of our method on a number of different datasets and selection criteria, and we propose four applications that employ our interactive curation system: (1) modeling a targeted GAN based on user-defined criteria, (2) cleaning training data, (3) changing the density of the learned distribution modeled by a GAN, and (4) removing unwanted exemplars from a GAN.

In summary, our contributions are:

1. An interactive system for curating datasets;
2. four applications of our system to aid in training, correcting, and refining generative models; and
3. a novel non-binary batch-based query-by-committee active learning method for classification that is more broadly applicable than curating training data for GANs.

2. Related Work

To the best of our knowledge, our work is the first to tackle the problem of interactively curating datasets for training or refining deep generative models with complex distributions. We will therefore focus our discussion of related work on five topics that share common traits with our work: generative adversarial networks, texture synthesis with deep generative models, interactive design with deep generative models, example-based procedural modeling, and active deep learning.

Generative Adversarial Networks Generative Adversarial Networks (GANs) [ACB17, GPAM*14, RMC15] are a type of deep

neural network architecture that consists of a generator network and a discriminator network. The generator provides a mapping from an (random) input vector to a learned distribution, making it ideally suited for generating (synthesizing) random samples from the distribution (i.e., evaluating the generator for a random input vector). The role of the discriminator is to judge whether a sample belongs to the distribution or not. A key distinguishing feature of GANs is that both networks, the generator and discriminator, are trained in competition: each network tries to outdo the other. To successfully train a GAN a large collection of samples from the input distribution is needed. While GANs do not require explicit labeling of the training data, it makes the implicit labeling assumption that all the training data belong to the target distribution. For truly unlabeled training data, gathering such a training set is labor intensive. Our work is complementary to existing research on GANs; we focus on accelerating and simplifying the process for selecting an appropriate set of training samples.

Texture Synthesis with Deep Generative Models A natural application of GANs is texture synthesis. Zhou et al. [ZZB*18] use a GAN to double the resolution of a non-stationary texture. Bergmann et al. [BJV17] employ a fully convolutional GAN to learn the texture manifold from a single high resolution image or from a dataset. Jetchev et al. [JBS17] use a GAN to learn mosaic images based on a guide image. Related to this, is the work of Frühstück et al. [FAW19] who combine texture patches generated by a GAN in a seamless fashion. However, all these methods either assume that a single exemplar or a dataset has been selected by the user. The proposed method is complementary to these methods, and the resulting GANs trained from curated training sets can be employed in a texture synthesis framework.

Interactive Design with Generative Models Generative models have also been employed to aid, among others, in editing and authoring natural images [ZKSE16], faces [PHS*18], terrains [GDG*17], and buildings [KGS*18]. These methods employ a GAN to design a single instance and the generative model is used to ensure that the edits remain plausible. Our method in contrast aims to empower the user to steer the output *distribution* of a generative models via its training data rather than manipulating single instances. However, the resulting GANs can afterward still be employed with GAN-based editing tools.

Example-based Procedural Modeling Another class of generative models are procedural methods. However, procedural models are notoriously difficult to create. Lagae et al. [LVLD10] introduced an example-based solution for multi-resolution noise for isotropic stochastic textures. Galerne et al. [GLLD12] use Gaussian example textures to control Gabor noise. Both methods take a single input image as an example, and are more akin to texture synthesis; in contrast our goal is to curate a dataset of many exemplars used to train generative models that sample complex distributions beyond textures.

Active Deep Learning Our work builds on results from active learning that aims to learn a task at higher accuracy with fewer labeled training data. Active learning incrementally grows a small subset of labeled training data by querying an oracle to label a candidate training sample. We refer to Settles et al. [Set10] for an extensive overview of (pre-deep learning) active learning methods. Recently, researchers have started looking at extending active learning to deep learning [WZL*17, SS18, SED19]. In contrast to many classic active learning strategies that grow the set of labeled training data one by one, due to the high training cost, deep networks are best trained in batches.

Our work differs to prior work in two critical aspects related to including a *real* human in the loop. First, including a human as oracle restricts us practically to small batch sizes that can be processed effectively by the user (in contrast, Sinha et al. [SED19] use batch sizes of 5% ($\sim$ 1000s of samples) of the total training set). For small batch sizes it is not only the informativeness of each candidate in the selected batch that is important, but also the diversity of the candidate set is critical (i.e., how different are the different candidates in each batch). We introduce a novel metric to select a diverse set of candidates. Second, prior work assumed a, possibly noisy, oracle that can provide a decisive label. In contrast, we account for cases where the oracle (i.e., user) is indecisive and cannot assign a label to a candidate. This implies that we cannot simply use the uncertainty of the classifier as a sampling guide (e.g., as in [WZL*17]). Instead we build on Query-by-Committee [SOS92] (also named ensembles in the context of deep learning), which has been shown to be a robust selection algorithm for active deep learning [BGNK18]. However, in contrast to Beluch et al. [BGNK18] we also take diversity in account.

Similar to our work, Zhu et al. [ZB17] also combine GANs and active learning, and use a GAN to generate new samples to train a classifier. Our work is in some sense the inverse; we actively learn a classifier to filter exemplars for training a GAN.

3. Interactive Curation by Intent Learning

Our interactive system for curating datasets for training or refining GANs builds on the idea that the training set should reflect the intent of the user's preferences. However, letting the user manually mark the whole training set is labor intensive and cumbersome. Instead, we endeavor to learn the user's intent interactively. We will model the user's intent as a convolutional neural classifier with fully connected layers that outputs the probability that the input exemplar meets the user's selection criteria. We train this classifier in an iterative fashion where we alternate between querying the user's selection criteria on exemplars to refine the classifier, and selecting novel candidates for the user to label based on the current classifier's state. Once the user's intent is modeled in the form of a classifier, we can then filter a set of candidate training exemplars for the instances that match the user's selection criteria, and use these to train or refine a GAN.

Our interactive system for learning the users' intent consists of two parts: (1) the interactive interface for querying the user's intent (subsection 3.1), and (2) the adaptive selection of candidates used in each round of the interactive system for refining the classifier (subsection 3.2). Our key concern in designing the interactive system is that we want to minimize the user's labeling effort while at the same time maximize the relevant information for learning the user's intent.

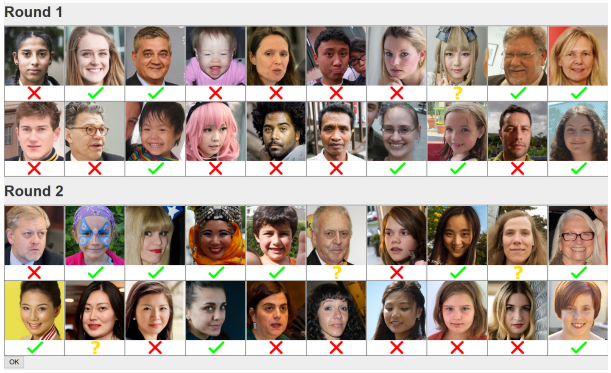


Figure 2: *Example of the interactive user interface shown for the 2nd round of selecting* "happy faces" *as a selection criterion. At each round the user marks exemplars that "meet" (✓), "do not meet" (✗) the selection criteria, and exemplars for which the user cannot decide (?).*

3.1. Interactive Interface

Our interactive interface is simple but intuitive (Figure 2). A small set of 20 exemplars are presented to the user for labeling (i.e., marking exemplars that meet the user's selection criteria). The selection criteria can range from simple conditions such as "only wood textures" to more complex criteria such as "faces that appear happy". However, for many criteria, the user's intent might be subjective with soft boundaries (i.e., there are many expressions between "happy" and "not happy"), or the user is selecting a subset from a continuous range of criteria (e.g., "straight" versus "curved" wood grain), or the user might not care whether a particular exemplar is included or not. For such cases, the user might find it difficult to decide whether a particular exemplar meets the envisioned selection criteria. Therefore, in addition to marking positive exemplars and negative exemplars, we also allow the user to mark an exemplar as "undecided". We will show in section 5 that adding such a third category improves the quality and efficiency of our interactive curation strategy.

3.2. Adaptive Candidate Selection

We aim to present the user with a small set of candidates that, when labeled, improves the accuracy of the classifier the most. This implies that we should sample the exemplars in the regions where the decision of the *classifier* is least certain.

Query-by-Committee A naive strategy for selecting a batch of candidates for the user to label would be to sample exemplars from the input set for which the classifier yields an approximately 50% probability that it matches the user's intent. However, this probability does not express the (un)certainty of the classifier, it only expresses the user's uncertainty. Hence, little is gained by asking the user again to label an exemplar for which the user has trouble making a decision.

Instead, we will approximate the uncertainty of the classifier by adapting a classic method from active learning, namely *Query-by-Committee* (QBC) [SOS92]. The overall idea is to train multiple classifiers, each initialized differently, over different training data sequences. Essentially, we aim to compel the different classifiers to converge to different local minima (if present). These multiple classifiers form a committee and we select the candidate exemplars presented to user based on the output of each committee member (i.e., classifier). Intuitively, if the committee unanimously agrees on a certain exemplar, then this indicates that that exemplar is likely reliably classified. Conversely, if the committee disagrees, then it indicates that we have found an unstable classification that can easily be flipped by making minor adjustments to the training. Hence, a (conclusive) judgment by the user will strengthen the quality of the classifiers for this exemplar (i.e., make the decisions more unanimous).

Disagreement Measure Our classifiers are trained to output a probability of whether the input meets the user's intent. As in classic Query-by-Committee, we could convert this probability to a binary decision to measure the disagreement. However, the continuous probability values (e.g., the user's degree of uncertainty) provide information too. We therefore use KL-divergence [KL51] to measure the degree of disagreement [MN98]. Given an exemplar x, we denote the output of the i-th classifier as $P_i(x)$. The consensus, or average, probability of the committee is denoted as $P_c(x) = \frac{1}{N}\sum_{i=1}^{N} P_i(x)$, with N the size of the committee (i.e., the number of classifiers). The disagreement between the committee members can then be measured by:

$$D(x) = \sum_{i=1}^{N} d_{KL}(P_i(x)||P_c(x)), \tag{1}$$

where $d_{KL}(X||Y)$ is the KL-divergence.

Disagreement Distance Our interactive system presents the user with a number of candidates in batches instead of a single exemplar with large disagreement. The reason is twofold. First, it will be easier for a user to decide whether an exemplar follows their intent if it is observed in relation to other exemplars. Second, our goal is to create an interactive system. Refining the classifier is a time consuming operation. Adding exemplar by exemplar as in many active learning methods would be computationally too expensive. Adding a batch of exemplars (e.g., 20 as in our implementation) distributes the cost of refinement over multiple exemplars.

Care should be taken when selecting candidate exemplars. Ideally, each exemplar, when labeled by the user, should maximize the potential of improving the classifier. While for a single exemplar, selecting a sample with largest disagreement (Equation 1) achieves this goal. However, when working in batches, then not only should these candidate exemplars be selected with respect to the previously labeled data (i.e., disagreement), but also to the current set of candidate exemplars (i.e., diversity). If we are to only select based on disagreement, then two exemplars with similar disagreement are likely to contribute the same cues to the classifier. Hence, we want to maximize the minimum variance in disagreement between the exemplars:

$$V(x) = \min_{x_i \in \mathbf{S}} d_c(x, x_i), \tag{2}$$

where $\mathbf{S}$ represents the set of exemplars already labeled by the user *and* the candidates already selected in the current round respectively. $d_c(\cdot,\cdot)$ expresses the difference in disagreement between two

exemplars, which we define as:

$$d_c(x_0,x_1) = \sum_{i=1}^{N} (P_i(x_0) - P_i(x_1))^2. \quad (3)$$

Candidate Selection Our selection procedure is a two step process. We first randomly sample a large set $\mathbf{T}$ of X_t potential candidates from the source database or source GAN. In our implementation we set $X_t = 5{,}000$. This initial sampling step serves two roles: (1) to support using a continuous distribution (e.g., GAN) as input, and (2) to speed up the selection process. Next, we progressively select $X_c = 20$ exemplars that maximize the disagreement (Equation 1) and the variance (Equation 2); we normalize both measurements and combine them via the harmonic mean:

$$\Phi(x) = \left(\frac{\sum_{i \in \mathbf{T}} D(x_i)}{D(x)} + \frac{\sum_{i \in \mathbf{T}} V(x_i)}{V(x)} \right)^{-1} \quad (4)$$

The candidate that maximizes Equation 4 is added to the candidate set $\mathbf{S}$, and we repeat the process until X_c candidate exemplars are added. Note that for an already selected candidate x_a, the resulting $V(x_a) = 0$, and thus it will not be considered for selection again.

3.3. Implementation & Optimization

In addition to the above core algorithm, we apply a number of algorithmic optimizations to further improve the interactivity and robustness of our dataset curation method.

Bootstrapping As long as the classifier is not trained with at least one positive (i.e., matching the user's intent), and one negative (i.e., not matching the user's intent) exemplar, the classifier is not a good oracle to drive the adaptive selection process. We therefore bootstrap the design process by presenting the user X_c exemplars sampled at random. Only when at least one positive and one negative exemplar has been labeled by the user, do we switch to the adaptive selection process.

Reference Exemplars In certain use cases, the user might want to provide an exemplar that meets (or does not meet) the selection criteria. Such guidance exemplars can trivially be included in the labeled training set for the classifier. Note, these exemplars do not need to be part of the source database or source GAN, making this a convenient alternative for (speeding up) bootstrapping.

Parallelization of User Interaction and Candidate Selection Ideally we would like to execute the user interaction component, training of the classifier, and the adaptive candidate selection component of the algorithm sequentially. However, classifier training and the adaptive candidate selection are not instantaneous and the user is left waiting for the classifier to be refined and for the candidates to be selected. We therefore parallelize both interaction and processing; we refine the classifier and generate the next batch of candidate samples while the user is labeling the current batch of candidates. As a consequence, the classifier training and adaptive candidate selection do not use the labeled data from the same round, and only rely on labeled data from prior rounds. We found that this parallelization does not the impact the final accuracy significantly, while at the same time significantly reducing the wait time for the user.

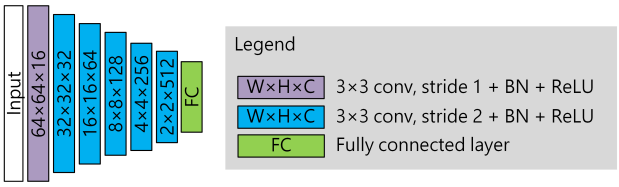


Figure 3: *Texture classifier network architecture.*

Adaptive Determination of X_t In the first stage of the adaptive selection process we randomly presample X_t exemplars from the source distribution. This presampling enables continuous sources such as GANs, while at the same time it also accelerates the selection process for a large source dataset. The number of exemplars (i.e., X_t) sampled can significantly impact the performance as well as efficiency. If X_t is too large, then computing the candidate subset can be costly. If X_t is too low, then the candidate selection might not produce the best exemplars. However, exactly what number is "too low" depends on the user's selection criteria and the distribution of exemplars that meet this criteria in the source distribution. For example, if only few samples match the user's intent (or conversely, if very few do not match the user's intent), then X_t needs to be large enough so that such rare exemplars are considered for presentation to the user. To alleviate this issue, we employ an adaptive sampling strategy that strikes a balance between accuracy and speed. We start by first selecting $X_t = 5{,}000$ exemplars to obtain an initial set of candidates. Next, we measure the disagreement of the selected candidates, and if the KL divergence of any selected candidate is smaller than 0.05, we discard the selected batch and sample a new set of 5,000 samples, and repeat the process.

3.4. Architecture & Training

Classifier Network Our dataset curation methodology is flexible and can be applied to different types of data. The exact architecture of the classifier depends on the type of data and resolution in the input dataset or GAN. For the results in this paper we use the following data types and classifier architectures:

- For textures, we train, from scratch, a classifier network consisting of a convolutional neural network with 1 fully connected layer. Figure 3 summarizes the network architecture of the texture classifier.
- For generic images (e.g, of bedrooms as in Figure 1), we train a 2-layer fully connected network on top of an embedding consisting of the feature map of *pool5* from VGG16 [SZ15].
- For face images we train a 3-layer fully connected network to output the classification on top of a pre-trained embedding [SKP15]. Using a pre-trained embedding enables us to use a smaller classification network, thereby improving interactivity and accuracy.

During classifier training, we form batches with a balanced number of positive and negative exemplars. We do not use exemplars marked by the user as *"undecided"* as these can be marked so for different reasons: e.g., hard to decide, or exactly in the middle between positive and negative, etc.

We implement our interactive curation system using Tensorflow [AAB*15]. For each example, we use 4 classifiers to form a committee, and train them efficiently in parallel on a 4-GPU configuration using Adam [KB15], with a 10^{-4} learning rate; other parameters are kept to their default settings. Each classifier is initialized with a different random seed, and for each iteration, a different training batch is generated for each classifier (albeit from the same labeled set). In practice we use a batch size of 32 for all our experiments and uniformly sample labeled exemplars. For all the experiments, we train 2,500 iterations per round of interaction, except for the initialization round where we train for 5,000 iterations.

Depending on the data source and type, we apply different training data augmentation strategies:

- For textures, we collect datasets of wood, metal, and stone textures from Mayang's Textures [Ma10] and Adobe Stock Images [Ado19]. During training we crop random patches at 64×64 resolution.
- For face images, we use the FFHQ [KLA19] and CelebA [LLWT15] datasets. For FFHQ we downsample to a 256×256 resolution, and randomly crop a 236×236 patch that is resized to the resolution expected by the classification network. For CelebA we crop 160×160 patches within the central face area (i.e., center 176×176 window). Furthermore, we apply a random intensity and contrast change within a range of $\pm 5\%$.
- For bedroom images, we use the LSUN dataset [YZS*15]. We randomly crop a 236×236 window which is then resampled to the resolution the classifier expects. We also apply the same random intensity and contrast augmentation within a range of $\pm 5\%$.

Generative Adversarial Network Our interactive curation strategy is agnostic to the GAN architecture and/or training strategy. When the input source is provided as a GAN (instead of a dataset of input exemplars), then instead of training a new GAN from scratch in the last stage, we finetune the input GAN based on the filtered training data (sampled from the input GAN). Finetuning can significantly reduce the training time, especially for high resolution GANs.

For the results shown in this paper we use the following GAN architectures:

- For textures, we train a Wasserstein GAN [GAA*17] for each kind of material, i.e., wood, metal, and stone.
- For both the face and bedroom images, we use a pre-trained StyleGAN [KLA19].

4. Applications

Our interactive dataset curation system for training or refining generative models is general and suitable for a diverse range of applications. In this section, we demonstrate this versatility on four possible GAN applications: creating a targeted generative model based on user-defined criteria, cleaning training data, modifying the distribution of a generative model, and removing unwanted samples from an existing generative model.

Targeted Generative Model Creation A direct application of our framework is to use the interactive curation system to create a targeted generative model based on user-specified criteria. Given a large database of exemplars, or alternatively a more general GAN, we first learn the user's intent using our system. We combine the trained classifiers by ensembling their outputs [HS90]. Next, we sample the input distribution uniformly and retain the positive classified exemplars. Finally, given this filtered training data we train a novel GAN or, in case of an input source GAN, refine it.

Figure 1 shows four examples of created GANs that synthesize: rough stone textures, rusted metals, bedrooms with twin beds, and portraits of happy faces. For each GAN we show a collage of randomly generated exemplars. Each of these GANs are trained from datasets curated from one of the general datasets listed in subsection 3.4. Our adaptive and interactive system is more effective than having the user label an equal number of random exemplars from the input dataset (Figure 4).

It should be noted that the quality of the created GAN can only be as good as the training data. A large source dataset does not guarantee a sufficiently large training set if the user's intent is only sparsely represented. As a general guideline, if the input dataset is small, or if there are few exemplars in the dataset that match the user's intent, then we first train a general GAN (which in theory can generate an infinite number of samples) and use this as the input training set.

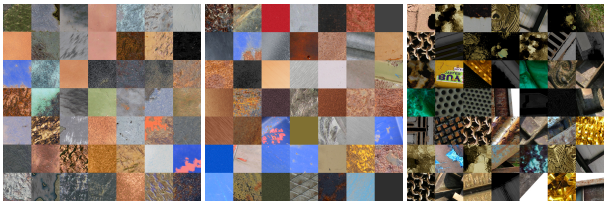
Cleaning Training Data Training data for deep learning are commonly mined from uncontrolled internet sources. Such datasets often contain exemplars that have artifacts or that do not match the search criteria. Our system can be used to remove such unwanted training data. In this case, we train a classifier to distinguish between good and bad training data. Figure 5 demonstrates this on the metal texture dataset that consists of metal textures cropped from larger images. Some of the cropped patches only contain a part of a metal texture or contain other (non-metal) textures.

Distribution Modification Our interactive system can also be employed to modify the output distribution of a generative model. For example, the user might want to reduce the occurrence probability of a certain pattern in a texture synthesis GAN.

As before, we follow the same selection process. We first learn the subset for which the user would like to change the occurrence probability, and allow the user to set the new occurrence ratio P_u of the selection versus the remainder. Next, we sample a large training set from the input GAN, and store them in two pools based on the trained classifier. During GAN finetuning, we select training data according to the user's indicated distribution. E.g., if the user indicated that the selected subset should comprise 66% of the training set, we sample 66% of the samples from the *selected* pool, and 34% from the *unselected* pool. Since we only indirectly control the output distribution (via the sampled input distribution), the resulting output distribution from the refined GAN does not necessarily mimic the desired distribution exactly. We therefore, estimate the output distribution by sampling the GAN and computing the rate P_g at which the samples pass the classifier. We then refine the GAN again with an adjusted sampling ratio of $2P_u - P_g$. We repeat this process until the desired ratio is met (within a threshold of 5%). Due to the stochastic nature of the GAN training process, the resulting ratio fluctuates over subsequent iterations. We therefore monitor the output occurrence ratio every 100 training iterations

(a) GAN obtained from interactively curated candidates (b) GAN obtained from 600 random selected candidates

Figure 4: *Comparison of a GAN created from a training set selected with a classifier obtained with our interactive system versus training a classifier on an equal number (*600*) of randomly selected exemplars using* "portraits of females with glasses" *as the selection criterion. The latter exhibits more exemplars that do not match the selection criteria; failure cases are highlighted in yellow.*



(a) Generated textures (b) Selected training data (c) Discarded training data

Figure 5: *Our interactive system is also suited to clean up training data mined from unreliable sources. In this example we remove unwanted texture patterns and incompletely cropped patches. Textures generated by the resulted GAN (a) closely matches the desired metal textures (b) and contains none of the unwanted patterns (c).*

and find the generator with the best matching ratio that follows the user's intent.

Two examples of altered distributions are shown in Figure 6 where we change the occurrence of wood textures with a cold hue to 25%, and of baby portraits to 50%.

Unwanted Sample Removal The training process of a GAN attempts to learn and mimic the distribution of the input exemplars. However, due to local minima in training, or missing or corrupted training exemplars in some regions of the target distribution, this can result in a GAN whose output distribution does not match the user's intent. Our interactive framework can also be used to identify unwanted samples and remove them from the GAN.

Similar as before the user marks wanted and unwanted exemplars sampled from the input GAN, and an appropriate classifier is trained. We can then synthesize training data by sampling the original GAN and checking the validity of each sample with the trained classifier. Using this synthesized training set, we then finetune the original GAN, such that the resulting GAN exhibits less unwanted samples.

Figure 7 shows a bedroom GAN. Highlighted are three images that exhibit artifacts and which are marked as "unwanted". These unwanted samples are replaced in the improved GAN (sampled with the same random latent codes).

5. Discussion

The applications in section 4 demonstrate that our interactive framework for curating training sets can yield GANs that qualitatively match the user's intent. In this section we quantitatively validate the accuracy as well as examine the impact of each of the components and parameters of our interactive system.

5.1. Quantitative Validation

We validate our interactive curation system on two dataset types: (wood, metal, and stone) textures and face images. To quantitatively compare the quality of the results, we employ different predefined selection criteria for which a ground truth classifier is available. For the texture datasets, we consider the following easily computable selection criteria:

- **Contrast:** we compute a measure of contrast as the standard deviation over all gray-scale pixels of an image. We set the *high contrast* selection criterion as having a contrast larger than 9.5. The *contrast low* selection criterion is set to an upper contrast threshold of 8. Contrast values in between 8 and 9.5 are marked as "undecided".
- The **hue cold** selection criterion is defined as having an $H \in (45, 315)$ and $S \in (0.18, 1.0)$, where H and S are the average Hue and Saturation values of the image in the HSV color space. Images with $H \in (0, 35) \cup (325, 360)$ or $S \in (0.0, 0.14)$ are marked as not meeting the selection criteria. All other images are marked "undecided".
- **Directionality** is computed as the ratio of the largest directional Gabor filter response [Gab47,FS89] over the smallest directional Gabor filter response. The directional Gabor filters are computed for 8 evenly distributed directions and with Gabor filter parameters $\sigma = 5$, $\lambda = 5$, $\psi = 0$, and $\gamma = 1$. We consider the texture

(a) Modified wood texture GAN (25% probability for *"cold hue textures"*) (b) Modified face GAN (50% probability for *"baby faces"*)

Figure 6: *Modifying the GAN distribution by changing the occurrence of certain types of instance types: (a) changing the probability of wood texture with a cold hue to a* 25% *occurrence rate, and (b) changing the probability of baby portraits to* 50%.

(a) Original GAN (b) Improved GAN

Figure 7: *Example of removing unwanted samples from a GAN distribution. (a) Original GAN with unwanted samples highlighted. (b) Improved GAN refined from the original GAN without the unwanted samples.*

Figure 8: *Exemplars from a custom wood GAN that generates "red wood with detailed grain". Note that labeling a small random set of exemplars from the input dataset fails to produce a reasonable classifier for this selection criterion, showcasing the necessity of our system for some selection criteria.*

to be *directional* if the directionality is larger than 5, and *non-directional* if smaller than 4; as before anything in between is marked as "undecided".

- A texture is considered **horizontal** if the ratio of the horizontal Gabor filter response over the vertical response is larger than 3. We consider any ratio less than 2.5 as *not horizontal*, and in between 2.5 and 3 as "undecided".
- **Manual Selection Criterion:** to avoid bias to computable selection criteria, we also create a manually marked selection criterion that roughly corresponds to *"red wood with detailed grain patterns"* (Figure 8). To create a ground truth classifier, we manually mark 22,000 exemplars and use these as the training set for a reference neural network classifier. We use the probability output to determine the label. We set a positive label when the probability is larger than 0.5, a negative label when less than 0.2, and "undecided" for anything in between.

For the face dataset we define ground truth reference classifiers based on the labels in the CelebA dataset which we regard as correct. Since the CelebA labels are binary, we do not consider an "undecided" label for the validation tests on faces.

We use the ground truth classifier also as an oracle for labeling the candidate exemplars during the interactive selection process in

	TAR		
	FAR 0.01	FAR 0.05	FAR 0.1
Wood texture			
Low Contrast	0.661	0.896	0.961
Hue Cold	0.920	0.944	0.947
Horizontal	0.889	0.985	0.996
Directional	0.380	0.651	0.771
Manually Marked	0.971	0.989	0.994
Metal texture			
High Contrast	0.909	0.964	0.980
Stone texture			
Hue Cold	0.773	0.827	0.845
Face image			
Gray hair	0.648	0.862	0.929
Double chin & High cheekbones	0.228	0.466	0.647
Eyeglasses & No Mustache	0.947	0.982	0.987
Wearing hat	0.837	0.939	0.968
Chubby & Oval face	0.210	0.366	0.450
Goatee	0.426	0.776	0.880
Bald	0.735	0.920	0.962
Eyeglasses & female	0.927	0.968	0.981

Table 1: *Summary of the classification accuracy for each of the selection criteria.*

	Average Verification TAR					
FAR	0.001	0.01	0.02	0.05	0.1	0.2
Random	0.152	0.413	0.517	0.652	0.738	0.818
Our	0.296	0.620	0.696	0.785	0.851	0.903
All data	0.343	0.686	0.775	0.883	0.939	0.979

Table 2: *A comparison of average TAR over different face selection tasks for different labeling strategies: labeling* 600 *random selected candidates, labeling* 600 *candidates selected with our interactive system, and using* all *reference labels over the whole dataset.*

lieu of a human user. The above selection criteria have been intentionally designed to cover a wide variety in ratio and distribution of positive samples over the input dataset.

Table 1 lists the accuracy of our system on the above cases using only 600 (i.e., 30 rounds of 20) labeled exemplars. Typically the accuracy of a classifier is quantified by positive and negative precision and recall. However, precision and recall fail to provide a nuanced picture when the ratio between positive and negative samples is unbalanced. Therefore, as in prior work on face recognition [MaTaH*16, YRZ*17], we will use the *true accept rate (TAR)* for a fixed *false positive rate (FAR)* as a quality metric. In the majority of cases, our method is able to produce a classifier with high accuracy. The cause for the lower accuracy in *"directional"*, *"double chin & high cheekbones"*, *"chubby & oval face"*, and *"Goatee"*) is related to the number of positive exemplars in the training set versus the complexity of the selection criteria (i.e., insufficient training samples that meet the selection criteria).

To gain a better understanding on the general performance, we compare the accuracy, averaged over all *face* selection cases, of a labeling on 600, randomly selected candidates, a labeling over 600 adaptively selected candidates, and a labeling trained over all the labels (Table 2). The first serves as a baseline, whereas the last is

	Average Verification TAR		
	FAR 0.01	FAR 0.05	FAR 0.1
Random	0.600	0.786	0.852
QBC	0.678	0.804	0.856
QBC + UL	0.742	0.854	0.893
QBC + DD	0.765	0.868	0.910
QBC + UL + DD	**0.812**	**0.915**	**0.946**
Our + Parallel	0.786	0.894	0.928

Table 3: *Ablation study on wood texture curation by enabling/disabling various combinations of: query-by-committee (QBC), allowing an "undecided" label (UL), using the disagreement distance (DD), and using parallel candidate selection and labeling (Parallel) to improve performance.*

the theoretical upperbound on what is possible with the training data. The random selection fails to produce accurate results, while the accuracy of our interactive adaptive method is closer to the upperbound. We refer to the supplemental material for a comparison of the accuracy for each selection criterion.

5.2. Ablation Study

We perform an ablation study on the texture selection criteria to gauge the effect of each of the components of our system. For each test, we keep the total number of candidates the oracle has to label fixed at 600, and select 20 candidates per round. As a baseline we let the oracle label randomly selected candidates. We compare the average baseline TAR scores (for different fixed FAR thresholds) over all the test cases to our system's scores when selecting candidates with a Query-by-Committee strategy with/without using the "undecided" label, and with/without the disagreement distance metric for promoting diversity. For the ablation case without the "undecided" label, we binarize the label process by using the middle (average) threshold for determining the positive and negative label (e.g., if contrast is larger than $8.75 = (9.5 + 8)/2$ then we label it as *binary high contrast*, if less than 8.75 we label it as negative). In the case without the disagreement metric we simply use the top 20 exemplars with the largest disagreement. Table 3 shows that, on average, naive Query-by-Committee already outperforms random sampling. Each of the components further improves accuracy, and when all work together, an additional improvement is observed. Note that we exclude the manually marked wood selection (Figure 8) as we found that the baseline failed to find a sufficient number of positive exemplars to train a suitable classifier. This is an example of a case, where, unless the user labels a massive amount of textures, our interactive system is necessary to create a GAN that matches the user's target criteria.

In our current system we parallelize the labeling process with the training and selection process. As a consequence, selection is made on a classifier that does not use the latest round of labeled exemplars. The results in Table 3 are computed without this parallelization. To better understand the potential negative impact of parallelization, we also include the scores for our full system with parallelization. On average, this parallelization has a modest impact on the accuracy of the final results, while simultaneously yielding a significant performance improvement. We refer to the supplemental

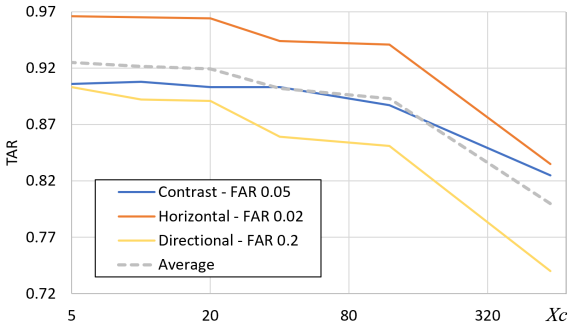


Figure 9: *Impact of label batch size (X_c) on the accuracy of the classifier.*

material for a listing of the ablation study results for each selection criterion.

5.3. Impact of batch size X_c

A key parameter in our system is the size of the batch of candidates shown to the user per round. In theory, a smaller batch of candidates means that the selection process can more quickly benefit from the assigned labels. However, decreasing the batch size also negatively affects the interactivity (i.e., high cost of training the classifier).

To better understand the impact on the accuracy (ignoring computational costs), we plot the batch size versus the accuracy in Figure 9 over three selected criteria for wood textures. From this we can see that the accuracy increases with smaller batch size (as expected). However, we also note that the accuracy gain is limited when the batch size is less than 20. We find that the performance scales linearly and inversely proportional with batch size (e.g., $X_c = 5$ takes 4 times as long as $X_c = 20$). Therefore, we recommend $X_c = 20$ as it strikes a good balance between computation time and accuracy; we have used $X_c = 20$ for all results (unless indicated otherwise) in this paper.

5.4. Impact of Number of Rounds/Labeled Candidates

Table 4 further compares the accuracy for an increasing number of rounds, and thus total number of labeled candidates. While increasing the number of rounds/candidates increases the accuracy, it also comes at an increased labeling cost for the user. For the examples in this paper, we found that 30 rounds offers a good balance between labeling effort versus accuracy. However, more complex selection criteria might require more rounds to achieve the desired accuracy.

5.5. Impact of Number of Classifiers

Another key parameter is the size of the committee for QBC. Whereas a larger committee will require more computational resources to train the classifiers, a smaller committee can incur a performance drop due to lack of variation. Table 5 lists the accuracy for a committee size of 2, 4, and 8 using the same setup as in Table 2 and 4. For the minimum committee size (i.e., 2), a notable drop in performance is observed. In contrast, the accuracy only slightly increases for 8 classifiers at a significant increase in computational cost. Therefore, we opt for using a committee of 4 classifiers which maintains interactivity while retaining good accuracy.

Rounds	Total Cand.	Average Verification TAR FAR 0.01	FAR 0.05	FAR 0.1
30	600	0.620	0.785	0.851
60	1200	0.619	0.805	0.864
120	2400	0.630	0.821	0.880
240	4800	0.641	0.838	0.907
All Data (180K)		0.686	0.883	0.939

Table 4: *A comparison of average TAR over different face selection tasks for varying numbers of rounds (*30*,* 60*,* 120*, and* 240 *rounds) of* 20 *adaptively selected candidates labeled with our interactive system.*

	Average Verification TAR					
FAR	0.001	0.01	0.02	0.05	0.1	0.2
2 classifiers	0.261	0.591	0.683	0.776	0.840	0.900
4 classifiers	0.296	0.620	0.696	0.785	0.851	0.903
8 classifiers	0.312	0.625	0.698	0.796	0.860	0.919

Table 5: *Impact of the committee size (i.e., number of classifiers) for QBC on the average TAR over different face selection tasks.* 2 *classifiers incur a significant accuracy drop, whereas* 8 *classifiers require significantly more computational resources for moderate gains in accuracy.*

5.6. Performance

Our dataset curation framework runs at interactive rates on a workstation with 4 NVidia TitanX GPUs. The computational cost (per round) depends on the curation task:

- For most **texture** curation tasks, candidate selection takes 21 seconds, and classifier training takes 45 seconds. The computational cost of the selection phase increases for tasks with low target distribution density (due to the adaptive selection): *wood with cold hue* and the manually created *red wood with detailed grain* take 41 and 206 seconds respectively.
- The **face** curation tasks all have a similar running time: candidate selection takes 10 seconds, classifier training takes only 3 seconds, and feature map pre-computation takes an additional 6 seconds. The *face* curation tasks run faster since they operate in a pre-optimized feature space, which only needs to be evaluated once per sample (i.e., it can be pre-computed).

Note that with our parallel candidate selection, the majority of the computation is performed while the user is labeling candidates; typically labeling takes longer than the required computation time. Furthermore, we only label significantly less exemplars than the full dataset while achieving good results. E.g., for the 180K image CelebA dataset, it is practically impossible to label all exemplars, while we only label less than 1%.

6. Conclusions

This paper presents a novel interactive system for curating datasets for training or refining generative models. Our method builds on batch-based active learning and extends query-by-consensus to correctly handle uncertain oracles (i.e., users). Furthermore, we propose a novel disagreement metric to promote diversity in the sampled batches for the user to label. In each round, a small diverse set of exemplars is presented to the user to label, and the results are used to refine a(n ensemble of) classifier(s), which in turn guide the selection process of candidate exemplars. We demonstrated the flexibility of our framework on four applications: creating GANs that meet a user-defined criteria, cleaning up training data, modifying the distribution of a generative model, and removing unwanted samples from a GAN.

For future work, we would like to explore other operations to modify or refine a generative model as well as investigate novel acceleration methods to further enhance the interactivity of our system.

Acknowledgments We thank the reviewers for their insightful feedback. Pieter Peers was partially supported by NSF grant IIS-1350323 and a gift from Nvidia.

References

[AAB*15] ABADI M., AGARWAL A., BARHAM P., BREVDO E., CHEN Z., CITRO C., CORRADO G. S., DAVIS A., DEAN J., DEVIN M., GHEMAWAT S., GOODFELLOW I., HARP A., IRVING G., ISARD M., JIA Y., JOZEFOWICZ R., KAISER L., KUDLUR M., LEVENBERG J., MANÉ D., MONGA R., MOORE S., MURRAY D., OLAH C., SCHUSTER M., SHLENS J., STEINER B., SUTSKEVER I., TALWAR K., TUCKER P., VANHOUCKE V., VASUDEVAN V., VIÉGAS F., VINYALS O., WARDEN P., WATTENBERG M., WICKE M., YU Y., ZHENG X.: TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL: https://www.tensorflow.org/. 6

[ACB17] ARJOVSKY M., CHINTALA S., BOTTOU L.: Wasserstein generative adversarial networks. In *ICML* (2017), pp. 214–223. 2

[Ado19] ADOBE: Adobe stock images, 2019. URL: https://stock.adobe.com/. 6

[BGNK18] BELUCH W. H., GENEWEIN T., NÃIJRNBERGER A., KÃŰHLER J. M.: The power of ensembles for active learning in image classification. In *CVPR* (2018). 3

[BJV17] BERGMANN U., JETCHEV N., VOLLGRAF R.: Learning texture manifolds with the periodic spatial GAN. In *ICML* (2017), vol. 70, pp. 469–477. 3

[FAW19] FRÜHSTÜCK A., ALHASHIM I., WONKA P.: TileGAN: Synthesis of large-scale non-homogeneous textures. *ACM Trans. Graph. 38*, 4 (2019). 3

[FS89] FOGEL I., SAGI D.: Gabor filters as texture discriminator. *Biological Cybernetics 61*, 2 (Jun 1989). 7

[GAA*17] GULRAJANI I., AHMED F., ARJOVSKY M., DUMOULIN V., COURVILLE A. C.: Improved training of wasserstein gans. In *NIPS* (2017), pp. 5767–5777. 6

[Gab47] GABOR D.: Theory of communication. *Journal of the Institution of Electrical Engineers - Part I: General 94*, 73 (January 1947), 58–. 7

[GDG*17] GUÉRIN E., DIGNE J., GALIN E., PEYTAVIE A., WOLF C., BENES B., MARTINEZ B.: Interactive example-based terrain authoring with conditional generative adversarial networks. *ACM Trans. Graph. 36*, 6 (Nov. 2017). 3

[GLLD12] GALERNE B., LAGAE A., LEFEBVRE S., DRETTAKIS G.: Gabor noise by example. *ACM Trans. Graph. 31*, 4 (July 2012), 73:1–73:9. 3

[GPAM*14] GOODFELLOW I. J., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A., BENGIO Y.: Generative Adversarial Nets. In *NIPS* (2014). 2

[HS90] HANSEN L. K., SALAMON P.: Neural network ensembles. *PAMI 12*, 10 (1990), 993–1001. 6

[JBS17] JETCHEV N., BERGMANN U., SEWARD C.: Ganosaic: Mosaic creation with generative texture manifolds. In *NIPS workshop on Machine Learning for Creativity and Design* (2017). 3

[KB15] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. In *ICLR* (May 2015). 6

[KGS*18] KELLY T., GUERRERO P., STEED A., WONKA P., MITRA N. J.: Frankengan: Guided detail synthesis for building mass models using style-synchonized gans. *ACM Trans. Graph. 37*, 6 (2018). 3

[KL51] KULLBACK S., LEIBLER R. A.: On information and sufficiency. *Ann. Math. Statist. 22*, 1 (03 1951), 79–86. 4

[KLA19] KARRAS T., LAINE S., AILA T.: A style-based generator architecture for generative adversarial networks. In *CVPR* (2019). 6

[LLWT15] LIU Z., LUO P., WANG X., TANG X.: Deep learning face attributes in the wild. In *ICCV* (2015). 6

[LVLD10] LAGAE A., VANGORP P., LENAERTS T., DUTRÉ P.: Procedural isotropic stochastic textures by example. *Comput. Graph. 34*, 4 (Aug. 2010), 312–321. 3

[Ma10] MA Y.: Mayang's free texture library, 2010. URL: http://mayang.com/textures/. 6

[MaTaH*16] MASI I., AN TRÃN A. T., HASSNER T., LEKSUT J. T., MEDIONI G.: Do we really need to collect millions of faces for effective face recognition? In *ECCV* (2016). 9

[MN98] MCCALLUM A., NIGAM K.: Employing em and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning* (1998), ICML '98, pp. 350–358. 4

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. *ACM Trans. Graph. 22*, 3 (July 2003), 313–318. 2

[PHS*18] PORTENIER T., HU Q., SZABÓ A., BIGDELI S. A., FAVARO P., ZWICKER M.: Faceshop: Deep sketch-based face image editing. *ACM Trans. Graph. 37*, 4 (July 2018), 99:1–99:13. 3

[RDDM17] RAAD L., DAVY A., DESOLNEUX A., MOREL J.: A survey of exemplar-based texture synthesis. *CoRR abs/1707.07184* (2017). 2

[RMC15] RADFORD A., METZ L., CHINTALA S.: Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv* (2015). 2

[SED19] SINHA S., EBRAHIMI S., DARRELL T.: Variational adversarial active learning. *CoRR abs/1904.00370* (2019). 3

[Set10] SETTLES B.: *Active learning literature survey*. Tech. rep., 2010. 2, 3

[SKP15] SCHROFF F., KALENICHENKO D., PHILBIN J.: Facenet: A unified embedding for face recognition and clustering. In *CVPR* (2015), pp. 815–823. 5

[SOS92] SEUNG H. S., OPPER M., SOMPOLINSKY H.: Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (1992), pp. 287–294. 2, 3, 4

[SS18] SENER O., SAVARESE S.: Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations* (2018). 3

[SZ15] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. In *ICLR* (2015). 5

[WZL*17] WANG K., ZHANG D., LI Y., ZHANG R., LIN L.: Cost-effective active learning for deep image classification. *IEEE Trans. Cir. and Sys. for Video Technol. 27*, 12 (Dec. 2017), 2591–2600. 3

[YRZ*17] YANG J., REN P., ZHANG D., CHEN D., WEN F., LI H., HUA G.: Neural aggregation network for video face recognition. In *CVPR* (2017), pp. 4362–4371. 9

[YZS*15] YU F., ZHANG Y., SONG S., SEFF A., XIAO J.: Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR abs/1506.03365* (2015). 6

[ZB17] ZHU J., BENTO J.: Generative adversarial active learning. *CoRR abs/1702.07956* (2017). 3

[ZKSE16] ZHU J.-Y., KRÄHENBÜHL P., SHECHTMAN E., EFROS A. A.: Generative visual manipulation on the natural image manifold. In *ECCV* (2016). 3

[ZZB*18] ZHOU Y., ZHU Z., BAI X., LISCHINSKI D., COHEN-OR D., HUANG H.: Non-stationary texture synthesis by adversarial expansion. *ACM Trans. Graph. 37*, 4 (2018), 49:1–49:13. 3